Document Identifier: DSP1115

Date: 2015-04-16

Version: 1.0.0a

System Memory Diagnostics Profile

Information for Work in Progress version:

IMPORTANT: This document is not a standard. It does not necessarily reflect the views of the DMTF or all of its members. Because this document is a Work in Progress, it may still change, perhaps profoundly and without notice. This document is available for public review and comment until superseded.

Provide any comments through the DMTF Feedback Portal:
http://www.dmtf.org/standards/feedback

Supersedes: None

Document Type: Specification

Document Class: Normative

Document Status: Work in Progress

Document Language: en-US

Copyright notice

Copyright © 2015 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a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

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

CONTENTS

Foreword 5
Introduction 6
1 Scope 7
2 Normative references 7
3 Terms and definitions 8
4 Symbols and abbreviated terms 8
5 Synopsis 10
6 Description 10
7 Implementation 13
7.1 System Memory test information 13
7.2 CIM_MemoryDiagnosticTest 17
7.3 CIM_MemoryDiagnosticSettingData 19
7.4 CIM_MemoryDiagnosticServiceCapabilities 22
7.5 System Memory Diagnostics Profile indications support 24
7.6 Diagnostics alert indications and standard messages 25
8 Methods 36
8.1 CIM_MemoryDiagnosticTest.RunDiagnosticService( ) 36
8.2 Profile conventions for operations 36
9 Use cases 36
9.1 Overview 36
9.2 Use case summary 36
9.3 Verifying System Memory health 37
9.4 Troubleshooting System Memory issues 38
10 CIM elements 39
10.1 CIM_MemoryDiagnosticTest 41
10.2 CIM_MemoryDiagnosticSettingData 41
10.3 CIM_MemoryDiagnosticServiceCapabilities 41
10.4 CIM_RegisteredProfile 42
10.5 CIM_AffectedJobElement 42
10.6 CIM_AvailableDiagnosticService 42
10.7 CIM_ElementCapabilities 43
10.8 CIM_ElementSettingData (DiagnosticSettingData) 43
10.9 CIM_ElementSettingData (JobSettingData) 43
10.10 CIM_ElementSoftwareIdentity 44
10.11 CIM_HostedService 44
10.12 CIM_OwningJobElement 44
10.13 CIM_RecordAppliesToElement 45
10.14 CIM_ServiceAffectsElement 45
10.15 CIM_ServiceAvailableToElement 45
10.16 CIM_ServiceComponent 46
10.17 CIM_UseOfLog 46
10.18 CIM_FilterCollection 46
10.19 CIM_IndicationFilter 47
10.20 CIM_MemberOfCollection 47
10.21 CIM_OwningCollectionElement 47
ANNEX A (informative) Change log 49

Figures

Figure 1 – System Memory Diagnostics Profile: Profile class diagram 12

Tables

Table 1 – Referenced profiles 10
Table 2 – Test type information 13
Table 3 – CIM_MemoryDiagnosticTest property requirements 18
Table 4 – CIM_MemoryDiagnosticTest property requirements 19
Table 5 – CIM_MemoryDiagnosticSettingData property requirements 20
Table 6 – CIM_MemoryDiagnosticServiceCapabilities property requirements 23
Table 7 – System Memory Diagnostics Profile use cases 37
Table 8 – CIM elements: System Memory Diagnostics Profile 39
Table 9 – Class: CIM_MemoryDiagnosticTest 41
Table 10 – Class: CIM_MemoryDiagnosticSettingData 41
Table 11 – Class: CIM_MemoryDiagnosticServiceCapabilities 42
Table 12 – Class: CIM_RegisteredProfile 42
Table 13 – Class: CIM_AffectedJobElement 42
Table 14 – Class: CIM_AvailableDiagnosticService 43
Table 15 – Class: CIM_ElementCapabilities 43
Table 16 – Class: CIM_ElementSettingData 43
Table 17 – Class: CIM_ElementSettingData 44
Table 18 – Class: CIM_ElementSoftwareIdentity 44
Table 19 – Class: CIM_HostedService 44
Table 20 – Class: CIM_OwningJobElement 45
Table 21 – Class: CIM_RecordAppliesToElement 45
Table 22 – Class: CIM_ServiceAffectsElement 45
Table 23 – Class: CIM_ServiceAvailableToElement 46
Table 24 – Class: CIM_ServiceComponent 46
Table 25 – Class: CIM_UseOfLog 46
Table 26 – Class: CIM_FilterCollection 46
Table 27 – Class: CIM_IndicationFilter 47
Table 30 – Class: CIM_MemberOfCollection 47
Table 31 – Class: CIM_OwningCollectionElement 48

Foreword

The *System Memory Diagnostics Profile* (DSP1115) was prepared by the Diagnostics Working Group of 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:

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

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 the profile is to ensure interoperability of Web-Based Enterprise Management (WBEM) services for a specific subset of the CIM schema — in this case, System Memory 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, which enable seamless integration of vendor-supplied diagnostic services into system and storage area network management frameworks.

The goal of the *System Memory Diagnostics Profile* is to define industry-standard building blocks that enable seamless problem determination support for System Memory and to troubleshoot memory issues involving volatile memory. The profile extends the standard diagnostic profile by identifying a base set of memory functions that should be diagnosed by provider implementations. Suppliers can differentiate their diagnostic offering by providing this base set of diagnostics and developing diagnostics to analyze the proprietary features of System Memory.

Document conventions

Typographical conventions

The following typographical conventions are used in this document:

- Document titles are marked in *italics*.
- Important terms that are used for the first time are marked in *italics*.

ABNF usage conventions

Format definitions in this document are specified using ABNF (see RFC5234), with the following deviations:

- Literal strings are to be interpreted as case-sensitive Unicode characters, as opposed to the definition in RFC5234 that interprets literal strings as case-insensitive US-ASCII characters.

System Memory Diagnostics Profile

1 Scope

The *System Memory Diagnostics Profile* specializes the *Diagnostics Profile* (DSP1002) by defining the diagnostic tests needed to determine the health of System Memory as well as the tests needed to troubleshoot computing problems involving System Memory. The diagnostic tests are defined as subclasses of CIM_DiagnosticTest. System Memory represents the total memory installed and available to the system.

2 Normative references

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

DMTF DSP0004, *CIM Infrastructure Specification 2.6*,
http://dmtf.org/sites/default/files/standards/documents/DSP0004_2.6.pdf

DMTF DSP0200, *CIM Operations over HTTP 1.3*,
http://dmtf.org/sites/default/files/standards/documents/DSP0200_1.3.pdf

DMTF DSP0223, *Generic Operations 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0223_1.0.pdf

DMTF DSP1001, *Management Profile Specification Usage Guide 1.0*,
http://dmtf.org/sites/default/files/standards/documents/DSP1001_1.0.pdf

DMTF DSP1002, *Diagnostics Profile 2.1*,
http://dmtf.org/sites/default/files/standards/documents/DSP1002_2.1.0a.pdf

DMTF DSP1026, System Memory Profile 1.0.1,
http://dmtf.org/sites/default/files/standards/documents/DSP1026_1.0.1.pdf

DMTF DSP1033, *Profile Registration Profile 1.0*,
http://dmtf.org/sites/default/files/standards/documents/DSP1033_1.0.pdf

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

DMTF DSP1119, Diagnostics Job Control Profile 1.0.0,
http://dmtf.org/sites/default/files/standards/documents/DSP1119_1.0.0b.pdf

DMTF DSP8055, Diagnostics Message Registry 1.0.0d,
http://www.dmtf.org/sites/default/files/standards/documents/DSP8055_1.0.0d.xml

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

ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*,
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, and DSP1001 apply to this document.

3.1
Device Moniker

A Device Memory Moniker can be any of the following:

- Device Moniker – Identifies the unique name for a physical memory device under test.

  This can be one of the following names:

  – The Object path of the physical memory device
  – The ElementName of the physical memory device
  – A unique, user-friendly name not in the model (such as, asset name)

Whichever n is used shall be used consistently for all devices within the scoping profile.

4 Symbols and abbreviated terms

The following symbols and 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 CT

Common Transport

4.6 FRU

Field Replaceable Unit

4.7 ICMP

Internet Control Message Protocol

4.8 LED

Light-Emitting Diode

4.9 LUN

Logical Unit Number

4.10 ME

Managed Element

4.11 MOF

Managed Object Format

4.12 PD

Problem Determination

4.13 PFA

Predictive Failure Analysis

4.14 POST

Power-On Self-Test

4.15 SLP

Service Location Protocol

4.16 WBEM

Web-Based Enterprise Management

4.17 WWPN

World Wide Port Name

4.18
UEFI
Unified Extensible Firmware Interface

4.19
BIOS
Basic Input/Output System

5 Synopsis

Profile Name: System Memory Diagnostics

Version: 1.0.0a

Organization: DMTF

CIM schema version: 2.44

Central Class: CIM_MemoryDiagnosticTest

Scoping Class: CIM_ComputerSystem

Specializes: Diagnostics Profile 2.1.0

The *System Memory Diagnostics Profile* extends the management capability of referencing profiles by adding common methods for determining that System Memory is operating normally and for troubleshooting volatile memory problems involving System Memory in a managed system.

CIM_MemoryDiagnosticTest shall be the Central Class of this profile. The instance of CIM_MemoryDiagnosticTest 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.

The CIM_ManagedElement is CIM_Memory or a subclass of it.

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

Table 1 – Referenced profiles

Profile Name	Organization	Version	Description
Diagnostics	DMTF	2.1	Specializes
Profile Registration	DMTF	1.0	Mandatory
System Memory	DMTF	1.0.1	Optional

6 Description

System Memory diagnostics can first be categorized based on the computing system environment in which they execute. In-band diagnostics are those that execute within the Operating System. Out-of-band diagnostics run within a pre-boot environment. Within each of these categories, diagnostics can be either destructive or nondestructive.

DSP1002 defines destructive tests as those that have the potential for destroying data, permanently altering the state, or reconfiguring the device. In the case of System Memory, any test that could cause a previously executing application to experience a data failure should be considered destructive because it could cause the current operation to fail. An example would be a write-test pattern to verify device data integrity. When the test runs, System Memory cannot allow normal access.

Nondestructive diagnostics are those that can be safely executed without disrupting normal access, such as performing a memory read to verify the accessibility of a memory device.

Comprehensive memory management requires both categories and types of diagnostics to maintain operations in production environments. Memory diagnostics shall work in both pre-boot and normal operating system environments.

The diagnostic tests specified in this profile may be implemented in firmware, BIOS, or the System Memory Diagnostics Provider itself. The goal of the *System Memory Diagnostics Profile* is to define a set of standard diagnostics that meet these needs and are both vendor and hardware agnostic.

