Document Number: DSP1002

Date: 2010-10-21

Version: 2.0.0

Diagnostics Profile

Document Type: Specification

Document Status: DMTF Standard

Document Language: en-US

CONTENTS

Figures

Tables

192

3.1 Foreword

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

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

Acknowledgments

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

Editors:

- Barbara Craig – Hewlett-Packard Company
- Carl Chan – WBEM Solutions, Inc.
- Jim Davis – WBEM Solutions, Inc.
- Mateus Baur – Hewlett-Packard Company
- Ray Pedersen – IBM Corporation

Contributors:

- Aaron Merkin – IBM Corporation
- Carl Chan – WBEM Solutions, Inc.
- Dave Barrett – Emulex
- Eric Tend – Hewlett-Packard Company
- Jon Hass – Dell Inc.
- Ken Kotyuk – Hewlett-Packard Company
- Kevin Kuelbs – Hewlett-Packard Company
- Rodney Brown – IBM Corporation

3.2 Introduction

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

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

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

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

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

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

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

245 # Diagnostics Profile

246 ## 1 Scope

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

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

253 ## 2 Normative References

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

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

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

262 DMTF DSP0215, *SM Managed Element Addressing Specification (SM ME Addressing) 1.0*,
263 http://www.dmtf.org/standards/published_documents/DSP0215_1.0.pdf

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

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

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

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

3 Terms and Definitions

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

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

The terms "clause," "subclause," "paragraph," and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 5.

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

The terms defined in DSP0004, DSP0200, DSP1001, and DSP1033 apply to this document. For the purposes of this document, the following terms and definitions also apply.

3.1
conditional
indicates requirements to be followed strictly in order to conform to the document when the specified CIM testable conditions are met

3.2
mandatory
indicates requirements to be followed strictly in order to conform to the document and from which no deviation is permitted

3.3
optional
indicates a course of action permissible within the limits of the document

4 Symbols and Abbreviated Terms

The following abbreviations are used in this document.

4.1
CDM
Common Diagnostic Model

4.2
CIM
Common Information Model

4.3
CIMOM
CIM Object Manager

4.4
CRU
Customer Replaceable Unit

4.5
FRU
Field Replaceable Unit

4.6
ME
Managed Element

4.7
MOF
Managed Object Format

4.8
PD
Problem Determination

4.9
PFA
Predictive Failure Analysis

4.10
SAN
Storage Area Network

4.11
WBEM
Web-Based Enterprise Management

5 Synopsis

Profile Name: Diagnostics Profile

Version: 2.0.0

Organization: DMTF

CIM schema version: 2.23

Central Class: CIM_DiagnosticTest

Scoping Class: CIM_ComputerSystem

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

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

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

Table 1 – Related Profiles

Profile Name	Organization	Version	Relationship	Behavior

Profile Name	Organization	Version	Relationship	Behavior
Profile Registration	DMTF	1.0	Mandatory	

6 Description

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

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

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

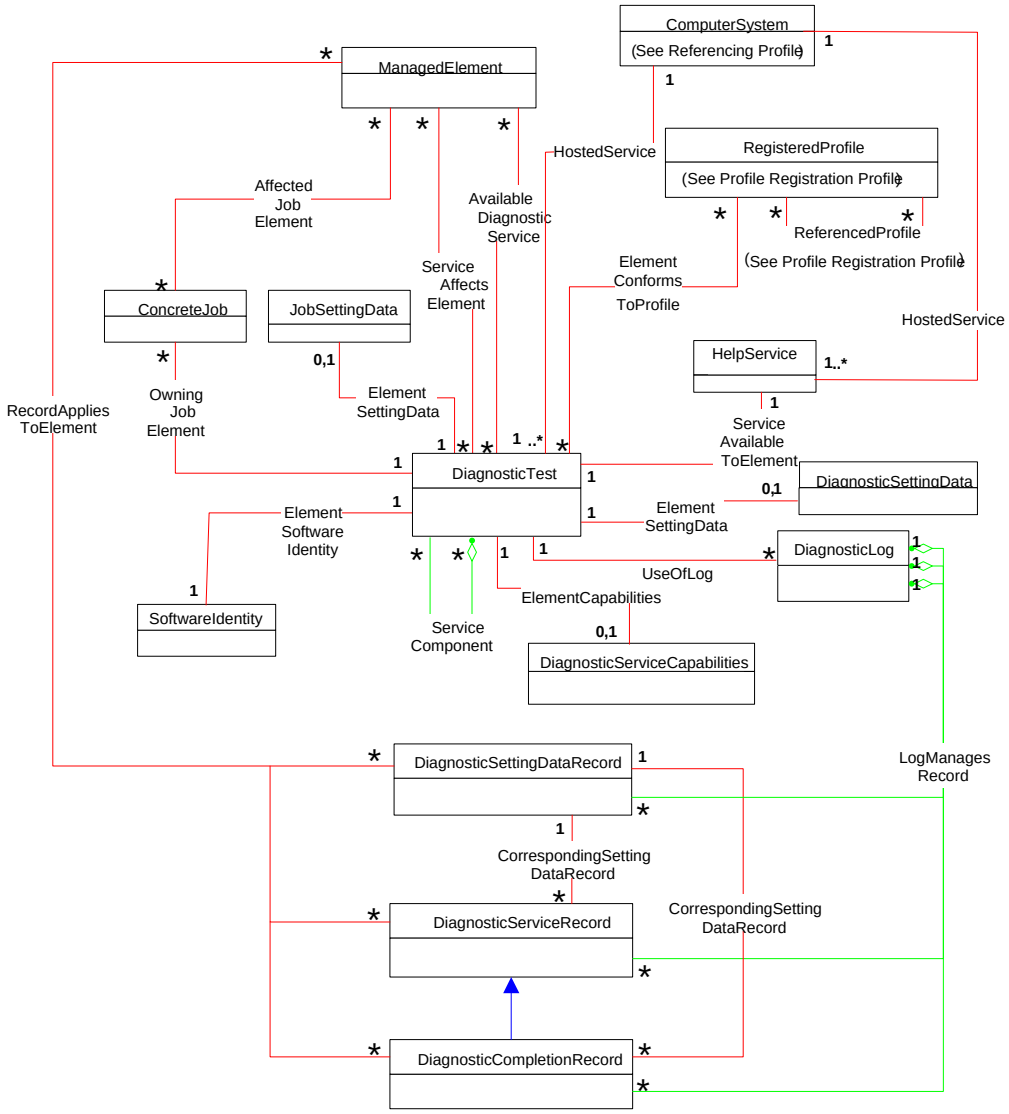


Figure 1 – Diagnostics Profile: Class Diagram

7 Implementation

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

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

7.1 CIM_DiagnosticTest

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

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

An implementation may use

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

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

7.1.1 CIM_DiagnosticTest.Name

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

7.1.2 CIM_DiagnosticTest.ElementName

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

7.1.3 CIM_DiagnosticTest.Characteristics

This clause defines the values of the Characteristics property.

7.1.3.1 Is Exclusive (value=2)

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

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

7.1.3.2 Is Interactive (value=3)

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

7.1.3.3 Is Destructive (value=4)

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

7.1.3.4 Is Risky (value=5)

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

7.1.3.5 Is Package (value=6)

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

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

7.1.3.6 Reserved (value=7)

This value originally contained “Supports PercentOfTestCoverage”, which was deprecated and added to the CIM_DiagnosticServiceCapabilities class.

7.1.3.7 Is Synchronous (value=8)

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

7.1.3.8 Media Required (value=9)

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

7.1.3.9 Additional Hardware Required (value=10)

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

7.1.4 Looping Tests

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

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

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

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

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

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

7.1.5 Test Effectiveness

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

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

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

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

7.2 CIM_AvailableDiagnosticService

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

7.2.1 CIM_AvailableDiagnosticService.EstimatedDurationOfService

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

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

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

7.2.2 CIM_AvailableDiagnosticService.EstimatedDurationQualifier

The EstimatedDurationQualifier property allows for more accurate quantification of the value specified for the EstimatedDurationOfService property. For example, if EstimatedDurationOfService has the value 2

(Seconds) and EstimatedDurationQualifier has a value of 20, then the service has an estimated duration of 20 seconds. This property should be implemented if further quantification is possible. In contrast, if EstimatedDurationOfService has the value 0 (Unknown), then EstimatedDurationQualifier may be NULL.

