Document Number: DSP1105

Date: 2011-06-30

Version: 1.0.0

CPU Diagnostics Profile

Document Type: Specification

Document Status: DMTF Standard

Document Language: en-US

CONTENTS

Figures

Tables

Foreword

The CPU Diagnostics Profile (DSP1105) 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:

Editors:

- Andre Asselin– IBM Corporation
- Rodney Brown – IBM Corporation
- Carl Chan – WBEM Solutions, Inc.

Participants:

- Carl Chan – WBEM Solutions, Inc.
- Rodney Brown – IBM Corporation
- Ken Kotyuk – Hewlett-Packard Company
- Kevin Kuelbs – Hewlett-Packard Company
- Eric Tend – Hewlett-Packard Company
- Dave Barrett – Emulex
- Mike Lowe – Advanced Micro Devices

Introduction

A *profile* is a collection of Common Information Model (CIM) elements and behavior rules that represents 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 Optical Drive 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 enables seamless integration of vendor-supplied diagnostic services into system management frameworks.

The goal of the *CPU Diagnostics Profile* is to define industry-standard building blocks that enable seamless problem determination support for CPUs. The profile extends the standard diagnostic profile by identifying a base set of CPU 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 proprietary features of the CPU.

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.

CPU Diagnostics Profile

1 Scope

The *CPU Diagnostics Profile* defines the set of classes, properties, methods, and default values needed to perform effective problem determination for processors within a management domain. The set of classes that model CPU presence and CPU characteristics are not described within the scope of this profile.

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

2 Normative References

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

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) 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 DSP1001, *Management Profile Specification Usage Guide 1.0*,
http://dmtf.org/sites/default/files/standards/documents/DSP1001_1.0.pdf

DMTF DSP1002, *Diagnostics Profile 2.0*,
http://dmtf.org/sites/default/files/standards/documents/DSP1002_2.0.pdf

DMTF DSP1022, *CPU Profile 1.0*
http://dmtf.org/sites/default/files/standards/documents/DSP1022_1.0.pdf

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

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.

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
CPU
Central Processor Unit

4.5
CRU
Customer Replaceable Unit

4.6
FPU
Floatiing Point Unit

4.7
FRU
Field Replaceable Unit

4.8
IPC
Inter Processor Communication

4.9
LBA
Logical Block Addressing

4.10
ME
Managed Element

4.11
MMX
Matrix Math Extensions instructions using 64-bit registers to support floating point operations

4.12
MOF
Managed Object Format

4.13
OS
Operating System

4.14
PD
Problem Determination

4.15
PFA
Predictive Failure Analysis

4.16
POST
Power-On Self Test

4.17
RAS
Reliability, Availability, Serviceability

4.18
QA
Quality Assurance

4.19
SIMD
Single Instruction Multiple Data instructions used to support parallel computing

4.20
SLP
Service Location Protocol

4.21
SSE
Streaming SIMD Extension instructions using 128-bit registers to support floating point operations

4.22
SSE2
Second-generation SSE instructions, which adds cache control instructions and improved operation for an OS running in 64-bit mode

4.23

SSE3
Third-generation SSE instructions, which adds the capability to work horizontally in a register

4.24
WBEM
Web-Based Enterprise Management

5 Synopsis

CPU Diagnostics

Version: 1.0.0

Organization: DMTF

CIM schema version: 2.28

Central Class: CIM_CPUDiagnosticTest

Scoping Class: CIM_ComputerSystem

Specializes: Diagnostics Profile version 2.0.0

The *CPU Diagnostics Profile* extends the management capability of referenced profiles by adding common methods for determining that the state of managed processors in a system is optimal.

CIM_CPUDiagnosticTest shall be the central class of this profile. The instance of CIM_CPUDiagnosticTest 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_Processor, CIM_ProcessorCore or CIM_HardwareThread or a subclass of them.

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

Table 1 – Referenced Profiles

Profile Name	Organization	Version	Description
Diagnostics	DMTF	2.0	Specializes
Profile Registration	DMTF	1.0	Mandatory
CPU	DMTF	1.0	Optional

6 Description

Diagnostic programs can be developed to support two primary diagnostic modes.

One mode tests the CPU in an operational state after its operating system has started. In this mode, diagnostic tests exercise various functional components or collect metrics within the context of a running system. Typically, most diagnostics in this mode are launched concurrently with other user programs atop a fully functioning general purpose operating system. Such diagnostics will test functional features (for example, floating point instructions) and RAS features (for example, stress tests). Testing of operating system functions and other low level component testing is not conducted in this environment because it would disrupt the normal usage of the system.

The other mode tests the CPU in a preboot state before a general purpose operating system has been started. In this mode, it is understood that the system is not under normal usage. Thus, invasive and destructive tests can be executed. Typically, diagnostics are launched in this environment for manufacturing quality assurance to test operating system functions and other low-level components. Diagnostics are also run in this mode when serious component errors are suspected in a commercial environment. In either scenario, one cannot assume that even basic OS functions or low-level components (for example, registers) will perform properly. Thus, a small limited function OS may be required to execute some pre-boot diagnostic tests.

There may also be a third type of hybrid diagnostic test that is able to provide reduced levels of coverage in a normal running environment and enhanced coverage in a preboot environment.