Physical Memory is a field replaceable unit (FRU); when defective, it can be simply replaced. When the host system wishes to verify the health of System Memory, the diagnostic test should not have to be concerned with testing the individual memory components. Rather the diagnostic test needs to be able to call upon a single diagnostic that tests all of System Memory. This self-test shall be comprehensive, similar to a Power-On Self-Test (POST). By its nature, this test is destructive. All System Memory diagnostic providers shall support a self-test.

Verifying the health of System Memory nondestructively is problematic. Any definitive health verification disrupts, suspends, or corrupts normal data access. However, it is possible to determine relative health of System Memory by using data, such as its current operational state, error counts, and the results of its last POST. Diagnostics providers should take advantage of this test to report any detected degraded conditions before they become problems. Executing this test would also verify that basic access with System Memory is operational. All System Memory diagnostic providers shall support a Status test.

To enable the isolation of certain types of faults, System Memory should also be testable at its boundaries. The boundaries of System Memory are its connection to the Memory Controller and the internal bus; system; or memory bus. Testing at these boundaries makes it possible to isolate problems to System Memory or the Memory Controller. For instance, if a memory cell is dead and a write targeted at it succeeds, the CPU would be unaware of the problem. A subsequent read may or may not fail and the CPU would be unaware of the existing fault. It could be a problem with the memory or memory controller. Performing memory diagnostics would indicate whether the problem was with the System Memory. In-band and out-of-band diagnostics, both destructive and nondestructive may be required to isolate the specific fault.

Many host systems contain multiple physical memory devices. If one of these devices is known to be malfunctioning, it can be difficult to visually identify which device is the defective unit when attempting to replace it. Flashing one or more LEDs on the component board in a known pattern, or beaconing, resolves this problem. The flashing LEDs allow the memory device in question to be easily identified. Beaconing is nondestructive. All System Memory diagnostic providers shall support a Beacon test only if the System Memory under test supports it.

The *System Memory Diagnostics Profile* describes the set of tests necessary for diagnosing System Memory issues and troubleshooting some computing issues. Each test is a specialization of CIM_DiagnosticTest. The supported service modes, user controls, log options, and test patterns for each test are advertised through the CIM_MemoryDiagnosticsServiceCapabilities instance. For tests with specifiable parameters, the default parameter values are advertised through instances of CIM_ElementSettingData that associate an instance of CIM_MemoryDiagnosticSettingData to the test. Where supported, clients specify nondefault test parameters by creating instances of CIM_MemoryDiagnosticSettingData that are associated to instances of CIM_MemoryDiagnosticTest. This configuration is illustrated in Figure 1.

The tests are designed such that they can be executed to effectively test actually physical memory without regard to whether caching is present or not in the system.

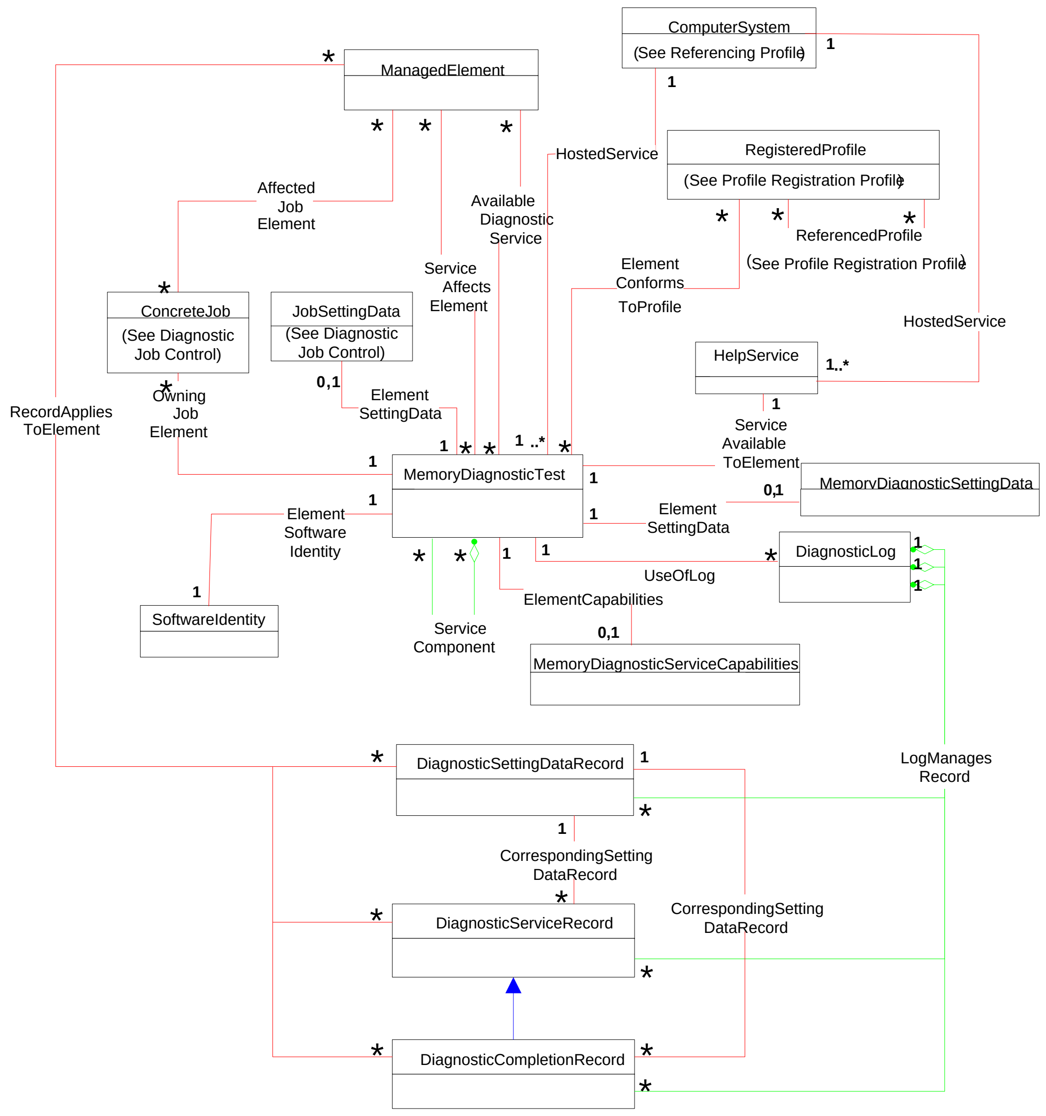
ComputerSystem
(See Referencing Profile)
ManagedElement
RegisteredProfile
(See Profile Registration Profile)
HostedService
Affected Job Element
Available Diagnostic Service
Service Affects Element
Element Conforms ToProfile
ReferencedProfile
(See Profile Registration Profile)
HostedService
ConcreteJob
(See Diagnostic Job Control)
JobSettingData
(See Diagnostic Job Control)
HelpService
RecordApplies ToElement
Owning Job Element
Element SettingData
Service Available ToElement
MemoryDiagnosticTest
MemoryDiagnosticSettingData
Element Software Identity
Element SettingData
DiagnosticLog
UseOfLog
ElementCapabilities
SoftwareIdentity
Service Component
MemoryDiagnosticServiceCapabilities
LogManages Record
DiagnosticSettingDataRecord
CorrespondingSetting DataRecord
DiagnosticServiceRecord
CorrespondingSetting DataRecord
DiagnosticCompletionRecord
1
*
1..*
0,1

Figure 1 – System Memory Diagnostics Profile: Profile class diagram

The ManagedElement that is the UserOfService reference on the AvailableDiagnosticService association is System Memory (as represented by the CIM_Memory class). The ManagedElements that are AffectedElement references on the ServiceAffectsElement associations can be any element that is affected by the DiagnosticTest (for example, the PhysicalMemory, Memory, or the system that contains them). The ServiceAffectsElement has a broader scope than the AvailableDiagnosticService association.

7 Implementation

This clause provides additional implementation details for the various diagnostic tests of this profile.

7.1 System Memory test information

Table 2 contains information about the test types.

Table 2 – Test type information

Test Name	Test Information	
Electrical Wiring	**Description**	The diagnostic checks for the existence of a physical memory device, a memory chip, in the system.
	Coverage Range	Missing or incorrectly connected physical memory.
	Destructive	Yes
	User Control	The user may specify a list of addresses for the targeted physical memory device. At least three addresses shall be specified.
	Execution Time	The test shall run on the order of less than a second.
	Built into Device	No
	Details	Write the byte value 1 to the first address, 2 to the second address, and 3 to the third address. Next verify the data at the first, second, and third addresses. If the first data value read corresponds to the last value written, instead of the first, the memory chip is missing. The test is simply reading the capacitance on the data bus.
Data Bus Walking 1s	**Description**	The diagnostic verifies the data path from host to target is working properly.
	Coverage Range	Data Bus
	Destructive	Yes
	User Control	The user may specify a single address. If null, the lowest accessible address is used.
	Execution Time	The test shall run on the order of less than a second.
	Built into Device	No
	Details	A single data bit, the lowest order bit on the data bus, is set to 1 and then "walked" up through all the data bits on the data bus at the same address. After each write, the data value is read back and verified.
Address Bus Walking 1s	**Description**	The diagnostic verifies the address path from host to target is working properly.
	Coverage Range	Address Bus
	Destructive	Yes
	User Control	The user may specify a single data value. If null, the default value will be an alternating pattern of 1's and 0's, i.e., 01010101.
	Execution Time	The test shall run on the order of less than a second.
	Built into Device	No
	Details	A single address bit, the lowest order bit on the address bus, is set to 1 and then "walked" up through all the address bits on the address bus

Test Name	Test Information	
		writing the specified data value at each address. After each write, the data value is read back and verified.
Power-of-Two Addressing	**Description**	The diagnostic verifies that the address path from host to target is working properly and that there are no overlapping addresses.
	Coverage Range	Address Bus
	Destructive	Yes
	User Control	The user may specify the base address, number of bytes to write, and the data value to use. If an address is not specified, 0 is used as the default for the most effective coverage. If the number of bytes is not given, all available memory is covered. If no data value is specified, an alternating pattern of 1's and 0's will be used, i.e., 01010101.
	Execution Time	The test shall run on the order of seconds.
	Built into Device	No
	Details	The data value is written to the base address and then at each power-of-two offset within the memory range. This write action is followed by writing again to the base address with a new data value, a complement of the initial data value. The value is read and verified at the base address and each of the other power-of-two offsets. If the value matches at any address other than the base address, that address is an overlapping address and the test is complete. If no overlapping address is found, continue this procedure for each of the remaining offsets.
Self Addressing	**Description**	The diagnostic verifies that the address path from host to target is working properly.
	Coverage Range	Address Bus
	Destructive	Yes
	User Control	None
	Execution Time	The test shall run on the order of seconds.
	Built into Device	No
	Details	Starting with the first address, each address is written with its own address and then read to verify the data value against the current address.
Increment and Decrement	**Description**	The diagnostic verifies that both the address and data paths from the host to target are working properly.
	Coverage Range	Address Bus, Data Bus and Device
	Destructive	Yes
	User Control	The user may specify the base address and number of bytes for the test. If the base address is not specified, 0 is used as the default for the most effective coverage. If the number of bytes is not given, all available memory is covered.
	Execution Time	The test shall run on the order of seconds.
	Built into Device	No

