



1

2

3

4

Document Number: DSP1002

Date: 2007-05-10

Version: 2.0.0a

5 **Diagnostics Profile**

6 **Document Type: Specification**

7 **Document Status: Preliminary**

8 **Document Language: E**

9 Copyright Notice

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

11 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems
12 management and interoperability. Members and non-members may reproduce DMTF specifications and
13 documents for uses consistent with this purpose, provided that correct attribution is given. As DMTF
14 specifications may be revised from time to time, the particular version and release date should always be
15 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26 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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	1	Scope	9	
35	2	Normative References	9	
36	2.1	Approved References	9	
37	2.2	References under Development	9	
38	2.3	Other References	9	
39	3	Terms and Definitions	9	
40	4	Symbols and Abbreviated Terms.....	11	
41	5	Synopsis.....	12	
42	6	Description.....	12	
43	7	Implementation	13	
44	7.1	CIM_DiagnosticTest.....	13	Dele'
45	7.2	CIM_AvailableDiagnosticService.....	16	
46	7.3	CIM_DiagnosticServiceCapabilities.....	16	Dele'
47	7.4	CIM_DiagnosticSettingData	16	Dele'
48	7.5	CIM_ConcreteJob	17	
49	7.6	CIM_DiagnosticLog.....	17	
50	7.7	CIM_DiagnosticRecord	18	
51	7.8	CIM_ServiceComponent	18	
52	8	Methods.....	19	
53	8.1	CIM_DiagnosticService.RunDiagnosticService() Extrinsic Method.....	19	
54	8.2	CIM_ConcreteJob.RequestStateChange() Extrinsic Method.....	20	
55	8.3	CIM_Log.ClearLog() Extrinsic Method.....	21	
56	8.4	CIM_HelpService.GetHelp() Extrinsic Method	21	
57	8.5	Profile Conventions for Operations.....	22	
58	8.6	CIM_DiagnosticTest.....	23	
59	8.7	CIM_AvailableDiagnosticService.....	23	
60	8.8	CIM_ServiceAffectsElement.....	23	
61	8.9	CIM_SoftwareIdentity.....	24	
62	8.10	CIM_ElementSoftwareIdentity	24	
63	8.11	CIM_HelpService.....	24	
64	8.12	CIM_ServiceAvailableToElement	25	
65	8.13	CIM_DiagnosticSettingData	25	
66	8.14	CIM_DiagnosticServiceCapabilities.....	25	
67	8.15	CIM_ElementCapabilities	26	
68	8.16	CIM_ConcreteJob.....	26	
69	8.17	CIM_OwningJobElement	26	
70	8.18	CIM_AffectedJobElement.....	27	
71	8.19	CIM_JobSettingData.....	27	
72	8.20	CIM_ElementSettingData.....	27	
73	8.21	CIM_DiagnosticLog.....	28	
74	8.22	CIM_UseOfLog.....	28	
75	8.23	CIM_DiagnosticServiceRecord.....	28	
76	8.24	CIM_DiagnosticCompletionRecord.....	29	
77	8.25	CIM_DiagnosticSettingDataRecord	29	
78	8.26	CIM_LogManagesRecord	30	
79	8.27	CIM_RecordAppliesToElement	30	
80	8.28	CIM_CorrespondingSettingDataRecord	30	
81	8.29	CIM_ServiceComponent	30	
82	9	Use Cases.....	31	
83	9.1	Profile Conformance	31	
84	9.2	Use Case Summary.....	32	
85	9.3	Diagnostic Services Object Diagram	34	

Diagnostics Profile

86	9.4	Discover Available Diagnostics	35
87	9.5	Configure Diagnostic.....	36
88	9.6	Execute and Control Diagnostic	38
89	9.7	Discover Diagnostic Executions	41
90	9.8	Discover Diagnostic Results (In Progress and Final)	43
91	10	CIM Elements	46
92	10.1	CIM_AffectedJobElement.....	48
93	10.2	CIM_AvailableDiagnosticService.....	48
94	10.3	CIM_ConcreteJob	48
95	10.4	CIM_CorrespondingSettingDataRecord	49
96	10.5	CIM_DiagnosticCompletionRecord.....	49
97	10.6	CIM_DiagnosticLog.....	50
98	10.7	CIM_DiagnosticServiceCapabilities	51
99	10.8	CIM_DiagnosticServiceRecord.....	52
100	10.9	CIM_DiagnosticSettingData	53
101	10.10	CIM_DiagnosticSettingDataRecord	54
102	10.11	CIM_DiagnosticTest.....	55
103	10.12	CIM_ElementCapabilities	55
104	10.13	CIM_ElementSettingData.....	56
105	10.14	CIM_ElementSoftwareIdentity	56
106	10.15	CIM_HelpService.....	56
107	10.16	CIM_HostedService	57
108	10.17	CIM_JobSettingData	57
109	10.18	CIM_LogManagesRecord	58
110	10.19	CIM_OwningJobElement	58
111	10.20	CIM_RecordAppliesToElement	58
112	10.21	CIM_RegisteredProfile	59
113	10.22	CIM_ServiceAffectsElement.....	59
114	10.23	CIM_ServiceAvailableToElement	59
115	10.24	CIM_ServiceComponent	60
116	10.25	CIM_SoftwareIdentity.....	60
117	10.26	CIM_UseOfLog.....	60
118	ANNEX A (informative)	Change Log	61
119	ANNEX B (informative)	Acknowledgements.....	62
120			

121 Figures

122	Figure 1 – Diagnostics Profile: Class Diagram	13
123	Figure 2 – Registered Profile	32
124	Figure 3 – Diagnostic Services Object Diagram	34
125	Figure 4 – Job Example.....	39
126	Figure 5 – Diagnostic Logging Object Diagram	43
127		

128 Tables

129	Table 1 – Related Profiles	12
130	Table 2 – RunDiagnosticService() Method: Return Code Values	19
131	Table 3 – RunDiagnosticService() Method: Parameters	20
132	Table 4 – RequestStateChange() Method: Return Code Values	20
133	Table 5 – RequestStateChange() Method: Parameters	21
134	Table 6 – ClearLog() Method: Return Code Values	21
135	Table 7 – GetHelp() Method: Return Code Values.....	22
136	Table 8 – GetHelp() Method: Parameters.....	22
137	Table 9 – Operations: CIM_DiagnosticTest.....	23
138	Table 10 – Operations: CIM_AvailableDiagnosticService	23
139	Table 11 – Operations: CIM_ServiceAffectsElement.....	23
140	Table 12 – Operations: CIM_SoftwareIdentity.....	24
141	Table 13 – Operations: CIM_ElementSoftwareIdentity	24
142	Table 14 – Operations: CIM_HelpService	24
143	Table 15 – Operations: CIM_ServiceAvailableToElement	25
144	Table 16 – Operations: CIM_DiagnosticSettingData	25
145	Table 17 – Operations: CIM_DiagnosticServiceCapabilities	25
146	Table 18 – Operations: CIM_ElementCapabilities	26
147	Table 19 – Operations: CIM_ConcreteJob	26
148	Table 20 – Operations: CIM_OwningJobElement.....	26
149	Table 21 – Operations: CIM_AffectedJobElement.....	27
150	Table 22 – Operations: CIM_JobSettingData	27
151	Table 23 – Operations: CIM_ElementSettingData.....	27
152	Table 24 – Operations: CIM_DiagnosticLog.....	28
153	Table 25 – Operations: CIM_UseOfLog	28
154	Table 26 – Operations: CIM_DiagnosticServiceRecord.....	28
155	Table 27 – Operations: CIM_DiagnosticCompletionRecord.....	29
156	Table 28 – Operations: CIM_DiagnosticSettingDataRecord	29
157	Table 29 – Operations: CIM_LogManagesRecord.....	30
158	Table 30 – Operations: CIM_RecordAppliesToElement	30
159	Table 31 – Operations: CIM_CorrespondingSettingDataRecord.....	30
160	Table 32 – Operations: CIM_ServiceComponent	30
161	Table 33 – Diagnostics Profile Use Cases	32
162	Table 34 – CIM Elements: <i>Diagnostics Profile</i>	46
163	Table 35 – Class: CIM_AffectedJobElement.....	48
164	Table 36 – Class: CIM_AvailableDiagnosticService	48
165	Table 37 – Class: CIM_ConcreteJob	48
166	Table 38 – Class: CIM_CorrespondingSettingDataRecord	49

Diagnostics Profile

167	Table 39 – Class: CIM_DiagnosticCompletionRecord	49
168	Table 40 – Class: CIM_DiagnosticLog	50
169	Table 41 – Class: CIM_DiagnosticServiceCapabilities	51
170	Table 42 – Class: CIM_DiagnosticServiceRecord	52
171	Table 43 – Class: CIM_DiagnosticSettingData	53
172	Table 44 – Class: CIM_DiagnosticSettingDataRecord	54
173	Table 45 – Class: CIM_DiagnosticTest	55
174	Table 46 – Class: CIM_ElementCapabilities	55
175	Table 47 – Class: CIM_ElementSettingData	56
176	Table 48 – Class: CIM_ElementSoftwareIdentity	56
177	Table 49 – Class: CIM_HelpService	56
178	Table 50 – Class: CIM_HostedService	57
179	Table 51 – Class: CIM_JobSettingData	57
180	Table 52 – Class: CIM_LogManagesRecord	58
181	Table 53 – Class: CIM_OwningJobElement	58
182	Table 54 – Class: CIM_RecordAppliesToElement	58
183	Table 55 – Class: CIM_RegisteredProfile	59
184	Table 56 – Class: CIM_ServiceAffectsElement	59
185	Table 57 – Class: CIM_ServiceAvailableToElement	59
186	Table 58 – Class: CIM_ServiceComponent	60
187	Table 59 – Class: CIM_SoftwareIdentity	60
188	Table 60 – Class: CIM_UseOfLog	60
189		

190

Foreword

191 The *Diagnostics Profile* (DSP1002) was prepared by the Core Schema Working Group.

192 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems
193 management and interoperability.

Introduction

195 A *profile* is a collection of Common Information Model (CIM) elements and behavior rules that represent a
196 specific area of management. The purpose of a profile is to ensure interoperability in the use of web-
197 based enterprise management (WBEM) services for a specific subset of the Distributed Management
198 Task Force (DMTF) CIM schema for a specific management area—in this case, diagnostics.

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

205 The CDM is an architecture and methodology for exposing system diagnostic instrumentation through the
206 CIM standard interfaces. IBM, Intel, and PC-Doctor, Inc., introduced the CDM at the DMTF annual
207 conference in June 1999. Since then, the proposed extensions required to support diagnostics have been
208 accepted by the DMTF and included in version 2.3 of the CIM schema.

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

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

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

222 Reduced cost through standardization is accompanied by the initial investment of coding to a new
223 interface. The CDM Forum intends to ease this burden by developing tools to generate most of the
224 interface code necessary to communicate with CIM.

225 Since its introduction, the CDM has been promoted at various industry events including the Intel
226 Developer Forums, DMTF Annual Conferences, and Microsoft WinHEC. It has been met with strong
227 support from the technical community and is quickly becoming the de facto standard for developing OS-
228 Present Diagnostic tools. Major OEMs are developing service tools that rely on the CDM and will require
229 their vendors to deliver CDM-compliant diagnostic tests with their products.

230

Diagnostics Profile

231 1 Scope

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

236 The target audience for this specification is implementers who are writing CIM-based providers or
237 consumers of management interfaces that represent the component described in this document.

238 2 Normative References

239 The following referenced documents are indispensable for the application of this document. For dated
240 references, only the edition cited applies. For undated references, the latest edition of the referenced
241 document (including any amendments) applies.

242 2.1 Approved References

243 DMTF DSP0200, *CIM Operations over HTTP 1.2.0*

244 DMTF DSP0004, *CIM Infrastructure Specification 2.3.0*

245 DMTF DSP1000, *Management Profile Specification Template*

246 DMTF DSP1001, *Management Profile Specification Usage Guide*

247 DMTF DSP2000, *CIM Diagnostic Model White Paper*

248 2.2 References under Development

249 DMTF DSP1033, *Profile Registration Profile*

250 DMTF DSP1004, *Base Server Profile*

251 2.3 Other References

252 ISO/IEC Directives, Part 2, [Rules for the structure and drafting of International Standards](#)

253 [Unified Modeling Language \(UML\) from the Open Management Group \(OMG\)](#)

254 DMTF DSP0215, *SM Managed Element Addressing Specification (SM ME Addressing)*

255 3 Terms and Definitions

256 For the purposes of this document, the following terms and definitions apply. The terms and definitions
257 given in DSP1033 and DSP1001 also apply.