7.3 CIM_DiagnosticServiceCapabilities

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

7.4 CIM_DiagnosticSettingData

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

NOTE: A CIM_DiagnosticSettingData object shall not contain any values that conflict with the diagnostic service capabilities as indicated by its CIM_DiagnosticServiceCapabilities object. For example, if CIM_DiagnosticServiceCapabilities.SupportedLoopControl includes the value 5 (No Loop Control), then CIM_DiagnosticSettingData.LoopControl cannot include the value 3 (Count). Conflicting values shall be ignored by the implementation.

7.4.1 Default Setting

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

7.4.2 Client Override

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

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

If a client choose to override default settings, the Setting argument to the RunDiagnosticService( ) method of DiagnosticTest is set to an encoded form of the CIM_DiagnosticSettingData object.

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

7.5 CIM_ConcreteJob

This clause defines the properties of the CIM_ConcreteJob class. Each execution of a test will create an instance of CIM_ConcreteJob so that a client can track the progress and control the execution of the executing diagnostic.

7.5.1 CIM_ConcreteJob.TimeBeforeRemoval

This property represents the amount of time that must elapse before the CIM_ConcreteJob object can be deleted after the job has terminated. A default time is defined in the MOF class definition.

To determine the time at which the CIM_ConcreteJob object can be deleted, one must first calculate the Completion Time. The algorithm is as follows:

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

The CIM_ConcreteJob object may be deleted at Completion Time + TimeBeforeRemoval.

7.5.2 CIM_ConcreteJob.PercentComplete

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

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

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

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

7.5.3 CIM_ConcreteJob.InstanceID

CIM_ConcreteJob.InstanceID should be constructed using the following preferred algorithm:

<OrgID>:<LocalID>

where <OrgID> identifies the business entity (for example, ACME) and <LocalID> is a value that uniquely identifies each ConcreteJob instance that is launched on a system when a test is executed. See the MOF file description for further information.

The purpose for <LocalID> is to provide some form of uniqueness within the context of running separate diagnostic tests over a period of time for the domain of the test execution (whether just the local system or several remote systems). In practice, <LocalID> could be an incremented counter or a timestamp in combination with other test identifiers or factors.

A unique <LocalID> allows a user to easily retrieve test results from the diagnostic log for a specific test execution because the InstanceID values of CIM_ConcreteJob and the subclasses of CIM_DiagnosticRecord are closely related.

7.6 CIM_DiagnosticLog

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

7.6.1 Logging Results

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

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

567 • The heterogeneous nature of the log entries more easily fits into a self-describing record
568 paradigm.

569 • Keyed records are easier to manage and retrieve.

570 7.7 CIM_DiagnosticRecord

571 CIM_DiagnosticRecord has two subclasses: CIM_DiagnosticServiceRecord and
572 CIM_DiagnosticSettingDataRecord. CIM_DiagnosticServiceRecord has a single subclass:
573 CIM_DiagnosticCompletionRecord.

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

577 CIM_DiagnosticSettingDataRecord is structured to hold the attributes of the setting object that was used
578 as the DiagSetting parameter to the RunDiagnosticService( ) method. At most, a single
579 CIM_DiagnosticSettingDataRecord may be created during a single execution of a test.

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

583 7.7.1 CIM_DiagnosticRecord.ExpirationDate

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

587 CIM_DiagnosticSettingData.ResultPersistence shall be used by the client to specify to the diagnostic
588 service implementation how long the results generated by that service shall persist. A value shall be
589 chosen that allows the minimum time needed by the client to record the data. When the timeout value has
590 been reached, the implementation shall delete the data objects that contain the results.

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

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

599 7.7.2 CIM_DiagnosticRecord.InstanceID

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

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

603 <ConcreteJob.InstanceID>:<n>

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

7.8 CIM_ServiceComponent

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

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

8 Methods

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

8.1 CIM_DiagnosticService.RunDiagnosticService( ) Extrinsic Method

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

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

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

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

Table 2 – RunDiagnosticService( ) Method: Return Code Values

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

638 **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

639 ## 8.2 CIM_ConcreteJob.RequestStateChange( ) Extrinsic Method

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

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

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

650 **Table 4 – RequestStateChange( ) Method: Return Code Values**

Value	Description
0	Completed with No Error
2	Unknown/Unspecified Error
3	Cannot complete within Timeout Period
4	Failed
5	Invalid Parameter
6	In Use
4096	Method parameters checked — transition started
4097	Invalid state transition
4098	Use of timeout parameter not supported
4099	Busy — indicates that the method cannot be invoked "at this time." It is not an error condition, but signals that the implementation is doing something else and cannot respond."
32768..65535	Vendor specific

651 **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.

652 ## 8.3 CIM_Log.ClearLog( ) Extrinsic Method

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

656 ClearLog return values are specified in Table 6.

657 **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
0x8000..0xFFFF	Vendor specific

658 ## 8.4 CIM_HelpService.GetHelp( ) Extrinsic Method

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

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

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

666 **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
0x1000	Busy — indicates that the method cannot be invoked "at this time." It is not an error condition, but signals that the implementation is doing something else and cannot respond.
0x1001	Requested document not found
0x8000..0xFFFF	Vendor Reserved

667 **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.

668 ## 8.5 Profile Conventions for Operations

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

671 The default list of operations is as follows:

672 • GetInstance
673 • EnumerateInstances
674 • EnumerateInstanceNames
675 • Associators
676 • AssociatorNames
677 • References
678 • ReferenceNames

8.6 CIM_DiagnosticTest

Table 9 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 9, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

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

8.7 CIM_AvailableDiagnosticService

Table 10 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 10, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

Table 10 – Operations: CIM_AvailableDiagnosticService

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

8.8 CIM_ServiceAffectsElement

Table 11 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 11, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

Table 11 – Operations: CIM_ServiceAffectsElement

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

8.9 CIM_SoftwareIdentity

Table 12 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 12, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

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

8.10 CIM_ElementSoftwareIdentity

Table 13 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 13, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

Table 13 – Operations: CIM_ElementSoftwareIdentity

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

8.11 CIM_HelpService

Table 14 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 14, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

714 **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

715 ## 8.12 CIM_ServiceAvailableToElement

716 Table 15 lists implementation requirements for operations. If implemented, these operations shall be
717 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 15, all operations
718 in the default list in 8.5 shall be implemented as defined in DSP0200.

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

720 **Table 15 – Operations: CIM_ServiceAvailableToElement**

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

721 ## 8.13 CIM_DiagnosticSettingData

722 Table 16 lists implementation requirements for operations. If implemented, these operations shall be
723 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 16, all operations
724 in the default list in 8.5 shall be implemented as defined in DSP0200.

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

726 **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

8.14 CIM_DiagnosticServiceCapabilities

Table 17 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 17, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

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

8.15 CIM_ElementCapabilities

Table 18 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 18, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

Table 18 – Operations: CIM_ElementCapabilities

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

8.16 CIM_ConcreteJob

Table 19 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 19, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

744 **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

745 ## 8.17 CIM_OwningJobElement

746 Table 20 lists implementation requirements for operations. If implemented, these operations shall be
747 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 20, all operations
748 in the default list in 8.5 shall be implemented as defined in DSP0200.

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

750 **Table 20 – Operations: CIM_OwningJobElement**

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

751 ## 8.18 CIM_AffectedJobElement

752 Table 21 lists implementation requirements for operations. If implemented, these operations shall be
753 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 21, all operations
754 in the default list in 8.5 shall be implemented as defined in DSP0200.

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

756 **Table 21 – Operations: CIM_AffectedJobElement**

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

8.19 CIM_JobSettingData

Table 22 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 22, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

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

8.20 CIM_ElementSettingData

Table 23 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 23, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

Table 23 – Operations: CIM_ElementSettingData

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

8.21 CIM_DiagnosticLog

Table 24 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 24, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

774 **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

775 ## 8.22 CIM_UseOfLog

776 Table 25 lists implementation requirements for operations. If implemented, these operations shall be
777 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 25, all operations
778 in the default list in 8.5 shall be implemented as defined in DSP0200.

