Document Number: DSP1119

Date: 2012-09-12

Version: 1.0.0a

Diagnostic Job Control Profile

Information for Work-in-Progress version:

IMPORTANT: This specification 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, this specification may still change, perhaps profoundly. This document is available for public review and comment until the stated expiration date.

It expires on: 2013-02-01

Target version for DMTF Standard: 2.34.0

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

Document Type: Specification

Document Status: Work In Progress

Document Language: en-US

CONTENTS

Figures

Tables

Foreword

The *Diagnostic Job Control Profile* (DSP1119) 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:

- Dave Barrett – Emulex Corporation
- 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 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, Diagnostic Job Control.

The goal of the *Diagnostic Job Control Profile* is to define industry-standard building blocks that enable management diagnostic tests running in a standard job infrastructure. The *Diagnostic Job Control Profile* extends the *Job Control Profile* (DSP1103) by identifying a set of job control functions that should be included in provider implementations.

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.

Experimental material

Experimental material has yet to receive sufficient review to satisfy the adoption requirements set forth by the DMTF. Experimental material is included in this document as an aid to implementers who are interested in likely future developments. Experimental material may change as implementation experience is gained. It is likely that experimental material will be included in an upcoming revision of the specification. Until that time, experimental material is purely informational.

The following typographical convention indicates experimental material:

EXPERIMENTAL

Experimental material appears here.

EXPERIMENTAL

In places where this typographical convention cannot be used (for example, tables or figures), the "EXPERIMENTAL" label is used alone.

157 # Diagnostic Job Control Profile

158 ## 1 Scope

159 The *Diagnostic Job Control Profile* specializes the *Job Control Profile* (DSP1103) by defining the job
160 control functions used to monitor and interact with diagnostic tests.

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

163 ## 2 Normative references

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

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

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

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

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

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

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

180 DMTF DSP1103, *Job Control Profile 1.0.0*,
181 http://dmtf.org/sites/default/files/standards/documents/DSP1103_1.0.pdf

182 DMTF DSP1105, *CPU Diagnostics Profile 1.0*
183 http://dmtf.org//sites/default/files/standards/documents/DSP1105_1.0.pdf

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

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

188 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*,
189 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
job
a task that takes some time to execute

3.2
organization
consortium, standards group, or company creating a DMTF profile specification

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
CQL
CIM Query Language

4.5
ME
Managed Element

4.6

MOF
Managed Object Format

4.7
OS
Operating System

4.8
QoS
Quality of Service

4.9
URI
Uniform Resource Identifier

4.10
WBEM
Web-Based Enterprise Management

5 Synopsis

Profile Name: Diagnostics Job Control

Version: 1.0.0a

Organization: DMTF

CIM schema version: 2.34

Central Class: CIM_ConcreteJob

Scoping Class: CIM_System

Specializes: Job Control Profile 1.0.0

The *Diagnostic Job Control Profile* extends the management capability of referencing profiles by adding common methods for managing the jobs associated with diagnostic tests that are run on a managed system

The Central Instance of this profile shall be an instance of CIM_ConcreteJob. The Scoping Instance shall be the instance of CIM_System (the central instance of the referencing profile) with which the Central Instance (the instance of CIM_ConcreteJob) is associated through CIM_HostedDependency. The CIM_System is the system running the CIM_DiagnosticTest and its associated CIM_ConcreteJob.

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

Table 1 – Referenced profiles

Profile Name	Organization	Version	Description
Job Control	DMTF	1.0	Specializes
Indications	DMTF	1.2	Mandatory
Diagnostics	DMTF	2.0	Mandatory
Profile Registration	DMTF	1.0	Mandatory

6 Description

The *Diagnostics Profile* (DSP1002) defines the behavior and interfaces to be used for running and monitoring diagnostic tests and reviewing their results. DSP1103 defines the behavior and interfaces to be used for running and monitoring jobs associated with those executing tests. This profile extends and constrains the elements of the DSP1103 elements that have diagnostic test-specific behavior and interfaces.

Specifically, a user invokes CIM_DiagnosticTest.RunDiagnosticService( ) to start a diagnostic test. A user may optionally pass an embedded instance of CIM_JobSettingData as an input parameter to specify the behavior of the associated CIM_ConcreteJob instance. A CIM_ConcreteJob instance is created when a diagnostic test starts. When the CIM_ConcreteJob instance is deleted is controlled by the values of the properties in CIM_ConcreteJob and CIM_JobSettingData.

To start a diagnostic test, the client calls CIM_DiagnosticTest.RunDiagnosticService( ) which returns 0 (Success) and the objectpath of the CIM_ConcreteJob instance. Thereafter, CIM_ConcreteJob manages the diagnostic test execution. Additionally, a client monitors and interacts with the diagnostic test execution or workflow via the returned CIM_ConcreteJob instance.

Some diagnostic tests may launch other diagnostic tests. Others may require user interaction. The CIM_JobSettingData instance contains properties that define the behavior for interactive diagnostic tests.

When a diagnostic test starts, only one CIM_ConcreteJob instance is created. When the diagnostic test completes, its CIM_ConcreteJob instance will persist for a predetermined length of time before deletion. The same diagnostic test could start again, creating another CIM_ConcreteJob instance before the previous CIM_ConcreteJob instance is deleted. For this reason, the cardinality of CIM_OwningJobElement is one-to-many.

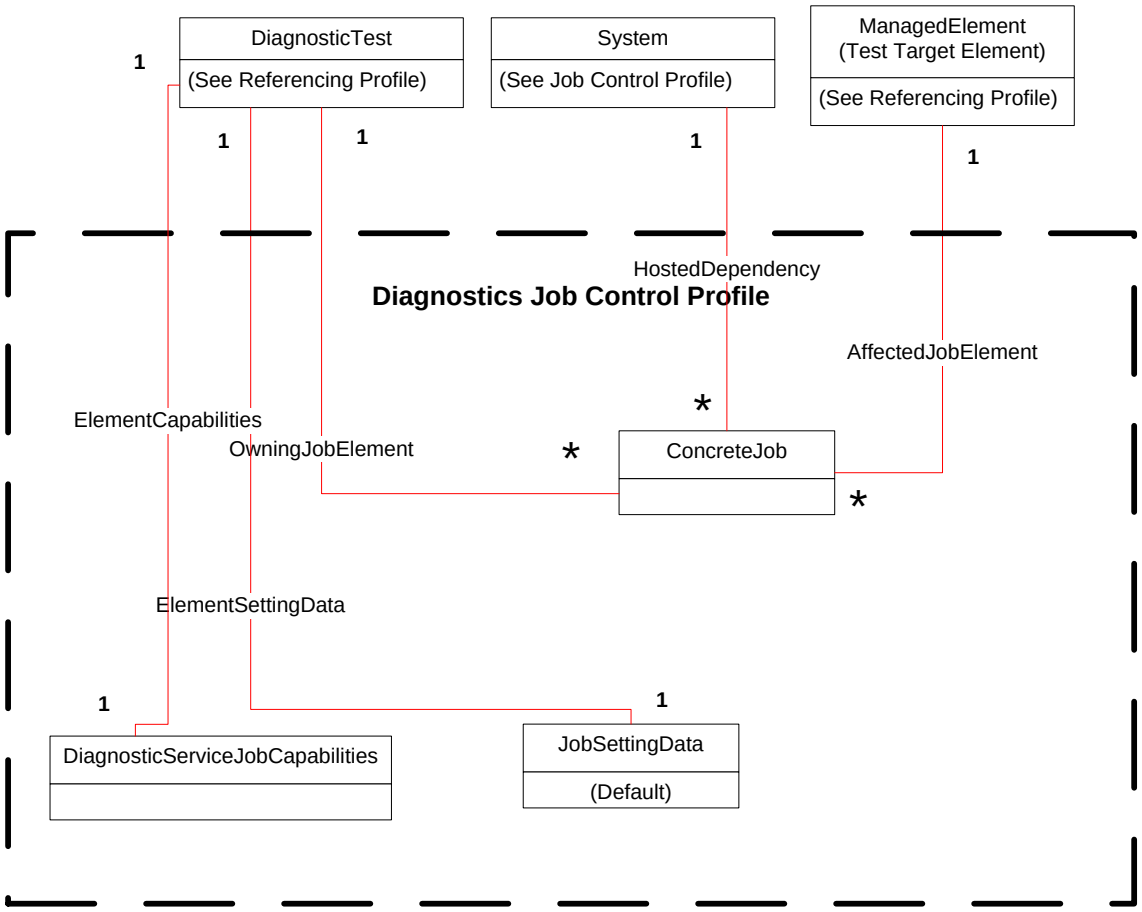

Figure 1 – Diagnostic Job Control Profile: Profile class diagram

The Referencing Profile for DiagnosticTest is a component profile. For example, if the Referencing Profile is the *CPU Diagnostics Profile* (DSP1105) DiagnosticTest is a subclass called CPUDiagnosticTest. Similarly, if the Referencing Profile for ManagedElement is the same component profile as for DiagnosticTest. Thus, the DSP1105 ManagedElement is CIM_Processor, CIM_ProcessorCore or CIM_HardwareThread.