Diagnostics Profile

- 258 **3.1**
259 **can**
260 used for statements of possibility and capability, whether material, physical, or causal
- 261 **3.2**
262 **cannot**
263 used for statements of possibility and capability, whether material, physical, or causal
- 264 **3.3**
265 **conditional**
266 indicates requirements to be followed strictly in order to conform to the document when the specified
267 conditions are met
- 268 **3.4**
269 **mandatory**
270 indicates requirements to be followed strictly in order to conform to the document and from which no
271 deviation is permitted
- 272 **3.5**
273 **may**
274 indicates a course of action permissible within the limits of the document
- 275 **3.6**
276 **need not**
277 indicates a course of action permissible within the limits of the document
- 278 **3.7**
279 **optional**
280 indicates a course of action permissible within the limits of the document
- 281 **3.8**
282 **referencing profile**
283 indicates a profile that owns the definition of this class and can include a reference to this profile in its
284 “Related Profiles” table
- 285 **3.9**
286 **shall**
287 indicates requirements to be followed strictly in order to conform to the document and from which no
288 deviation is permitted
- 289 **3.10**
290 **shall not**
291 indicates requirements to be followed strictly in order to conform to the document and from which no
292 deviation is permitted
- 293 **3.11**
294 **should**
295 indicates that among several possibilities, one is recommended as particularly suitable, without
296 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 297 **3.12**
298 **should not**
299 indicates that a certain possibility or course of action is deprecated but not prohibited

300 **3.13**
301 **unspecified**
302 indicates that this profile does not define any constraints for the referenced CIM element or operation

303 **4 Symbols and Abbreviated Terms**

304 The following abbreviations are used in this document.

305 **4.1**

306 **CDM**

307 Common Diagnostic Model

308 **4.2**

309 **CIM**

310 Common Information Model

311 **4.3**

312 **CIMOM**

313 CIM Object Manager

314 **4.4**

315 **CRU**

316 Customer Replaceable Unit

317 **4.5**

318 **FRU**

319 Field Replaceable Unit

320 **4.6**

321 **ME**

322 Managed Element

323 **4.7**

324 **MOF**

325 Managed Object Format

326 **4.8**

327 **PD**

328 Problem Determination

329 **4.9**

330 **PFA**

331 Predictive Failure Analysis

332 **4.10**

333 **SAN**

334 Storage Area Network

335 **4.11**

336 **WBEM**

337 Web-Based Enterprise Management

338 5 Synopsis

339 **Profile Name:** *Diagnostics Profile*

340 **Version:** 1.0.0a

341 **Organization:** DMTF – Core Schema Working Group, Diagnostics SIG

342 **CIM schema version:** [2.11](#)

343 **Central Class:** CIM_DiagnosticTest

344 **Scoping Class:** CIM_ComputerSystem

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

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

350 CIM_DiagnosticTest shall be the Central Class of this profile. The instance of CIM_DiagnosticTest shall
351 be the Central Instance of this profile. CIM_ComputerSystem shall be the Scoping Class of this profile.
352 The instance of CIM_ComputerSystem with which the Central Instance is associated through an instance
353 of CIM_HostedService shall be the Scoping Instance of this profile.

354

Table 1 – Related Profiles

Profile Name	Organization	Version	Relationship	Behavior
Profile Registration Profile	DMTF	1.0	Mandatory	

355 6 Description

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

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

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

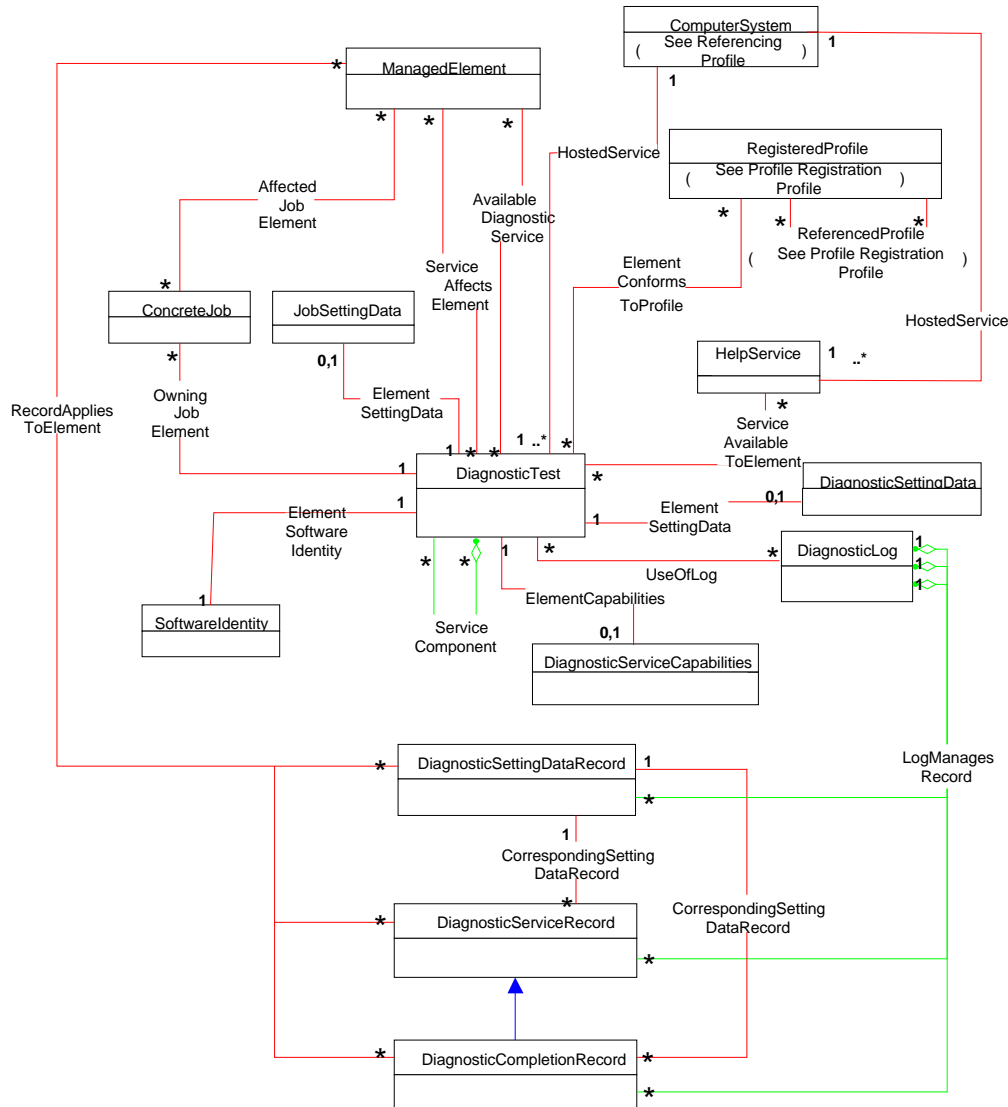


Figure 1 – Diagnostics Profile: Class Diagram

369

370

371 7 Implementation

372 This section details the requirements related to the arrangement of instances and their properties for
 373 implementations of this profile.

374 The *Diagnostics Profile* consists of definitions for classes related to the CIM_DiagnosticService class,
 375 such as CIM_DiagnosticTest, CIM_DiagnosticSettingData, and CIM_DiagnosticServiceCapabilities. It
 376 also defines the CIM_DiagnosticLog class and its related classes, CIM_DiagnosticRecord,
 377 CIM_DiagnosticServiceRecord, and CIM_DiagnosticSettingDataRecord. Requirements for propagating
 378 and formulating certain properties of these classes and their parents are discussed in this section.
 379 Required methods are listed in section 8, and properties are listed in section 10.

380 7.1 CIM_DiagnosticTest

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

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

386 Depending on the implementation, a provider may use

- 387 • an instance of CIM_DiagnosticTest for each test
- 388 • an instance of a single subclass (for example, ST_DiskDiagnosticTest) for each test
- 389 • a different subclass and its instance (for example, ST_DiskDiagnosticSelfTest,
390 ST_DiskDiagnosticRWVTest) for each test

391 The same provider may use a combination of the preceding approaches.

392 **7.1.1 CIM_DiagnosticTest.Name**

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

396 **7.1.2 CIM_DiagnosticTest.ElementName**

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

399 **7.1.3 CIM_DiagnosticTest.Characteristics**

400 This section defines the values of the Characteristics property.

401 **7.1.3.1 Is Exclusive (value=2)**

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

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

406 **7.1.3.2 Is Interactive (value=3)**

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

409 **7.1.3.3 Is Destructive (value=4)**

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

412 **7.1.3.4 Is Risky (value=5)**

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

417 **7.1.3.5 Is Package (value=6)**

418 Use this value to indicate that the test is actually a set of lower-level diagnostics that are packaged
419 together by the test. This packaging is implemented by the test, not aggregated by CIM. Information and

420 results associated with the individual tests in the package may be requested by using the Subtests value
421 in the CIM_DiagnosticSettingData.LogOptions array.

422 If the lower-level diagnostics are themselves CIM_DiagnosticTest instances, the packaging test shall be
423 associated to those lower-level diagnostics through an instance of the CIM_ServiceComponent
424 association. See section 7.8.

425 **7.1.3.6 Reserved (value=7)**

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

428 **7.1.3.7 Is Synchronous (value=8)**

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

434 **7.1.3.8 Media Required (value=9)**

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

436 **7.1.3.9 Additional Hardware Required (value=10)**

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

439 **7.1.4 Looping Tests**

440 Looping tests or groups of tests is useful for detecting intermittent faults. The client, provider, or test may
441 control looping, and the method chosen depends on many factors, a few of which follow:

- 442 • A client may want to loop a test that does not support looping.
- 443 • A provider may choose to support looping even though its tests do not.
- 444 • A stress test may, by its nature, want to repeat a certain operation a large number of times.

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

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

451 **Note:** A remote client may incur network delays and CIMOM delays during every iteration of its loop, and
452 this is not an effective way to stress a device.

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

456 **7.1.5 Test Effectiveness**

457 Although the focus of this profile is use of the CIM schema, the CDM includes the notion of test
458 effectiveness. A perfectly implemented CDM provider wrapped around an ineffective test is not very
459 useful.

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

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

465 **Note:** QuickMode is distinct from PercentOfTestCoverage in that it is a Boolean property that may be set
466 by a client without any particular knowledge of the test. Use of PercentOfTestCoverage requires that the
467 client be aware of the effects and expected outcome of this “throttling” setting control. Diagnostic tests
468 should support the ability to surface logs that may be useful in the problem-determination process.

469 **7.2 CIM_AvailableDiagnosticService**

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

473 **7.2.1 CIM_AvailableDiagnosticService.EstimatedDurationOfService**

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

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

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

487 **7.2.2 CIM_AvailableDiagnosticService.EstimatedDurationQualifier**

488 The EstimatedDurationQualifier property allows for more accurate quantification of the value specified for
489 the EstimatedDurationOfService property. This property shall be implemented only if further quantification
490 is possible.

491 **7.3 CIM_DiagnosticServiceCapabilities**

492 CIM_DiagnosticServiceCapabilities is the means by which a diagnostic service may publish its support for
493 various options—in particular, settings. If a setting is supported, the client may assign it, usually in
494 satisfaction of a user request. The client gains access to an instance of
495 CIM_DiagnosticServiceCapabilities through an instance of CIM_ElementCapabilities.

496 **7.4 CIM_DiagnosticSettingData**

497 This class defines specific diagnostic service parameters and execution instructions. To provide more
498 detailed settings for a type of test (that is, additional properties), subclassing is appropriate.

499 The default settings for a diagnostic service are obtained by using the CIM_ElementSettingData
500 association to an instance of (a subclass of) CIM_DiagnosticSettingData. If a service does not publish
501 defaults in this manner, the client should either avoid settings altogether or use only those settings
502 supported by an instance of CIM_DiagnosticServiceCapabilities.

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

505 If a client chooses to accept the default settings (published or not), the CIM_DiagnosticSettingData object
506 may be excluded from the method parameter list (entered as NULL).

507 **7.5 CIM_ConcreteJob**

508 This section defines the properties of the CIM_ConcreteJob class. All executing diagnostics will be
509 represented by instances of CIM_ConcreteJob so that a client can track the progress and control the
510 execution of the executing diagnostic.

511 **7.5.1 CIM_ConcreteJob.TimeBeforeRemoval**

512 To properly implement the functionality implied by this property, the job completion time shall be
513 determined. The algorithm is

514 If JobState=Completed OR Terminated OR Killed, then Completion Time=StartTime+ElapsedTime.

515 The job may be deleted at Completion Time+TimeBeforeRemoval.

516 **7.5.2 CIM_ConcreteJob.PercentComplete**

517 This property indicates the percentage of the job that has completed at the time that this value is
518 requested.

519 Implementation of this property is mandatory in order to provide progress indication to clients.

520 The value of this property shall be kept current to be useful. Service providers should update this property
521 within one second of becoming aware of a progress change.