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

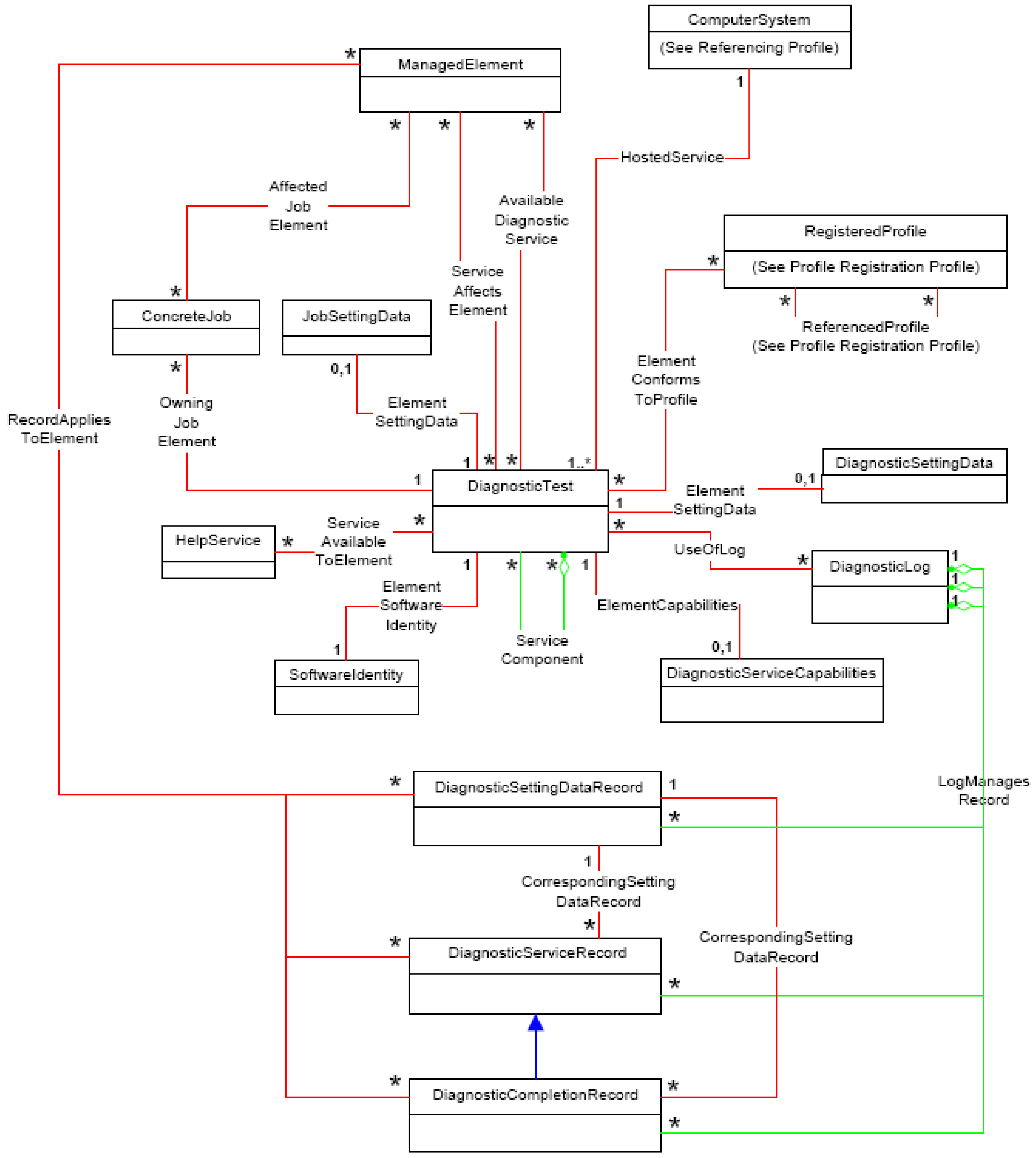

Figure 1 – CPU Diagnostics Profile: Profile Class Diagram

7 Implementation

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

7.1 CPU Test Information

This clause outlines the CPU diagnostic categories and test types. While CPU architectures may differ, the intent is to provide an outline that should be applicable to any CPU architecture. The tests are grouped into the following five categories and they are recommended to be run in a bootstrapping manner as outlined below.

1) Basic Functionality tests test resource access (registers and memory), arithmetic operations, and control operations. This set of tests should be run first in a pre-production or specialized minimal operating system. This set of tests can then be run again in a standard OS environment upon completion of OS services.

2) OS Services tests require a pre-production or specialized operating system to allow control of resources that are restricted from use by application programs.

3) RAS (Reliability, Accessibility, and Serviceability) tests test functions of the CPU that are used to assure proper operation, and interrupt/exception handling

4) Power/Performance tests assure that the CPU can change frequency and voltage for power/performance tuning.

5) System Stress and Coherency tests include I/O interfaces, internal caches, and Inter-Processor Communication (and/or Inter-Core Communication for multi-core CPUs) which is also known as IPC.

The tests are also classified as optional or mandatory. It is expected that all test classifications will be developed unless the CPU does not support the features required to perform the tests.

NOTE: The diagnostic tests assume that the target CPU is on a motherboard that is at least capable of running a diagnostic OS. In order to run preboot diagnostic tests, a CIMOM must be available.

Table 2 provides additional information for each test type. The five categories are broken down into more specifically focused tests. For each test, the following information is provided:

- Coverage Area – This describes the objectives and intended coverage for the test. It is intentionally abstracted to allow applicability to multiple CPU architectures.

- Coverage Range – This specifies any specific requirements or restrictions to the test environment or to the scope of the test.

- User Control – This field specifies the intended user configurability of the test. As a general rule, it is desired that all tests have user controllability to specify the duration of the test. This control allows users to make tradeoffs in coverage versus a test's time/cost of test for their specific manufacturing or diagnostic applications. Some examples of user control are:

  – Users may specify the degree of processor stressing, which may also affect the execution time of a single iteration.

  – Users may use Loop control to affect the stress level applied to the processor.

  – Users may provide a seed for randomization of the test operation to provide test result predictability.

- Execution Time – This field identifies a rough estimate for how much time the test requires to be effective. For most of the tests, given the speed of CPUs, the execution time will be on the order of seconds or less.

Built into Device – This field indicates whether any of the diagnostic capabilities are required to be "built-in," that is, capable of executing completely internally such that only a command is issued to the device, which then runs internal diagnostics and reports pass/fail.

Details – This field lists any additional relevant information or instructions for users of the tests.

Table 2 contains information about the test types.

Table 2 – Test Type Information

Test Name	Test Information	
Register (Basic Functionality)	**Coverage Area**	The register test verifies access to all available registers. Basic load and store functions are covered. The test then checks basic addressing modes (register, memory, indirect memory, etc.). Proper data access and movement are the primary focus.
	Coverage Range	These tests should run under either a specialized diagnostic or a production OS.
	User Control	The user may define a subset of addressing modes and/or OS limitations (such as 64-bit registers accessible only in 64-bit mode).
	Execution Time	The diagnostic runs on order of seconds per CPU.
	Built into Device	No
	Details	Addressing modes that are not intended for use in the OS are optional.
Instruction (Basic Functionality)	**Coverage Area**	This diagnostic verifies the functionality of the general CPU instruction set. All instructions are validated except FPU instructions.
	Coverage Range	These tests should run under either a specialized diagnostic or a production OS.
	User Control	A user may elect a subset of instructions. The selection mechanism is vendor-specific.
	Execution Time	The diagnostic runs on the order of seconds per CPU.
	Built into Device	No
	Details	Instructions that are not intended for use in the OS or application space are optional.
FPU Instruction (Basic Functionality)	**Coverage Area**	This diagnostic verifies MMX/SSE/SSE2 instructions. The FPU instruction test verifies floating point addition, subtraction, multiplication, and division operations in all supported precision modes against known values. This diagnostic verifies MMX/SSE/SSE2 registers. Transcendental operations (sine, cosine, etc.) are optional, as are precision modes, which are not intended for use.
	Coverage Range	These tests should run under either a specialized diagnostic or a production OS.
	User Control	A user may select a subset of instructions and precision modes. The selection mechanism is vendor-specific.
	Execution Time	The diagnostic runs on the order of seconds per CPU.
	Built into Device	No
	Details	Instructions and precision modes that are not intended for use in the OS or application space are optional.
Mixed Instruction Width	**Coverage Area**	The diagnostic verifies mixed 32-bit and 64-bit instructions and addressing modes in a 64-bit OS.
	Coverage Range	The width test shall provide complete coverage in a 64-bit OS environment