7 Implementation

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

7.1 CIM_AffectedJobElement

This subclause defines the properties of the CIM_AffectedJobElement class.

7.1.1 CIM_AffectedJobElement.ElementEffects

This optional property shall include only the following values: 1 (Other), 2 (Exclusive Use), 3 (Performance Impact), 4 (Element Integrity), and 5 (Create). If 1 (Other) is specified, the OtherElementEffectsDescriptions shall have a value.

7.2 CIM_ConcreteJob

This subclause indicates the properties of the CIM_ConcreteJob class. Each execution of a test will create an instance of CIM_ConcreteJob so that a client can track the progress and control the execution of the diagnostic. To quickly and directly find the CIM_ConcreteJob instance of a executing test, a client should retain the value of the Job output parameter returned by CIM_DiagnosticTest.RunDiagnosticService( ) when the test is stared.

7.2.1 CIM_ConcreteJob.InstanceID

This string property is the key property for this class. It should be constructed using the following preferred algorithm:

<OrgID>:<LocalID>

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

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

A unique <LocalID> allows a user to easily retrieve test results from the diagnostic log for a specific test execution because the InstanceID values of CIM_ConcreteJob, and the subclasses of CIM_DiagnosticRecord are closely related. See Figure 5 in DSP1002 and use case 9.8.4 GetDiagnosticResults for further information.

Specifically, CIM_DiagnosticRecord.InstanceID has the same value as its related CIM_ConcreteJob.InstanceID with an appended record number identifier. As an example, if CIM_ConcreteJob.InstanceID has the form “Widget:<StartTime>”, for the third record, CIM_DiagnosticRecord.InstanceID has the form “Widget:<StartTime>:3”, where <StartTime> is the value of CIM_ConcreteJob.StartTime.

7.2.2 CIM_ConcreteJob.Name

The value of this string property shall correspond to the value of the Name property of its associated CIM_DiagnosticTest instance.

7.2.3 CIM_ConcreteJob.JobState

As defined in DSP1103, this enumerated integer may have the values of 2 (New), 3 (Starting), 4 (Running), 5 (Suspended), 6 (Shutting Down), 7 (Completed), 8 (Terminated), 9 (Killed), 10 (Exception). See Table 3 in DSP1103 for further information. For this profile, 12 (Query Pending) is also permitted to provide the ability for a client to interact with a diagnostic test. The job changes the value JobState to 12 (Query Pending) when it sends an AlertIndication to the client requesting input or action. The job changes the JobState from 12 (Query Pending) when it successfully receives a ResumeWithInput( ) or ResumeWithAction( ) request, or the client fails to respond within the CIM_JobSettingData.InteractiveTimeout period.

On a successful ResumeWithInput( ) or ResumeWithAction invocation( ), the job changes JobState to 4 (Running). If the extrinsic method fails, then the job may wait for a client retry. If the job waits for a client retry, it would stay in the 12 (Query Pending) state. If the client exceeds the number of retries (see CIM_JobSettingData.ClientRetries) or the CIM_ConcreteJob.InteractiveTimeout expires the job may terminate and set JobState to 8 (Terminated).

Table 4 in DSP1103 defines the mapping of values between OperationalStatus and JobState. Table 2 defines the additional mapping for this profile.

Table 2 – OperationalStatus to JobState mapping

Operational Status	JobState	Description
2 (OK)	4 (Running)	Client has responded to the prompt
10 (Stopped)	12 (Query Pending)	Waiting for the client to respond to the prompt

7.2.4 CIM_ConcreteJob.DeleteOnCompletion

This profile extends DSP1103 to define that the default value shall be TRUE. This boolean property indicates whether the CIM_ConcreteJob instance associated to a diagnostic test execution is automatically deleted when test execution completes. See DSP1103 for further implementation details.

7.2.5 CIM_ConcreteJob.TimeBeforeRemoval

This profile extends DSP1103 to define the time to wait before removing a job after the job completes, is terminated or killed. The value supplied must be a datetime offset.

See DSP1103 for further implementation details.

DSP1103 recommends a value of five or more minutes. For diagnostic test environments, clients should consider using a scaled approach. For example, start with a value ten times the typical elapsed time experienced in a normally loaded environment and increase to possibly twenty times for a highly loaded environment.

7.2.6 CIM_ConcreteJob.StartTime

For this profile, the value of this timestamp datetime property represents the start time for the diagnostic test. Such information should also be written to a CIM_DiagnosticLog associated to the diagnostic test using a CIM_DiagnosticServiceRecord entry.

7.2.7 CIM_ConcreteJob.ElapsedTime

For this profile, the value of this interval datetime property shall be updated at a vendor-defined interval. A client can monitor this property at a client-defined interval. When the property changes its value, the client knows that the test is still making progress.

7.2.8 CIM_ConcreteJob.PercentComplete

In addition to the requirements specified in DSP1103, this profile uses PercentComplete to show the amount of testing done in terms of actual percent complete. Service implementations should update this property within a reasonable time of becoming aware of a progress change. If progress can not be determined with that reasonable amount of time it should be set to 50 percent. It shall be set to 100 percent only when the test is complete. It shall not be set to 100 percent if the test stops for any other reason (for example, the test stopped or was killed by user, the test exited due to a critical failure, or the test found an error and HaltOnError is TRUE) because the actual percent complete is not 100 percent.

7.2.9 CIM_ConcreteJob.ElementName

The value of this string property shall be initialized to the value of the ElementName property of its associated CIM_DiagnosticTest instance. Clients can modify this value to further identify the test instance.

EXPERIMENTAL

7.3 CIM_DiagnosticServiceJobCapabilities

This subclause indicates the properties of the optional CIM_DiagnosticServiceJobCapabilities class. This class should be implemented for the convenience of clients. However, a client cannot modify any properties in CIM_DiagnosticServiceJobCapabilities.

To start a diagnostic test, a client invokes the CIM_DiagnosticTest.RunDiagnosticService( ) extrinsic method. The Setting input parameter is an instance of CIM_JobSettingData (Client). When implemented, the property values of CIM_JobSettingData (Client) instance shall be consistent with the values of the CIM_DiagnosticServiceJobCapabilities instance.

If CIM_DiagnosticServiceJobCapabilities is implemented, a client can create a CIM_JobSettingData (Client) instance containing the default values specified in the CIM_DiagnosticServiceJobCapabilities instance by invoking the CIM_DiagnosticServiceJobCapabilities.CreateJobSettings( ) extrinsic method. See subclause 8.3.1 for further information.

For interactive diagnostic tests that will wait for a client response, the CIM_DiagnosticTest.Characteristics property shall contain the value 3 (Is Interactive). An interactive test may define the time interval it shall wait for a client to respond. While waiting for the client to respond, the CIM_ConcreteJob.JobState property associated to the diagnostic test has the value of 12 (Query Pending). If the client fails to respond within the specified time interval, the diagnostic test may terminate, resume using default responses, or wait another time interval. The CIM_DiagnosticServiceJobCapabilties properties InteractiveTimeoutMax, DefaultValuesSupported, and ClientRetriesMax control such behavior. Thus, when the CIM_DiagnosticTest.Characteristics property contains the value 3 (Is Interactive), these properties shall have a value. Otherwise, they are ignored.

7.3.1 CIM_DiagnosticServiceJobCapabilities.InstanceID

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

<OrgID>:<LocalID>

where <OrgID> identifies the business entity (for example, ACME) and <LocalID> is a value that uniquely identifies each DiagnosticServiceJobCapabilities instance that is instantiated on a system.

The purpose for <LocalID> is to provide some form of uniqueness within the context of different DiagnosticServiceJobCapabilities instances within the system. In practice, since there would be only one CIM_DiagnosticServiceJobCapabilities for an instance of the CIM_DiagnosticTest.

7.3.2 CIM_DiagnosticServiceJobCapabilities.ElementName

The value of this string property shall correspond to the value of the ElementName property of its associated CIM_DiagnosticTest instance.

7.3.3 CIM_DiagnosticServiceJobCapabilities.DeleteJobSupported

This boolean property indicates whether the diagnostic test implementation allows a client to perform a DeleteInstance operation on a CIM_ConcreteJob instance. It also defines whether a client can set the value of CIM_JobSettingData.DeleteOnCompletion when it passes an instance of CIM_JobSettingData as the JobSetting parameter to the CIM_DiagnosticTest.RunDiagnosticService( ) extrinsic method.