Test Name	Test Information	
	Details	A byte value of 1 is written to the base address and then read and verified. Next a value of 2 is written to the next address, read, and verified. Address and data values are incremented in this manner until all the specified bytes are written. This procedure is repeated again starting with the base address, but with the complement value of 1. Each subsequent address will have a data value written that is the previous value decremented by 1. The tests are complete when an error is found or all specified bytes have been tested.
Moving Inversions 0s and 1s	**Description**	The diagnostic verifies that both the address and data paths from the host to target are working properly.
	Coverage Range	Address Bus, Data Bus, and Device
	Destructive	Yes
	User Control	The user may specify the base address and number of bytes for the test. If the base address is not specified, 0 is used as the default for the most effective coverage. If the number of bytes is not given, all available memory is covered.
	Execution Time	The test shall run on the order of seconds.
	Built into Device	No
	Details	A byte value of 0 is written to the base address and then read and verified. Next the byte's complement is written, read, and verified. The address is incremented and the procedure is repeated until all bytes have been processed.
Moving Inversions Random	**Description**	The diagnostic verifies that both the address and data paths from the host to target are working properly.
	Coverage Range	Address Bus, Data Bus, and Device
	Destructive	Yes
	User Control	The user may specify the base address and number of bytes for the test. If the base address is not specified, 0 is used as the default for the most effective coverage. If the number of bytes is not given, all available memory is covered. An optional number of passes parameter may be present. Additional passes with a different seed and random value increases the effectiveness of the test.
	Execution Time	The test shall run on the order of seconds.
	Built into Device	No
	Details	A byte of a random value is written to the base address and then read and verified. Next the byte's complement is written, read, and verified. The address is incremented and the procedure is repeated until all bytes have been processed.
Bit Fade	**Description**	The diagnostic verifies that the device is working properly.
	Coverage Range	Device
	Destructive	Yes
	User Control	The user may specify the base address, the number of bytes to write, and the length of time to wait before verifying the data value. If the base address is not specified, 0 is used as the default for the most effective coverage. If the number of bytes is not given, all available memory is covered. If a wait time is not specified, a value of 1 minute is used.
	Execution Time	The test shall run on the order of minutes.

Test Name	Test Information	
	Built into Device	No
	Details	A byte value of 0 is written to the base address, the address is incremented, and the next byte written. This write action is repeated until all the specified bytes have been written to. The test then waits for the specified time before reading and verifying the data value at each of the addresses. The procedure is then repeated using a byte value of FFh.
Reset	**Coverage Area**	The diagnostic causes a physical memory device, as identified by a logical memory address, to reinitialize itself.
	Coverage Range	Device
	Destructive	Yes
	User Control	The physical component to reset can be specified.
	Execution Time	The test shall run on the order of seconds.
	Built into Device	Yes
	Details	The implementation of this test is vendor specific.
Self-Test	**Coverage Area**	The diagnostic causes the internal components of a physical memory device, as identified by a logical memory address, to be tested.
	Coverage Range	Device
	Destructive	Yes
	User Control	The physical component can be specified if the device supports a self-test.
	Execution Time	The test shall run on the order of seconds.
	Built into Device	Yes
	Details	The details of the self-test are vendor specific. It is expected that the test will be comprehensive, testing all possible components on the physical memory device, such as a serial bit shift test. The test must leave the device in the same state that it was in before the test was run or in a ready state so that it can be used normally.
Status	**Coverage Area**	The diagnostic checks the status of a physical memory device, as identified by a logical memory address.
	Coverage Range	Device
	Destructive	No
	User Control	The physical component can be specified if the device supports querying status.
	Execution Time	The test shall run on the order of seconds.
	Built into Device	Yes
	Details	The implementation of this test is vendor unique, but should take into consideration the results of the last POST, Self-Test, error count trends, and vendor-specific data.
Beacon	**Coverage Area**	The diagnostic causes at least one of the LEDs of a physical component, as identified by a logical memory address, to flash on and off.
	Coverage Range	Memory LEDs
	Destructive	No

Test Name	Test Information	
	User Control	The user may specify the number of iterations or the duration that the LED blinks on and off.
	Execution Time	The test shall run on the order of seconds or minutes.
	Built into Device	Yes
	Details	The LED flash pattern is determined by the vendor, but the pattern shall be distinct from that of normal activity. The LED to be flashed may be the normal activity/status LEDs or separate LEDs provided solely for beaconing.

7.2 CIM_MemoryDiagnosticTest

The CIM_MemoryDiagnosticTest class can be used for a variety of tests necessary for diagnosing memory issues. Table 3 defines the valid property values and whether the test is mandatory or optional. An implementation may extend this class and add vendor-defined tests by using the Vendor Defined range of the MemoryTestType valuemap.

Table 4 provides additional information about the CIM_MemoryDiagnosticTest class.

365 **Table 3 – CIM_MemoryDiagnosticTest property requirements**

Test Name	Criteria	ElementName*	MemoryTestType	TestTypes*
Other (vendor extension test)	Optional	Memory<vendor extension> Test	1	(1) Other, (2) Functional, (3) Stress, (4) Health Check and/or (5) Access Test
Electrical Wiring	Mandatory	Memory Electrical Wiring Test	2	(5) Access Test
Data Bus Walking 1s	Mandatory	Memory Data Bus Walking 1s Test	3	(5) Access Test
Address Bus Walking 1s	Mandatory	Memory Address Bus Walking 1s Test	4	(5) Access Test
Power-of-Two Addressing	Mandatory	Memory Power-of-Two Addressing Test	5	(5) Access Test
Self Addressing	Optional	Memory Self Addressing Test	6	(5) Access Test
Increment and Decrement	Mandatory	Memory Increment and Decrement Test	7	(2) Functional, (5) Access Test
Moving Inversions 0s and 1s	Optional	Memory Moving Inversions 0s and 1s Test	8	(2) Functional, (5) Access Test
Moving Inversions Random	Optional	Memory Moving Inversions Random Test	9	(2) Functional, (5) Access Test
Bit Fade	Mandatory	Memory Bit Fade Test	10	(5) Stress
Reset	Mandatory	Memory Reset Test	11	(2) Functional
Self-Test	Mandatory	Memory Self-Test	12	(2) Functional
Status	Mandatory	Memory Status Test	13	(4) Health Check
Beacon	Optional	Memory Beacon Test	14	(2) Functional

366 An asterisk (*) indicates that the property is inherited from the parent class CIM_DiagnosticTest.

Table 4 – CIM_MemoryDiagnosticTest property requirements

Test Name	Characteristics*	Comment
Electrical Wiring	4 (Is Destructive)	Can detect if a memory chip is not properly connected.
Data Bus Walking 1s	4 (Is Destructive)	
Address Bus Walking 1s	4 (Is Destructive)	
Power-of-Two Addressing	4 (Is Destructive)	
Self- Addressing	4 (Is Destructive)	
Increment and Decrement	4 (Is Destructive)	
Moving Inversions 0s and 1s	4 (Is Destructive)	
Moving Inversions Random	4 (Is Destructive)	
Bit Fade	4 (Is Destructive)	
Reset	4 (Is Destructive)	
Self-Test	4 (Is Destructive) or 0 (Unknown)	This test is built-in to the device.
Status	0 (Unknown)	
Beacon	0 (Unknown)	

An asterisk (*) indicates that the property is inherited from the parent class CIM_DiagnosticTest

7.3 CIM_MemoryDiagnosticSettingData

None or one instance of the CIM_MemoryDiagnosticSettingData class may be implemented. If an instance exists, it will be associated to CIM_MemoryDiagnosticTest by using CIM_ElementSettingData. The vendor-defined default values may be specified and advertised by using this instance of CIM_MemoryDiagnosticSettingData that is referenced by the instance of CIM_ElementSettingData whose property value for IsDefault is 1 (Is Default).

If no default CIM_MemoryDiagnosticSettingData instance exists, the client must specify a setting data instance for tests that require input parameters. It is recommended that only applicable properties be specified; otherwise, alert indications may be raised.

A diagnostic test may require parameters to run. Some parameters might affect how the test is run, while other parameters provide the values to be used by the test.

The CIM_DiagnosticSettingData class contains properties that affect how a diagnostic test is run (for example, LoopControl, QuickMode); how errors are handled (for example, HaltOnError); or how results are logged (for example, LogOptions). CIM_DiagnosticSettingData is an argument to the CIM_DiagnosticTest.RunDiagnosticService( ) extrinsic method.

The client may use the vendor-defined default CIM_MemoryDiagnosticSettingData instance as an argument to the CIM_MemoryDiagnosticTest.RunDiagnosticService( ) extrinsic method. Alternatively, the client may create its own instance of CIM_MemoryDiagnosticSettingData and use it instead. If additional properties are needed that control the behavior of the diagnostic test, they should be defined in a subclass of CIM_MemoryDiagnosticSettingData.

The CIM_MemoryDiagnosticSettingData class defines the parameters that may be used by some of the System Memory tests. Table 5 lists these test parameters and shows which tests might use them. An implementation may extend this class and define additional parameters for any other vendor-defined tests.

Table 5 – CIM_MemoryDiagnosticSettingData property requirements

Test Name	ElementName*	Address[]	Address[0]	Target Device	Data Pattern	Number of Bytes	Loop Control *	Seed	Wait Time
Electrical Wiring	Memory Electrical Wiring Test	Used							
Data Bus Walking 1s	Memory Data Bus Walking 1s Test		Used						
Address Bus Walking 1s	Memory Address Bus Walking 1s Test				Used				
Power-of-Two Addressing	Memory Power-of-Two Addressing Test		Used		Used	Used			
Self Addressing	Memory Self Addressing Test								
Increment and Decrement	Memory Increment and Decrement Test		Used			Used			
Moving Inversions 0s and 1s	Memory Moving Inversions 0s and 1s Test		Used			Used			
Moving Inversions Random	Memory Moving Inversions Random Test		Used			Used	Used	Used	
Bit Fade	Memory Bit Fade Test		Used			Used			Used
Reset	Memory Reset Test			Used					
Self-Test	Memory Self-Test			Used					
Status	Memory Status Test			Used					
Beacon	Memory Beacon Test						Used		Used

An asterisk (*) indicates that the property is inherited from the parent class CIM_DiagnosticSettingData.