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

780 **Table 25 – Operations: CIM_UseOfLog**

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

781 ## 8.23 CIM_DiagnosticServiceRecord

782 Table 26 lists implementation requirements for operations. If implemented, these operations shall be
783 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 26, all operations
784 in the default list in 8.5 shall be implemented as defined in DSP0200.

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

786 **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

787 ## 8.24 CIM_DiagnosticCompletionRecord

788 Table 27 lists implementation requirements for operations. If implemented, these operations shall be
789 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 27, all operations
790 in the default list in 8.5 shall be implemented as defined in DSP0200.

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

792 **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

793 ## 8.25 CIM_DiagnosticSettingDataRecord

794 Table 28 lists implementation requirements for operations. If implemented, these operations shall be
795 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 28, all operations
796 in the default list in 8.5 shall be implemented as defined in DSP0200.

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

798 **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

799 ## 8.26 CIM_LogManagesRecord

800 Table 29 lists implementation requirements for operations. If implemented, these operations shall be
801 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 29, all operations
802 in the default list in 8.5 shall be implemented as defined in DSP0200.

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

804 **Table 29 – Operations: CIM_LogManagesRecord**

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

805 ## 8.27 CIM_RecordAppliesToElement

806 Table 30 lists implementation requirements for operations. If implemented, these operations shall be
807 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 30, all operations
808 in the default list in 8.5 shall be implemented as defined in DSP0200.

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

810 **Table 30 – Operations: CIM_RecordAppliesToElement**

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

8.28 CIM_CorrespondingSettingDataRecord

Table 31 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 31, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

Table 31 – Operations: CIM_CorrespondingSettingDataRecord

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

8.29 CIM_ServiceComponent

Table 32 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 32, all operations in the default list in 8.5 shall be implemented as defined in DSP0200.

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

Table 32 – Operations: CIM_ServiceComponent

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

9 Use Cases

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

9.1 Profile Conformance

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

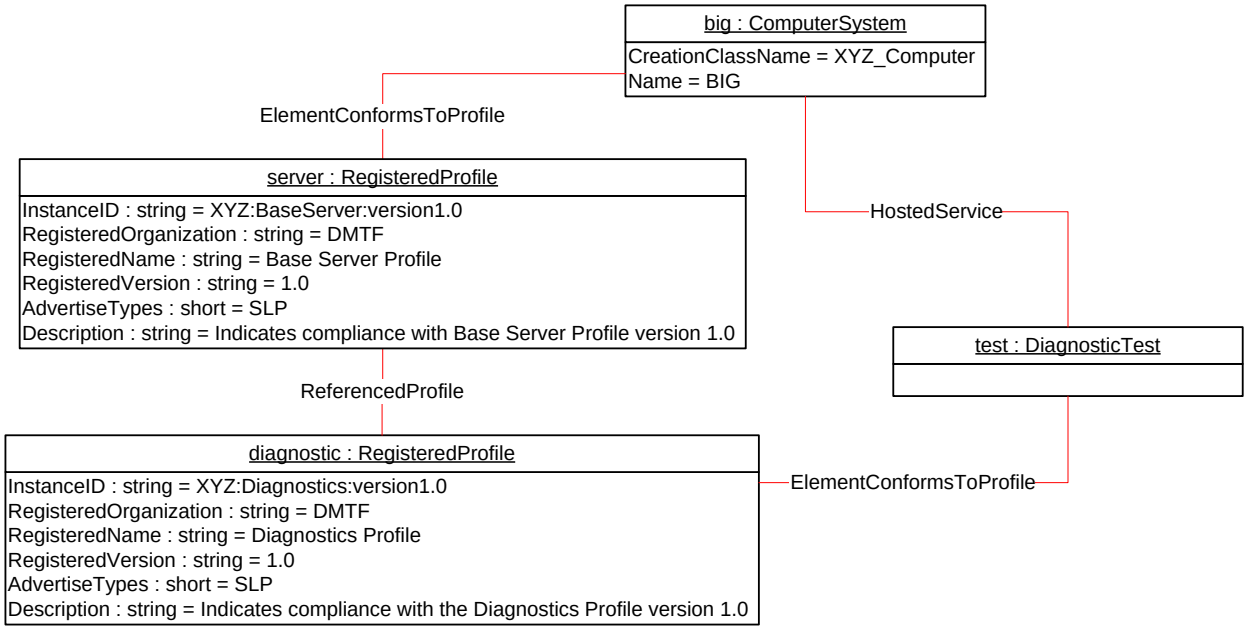
With CIM_ComputerSystem as the Scoping Class of this profile, the object diagram in Figure 2 shows how instances of CIM_RegisteredProfile may be used to identify the version of the *Diagnostics Profile* to which an instance of CIM_DiagnosticTest and its associated instances conform. In this example (using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

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

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

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

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



850

851 **Figure 2 – Registered Profile**

852 9.2 Use Case Summary

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

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

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

858 **Table 33 – Diagnostics Profile Use Cases**

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

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

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

9.3 Diagnostic Services Object Diagram

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

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

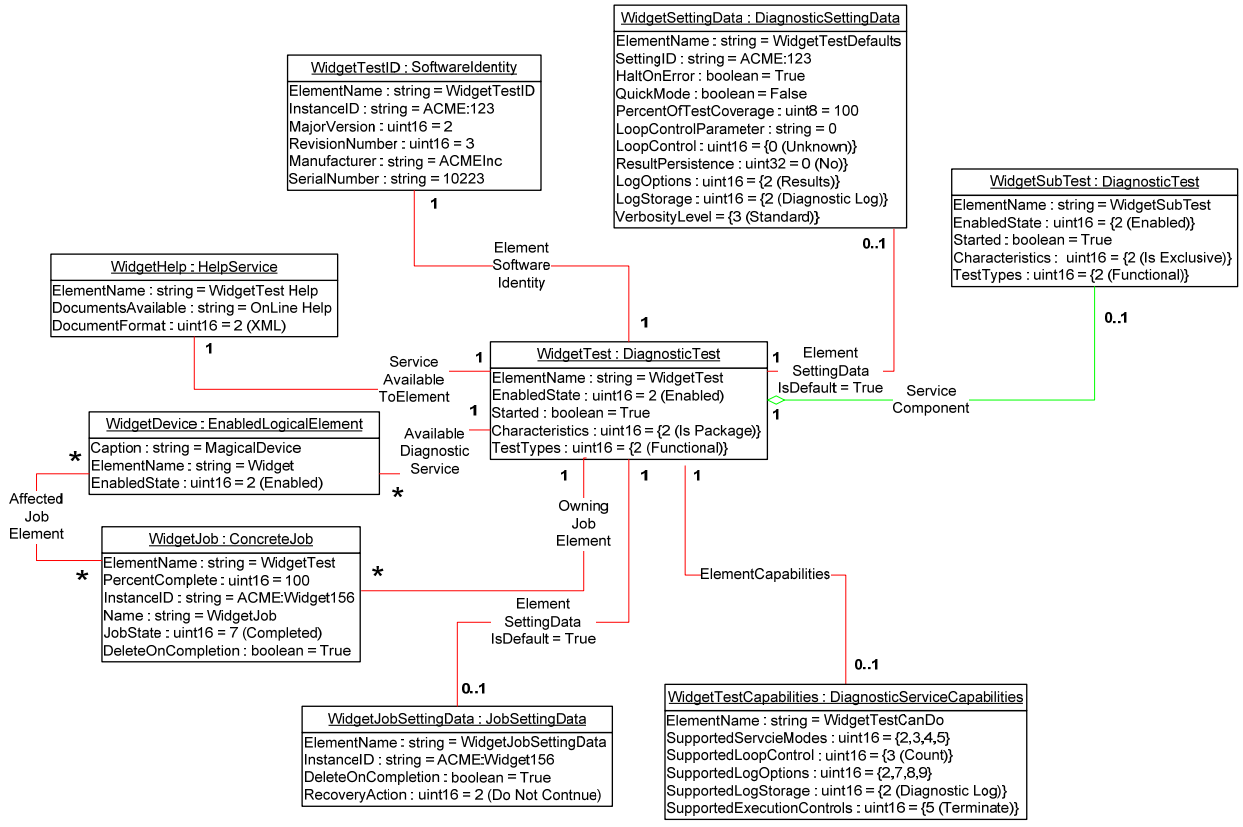


Figure 3 – Diagnostic Services Object Diagram

9.4 Discover Available Diagnostics

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

9.4.1 GetAllDiagnostics

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

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

9.4.2 GetAllDiagnosticMEPairs

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

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

9.4.3 GetDiagnosticsForME

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

1) From the ManagedElement instance, the client calls the Associators operation using AvailableDiagnosticService as the association class.
2) The operation returns DiagnosticTest instances that represent a diagnostic that can be launched against the ManagedElement.