522 The PercentComplete property shall always report the actual percent complete of how much testing was
523 done. It shall be set to 100 percent only when the test is complete. It shall not be set to 100 percent if the
524 test stops for any other reason (for example, the test stopped or was killed by user, the test exited due to
525 a critical failure, or the test found an error and HaltOnError is TRUE) because the actual percent complete
526 is not 100 percent.

527 **7.6 CIM_DiagnosticLog**

528 All diagnostic result messages shall be represented by instances of CIM_DiagnosticRecord subclasses.
529 Moreover, those records shall be aggregated to an instance of CIM_DiagnosticLog. A diagnostic service
530 may also implement other additional logging mechanisms. Any other implemented logging mechanism
531 shall be indicated in the LogStorage property of the published capabilities.

532 **7.6.1 Logging Results**

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

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

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

541 **7.7 CIM_DiagnosticRecord**

542 CIM_DiagnosticRecord has two subclasses: CIM_DiagnosticServiceRecord and
543 CIM_DiagnosticSettingDataRecord. CIM_DiagnosticServiceRecord has a single subclass:
544 CIM_DiagnosticCompletionRecord.

545 CIM_DiagnosticServiceRecord is structured to hold the information that is generated while a particular
546 service is running.

547 CIM_DiagnosticSettingDataRecord is structured to hold the attributes of the setting object that was used
548 as an input parameter to the RunDiagnosticService() method.

549 CIM_DiagnosticCompletionRecord is structured to hold the information that is generated as a result of
550 running the particular service.

551 **7.7.1 CIM_DiagnosticRecord.ExpirationDate**

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

555 CIM_DiagnosticSettingData.ResultPersistence shall be used by the client to specify to the diagnostic
556 service provider how long the results generated by that service shall persist. A value shall be chosen that
557 allows the minimum time needed by the client to record the data. When the timeout value has been
558 reached, the provider shall delete the data objects that contain the results.

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

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

567 **7.8 CIM_ServiceComponent**

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

572 The packaging test shall ensure that the values in CIM_DiagnosticTest.Characteristics of the packaging
573 test are consistent with the values in CIM_DiagnosticTest.Characteristics of the subtests unless the
574 packaging test can execute the subtest such that it does not have those characteristics. For example, if a
575 subtest sets the values Is Destructive or Is Interactive, the packaging test values in
576 CIM_DiagnosticTest.Characteristics should reflect those same characteristics, unless the packaging test
577 can execute the subtest so that it is not destructive or interactive.

578 8 Methods

579 This section details the requirements for supporting intrinsic operations and extrinsic methods for the CIM
580 elements defined by this profile.

581 8.1 CIM_DiagnosticService.RunDiagnosticService() Extrinsic Method

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

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

591 A job shall be instantiated to monitor the diagnostic service as it runs and to provide useful accounting
592 and status information when the diagnostic service has completed.

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

595 **Table 2 – RunDiagnosticService() Method: Return Code Values**

Value	Description
0	Request was successfully executed.
2	Unknown or unspecified error
3	Cannot complete within the timeout period
4	Failed
5	Invalid parameter
6	Busy – indicates that the method cannot be invoked "at this time." It is not an error condition, but signals that the provider is doing something else and cannot respond.
7..0x0FFF	DMTF reserved
0x1000..0x7FFF	Method reserved
0x8000..0xFFFF	Vendor specific

596

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	DiagSetting	[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	JobSetting	[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	Returns a reference to the resulting job

597 8.2 CIM_ConcreteJob.RequestStateChange() Extrinsic Method

598 All CIM_DiagnosticService.RunDiagnosticService() calls will return a reference to a CIM_ConcreteJob
 599 instance, which represents the diagnostic execution. The CIM_ConcreteJob.RequestStateChange()
 600 method is invoked to control the diagnostic program execution. The input parameters specify the
 601 execution control to be performed (Suspend, Kill, Terminate) and a timeout period that specifies the
 602 maximum amount of time that the client expects the transition to the new state to take.

603 Before invoking this method, clients examine the appropriate capabilities to verify whether the execution
 604 control is supported. The RequestStateChange() method shall change the JobState value if the transition
 605 is successfully performed.

606 RequestStateChange() return values are specified in Table 4 and parameters are specified in Table 5.
 607 No standard messages are defined.

608

Table 4 – RequestStateChange() Method: Return Code Values

Value	Description
0	Request was successfully executed.
2	Unknown/unspecified error
3	Cannot complete within the timeout period
4	Failed
5	Invalid parameter
6	Is in use
7..0x0FFF	DMTF reserved
0x1000	Method parameters checked – transition started
0x1001	Invalid state transition
0x1002	Use of timeout parameter not supported
0x1003	Busy – indicates that the method cannot be invoked "at this time." It is not an error condition, but signals that the provider is doing something else and cannot respond."
0x1004..0x7FFF	Method reserved
0x8000..0xFFFF	Vendor specific

609

Table 5 – RequestStateChange() Method: Parameters

Qualifiers	Name	Type	Description/Values
IN	RequestedState	uint16	The requested state of a job, which may be one of the following values: Start (2), Suspend (3), Terminate (4), Kill (5), or Service (6)
IN	TimeoutPeriod	datetime	A timeout period that specifies the maximum amount of time that the client expects the transition to the new state to take. The interval format shall be used to specify the TimeoutPeriod.

610 8.3 CIM_Log.ClearLog() Extrinsic Method

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

614 ClearLog return values are specified in Table 6.

615

Table 6 – ClearLog() Method: Return Code Values

Value	Description
0	Request was successfully executed.
2	Unknown or unspecified error
3	Cannot complete within the timeout period
4	Failed
5	Invalid parameter
"6..0x0FFF"	DMTF reserved
0x1000..0x7FFF	Method reserved
0x8000..0xFFFF	Vendor specific

616 8.4 CIM_HelpService.GetHelp() Extrinsic Method

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

619 The CIM_HelpService class has some attributes that publish the available documents, supported delivery
 620 types, and formats. See Table 8 for additional information. Before invoking this method, clients check
 621 these attributes in order to request an available document, format, and delivery type.

622 GetHelp() return values are specified in Table 7 and parameters are specified in Table 8. No standard
 623 messages are defined.

624

Table 7 – GetHelp() Method: Return Code Values

Value	Description
0	Request was successfully executed.
2	Unknown or unspecified error
3	Cannot complete within the timeout period
4	Failed
5	Invalid parameter
6..0x0FFF	DMTF reserved
0x1000	Busy – indicates that the method cannot be invoked "at this time." It is not an error condition, but signals that the provider is doing something else and cannot respond.
0x1001	Requested document not found
0x1002..0x7FFF	Method reserved
0x8000..0xFFFF	Vendor specific

625

Table 8 – 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.

626 8.5 Profile Conventions for Operations

627 Support for operations for each profile class (including associations) is specified in the following
 628 subclauses. Each subclause includes either the statement "All operations in the default list in section 8.5
 629 are supported as described by DSP0200 version 1.2" or a table listing all of the operations that are not
 630 supported by this profile or where the profile requires behavior other than that described by DSP0200
 631 version 1.2.

632 The default list of operations is as follows:

- 633 • GetInstance
- 634 • Associators
- 635 • AssociatorNames
- 636 • References
- 637 • ReferenceNames
- 638 • EnumerateInstances
- 639 • EnumerateInstanceNames

640 A compliant implementation shall support all of the operations in the default list for each class, unless the
641 "Requirement" column states something other than *Mandatory*.

642 8.6 CIM_DiagnosticTest

643 Table 9 lists operations that either have special requirements beyond those from DSP0200 version 1.2 or
644 shall not be supported.

645 **Table 9 – 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

646 8.7 CIM_AvailableDiagnosticService

647 Table 10 lists operations that either have special requirements beyond those from DSP0200 version 1.2
648 or shall not be supported.

649 **Table 10 – Operations: CIM_AvailableDiagnosticService**

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

650 8.8 CIM_ServiceAffectsElement

651 Table 11 lists operations that either have special requirements beyond those from DSP0200 version 1.2
652 or shall not be supported.

653 **Table 11 – Operations: CIM_ServiceAffectsElement**

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

654 **8.9 CIM_SoftwareIdentity**

655 Table 12 lists operations that either have special requirements beyond those from DSP0200 version 1.2
 656 or shall not be supported.

657 **Table 12 – 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

658 **8.10 CIM_ElementSoftwareIdentity**

659 Table 13 lists operations that either have special requirements beyond those from DSP0200 version 1.2
 660 or shall not be supported.

661 **Table 13 – Operations: CIM_ElementSoftwareIdentity**

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

662 **8.11 CIM_HelpService**

663 Table 14 lists operations that either have special requirements beyond those from DSP0200 version 1.2
 664 or shall not be supported.

665 **Table 14 – 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

666 **8.12 CIM_ServiceAvailableToElement**

667 Table 15 lists operations that either have special requirements beyond those from DSP0200 version 1.2
 668 or shall not be supported.

669 **Table 15 – Operations: CIM_ServiceAvailableToElement**

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

670 **8.13 CIM_DiagnosticSettingData**

671 Table 16 lists operations that either have special requirements beyond those from DSP0200 version 1.2
 672 or shall not be supported.

673 **Table 16 – 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

674 **8.14 CIM_DiagnosticServiceCapabilities**

675 Table 17 lists operations that either have special requirements beyond those from DSP0200 version 1.2
 676 or shall not be supported.

677 **Table 17 – 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

678 **8.15 CIM_ElementCapabilities**

679 Table 18 lists operations that either have special requirements beyond those from DSP0200 version 1.2
680 or shall not be supported.

681 **Table 18 – Operations: CIM_ElementCapabilities**

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

682 **8.16 CIM_ConcreteJob**

683 Table 19 lists operations that either have special requirements beyond those from DSP0200 version 1.2
684 or shall not be supported.

685 **Table 19 – Operations: CIM_ConcreteJob**

Operation	Requirement	Messages
GetInstance	Mandatory	None
ModifyInstance	Optional	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

686 **8.17 CIM_OwningJobElement**

687 Table 20 lists operations that either have special requirements beyond those from DSP0200 version 1.2
688 or shall not be supported.

689 **Table 20 – Operations: CIM_OwningJobElement**

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

690 **8.18 CIM_AffectedJobElement**

691 Table 21 lists operations that either have special requirements beyond those from DSP0200 version 1.2
 692 or shall not be supported.

693 **Table 21 – Operations: CIM_AffectedJobElement**

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

694 **8.19 CIM_JobSettingData**

695 Table 22 lists operations that either have special requirements beyond those from DSP0200 version 1.2
 696 or shall not be supported.

697 **Table 22 – Operations: CIM_JobSettingData**

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

698 **8.20 CIM_ElementSettingData**

699 Table 23 lists operations that either have special requirements beyond those from DSP0200 version 1.2
 700 or shall not be supported.

701 **Table 23 – Operations: CIM_ElementSettingData**

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

702 **8.21 CIM_DiagnosticLog**

703 Table 24 lists operations that either have special requirements beyond those from DSP0200 version 1.2
704 or shall not be supported.

705 **Table 24 – Operations: CIM_DiagnosticLog**

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

706 **8.22 CIM_UseOfLog**

707 Table 25 lists operations that either have special requirements beyond those from DSP0200 version 1.2
708 or shall not be supported.

709 **Table 25 – Operations: CIM_UseOfLog**

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

710 **8.23 CIM_DiagnosticServiceRecord**

711 Table 26 lists operations that either have special requirements beyond those from DSP0200 version 1.2
712 or shall not be supported.

713 **Table 26 – Operations: CIM_DiagnosticServiceRecord**

Operation	Requirement	Messages
GetInstance	Mandatory	None
CreateInstance	Optional	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

714 **8.24 CIM_DiagnosticCompletionRecord**

715 Table 27 lists operations that either have special requirements beyond those from DSP0200 version 1.2
 716 or shall not be supported.

717 **Table 27 – Operations: CIM_DiagnosticCompletionRecord**

Operation	Requirement	Messages
GetInstance	Mandatory	None
CreateInstance	Optional	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

718 **8.25 CIM_DiagnosticSettingDataRecord**

719 Table 28 lists operations that either have special requirements beyond those from DSP0200 version 1.2
 720 or shall not be supported.

721 **Table 28 – Operations: CIM_DiagnosticSettingDataRecord**

Operation	Requirement	Messages
GetInstance	Mandatory	None
CreateInstance	Optional	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

722 **8.26 CIM_LogManagesRecord**

723 Table 29 lists operations that either have special requirements beyond those from DSP0200 version 1.2
724 or shall not be supported.

725 **Table 29 – Operations: CIM_LogManagesRecord**

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

726 **8.27 CIM_RecordAppliesToElement**

727 Table 30 lists operations that either have special requirements beyond those from DSP0200 version 1.2
728 or shall not be supported.