If the value of DeleteJobSupported is FALSE, a client cannot perform a DeleteInstance operation on a CIM_ConcreteJob instance associated to its CIM_DiagnosticTest instance. Furthermore, the value of the CIM_JobSettingData.DeleteOnCompletion property shall be TRUE. In addition, a client shall not set the value of the DeleteOnCompletion property in the CIM_JobSettingData instance that it passes as the JobSetting parameter to the CIM_DiagnosticTest.RunDiagnosticService( ) extrinsic method.

If the value of DeleteJobSupported is TRUE, a client can perform a DeleteInstance operation on a CIM_ConcreteJob instance associated to its CIM_DiagnosticTest instance. Furthermore, the value of the CIM_JobSettingData.DeleteOnCompletion property may be TRUE or FALSE. In addition, a client may set the value of the DeleteOnCompletion property in the CIM_JobSettingData instance that it passes as the JobSetting parameter to the CIM_DiagnosticTest.RunDiagnosticService( ) extrinsic method.

However, a client cannot perform the DeleteInstance operation when CIM_ConcreteJob.JobState has the value 2 (New), 3 (Starting), 4 (Running), 5 (Suspended) or 12 (Query Pending) even if DeleteJobSupported is TRUE and CIM_ConcreteJob.DeleteOnCompletion is FALSE.

To delete a non-completed job, a client can terminate the job by changing its state to 8 (Terminated) or 9 (Killed) by invoking the CIM_ConcreteJob.RequestedStateChange( ) extrinsic method.

7.3.4 CIM_DiagnosticServiceJobCapabilities.RequestedStatesSupported

This array property indicates the permitted values that a client may pass as the RequestedState parameter to the CIM_ConcreteJob.RequestStateChange( ) extrinsic method. The permitted values are 2 (Start), 3 (Suspend), 4 (Terminate), and 5 (Kill).

A client specifies 3 (Suspend) to suspend a diagnostic test and specifies 2 (Start) to resume a suspended diagnostic test.

7.3.5 CIM_DiagnosticServiceJobCapabilities.InteractiveTimeoutMax

This interval datetime offset property shall have a value if the CIM_DiagnosticTest.Characteristics property contains the value 3 (Is Interactive).

For an interactive diagnostic test that prompts a client for a response, this property defines the maximum time interval a test shall wait for a client to respond. If a diagnostic test prompts a client multiple times, the specified maximum time interval applies to each prompt.

If a client passes an instance of CIM_JobSettingData as the JobSetting parameter to the CIM_DiagnosticTest.RunDiagnosticService( ) extrinsic method, and the value of CIM_JobSettingData.InteractiveTimeout exceeds the value of CIM_DiagnosticServiceJobCapabilities.InteractiveTimeoutMax, the associated CIM_ConcreteJob instance shall use the value of CIM_DiagnosticServiceJobCapabilities.InteractiveTimeoutMax

NOTE If a default for InteractiveTimeout is supported, then this would be specified in the Default CIM_JobSettingData.InteractiveTimeout property.

7.3.6 CIM_DiagnosticsServiceJobCapabilities.DefaultValuesSupported

This boolean property indicates whether an interactive diagnostic test will supply default input values when the test prompts a client for a response but the client fails to respond.. This property shall have a value if the CIM_DiagnosticTest.Characteristics property contains the value of 3 (Is Interactive). Otherwise, it is ignored.

When the value is TRUE, the values of CIM_JobSettingData.DefaultInputValues and JobSettingData.DefaultInputNames are used as needed by the CIM_ConcreteJob instance. Otherwise, the value shall be FALSE.

7.3.7 CIM_DiagnosticsServiceJobCapabilities.ClientRetriesMax

This property shall have a value if the CIM_DiagnosticTest.Characteristics property contains the value of 3 (Is Interactive). Otherwise, it is ignored.

When an interactive diagnostic test prompts a client for a response, the test may define the time interval it shall wait for a client to respond. This property indicates the maximum number of times a diagnostic test shall wait for another time interval.

7.3.8 CIM_DiagnosticsServiceJobCapabilities.CleanupInterval

As described in DSP1103, if the value of the CIM_ConcreteJob.DeleteOnCompletion property is FALSE, the job associated to the diagnostic test execution shall remain until it is explicitly deleted. When the value of the CIM_ConcreteJob.DeleteOnCompletion property is FALSE, the CleanupInterval datetime property defines the time interval before the job is removed.

7.3.9 CIM_DiagnosticServiceJobCapabilities.SilentModeSupported

If the value of the property is TRUE, the interactive diagnostic test is capable of running without prompting the client for responses. Instead, the test uses the default input argument values defined in CIM_JobSettingdData. If the value of the property is FALSE, the interactive diagnostic test shall prompt the client for responses.

EXPERIMENTAL

7.4 CIM_JobSettingData (Client)

This subclause indicates the properties of the CIM_JobSettingData class that may be used by a client as the JobSetting parameter when invoking the CIM_DiagnosticTest.RunDiagnosticService( ) extrinsic method to start a diagnostic test. An instance of this class controls the execution of CIM_ConcreteJob instance related to the executing diagnostic test.

If CIM_DiagnosticServiceJobCapabilities is implemented, a client can create a CIM_JobSettingData (Client) instance containing the default values specified in the CIM_DiagnosticServiceJobCapabilities instance by invoking the CIM_DiagnosticServiceJobCapabilities.CreateJobSettings( ) extrinsic method. See subclause 8.3.1 for further information.

CIM_JobSettingData is specified by a client as an embedded instance input parameter. The class CIM_JobSettingData (Client) is defined in the CIM Elements tables to define what the client may include in the embedded instance. In addition, the client should refer to the CIM_DiagnosticServiceJobCapabilities class to see what restrictions the implementation may impose on the client providing the CIM_JobSettingData embedded instance.

For interactive diagnostic tests that will wait for a client response, the CIM_DiagnosticTest.Characteristics property shall contain the value 3 (Is Interactive). An interactive test may define the time interval it shall wait for a client to respond. While waiting for the client to respond, the CIM_ConcreteJob.JobState

property associated to the diagnostic test has the value of 12 (Query Pending). If the client fails to respond within the specified time interval, the diagnostic test may terminate, resume using default responses, or wait another time interval. The CIM_JobSettingData properties InteractiveTimeout, TerminateOnTimeout, DefaultInputValues, and DefaultInputNames control such behavior. Thus, when the CIM_DiagnosticTest.Characteristics property contains the value 3 (Is Interactive) and the value of CIM_DiagnosticServiceJobCapabilities.DefaultValuesSupported is TRUE, these properties shall have a value. Otherwise, they are ignored.

7.4.1 CIM_JobSettingData.InstanceID

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

<OrgID>:<LocalID>

where <OrgID> identifies the business entity (for example, ACME) and <LocalID> is a value that uniquely identifies each JobSettingData instance that is instantiated on a system.

The purpose for <LocalID> is to provide some form of uniqueness within the context of different JobSettingData instances within the system. In practice, <LocalID> could be an incremented counter or a timestamp in combination with other test identifiers or factors.

7.4.2 CIM_JobSettingData.DeleteOnCompletion

This boolean property indicates whether the job should be automatically deleted upon completion. The property is mandatory. When the value is TRUE, the job shall be deleted after the CIM_ConcreteJob.TimeBeforeRemoval time interval has elapsed. When the value is FALSE, the job must be deleted by an DeleteInstance operation.

NOTE When the value of the CIM_DiagnosticServiceJobCapabilities.DeleteJobSupported property is FALSE, the value of CIM_JobSettingData.DeleteOnCompletion shall be TRUE.

EXPERIMENTAL

7.4.3 CIM_JobSettingData.InteractiveTimeout

This interval datetime property shall have a value if the CIM_DiagnosticTest.Characteristics property contains the value of 3 (Is Interactive). Otherwise, this property is ignored.

If the client fails to respond within the specified time interval, the test may terminate, resume using default responses, or wait another time interval. The default value is 15 minutes (00000000001500.000000:000).

7.4.4 CIM_JobSettingData.TerminateOnTimeout

This boolean property shall have a value if the CIM_DiagnosticTest.Characteristics property contains the value of 3 (Is Interactive). Otherwise, this property is ignored.

This property defines the behavior when a client fails to respond within the time interval defined by CIM_JobSettingData.InteractiveTimeout. When this value is TRUE the job will terminate when the InteractiveTimeout is exceeded on the last retry, if applicable. When this value is FALSE the job will use DefaultInputValues and DefaultInputNames.

7.4.5 CIM_JobSettingData.DefaultInputValues

This string array property shall contain one or more values if the CIM_DiagnosticTest.Characteristics property contains the value of 3 (Is Interactive) and the value of CIM_DiagnosticServiceJobCapabilities.DefaultValuesSupported is TRUE. Otherwise, this property is ignored.

This string array property contains the default values for a client response when the test resumes. The name of each DefaultInputValues array element is defined at the same array index in the CIM_JobSettingData.DefaultInputNames string array.

NOTE These values override any values that may be defined in the CIM_JobSettingData (Default) instance.