9.4.4 GetMEsForDiagnostic

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

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

9.4.5 GetCapabilitiesOfDiagnostic

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

1) From the DiagnosticTest instance, the client calls the Associators operation using ElementCapabilities as the association class and DiagnosticServiceCapabilities as the result class.

2) The operation should return only one DiagnosticServiceCapabilities instance, which represents the diagnostic capabilities.

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

9.4.6 GetCharacteristicsOfDiagnostic

The client can discover all of the characteristics (is destructive, is interactive, is synchronous, and so on) of a diagnostic. From the DiagnosticTest instance, the client reads just the Characteristics and OtherCharacteristicsDescriptions attributes, which contain the diagnostic characteristics. See the MOF file class definition for DiagnosticTest for further information.

9.4.7 GetDiagnosticsWithCharacteristicsForME

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

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

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

3) If the characteristics of the DiagnosticTest instance match the desired characteristics, the DiagnosticTest instance is the one desired.

9.4.8 GetDiagnosticsWithCapabilitiesForME

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

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

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

    2) If the capabilities of the DiagnosticTest instance match the desired capabilities, the DiagnosticTest instance is the one desired.

9.4.9 GetPackageSubtests

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

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

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

    2) The operation returns the DiagnosticTest instances that are subtests of the known DiagnosticTest.

9.5 Configure Diagnostic

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

9.5.1 GetDefaultDiagnosticSettings

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

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

    1) For each DiagnosticSettingData instance, the client calls the References operation using ElementSettingData as the result class. The operation returns ElementSettingData instances.

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

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

9.5.2 CreateDiagnosticSettings

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

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

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

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

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

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

9.5.3 GetDefaultJobSettings

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

1) From the DiagnosticTest instance, the client calls the Associators operation using ElementSettingData as the association class and JobSettingData as the result class. The operation returns JobSettingData instances.

1) For each JobSettingData instance, the client calls the References operation using ElementSettingData as the result class. The operation returns ElementSettingData instances.

2) For each ElementSettingData instance, the client determines whether the value of the ElementSettingData.ManagedElement property matches the DiagnosticTest name and the value of the ElementSettingData.IsDefault property is 1 (“Is Default”). If so, the JobSettingData instance represents the default job settings. The name of this JobSettingData instance may also be retrieved from ElementSettingData.SettingData property.

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

9.5.4 CreateJobSettings

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

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

1) The client can attempt to discover the default job settings of the DiagnosticTest instance. The GetDefaultJobSettings use case (see 9.5.3) describes the necessary steps.

    1) If Step 1 does not return an instance or if the client chooses to create an instance of the JobSettingData class, a GetClass operation for JobSettingData can be performed and then used to create an instance locally in the client scope (for example, IwbemClassObject or CIMInstance object) based on the class definition.

    2) The client modifies the created JobSettingData instance as necessary.

9.6 Execute and Control Diagnostic

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

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

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

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

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

NOTE: Not all diagnostic tests are capable of running on the same device simultaneously; that is, DiagnosticTest.Characteristics has the value of 2 (Is Exclusive). If this had been the case in this example, the

DiagnosticTest would have returned an error on the second RunDiagnosticService() method call to run a test on ManagedElement2.

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

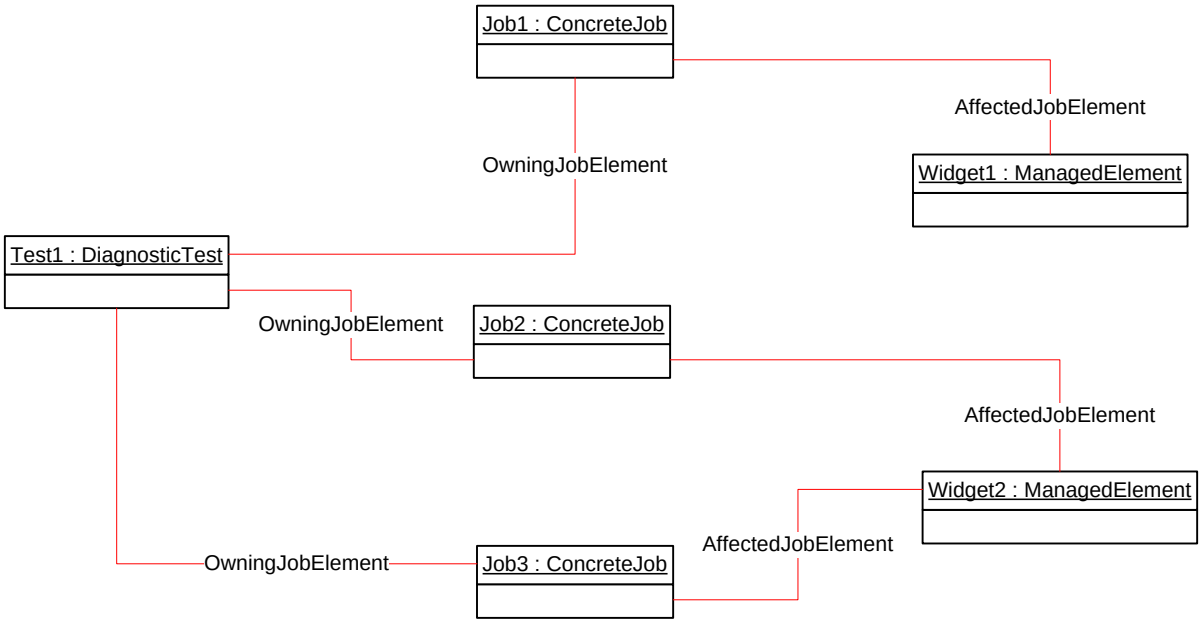


Figure 4 – Job Example

9.6.1 RunDiagnostic

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

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

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

9.6.2 SuspendDiagnostic

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

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

    1) The client checks the DiagnosticServiceCapabilities.SupportedExecutionControls() property for the value of 4 (“Suspend Job”). If the value exists, the Job supports suspension.

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

3) The client calls the RequestStateChange() method, passing in a RequestedState value of 3 (“Suspend”).

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

9.6.3 ResumeDiagnostic

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

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

    1) The client checks the DiagnosticServiceCapabilities.SupportedExecutionControls() property for the value of 4 (“Suspend Job”). If the value exists, the Job supports resumption.

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

    3) The client calls the RequestStateChange() method of DiagnosticTest, passing in a RequestedState value of 2 (“Enabled”).

    4) When the transition completes successfully, the ConcreteJob that represents the test will set the value of the JobState property to 4 (“Running”) and set the value of TimeOfLastStateChange to the current time.

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

9.6.4 AbortDiagnostic

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

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

    1) The client checks the DiagnosticServiceCapabilities.SupportedExecutionControls() property for the value of 5 (“Terminate Job”). If the value exists, the Job supports termination.

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

    3) The client calls the RequestStateChange() method, passing in a RequestedState value of 4 (“Terminate”).

    4) When the transition completes successfully, the ConcreteJob that represents the test will set the value of the JobState property to 8 (“Terminated”) and set the value of TimeOfLastStateChange to the current time.

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

9.6.5 KillDiagnostic

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

1) The client follows the ElementCapabilities association from the DiagnosticTest to the DiagnosticServiceCapabilities for the service.
    1) The client checks the DiagnosticServiceCapabilities.SupportedExecutionControls() property for the value of 3 (“Kill Job”). If the value exists, the Job supports kill.
    2) The client finds the appropriate Job instances. The GetSpecificDiagnosticExecutions use case (see 9.7.3) describes the necessary steps.
    3) The client calls the RequestStateChange() method, passing in a RequestedState value of 5 (“Kill”).
    4) When the transition completes successfully, the ConcreteJob that represents the test will set the value of the JobState property to 9 (“Killed”) and set the value of TimeOfLastStateChange to the current time.