729 **Table 30 – Operations: CIM_RecordAppliesToElement**

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

730 **8.28 CIM_CorrespondingSettingDataRecord**

731 Table 31 lists operations that either have special requirements beyond those from DSP0200 version 1.2
732 or shall not be supported.

733 **Table 31 – Operations: CIM_CorrespondingSettingDataRecord**

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

734 **8.29 CIM_ServiceComponent**

735 Table 32 lists operations that either have special requirements beyond those from DSP0200 version 1.2
736 or shall not be supported.

737 **Table 32 – Operations: CIM_ServiceComponent**

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

738 9 Use Cases

739 This section contains object diagrams and use cases for the *Diagnostics Profile*.

740 9.1 Profile Conformance

741 Conformance of a central class instance and its associated instances to a particular profile may be
742 identified by examining instances of the CIM_ElementConformsToProfile association class according to
743 the Central Class Methodology (see section 2.2). In some environments, an alternative method that relies
744 on the Scoping Class Methodology (see section 2.2) through the scoping class instance may be
745 desirable.

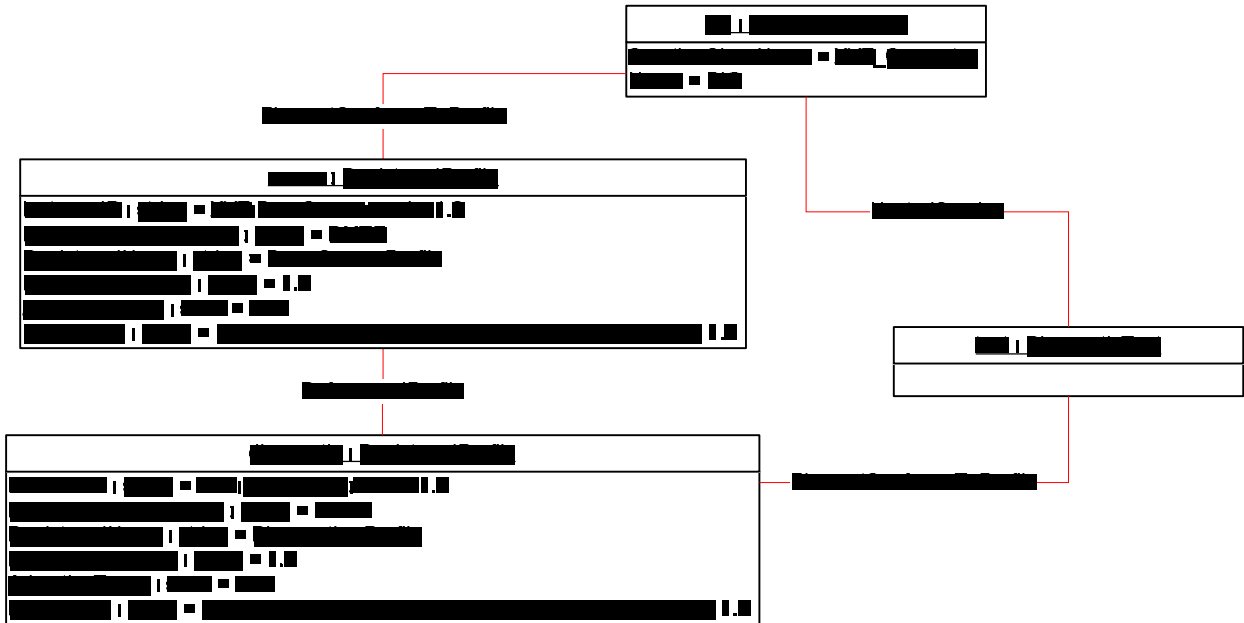
746 With CIM_ComputerSystem as the Scoping Class of this profile, the object diagram in Figure 2 shows
747 how instances of CIM_RegisteredProfile may be used to identify the version of the *Diagnostics Profile* to
748 which an instance of CIM_DiagnosticTest and its associated instances conform. In this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

751 To support the Scoping Class Methodology (see section 2.2) for advertising profile implementation
752 conformance, a CIM_DiagnosticTest instance is associated to an instance of the Scoping Class,
753 CIM_ComputerSystem, through an instance of CIM_HostedService. This instance of
754 CIM_ComputerSystem is advertised as being in implementation conformance with the *Base Server
755 Profile v1.0* as indicated by the CIM_ElementConformsToProfile association to the "server"
756 CIM_RegisteredProfile instance. The CIM_ReferencedProfile relationship between "server" and
757 "diagnostic" places the CIM_DiagnosticTest instance within the scope of "diagnostic." Thus, the
758 CIM_DiagnosticTest instance is conformant with the *Diagnostics Profile v2.0*.

759 To support the Central Class Methodology (see section 2.2) for advertising profile implementation
760 conformance, a CIM_ElementConformsToProfile association is established between the
761 CIM_DiagnosticTest central class instance and the instance of CIM_RegisteredProfile that represents the
762 *Diagnostics Profile*.

763 For these methodologies to be successful, profiles for systems that can support diagnostics need to
764 reference the *Diagnostics Profile*. In this example, the *Base Server Profile* would need to include the
765 *Diagnostics Profile* in its "Related Profiles" table.

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



767

768

Figure 2 – Registered Profile

769 **9.2 Use Case Summary**

770 Table 33 summarizes the use cases that are described in this section. The use cases are categorized
 771 and named, and references are provided to the body text that describes the use case.

772 **Note:** Although use case names follow the convention for naming classes, properties, and methods in the
 773 schema, this naming was done for readability only and does not imply any functionality attached to the
 774 name.

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

776

Table 33 – Diagnostics Profile Use Cases

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

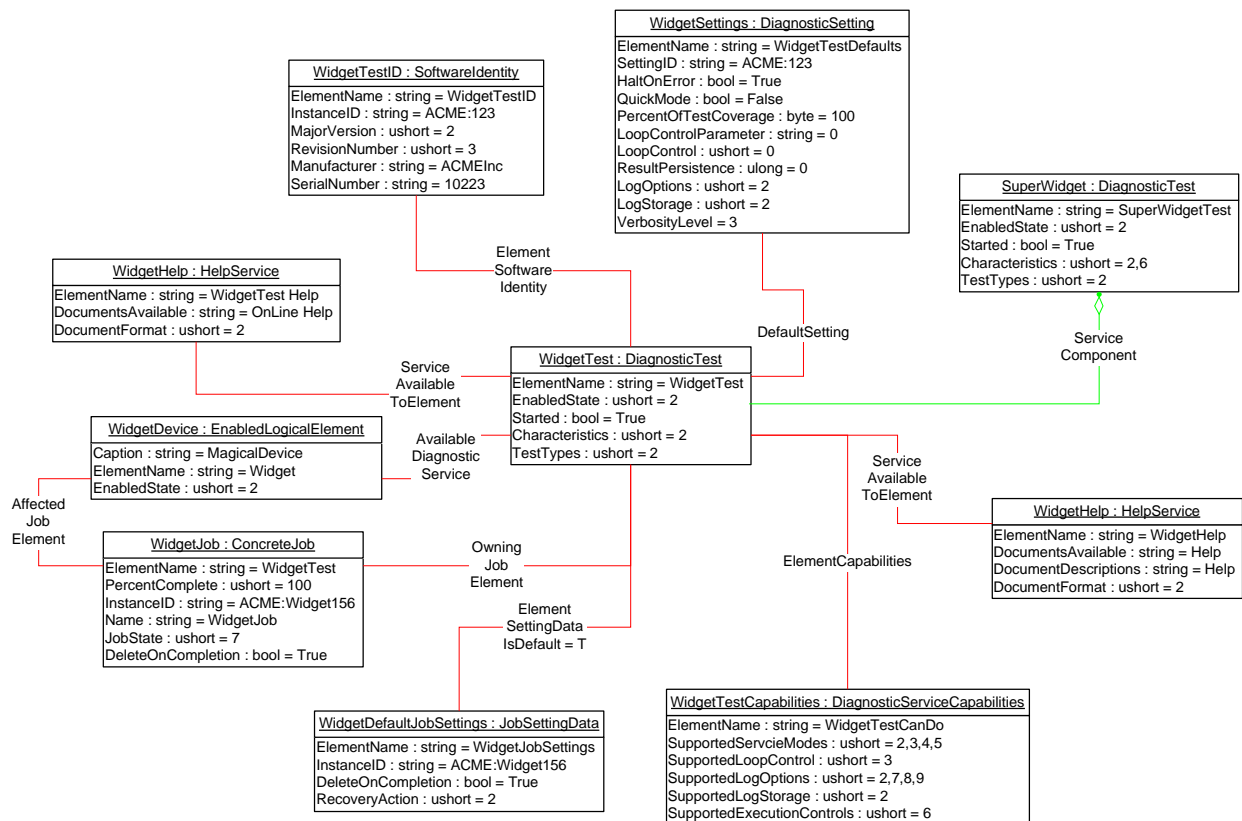
Category	Name	Description
	GetDiagnosticsWithCharacteristicsForME	Find all the diagnostics available on a system, for a managed element, with certain characteristics. See section 9.4.7.
	GetDiagnosticsWithCapabilitiesForME	Find all the diagnostics available on a system, for a managed element, with certain capabilities. See section 9.4.8.
	GetPackageSubtests	Find the subtests for a diagnostic test with the value of the DiagnosticTest.Characteristics property set to Is Package. See section 9.4.9.
Configure Diagnostic See section 9.5.	GetDefaultDiagnosticSettings	Find the default diagnostic settings for a diagnostic. See section 9.5.1.
	CreateDiagnosticSettings	Create a unique setting for a diagnostic. See section 9.5.2.
	GetDefaultJobSettings	Find the default job settings for a diagnostic. See section 9.5.3.
	CreateJobSettings	Create a unique setting for a diagnostic job. See section 9.5.4.
Execute and Control Diagnostic See section 9.6.	RunDiagnostic	Run a diagnostic with default and unique settings. See section 9.6.1.
	SuspendDiagnostic	Suspend a running diagnostic. See section 9.6.2.
	ResumeDiagnostic	Resume a running diagnostic. See section 9.6.3.
	AbortDiagnostic	Abort a running diagnostic. See section 9.6.4.
	KillDiagnostic	Abort a running diagnostic immediately, with no attempt to perform a clean shutdown. See section 9.6.5.
Discover Diagnostic Executions See section 9.7.	GetAffectedMEs	Find all the managed elements affected by a running diagnostic. See section 9.7.1.
	GetAllDiagnosticExecutionsForME	Find all the diagnostic executions on a system for a managed element. See section 0.
	GetSpecificDiagnosticExecutions	Find all the executions of a specific diagnostic. See section 0.
	GetSpecificDiagnosticExecutionsForME	Find all the executions of a specific diagnostic for a particular managed element. See section 9.7.4.
Discover Diagnostic Results (in-progress and final) See section 9.8.	GetLogRecordsForDiagnostic	Find all the diagnostic log records for a particular diagnostic. See section 9.8.1.
	GetLogRecordsForME	Find all the diagnostic log records for a particular managed element. See section 9.8.2.
	GetLogRecordsForMEAndDiagnostic	Find all the diagnostic log records for a particular diagnostic run on a particular managed element. See section 9.8.3.

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

777 **9.3 Diagnostic Services Object Diagram**

778 Figure 3 is an object diagram for diagnostic services for a fictitious device called “Widget.” Only classes,
779 properties, and methods that are of particular interest for the diagnostic model are shown. Refer to this
780 diagram for the use cases in this section.

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



782

783

Figure 3 – Diagnostic Services Object Diagram

784 **9.4 Discover Available Diagnostics**

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

787 **9.4.1 GetAllDiagnostics**

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

- 789 1) The client calls the EnumerateInstances (or EnumerateInstanceNames) operation using the
790 DiagnosticTest class.
- 791 2) The operation returns DiagnosticTest instances that represent a diagnostic that is available on
792 the system.

793 **9.4.2 GetAllDiagnosticMEPairs**

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

- 796 1) The client calls the EnumerateInstances (or EnumerateInstanceNames) operation using the
797 AvailableDiagnosticService class.
- 798 2) The operation returns AvailableDiagnosticService instances that have a reference to the
799 DiagnosticTest instance and another reference to the ManagedElement instance.

800 **9.4.3 GetDiagnosticsForME**

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

- 804 3) From the ManagedElement instance, the client calls the Associators operation
805 using AvailableDiagnosticService as the association class.
- 806 4) The operation returns DiagnosticTest instances that represent a diagnostic that can be
807 launched against the ManagedElement.

808 **9.4.4 GetMEsForDiagnostic**

809 The client can find all managed elements (devices) that are supported by a specific diagnostic as follows.
810 Assume that the client starts at a known DiagnosticTest instance.