7.4.6 CIM_JobSettingData.DefaultInputNames

This string array property shall contain one or more values if the CIM_DiagnosticTest.Characteristics property contains the value of 3 (Is Interactive) and the value of CIM_DiagnosticServiceJobCapabilities.DefaultValuesSupported is TRUE. Otherwise, this property is ignored.

This string array property contains the possible argument names requested by the diagnostic test. The default value of each DefaultInputNames array element is defined at the same array index in the CIM_JobSettingData.DefaultInputValues string array.

7.4.7 CIM_JobSettingData.ClientRetries

This integer property indicates the number of times the diagnostic test will prompt the client for the same response after the client fails to invoke the CIM_ConcreteJob.ResumeWithInput( ) or CIM_ConcreteJob.ResumeWithAction( ) extrinsic method within a specified period of time . A non-zero value for this property indicates that the diagnostic test will issue another DIAG34 or DIAG35 message for the same response.

This property is required if CIM_DiagnosticTest.Characteristics has the value of 3 (Is Interactive). Otherwise, this property value is ignored.

7.4.8 CIM_JobSettingData.RunInSilentMode

This boolean property indicates whether the diagnostic test will not prompt the client for responses even though CIM_DiagnosticTest.Characteristics contains the value of 3 (Is Interactive). When the value is TRUE, no prompts are issued. Instead, the diagnostic test will execute using the default values defined in CIM_JobSettingData. When the value is FALSE, the interactive diagnostic test will prompt the client for a response.

If CIM_DiagnosticServiceJobCapabilties.SilentModeSupported has the value of FALSE, this property is ignored.

EXPERIMENTAL

7.5 CIM_JobSettingData (Default)

This subclause indicates the properties of the default CIM_JobSettingData class. An instance of this class controls the execution of a diagnostic test job. This class is optional. If it is implemented it shall represent the CIM_JobSettingData (Default) for the CIM_DiagnosticTest. This instance is identified by CIM_ElementSettingData with IsDefault=”true” between the instance and the CIM_DiagnosticTest instance. A different default CIM_JobSettingData instance may be defined for each CIM_DiagnosticTest test type. For example, each of the different CPU diagnostic tests may define a different set of default CIM_JobSettingData values.

A CIM_JobSettingData (Client) may be specified by a client as an embedded instance input to an invocation of the CIM_DiagnosticTest.RunDiagnosticService( ) method. This embedded instance is not instantiated as an instance of CIM_JobSettingData (Default), but the class CIM_JobSettingData (Client) is defined in the CIM Elements tables to indicate what the client may include in the embedded instance. To use all of the default values, the client can create an identical instance of CIM_JobSettingData except that the InstanceID key property shall have a different value.

If CIM_DiagnosticServiceJobCapabilities is implemented, the client should refer to the CIM_DiagnosticServiceJobCapabilities instance to see what restrictions the implementation may impose on the client providing the CIM_JobSettingData embedded instance.

If CIM_DiagnosticServiceJobCapabilities is implemented, a client can create a CIM_JobSettingData (Client) instance containing the default values specified in the CIM_DiagnosticServiceJobCapabilities instance by invoking the CIM_DiagnosticServiceJobCapabilities.CreateJobSettings( ) extrinsic method. See subclause 8.3.1 for further information.

7.5.1 CIM_JobSettingData.InstanceID

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

<OrgID>:<LocalID>

where <OrgID> identifies the business entity (for example, ACME) and <LocalID> is a value that uniquely identifies each JobSettingData instance that is instantiated on a system.

The purpose for <LocalID> is to provide some form of uniqueness within the context of different JobSettingData instances within the system. In practice, <LocalID> could be an incremented counter or a timestamp in combination with other test identifiers or factors.

7.5.2 CIM_JobSettingData.DeleteOnCompletion

This boolean property indicates whether the job should be automatically deleted upon completion. The property is mandatory. When the value is TRUE, the job shall be deleted after the CIM_ConcreteJob.TimeBeforeRemoval time interval. When the value is FALSE, the job shall be deleted by an DeleteInstance operation.

NOTE When the value of the CIM_DiagnosticServiceJobCapabilities.DeleteJobSupported property is FALSE, the value of CIM_JobSettingData.DeleteOnCompletion shall be TRUE.

EXPERIMENTAL

7.5.3 CIM_JobSettingData.InteractiveTimeout

This interval datetime property shall have a value if the CIM_DiagnosticTest.Characteristics property contains the value of 3 (Is Interactive). Otherwise, this property is ignored.

When an interactive diagnostic test prompts a client for a response, the test may define the time interval it shall wait for a client to respond. While waiting for the client to respond, the CIM_ConcreteJob.JobState property associated to the diagnostic test has the value of 12 (Query Pending). This property indicates the time interval that an interactive diagnostic test will wait for a client to respond.

If the client fails to respond within the specified time interval, the test may terminate, resume using default responses, or wait another time interval. The default value is 15 minutes (00000000001500.000000:000).

7.5.4 CIM_JobSettingData.TerminateOnTimeout

This boolean property shall have a value if the CIM_DiagnosticTest.Characteristics property contains the value of 3 (Is Interactive). Otherwise, this property is ignored.

This property defines the behavior when a client fails to respond within the time interval defined by CIM_JobSettingData.InteractiveTimeout. If this value is TRUE the job will terminate when the InteractiveTimeout is exceeded on the last retry, if applicable. If FALSE the job will use DefaultInputs.

7.5.5 CIM_JobSettingData.DefaultInputValues

This string array property shall contain one or more values if the CIM_DiagnosticTest.Characteristics property contains the value of 3 (Is Interactive) and the value of CIM_DiagnosticServiceJobCapabilities.DefaultValuesSupported is TRUE. Otherwise, this property is ignored.

If the client fails to respond within the specified time interval, the diagnostic test may terminate, resume using default responses, or wait another time interval.

This string array property contains the default values for client responses when the test resumes. The name of each DefaultInputValues array element is defined at the same array index in the CIM_JobSettingData.DefaultInputNames string array.

7.5.6 CIM_JobSettingData.DefaultInputNames

This string array property shall contain one or more values if the CIM_DiagnosticTest.Characteristics property contains the value of 3 (Is Interactive) and the value of CIM_DiagnosticServiceJobCapabilities.DefaultValuesSupported is TRUE. Otherwise, this property is ignored.

This string array property contains the possible argument names requested by the diagnostic test. The default value of each DefaultInputNames array element is defined at the same array index in the CIM_JobSettingData.DefaultInputValues string array.

7.5.7 CIM_JobSettingData.ClientRetries

This integer property indicates the number of times the diagnostic test will prompt the client for the same response after the client fails to invoke the CIM_ConcreteJob.ResumeWithInput( ) or CIM_ConcreteJob.ResumeWithAction( ) extrinsic method within a specified period of time . A value of zero for this property indicates that the diagnostic test will issue a second prompt for the same response.

This property is required if CIM_DiagnosticTest.Characteristics has the value of 3 (Is Interactive). Otherwise, this property value is ignored.

7.5.8 CIM_JobSettingData.RunInSilentMode

This boolean property indicates whether the diagnostic test will not prompt the client for responses even though CIM_DiagnosticTest.Characteristics contains the value of 3 (Is Interactive). When the value is TRUE, no prompts are issued. Instead, the diagnostic test will execute using the default values defined in CIM_JobSettingData. When the value is FALSE, the interactive diagnostic test will prompt the client for a response.

If CIM_DiagnosticServiceJobCapabilties.SilentModeSupported has the value of FALSE, this property is ignored.

EXPERIMENTAL

7.6 Interactive options

An interactive diagnostic test is controlled by properties in the optional CIM_DiagnosticServiceJobCapabilities class, the properties in the mandatory CIM_JobSettingData (Default) class, and the JobSetting input parameter used when the client invokes the CIM_DiagnosticTest.RunDiagnosticService( ) extrinsic method.

When a diagnostic test prompts the client for a response, the time interval a test shall wait for a client to respond is determined by the values in the following properties. Table 3 shows the behavior when the following properties and JobSetting parameter have certain and possibly conflicting values.

- CIM_DiagnosticServiceJobCapabilities.InteractiveTimeoutMax
- CIM_DiagnosticServiceJobCapabilities.ClientRetriesMax
- CIM_JobSettingData.InteractiveTimeout
- CIM_JobSettingData.TerminateOnTimeout

When a timeout occurs or when running in silent mode, the diagnostic test may resume using default values for the arguments that the client should have provided. How default values can be used is determined by the combination of values in the following properties: Table 3 shows the behavior when the following properties and JobSetting parameter have certain and possibly conflicting values.

- CIM_DiagnosticServiceJobCapabilities.DefaultValuesSupported
- CIM_JobSettingData.DefaultInputValues
- CIM_JobSettingData.DefaultInputNames

An interactive diagnostic test can be configured to run using default values shown above without prompting the client for any responses using the values in the following properties. Table 3 shows the behavior when the following properties have certain values.