Test Name	Test Information	
(Basic Functionality)		only.
	User Control	A user may select a subset of instructions and addressing modes. The selection mechanism is vendor-specific.
	Execution Time	The diagnostic shall run on the order of seconds per CPU.
	Built into Device	No
	Details	The diagnostic verifies the ability of the processor to switch between executing 32-bit and 64-bit instructions, including instructions that load, store, and manipulate data. This diagnostic is not intended to execute all possible instructions and data combinations. This is not a full instruction verification test.
Paging and Protected Mode Entry (OS Services)	**Coverage Area**	Computer systems may utilize virtual memory methods to extend and homogenize access to system memory and other forms of data storage (for example, hard disks). Page tables contain mappings between virtual addresses and physical locations in memory. A paging diagnostic may test all supported paging modes or may test only those paging modes that are relevant to the system or application. The diagnostic may check detailed paging operation, or may set up specific page tables and assure that the physical data is properly accessed via logical addressing.
	Coverage Range	This test is only valuable in • a non-protected-mode preboot OS (for example, DOS) because booting a protected-mode OS requires a functioning page table, • a diagnostic OS that allows the user to set up its own page tables that are kept separate from system page tables.
	User Control	None
	Execution Time	The diagnostic runs on the order of milliseconds to seconds per CPU
	Built into Device	No
	Details	The ME may affect the scope of the diagnostic tests. A 64-bit OS may be required to run a complete test on certain processors.
Virtual Machine (OS Services)	**Coverage Area**	This diagnostic shall verify supported VM instructions and the ability to intercept VM privileged instructions. Validation of additional architecture-specific VM features should fall into this category as well. Also included are any specialized features that expedite VM process switching.
	Coverage Range	This diagnostic shall be executed in a preboot environment or in a specialized diagnostic OS that allows the user to set up its own virtual contexts.
	User Control	None
	Execution Time	The diagnostic shall run on the order of seconds per CPU.
	Built into Device	No
	Details	This diagnostic is optional, because not all CPUs support virtualized operation. This diagnostic should cover all intended production configurations.
Exceptions (OS Services)	**Coverage Area**	Exceptions are interrupts that are generated internally by the CPU when certain conditions are detected during the execution of a program. At a high level this diagnostic should install exception handlers, generate exceptions, and verify that the exceptions are handled appropriately.
	Coverage Range	This diagnostic shall provide complete coverage for verifying exception handling in a preboot environment or in a specialized diagnostic OS that allows the user to enable exceptions and either use standard OS exceptions or provide its own exception handlers.
	User Control	None

Test Name	Test Information	
	Execution Time	The diagnostic runs on the order of milliseconds to seconds per CPU.
	Built into Device	No
	Details	The ME may affect the scope the diagnostic tests. A 64-bit OS may be required for complete testing on certain processors. Test algorithms may be required to determine proper exception handling actions based on the present state of the ME.
Status (RAS)	**Coverage Area**	This diagnostic shall verify the overall status of the CPU. The method for verifying the status of a CPU will be architecture specific. For example, for x86 architecture CPUs, a machine check could be used. Example CPU features that may be covered are: • Cache and data path correctable error counts and threshold • Error injection into data transactions (including data poisoning) • Machine Specific Registers that report status of malfunctions • Data cycle or transaction logging features. • Triggers that allow data collection or trapping upon specific internal or external events. • Debug data collection features that allow access to an extended state of the machine upon a failure. • Both in-band and out-of-band and interface access to the state of the machine upon failure.
	Coverage Range	Security and Protection restrictions may depend upon the architecture of the product. Some tests may require a preboot environment or specialized diagnostic OS that allows the user access to these features. Other tests may only be applicable to manufacturer testing.
	User Control	None
	Execution Time	The diagnostic shall run on the order of seconds per CPU.
	Built into Device	No
	Details	
Power Management (Power/ Performance)	**Coverage Area**	The diagnostic verifies the power management features of the processor such as C-States (Core) where a core halts execution and waits for restart, S-States (System) where disks or other peripheral components are suspended by removing the voltage (content must be saved and restored), and P-States (Processor) where voltage and CPU speed is changed to save power. The diagnostic checks that each state can be entered and exited. It also tests the throttling aspect of power management, sets power consumption parameters in the CPU or in the chipset, and verifies whether the CPU is throttled appropriately (such as speed and voltage). This functionality may be combined with the voltage and frequency tests specified below in this table, as power management activities may provide the necessary sequencing to allow proper voltage and frequency transitions. This diagnostic may also be run without changing voltage or frequency to check the functional aspects of power management.
	Coverage Range	Restrictions may apply depending on the architecture of the product, so these tests may require a preboot environment or specialized diagnostic OS which allows the user access to these features.
	User Control	A user may select which power states or features to test. The selection mechanism is vendor-specific. The default behavior shall verify all accessible power management features. To test some C-State and S-States (such as sleep and suspend), user interaction may be required.

Test Name	Test Information	
	Execution Time	The diagnostic shall run on the order of seconds per CPU.
	Built into Device	No
	Details	Test operation may require preboot for complete coverage. Running this test online could cause problems. Thus, DiagnosticTest.Characteristics shall contain the value 4 (Is Risky).
Speed (Power/ Performance)	**Coverage Area**	This diagnostic shall set and verify various clock speeds within the specification of the processor. At a minimum, the test must verify that the CPU is capable of operating at the maximum clock speed advertised by the CPU. Maximum clock speed may be determined through the processor or through the associated CIM instance representing the processor under test.
	Coverage Range	This diagnostic shall provide the ability to set the frequency of the device to all legal operational settings. Typically, changes of frequency are applicable to both performance and power management specifications. Some changes will require software or hardware sequencing (such as power state transitions or thermal throttling) in order to properly execute a change in frequency. In addition, if the device or board has the ability to alter voltage, some frequency changes will be combined with voltage changes.
	User Control	The user may be given the option to select a subset of the speeds and sequences to be tested.
	Execution Time	The diagnostic shall run on the order of seconds per CPU.
	Built into Device	No
	Details	After the diagnostic completes, this test will set the CPU speed back to the speed at which the CPU was running before the test was invoked. The test should check if the CPU speed gets changed while the test is running (such as thermal throttling; if so, the test should generate an appropriate DiagnosticServiceRecord instance.
Voltage (Power/ Performance)	**Coverage Area**	The diagnostic shall set and verify various core voltages. At a minimum the test shall verify that the processor operates appropriately at the maximum voltage specification for the processor.
	Coverage Range	The diagnostic may provide the ability to set the voltage of the device to all legal operational settings, provided that the silicon or board infrastructure allows user control of this feature. There may be a need to write to off-chip resources via IO to accomplish these voltage changes. Typically, changes of voltage are applicable to both performance as well as power management specifications. Some changes (such as power state transitions or thermal throttling) will require software or hardware sequencing in order to properly execute a change in voltage.
	User Control	The user may be given the option to specify the voltages to be used during the test. Additional logic may be required in the provider to determine if specified values are valid for the processor specification. The diagnostic shall generate an appropriate DiagnosticServiceRecord instance in this scenario and should exit without running the diagnostic.
	Execution Time	The diagnostic shall run on the order of seconds per CPU.
	Built into Device	No
	Details	It is possible for the voltage to change during execution of this test. The implementation shall detect voltage changes and report an appropriate message within a DiagnosticServiceRecord. Failure messages should indicate whether a failure was due to an unexpected value for the voltage or a voltage change by an external entity. After the diagnostic completes, this test will set the CPU voltage back to