9.7 Discover Diagnostic Executions

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

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

9.7.1 GetAffectedMEs

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

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

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

9.7.2 GetAllDiagnosticExecutionsForME

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

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

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

9.7.3 GetSpecificDiagnosticExecutions

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

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

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

9.7.4 GetSpecificDiagnosticExecutionsForME

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

starting at the known ManagedElement instance

starting at the known DiagnosticTest instance

9.7.4.1 Starting at the Managed Element

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

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

1) From the ManagedElement instance, the client calls the Associators operation using AffectedJobElement as the association class and ConcreteJob as the result class. The operation returns the ConcreteJob instances that are running against this ManagedElement.

    1) For each ConcreteJob instance, the client calls the AssociatorNames operation using OwningJobElement as the association class and DiagnosticTest as the result class. The operation returns the instance paths to the DiagnosticTest instances that launched the ConcreteJob instances against this ManagedElement.

    2) For each DiagnosticTest instance path returned, the client determines if it is the instance path of the known DiagnosticTest instance. If the instance path matches, the ConcreteJob instance represents an execution of that diagnostic service on that ManagedElement.

9.7.4.2 Starting at the DiagnosticTest

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

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

1) From the DiagnosticTest instance, the client calls the Associators operation using OwningJobElement as the association class and ConcreteJob as the result class. The operation returns the ConcreteJob instances launched by the DiagnosticTest.

    1) For each ConcreteJob instance, the client calls the AssociatorNames operation using AffectedJobElement as the association class and ManagedElement as the result class. The operation returns the instance paths to the ManagedElement instances against which this DiagnosticTest launched the ConcreteJob instances.

    2) For each ManagedElement instance path returned, the client determines if it is the instance path of the known ManagedElement instance. If the instance path matches, the

ConcreteJob instance represents an execution of that diagnostic service on that ManagedElement.

9.8 Discover Diagnostic Results (In Progress and Final)

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

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

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

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

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

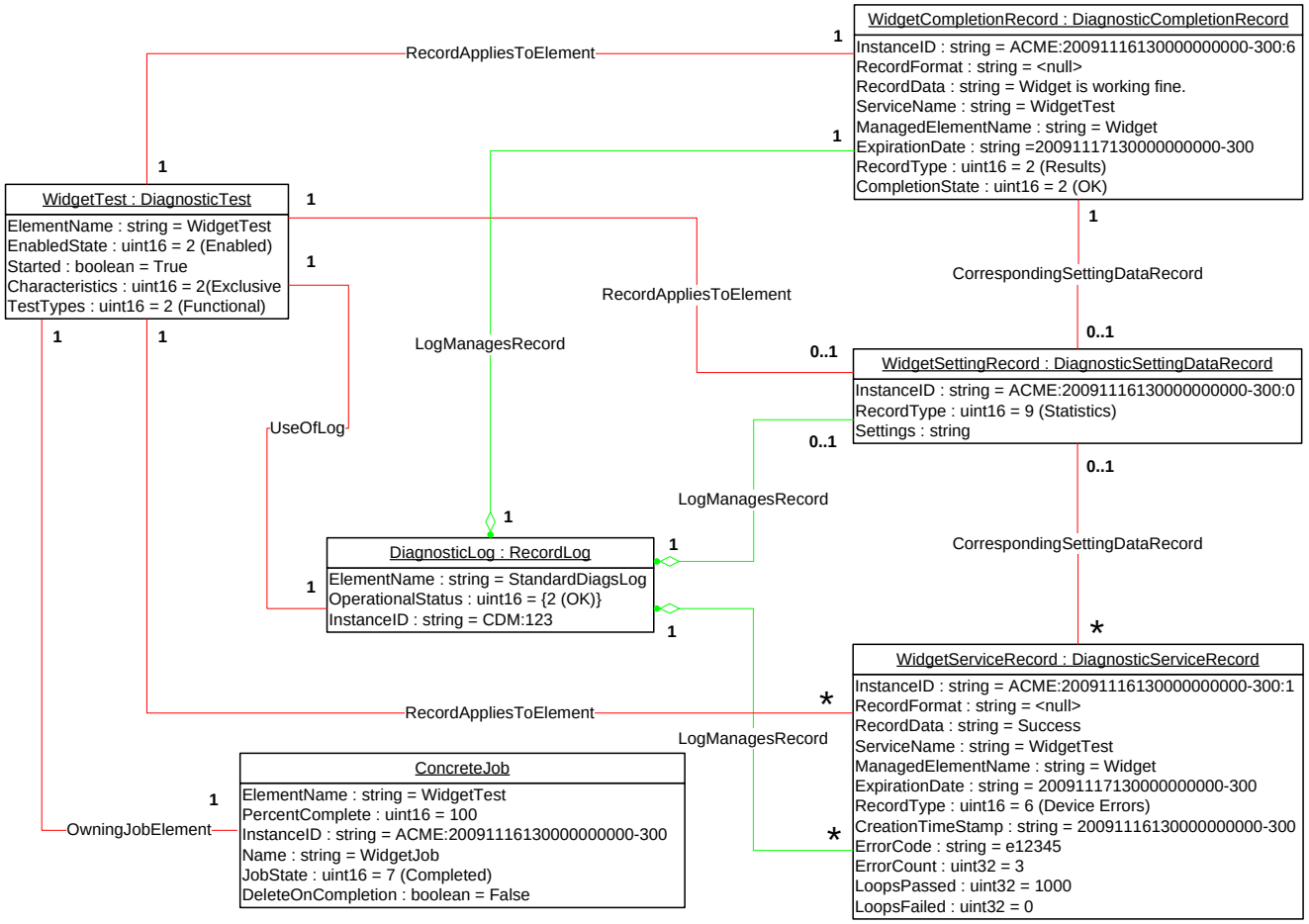


Figure 5 – Diagnostic Logging Object Diagram

9.8.1 GetLogRecordsForDiagnostic

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

1) The client calls the ExecQuery operation as follows:

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

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

An alternate method without using ExecQuery is as follows:

Assume that the client starts at the known DiagnosticTest instance.

1) From the DiagnosticTest instance, the client calls the Associators operation using UseOfLog as the association class and DiagnosticsLog as the result class. The operation returns the DiagnosticLog instances that contain records for the DiagnosticTest.

   1) For each DiagnosticLog instance, the client calls the Associators operation using LogManagesRecord as the association class and DiagnosticRecord as the result class. The operation returns the DiagnosticRecord instances in the Log.

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

9.8.2 GetLogRecordsForME

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

1) The client calls the ExecQuery operation as follows:

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

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

An alternate method without using ExecQuery is as follows:

Assume that the client starts at the known ManagedElement instance.

1) From the ManagedElement instance, the client calls the Associators operation using ServiceAvailableToElement as the association class and DiagnosticTest as the result class. The operation returns the DiagnosticTest instances for the ManagedElement.

   1) For each DiagnosticTest instance, the client calls the Associators operation using UseOfLog as the association class and DiagnosticLog as the result class. The operation returns the DiagnosticLog instances that contain records for the DiagnosticTest.

   2) For each DiagnosticLog instance, the client calls the Associators operation using LogManagesRecord as the association class and DiagnosticRecord as the result class. The operation returns the DiagnosticRecord instances in the Log.

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

9.8.3 GetLogRecordsForMEAndDiagnostic

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

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

1) The client calls the ExecQuery operation as follows:

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

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

An alternate method without using ExecQuery is as follows:

Assume that the client starts at the known DiagnosticTest instance.

1) From the DiagnosticTest instance, the client calls the Associators operation using UseOfLog as the association class and DiagnosticLog as the result class. The operation returns the DiagnosticLog instances that contain records for the DiagnosticTest.

   1) For each DiagnosticLog instance, the client calls the Associators operation using LogManagesRecord as the association class and DiagnosticRecord as the result class. The operation returns the DiagnosticRecord instances in the Log.

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

9.8.4 GetDiagnosticExecutionFinalResults

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

1) The client calls the ExecQuery operation as follows:

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

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

   NOTE: Only one DiagnosticCompletionRecord shall be returned.