- CIM_DiagnosticServiceJobCapabilities.SilentModeSupported
- CIM_JobSettingData.RunInSilentMode

672 **Table 3 – Interactive options**

Capability	JobSettingData (Default)	JobSetting	Behavior
InteractiveTimeoutMax	InteractiveTimeout <= InteractiveTimeoutMax	InteractiveTimeout <= InteractiveTimeoutMax	Use JobSetting value
		InteractiveTimeout >= InteractiveTimeoutMax	Use Capability value (2)
DefaultValuesSupported = TRUE	DefaultInputValues = non-NULL DefaultInputNames = non-NULL	DefaultInputValues = non-NULL DefaultInputNames = non-NULL	Use JobSetting
		DefaultInputValues = NULL DefaultInputNames = NULL	Use JobSettingData.(2)
DefaultValuesSupported = FALSE	DefaultInputValues = NULL DefaultInputNames = NULL	DefaultInputValues = NULL DefaultInputNames = NULL	The client shall supply input values.
		DefaultInputValues = non-NULL DefaultInputNames = non-NULL	JobSetting is ignored. (3)
DefaultValuesSupported = TRUE SilentModeSupported = TRUE	DefaultInputValues = non-NULL DefaultInputNames = non-NULL	DefaultInputValues = non-NULL DefaultInputNames = non-NULL RunInSilentMode = TRUE	Run in silent mode using JobSetting default values.
	DefaultInputValues = non-NULL DefaultInputNames = non-NULL	DefaultInputValues = NULL DefaultInputNames = NULL RunInSilentMode = TRUE	Run in silent mode using JobSettingData default values.
ClientRetriesMax = N DefaultValuesSupported = FALSE	TerminateOnTimeout = TRUE	TerminateOnTimeout = TRUE	The Job will terminate after N tries to solicit input.
		TerminateOnTimeout = FALSE	Use JobSettingData (2)
ClientRetriesMax = N	ClientRetries = M where M <= N	ClientRetries = R where R <= N	After R retries, the job terminates or runs with defaults. (1)
		ClientRetries = NULL	After M retries, the job terminates or runs with defaults. (1) (2)

Capability	JobSettingData (Default)	JobSetting	Behavior
ClientRetriesMax = N	ClientRetries = M where M <= N	ClientRetries = R where R > N	After M retries, the job terminates or runs with defaults. (1) (2)
InteractiveTimeoutMax	InteractiveTimeout > InteractiveTimeoutMax		Undefined
DefaultValuesSupported = TRUE	DefaultInputValues = NULL DefaultInputNames = NULL		Undefined
DefaultValuesSupported = FALSE	DefaultInputValues = non-NULL DefaultInputNames = non-NULL		Undefined
DefaultValuesSupported = TRUE SilentModeSupported = TRUE	DefaultInputValues = NULL DefaultInputNames = NULL RunInSilentMode = TRUE or FALSE	DefaultInputValues = NULL DefaultInputNames = NULL RunInSilentMode = TRUE	Undefined
DefaultValuesSupported = FALSE SilentModeSupported = TRUE			Undefined
DefaultValuesSupported = FALSE ClientRetries = N	TerminateOnTimeout = FALSE		Undefined
ClientRetries = N	ClientRetries = M where M > N		Undefined

(1) The job will wait one InteractiveTimeout for each response. After the timeout, another prompt is issued.

(2) An alert indication (DIAG39) is sent indicating that the JobSetting was reset. The overridden effective JobSetting should be logged.

(3) An alert indication (DIAG40) is sent indicating that the default values were not used. The overridden effective JobSetting should be logged.

7.7 Job deletion options

To start a diagnostic test, the client invokes the CIM_DiagnosticTest.RunDiagnosticService extrinsic method, which returns an instance of CIM_ConcreteJob. When a diagnostic test completes execution, the CIM_ConcreteJob instance will be deleted. When and how the CIM_ConcreteJob instance is deleted is controlled by properties in the optional CIM_DiagnosticServiceJobCapabilities class (the first column), the properties in the mandatory CIM_JobSettingData (Default) class (the second column), and the optional

685 JobSetting input parameter (the third column), which is an embedded CIM_JobSettingData instance,
686 used when the client invokes the CIM_DiagnosticTest.RunDiagnosticService( ) extrinsic method. Table 4
687 shows the behavior (the fourth column) when these following properties have certain and possibly
688 conflicting values.

689 • CIM_DiagnosticServiceJobCapabilties.DeleteJobSupported

690 • CIM_DiagnosticServiceJobCapabilties.CleanupInterval

691 • CIM_JobSettingData.DeleteOnCompletion

692 • CIM_JobSettingData.TimeBeforeRemoval

693 Since the JobSetting input parameter is optional, its value may by NULL In this case, the
694 CIM_DiagnosticTest.RunDiagnosticService extrinsic method shall use the values of the mandatory
695 CIM_JobSettingData (Default) instance.

696 An instance of CIM_DiagnosticServiceJobCapabilities may not exist because its implementation is
697 optional. In this case, ignore the first column when interpreting Table 4.

698 **Table 4 – Job deletion options**

Capability	JobSettingData (Default)	JobSetting	Behavior
DeleteJobSupported = TRUE AND CleanupInterval = non-NULL	DeleteOnCompletion = TRUE or FALSE	DeleteOnCompletion = TRUE	The provider deletes the instance TimeBeforeRemoval after the job completes.
		DeleteOnCompletion = FALSE	The client should delete the instance. The provider may delete the instance CleanupInterval after the job completes.
DeleteJobSupported = TRUE AND CleanupInterval = NULL	DeleteOnCompletion = TRUE	DeleteOnCompletion = TRUE	The provider deletes the instance TimeBeforeRemoval after the job completes.
		DeleteOnCompletion = FALSE	The client should delete the instance. CleanupInterval is ignored.
DeleteJobSupported = FALSE AND CleanupInterval = non-NULL	DeleteOnCompletion = TRUE	DeleteOnCompletion = TRUE	The provider deletes the instanceTimeBeforeRemoval after the job completes
		DeleteOnCompletion = FALSE	JobSetting.DeleteOnCompletion is reset to TRUE. The provider deletes the instance TimeBeforeRemoval after the job completes. (1)
DeleteJobSupported = FALSE AND CleanupInterval = NULL	DeleteOnCompletion = FALSE		Undefined
DeleteJobSupported = FALSE AND CleanupInterval = non-NULL	DeleteOnCompletion = FALSE		Undefined

699 (1) An alert indication (DIAG39) is sent indicating that the JobSetting was reset. The overridden
700 effective JobSetting should be logged.

701 # 8 Methods

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

704 ## 8.1 Profile conventions for operations

705 The default list of operations shall be as mandated in DSP1103, subclauses 8.1.

706 Support for operations for each profile class (including associations) shall be as mandated in DSP1103,
707 subclauses 8.3 through 8.7.

8.2 CIM_ConcreteJob

All operations are supported as for CIM_ConcreteJob in DSP1103, subclause 8.2. Additionally, the DeleteInstance operation shall be supported when the CIM_JobSettingData.DeleteOnCompletion property has the value of FALSE.

8.2.1 CIM_ConcreteJob.RequestStateChange( )

All CIM_DiagnosticService.RunDiagnosticService( ) calls will return a reference to a CIM_ConcreteJob instance, which represents the diagnostic execution. The CIM_ConcreteJob.RequestStateChange( ) method is invoked to control the diagnostic program execution. The RequestedState input parameter specifies the new desired state (Start, Suspend, Kill, Terminate).

Otherwise, the behavior of this extrinsic method shall be as mandated in DSP1103, subclause 8.2.

Before invoking this method, a client examines CIM_DiagnosticServiceJobCapabilities.RequestedStatesSupported to determine the values to use for the RequestedState input parameter. The RequestStateChange( ) extrinsic method shall change the JobState value if the transition is successfully performed.

EXPERIMENTAL

8.2.2 CIM_ConcreteJob.ResumeWithInput( )

The CIM_ConcreteJob.ResumeWithInput( ) extrinsic method is invoked to resume the diagnostic program execution when it has a JobState of 12 (Query Pending).

The return values are specified in Table 5. The input parameter is specified in Table 6. No output parameters are defined. No standard messages are defined.

For an interactive test, the CIM_ConcreteJob provider prompts the client to respond with DIAG34 message, which is a comma separated string of argument names. The client calls CIM_ConcreteJob.ResumeWithInput( ) to respond with values in the Inputs string array. The first value in the Inputs string array corresponds to the first argument in the DIAG34 message, and so on

The Inputs string array shall have a value for each requested argument. A NULL value shall not be used. To use a default value for a requested argument, the client looks for a matching name in CIM_JobSettingData.DefaultInputNames. If found, the client uses the corresponding value from CIM_JobSettingData.DefaultInputValues; that is, at the same array index.