Test Name	Test Information	
		the voltage at which the CPU was running before the test was invoked. The test will check whether the CPU voltage changes while the test is running (such as thermal throttling); if so, the test will perform vendor-specific actions.
Stress (System Stress and Coherency)	**Coverage Area**	The CPU stress test uses a variety of instruction sets that enable complete access to CPU operations, caches, and memory. Any subsystem (such as cores, threads, caches, bus interfaces, etc.) that can function in parallel should be tested simultaneously. Stress testing does not imply that instructions are selected for the purpose of heating up the core. Instead, instructions should be chosen to functionally stress the architectural features of the device (such as the core, IO interfaces, bus interface unit, cache coherency, etc.). The diagnostic shall activate a variety of CPU features simultaneously. This may involve randomization and testing various combinations of functionality.
	Coverage Range	Restrictions may apply depending on the architecture of the product, so these tests may require a preboot environment or specialized diagnostic OS that allows the user access to these features.
	User Control	Users may specify the duration time of the test. Random code sequence testing is particularly effective for these tests. Users may provide a seed for randomization of the test operation to provide test result predictability while still allowing a variety of stressful stimuli.
	Execution Time	The diagnostic shall run on the order of seconds to minutes per CPU.
	Built into Device	No
	Details	The diagnostic can be used as a “burn-in” test by the manufacturer. DiagnosticSettingData.LoopControl is set to 4 (Timer), and DiagnosticSettingData.LoopControlParameter is set to the duration time of the test.
Cache (System Stress and Coherency)	**Coverage Area**	This diagnostic verifies the integrity and accessibility of all available caches such as the instruction cache, data cache, write policies (write-through, write-back), and coherency protocols. The diagnostics should monitor for protocol and data errors (both correctable and non-correctable). Note that additional associations are required to indicate the managed system elements affected by this test. Associations shall be maintained to indicate AffectedManagedElement relationships with CIM_ComputerSystem, CIM_ProcessorCore, and CIM_Processor (in the case of testing shared cache).
	Coverage Range	Restrictions may apply depending on the architecture of the product. Some tests may require a preboot environment or specialized diagnostic OS that allows the user access to these features.
	User Control	The user may set the threshold values for correctable errors.
	Execution Time	The diagnostic runs on the order of milliseconds to minutes per CPU. The typical execution time should be seconds.
	Built into Device	No
	Details	A 64-bit OS is required for a complete test on certain processors. DiagnosticSettingData.LoopControl is set to 5 (Error Count) and DiagnosticSettingData.LoopControlParameter is set to the threshold value.
IPC (System Stress and Coherency)	**Coverage Area**	This test is applicable to systems with multiple processors only. This diagnostic tests IPC, Caches, Memory, and Bus Controllers, which may be included as part of this diagnostic. Alternatively, this diagnostic may simply re-run the other System Stress and Coherency tests. The diagnostic could also target any features of the CPU architecture that are not covered by

Test Name	Test Information	
		the other diagnostic tests listed in this table. Additionally, they should target all possible communication transactions between CPUs, such as interrupt processing, cache coherency testing, and error signaling.
	Coverage Range	Restrictions may apply depending on the architecture of the product, so these tests may require a preboot environment or specialized diagnostic OS that allows the user access to these features.
	User Control	None
	Execution Time	The diagnostic shall run on the order of seconds per CPU.
	Built into Device	No
	Details	

7.2 CIM_CPUDiagnosticTest

The CIM_CPUDiagnosticTest class defines the tests that can be used to diagnose CPU issues. Table 3 and Table 4 list the set of CPU tests defined by this profile, whether the test implementation is Mandatory or Optional, and the values of certain class properties. An implementation may extend this class and add vendor-defined tests using the Vendor Defined range of the CPUDiagnosticTestType valuemap.

The current values for TestType array property are: 0 (Unknown), 1 (Other), 2 (Functional), 3 (Stress), 4 (Health Check), 5 (Access Test), 6 (Media Verify), 7 (DMTF Reserved), 8 (Vendor Reserved).

Table 3 – CIM_CPUDiagnosticTest Property Requirements

Test Name	Criteria	ElementName*	CPUTestType	TestType *
Register	Optional	CPU Register Test	2	(2) Functional
Instruction	Optional	CPU Instruction Test	3	(2) Functional
FPU Instruction	Optional	CPU FPU Instruction Test	4	(2) Functional
Mixed Instruction Width	Mandatory	CPU Mixed Instruction Width Test	5	(2) Functional
Paging and Protected Mode Entry	Mandatory	CPU Paging and Protected Mode Entry Test	6	(2) Functional
Virtual Machine	Mandatory	CPU Virtual Machine Test	7	(2) Functional
Exceptions	Optional	CPU Exceptions Test	8	(2) Functional
Status	Mandatory	CPU Status Test	9	(2) Functional (4) Health Check
Power Management	Mandatory	CPU Power Management Test	10	(2) Functional
Speed	Mandatory	CPU Speed Test	11	(2) Functional
Voltage	Optional	CPU Voltage Test	12	(2) Functional
Stress	Mandatory	CPU Stress Test	13	(3) Stress
Cache	Mandatory	CPU Cache Test	14	(2) Functional
IPC	Optional	CPU IPC Test	15	(2) Functional

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

The current values for the Characteristics array property inherited from the CIM_DiagnosticTest parent class are: 0 (Unknown), 1 (Other), 2 (Is Exclusive), 3 (Is Interactive), 4 (Is Destructive), 5 (Is Risky), 6 (Is Package), 7 (Reserved), 8 (Is Synchronous), 9 (Media Required), 10 (Additional Hardware Required). The OtherCharacteristicsDescription property is used to provide additional information about the nature of the test. The content of the OtherCharacteristicsDescription property is vendor-specific.

The Characteristics property shall contain the value 4 (Is Destructive) for the Sequential Write test. The property can be NULL for the other tests.

Table 4 – CIM_CPUDiagnosticTest Property Requirements

Test Name	Characteristics*	OtherCharacteristicsDescriptions*	Comment
Register	1 (Other)	Vendor specific	
Instruction	1 (Other)	Vendor specific	User may select instruction subsets to test that will be architecture specific.
FPU Instruction	1 (Other)	Vendor specific	User may select instruction subsets to test that will be architecture specific.
Mixed Instruction Width	1 (Other)	Vendor specific	User may select instruction subsets to test that will be architecture specific.
Paging and Protected Mode Entry	1 (Other)	Vendor specific	
Virtual Machine	1 (Other)	Vendor specific	
Exceptions	1 (Other)	Vendor specific	
Status			
Power Management	1 (Other) 5 (Is Risky)	Vendor specific	To test some C-State and S-States (such as sleep and suspend), user interaction may be required.
Speed			
Voltage	1 (Other) 3 (Is Interactive)	Vendor specific	
Stress	1 (Other) 6 (Is package)	Vendor specific	
Cache	1 (Other)	Vendor specific	
IPC	1 (Other)	Vendor specific	

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

7.3 CIM_CPUDiagnosticSettingData

One or more instances of CIM_CPUDiagnosticSettingData may be implemented. They are associated to CIM_CPUDiagnosticTest using CIM_ElementSettingData. The vendor-defined default values may be specified and advertised using an instance of CIM_CPUDiagnosticSettingData that is referenced by the instance of CIM_ElementSettingData whose property value for IsDefault is 1 (Is Default).