- 811 1) From the DiagnosticTest instance, the client calls the Associators operation
812 using AvailableDiagnosticService as the association class.
- 813 2) The operation returns ManagedElement instances that represent a device that is supported by
814 the DiagnosticTest.

815 **9.4.5 GetCapabilitiesOfDiagnostic**

816 A diagnostic service publishes its support for various options—in particular, settings—through a
817 DiagnosticServiceCapabilities instance. If a setting is supported, the client can assign it, usually to satisfy
818 a user request. The client should be able to find the capabilities of a diagnostic as follows. Assume that
819 the client starts at a known DiagnosticTest instance.

- 820 1) From the DiagnosticTest instance, the client calls the Associators operation
821 using ElementCapabilities as the association class and DiagnosticServiceCapabilities as the
822 result class.
- 823 2) The operation should return only one DiagnosticServiceCapabilities instance, which represents
824 the diagnostic capabilities.

825 **Note:** Because the implementation of DiagnosticServiceCapabilities is optional, it may not be available. In
826 this case, no assumptions should be made regarding the diagnostic capabilities.

827 **9.4.6 GetCharacteristicsOfDiagnostic**

828 The client can discover all of the characteristics (is destructive, is interactive, is synchronous, and so on)
829 of a diagnostic. From the DiagnosticTest instance, the client reads just the Characteristics and
830 OtherCharacteristicsDescriptions attributes, which contain the diagnostic characteristics.

831 **9.4.7 GetDiagnosticswithCharacteristicsForME**

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

- 835 1) The client discovers all of the diagnostics that are available for the specific ManagedElement.
836 The GetDiagnosticsForME use case (section 9.4.3) describes the necessary steps.
- 837 2) For each DiagnosticTest instance, the client checks the diagnostic characteristics. The
838 GetCharacteristicsOfDiagnostic use case (section 9.4.6) describes the necessary steps.
- 839 3) If the characteristics of the DiagnosticTest instance match the desired characteristics, the
840 DiagnosticTest instance is the one desired.

841 **9.4.8 GetDiagnosticswithCapabilitiesForME**

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

- 845 1) The client discovers all of the diagnostics that are available for the specific ManagedElement.
846 The GetDiagnosticsForME use case (section 9.4.3) describes the necessary steps.
- 847 2) For each DiagnosticTest instance, the client checks the diagnostic capabilities. The
848 GetCapabilitiesOfDiagnostic use case (section 9.4.5) describes the necessary steps.
- 849 3) If the capabilities of the DiagnosticTest instance match the desired capabilities, the
850 DiagnosticTest instance is the one desired.

851 **9.4.9 GetPackageSubtests**

852 The client can find the subtests for a diagnostic test with the IsPackage value set in the
853 DiagnosticTest.Characteristics property, using the following procedure. Assume that the client starts at a
854 known DiagnosticTest instance.

- 855 1) The client checks the DiagnosticTest.Characteristics property for the IsPackage value.
- 856 2) If the IsPackage value is present, the client calls the Associators operation using
857 ServiceComponent as the association class and DiagnosticTest as the result class.
- 858 3) The operation returns the DiagnosticTest instances that are subtests of the known
859 DiagnosticTest.

860 **9.5 Configure Diagnostic**

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

863 9.5.1 GetDefaultDiagnosticSettings

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

- 866 1) From the DiagnosticTest instance, the client calls the Associators operation
867 using ElementSettingData as the association class and DiagnosticSettingData as the result
868 class. The operation returns DiagnosticSettingData instances.
- 869 2) For each DiagnosticSettingData instance, the client calls the References operation using
870 ElementSettingData as the result class. The operation returns ElementSettingData instances.
- 871 3) For each ElementSettingData instance, the client determines whether the value of the
872 ElementSettingData.ManagedElement property matches the DiagnosticTest name and the
873 value of the ElementSettingData.IsDefault property is 1 (Is Default). If so, the
874 DiagnosticSettingData instance represents the default diagnostic settings. The name of this
875 DiagnosticSettingData instance may also be retrieved from ElementSettingData.SettingData
876 property.

877 **Note:** Because the implementation of DiagnosticSettingData is optional, it may not be available.

878 9.5.2 CreateDiagnosticSettings

879 The client may modify the diagnostic settings as follows if it wants to use settings different than the
880 default settings. Note that the diagnostic default settings are represented by a DiagnosticSettingData
881 subclass that may have extensions. If the client is aware of the extensions, they may be modified as well.
882 If the client is unaware, the default values should be used. Assume that the client starts at a known
883 DiagnosticTest instance.

- 884 1) The client discovers the diagnostic capabilities of the DiagnosticTest instance. The
885 GetCapabilitiesOfDiagnostic use case (section 9.4.5) describes the necessary steps. If no
886 capability information is available, the client shall use the default settings (empty string or NULL)
887 because it cannot assume any diagnostic capability.
- 888 2) The client discovers the diagnostic default settings of the DiagnosticTest instance. The
889 GetDefaultDiagnosticSettings use case (section 9.5.1) describes the necessary steps.
- 890 3) If step 2 does not return an instance, the client performs a GetClass operation on the
891 DiagnosticSettingData class and locally (in the client scope—for example, IwbemClassObject or
892 CIMInstance object) creates an instance based on the DiagnosticSettingData class definition.
- 893 4) The client modifies the created DiagnosticSettingData instance as necessary. However, the
894 client should consider the diagnostic capabilities during the changes.

895 9.5.3 GetDefaultJobSettings

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

- 898 1) From the DiagnosticTest instance, the client calls the Associators operation
899 using ElementSettingData as the association class and JobSettingData as the result class. The
900 operation returns JobSettingData instances.
- 901 2) For each JobSettingData instance, the client calls the References operation using
902 ElementSettingData as the result class. The operation returns ElementSettingData instances.
- 903 3) For each ElementSettingData instance, the client determines whether the value of the
904 ElementSettingData.ManagedElement property matches the DiagnosticTest name and the
905 value of the ElementSettingData.IsDefault property is 1 ("Is Default"). If so, the JobSettingData
906 instance represents the default job settings. The name of this JobSettingData instance may also
907 be retrieved from ElementSettingData.SettingData property.

908 **Note:** Because the implementation of JobSettingData is optional, it may not be available.

909 **9.5.4 CreateJobSettings**

910 The client can modify the diagnostic job settings as follows if it wants to use settings different than the
911 default job settings. Note that the diagnostic default job settings are represented by a JobSettingData
912 subclass that may have extensions. If the client is aware of the extensions, they may be modified as well.
913 If the client is unaware, the default values should be used. Assume that the client starts at a known
914 DiagnosticTest instance.

- 915 1) The client discovers the diagnostic capabilities of the DiagnosticTest instance. The
916 GetCapabilitiesOfDiagnostic use case (section 9.4.5) describes the necessary steps. If no
917 capability information is available, the client shall use the default settings (empty string or NULL)
918 because it cannot assume any diagnostic capability.
- 919 2) The client discovers the diagnostic default settings of the DiagnosticTest instance. The
920 GetDefaultJobSettings use case (section 9.5.3) describes the necessary steps.
- 921 3) If step 2 does not return any instance, the client performs a GetClass operation on the
922 JobSettingData class and locally (in the client scope—for example, IwbemClassObject or
923 CIMInstance object) creates an instance based on the JobSettingData class definition.
- 924 4) The client modifies the created JobSettingData instance as necessary. However, the client
925 should consider the diagnostic capabilities during the changes.

926 **9.6 Execute and Control Diagnostic**

927 The RunDiagnosticService() method is invoked to start the diagnostic service. Input parameters are the
928 ManagedElement being tested and the settings (optional). A reference to a ConcreteJob instance is
929 returned.

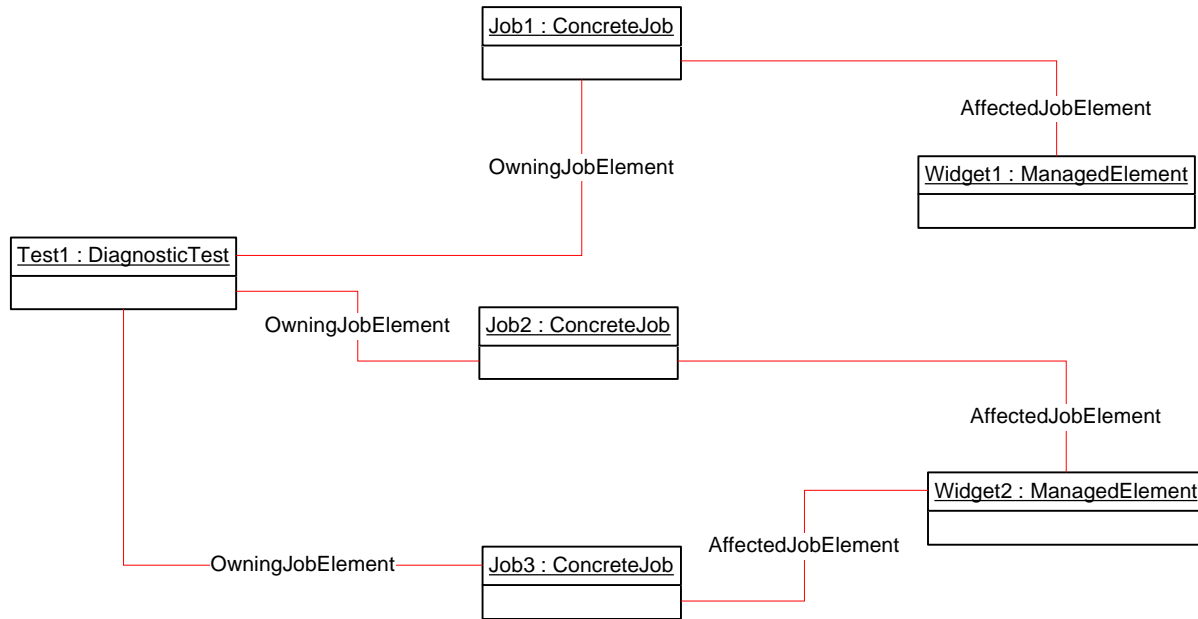
930 An instance of ConcreteJob is created by the diagnostic provider to allow monitoring and control of the
931 running service. By invoking the RequestStateChange method, the client may start, stop, suspend, and
932 resume the job. By inspecting the value of PercentComplete, the client may determine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

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

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

939 **Note:** Not all diagnostic tests are capable of running on the same device simultaneously. If this had been
940 the case in this example, the DiagnosticTest would have returned an error on the second
941 RunDiagnosticService() method call to run a test on ManagedElement2.

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



943

944

Figure 4 – Job Example

945 **9.6.1 RunDiagnostic**

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

- 949 1) The client calls the RunDiagnosticService() method, passing in EmbeddedInstances of
 950 DiagnosticSettings and JobSettings to use to execute the test as well as the reference to the
 951 ManagedElement to test. If the client passes in NULL or an empty string for these classes, the
 952 default values are used.
- 953 2) The diagnostic service creates a Job instance to represent that test running on that managed
 954 element and returns a reference to it in the return call from RunDiagnosticService(). In addition,
 955 the diagnostic service creates the OwningJobElement association between the Job and the
 956 Service and the AffectedJobElement association between the Job and the ManagedElement.

957 **9.6.2 SuspendDiagnostic**

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

- 961 1) The client follows the ElementCapabilities association from the DiagnosticTest to the
 962 DiagnosticServiceCapabilities for the service.
- 963 2) The client checks the DiagnosticServiceCapabilities.SupportedExecutionControls() property for
 964 the value of "Suspend Job". If the value exists, the Job supports suspending.
- 965 3) The client finds the appropriate Job instances. The GetSpecificDiagnosticExecutions use case
 966 (section 0) describes the necessary steps.
- 967 4) The client calls the RequestStateChange() method, passing in a RequestedState value of
 968 "Suspend".

- 969 5) When the transition completes successfully, the ConcreteJob that represents the test will set the
970 value of the JobState property to “Suspended” and set the value of TimeOfLastStateChange to
971 the current time.

972 **9.6.3 ResumeDiagnostic**

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

- 976 1) The client follows the ElementCapabilities association from the DiagnosticTest to the
977 DiagnosticServiceCapabilities for the service.
- 978 2) The client checks the DiagnosticServiceCapabilities.SupportedExecutionControls() property for
979 the value of “Resume Job”. If the value exists, the Job supports resuming.
- 980 3) The client finds the appropriate Job instances. The GetSpecificDiagnosticExecutions use case
981 (section 0) describes the necessary steps.
- 982 4) The client calls the RequestStateChange() method, passing in a RequestedState value of
983 “Start”.
- 984 5) When the transition completes successfully, the ConcreteJob that represents the test will set the
985 value of the JobState property to “Running” and set the value of TimeOfLastStateChange to the
986 current time.

987 **Note:** The JobState property may transition to “Starting” before the final transition to “Running”.