7.3.1 CIM_MemoryDiagnosticSettingData.Address[]

This property is an array of addresses used by a client for the following tests:

- Electrical Wiring
- DataBus Walking 1s
- Power-of-Two Addressing
- Increment and Decrement
- Moving Inversions 1s and 1s
- Moving Inversions Random
- Bit Fade

It allows the client to specify:

- Address[0] as the base address
- Address[0] and Address[1] as an address range
- Address[0], Address[1], Address[2], etc., as a discrete set of addresses

The values are used by the test to logically address memory.

The default value will depend upon the specific test. If no value is specified by the client, the vendor-defined default value will be used. The vendor-defined default value is advertised by using the default instance of CIM_MemoryDiagnosticSettingData. A null value indicates that lowest accessible address shall be used as the base address.

To test all available memory, a client can simply specify both Address[] and NumberOfBytes as null.

7.3.2 CIM_MemoryDiagnosticSettingData.TargetDevice

This property is used by a client for the Reset, Self-Test, and Status tests to specify which device they are targeting.

These tests are typically controlled through vendor-specific control lines on the device. The CIM_DiagnosticService.RunDiagnosticService( ) extrinsic method requires a reference to the managed element (local physical component or device) to be used in the test. However, to run the test, the physical selection of the device is first needed. How this selection is done depends on the memory controller. It is expected that the controller will use a dedicated set of chip selection lines. The value placed on the selection lines would be incorporated into the reference specified.

TargetDevice has no default value; that is, a value must be specified. The target is identified by a Device Moniker. (See 3.1.)

7.3.3 CIM_MemoryDiagnosticSettingData.DataPattern

This property is a value specified by the client that is to be used to test memory data access in the following tests:

- Address Bus Walking 1s
- Power-of-Two Addressing

A specific data pattern is sometimes required for the test to be effective.

The default value will depend upon the specific test. If no value is specified by the client, the vendor-defined default value will be used. The vendor-defined default value is advertised by using the default instance of CIM_MemoryDiagnosticSettingData. A null value indicates that the property does not apply to the test.

7.3.4 CIM_MemoryDiagnosticSettingData.NumberOfBytes

This property is the number of bytes specified by the client to be written and read by the following tests:

- Power-of-Two Addressing
- Increment and Decrement
- Moving Inversions 0s and 1s
- Moving Inversions Random
- Bit Fade

If a value is specified, it will indicate the number of address locations to be tested starting from the base address. If null, all available memory will be tested from the base address.

To test all available memory, a client can simply specify both Address[] and NumberOfBytes as null.

7.3.5 CIM_MemoryDiagnosticSettingData LoopControl properties

This is a set of two properties that can be used by the client to specify the number of times the test shall run until it terminates.

- LoopControl – Set to 3 to indicate that the count specified in the LoopControlParameter property should be used to perform loop control.
- LoopControlParameter – When LoopControl is 3, indicates the number of loops to perform.

These properties apply to the following tests:

- Moving Inversions Random
- Beacon

These properties can be used by the client to re-run a test any number times to stress memory.

The default values will depend upon the specific test. If no values are specified by the client, the vendor-defined default values will be used. The vendor-defined default values are advertised by using the default instance of CIM_MemoryDiagnosticSettingData. Null values indicate that a single loop will be executed.

7.3.6 CIM_MemoryDiagnosticSettingData.Seed

This property is used by the client to specify the seed for generating a random number within the following test:

- Moving Inversions Random

This property allows the client to control a test with a pseudo-random behavior. If no value is specified by the client, the vendor-defined default value will be used. The vendor-defined default value is advertised by using the default instance of CIM_MemoryDiagnosticSettingData. A null value indicates that the property does not apply to the test.

7.3.7 CIM_MemoryDiagnosticSettingData.WaitTime

This property is used by the client to specify a wait time to apply within the test execution for the following tests:

- Bit Fade
- Beacon

For example, in the Bit Fade Test this value controls how long the test will wait, after performing a memory write, before reading the data value back. In the Beacon Test it controls how long a light, an LED possibly, will remain on or off, as the case may be. When combined with the LoopControl Properties specifying a Count, it can implement a flashing lamp.

If no value is specified by the client, the vendor-defined default value will be used. The vendor-defined default value is advertised by using the default instance of CIM_MemoryDiagnosticSettingData. A null value indicates that the property does not apply to the test.

7.4 CIM_MemoryDiagnosticServiceCapabilities

None or one instance of the CIM_MemoryDiagnosticServiceCapabilities class may be implemented. If an instance exists, it will be associated to CIM_MemoryDiagnosticTest by using CIM_ElementCapabilities.

The vendor-defined capabilities of the test may be specified and advertised by using an instance of CIM_MemoryDiagnosticServicesCapabilities.

CIM_MemoryDiagnosticServicesCapabilities constrains what can be specified in an instance of the CIM_DiagnosticSettingData class.

If a CIM_MemoryDiagnosticServiceCapabilities does not exist, the client should use the default CIM_MemoryDiagnosticSettingData instance for the test.

Table 6 shows the different capabilities and to what tests they apply.

Table 6 – CIM_MemoryDiagnosticServiceCapabilities property requirements

Test Name	SupportedLoopControl*	DataPattern	Seed	WaitTime
Electrical Wiring	5 (ErrorCount) 0x8000 (No Loop Control)			
Data Bus Walking 1s	5 (ErrorCount) 0x8000 (No Loop Control)			
Address Bus Walking 1s	5 (ErrorCount) 0x8000 (No Loop Control)	Used		
Power-of-Two Addressing	5 (ErrorCount) 0x8000 (No Loop Control)	Used		
Self-Addressing	5 (ErrorCount) 0x8000 (No Loop Control)			
Increment and Decrement	5 (ErrorCount) 0x8000 (No Loop Control)			
Moving Inversions 0s and 1s	5 (ErrorCount) 0x8000 (No Loop Control)			
Moving Inversions Random	5 (ErrorCount) 3 (Count)		Used	
Bit Fade	5 (ErrorCount) 0x8000 (No Loop Control)			Used
Reset	0x8000 (No Loop Control)			
Self-Test	0x8000 (No Loop Control)			
Status	0x8000 (No Loop Control)			
Beacon	3 (Count) 4 (Timer)			

An asterisk (*) indicates that the property is inherited from the parent class CIM_DiagnosticServiceCapabilities.

7.4.1 CIM_MemoryDiagnosticServiceCapabilities.SupportedLoopControl

This array property is used by a provider for the tests shown in Table 6 to specify whether the test supports loop control.

The SupportedLoopControl property lists the loop controls that are supported by the Diagnostic Service. The values are: 0 (Unknown), 1 (Other), 2 (Continuous), 3 (Count), 4 (Timer), 5 (ErrorCount), and 0x8000 (No Loop Control).

If loop control is not supported, the value of this property is 0x8000 (No Loop Control). If the test can be run a specified number of iterations, this array property shall contain the value 3 (Count). If the test can be run in a continuous manner, this array property shall contain the value 2 (Continuous).

7.4.2 CIM_MemoryDiagnosticServiceCapabilities.DataPattern

This array property is used by a provider for those tests shown in Table 6 to specify data patterns supported by the test. Careful selection of a data pattern can have a big impact on the effectiveness of the test.

7.4.3 CIM_MemoryDiagnosticServiceCapabilities.Seed

This Boolean property is used by a provider for those tests shown in Table 6 to specify whether random number seeds are supported by the test. The seed is used to generate a random number or a sequence of random numbers. Being able to change the seed value will change the random nature of the test and consequently impact the effectiveness of the test. To replicate the same random number sequence for successive tests, one should use the same seed value.

7.4.4 CIM_MemoryDiagnosticServiceCapabilities.WaitTime

This array property is used by a provider for those tests shown in Table 6 to specify the minimum and maximum wait times supported by the test. This property is important for tests that are duration dependent, such as the Bit Fade. For example, in the case of the Bit Fade Test, it will specify the amount of time to wait before reading data after a write.

7.5 System Memory Diagnostics Profile indications support

The *System Memory Diagnostics Profile* constrains certain elements in its support for the DMTF Indications Profile. This subclause identifies those constraints.

7.5.1 CIM_IndicationFilter (StaticIndicationFilter)

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

7.5.1.1 **CIM_IndicationFilter.Name**

The *System Memory Diagnostics Profile* constrains names of the profile-defined alert indication filters as prescribed by DSP1054. The names for the indication filters are identified in the entries for the indications in Table 8. The Name property shall be formatted as defined by the following ABNF rule:

“DMTF:System Memory Diagnostics:” MessageID

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

7.5.1.2 **CIM_IndicationFilter.Query**

The *System Memory Diagnostics Profile* constrains the Query property of the profile-defined alert indication filters as prescribed by DSP1054. The Query property for indication filters are identified in the entries for the indications in Table 8.

7.5.1.3 CIM_IndicationFilter.QueryLanguage

The *System Memory Diagnostics Profile* constrains the QueryLanguage properties of the profile-defined alert indication filters as prescribed by DSP1054. The QueryLanguage properties for the indication filters are identified in the entries for the indications in Table 8.

7.5.2 CIM_FilterCollection (ProfileSpecificFilterCollection)

The *System Memory Diagnostics Profile* constrains the CollectionName property of the ProfileSpecificFilterCollection version of the CIM_FilterCollection class.

7.5.2.1 CIM_FilterCollection.CollectionName

The *System Memory Diagnostics Profile* constrains the CollectionName of the profile-defined ProfileSpecificFilterCollection filter collection as prescribed by DSP1054. The CollectionName for the filter collection shall be formatted as defined by the following ABNF rule:

"DMTF: System Memory Diagnostics:ProfileSpecifiedAlertIndicationFilterCollection"

7.5.3 CIM_MemberOfCollection (IndicationFilterInFilterCollection)

7.5.3.1 CIM_MemberOfCollection.Collection

The *System Memory Diagnostics Profile* constrains the Collection property to be the reference to the ProfileSpecificFilterCollection filter collection.

7.5.3.2 CIM_MemberOfCollection.Member

The *System Memory Diagnostics Profile* constrains the Member property to be a reference to one of the profile-defined alert indication filters.

7.5.4 CIM_OwningCollectionElement (IndicationServiceOfFilterCollection)

7.5.4.1 CIM_OwningCollectionElement.OwnedElement

The *System Memory Diagnostics Profile* constrains the OwnedElement property to be the reference to the ProfileSpecifiedFilterCollection filter collection.