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

CIM_DiagnosticSettingData 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. If additional properties are needed that control the behavior of the diagnostic test, then they should be defined in a subclass of CIM_DiagnosticSettingData.

The client may use one of the vendor-defined default CIM_CPUDiagnosticSettingData instances as an argument to the CIM_CPUDiagnosticTest.RunDiagnosticService extrinsic method. Alternatively, the client may create its own instance of CIM_CPUDiagnosticSettingData and use it instead.

The CIM_CPUDiagnosticSettingData class defines the parameters that may be used by some of the CPU 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_CPUDiagnosticSettingData Property Requirements

Test Name	ElementName*	CPUSpeeds	CPUVoltages	LoopControl*	LoopControl Parameter*	Seed
Register	CPU Register Test					
Instruction	CPU Instruction Test					
FPU Instruction	CPU FPU Instruction Test					
Mixed Instruction Width	CPU Mixed Instruction Width Test					
Paging and Protected Mode Entry	CPU Paging and Protected Mode Entry Test					
Virtual Machine	CPU Virtual Machine Test					
Exceptions	CPU Exceptions Test					
Status	CPU Status Test					
Power Management	CPU Power Management Test					
Speed	CPU Speed Test	Used				
Voltage	CPU Voltage Test		Used			
Stress	CPU Stress Test			4 (Timer)	Used	Used
Cache	CPU Cache Test			5 (Error Count)	Used	
IPC	CPU IPC Test					

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

If any CIM_CPUDiagnosticSettingData property does not have a value when passed as an argument to the CIM_DiagnosticTest.RunDiagnosticService extrinsic method, then the default values for the test arguments shall be used. The default values are defined by the test implementer.

NOTE: The Test Names shown with an asterisk (*) indicate tests that have user controls. However, such controls are too dependent upon the CPU architecture to be generically defined as a CIM_CPUDiagnosticSettingData property. Instead, a vendor should define such properties in a subclass of CIM_CPUDiagnosticSettingData.

7.3.1 CIM_CPUDiagnosticSettingData.CPUSpeeds

This array property is used by a client for the tests shown in Table 5 to specify the CPU speeds to be used during the test.

The vendor-defined default value is advertised using the default instance of CIM_CPUDiagnosticSettingData.

The vendor-defined default value is specified using an instance of CIM_CPUDiagnosticSettingData that is referenced by the instance of CIM_ElementSettingData whose property value for IsDefault is 1 (Is Default).

The vendor-defined maximum value is specified using an instance of CIM_CPUDiagnosticSettingData that is referenced by the instance of CIM_ElementSettingData whose property value for IsMaximum is 1 (Is Maximum).

The vendor-defined minimum value is specified using an instance of CIM_CPUDiagnosticSettingData that is referenced by the instance of CIM_ElementSettingData whose property value for IsMinimum is 1 (Is Minimum).

If no value is specified, the vendor-defined default values will be used.

7.3.2 CIM_CPUDiagnosticSettingData.CoreVoltages

This array property is used by a client for the tests shown in Table 5 to specify the voltages to be used during the test.

The vendor-defined default value is advertised using the default instance of CIM_CPUDiagnosticSettingData.

The vendor-defined maximum value is specified using an instance of CIM_CPUDiagnosticSettingData that is referenced by the instance of CIM_ElementSettingData whose property value for IsMaximum is 1 (Is Maximum).

The vendor-defined minimum value is specified using an instance of CIM_CPUDiagnosticSettingData that is referenced by the instance of CIM_ElementSettingData whose property value for IsMinimum is 1 (Is Minimum).

If no value is specified, the vendor-defined default values will be used.

7.3.3 CIM_CPUDiagnosticSettingData.Seed

This property is used by a client for the Stress test shown in Table 5 to specify the seed to use when random combinations of tests or test values are used.

7.3.4 CIM_DiagnosticSettingData.LoopControl

To specify the time that the Stress test runs, the client sets this property to 4 (Timer).

To specify the threshold error count for the Cache test, the client sets this property to 5 (Error Count).

7.3.5 CIM_DiagnosticSettingData.LoopControlParameter

To specify the time that the Stress test runs, the client sets this property to the desired length of time.

To specify the threshold error count for the Cache test, the client sets this property to the threshold value

7.4 CIM_CPUDiagnosticServiceCapabilities

The SupportedExecutionControls property lists the job controls that are supported by the Diagnostic Service. The values are: 0 (Unknown), 1 (Other), 2 (Job Creation), 3 (Kill Job), 4 (Suspend Job), 5 (Terminate Job), 0x8000 (No Execution Controls)

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), 0x8000 (No Loop Control)

Table 6 and Table 7 specify the possible values for each test for CIM_CPUDiagnosticServiceCapabilities.

Table 6 – CIM_CPUDiagnosticServiceCapabilities Property Requirements

Test Name	ElementName*	SupportedExecutionControls*	Other SupportedExecutionControls*
Register	CPU Register Test	0x8000 (No Execution Control)	
Instruction	CPU Instruction Test	0x8000 (No Execution Control)	
FPU Instruction	CPU FPU Instruction Test	0x8000 (No Execution Control)	
Mixed Instruction Width	CPU Mixed Instruction Width Test	0x8000 (No Execution Control)	
Paging and Protected Mode Entry	CPU Paging and Protected Mode Entry Test	0x8000 (No Execution Control)	
Virtual Machine	CPU Virtual Machine Test	0x8000 (No Execution Control)	
Exceptions	CPU Exceptions Test	0x8000 (No Execution Control)	
Status	CPU Status Test	0x8000 (No Execution Control)	
Power Management	CPU Power Management Test	0x8000 (No Execution Control)	
Speed	CPU Speed Test	1 (Other)	CPUSpeed
Voltage	CPU Voltage Test	1 (Other)	CPUVoltage
Stress	CPU Stress Test	0x8000 (No Execution Control)	
Cache	CPU Cache Test	0x8000 (No Execution Control)	
IPC	CPU IPC Test	0x8000 (No Execution Control)	

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

Table 7 – CIM_CPUDiagnosticServiceCapabilities Property Requirements

Test Name	SupportedLoopControl*	CPUSpeeds	CoreVoltages	Seed
Register	0x8000 (No Loop Control)			
Instruction	0x8000 (No Loop Control)			
FPU Instruction	0x8000 (No Loop Control)			
Mixed Instruction Width	0x8000 (No Loop Control)			

Test Name	SupportedLoopControl*	CPUSpeeds	CoreVoltages	Seed
Paging and Protected Mode Entry	0x8000 (No Loop Control)			
Virtual Machine	0x8000 (No Loop Control)			
Exceptions	0x8000 (No Loop Control)			
Status	0x8000 (No Loop Control)			
Power Management	0x8000 (No Loop Control)			
Speed	0x8000 (No Loop Control)	Used		
Voltage	0x8000 (No Loop Control)		Used	
Stress	4 (Timer)			Used
Cache	5 (Error Count)			
IPC	0x8000 (No Loop Control)			

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

7.4.1 CIM_CPUDiagnosticServiceCapabilities.SupportedExecutionControls

This array property is used by a provider for the tests shown in Table 6 to specify whether or not the test supports execution controls. If there are no execution controls, the value of this property is 0x8000 (No Execution Control).