988 **9.6.4 AbortDiagnostic**

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

- 992 1) The client follows the ElementCapabilities association from the DiagnosticTest to the
993 DiagnosticServiceCapabilities for the service.
- 994 2) The client checks the DiagnosticServiceCapabilities.SupportedExecutionControls() property for
995 the value of “Terminate Job”. If the value exists, the Job supports termination.
- 996 3) The client finds the appropriate Job instances. The GetSpecificDiagnosticExecutions use case
997 (section 0) describes the necessary steps.
- 998 4) The client calls the RequestStateChange() method, passing in a RequestedState value of
999 “Terminate”.
- 1000 5) When the transition completes successfully, the ConcreteJob that represents the test will set the
1001 value of the EnabledState property to “Terminated” and set the value of
1002 TimeOfLastStateChange to the current time.

1003 **Note:** The JobState property may transition to “Shutting Down” before the final transition to “Terminated”.

1004 **9.6.5 KillDiagnostic**

1005 The client can immediately abort the execution of a test, with no attempt to perform a clean shutdown, by
1006 using the RequestStateChange() method call on the Job instance that is returned from the
1007 RunDiagnosticService() method, as shown in the following procedure. Assume that the client starts at a
1008 known DiagnosticTest instance.

- 1009 1) The client follows the ElementCapabilities association from the DiagnosticTest to the
1010 DiagnosticServiceCapabilities for the service.

- 1011 2) The client checks the DiagnosticServiceCapabilities.SupportedExecutionControls() property for
1012 the value of "Kill Job". If the value exists, the Job supports kill.
- 1013 3) The client finds the appropriate Job instances. The GetSpecificDiagnosticExecutions use case
1014 (section 0) describes the necessary steps.
- 1015 4) The client calls the RequestStateChange() method, passing in a RequestedState value of "Kill".
- 1016 5) When the transition completes successfully, the ConcreteJob that represents the test will set the
1017 value of the EnabledState property to "Killed" and set the value of TimeOfLastStateChange to
1018 the current time.

1019 9.7 Discover Diagnostic Executions

1020 In the following use cases, the term *execution* refers to an instance of the ConcreteJob class created to
1021 control a diagnostic service that was started on a managed element. The job may be in any of the states
1022 represented by the JobState property value, not necessarily active and running.

1023 The CIM_ prefix has been omitted from the class names in the use cases for readability.

1024 9.7.1 GetAffectedMEs

1025 The client can find all of the managed elements that are affected by a diagnostic execution as follows.
1026 Assume that the client starts at a known DiagnosticTest instance.

- 1027 1) From the DiagnosticTest instance, the client calls the Associators operation using
1028 OwningJobElement as the association class and ConcreteJob as the result class. The operation
1029 returns the ConcreteJob instances launched by the DiagnosticTest.
- 1030 2) For each ConcreteJob instance, the client calls the Associators operation using
1031 AffectedJobElement as the association class and ManagedElement as the result class. The
1032 operation returns the ManagedElement instances that this DiagnosticTest affects.

1033 **Note:** This use case depends on the optional AffectedJobElement association. If that association does
1034 not exist, this use case is invalid.

1035 9.7.2 GetAllDiagnosticExecutionsForME

1036 The client can find all of the diagnostic executions on a system for a managed element as follows.
1037 Assume that the client starts at a known ManagedElement instance.

- 1038 1) From the ManagedElement instance, the client calls the Associators operation
1039 using AffectedJobElement as the association class. The operation returns the ConcreteJob
1040 instances launched against this ManagedElement.
- 1041 2) For each ConcreteJob instance, the client calls the AssociatorNames operation using
1042 OwningJobElement as the association class and DiagnosticTest as the result class. The
1043 operation returns the instance paths to the DiagnosticTest instances that launched the
1044 ConcreteJob against this ManagedElement.
- 1045 3) Each ConcreteJob instance that is associated with a DiagnosticTest represents an execution of
1046 a diagnostic service on that ManagedElement.

1047 **Note:** This use case depends on the optional AffectedJobElement association. If that association does
1048 not exist, this use case is invalid.

1049 9.7.3 GetSpecificDiagnosticExecutions

1050 The client can find all of the executions of a specific diagnostic as follows. Assume that the client starts at
1051 a known DiagnosticTest instance.

Diagnostics Profile

1052 1) From the DiagnosticTest instance, the client calls the Associators operation
1053 using OwingJobElement as the association class. The operation returns the ConcreteJob
1054 instances launched by the DiagnosticTest.

1055 2) Each ConcreteJob instance represents an execution of that diagnostic service.

1056 **9.7.4 GetSpecificDiagnosticExecutionsForME**

1057 The client can find all of the executions of a specific diagnostic for a particular managed element using
1058 either of the following methods:

- 1059 • starting at the known ManagedElement instance
- 1060 • starting at the known DiagnosticTest instance

1061 **9.7.4.1 Starting at the Managed Element**

1062 **Note:** This use case depends on the optional AffectedJobElement association. If that association does
1063 not exist, this use case is invalid.

1064 Assume that the client starts at the known ManagedElement instance and knows the particular
1065 DiagnosticTest instance.

- 1066 1) From the ManagedElement instance, the client calls the Associators operation
1067 using AffectedJobElement as the association class and ConcreteJob as the result class. The
1068 operation returns the ConcreteJob instances that are running against this ManagedElement.
- 1069 2) For each ConcreteJob instance, the client calls the AssociatorNames operation using
1070 OwingJobElement as the association class and DiagnosticTest as the result class. The
1071 operation returns the instance paths to the DiagnosticTest instances that launched the
1072 ConcreteJob instances against this ManagedElement.
- 1073 3) For each DiagnosticTest instance path returned, the client determines if it is the instance path of
1074 the known DiagnosticTest instance. If the instance path matches, the ConcreteJob instance
1075 represents an execution of that diagnostic service on that ManagedElement.

1076 **9.7.4.2 Starting at the DiagnosticTest**

1077 **Note:** This use case depends on the optional AffectedJobElement association. If that association does
1078 not exist, this use case is invalid.

1079 Assume that the client starts at the known DiagnosticTest instance and knows the particular
1080 ManagedElement instance.

- 1081 1) From the DiagnosticTest instance, the client calls the Associators operation using
1082 OwingJobElement as the association class and ConcreteJob as the result class. The operation
1083 returns the ConcreteJob instances launched by the DiagnosticTest.
- 1084 2) For each ConcreteJob instance, the client calls the AssociatorNames operation using
1085 AffectedJobElement as the association class and ManagedElement as the result class. The
1086 operation returns the instance paths to the ManagedElement instances against which this
1087 DiagnosticTest launched the ConcreteJob instances.
- 1088 3) For each ManagedElement instance path returned, the client determines if it is the instance
1089 path of the known ManagedElement instance. If the instance path matches, the ConcreteJob
1090 instance represents an execution of that diagnostic service on that ManagedElement.

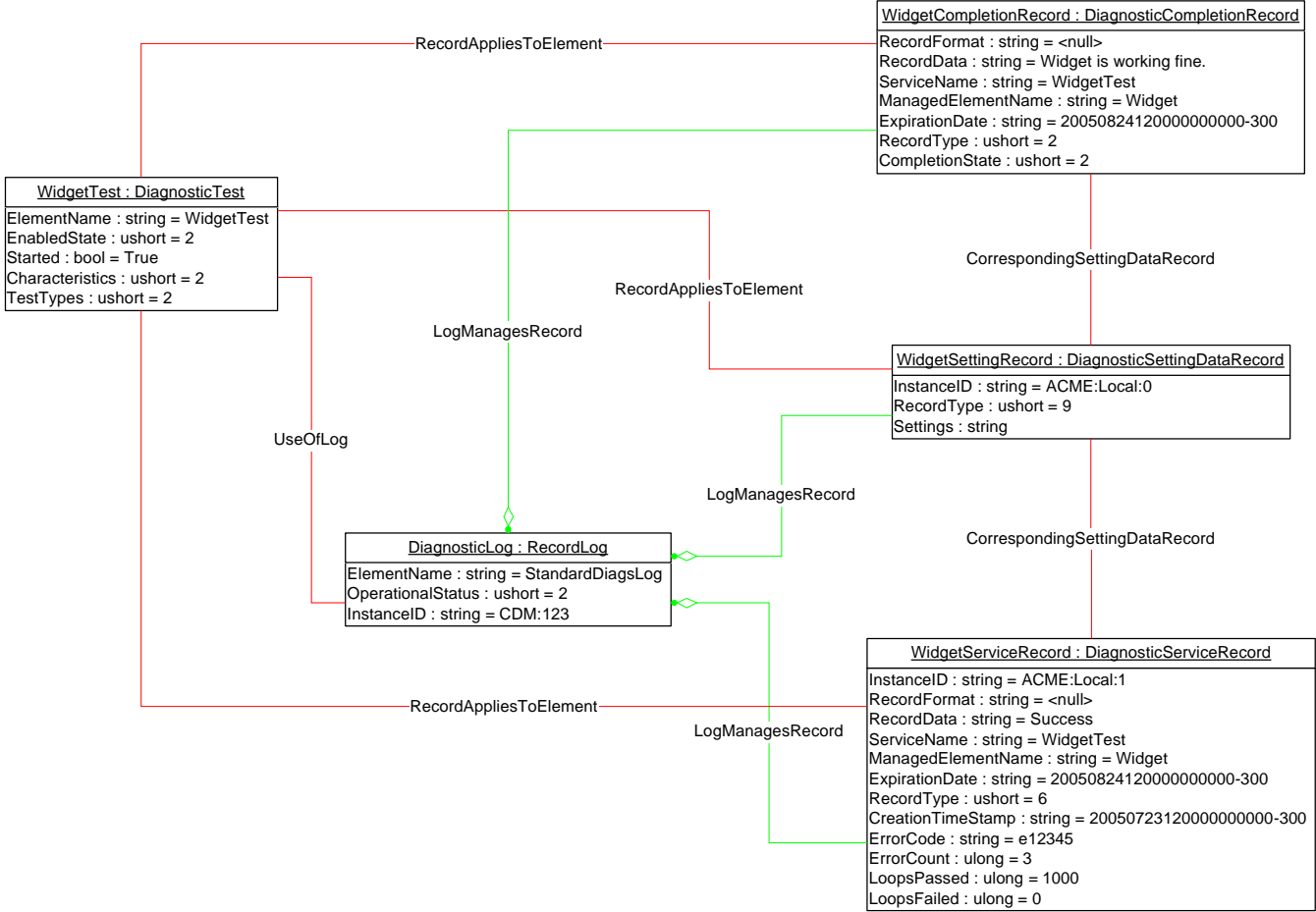
1091 **9.8 Discover Diagnostic Results (In Progress and Final)**

1092 In the following use cases, the term *execution* refers to an instance of the ConcreteJob class created to
1093 control a diagnostic service that was started on a managed element. The job may be in any of the states
1094 represented by the JobState property value, not necessarily active and running.

1095 Figure 5 is an object diagram that represents the results logging process for a diagnostic service on a
1096 fictitious device called "Widget". Only classes, properties, and methods that are of particular interest for
1097 the diagnostic model are shown.

1098 Figure 5 shows the required logging implementation, using the DiagnosticLog class. DiagnosticLog is a
1099 special subclass of RecordLog that supports a standard mechanism for organizing and retrieving (using
1100 ExecQuery) the records that diagnostics services generate. Use of this common logging mechanism can
1101 substantially increase interoperability and simplify client design.

1102 The CIM_ prefix has been omitted from the class names in the diagram and use cases for readability.



1103
1104

1105 **Figure 5 – Diagnostic Logging Object Diagram**

1106 9.8.1 GetLogRecordsForDiagnostic

1107 The client can find all of the diagnostic log records for a particular diagnostic as follows. Assume that the
 1108 client starts at the known DiagnosticTest instance and that the DiagnosticRecord.ServiceName property
 1109 is implemented according to this profile.

1110 1) The client calls the ExecQuery operation as follows:

```
1111 SELECT * FROM CIM_DiagnosticRecord
1112 WHERE ServiceName = '<DiagnosticTest.Name>'
```

1113 2) The operation returns the DiagnosticRecord instances created for the specific DiagnosticTest,
 1114 independently if they are related to different managed elements or executions.

1115 9.8.2 GetLogRecordsForME

1116 The client can find all of the diagnostic log records for a particular managed element as follows. Assume
 1117 that the client starts at the known ManagedElement instance and that the
 1118 DiagnosticRecord.ManagedElementName property is implemented according to this profile.

1119 1) The client calls the ExecQuery operation as follows:

```
1120 SELECT * FROM CIM_DiagnosticRecord
1121 WHERE ManagedElementName = '<ManagedElement.ElementName>'
```

1122 2) The operation returns the DiagnosticRecord instances created for the specific
 1123 ManagedElement, independently if they are related to different diagnostics or executions.