7.6 Diagnostics alert indications and standard messages

7.6.1 DIAG701 – Memory Device not present

The test ran to completion, but a memory device was not present.

This alert would only be sent if the test discovers an empty memory socket in the system. The Electrical Wiring test specifically tests for this condition.

The variables in this message are:

- Diagnostic Test Name – Identifies the DiagnosticTest instance that was run. This is the Name property of the DiagnosticTest instance.
- Memory Device Moniker – Identifies a unique name for the Memory Device under test that was specified.

  This could be one of the following names:

  – The Object path of the Memory Device

  - The ElementName of the Memory Device
  - A unique, user friendly name not in the model (such as, asset name)

  The Memory Device Moniker can be any of these, but whichever one is used shall be used consistently for all Memory devices within the scoping profile.

- Physical Device Moniker – Identifies a unique name for the physical device associated with the Memory Device Moniker.

  This could be one of the following names:

  - The Object path of the physical device
  - The ElementName of the physical device
  - A unique, user friendly name not in the model (such as, asset name)

  The Physical Device Moniker can be any of these, but whichever one is used shall be used consistently for all physical devices within the scoping profile.

With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to "Memory Device Missing".

With this alert, the PerceivedSeverity shall have one of the values of 0 (Unknown), 1 (Other), 3 (Warning), 4 (Minor), 5 (Major), or 6 (Critical).

7.6.2 DIAG702 – Memory Device incorrectly connected

The test ran to completion, but a memory device was found to be incorrectly connected.

This alert would only be sent if the test discovers that the device is incorrectly inserted into the memory socket in the system. The Electrical Wiring test specifically tests for this condition.

The variables in this message are:

- Diagnostic Test Name – Identifies the DiagnosticTest instance that was run. This is the Name property of the DiagnosticTest instance.
- Memory Device Moniker – Identifies a unique name for the Memory Device under test that was specified.

  This could be one of the following names:

  - The Object path of the Memory Device
  - The ElementName of the Memory Device
  - A unique, user friendly name not in the model (such as, asset name)

  The Memory Device Moniker can be any of these, but whichever one is used shall be used consistently for all Memory devices within the scoping profile.

- Physical Device Moniker – Identifies a unique name for the physical device associated with the Memory Device Moniker.

  This could be one of the following names:

  - The Object path of the physical device
  - The ElementName of the physical device
  - A unique, user friendly name not in the model (such as, asset name)

  The Physical Device Moniker can be any of these, but whichever one is used shall be used consistently for all physical devices within the scoping profile.

With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to "Memory Device Incorrectly Connected". With this alert, the PerceivedSeverity shall have one of the values of 0 (Unknown), 1 (Other), 3 (Warning), 4 (Minor), 5 (Major), or 6 (Critical).

7.6.3 DIAG703 – Memory Device offline

The test may or may not have run to completion, but a Memory Device was found to be offline.

This alert would only be sent if the device to be exercised by the test and the OperationalStatus of the device in question was set to 10 (Stopped). For the following tests, the alert may cause the test to fail to execute to completion.

- Electrical Wiring
- Data Bus Walking 1s
- Address Bus Walking 1s
- Power-of-Two Addressing
- Self-Addressing
- Increment and Decrement
- Moving Inversions 0s and 1s
- Moving Inversions Random
- Bit Fade
- Reset
- Self-Test
- Status

If multiple devices are reported as offline, multiple alert messages are sent (one for each device that was discovered to be offline).

The variables in this message are:

- Diagnostic Test Name – Identifies the DiagnosticTest instance that was run. This is the Name property of the DiagnosticTest instance.
- Memory Device Moniker – Identifies a unique name for the Memory Device under test that was specified.

  This could be one of the following names:

  – The Object path of the Memory Device
  – The ElementName of the Memory Device
  – A unique, user friendly name not in the model (such as, asset name)

  The Memory Device Moniker can be any of these, but whichever one is used shall be used consistently for all devices within the scoping profile.
- Physical Device Moniker – Identifies a unique name for the physical device associated with the Memory Device Moniker.

  This could be one of the following names:

  – The Object path of the physical device
  – The ElementName of the physical device
  – A unique, user friendly name not in the model (such as, asset name)

  The Physical Device Moniker can be any of these, but whichever one is used shall be used consistently for all physical devices within the scoping profile.

With this alert, the AlertType shall have the value 1 (Other) or 5 (Device Alert). For tests other than Self-Test and Status, "1" indicates that the test failed because a device is offline (the OtherAlertType should be set to “Device Offline”). For Self-Test and Status tests, the “5” indicates that the test may not have executed because a needed device was offline.

With this alert, the PerceivedSeverity shall have the value 3 (Warning) if it ran to completion or 5 (Major) if it failed to run.

7.6.4 DIAG704 – Memory Device bypassed

The test may or may not have run to completion, but a Memory Device was bypassed.

This alert is only sent if the device in question was to be exercised by the test and the device was not tested. Reasons why the device was bypassed might be:

- DIAG702 – The device was offline.
- DIAG709 – The device was in error.
- DIAG710 – The device was in service.
- DIAG711 – The device was in an unrecognized state.

If the device was bypassed for one of these reasons, the appropriate DIAG message would have been sent before this message.

If the bypassed device was required by the test, this alert will cause the test to fail to execute to completion. For other tests, this alert is only a warning that one of the devices was not tested. If multiple devices are reported as bypassed, multiple alert messages are sent (one for each device that was bypassed).

The variables in this message are:

- Diagnostic Test Name – Identifies the DiagnosticTest instance that was run. This is the Name property of the DiagnosticTest instance.
- Memory Device Moniker – Identifies a unique name for the memory device under test that was specified.

  This could be one of the following names:

  – The Object path of the Memory Device
  – The ElementName of the Memory Device
  – A unique, user friendly name not in the model (such as, asset name)

  The Device Moniker can be any of these, but whichever one is used shall be used consistently for all devices within the scoping profile.
- Physical Device Moniker – Identifies a unique name for the physical device associated with the Memory Device Moniker.

  This could be one of the following names:

  – The Object path of the physical device
  – The ElementName of the physical device
  – A unique, user friendly name not in the model (such as, asset name)

  The Physical Device Moniker can be any of these, but whichever one is used shall be used consistently for all physical devices within the scoping profile.

With this alert, the AlertType shall have the value 1 (Other) or 5 (Device Alert). The OtherAlertType should be set to “Device Bypassed”.

With this alert, the PerceivedSeverity shall have the value 3 (Warning), 5 (Major), 6 (Critical), or 7 (Fatal/Nonrecoverable). If the AlertType is 1, the PerceivedSeverity shall be 3.

7.6.5 DIAG705 - Data read did not match the data written to memory

The test ran to completion, but the data read did not match the data written.

This alert would only be sent if the test was one of the following tests and the data read did not match the data written:

- Data Bus Walking 1s
- Address Bus Walking 1s
- Power-of-Two Addressing
- Self Addressing
- Increment and Decrement
- Moving Inversions 0s and 1s
- Moving Inversions Random
- Bit Fade

If multiple addresses have a mismatch, multiple alerts will be sent. The variables in this message are:

- Diagnostic Test Name – Identifies the DiagnosticTest instance that was run. This is the Name property of the DiagnosticTest instance.
- Address Value – Identifies the address at which the fault was detected.
- Write Data Value – Identifies the data value written when the fault was detected.
- Read Data Value – Identifies the data value read when the fault was detected.
- Device Moniker – Identifies a unique name for the device under test that was specified.

  This could be one of the following names:

  – The Object path of the Memory Device
  – The ElementName of the Memory Device
  – A unique, user friendly name not in the model (such as, asset name)

  The Device Moniker can be any of these, but whichever one is used shall be used consistently for all devices within the scoping profile.

- Physical Device Moniker – Identifies a unique name for the physical device associated with the Memory Device Moniker.

  This could be one of the following names:

  – The Object path of the physical device
  – The ElementName of the physical device
  – A unique, user friendly name not in the model (such as, asset name)

  The Physical Device Moniker can be any of these, but whichever one is used shall be used consistently for all physical devices within the scoping profile.

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

With this alert, the PerceivedSeverity shall have the value 5 (Major), 6 (Critical), or 7 (Fatal/Nonrecoverable).

7.6.6 DIAG706 – Unable to reset memory device

The test failed to run to completion after signaling the reset control line on a device.

This alert is only sent if the Reset Test failed to be completed.

The variables in this message are:

- Diagnostic Test Name – Identifies the DiagnosticTest instance that was run. This is the Name property of the DiagnosticTest instance.
- Device Moniker – Identifies a unique name for the device under test that was specified.

  This could be one of the following names:

  – The Object path of the Memory Device
  – The ElementName of the Memory Device
  – A unique, user friendly name not in the model (such as, asset name)

  The Device Moniker can be any of these, but whichever one is used shall be used consistently for all devices within the scoping profile.
- Physical Device Moniker – Identifies a unique name for the physical device associated with the Memory Device Moniker.

  This could be one of the following names:

  – The Object path of the physical device
  – The ElementName of the physical device
  – A unique, user friendly name not in the model (such as, asset name)

  The Physical Device Moniker can be any of these, but whichever one is used shall be used consistently for all physical devices within the scoping profile.

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

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

7.6.7 DIAG707 Memory Device failed

The test may or may not have run to completion, but a subtest failed.

This alert is only sent when a subtest fails to execute to completion.

The variables in this message are:

- Diagnostic Test Name – Identifies the DiagnosticTest instance that was run. This is the Name property of the DiagnosticTest instance.
- Device Moniker – Identifies a unique name for the device under test that was specified.

  This could be one of the following names:

  – The Object path of the Memory Device
  – The ElementName of the Memory Device
  – A unique, user friendly name not in the model (such as, asset name)

  The Device Moniker can be any of these, but whichever one is used shall be used consistently for all devices within the scoping profile.
- Physical Device Moniker – Identifies a unique name for the physical device associated with the Memory Device Moniker.

  This could be one of the following names:

  – The Object path of the physical device
  – The ElementName of the physical device
  – A unique, user friendly name not in the model (such as, asset name)

The Physical Device Moniker can be any of these, but whichever one is used shall be used consistently for all physical devices within the scoping profile.

- Failure Description – Provides a description of why the subtest failed. This can also include a DIAG standard message reference or a vendor-specific message. The Physical Device Moniker will specify the failing physical device.
- Subtest Name – Identifies the name of the subtest that reported the failure.