7.4.2 CIM_CPUDiagnosticServiceCapabilities.OtherSupportedExecutionControls

This array property is used by a provider for the tests shown in Table 6 to specify the execution controls supported by the test when the value of SupportedExecutionControls is 1 (Other).

7.4.3 CIM_CPUDiagnosticServiceCapabilities.SupportedLoopControl

This array property is used by a provider for the tests shown in Table 7 to specify whether or not the test supports loop control. If loop control is not supported, the value of this property is 0x8000 (No Loop Control). If the test is to be run for a specified amount of time, this array property shall contain the value 4 (Timer). If the test is to be run until a threshold error count is reached, this array property shall contain the value 5 (Error Count).

7.4.4 CIM_CPUDiagnosticServiceCapabilities.CPUSpeeds

This array property is used by a provider for the tests shown in Table 7 to specify the CPU speeds supported by the test.

7.4.5 CIM_CPUDiagnosticServiceCapabilities.CoreVoltages

This array property is used by a provider for those tests shown in Table 7 to specify the voltages supported by the test.

7.4.6 CIM_CPUDiagnosticServiceCapabilities.Seed

For those tests shown in Table 7, this boolean property indicates that one can specify the seed for a random sequence to be used by the test.

7.5 CIM_DiagnosticServiceRecord

For certain tests, when unexpected results occur, an instance of CIM_DiagnosticServiceRecord shall be created that uses the message format specified in Table 8.

Table 8 – CIM_DiagnosticServiceRecord Property Requirements

Test Name	DataRecord	RecordFormat	RecordType
Instruction			
FPU Instruction			
Mixed Instruction Width			
Paging and Protected Mode Entry			
Virtual Machine			
Exceptions			
Status			
Power Management			
Speed	“CPU Speed was changed during the test. Expected speed = “YYYY”, Detected speed = “YYYY”	“*string Msg1*uint32 ExpectedSpeed*string Msg2*uint32 DetectedSpeed”	“Warnings”
Voltage	“*Unexpected voltage detected. Expected voltage = *XXXX*, Detected voltage = *XXXX”	“*string Msg1*uint32 ExpectedVoltage*string Msg2*uint32 DetectedVoltage”	“Warnings”
Stress			
Cache			
IPC			
Instruction			
FPU Instruction			

NOTE 1: XXXX in RecordData shall contain the expected voltage and the actual voltage detected by the test when it logs the message.

NOTE 2: YYYY in RecordData shall contain the CPU speed detected by the test when it logs the message.

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_CPUDiagnosticTest.RunDiagnosticService( )

The RunDiagnosticService method shall return one of the return code values defined in DSP1002, Table 2 – RunDiagnosticService 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 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 DSP1002 clauses 8.5 through 8.29.

9 Use Cases

This clause contains use cases for the CPU Diagnostics Profile.

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

9.1 Use Case Summary

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

This clause should be read in combination with the use cases described in the *Diagnostics Profile* (DSP1002), which defines the common methodology for discovering, configuring, and executing diagnostic tests on a system. The following use case descriptions provide the additional information for running the CPU-specific diagnostic tests.

Table 9 summarizes the use cases that are described in this clause. The use cases are categorized and named, and references are provided to the clauses that further describe each use case.

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

Table 9 – CPU Diagnostics Profile Use Cases

Category	Tests	Description
Quick Preboot Verification	Paging and Protected Mode Entry, Registers	Provides quick verification that basic components and OS functions operate properly. See 9.2.
Full Preboot Verification	Paging and Protected Mode Entry, Registers, Virtual Machine, Exceptions	Provides additional verification that other OS functions operate properly. See 9.3.
Quick Functional Verification	Status, Instructions, Mixed Instruction Width	Provides quick verification of basic functionality with no to minimal user interaction required. See 9.4.

Before performing the use cases in this profile, it is assumed that a client has already utilized the use case methodology defined in the *Diagnostics Profile* to discover the following instances:

- ManagedSystemElement (that is, CPU) instances to be tested
- CPUDiagnosticTest instances to be used by this profile

- CPUDiagnosticSettingData instances to be used by this profile that will be passed to the CPUDiagnosticTest.RunDiagnostic extrinsic method.

9.2 Quick Preboot Functional Verification

To quickly verify that basic components of a CPU are operating properly before the system is booted, a client performs the following steps:

1. Select the ManagedSystemElement instance to be tested.
2. Initialize the property values of DiagnosticSettingData as desired (for example HaltOnError, LogOptions, etc.).
3. Select the CPUDiagnosticTest instance that tests page tables, that is, CPUTestType = 6 (Paging and Protected Mode Entry).
4. Invoke the CPUDiagnosticTest.RunDiagnostic extrinsic method using the instances from Step 1 and 2 as arguments.
5. Repeat Steps 2, 3, and 4 for launching the diagnostic tests for Registers.

NOTE: Any failures probably indicate serious functional problems that would probably cause other tests to fail.

9.3 Full Preboot Functional Verification

To more completely verify the proper operation of a CPU before the system is booted, a client performs the following steps:

1. Select the ManagedSystemElement instance to be tested.
2. Initialize the property values of DiagnosticSettingData as desired (for example, HaltOnError, LogOptions, etc.).
3. Select the CPUDiagnosticTest instance that tests page tables, that is, CPUTestType = 6 (Paging and Protected Mode Entry).
4. Invoke the CPUDiagnosticTest.RunDiagnostic extrinsic method using the instances from Step 1 and 2 as arguments.
5. Repeat Steps 2, 3, and 4 for launching the diagnostic tests for Registers, Virtual Machine, and Exceptions.
6. Repeat Steps 2, 3, and 4 for testing FPU Instructions, Registers, Cache, Speed, and Power Management.

9.4 Quick Functional Verification

To quickly verify the proper operation of a CPU, a client performs the following steps after the system is booted:

1. Select the ManagedSystemElement instance to be tested.
2. Initialize the property values of DiagnosticSettingData as desired (for example, HaltOnError, LogOptions, etc.).
3. Select the CPUDiagnosticTest instance that Status test, that is, CPUTestType = 9 (Status).
4. Invoke the CPUDiagnosticTest.RunDiagnostic extrinsic method using the instances from Step 1 and 2 as arguments.

Repeat Steps 2, 3, and 4 for testing Status, Instructions, and Mixed Instruction Width.

9.5 Full Functional Verification

To more completely verify the proper operation of a CPU, a client performs the following steps after the system is booted:

1) Select the ManagedSystemElement instance to be tested.
2) Initialize the property values of DiagnosticSettingData as desired (for example, HaltOnError, LogOptions, etc.).
3) Select the CPUDiagnosticTest instance that tests the Instruction set, that is, CPUTestType = 3 (Instruction).
4) Invoke the CPUDiagnosticTest.RunDiagnostic extrinsic method using the instances from Step 1 and 2 as arguments.
5) Repeat Steps 2, 3, and 4 for testing Status, FPU Instructions, Mixed Instruction Width, Cache, Speed, Voltage, Power Management, and IPC.