1124 9.8.3 GetLogRecordsForMEAndDiagnostic

1125 The client can find all of the diagnostic log records for a particular diagnostic run on a particular managed
 1126 element as follows.

1127 Assume that the client starts at the known DiagnosticTest and ManagedElement instances and that the
 1128 DiagnosticRecord.ServiceName and DiagnosticRecord.ManagedElementName properties are
 1129 implemented according to this profile.

1130 1) The client calls the ExecQuery operation as follows:

```
1131 SELECT * FROM CIM_DiagnosticRecord
1132 WHERE ManagedElementName = '<ManagedElement.ElementName>' and ServiceName =
1133 '<DiagnosticTest.Name>'
```

1134 2) The operation returns the DiagnosticRecord instances created for the specific ManagedElement
 1135 and DiagnosticTest, independently if they were created in different executions.

1136 9.8.4 GetDiagnosticExecutionFinalResults

1137 The client can determine the final result of a diagnostic as follows. Assume that the client starts at the
 1138 known ConcreteJob instance and that the DiagnosticRecord.InstanceID property follows the format
 1139 defined in this profile (CIM_DiagnosticRecord.InstanceID *should* be <ConcreteJob.InstanceID>:<n>).
 1140 This use case is also applicable after the job completes and is removed if the client knows the original
 1141 ConcreteJob.InstanceID.

1142 1) The client calls the ExecQuery operation as follows:

```
1143 SELECT * FROM CIM_DiagnosticCompletionRecord
1144 WHERE InstanceID LIKE '<ConcreteJob.InstanceID>%'
```

1145 2) The operation returns the DiagnosticCompletionRecord instance created for the specific
 1146 ConcreteJob.

1147 3) The client reads the DiagnosticCompletionRecord.CompletionState property, which shows the
 1148 final result (Passed, Warning, Failed, Aborted, Incomplete, and so on) of the diagnostic
 1149 execution.

1150 **9.8.5 GetDiagnosticExecutionResults**

1151 The client can find all diagnostic log records for a particular execution (job) as follows.

1152 The diagnostic provider will store the results of running the diagnostic in the manner selected through the
 1153 LogStorage setting. The most common mechanism is for the provider to create instances of
 1154 DiagnosticRecord to record the results and status of running diagnostic services. DiagnosticRecord has
 1155 two subclasses: DiagnosticServiceRecord for recording test results, and DiagnosticSettingDataRecord for
 1156 preserving the test settings. The providers for these classes will implement ExecQuery to simplify the
 1157 retrieval of records.

1158 The records are aggregated to a log by the LogManagesRecord association.

1159 Assume that the client starts at the known ConcreteJob instance and that the
 1160 DiagnosticRecord.InstanceID property follows the format defined in this profile
 1161 (CIM_DiagnosticRecord.InstanceID *should* be <ConcreteJob.InstanceID>:<n>). This use case is also
 1162 applicable after the job completes and is removed if the client knows the original ConcreteJob.InstanceID.

1163 1) The client calls the ExecQuery operation as follows:

```
1164 SELECT * FROM CIM_DiagnosticRecord
1165 WHERE InstanceID LIKE '<ConcreteJob.InstanceID>%'
```

1166 2) The operation returns the DiagnosticRecord instances created for the specific ConcreteJob.

1167 **9.8.6 GetDiagnosticExecutionSettings**

1168 The client can find the settings used to execute a diagnostic as follows.

1169 Assume that the client starts at the known ConcreteJob instance and that the
 1170 DiagnosticRecord.InstanceID property follows the format defined in this profile
 1171 (CIM_DiagnosticRecord.InstanceID *should* be <ConcreteJob.InstanceID>:<n>). This use case is also
 1172 applicable after the job completes and is removed if the client knows the original ConcreteJob.InstanceID.

1173 1) The client calls the ExecQuery operation as follows:

```
1174 SELECT * FROM CIM_DiagnosticSettingDataRecord
1175 WHERE InstanceID LIKE '<ConcreteJob.InstanceID>%'
```

1176 2) The operation returns the DiagnosticSettingDataRecord instance created for the specific
 1177 ConcreteJob.

1178 3) The client reads the DiagnosticCompletionRecord.Settings property, which is a
 1179 DiagnosticSettingData embedded instance that contains the settings of the diagnostic
 1180 execution.

1181 **9.8.7 GetDiagnosticProgress**

1182 The client can get the progress of a running diagnostic as follows.

1183 The client may poll the ConcreteJob.PercentComplete property to determine test progress or register for
 1184 an indication that this property has changed. The value of this property shall be kept current to be useful.
 1185 Service providers should update this property within one second of becoming aware of a progress
 1186 change.

1187 1) The client may use any of the Discover Diagnostic Execution use cases (section 9.7) to find the
 1188 desired ConcreteJob instances.

Element Name	Requirement	Description
CIM_DiagnosticServiceRecord	Mandatory	Records that contain the results of running a diagnostic service See sections 7.7 and 10.8.
CIM_DiagnosticSettingData	Optional	See sections 7.4 and 10.9.
CIM_DiagnosticSettingDataRecord	Conditional	Records that contain the settings that were used by the diagnostic service. If CIM_DiagnosticSettingData is implemented, this class is Mandatory. See sections 7.7 and 10.10.
CIM_DiagnosticTest	Mandatory	See sections 7.1 and 10.11.
CIM_ElementCapabilities	Conditional	Association to link a Capabilities object to a diagnostic service. If Capabilities is implemented, this association is Mandatory. See section 10.12.
CIM_ElementSettingData	Conditional	If CIM_JobSettingData or CIM_DiagnosticSettingData are implemented, this association is Mandatory. See section 10.13.
CIM_ElementSoftwareIdentity	Mandatory	See section 10.14.
CIM_HelpService	Mandatory	See section 10.15.
CIM_HostedService	Mandatory	See sections 10.16 and 9.1.
CIM_JobSettingData	Optional	See section 10.17.
CIM_LogManagesRecord	Mandatory	See section 10.18.
CIM_OwningJobElement	Mandatory	Association to link a job to a diagnostic service See section 10.19.
CIM_RecordAppliesToElement	Optional	Association to link records to the managed element to which they apply See section 10.20.
CIM_RegisteredProfile	Mandatory	See section 10.21.
CIM_ServiceAffectsElement	Mandatory	See section 10.22.
CIM_ServiceAvailableToElement	Mandatory	See section 10.23.
CIM_ServiceComponent	Conditional	Association to link a packaging test to its subtests. If a DiagnosticTest.Characteristic property contains the IsPackage value and the subtests are also instances of DiagnosticTest, this association shall be used to associate those subtests to the packaging DiagnosticTest. See section 10.24.
CIM_SoftwareIdentity	Mandatory	An instance of CIM_SoftwareIdentity shall exist for each CIM_DiagnosticTest instance. See section 10.25.
CIM_UseOfLog	Mandatory	See section 10.26.
Indications		
None defined in this profile		

1208 **10.1 CIM_AffectedJobElement**

1209 CIM_AffectedJobElement is used to associate a job with its affected managed elements (devices). Table
 1210 35 provides information about the properties of CIM_AffectedJobElement.

1211 **Table 35 – Class: CIM_AffectedJobElement**

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_ConcreteJob.

1212 **10.2 CIM_AvailableDiagnosticService**

1213 CIM_AvailableDiagnosticService is used to discover the diagnostic services that are installed for a
 1214 particular managed element. Table 36 provides information about the properties of
 1215 CIM_AvailableDiagnosticService.

1216 **Table 36 – 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 section 7.2.1.
EstimatedDurationQualifier	Optional	See section 7.2.2.

1217 **10.3 CIM_ConcreteJob**

1218 Each successful RunDiagnosticService() call will return a CIM_ConcreteJob instance. Each
 1219 CIM_ConcreteJob instance represents a diagnostic execution. Table 37 provides information about the
 1220 properties of CIM_ConcreteJob.

1221 **Table 37 – Class: CIM_ConcreteJob**

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 "^.*[:].*\$")
Name	Mandatory	The property will be formatted as a free-form string of variable length. (pattern ".**")
JobState	Mandatory	None
TimeBeforeRemoval	Mandatory	See section 7.5.1.

Properties	Requirement	Notes
StartTime	Mandatory	None
ElapsedTime	Mandatory	This property should be updated periodically so as to be useful as a "heartbeat."
PercentComplete	Mandatory	See section 7.5.2.
DeleteOnCompletion	Optional	The default value for this property is TRUE.
ErrorDescription	Conditional	If ErrorCode is implemented, ErrorDescription should be filled in to explain the error.
RequestStateChange()	Mandatory	See section 8.2.

1222 10.4 CIM_CorrespondingSettingDataRecord

1223 CIM_CorrespondingSettingDataRecord is used to associate a service (or completion) record with the
 1224 corresponding setting record. Table 38 provides information about the properties of
 1225 CIM_CorrespondingSettingDataRecord.

1226 **Table 38 – 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

1227 10.5 CIM_DiagnosticCompletionRecord

1228 CIM_DiagnosticCompletionRecord is used to report the final state of diagnostic execution (OK, Failed,
 1229 Incomplete, Aborted, and so on). Table 39 provides information about the properties of
 1230 CIM_DiagnosticCompletionRecord.

1231 **Table 39 – Class: CIM_DiagnosticCompletionRecord**

Properties	Requirement	Notes
InstanceID	Mandatory	Key InstanceID should be constructed using the following preferred algorithm: <ConcreteJob.InstanceID>:<n> < 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. (pattern "^.*[:].*[:][0123456789]*\$")
CreationTimeStamp	Mandatory	None

Properties	Requirement	Notes
RecordData	Mandatory	None
RecordFormat	Mandatory	None
ServiceName	Mandatory	The ServiceName property shall be constructed as follows: <OrgID>:<TestName>. (pattern "^.*[:].*\$")
ManagedElementName	Mandatory	This property will be formatted as a free-form string of variable length. (pattern ".*")
RecordType	Mandatory	The record type shall be "Results". Matches 2 (Results)
ExpirationDate	Mandatory	See section 7.7.1.
CompletionState	Mandatory	None
OtherCompletionStateDescription	Conditional	If CompletionState is set to "Other", this property shall be filled in.
LoopsPassed	Conditional	If looping is supported, this property is Mandatory.
LoopsFailed	Conditional	If looping is supported, this property is Mandatory.
ErrorCode()	Mandatory	This property shall be an array that contains the error codes of all errors generated by the diagnostic service execution.
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.

1232 **10.6 CIM_DiagnosticLog**

1233 CIM_DiagnosticLog represents a log that aggregates all of the results (records) that the execution of a
 1234 diagnostic generates. Table 40 provides information about the properties of CIM_DiagnosticLog.

1235 **Table 40 – 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 section 8.3.

1236 **10.7 CIM_DiagnosticServiceCapabilities**

1237 CIM_DiagnosticServiceCapabilities publishes the diagnostic service's capabilities, such as settings and
 1238 execution controls that are supported. Table 41 provides information about the properties of
 1239 CIM_DiagnosticServiceCapabilities.

1240 **Table 41 – 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 ".*")
SupportedServiceModes()	Conditional	If service modes are supported, they shall be published using this property.
OtherSupportedServiceModesDescriptions()	Conditional	If "Other" is indicated in the SupportedServiceModes array, this property shall be filled in.
SupportedLoopControl()	Conditional	If looping is supported, its controls shall be published using this property.
OtherSupportedLoopControlDescriptions()	Conditional	If "Other" is indicated in the SupportedLoopControl array, this property shall be filled in.
SupportedLogOptions()	Conditional	If any log options are supported, they shall be published using this property.
OtherSupportedLogOptionsDescriptions()	Conditional	If "Other" is indicated in the SupportedLogOptions array, this property shall be filled in.
SupportedLogStorage()	Conditional	If any log storage options are supported, they shall be published using this property.
OtherSupportedLogStorageDescriptions()	Conditional	If "Other" is indicated in the SupportedLogStorage array, this property shall be filled in.
SupportedExecutionControls()	Conditional	If any execution controls are supported, they shall be published using this property.
OtherSupportedExecutionControls Descriptions()	Conditional	If "Other" is indicated in the SupportedExecutionControls array, this property shall be filled in.

1241 **10.8 CIM_DiagnosticServiceRecord**

1242 CIM_DiagnosticServiceRecord is used to report diagnostic service messages such as results, errors,
 1243 warnings, and status. Table 42 provides information about the properties of
 1244 CIM_DiagnosticServiceRecord.