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

An alternate method without using ExecQuery is as follows:

Assume that the client starts at the known DiagnosticTest instance.

1) From the DiagnosticTest instance, the client calls the Associators operation using UseOfLog as the association class and DiagnosticLog as the result class. The operation returns the DiagnosticLog instances that contain records for the DiagnosticTest.

    1) For each DiagnosticLog instance, the client calls the Associators operation using LogManagesRecord as the association class and DiagnosticCompletionRecord as the result class. The operation returns the DiagnosticCompletionRecord instances in the Log.

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

9.8.5 GetDiagnosticExecutionResults

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

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

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

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

1) The client calls the ExecQuery operation as follows:

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

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

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

9.8.6 GetDiagnosticExecutionSettings

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

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

1) The client calls the ExecQuery operation as follows:

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

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

    NOTE: Only one DiagnosticSettingDataRecord instance shall be returned.

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

An alternate method without using ExecQuery is as follows:

Assume that the client starts at the known DiagnosticTest instance.

1) From the DiagnosticTest instance, the client calls the Associators operation using UseOfLog as the association class and DiagnosticsLog as the result class. The operation returns the DiagnosticsLog instances that contain records for the DiagnosticTest.
2) For each DiagnosticsLog instance, the client calls the Associators operation using LogManagesRecord as the association class and DiagnosticSettingDataRecord as the result class. The operation returns the DiagnosticSettingDataRecord instances in the Log.
3) For each returned instance, the client compares portion of DiagnosticRecord.InstanceID that contains the ConcreteJob.InstanceID with ConcreteJob.InstanceID to determine whether the instance is one created for the specific execution of the DiagnosticTest.
    1) The client reads the DiagnosticSettingDataRecord.Settings property, which is a DiagnosticSettingData embedded instance that contains the settings of the diagnostic execution.

Another alternate method without using ExecQuery is as follows:

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

Assume that the client starts at the known DiagnosticTest instance.

1) From the DiagnosticTest instance, the client calls the Associators operation using UseOfLog as the association class and DiagnosticLog as the result class. The operation returns the DiagnosticLog instances that contain records for the DiagnosticTest.
2) For each DiagnosticLog instance, the client calls the Associators operation using LogManagesRecord as the association class and DiagnosticSettingDataRecord as the result class. The operation returns the DiagnosticSettingRecord instances in the Log.
3) For each returned instance, the client compares portion of DiagnosticSettingDataRecord.InstanceID with ConcreteJob.InstanceID to determine whether the instance is the one created for the specific execution of the DiagnosticTest.
4) From the DiagnosticSettingDataRecord instance, the client calls the Associators operation using CorrespondingSettingsRecord as the association class and DiagnosticServiceRecord as the result class. The operation returns the DiagnosticServiceRecord instances created for the specific execution of the DiagnosticTest

9.8.7 GetDiagnosticProgress

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

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

1) The client may use any of the Discover Diagnostic Execution use cases (see 9.7) to find the desired ConcreteJob instances.
    1) The client reads the ConcreteJob.PercentComplete property to determine test progress.

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

1385 1) The client can use any of the Discover Diagnostic Execution use cases (see 9.7) to find the
1386 desired ConcreteJob instances.

1387 1) The client can register to receive a CIM_InstModification indication by creating an
1388 indication subscription using the following CIM_IndicationFilter.Query:

1389 SELECT * FROM CIM_InstModification
1390 WHERE “SourceInstance.ISA(“CIM_ConcreteJob”) and SourceInstance.InstanceID =
1391 <ConcreteJob.InstanceID> and PreviousInstance.PercentComplete <>
1392 SourceInstance.PercentComplete

1393 2) The indication received will notify the client that the PercentComplete property for the
1394 specific ConcreteJob has changed. The client can use the SourceInstance property in the
1395 indication to see the actual PercentComplete value to determine test progress.

1396

1397 10 CIM Elements

1398 Table 34 shows the instances of CIM Elements for this profile. Instances of the CIM Elements shall be
1399 implemented as described in Table 34. Clauses 7 (“Implementation”) and 8 (“Methods”) may impose
1400 additional requirements on these elements.

1401 **Table 34 – CIM Elements: Diagnostics Profile**

Element Name	Requirement	Description
Classes		
CIM_AffectedJobElement	Optional	Association to link a job to a managed element See 10.1.
CIM_AvailableDiagnosticService	Mandatory	Association to link diagnostic services that can be launched against managed elements See 10.2.
CIM_ConcreteJob	Mandatory	Used by the client to monitor and control the execution of a diagnostic service See 10.3.
CIM_CorrespondingSettingDataRecord (DiagnosticServiceRecord)	Optional	Association to link a settings record to its corresponding service records. If CIM_DiagnosticSettingDataRecord is implemented, this class is Mandatory. See 10.4.
CIM_CorrespondingSettingDataRecord (DiagnosticCompletionRecord)	Optional	Association to link a settings record to its corresponding completion records. If CIM_DiagnosticSettingDataRecord is implemented, this class is Mandatory. See 10.5.
CIM_DiagnosticCompletionRecord	Mandatory	Records that contain serviced completion information See 7.7 and 10.6.
CIM_DiagnosticLog	Mandatory	Although several legitimate mechanisms for logging results exist (see CIM_DiagnosticSettingData.LogStorage), aggregation of diagnostic records to a diagnostic log is Mandatory. See 7.6 and 10.7.

Element Name	Requirement	Description
CIM_DiagnosticServiceCapabilities	Optional	See 7.3 and 10.8.
CIM_DiagnosticServiceRecord	Mandatory	See 7.7 and 10.9.
CIM_DiagnosticSettingData (Default)	Optional	See 7.4 and 10.10.
CIM_DiagnosticSettingData (Client)	Optional	See 7.4 and 10.11.
CIM_DiagnosticSettingDataRecord	Optional	See 7.7 and 10.12.
CIM_DiagnosticTest	Mandatory	See 7.1 and 10.13.
CIM_ElementCapabilities	Optional	See 10.14.
CIM_ElementSettingData (JobSettingData)	Optional	See 10.15.
CIM_ElementSettingData (DiagnosticSettingData)	Optional	See 10.16.
CIM_ElementSoftwareIdentity	Mandatory	See 10.17.
CIM_HelpService	Optional	See 10.18.
CIM_HostedService	Mandatory	See 10.19 and 9.1.
CIM_JobSettingData (Default)	Optional	See 10.20.
CIM_JobSettingData (Client)	Optional	See 10.21.
CIM_LogManagesRecord	Mandatory	See 10.22.
CIM_OwningJobElement	Mandatory	See 10.23.
CIM_RecordAppliesToElement	Optional	See 10.24.
CIM_RegisteredProfile	Mandatory	See 10.25.
CIM_ServiceAffectsElement	Mandatory	See 10.26.
CIM_ServiceAvailableToElement	Mandatory	See 10.27.
CIM_ServiceComponent	Optional	See 10.28.
CIM_SoftwareIdentity	Mandatory	See 10.29.
CIM_UseOfLog	Mandatory	See 10.30.
Indications		
None defined in this profile		

10.1 CIM_AffectedJobElement

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

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.

10.2 CIM_AvailableDiagnosticService

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

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 7.2.1.
EstimatedDurationQualifier	Optional	See 7.2.2.

10.3 CIM_ConcreteJob

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

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 7.5.1.
StartTime	Mandatory	None
ElapsedTime	Mandatory	This property should be updated periodically so as to be useful as a “heartbeat.”
PercentComplete	Mandatory	See 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.

Properties	Requirement	Notes
RequestedState	Mandatory	None
RequestStateChange()	Mandatory	See 8.2.

10.4 CIM_CorrespondingSettingDataRecord (DiagnosticServiceRecord)

CIM_CorrespondingSettingDataRecord is used to associate a service record with the corresponding setting data record. Table 38 provides information about the properties of CIM_CorrespondingSettingDataRecord.

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

10.5 CIM_CorrespondingSettingDataRecord (DiagnosticCompletionRecord)

CIM_CorrespondingSettingDataRecord is used to associate a completion record with the corresponding setting data record. Table 39 provides information about the properties of CIM_CorrespondingSettingDataRecord.