If the client invokes the CIM_ConcreteJob.ResumeWithInput( ) extrinsic method where the Inputs string array argument has invalid values or not enough values, then the CIM_ConcreteJob provider returns 5 (Invalid Parameter) and sets CIM_ConcreteJob.JobState = 10 (Exception) provided no more retries remain.

If CIM_ConcreteJob.ResumeWithInput( ) returns value=3 (Can NOT complete within Timeout Period), the client can retry provided more retries remain.

742 **Table 5 – ResumeWithInput( ) method: Return code values**

Value	Description
0	Completed with No Error
	The ResumeWithInput was accepted and the job has resumed. The JobState has changed from “12” (Query Pending) to “4” (Running)
2	Unknown/Unspecified Error
	The JobState was “12” (Query Pending) and the inputs were valid, but the request failed for other reasons.
3	Can NOT complete within Timeout Period
4	Failed
5	Invalid Parameter
6	JobState not Query Pending
32768..65535	Vendor specific

743 **Table 6 – ResumeWithInput( ) method: Parameters**

Qualifiers	Name	Type	Description/Values
IN	Inputs	String[]	The client inputs being requested by the job when its state changed to 12 (Query Pending).

744 ### 8.2.3 CIM_ConcreteJob.ResumeWithAction( )

745 The CIM_ConcreteJob.ResumeWithAction( ) extrinsic method is invoked to resume the diagnostic
746 program execution when it has a JobState of 12 (Query Pending).

747 The return values are specified in Table 7. No input or output parameters are defined. No standard
748 messages are defined.

749 For an interactive test, the CIM_ConcreteJob provider prompts the client to respond with DIAG35
750 message. The client invokes the CIM_ConcreteJob.ResumeWithAction( ) extrinsic method when no
751 arguments are requested. For example, the diagnostic test might prompt the user to attach the network
752 cable before allowing the test to proceed.

753 **Table 7 – ResumeWithAction( ) method: Return code values**

Value	Description
0	Completed with No Error
	The ResumeWithInput was accepted and the job has resumed. The JobState has changed from “12” (Query Pending) to “4” (Running).
2	Unknown/Unspecified Error
	The JobState was “12” (Query Pending) and the inputs were valid, but the request failed for other reasons.
3	Can NOT complete within Timeout Period
4	Failed
6	JobState not Query Pending
32768..65535	Vendor specific

754 ## 8.3 CIM_DiagnosticServiceJobCapabilities

755 ### 8.3.1 CreateGoalSettings( )

756 The CIM_DiagnosticServiceJobCapabilities.CreateGoalSettings( ) method, which is inherited from
757 CIM_Capabilties, is invoked in the context of a specific CIM_DiagnosticServiceJobCapabilities instance.

758 This method is used to create a CIM_JobSettingData instance using the
759 CIM_DiagnosticServiceJobCapabilities as a template. The purpose of this method is to create a
760 CIM_JobSettingData based on the CIM_DiagnosticServiceJobCapabilities on which this method is
761 invoked and has properties set in line with those CIM_DiagnosticServiceJobCapabilities.

762 The return values are specified in Table 8. The parameters are specified in Table 9. No standard
763 messages are defined.

764 **Table 8 – CreateGoalSettings( ) method: Return code values**

Value	Description
0	Completed with No Error
1	Not supported
2	Unknown
3	Timeout
4	Failed
5	Invalid Parameter
6	Alternative Proposed
32768..65535	Vendor specific

765 **Table 9 – CreateGoalSettings( ) method: Parameters**

Qualifiers	Name	Type	Description/Values
IN	TemplateGoalSettings[ ]	String	An array of CIM_JobSettingData embedded instances that reflect what the client wants. This parameter may be NULL. If NULL, the method returns a setting that conforms to the CIM_DiagnosticServiceJobCapabilities.
IN/OUT	SupportedGoalSettings[ ]	String	An array of CIM_JobSettingData embedded instances that are consistent with the CIM_DiagnosticServiceJobCapabilities and are closest matches to the input TemplateGoalSettings

766 **EXPERIMENTAL**

767 8.4 CIM_MethodResult

768 All operations are supported as for CIM_MethodResult in DSP1103.

769 8.5 CIM_OwningJobElement

770 All operations are supported as for CIM_OwningJobElement in DSP1103.

771 8.6 CIM_AffectedJobElement

772 All operations are supported as for CIM_AffectedJobElement in DSP1103.

773 8.7 CIM_AssociatedJobMethodResult

774 All operations are supported as for CIM_AssociatedJobMethodResult in DSP1103.

775 8.8 CIM_HostedDependency

776 All operations are supported as for CIM_HostedDependency in DSP1103.

777 8.9 CIM_RegisteredProfile

778 All operations are supported as for CIM_RegisterdProfile in the *Profile Registration Profile* (DSP1033).

779 8.10 CIM_JobSettingData

780 Table 10 lists operations that either have special requirements beyond those from DSP0200 or shall not
781 be supported.

782 **Table 10 – Operations: CIM_JobSettingData**

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

783 ## 8.11 CIM_ElementSettingdData

784 Table 11 lists operations that either have special requirements beyond those from DSP0200 or shall not
785 be supported.

786 **Table 11 – Operations: CIM_ElementSettingData**

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

787 ## 8.12 CIM_ElementCapabilities

788 Table 12 lists operations that either have special requirements beyond those from DSP0200 or shall not
789 be supported.

790 **Table 12 – Operations: CIM_ElementCapabilities**

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

791 ## 8.13 CIM_DiagnosticTest.RunDiagnosticService( )

792 DSP1002 describes this extrinsic method. This subclause describes how the
793 CIM_DiagnosticServiceJobCapabilites, the CIM_JobSettingData (Default), and the JobSetting input
794 parameter affects the execution of this extrinsic method.

795 The CIM_JobSettingData (Default) is mandatory. The CIM_DiagnosticServiceJobCapabilities and the
796 JobSetting parameter of the RunDiagnosticService method are optional. If the
797 CIM_DiagnosticServiceJobCapabilities is not implemented, the client application cannot alter the default
798 CIM_JobSettingData for the diagnostic test and the JobSetting parameter should be NULL or set to the
799 default CIM_JobSettingData. If the client application sets the JobSetting parameter to values that conflict
800 with the default CIM_JobSettingData, the test will not fail, but the JobSetting parameter will be reset to the

default values (the “effective” JobSetting) and a warning alert message will be issued. The effective JobSetting parameter values will also be logged in the associated CIM_DiagnosticLog.

If CIM_DiagnosticServiceJobCapabilities is implemented, the client application may specify values in the JobSetting parameter that conform to the corresponding capability. For example, the client application may specify an InteractiveTimeout that is equal or less than the InteractiveTimeoutMax..If the client application specifies a value that is in conflict with the options allowed by the CIM_DiagnosticServiceJobCapabilities for the diagnostic test, the conflicting value will be reset to one of two values: The value in the default JobSettingData or the maximum allowed by the CIM_DiagnosticServiceJobCapabilities. If the client invokes this extrinsic method and the JobSetting parameter has the value of NULL, the default CIM_JobSettingData will be used. In either case, if any value was changed, an alert message will be issued. Whether or not a value was changed, the effective Jobsetting used by the diagnostic test execution will be logged in the CIM_DiagnosticLog.

Table 13 shows the behavior for different combinations of CIM_DiagnosticServiceJobCapabilities, CIM_JobSettingData (Default), and the JobSetting parameter.

Table 13 – Job setting options

Capabilities	JobSettingData (Default)	JobSetting	Behavior
Absent	Present	NULL	Use JobSettingData (Default).
Absent	Present	No conflict in values exist	Use JobSetting.
		Conflict in values exist	Use JobSettingData (Default). An alert is issued.
Present	Present	NULL	Use JobSettingData (Default).
Present	Present	No conflict in values exist	Use JobSetting.
		Conflict in values exist	JobSetting is modified to conform to the capabilities. An alert is issued.
	Absent		Undefined.
Present	Present but conflicting		Undefined.

The effective JobSetting used is logged.

EXPERIMENTAL

9 Use cases (Informative)

This clause contains use cases for the *Diagnostic Job Control Profile* that describes how a diagnostic test behaves and interacts with a client. An interactive diagnostic test is a CIM_DiagnosticTest instance where its Characteristics property contains the value 3 (Is Interactive).

9.1 Use case summary

Table 14 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.