1245 **Table 42 – 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
LoopsFailed	Mandatory	None
ErrorCode()	Mandatory	This property should be an array that contains only the error code number when the record type is a “device error” or “service error”. If the RecordType value is “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. This property may be NULL if the record type is different than the aforementioned record types. The property will be formatted as a free-form string of variable length. (pattern ".*)")
ErrorCount()	Mandatory	This property should be an array that contains only “1” in the first position of that array when the record type is a “device error” or “service error”. If the RecordType value is “Results”, this property should be an array where each position should contain the number of times that an error (which can be identified by the same position of ErrorCode array) happened. This property may be NULL if record type is different than the aforementioned record types.
ServiceName	Mandatory	This property shall be constructed as follows: <OrgID>:<TestName>. (pattern "^.*[:].*\$")

Properties	Requirement	Notes
ManagedElementName	Mandatory	This property shall be formatted as a free-form string of variable length. (pattern ".*")
RecordType	Mandatory	A RecordType value of "Results" shall be used to log interim results from the diagnostic service or subtest execution. Final results shall use the DiagnosticCompletionRecord class.
OtherRecordTypeDescription	Conditional	If the RecordType value is "Other", this property shall be filled in.
ExpirationDate	Mandatory	See section 7.7.1.

1246 10.9 CIM_DiagnosticSettingData

1247 Diagnostic services use CIM_DiagnosticSettingData to publish default settings, and clients use this class
 1248 to change defaults and run a diagnostic service using specific settings. Table 43 provides information
 1249 about the properties of CIM_DiagnosticSettingData.

1250 **Table 43 – Class: CIM_DiagnosticSettingData**

Properties	Requirement	Notes
InstanceID	Mandatory	Key InstanceID should be constructed using the following preferred algorithm: <OrgID>:<LocalID> (See the MOF file for more detail.) <LocalID> should be set to a time stamp (CIM DateTime). For example: ACME:19980525133015.0000000-300 (pattern "^.*[:].*\$")
ElementName	Mandatory	This property shall be formatted as a free-form string of variable length. (pattern ".*")
HaltOnError	Optional	When this property is TRUE, the service should halt after finding the first error.
QuickMode	Optional	When this property is TRUE, the service should attempt to run in an accelerated fashion either by reducing the coverage or number of tests performed.
PercentOfTestCoverage	Optional	This property requests the service to reduce test coverage to the specified percentage.
LoopControl()	Optional	This property, which is used in conjunction with LoopControlParameter property, sets one or more loop control mechanisms that limit the number of times that a test should be repeated.

Properties	Requirement	Notes
LoopControlParameter()	Conditional	If LoopControl is specified, LoopControlParameter shall be filled in for corresponding LoopControl settings. If LoopControl matches "Count", "Timer", or "ErrorCount", LoopControlParameter represents a uint32 numeric value. (pattern "^b[01]* ^d[0123456789]* ^x[0123456789ABCDEFabcdef]* ^[0123456789]*")
OtherLoopControlDescriptions()	Conditional	If "Other" is indicated in the LoopControl array, this property shall be filled in.
ResultPersistence	Mandatory	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 section 7.7.1.
LogOptions()	Optional	This property specifies the types of data that should be logged by the diagnostic service.
OtherLogOptionsDescriptions()	Conditional	If "Other" is indicated in the LogOptions array, this property shall be filled in.
LogStorage()	Optional	This property specifies the logging mechanism to store the diagnostic results.
OtherLogStorageDescriptions()	Conditional	If "Other" is indicated in the LogStorage array, this property shall be filled in.
VerbosityLevel()	Optional	This property specifies the desired volume or detail logged by a diagnostic service.

1251 **10.10 CIM_DiagnosticSettingDataRecord**

1252 CIM_DiagnosticSettingDataRecord stores the settings used in a specific diagnostic service execution.

1253 Table 44 provides information about the properties of CIM_DiagnosticSettingDataRecord.

1254 **Table 44 – Class: CIM_DiagnosticSettingDataRecord**

Properties	Requirement	Notes
InstanceID	Mandatory	Key InstanceID should be constructed using the following preferred algorithm: <ConcreteJob.InstanceID>:<n> < 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. (pattern "^.*[:].*[:][0123456789]*\$")
CreationTimeStamp	Mandatory	None
ServiceName	Mandatory	This property shall be constructed as follows: <OrgID>:<TestName>. (pattern "^.*[:].*\$")

Properties	Requirement	Notes
ManagedElementName	Mandatory	This property will be formatted as a free-form string of variable length. (pattern ".*")
ExpirationDate	Mandatory	See section 7.7.1.
Settings	Conditional	This property is set to a string that encodes a DiagnosticSettingData instance. If DiagnosticSettingData is implemented, this property shall be supported.

1255 10.11 CIM_DiagnosticTest

1256 CIM_DiagnosticTest is a class that represents a diagnostic service developed to exercise and observe
1257 the behavior of a device that is implicated in some level of system malfunction. It contains properties
1258 useful in test configuration and the RunDiagnosticService() method, a standard mechanism for invoking
1259 the test.

1260 Table 45 provides information about the properties of CIM_DiagnosticTest.

1261

Table 45 – 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 section 7.1.3.
OtherCharacteristicsDescriptions()	Conditional	If "Other" is indicated in the Characteristics array, this property shall be filled in.
RunDiagnosticService()	Mandatory	See section 8.1.

1262 10.12 CIM_ElementCapabilities

1263 CIM_ElementCapabilities associates a diagnostic service with its capabilities. Table 46 provides
1264 information about the properties of CIM_ElementCapabilities.

1265

Table 46 – Class: CIM_ElementCapabilities

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

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

1266 **10.13 CIM_ElementSettingData**

1267 CIM_ElementSettingData associates the diagnostic service with the settings for the service itself and the
1268 resulting job. Table 47 provides information about the properties of CIM_ElementSettingData.

1269 **Table 47 – 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 or DiagnosticSettingData. Cardinality 0..1
IsDefault	Mandatory	None

1270 **10.14 CIM_ElementSoftwareIdentity**

1271 CIM_ElementSoftwareIdentity associates the diagnostic service with its version information. Table 48
1272 provides information about the properties of CIM_ElementSoftwareIdentity.

1273 **Table 48 – Class: CIM_ElementSoftwareIdentity**

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

1274 **10.15 CIM_HelpService**

1275 CIM_HelpService is the preferred way for a service to publish online help information. Table 49 provides
1276 information about the properties of CIM_HelpService.

1277 **Table 49 – Class: CIM_HelpService**

Properties	Requirement	Notes
SystemCreationClassName	Mandatory	Key
SystemName	Mandatory	Key
CreationClassName	Mandatory	Key

Properties	Requirement	Notes
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 "Other" is indicated in the DeliveryOptions array, this property shall be filled in.
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 "Other" is indicated in the DocumentFormat array, this property shall be filled in.
GetHelp()	Mandatory	See section 8.4.

1278 10.16 CIM_HostedService

1279 CIM_HostedService is used to associate an instance of CIM_DiagnosticTest with an instance of
 1280 CIM_ComputerSystem to which the CIM_DiagnosticTest is scoped and to associate an instance of
 1281 CIM_HelpService with an instance of CIM_ComputerSystem to which the CIM_HelpService is scoped..
 1282 Table 50 provides information about the properties of CIM_HostedService.

1283 **Table 50 – Class: CIM_HostedService**

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

1284 10.17 CIM_JobSettingData

1285 Diagnostic services use CIM_JobSettingData to publish default job settings (for the jobs that they launch),
 1286 and clients use this class to change the default job settings when invoking the RunDiagnosticService()
 1287 method. Table 51 provides information about the properties of CIM_JobSettingData.

1288 **Table 51 – Class: CIM_JobSettingData**

Properties	Requirement	Notes
ElementName	Mandatory	This property shall be formatted as a free-form string of variable length. (pattern ".**")

Properties	Requirement	Notes
DeleteOnCompletion	Conditional	This property indicates whether the job should be automatically deleted upon completion. The CIM_ConcreteJob.TimeBeforeRemoval property has priority over this property. If CIM_JobSettingData is supported, this property shall be filled in.

1289 **10.18 CIM_LogManagesRecord**

1290 CIM_LogManagesRecord associates a log with its records (service records, setting records, or
1291 completion records). Table 52 provides information about the properties of CIM_LogManagesRecord.

1292 **Table 52 – Class: CIM_LogManagesRecord**

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

1293 **10.19 CIM_OwningJobElement**

1294 CIM_OwningJobElement associate a diagnostic service with its jobs (jobs that are launched by this
1295 diagnostic). Table 53 provides information about the properties of CIM_OwningJobElement.

1296 **Table 53 – Class: CIM_OwningJobElement**

Properties	Requirement	Notes
OwningElement	Mandatory	Key This property shall be a reference to an instance of CIM_DiagnosticService. Cardinality 1
OwnedElement	Mandatory	Key This property shall be a reference to an instance of CIM_ConcreteJob.

1297 **10.20 CIM_RecordAppliesToElement**

1298 CIM_RecordAppliesToElement associates a record with the managed elements (diagnostic service and
1299 device) that have a relationship with this record. Table 54 provides information about the properties of
1300 CIM_RecordAppliesToElement.

1301 **Table 54 – 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.

1302 **10.21 CIM_RegisteredProfile**

1303 CIM_RegisteredProfile identifies the *Diagnostics Profile* in order for a client to determine whether an
 1304 instance of CIM_DiagnosticService is conformant with this profile. The CIM_RegisteredProfile class is
 1305 defined by the *Profile Registration Profile*. With the exception of the mandatory values specified in Table
 1306 55, the behavior of the CIM_RegisteredProfile instance is in accordance with the *Profile Registration*
 1307 *Profile*.

1308 **Table 55 – 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".
OwningEntity	Mandatory	This property shall have a value of "DMTF".

1309 **10.22 CIM_ServiceAffectsElement**

1310 CIM_ServiceAffectsElement is used to associate to the diagnostic service any managed elements that
 1311 are affected by the running of the service. Table 56 provides information about the properties of
 1312 CIM_ServiceAffectsElement.

1313 **Table 56 – 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.

1314 **10.23 CIM_ServiceAvailableToElement**

1315 CIM_ServiceAvailableToElement associates the diagnostic service with its help service information. Table
 1316 57 provides information about the properties of CIM_ServiceAvailableToElement.

1317 **Table 57 – 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

1318 **10.24 CIM_ServiceComponent**

1319 CIM_ServiceComponent associates a test that is also part of another test. Table 58 provides information
 1320 about the properties of CIM_ServiceComponent.

1321 **Table 58 – 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.

1322 **10.25 CIM_SoftwareIdentity**

1323 CIM_SoftwareIdentity is used to publish version information about the diagnostic service. Table 59
 1324 provides information about the properties of CIM_SoftwareIdentity.

1325 **Table 59 – 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 ".*")

1326 **10.26 CIM_UseOfLog**

1327 CIM_UseOfLog associates a log with a managed element (a device or diagnostic service) whose
 1328 information is stored in the log. Table 60 provides information about the properties of CIM_UseOfLog.

1329 **Table 60 – Class: CIM_UseOfLog**

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

ANNEX A (informative)

Change Log

Version	Date	Authors	Description
0.1	07/29/04	Ray Pedersen	Document created.
0.3.5	09/23/05	Mateus Baur	Updated sections 2.4 and 2.7.
0.3.6	10/12/05	Ray Pedersen	Incorporated team comments.
0.3.7	10/20/05	Mateus Baur	Modified the sections that mention DiagnosticLog as recommended because it is required now.
0.3.8	11/15/05	Ray Pedersen	Cleaned up and releveled to 2.11 final.
0.7	11/16/05	Ray Pedersen	Cleaned up and added use cases.
0.8	11/23/05	Mateus Baur	Added DiagnosticSettingData information and updated the "Use Case" section.
0.8.1	11/30/05	Mateus Baur	Updated "Use Case" section.
0.9.2	12/07/05	Ray Pedersen	Cleaned up for CORE Ballot.
0.9.3	02/01/06	Ray Pedersen	Responded to ballot comments.
0.9.4	02/07/06	Mateus Baur	Updated section 8.5 to address ballot comments.
0.9.6	02/07/06	Ray Pedersen Mateus Baur Barbara Craig	Added Regular Expressions, performed final cleanup for ballot 2.
0.9.7	02/28/06	Ray Pedersen Mateus Baur Barbara Craig	Entered ballot 2 comments.
1.0.0a	04/17/06		Formatted to ISO template requirements.

ANNEX B
(informative)

1334
1335
1336
1337
1338

Acknowledgements

1339 The authors wish to acknowledge the following people.

1340 Editors:

- 1341 • Ray Pedersen – IBM Corporation
- 1342 • Mateus Baur – Hewlett-Packard Company
- 1343 • Barbara Craig – Hewlett-Packard Company

1344 Contributors:

- 1345 • Aaron Merkin – IBM Corporation
- 1346 • Jon Hass – Dell Inc.
- 1347 • Members of the DMTF Diagnostics SIG (sub-Working Group of CIM Core)
- 1348