Table 39 – Class: CIM_CorrespondingSettingDataRecord

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

10.6 CIM_DiagnosticCompletionRecord

CIM_DiagnosticCompletionRecord is used to report the final state of diagnostic execution (OK, Failed, Incomplete, Aborted, and so on). Table 40 provides information about the properties of CIM_DiagnosticCompletionRecord.

Table 40 – 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]*$")
CreationTimeStamp	Mandatory	None
RecordData	Mandatory	None
RecordFormat	Mandatory	None
ServiceName	Mandatory	The ServiceName property shall be constructed as follows: <OrgID>:<TestName>. (pattern "^.*[:].*$")
ManagedElementName	Mandatory	This property will be formatted as a free-form string of variable length. (pattern ".*")
RecordType	Mandatory	The record type shall be “2 (Results). Matches 2 (Results)
ExpirationDate	Mandatory	See 7.7.1.
CompletionState	Mandatory	None
OtherCompletionStateDescription	Conditional	If CompletionState has the value 1 (Other), this property is Mandatory.
LoopsPassed	Optional	If looping is supported, this property is Mandatory.
LoopsFailed	Optional	If looping is supported, this property is Mandatory.
ErrorCode	Mandatory	This property shall be an array that contains the error codes of all errors generated by the diagnostic service execution. If there are no errors this property may have the value NULL.

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

10.7 CIM_DiagnosticLog

CIM_DiagnosticLog represents a log that aggregates all of the results (records) that the execution of a diagnostic generates. Table 41 provides information about the properties of CIM_DiagnosticLog.

Table 41 – Class: CIM_DiagnosticLog

Properties	Requirement	Notes
InstanceID	Mandatory	**Key** InstanceID should be constructed using the following preferred algorithm: <OrgID>:<LocalID> (See the MOF file for more detail.) (pattern "^.*[:].*$")
ClearLog()	Mandatory	See 8.3.

10.8 CIM_DiagnosticServiceCapabilities

CIM_DiagnosticServiceCapabilities publishes the diagnostic service's capabilities, such as settings and execution controls that are supported. Table 42 provides information about the properties of CIM_DiagnosticServiceCapabilities.

Table 42 – Class: CIM_DiagnosticServiceCapabilities

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

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

10.9 CIM_DiagnosticServiceRecord

CIM_DiagnosticServiceRecord is used to report diagnostic service messages such as results, errors, warnings, and status. Table 43 provides information about the properties of CIM_DiagnosticServiceRecord.

Table 43 – 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

Properties	Requirement	Notes
ErrorCode	Conditional	If the RecordType value is 7(Device Errors) or 8 (Service Errors), this property shall be an array that contains only one error code number. If the RecordType value is 2 (Results), this property shall be an array that contains the error codes of all errors generated by the diagnostic service or subtest execution at the time when the record was logged. If the RecordType value is not 2 (Results) or 7(Device Errors) or 8 (Service Errors), this this property may be NULL. The property will be formatted as a free-form string of variable length. (pattern ".*")
ErrorCount	Conditional	If the RecordType value is 7(Device Errors) or 8 (Service Errors), this property shall be an array that has just one element whose value is 1. If the RecordType value is 2 (Results), this property should be an array where each position should contain the number of times that an error occurred which can be identified by the same position in the ErrorCode array. If the RecordType value is not 2 (Results) or 7(Device Errors) or 8 (Service Errors), this this property may be NULL.
ServiceName	Mandatory	This property shall be constructed as follows: <OrgID>:<TestName>. (pattern "^.*[:].*$")
ManagedElementName	Mandatory	This property shall be formatted as a free-form string of variable length. (pattern ".*")
RecordType	Mandatory	A RecordType value of 2 (Results) shall be used to log interim results from the diagnostic service or subtest execution. In contrast, final results shall use the DiagnosticCompletionRecord class.
OtherRecordTypeDescription	Conditional	If RecordType has the value 1 (Other), this property is Mandatory.
ExpirationDate	Mandatory	See 7.7.1.

10.10 CIM_DiagnosticSettingData (Default)

Diagnostic services use CIM_DiagnosticSettingData to publish default settings using CIM_ElementSettingData where the IsDefault property has the value of TRUE. Table 44 provides information about the properties of CIM_DiagnosticSettingData.

Table 44 – 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	If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 4 (HaltOnError), this property can be used to affect test behavior. When this property is TRUE, the service should halt after finding the first error.
QuickMode	Optional	If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 3 (QuickMode), this property can be used to affect test behavior. 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	If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 2 (PercentOfTestCoverage), this property can be used to affect test behavior. This property requests the service to reduce test coverage to the specified percentage.
LoopControl	Optional	This property is used in combination with LoopControlParameter to set one or more loop control mechanisms that limit the number of times that a test should be repeated.

Properties	Requirement	Notes
LoopControlParameter	Conditional	If a LoopControl includes the value of 3 (Count) or 5 (ErrorCount), the corresponding LoopControlParameter array element shall represent a uint32 numeric value. If a LoopControl includes the value of 4 (Timer), the corresponding LoopControlParameter array element shall represent a datetime value. (pattern "^b[01]* | ^d[0123456789]* | ^x[0123456789ABCDEFabcdef]* | ^[0123456789]*")
OtherLoopControlDescriptions	Conditional	If LoopControl includes the value 1 (Other), this property is Mandatory.
ResultPersistence	Mandatory	If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 5 (ResultPersistence), this property can be used to affect test behavior. This property specifies how many seconds the records should persist after service execution finishes. 0 (zero) indicates “no persistence” and 0xFFFFFFFF indicates “persist forever”. See 7.7.1.
LogOptions	Optional	This property specifies the types of data that should be logged by the diagnostic service.
OtherLogOptionsDescriptions	Conditional	If LogOptions includes the value 1 (Other), this property is Mandatory.
LogStorage	Optional	This property specifies the logging mechanism to store the diagnostic results. This property must be one of the values in DiagnosticServiceCapabilities.LogStorage
OtherLogStorageDescriptions	Conditional	If LogStorage includes the value 1 (Other), this property is Mandatory.
VerbosityLevel	Optional	This property specifies the desired volume or detail logged by a diagnostic service.

10.11 CIM_DiagnosticSettingData (Client)

A client uses CIM_DiagnosticSettingData to override the defaults settings and run a diagnostic service using specific settings. Such settings are passed as the DiagSetting argument when the RunDiagnosticService( ) extrinsic method of CIM_DiagnosticTest is invoked. Table 45 provides information about the properties of CIM_DiagnosticSettingData.

Table 45 – 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	If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 4 (HaltOnError), this property can be used to affect test behavior. When this property is TRUE, the service should halt after finding the first error.
QuickMode	Optional	If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 3 (QuickMode), this property can be used to affect test behavior. 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	If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 2 (PercentOfTestCoverage), this property can be used to affect test behavior. This property requests the service to reduce test coverage to the specified percentage.
LoopControl	Optional	This property is used in combination with LoopControlParameter to set one or more loop control mechanisms that limit the number of times that a test should be repeated.

Properties	Requirement	Notes
LoopControlParameter	Conditional	If a LoopControl includes the value of 3 (Count) or 5 (ErrorCount), the corresponding LoopControlParameter array element shall represent a uint32 numeric value. If a LoopControl includes the value of 4 (Timer), the corresponding LoopControlParameter array element shall represent a datetime value. (pattern "^b[01]* | ^d[0123456789]* | ^x[0123456789ABCDEFabcdef]* | ^[0123456789]*")
OtherLoopControlDescriptions	Conditional	If LoopControl includes the value 1 (Other), this property is Mandatory.
ResultPersistence	Mandatory	If the DiagnosticServiceCapabilities.SupportedServiceModes array contains a value of 5 (ResultPersistence), this property can be used to affect test behavior. This property specifies how many seconds the records should persist after service execution finishes. 0 (zero) indicates “no persistence” and 0xFFFFFFFF indicates “persist forever”. See 7.7.1.
LogOptions	Optional	This property specifies the types of data that should be logged by the diagnostic service.
OtherLogOptionsDescriptions	Conditional	If LogOptions includes the value 1 (Other), this property is Mandatory.
LogStorage	Optional	This property specifies the logging mechanism to store the diagnostic results. This property must be one of the values in DiagnosticServiceCapabilities.LogStorage
OtherLogStorageDescriptions	Conditional	If LogStorage includes the value 1 (Other), this property is Mandatory.
VerbosityLevel	Optional	This property specifies the desired volume or detail logged by a diagnostic service.