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

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

828 **Table 14 – Diagnostic test use cases**

Category	Scenarios	Description
User input required	The test requires a single response for a single value. The client responds with valid values. See 9.2.1.	Some interactive diagnostic tests require the user to respond with input values before the test can proceed. See 9.2.
	The test requires a single response for multiple values. The client responds with valid values. See 9.2.2.	
	The test requires multiple responses. After the client responds with valid values, the test runs to partial completion and then prompts for another response. The client responds to each prompt with valid values. See 9.2.3.	
	The client fails to respond to a prompt. See 9.2.4.	
	The client responds to a prompt with invalid values. See 9.2.5.	
	The client responds to a prompt with an insufficient number of values. See 9.2.6.	
User action required	The test requires a single response. The client responds. See 9.3.1.	Some interactive diagnostic tests require the user to perform an action before the test can proceed. See 9.3.
	The test requires multiple responses required before running the test. The client responds to each prompt. See 9.3.2.	
	The test requires multiple responses. After the client responds, the test runs to partial completion and then prompts for another response. The client responds to each prompt. See 9.3.3.	
	The client fails to respond to a prompt. See 9.3.4.	
Silent mode operation		This profile defines the ability to run interactive diagnostic tests without user interaction by using predefined default values. See 9.4.
Finding a diagnostic job	Find all diagnostic jobs on a system	This profile defines the sequence of operations to perform this task. See 9.5.1.
	Find all diagnostic jobs for a ManagedElement	This profile defines the sequence of operations to perform this task. See 9.5.2.
Configuring a diagnostic job	Get default job settings	This profile defines the sequence of operations to perform this task. See 9.6.1.

Category	Scenarios	Description
	Create job settings	This profile defines the sequence of operations to perform this task. See 9.6.2.
Control a job for a diagnostic test	Suspend a job for a diagnostic test	See 9.7.1.
	Resume a job for a diagnostic test	See 9.7.2.
	Terminate a job for a diagnostic test	See 9.7.3.
	Kill a job for a diagnostic test	Abort a running diagnostic immediately, with no attempt to perform a clean termination. See 9.7.4.
Delete a job	Client deletes a job	See 9.8.1.
	Provider deletes a job	See 9.8.2.

Before performing the use cases in this profile, it is assumed that a client has already utilized the use case methodology defined in DSP1002 to discover the DiagnosticTest instance.

To start a test, the client invokes the DiagnosticTest.RunDiagnosticService( ) extrinsic method which returns 0 (Success) and the objectpath of a ConcreteJob instance with ConcreteJob.JobState = 4 (Running) and CIM_ConcreteJob.PercentComplete = 0. Thereafter, ConcreteJob manages the diagnostic test execution. Additionally, a client monitors and interacts with the diagnostic test via the returned ConcreteJob instance.

NOTE An interactive diagnostic test may prompt a client more than once during test execution where some prompts require user input while others do not.

In the following examples, responses are enclosed in brackets. [Enter] indicates that the client pressed the Enter key, typically to select the default. <timeout> indicates that the client did not respond.

9.2 User input required

For an interactive test where user input is required, the ConcreteJob provider prompts the client to respond with DIAG34 message, which is a comma separated string of argument names. The client calls ConcreteJob.ResumeWithInput( ) to respond with values in the Inputs string array. The first value in the Inputs string array corresponds to the first argument in the DIAG34 message, and so on.

9.2.1 Single prompt and response has a valid value

- How many minutes do you want the test to run? [20] (Test starts)

1) Client calls DiagnosticTest.RunDiagnosticService( ) which returns 0 (Success) and the objectpath of the ConcreteJob instance and starts the diagnostic test. ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 0.
2) ConcreteJob provider prompts client to respond with a DIAG34 message, which requests a value for the Minutes argument.
3) ConcreteJob provider sets ConcreteJob.JobState = 12 (Query Pending) to wait for client response.
4) Client calls ConcreteJob.ResumeWithInput( ) to respond with input argument Inputs[0]=”20” for the Minutes argument requested by DIAG34 message.
5) ConcreteJob provider sets ConcreteJob.JobState = 4 (Running).
6) ConcreteJob.ResumeWithAction( ) returns 0 (“Completed with No Error”) and test execution starts.

7) After the test completes successfully, the ConcreteJob provider sets ConcreteJob.JobState = 7 (Completed), ConcreteJob.OperationalStatus= 17 (Completed) and ConcreteJob PercentComplete=100.

9.2.2 Single prompt and response has multiple valid values

- Which CPU speeds in GHz to you want to test? [2.4,3.0] (Test starts)

1) Client calls DiagnosticTest.RunDiagnosticService( ) which returns 0 (Success) and the objectpath of the ConcreteJob instance and starts the diagnostic test. ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 0.

2) ConcreteJob provider prompts client to respond with a DIAG34 message, which requests a value for the CPUSpeed1 and CPUSpeed2 arguments.

3) ConcreteJob provider sets ConcreteJob.JobState = 12 (Query Pending) to wait for client response.

4) Client calls ConcreteJob.ResumeWithInput( ) to respond with input argument Inputs[0]=”2.4” for the CPUSpeed1 argument and Inputs[1]=”3.0” for the CPUSpeed2 argument as requested by the DIAG34 message.

5) ConcreteJob provider sets ConcreteJob.JobState = 4 (Running).

6) ConcreteJob.ResumeWithAction( ) returns 0 (“Completed with No Error”) and test execution starts.

7) After the test completes successfully, the ConcreteJob provider sets ConcreteJob.JobState = 7 (Completed), ConcreteJob.OperationalStatus= 17 (Completed) and ConcreteJob PercentComplete=100.

9.2.3 Multiple prompts and responses required with partial test execution after each

- Which network port do you want to test? [2] (Test execution starts)

- Which network port do you want to test next? [3] (Test execution resumes)

1) Client calls DiagnosticTest.RunDiagnosticService( ) which returns 0 (Success) and the objectpath of the ConcreteJob instance and starts the diagnostic test. ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 0.

2) ConcreteJob provider prompts client to respond with a DIAG34 message, which requests a value for the Port1 argument.

3) ConcreteJob provider sets ConcreteJob.JobState = 12 (Query Pending) to wait for client response.

4) Client calls ConcreteJob.ResumeWithInput( ) to respond with input argument Inputs[0]=”2” for the Port1 argument requested by DIAG34 message.

5) ConcreteJob provider sets ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 25 and test execution starts.

6) ConcreteJob.ResumeWithAction( ) returns 0 (“Completed with No Error”) and test execution resumes.

7) Test execution finishes with port 2. ConcreteJob provider sets ConcreteJob.PercentComplete = 50 and execution continues.

8) ConcreteJob provider prompts client to respond with a DIAG34 message, which requests a value for the Port2 argument.

9) ConcreteJob provider sets ConcreteJob.JobState = 12 (Query Pending) to wait for client response.

10) Client calls ConcreteJob.ResumeWithInput( ) to respond with input argument Inputs[0]=”3” for the Port2 argument requested by DIAG34 message.

11) ConcreteJob provider sets ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 50 and test execution starts.

12) After the test completes successfully, the ConcreteJob provider sets ConcreteJob.JobState = 7 (Completed) , ConcreteJob.OperationalStatus= 17 (Completed) and ConcreteJob PercentComplete=100.

9.2.4 Client does not respond to a prompt

- Which network port do you want to test? <timeout>

1) Client calls DiagnosticTest.RunDiagnosticService( ) which returns 0 (Success) and the objectpath of the ConcreteJob instance and starts the diagnostic test. ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 0.

2) ConcreteJob provider prompts client to respond with a DIAG34 message, which requests a value for the Port argument.

3) ConcreteJob provider sets ConcreteJob.JobState = 12 (Query Pending) to wait for client response.

4) When JobSettingData.InteractiveTimeout is exceeded, the ConcreteJob provider examines the value of JobSettingData.ClientRetriesMax.

5) If the value of JobSettingData.ClientRetriesMax is zero, or if the number of retries has been exceeded, the ConcreteJob provider sets ConcreteJob.JobState = 10 (Exception) provided JobSettingData.TerminateOnTimeout=TRUE.

6) If the value of JobSettingData.ClientRetriesMax is non-zero, steps 2, 3, 4, and 5 are repeated.

NOTE If the value of JobSettingData.ClientRetriesMax is non-zero, the ConcreteJob provider monitors how many timeouts that have occurred.

9.2.5 Client responds with an invalid value

- Which network port do you want to test? [-1]

1) Client calls DiagnosticTest.RunDiagnosticService( ) which returns 0 (Success) and the objectpath of the ConcreteJob instance and starts the diagnostic test. ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 0.

2) ConcreteJob provider prompts client to respond with a DIAG34 message, which requests a value for the Port argument.

3) Client calls ConcreteJob.ResumeWithInput( ) to respond with input argument Inputs[0]=”-1” for the Port1 argument requested by DIAG34 message.

4) The ConcretJob provider detects that Inputs[0]=”-1” is invalid.

5) ConcreteJob provider responds to ConcreteJob.ResumeWithInput with 5 (Invalid Parameter).

6) ConcreteJob provider sets ConcreteJob.JobState= 10 (Exception).