9.6 Stress Test

To perform a stress test of a CPU, a client performs the following steps before the system is booted:

1) Select the ManagedSystemElement instance to be tested.
2) Set DiagnosticSettingData.LoopControl to 4 (Timer).
3) Set DiagnosticSettingData.LoopControlParameter to the desired test time duration.
4) Initialize the other property values of DiagnosticSettingData as desired (for example, HaltOnError, LogOptions, etc.).
5) Select the CPUDiagnosticTest instance that performs the Stress test, that is, CPUTestType = 13 (Stress).
7. Invoke the CPUDiagnosticTest.RunDiagnostic extrinsic method using the DiagnosticSettingData instance as an argument

10 CIM Elements

Table 10 shows the instances of CIM Elements for this profile. Instances of the CIM Elements shall be implemented as described in Table 10. Clause 7 (“Implementation”) and Clause 8 (“Methods”) may impose additional requirements on these elements.

Table 10 – CIM Elements: CPU Diagnostics Profile

Element Name	Requirement	Description
Classes		
CIM_CPUDiagnosticTest	Mandatory	See 10.1.
CIM_CPUDiagnosticSettingData	Optional	See 10.2.
CIM_CPUDiagnosticServiceCapabilities	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.

Element Name	Requirement	Description
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	Optional	See 10.15.
CIM_ServiceComponent	Optional	See 10.16.
CIM_UseOfLog	Mandatory	See 10.17.
CIM_DiagnosticServiceRecord	Mandatory	See 10.18.
Indications		
None defined in this profile		

10.1 CIM_CPUDIagnosticTest (Specializes CIM_DIagnosticTest)

CIM_CPUDIagnosticTest is used to represent the Diagnostic Testing for an ODD. This class specializes CIM_DiagnosticTest as defined in the *Diagnostics Profile*. The constraints listed in Table 11 are in addition to those specified in the *Diagnostics Profile*. See the *Diagnostics Profile* for other mandatory elements that must be implemented.

Table 11 – Class: CIM_CPUDiagnosticTest

Elements	Requirement	Notes
ElementName	Mandatory	See 7.2.
Characteristics	Mandatory	See 7.2.
OtherCharacteristicsDescriptions	Conditional	If Characteristics includes the value of 1 (Other), then this property is Mandatory.
CPUTestType	Mandatory	See 7.2.
OtherCPUTestTypeDescription	Conditional	If CPUTestType has a value of 1 (Other), then this property is Mandatory.

10.2 CIM_CPUDiagnosticSettingData (Specializes CIM_DIagnosticSettingData)

CIM_CPUDiagnosticSettingData is used to pass in test parameters and to specify other test control parameters. This class specializes CIM_DiagnosticSettingData as defined in the *Diagnostics Profile*. The constraints listed in Table 12 are in addition to those specified in the *Diagnostics Profile*. See the *Diagnostics Profile* for other mandatory elements that must be implemented.

Table 12 – Class: CIM_CPUDiagnosticSettingData

Elements	Requirement	Notes
ElementName	Mandatory	See 7.3.
CPUSpeeds	Optional	See 7.3.1.
CoreVoltages	Optional	See 7.3.2.
Seed	Optional	See 7.3.3.

10.3 CIM_CPUDiagnosticServiceCapabilities (Specializes CIM_DIagnosticServiceCapabilities)

CIM_CPUDiagnosticServiceCapabilities is used to provide information on the capabilities for the System Meory Diagnostic Service. This class specializes CIM_DiagnosticServiceCapabilities as defined in the *Diagnostics Profile*. The constraints listed in Table 13 are in addition to those specified in the *Diagnostics Profile*. See the *Diagnostics Profile* for other mandatory elements that must be implemented.

Table 13 – Class: CIM_CPUDiagnosticServiceCapabilities

Elements	Requirement	Notes
ElementName	Mandatory	See 7.4.
CPUSpeeds	Optional	See 7.4.4.
CoreVoltages	Optional	See 7.4.5.
Seed	Optional	See 7.4.6.

10.4 CIM_RegisteredProfile

The CIM_RegisteredProfile class is defined by the *Profile Registration Profile*. The requirements denoted in Table 14 are in addition to those mandated by the *Profile Registration Profile*. See the *Profile Registration Profile* for the other mandatory elements that must be implemented.

Table 14 – Class: CIM_RegisteredProfile

Elements	Requirement	Notes
RegisteredName	Mandatory	Shall be “Optical Disk Diagnostics”.
RegisteredVersion	Mandatory	Shall be “1.0.0”.
RegisteredOrganization	Mandatory	Shall be 2 (DMTF).

10.5 CIM_AffectedJobElement

Although defined in the *Diagnostics Profile*, the CIM_AffectedJobElement class is listed here because the AffectedElement reference is scoped down to a subclass of CIM_ManagedElement as specified in clause 4.1. The constraints listed in Table 15 in addition to those specified in the *Diagnostics Profile*. See the *Diagnostics Profile* for other mandatory properties of CIM_AffectedJobElement that must be implemented.

Table 15 – Class: CIM_AffectedJobElement

Properties	Requirement	Notes
AffectedElement (overridden)	Mandatory	Shall be a reference to an instance of the CIM_ManagedElement subclass specified in clause 4.1.
AffectingElement	Mandatory	Shall be a reference to an instance of CIM_ConcreteJob.

10.6 CIM_AvailableDiagnosticService

Although defined in the *Diagnostics Profile*, the CIM_AvailableDiagnosticService class is listed here because the ServiceProvided reference is scoped down to CIM_CPUDiagnosticTest, which is a subclass of CIM_DiagnosticTest, and the UserOfService reference is scoped down to a subclass of CIM_ManagedElement as specified in clause 4.1. The constraints listed in Table 16 in addition to those specified in the *Diagnostics Profile*. See the *Diagnostics Profile* for other mandatory properties of CIM_AvailableDiagnosticService that must be implemented.

Table 16 – Class: CIM_AvailableDiagnosticService

Properties	Requirement	Notes
ServiceProvided (overridden)	Mandatory	Shall be a reference to an instance of CIM_CPUDiagnosticTest.
UserOfService (overridden)	Mandatory	Shall be a reference to an instance of the CIM_ManagedElement subclass specified in clause 4.1.

10.7 CIM_ElementCapabilties

Although defined in the *Diagnostics Profile*, the CIM_ElementCapabilities class is listed here because the ManagedElement reference is scoped down to CIM_CPUDiagnosticTest, which is a subclass of CIM_DiagnosticTest, and the Capabilities reference is scoped down to CIM_CPUDiagnosticServiceCapabilities, which is a subclass of CIM_DiagnosticServiceCapabilities. The constraints listed in Table 17 in addition to those specified in the *Diagnostics Profile*. See the *Diagnostics Profile* for other mandatory properties of CIM_ElementCapabilities that must be implemented.

Table 17 – Class: CIM_ElementCapabilities

Properties	Requirement	Notes
ManagedElement (overridden)	Mandatory	Shall be a reference to an instance of CIM_CPUDiagnosticTest.
Capabilities (overridden)	Mandatory	Shall be a reference to an instance of CIM_CPUDiagnosticServiceCapabilities.