With this alert, the AlertType shall have the value 1 (Other) or 5 (Device Alert). If 1 (Other) is specified, the OtherAlertType should be set to "Subtest failed", but this setting did not affect execution of the requested parent test. If 5 (Device Alert) is specified, the test failed to execute.

With this alert, the PerceivedSeverity shall have the value 3 (Warning), 5 (Major), 6 (Critical), or 7 (Fatal/Nonrecoverable). If the AlertType is 1, the PerceivedSeverity shall be 3.

7.6.8 DIAG708 – Memory device in error

The test may or may not have run to completion, but a Memory Device was found in error.

This alert is only sent if the device in question was found with a status error. If multiple devices are reported as in error, multiple alert messages are sent (one for each device that was discovered to be in error).

The variables in this message are:

- Diagnostic Test Name – Identifies the DiagnosticTest instance that was run. This is the Name property of the DiagnosticTest instance.
- Device Moniker – Identifies a unique name for the device under test that was specified.

  This could be one of the following names:

  – The Object path of the Memory Device
  – The ElementName of the Memory Device
  – A unique, user friendly name not in the model (such as, asset name)

  The Device Moniker can be any of these, but whichever one is used shall be used consistently for all devices within the scoping profile.
- Physical Device Moniker – Identifies a unique name for the physical device associated with the Memory Device Moniker.

  This could be one of the following names:

  – The Object path of the physical device
  – The ElementName of the physical device
  – A unique, user friendly name not in the model (such as, asset name)

  The Physical Device Moniker can be any of these, but whichever one is used shall be used consistently for all physical devices within the scoping profile.
- Device Status – Identifies the status detected for the device.

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

With this alert, the PerceivedSeverity shall have the value 5 (Major), 6 (Critical) or 7 (Fatal/Nonrecoverable).

7.6.9 DIAG709 – Memory device in service

The test may or may not have run to completion, but a Memory Device is in service.

This alert is only sent if the device in question was to be exercised by the test was found to be in service. For example, the device may not be able to run the test because it is currently running another test or being reconfigured. These are temporary operations that will require executing the test at a later time.

If multiple Memory Devices are reported as in service, multiple alert messages are sent (one for each device that was discovered to be in service).

Alert DIAG704 Memory Device bypassed may also be raised with this alert.

The variables in this message are:

- Diagnostic Test Name – Identifies the DiagnosticTest instance that was run. This is the Name property of the DiagnosticTest instance.
- Device Moniker – Identifies a unique name for the device under test that was specified.

  This could be one of the following names:

  – The Object path of the System Memory
  – The ElementName of the System Memory
  – A unique, user friendly name not in the model (such as, asset name)

  The Device Moniker can be any of these, but whichever one is used shall be used consistently for all devices within the scoping profile.
- Physical Device Moniker – Identifies a unique name for the physical device associated with the Memory Device Moniker.

  This could be one of the following names:

  – The Object path of the physical device
  – The ElementName of the physical device
  – A unique, user friendly name not in the model (such as, asset name)

  The Physical Device Moniker can be any of these, but whichever one is used shall be used consistently for all physical devices within the scoping profile.
- Service Action – Identifies the temporary service that is in progress. Possible values are
  - “Reconfigure”
  - “Testing”

With this alert, the AlertType shall have the value 5 (Device Alert). For tests other than Status “5”, this value indicates that the tests failed because a needed device was in service.

With this alert, the PerceivedSeverity shall have the value 3 (Warning) if DIAG704 was sent or 4 (Minor) if DIAG704 was not sent.

7.6.10 DIAG710 – Memory Device was in an unrecognized state

The test may or may not have run to completion, but a Memory Device is in an unrecognized state.

This alert is only sent if the device in question was to be exercised by the test. For the following tests this alert may cause the test to fail to execute to completion:

- Reset
- Self-Test

- Status
- Beacon

For other tests, this alert is only a warning that one of the devices was not tested. If multiple devices are reported as in an unrecognized state, multiple alert messages are sent (one for each device that was discovered to be in an unrecognized state).

Alert DIAG704 Memory Device bypassed may also be raised with this alert.

The variables in this message are:

- Diagnostic Test Name – Identifies the DiagnosticTest instance that was run. This is the Name property of the DiagnosticTest instance.
- Device Moniker – Identifies a unique name for the device under test that was specified.

  This could be one of the following names:

  – The Object path of the System Memory
  – The ElementName of the System Memory
  – A unique, user friendly name not in the model (such as, asset name)

  The Device Moniker can be any of these, but whichever one is used shall be used consistently for all devices within the scoping profile.
- Physical Device Moniker – Identifies a unique name for the physical device associated with the Memory Device Moniker.

  This could be one of the following names:

  – The Object path of the physical device
  – The ElementName of the physical device
  – A unique, user friendly name not in the model (such as, asset name)

  The Physical Device Moniker can be any of these, but whichever one is used shall be used consistently for all physical devices within the scoping profile.
- Device State – Identifies the state for the Memory Device that is in an unrecognized state

With this alert, the AlertType shall have the value 1 (Other) or 5 (Device Alert). For tests other than Reset, Self-Test, Status, and Beacon, “1” indicates that a device was in an unrecognized state (the OtherAlertType should be set to “Device in Unrecognized State”). For Reset, Self-Test, Status, and Beacon tests, the “5” indicates that the test failed because a needed device was in an unrecognized state.

With this alert, the PerceivedSeverity shall have the value 3 (Warning) if DIAG704 was sent or 4 (Minor) if DIAG704 was not sent.

7.6.11 System Memory alerts using common messages

In addition to the alert standard messages that are unique to System Memory, the *System Memory Diagnostics Profile* may also generate common diagnostic messages (including diagnostic job control messages). Of specific note, the *System Memory Diagnostics Profile* may generate completion status messages (such as DIAG0, DIAG3 or DIAG4) and job-related standard messages (such as DIAG19 or DIAG20).

In addition, the implementation may generate common messages such as DIAG43 or DIAG50 to cover capabilities or settings alerts.

7.6.11.1 **Common completion status messages**

The *System Memory Diagnostics Profile* should generate completion status messages to reflect the completion of the test (see DSP1002). These messages would include:

- DIAG0 - The test passed.
- DIAG3 - The device test failed.
- DIAG4 - The test was completed with warnings.
- DIAG44 - The test did not start.
- DIAG45 - The test aborted.

7.6.11.2 **Diagnostic Job Control messages**

The *System Memory Diagnostics Profile* should generate messages associated with the Diagnostic Job Control Profile (see DSP1119). The messages would include:

- DIAG9 - Test continued after last interactive timeout using Default Values.
- DIAG12 - Job could not be started.
- DIAG19 - Test killed by client.
- DIAG20 - Test terminated by client.
- DIAG21 - Test suspended by client.
- DIAG34 - Request for Inputs.
- DIAG35 - Request for action.
- DIAG36 - Test killed by test.
- DIAG37 - Test terminated by test.
- DIAG38 - Test resumed by client.
- DIAG39 - JobSetting reset.
- DIAG40 - JobSetting defaults not used.
- DIAG48 - Test continued after an interim interactive timeout.
- DIAG49 - Test terminated after an interactive timeout.

7.6.11.3 **Settings alert messages**

Errors in values supplied in the DiagnosticSettings parameter (an embedded instance of MemoryDiagnosticSettingData) of the RunDiagnosticService method would be reported by using DIAG43 (The Requested DiagnosticSettings is not supported).

The DIAG43 message has the following format:

> The <Diagnostic Test Name> test on the selected Element to test <Element Moniker> ran but the requested DiagnosticSettings property <DiagnosticSettings Property> of <DiagnosticSettings Value> is not supported. The value <DiagnosticSettings Used> was used instead.

The Element Moniker would be the Device Moniker. The <DiagnosticSettings Property> could be any one of the MemoryDiagnosticSettingData properties:

- ElementName
- Address[]
- Target Device
- Data Pattern
- Number of Bytes
- Loop Control
- Seed
- Wait Time

.

The <DiagnosticSettings Value> would be the value supplied for the property. This is the value that was not supported. The <DiagnosticSettings Used> would be the value that the test used instead of the value that was supplied.

7.6.11.4 **Capabilities alert messages**

Errors in properties supplied in the DiagnosticSettings parameter (an embedded instance of MemoryDiagnosticSettingData) of the RunDiagnosticService method would be reported by using DIAG50 (Capability to set the DiagnosticSettings parameter not supported for the test).

The DIAG50 message has the following format:

> The <Diagnostic Test Name> test on the selected element to test <Element Moniker> ran, but DiagnosticSettings parameter requested <Diag Setting Property> is not a supported capability and was ignored.

The Element Moniker would be the Memory Device Moniker. <Diag Setting Property> could be any one of the MemoryDiagnosticSettingData properties:

- ElementName
- Address[]
- Target Device
- Data Pattern
- Number of Bytes
- Loop Control
- Seed
- Wait Time

The message means that the parameter (property) does not apply to the test and was ignored.

7.6.11.5 **Other common messages**

In addition, the *System Memory Diagnostics Profile* may also generate other common messages (see DSP1002).

7.6.12 DIAG50 - Capability to set the DiagnosticSettings parameter not supported for test

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

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

The variables in this message are:

- Diag Setting Property – Identifies the property that was set, but not supported for the test
- Diagnostic Test Name – Identifies the DiagnosticTest instance that was run. This is the Name property of the DiagnosticTest instance.
- Element Moniker – Identifies a unique name for the element under test (such as, Memory Device) that was specified.

  This could be one of the following:

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

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

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

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

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_MemoryDiagnosticTest.RunDiagnosticService( )

The RunDiagnosticService( ) method shall return one of the return code values defined in the DSP1002, Table 2 – RunDiagnosticsService( ) Method: Return Code Values.

When failures occur during the execution of a diagnostic test, the failure shall be recorded in the instance of CIM_DiagnosticServiceRecord that is associated with the test. The reason for the failure shall be recorded in CIM_DiagnosticServiceRecord.ErrorCode[ ], and the corresponding CIM_DiagnosticServiceRecord.ErrorCount[ ] shall be incremented. Other occurrences of the same failure during the same test shall not create additional entries in CIM_DiagnosticServiceRecord.ErrorCode[ ], but they shall cause the corresponding CIM_DiagnosticServiceRecord.ErrorCount[ ] to be incremented.

8.2 Profile conventions for operations

Support for operations for each profile class (including associations) shall be as mandated in clause 8 of DSP1002.

9 Use cases

9.1 Overview

This clause contains use cases for the *System Memory Diagnostics Profile*.

How to discover, configure, and run the individual diagnostic tests is detailed in DSP1002. This clause focuses on how to use the System Memory diagnostic tests to diagnose common system issues.

9.2 Use case summary

Table 7 summarizes the use cases that are described in this clause. The use cases are categorized and named, and references are provided to the subclause that describes the use case.