10.12 CIM_DiagnosticSettingDataRecord

CIM_DiagnosticSettingDataRecord stores the settings used in a specific diagnostic service execution. Table 46 provides information about the properties of CIM_DiagnosticSettingDataRecord.

Table 46 – 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]*$")
CreationTimeStamp	Mandatory	None
ServiceName	Mandatory	This property shall be constructed as follows: <OrgID>:<TestName>. (pattern "^.*[:].*$")
ManagedElementName	Mandatory	This property will be formatted as a free-form string of variable length. (pattern ".*")
ExpirationDate	Mandatory	See 7.7.1.
Settings	Conditional	This property is set to a string that encodes a DiagnosticSettingData instance. If an instance of CIM_DiagnosticSettingData is associated through CIM_ElementSettingData to the instance of CIM_DiagnosticTest at the time the Diagnostic Service is run, this property is Mandatory.

10.13 CIM_DiagnosticTest

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

Table 47 provides information about the properties of CIM_DiagnosticTest.

Table 47 – Class: CIM_DiagnosticTest

Properties	Requirement	Notes
SystemCreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**

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

10.14 CIM_ElementCapabilities

CIM_ElementCapabilities associates a diagnostic service with its capabilities. Table 48 provides information about the properties of CIM_ElementCapabilities.

Table 48 – Class: CIM_ElementCapabilities

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

10.15 CIM_ElementSettingData (JobSettingData)

CIM_ElementSettingData associates the job settings with the job used to run a diagnostic test. Table 49 provides information about the properties of CIM_ElementSettingData.

1474 **Table 49 – Class: CIM_ElementSettingData**

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

1475 ## 10.16 CIM_ElementSettingData (DiagnosticSettingData)

1476 CIM_ElementSettingData associates the diagnostic service with its default. Table 50 provides information
1477 about the properties of CIM_ElementSettingData.

1478 **Table 50 – Class: CIM_ElementSettingData**

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

1479 ## 10.17 CIM_ElementSoftwareIdentity

1480 CIM_ElementSoftwareIdentity associates the diagnostic service with its version information. Table 51
1481 provides information about the properties of CIM_ElementSoftwareIdentity.

1482 **Table 51 – Class: CIM_ElementSoftwareIdentity**

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

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

10.18 CIM_HelpService

CIM_HelpService is the preferred way for a service to publish online help information. Table 52 provides information about the properties of CIM_HelpService.

Table 52 – Class: CIM_HelpService

Properties	Requirement	Notes
SystemCreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
CreationClassName	Mandatory	**Key**
Name	Mandatory	**Key** This property will be formatted as a free-form string of variable length. (pattern ".*")
ElementName	Mandatory	This property will be formatted as a free-form string of variable length. (pattern ".*")
DeliveryOptions	Mandatory	None
OtherDeliveryOptionDescription	Conditional	If DeliveryOptions has the value of 1 (Other), this property is Mandatory.
DocumentsAvailable	Mandatory	This property will be formatted as a free-form string of variable length. (pattern ".*")
DocumentDescriptions	Mandatory	None
DocumentFormat	Mandatory	None
OtherDocumentFormatDescription	Conditional	If DocumentFormat has the value of 1 (Other), this property is Mandatory.
GetHelp()	Mandatory	See 8.4.

10.19 CIM_HostedService

CIM_HostedService is used to associate an instance of CIM_DiagnosticTest with an instance of CIM_ComputerSystem to which the CIM_DiagnosticTest is scoped and to associate an instance of CIM_HelpService with an instance of CIM_ComputerSystem to which the CIM_HelpService is scoped. Table 53 provides information about the properties of CIM_HostedService.

Table 53 – Class: CIM_HostedService

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

10.20 CIM_JobSettingData (Default)

Diagnostic services use CIM_JobSettingData to publish default settings using CIM_ElementSettingData where the IsDefault property has the value of TRUE. Table 54 provides information about the properties of CIM_JobSettingData.

Table 54 – Class: CIM_JobSettingData

Properties	Requirement	Notes
InstanceID	Mandatory	**Key**
ElementName	Mandatory	This property shall be formatted as a free-form string of variable length. (pattern ".*")
DeleteOnCompletion	Conditional	This property indicates whether the job should be automatically deleted upon completion. The CIM_ConcreteJob.TimeBeforeRemoval property overrides this property. If CIM_JobSettingData is supported, this property is Mandatory.

10.21 CIM_JobSettingData (Client)

A client uses CIM_JobSettingData to override the defaults settings and run a diagnostic service using specific job settings. Such settings are passed as the JobSetting argument when the RunDiagnosticService( ) extrinsic method of CIM_DiagnosticTest is invoked. Table 55 provides information about the properties of CIM_JobSettingData.

Table 55 – Class: CIM_JobSettingData

Properties	Requirement	Notes
InstanceID	Mandatory	**Key**

Properties	Requirement	Notes
ElementName	Mandatory	This property shall be formatted as a free-form string of variable length. (pattern ".*")
DeleteOnCompletion	Conditional	This property indicates whether the job should be automatically deleted upon completion. The CIM_ConcreteJob.TimeBeforeRemoval property overrides this property. If CIM_JobSettingData is supported, this property is Mandatory.

10.22 CIM_LogManagesRecord

CIM_LogManagesRecord associates a log with its records (service records, setting records, or completion records). Table 56 provides information about the properties of CIM_LogManagesRecord.

Table 56 – Class: CIM_LogManagesRecord

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

10.23 CIM_OwningJobElement

CIM_OwningJobElement associate a diagnostic service with its jobs (jobs that are launched by this diagnostic). Table 57 provides information about the properties of CIM_OwningJobElement.

Table 57 – 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.

10.24 CIM_RecordAppliesToElement

CIM_RecordAppliesToElement associates a record with the managed elements (diagnostic service and device) that have a relationship with this record. Table 58 provides information about the properties of CIM_RecordAppliesToElement.

Table 58 – 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.

10.25 CIM_RegisteredProfile

CIM_RegisteredProfile identifies the *Diagnostics Profile* in order for a client to determine whether an instance of CIM_DiagnosticService is conformant with this profile. The CIM_RegisteredProfile class is defined by the *Profile Registration Profile*. With the exception of the mandatory values specified in Table 59, the behavior of the CIM_RegisteredProfile instance is in accordance with the *Profile Registration Profile*.

Table 59 – Class: CIM_RegisteredProfile

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

10.26 CIM_ServiceAffectsElement

CIM_ServiceAffectsElement is used to associate to the diagnostic service any managed elements that are affected by the running of the service. Table 60 provides information about the properties of CIM_ServiceAffectsElement.

Table 60 – 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.

10.27 CIM_ServiceAvailableToElement

CIM_ServiceAvailableToElement associates the diagnostic service with its help service information. Table 61 provides information about the properties of CIM_ServiceAvailableToElement.

Table 61 – 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

10.28 CIM_ServiceComponent

CIM_ServiceComponent associates a test that is also part of another test. This class is used when DiagnosticTest.Characteristics includes the value 6 (Is Package) and subtests are implemented as separate instances of DiagnosticTest. Table 62 provides information about the properties of CIM_ServiceComponent.

Table 62 – 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.

10.29 CIM_SoftwareIdentity

CIM_SoftwareIdentity is used to publish version information about the diagnostic service. Table 63 provides information about the properties of CIM_SoftwareIdentity.

Table 63 – 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 ".*")

10.30 CIM_UseOfLog

CIM_UseOfLog associates a log with a managed element (a device or diagnostic service) whose information is stored in the log. Table 64 provides information about the properties of CIM_UseOfLog.

Table 64 – Class: CIM_UseOfLog

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

ANNEX A
(informative)

Change Log

Version	Date	Description
1.0.0a	2006-04-17	Preliminary
1.0.1	2009-09-23	Final Standard
2.0.0	2010-10-21	DMTF Standard

3.3 Bibliography

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