10.8 CIM_ElementSettingData (DiagnosticSettingData)

Although defined in the *Diagnostics Profile*, the CIM_ElementSettingData class is listed here because the ManagedElement reference is scoped down to CIM_CPUDiagnosticTest, which is a subclass of CIM_DiagnosticTest, and the SettingData reference is scoped down to CIM_CPUDiagnosticSettingData, which is a subclass of CIM_DiagnosticSettingData. The constraints listed in Table 18 in addition to those specified in the *Diagnostics Profile*. See the *Diagnostics Profile* for other mandatory properties of CIM_ElementSettingData that must be implemented.

Table 18 – Class: CIM_ElementSettingData

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

10.9 CIM_ElementSettingData (JobSettingData)

Although defined in the *Diagnostics Profile*, the CIM_ElementSettingData class is listed here because the Dependent reference is scoped down to CIM_CPUDiagnosticTest, 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 19 in addition to those specified in the Diagnostics Profile. See the Diagnostics Profile for other mandatory properties of CIM_ElementSettingData that must be implemented.

Table 19 – Class: CIM_ElementSettingData

Properties	Requirement	Notes
ManagedElement (overridden)	Mandatory	Shall be a reference to an instance of CIM_CPUDiagnosticTest.
SettingData (overridden)	Mandatory	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 the Diagnostics Profile, the CIM_ElementSoftwareIdentity class is listed here because the Dependent reference is scoped down to CIM_CPUDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 20 in addition to those specified in the Diagnostics Profile. See the Diagnostics Profile for other mandatory properties of CIM_ElementSoftwareIdentity that must be implemented.

Table 20 – Class: CIM_ElementSoftwareIdentity

Properties	Requirement	Notes
Antecedent	Mandatory	Shall be a reference to an instance of CIM_SoftwareIdentity.
Dependent (overridden)	Mandatory	Shall be a reference to an instance of CIM_CPUDiagnosticTest.

10.11 CIM_HostedService

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

Table 21 – Class: CIM_HostedService

Properties	Requirement	Notes
Antecedent	Mandatory	Shall be a reference to an instance of CIM_ComputerSystem.
Dependent (overridden)	Mandatory	Shall be a reference to an instance of CIM_CPUDiagnosticTest.

10.12 CIM_OwningJobElement

Although defined in the Diagnostics Profile, the CIM_OwningJobElement class is listed here because the OwningElement reference is scoped down to CIM_CPUDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 22 in addition to those specified in the Diagnostics Profile. See the Diagnostics Profile for other mandatory properties of CIM_OwningJobElement that must be implemented.

Table 22 – Class: CIM_OwningJobElement

Properties	Requirement	Notes
OwningElement (overridden)	Mandatory	Shall be a reference to an instance of CIM_CPUDiagnosticTest.
OwnedElement	Mandatory	Shall be a reference to an instance of CIM_ConcreteJob.

10.13 CIM_RecordAppliesToElement

Although defined in the *Diagnostics Profile*, the CIM_RecordAppliesToElement class is listed here because the Dependent reference is scoped down to CIM_CPUDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 23 in addition to those specified in the *Diagnostics Profile*. See the *Diagnostics Profile* for other mandatory properties of CIM_RecordAppliesToElement that must be implemented.

Table 23 – Class: CIM_RecordAppliesToElement

Properties	Requirement	Notes
Antecedent	Mandatory	Shall be a reference to an instance of CIM_RecordForLog.
Dependent (overridden)	Mandatory	Shall be a reference to an instance of CIM_CPUDiagnosticTest.

10.14 CIM_ServiceAffectsElement

Although defined in the *Diagnostics Profile*, the CIM_ServiceAffectsElement class is listed here because the AffectedElement reference is scoped down to a subclass of CIM_ManagedElement as specified in clause 4.1, and the AffectingElement reference is scoped down to CIM_CPUDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 24 in addition to those specified in the *Diagnostics Profile*. See the *Diagnostics Profile* for other mandatory properties of CIM_ServiceAffectsElement that must be implemented.

Table 24 – Class: CIM_ServiceAffectsElement

Properties	Requirement	Notes
AffectedElement (overridden)	Mandatory	Shall be a reference to an instance of the CIM_ManagedElement subclass specified in clause 4.1.
AffectingElement (overridden)	Mandatory	Shall be a reference to an instance of CIM_CPUDiagnosticTest.

10.15 CIM_ServiceAvailableElement

Although defined in the *Diagnostics Profile*, the CIM_ServiceAvailableToElement class is listed here because the UsersOfService reference is scoped down to CIM_CPUDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 25 in addition to those specified in the *Diagnostics Profile*. See the *Diagnostics Profile* for other mandatory properties of CIM_ServiceAvailableToElement that must be implemented.

Table 25 – Class: CIM_ServiceAvailableToElement

Properties	Requirement	Notes
ServiceProvided	Mandatory	Shall be a reference to an instance of CIM_HelpService.

Properties	Requirement	Notes
UsersOfService (overridden)	Mandatory	Shall be a reference to an instance of CIM_CPUDiagnosticTest.

10.16 CIM_ServiceComponent

Although defined in the *Diagnostics Profile*, the CIM_ServiceComponent class is listed here because the GroupComponent reference is scoped down to CIM_CPUDiagnosticTest, which is a subclass of CIM_DiagnosticTest, and the PartComponent reference is scoped down to CIM_CPUDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 26 in addition to those specified in the *Diagnostics Profile*. See the *Diagnostics Profile* for other mandatory properties of CIM_ServiceComponent that must be implemented.

Table 26 – Class: CIM_ServiceComponent

Properties	Requirement	Notes
GroupComponent (overridden)	Mandatory	Shall be a reference to an instance of CIM_CPUDiagnosticTest.
PartComponent (overridden)	Mandatory	Shall be a reference to an instance of CIM_CPUDiagnosticTest.

10.17 CIM_UseOfLog

Although defined in the *Diagnostics Profile*, the CIM_UseOfLog class is listed here because the Dependent reference is scoped down to CIM_CPUDiagnosticTest, which is a subclass of CIM_DiagnosticTest. The constraints listed in Table 27 in addition to those specified in the *Diagnostics Profile*. See the *Diagnostics Profile* for other mandatory properties of CIM_UseOfLog that must be implemented.

Table 27 – Class: CIM_UseOfLog

Properties	Requirement	Notes
Antecedent	Mandatory	Shall be a reference to an instance of CIM_DiagnosticLog.
Dependent (overridden)	Mandatory	Shall be a reference to an instance of CIM_CPUDiagnosticTest.

10.18 CIM_DiagnosticServiceRecord

CIM_DiagnosticServiceRecord is used to provide information on the record log. The constraints listed in Table 28 are in addition to those specified in the *Diagnostics Profile*. See the *Diagnostics Profile* for other mandatory elements that must be implemented. See the *Diagnostics Profile* for other mandatory properties of CIM_DiagnosticServiceRecord that must be implemented

Table 28 – Class: CIM_DiagnosticServiceRecord

Properties	Requirement	Notes
RecordData	Mandatory	See 7.5.
RecordFormat	Mandatory	See 7.5.
RecordType	Mandatory	See 7.5.

Annex A
(informative)

Change Log

Version	Date	Description
1.0.0	2011-06-30	DMTF Standard