Table 7 – System Memory Diagnostics Profile use cases

Category	Use Case Name	Description
Verifying System Memory Health See 9.3.	Verify Health	Verify the health of System Memory without impacting system access to it. See 9.3.1.
	Verify Hardware	Examine System Memory to discover any hardware issues. See 9.3.2.
	Identify Device	Make System Memory easy to physically identify. See 9.3.3.
Troubleshooting System Memory Issues See 9.4.	Verify Device Accessibility	Verify that a Memory device in System Memory is accessible. See 9.4.1
	Stress Test	Create a high volume of traffic to a particular Memory device to help uncover System Memory issues. See 9.4.2.
	Troubleshoot Addressing	Discover why an address location can no longer be accessed. See 9.4.3.
	Troubleshoot Data Access	Discover why an address location can no longer be accessed. See 9.4.4.

9.3 Verifying System Memory health

The use cases in this clause describe how the client can use the diagnostic tests to verify the health of System Memory devices and to locate them.

9.3.1 Verify health

To substantiate that System Memory is healthy and not developing problems, without disrupting the functioning of the host system, the client can use Status Test.

9.3.2 Verify hardware

The client can confirm that the System Memory hardware is functioning properly with the following procedure:

1) Use the Electrical Wiring Test to check for the existence of physical memory devices in the system. It can determine missing or incorrectly connected memory chips.
2) Use the Status Test to get the current status of the memory device.
3) Use the Self-Test to verify the functionality of the memory devices. This test covers all internal components.
4) Use the Data Bus Walking 1s Test to verify that the data path to the memory device is working properly.
5) Use the Address Bus Walking 1s Test to verify that the address path to the memory device is working properly.

6) Use one or more of the following tests to verify address path, data path and memory device.
   - Self Addressing
   - Increment and Decrement
   - Moving Inversions 0s and 1s
   - Moving Inversions Random
   - Bit Fade

9.3.3 Identify device

When it has been determined that a particular System Memory device has to be replaced, the client can use the Beacon Test to cause the Memory Device LED to flash. The LEDs make it easy to visually identify the defective device in a host system with multiple devices.

9.4 Troubleshooting System Memory issues

The use cases in this clause describe how the client can use the diagnostic tests to isolate problems occurring with memory in the system.

9.4.1 Verify device accessibility

The client can use Electrical Wiring test to verify that a particular memory device can be physically accessed.

9.4.2 Stress test

Some problems only occur when there are high levels of data access to and from the device. To help reproduce memory access problems, clients can use the Moving Inversions Random Test. By specifying Address[] and Number of Bytes set to null, all available memory will be tested. Specifying different Random Number Seed values and high loop counts will generate a large amount of varying memory accesses.

9.4.3 Troubleshoot addressing

There are many reasons why memory may not be addressable: a device could be pulled out, broken, or in a state that prevents communication with it. Clients can use the following procedure to discover where the problem lies:

1) Use the Electrical Wiring Test to check for the existence of physical memory devices in the system. If the test passes, the memory device is not missing or incorrectly connected and the next test should be run.
2) Use the Status Test to get the current status of the memory device. If the returned status indicates that the device is healthy, run the next test.
3) Use the Self-Test to verify the functionality of the memory devices. This test covers all internal components. If the result does not indicate a malfunction in the device, run the next test.
4) Use the Address Bus Walking 1s Test to verify that the address path to the memory device is working properly. If the test fails, the test will indicate the address bus line with the problem.

9.4.4 Troubleshoot data access

There are many reasons why memory data may not be accessible: a device could be pulled out, broken, or in a state that prevents communication with it. Clients can use the following procedure to discover where the problem lies:

1) Use the Electrical Wiring Test to check for the existence of physical memory devices in the system. If the test passes, the memory device is not missing or incorrectly connected and the next test should be run.
2) Use the Status Test to get the current status of the memory device. If the returned status indicates that the device is healthy, run the next test.
3) Use the Self-Test to verify the functionality of the memory devices. This test covers all internal components. If the result does not indicate a malfunction in the device, run the next test.
4) Use the Data Bus Walking 1s Test to verify that the data path to the memory device is working properly. If the test fails, the test will indicate the data bus line with the problem.
5) Use the Bit Fade Test to verify that the device is working properly.

10 CIM elements

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

Table 8 – CIM elements: System Memory Diagnostics Profile

Element Name	Requirement	Description
Classes		
CIM_MemoryDiagnosticTest	Mandatory	See 10.1.
CIM_MemoryDiagnosticSettingData (Default)	Optional	See 10.2.
CIM_MemoryDiagnosticSettingData (Client)	Optional	See 10.2.
CIM_MemoryDiagnosticServiceCapabilities	Optional	See 10.3.
CIM_RegisteredProfile	Mandatory	See 10.4.
CIM_AffectedJobElement	Optional	See 10.5.
CIM_AvailableDiagnosticService	Mandatory	See 10.6.
CIM_ElementCapabilities	Optional	See 10.7.
CIM_ElementSettingData (DiagnosticSettingData)	Optional	See 10.8.
CIM_ElementSettingData (JobSettingData)	Optional	See 10.9.
CIM_ElementSoftwareIdentity	Mandatory	See 10.10.
CIM_HostedService	Mandatory	See 10.11.
CIM_OwningJobElement	Mandatory	See 10.12.
CIM_RecordAppliesToElement	Optional	See 10.13.
CIM_ServiceAffectsElement	Mandatory	See 10.14.
CIM_ServiceAvailableToElement	Mandatory	See 10.15.
CIM_ServiceComponent	Optional	See 10.16.

Element Name	Requirement	Description
CIM_UseOfLog	Mandatory	See 10.17.
CIM_FilterCollection	Optional	See 10.18.
CIM_IndicationFilter	Mandatory	See 10.19.
CIM_MemberOfCollection	Optional	See 10.20.
CIM_OwningCollectionElement	Optional	See 10.21.
Indications		
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG701"	Optional	Query Language = ”DMTF:CQL” Name = “DMTF: System Memory Diagnostics: DIAG701” See 7.6.1.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG702"	Optional	Query Language = ”DMTF:CQL” Name = “DMTF: System Memory Diagnostics:DIAG702” See 7.6.2.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG703"	Optional	Query Language = ”DMTF:CQL” Name = “DMTF: System Memory Diagnostics:DIAG703” See 7.6.3.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG704"	Optional	Query Language = ”DMTF:CQL” Name = “DMTF: System Memory Diagnostics:DIAG704” See 7.6.4.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG705"	Optional	Query Language = ”DMTF:CQL” Name = “DMTF: System Memory Diagnostics:DIAG705” See 7.6.6.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG706"	Optional	Query Language = ”DMTF:CQL” Name = “DMTF: System Memory Diagnostics:DIAG706” See 7.6.7.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG707"	Optional	Query Language = ”DMTF:CQL” Name = “DMTF: System Memory Diagnostics:DIAG707” See 7.6.7.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG708"	Optional	Query Language = ”DMTF:CQL” Name = “DMTF: System Memory Diagnostics:DIAG708” See 7.6.8.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG709"	Optional	Query Language = ”DMTF:CQL” Name = “DMTF: System Memory Diagnostics:DIAG709” See 7.6.9.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG710"	Optional	Query Language = ”DMTF:CQL” Name = “DMTF: System Memory Diagnostics:DIAG710” See 7.6.10.

10.1 CIM_MemoryDiagnosticTest

The CIM_MemoryDiagnosticTest class is used to represent the Diagnostic Testing for System Memory. This class specializes CIM_DiagnosticTest as defined in DSP1002. The constraints listed in Table 9 are in addition to those specified in DSP1002. See DSP1002 for other mandatory elements that must be implemented.

Table 9 – Class: CIM_MemoryDiagnosticTest

Elements	Requirement	Notes
ElementName	Mandatory	See 7.2.
Characteristics	Mandatory	See 7.2.
OtherCharacteristicsDescriptions	Conditional	If Characteristics includes the value of 1 (Other), this property is Mandatory.
MemoryTestType	Mandatory	See 7.2.
OtherMemoryTestTypeDescription	Conditional	If MemoryTestType has a value of 1 (Other), this property is Mandatory.
TestTypes	Optional	See 7.2.

10.2 CIM_MemoryDiagnosticSettingData

The CIM_MemoryDiagnosticSettingData class is used to pass in test parameters and to specify other test control parameters. This class specializes CIM_DiagnosticSettingData as defined in DSP1002. The constraints listed in Table 10 are in addition to those specified in DSP1002. See DSP1002 for other mandatory elements that must be implemented.

Table 10 – Class: CIM_MemoryDiagnosticSettingData

Elements	Requirement	Notes
ElementName	Mandatory	See 7.3.
Address[]	Optional	See 7.3.1.
TargetDevice	Optional	See 7.3.1.
DataPattern	Optional	See 7.3.3.
NumberOfBytes	Optional	See 7.3.4.
LoopControl	Optional	See 7.3.5.
LoopControlParameter	Optional	See 7.3.5.
Seed	Optional	See 7.3.6.
WaitTime	Optional	See 7.3.7.

10.3 CIM_MemoryDiagnosticServiceCapabilities

The CIM_MemoryDiagnosticServiceCapabilities class is used to provide information on the capabilities for the Memory Diagnostic Service. This class specializes CIM_DiagnosticServiceCapabilities as defined in DSP1002. The constraints listed in Table 11 are in addition to those specified in DSP1002. See DSP1002 for other mandatory elements that must be implemented.

Table 11 – Class: CIM_MemoryDiagnosticServiceCapabilities

Elements	Requirement	Notes
ElementName	Mandatory	See 7.4.
SupportedLoopControl[]	Optional	See 7.4.1.
DataPattern[]	Optional	See 7.4.2.
Seed	Optional	See 7.4.3.
WaitTime[]	Optional	See 7.4.3.

10.4 CIM_RegisteredProfile

The CIM_RegisteredProfile class is defined by the *Profile Registration Profile* (DSP1033). The requirements denoted in Table 12 are in addition to those mandated by DSP1033. See DSP1033 for the other mandatory elements that must be implemented.

Table 12 – Class: CIM_RegisteredProfile

Elements	Requirement	Notes
RegisteredName	Mandatory	The value of this property shall be “System Memory Diagnostics”.
RegisteredVersion	Mandatory	The value of this property shall be “1.0.0”.
RegisteredOrganization	Mandatory	The value of this property shall be 2 (DMTF).

10.5 CIM_AffectedJobElement

Although defined in DSP1002, the CIM_AffectedJobElement class is listed here because the AffectedElement reference is scoped down to CIM_Memory, which is a subclass of CIM_ManagedElement. The constraints listed in Table 13 are in addition to those specified in DSP1002. See DSP1002 for other mandatory properties of CIM_AffectedJobElement that must be implemented.

Table 13 – Class: CIM_AffectedJobElement

Properties	Requirement	Notes
AffectedElement (overridden)	Mandatory	The property shall be a reference to an instance of CIM_Memory.
AffectingElement	Mandatory	The property shall be a reference to an instance of CIM_ConcreteJob.

10.6 CIM_AvailableDiagnosticService

Although defined in DSP1002, the CIM_AvailableDiagnosticService class is listed here because the ServiceProvided reference is scoped down to CIM_MemoryDiagnosticTest, which is a subclass of CIM_DiagnosticTest, and the UserOfService reference is scoped down to CIM_Memory, which is a subclass of CIM_ManagedElement. The constraints listed in Table 14 are in addition to those specified in DSP1002. See DSP1002 for other mandatory properties of CIM_AvailableDiagnosticService that must be implemented.

Table 14 – Class: CIM_AvailableDiagnosticService

Properties	Requirement	Notes
ServiceProvided (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticTest.
UserOfService (overridden)	Mandatory	The property shall be a reference to an instance of CIM_Memory or CIM_PhysicalMemory.

10.7 CIM_ElementCapabilities

Although defined in DSP1002, the CIM_ElementCapabilities class is listed here because the ManagedElement reference is scoped down to CIM_MemoryDiagnosticTest, which is a subclass of CIM_DiagnosticTest, and the Capabilities reference is scoped down to CIM_MemoryDiagnosticServiceCapabilities, which is a subclass of CIM_DiagnosticServiceCapabilities. The constraints listed in Table 15 are in addition to those specified in DSP1002. See DSP1002 for other mandatory properties of CIM_ElementCapabilities that must be implemented.

Table 15 – Class: CIM_ElementCapabilities

Properties	Requirement	Notes
ManagedElement (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticTest.
Capabilities (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticServiceCapabilities.

10.8 CIM_ElementSettingData (DiagnosticSettingData)

Although defined in DSP1002, the CIM_ElementSettingData class is listed here because the ManagedElement reference is scoped down to CIM_MemoryDiagnosticTest, which is a subclass of CIM_DiagnosticTest, and the SettingData reference is scoped down to CIM_MemoryDiagnosticSettingData, which is a subclass of CIM_DiagnosticSettingData. The constraints listed in Table 16 are in addition to those specified in DSP1002. See DSP1002 for other mandatory properties of CIM_ElementSettingData that must be implemented.

Table 16 – Class: CIM_ElementSettingData

Properties	Requirement	Notes
ManagedElement (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticTest.
SettingData (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticSettingData.
IsDefault	Mandatory	If the instance of CIM_MemoryDiagnosticSettingData is the default setting, this property shall have the value of TRUE.

10.9 CIM_ElementSettingData (JobSettingData)

Although defined in DSP1002, the CIM_ElementSettingData class is listed here because the Dependent reference is scoped down to CIM_MemoryDiagnosticTest, which is a subclass of CIM_DiagnosticTest, and the SettingData reference is scoped down to CIM_JobSettingData, which is a subclass of

CIM_SettingData. The constraints listed in Table 17 are in addition to those specified in DSP1002. See DSP1002 for other mandatory properties of CIM_ElementSettingData that must be implemented.

Table 17 – Class: CIM_ElementSettingData

Properties	Requirement	Notes
ManagedElement (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticTest.
SettingData (overridden)	Mandatory	The property shall be a reference to an instance of CIM_JobSettingData.
IsDefault	Mandatory	If the instance of CIM_JobSettingData is the default setting, this property shall have the value of TRUE.

10.10 CIM_ElementSoftwareIdentity

Although defined in DSP1002, the CIM_ElementSoftwareIdentity class is listed here because the Dependent reference is scoped down to CIM_MemoryDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 18 are in addition to those specified in DSP1002. See DSP1002 for other mandatory properties of CIM_ElementSoftwareIdentity that must be implemented.

Table 18 – Class: CIM_ElementSoftwareIdentity

Properties	Requirement	Notes
Antecedent	Mandatory	The property shall be a reference to an instance of CIM_SoftwareIdentity.
Dependent (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticTest.

10.11 CIM_HostedService

Although defined in DSP1002, the CIM_HostedService class is listed here because the Dependent reference is scoped down to CIM_MemoryDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 19 are in addition to those specified in DSP1002. See DSP1002 for other mandatory properties of CIM_HostedService that must be implemented.

Table 19 – Class: CIM_HostedService

Properties	Requirement	Notes
Antecedent	Mandatory	The property shall be a reference to an instance of CIM_ComputerSystem.
Dependent (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticTest.

10.12 CIM_OwningJobElement

Although defined in DSP1119 and referenced in DSP1002, the CIM_OwningJobElement class is listed here because the OwningElement reference is scoped down to CIM_MemoryDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 20 are in addition to those specified in DSP1119. See DSP1119 for other mandatory properties of CIM_OwningJobElement that must be implemented.

Table 20 – Class: CIM_OwningJobElement

Properties	Requirement	Notes
OwningElement (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticTest.
OwnedElement	Mandatory	The property shall be a reference to an instance of CIM_ConcreteJob.

10.13 CIM_RecordAppliesToElement

Although defined in DSP1002, the CIM_RecordAppliesToElement class is listed here because the Dependent reference is scoped down to CIM_MemoryDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 21 are in addition to those specified in DSP1002. See DSP1002 for other mandatory properties of CIM_RecordAppliesToElement that must be implemented.

Table 21 – Class: CIM_RecordAppliesToElement

Properties	Requirement	Notes
Antecedent	Mandatory	The property shall be a reference to an instance of CIM_DiagnosticRecord.
Dependent (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticTest.

10.14 CIM_ServiceAffectsElement

Although defined in DSP1002, the CIM_ServiceAffectsElement class is listed here because the AffectedElement reference is scoped down to CIM_Memory or CIM_PhysicalMemory, which is a subclass of CIM_ManagedElement, and the AffectingElement reference is scoped down to CIM_MemoryDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 22 are in addition to those specified in DSP1002. See DSP1002 for other mandatory properties of CIM_ServiceAffectsElement that must be implemented.

Table 22 – Class: CIM_ServiceAffectsElement

Properties	Requirement	Notes
AffectedElement (overridden)	Mandatory	The property shall be a reference to an instance of CIM_Memory or CIM_PhysicalMemory.
AffectingElement (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticTest.

10.15 CIM_ServiceAvailableToElement

Although defined in DSP1002, the CIM_ServiceAvailableToElement class is listed here because the UserOfService reference is scoped down to CIM_MemoryDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 23 are in addition to those specified in DSP1002. See DSP1002 for other mandatory properties of CIM_ServiceAvailableToElement that must be implemented.

Table 23 – Class: CIM_ServiceAvailableToElement

Properties	Requirement	Notes
ServiceProvided	Mandatory	The property shall be a reference to an instance of CIM_HelpService.
UserOfService (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticTest.

10.16 CIM_ServiceComponent

Although defined in DSP1002, the CIM_ServiceComponent class is listed here because the GroupComponent reference is scoped down to CIM_MemoryDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 24 are in addition to those specified in DSP1002. See DSP1002 for other mandatory properties of CIM_ServiceComponent that must be implemented.

Table 24 – Class: CIM_ServiceComponent

Properties	Requirement	Notes
GroupComponent (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticTest.
PartComponent	Mandatory	The property shall be a reference to an instance of CIM_DiagnosticService.

10.17 CIM_UseOfLog

Although defined in DSP1002, the CIM_UseOfLog class is listed here because the Dependent reference is scoped down to CIM_MemoryDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 25 are in addition to those specified in DSP1002. See DSP1002 for other mandatory properties of CIM_UseOfLog that must be implemented.

Table 25 – Class: CIM_UseOfLog

Properties	Requirement	Notes
Antecedent	Mandatory	The property shall be a reference to an instance of CIM_DiagnosticLog.
Dependent (overridden)	Mandatory	The property shall be a reference to an instance of CIM_MemoryDiagnosticTest.

10.18 CIM_FilterCollection

CIM_FilterCollection represents an instance of the ProfileSpecificFilterCollection adaptation as defined in DSP1054. It defines the collection of all the alert indications of the *System Memory Diagnostics Profile*. Table 26 contains the requirements for elements of this class.

Table 26 – Class: CIM_FilterCollection

Properties	Requirement	Notes
InstanceID	Mandatory	**Key**: See DSP1054.
CollectionName (overridden)	Mandatory	The property shall be “DMTF:System Memory Diagnostics: ProfileSpecifiedAlertIndicationFilterCollection”.

10.19 CIM_IndicationFilter

CIM_IndicationFilter represents a StaticIndicationFilter as defined in DSP1054. It defines the format of all the alert indication filters of the *System Memory Diagnostics Profile*. Table 27 contains the requirements for elements of this class.

Table 27 – Class: CIM_IndicationFilter

Properties	Requirement	Notes
Name	Mandatory	**Key**: See the Name values as identified in Table 8.
CreationClassName	Mandatory	**Key**: See DSP1054.
SystemName	Mandatory	**Key**: See DSP1054.
SystemCreationClassName	Mandatory	**Key**: See DSP1054.
SourceNamespaces[]	Mandatory	See DSP1054.
IndividualSubscriptionSupported	Mandatory	See DSP1054.
Query (overridden)	Mandatory	See the Query values as identified in Table 8.
QueryLanguage (overridden)	Mandatory	See the QueryLanguage values as identified in Table 8.

10.20 CIM_MemberOfCollection

CIM_MemberOfCollection represents an association between the profile specific FilterCollection and the CIM_IndicationFilters for the alert indications. Table 28 contains the requirements for elements of this class.

Table 28 – Class: CIM_MemberOfCollection

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

10.21 CIM_OwningCollectionElement

CIM_OwningCollectionElement represents an association between the IndicationService that controls the profile specific FilterCollection and the profile specific CIM_FilterCollection for the alert indication filters. Table 29 contains the requirements for elements of this class.

Table 29 – Class: CIM_OwningCollectionElement

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

ANNEX A (informative)

Change log

Version	Date	Description
0.1	2014-04-29	Initial Version
0.2	2014-06-10	Updated
0.3	2014-09-24	Completed first pass review
0.4	2014-09-30	Updated for Work in Progress version
0.5	2014-10-08	Updates for Work in Progress version
1.0.0a	2015-02-25	BrightLeaf Group scrub
1.0.0a	2015-04-16	DIAG WG review of Brightleaf Group scrub – Updated for WIP approval/processing