9.2.6 Client does not respond with enough valid values

- Which CPU speeds in GHz to you want to test? [2.4]

1) Client calls DiagnosticTest.RunDiagnosticService( ) which returns 0 (Success) and the objectpath of the ConcreteJob instance and starts the diagnostic test. ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 0.

2) ConcreteJob provider prompts client to respond with a DIAG34 message, which requests a value for the CPUSpeed1 and CPUSpeed2 arguments.

3) Client calls ConcreteJob.ResumeWithInput( ) to respond with input argument Inputs[0]=”2.4” for the CPUSpeed1 argument requested by DIAG34 message but does not provide a value for Inputs[1] for the CPUSpeed2 argument.

4) The ConcretJob provider detects that Inputs[1] has no value.

5) ConcreteJob provider responds to ConcreteJob.ResumeWithInput with 5 (Invalid Parameter).

6) ConcreteJob provider sets ConcreteJob.JobState= 10 (Exception).

9.3 User action required

For an interactive diagnostic test where user action is required, the ConcreteJob provider prompts client to respond with a DIAG35 message. The client responds by invoking the ConcreteJob.ResumeWithAction( ) extrinsic method.

9.3.1 Single prompt and response required

- Press any key when the network cable has been attached. [Enter] (Test execution starts)

1) Client calls DiagnosticTest.RunDiagnosticService( ) which returns 0 (Success) and the objectpath of the ConcreteJob instance and starts the diagnostic test. ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 0.

2) ConcreteJob provider prompts client to respond with a DIAG35 message.

3) ConcreteJob provider sets ConcreteJob.JobState = 12 (Query Pending) to wait for client response.

4) Client calls ConcreteJob.ResumeWithAction( ) after attaching the cable.

5) ConcreteJob provider sets ConcreteJob.JobState = 4 (Running).

6) ConcreteJob.ResumeWithAction( ) returns 0 (“Completed with No Error”) and test execution starts.

7) After the test completes successfully, the ConcreteJob provider sets ConcreteJob.JobState = 7 (Completed), ConcreteJob.OperationalStatus= 17 (Completed), and ConcreteJob PercentComplete=100.

9.3.2 Multiple prompts and responses required before running the test

- Press any key when the network cable has been attached. [Enter]

- Press any key after the CD has been inserted into the drive. [Enter] (Test execution starts)

1) Client calls DiagnosticTest.RunDiagnosticService( ) which returns 0 (Success) and the objectpath of the ConcreteJob instance and starts the diagnostic test. ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 0.

2) ConcreteJob provider prompts client to respond with a DIAG35 message.

3) ConcreteJob provider sets ConcreteJob.JobState = 12 (Query Pending) to wait for client response.

4) Client calls ConcreteJob.ResumeWithAction( ) after attaching the cable.

5) ConcreteJob.ResumeWithAction( ) returns 0 (“Completed with No Error”).

6) ConcreteJob provider prompts client to respond with a DIAG35 message.

7) ConcreteJob provider sets ConcreteJob.JobState = 12 (Query Pending) to wait for client response.

8) Client calls ConcreteJob.ResumeWithAction( ) after inserting the CD into the drive.

9) ConcreteJob.ResumeWithAction( ) returns 0 (“Completed with No Error”)

10) ConcreteJob provider sets ConcreteJob.JobState = 4 (Running) and test execution starts.

11) After test completes successfully, ConcreteJob provider sets ConcreteJob.JobState = 7 (Completed) and ConcreteJob PercentComplete=100.

9.3.3 Multiple prompts and responses required with partial test execution after each

- Press any key when the network cable has been attached. [Enter]. (Test execution starts)
- Press any key when the LED is on. [Enter] (Test execution resumes)
- Press any key when the LED is off. [Enter] (Test execution resumes)

1) Client calls DiagnosticTest.RunDiagnosticService( ) which returns 0 (Success) and the objectpath of the ConcreteJob instance and starts the diagnostic test. ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 0.

2) ConcreteJob provider prompts client to respond with a DIAG35 message.

3) ConcreteJob provider sets ConcreteJob.JobState = 12 (Query Pending) to wait for client response.

4) Client calls ConcreteJob.ResumeWithAction( ) after attaching the cable.

5) ConcreteJob.ResumeWithAction( ) returns 0 (“Completed with No Error”).

6) ConcreteJob provider sets ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 25 and test execution resumes.

7) ConcreteJob provider prompts client to respond with a DIAG35 message.

8) ConcreteJob provider sets ConcreteJob.JobState = 12 (Query Pending) to wait for client response.

9) Client calls ConcreteJob.ResumeWithAction( ) after verifying the LED is ON.

10) ConcreteJob.ResumeWithAction( ) returns 0 (“Completed with No Error”)

11) ConcreteJob provider sets ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 50 and test execution resumes.

12) ConcreteJob provider prompts client to respond with a DIAG35 message.

13) ConcreteJob provider sets ConcreteJob.JobState = 12 (Query Pending) to wait for client response.

14) Client calls ConcreteJob.ResumeWithAction( ) after verifying the LED is ON.

15) ConcreteJob.ResumeWithAction( ) returns 0 (“Completed with No Error”).

16) ConcreteJob provider sets ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 75 and test execution resumes.

17) After test completes successfully, ConcreteJob provider sets ConcreteJob.JobState = 7 (Completed) and ConcreteJob PercentComplete=100.

9.3.4 Client does not respond to a prompt

- Press any key when the network cable has been attached. <timeout>

1) Client calls DiagnosticTest.RunDiagnosticService( ) which returns 0 (Success) and the objectpath of the ConcreteJob instance and starts the diagnostic test. ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 0.

2) ConcreteJob provider prompts client to respond with a DIAG35 message.

3) ConcreteJob provider sets ConcreteJob.JobState = 12 (Query Pending) to wait for client response.

4) When JobSettingData.InteractiveTimeout is exceeded, the ConcreteJob provider examines the value of JobSettingData.ClientRetriesMax.

5) If the value of JobSettingData.ClientRetriesMax is zero, or if the number of retries has been exceeded, the ConcreteJob provider sets ConcreteJob.JobState = 10 (Exception) provided JobSettingData.TerminateOnTimeout=TRUE.

6) If the value of JobSettingData.ClientRetriesMax is non-zero, steps 2, 3, 4, and 5 are repeated.

NOTE If the value of JobSettingData.ClientRetriesMax is non-zero, the ConcreteJob provider monitors how many timeouts that have occurred.

9.4 Silent mode operation

An interactive test can be run as a non-interactive test; that is, the test does not prompt the client for responses. Instead, the ConcreteJob provider uses the default values from JobSettingData.DefaultInputNames and JobSettingData.DefaultInputValues to run the diagnostic test instead of sending a DIAG34 message to prompt the client.

NOTE Silent mode works only if JobSettingData.DefaultInputNames and JobSettingData.DefaultInputValues have all the values needed to run the interactive test.

An interactive test can only be run if the diagnostic test supports this capability; that is, if DiagnosticServiceJobCapabilities.SilentModeSupported = "true". To run in silent mode, the client sets JobSettingData.RunInSilentMode = "true" before invoking the DiagnosticTest.RunDiagnosticService( ) extrinsic method.

9.5 Finding diagnostic jobs

9.5.1 Finding all diagnostic jobs on a system

A client can find all jobs for diagnostic tests on a System as follows. Assume that the client starts at a known System on which the diagnostic tests are run.

1) From the System instance, the client calls the Associators operation using HostedDependency as the association class argument, and ConcreteJob is the result class argument, which returns all instances of ConcreteJob.

2) For each ConcreteJob instance returned, the client calls the Associators operation using OwningJobElement as the association class argument, DiagnosticTest as the result class argument, and OwningElement as the result role argument. The operation returns the associated DiagnosticTest.

9.5.2 Finding all diagnostic jobs for a ManagedElement

A client can find all jobs for a ManagedElement as follows. Assume that the client starts at a known ManagedElement instance, which represents the device to be tested.

1) From the ManagedElement instance, the client calls the Associators operation using AffectedJobElement as the association class argument, ConcreteJob as the result class argument, and AffectingElement as the result role argument.

2) For each ConcreteJob instance returned, the client calls the Associators operation using OwningJobElement as the association class argument, DiagnosticTest as the result class argument, and OwningElement as the result role argument. The operation returns the associated DiagnosticTest.

9.6 Configuring a diagnostic job

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

Note that the diagnostic default job settings are represented by a JobSettingData subclass that may have extensions. If the client is aware of the extensions, they may be modified as well. If the client is unaware, the default values should be used.

9.6.1 Getting the default job settings

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

1) From the DiagnosticTest instance, the client calls the Associators operation using ElementSettingData as the association class argument, JobSettingData as the result class argument, and SettingData. The operation returns JobSettingData instances.
2) For each JobSettingData instance, the client calls the References operation using ElementSettingData as the result class. The operation returns ElementSettingData instances.
3) For each ElementSettingData instance, the client determines whether the value of the ElementSettingData.ManagedElement property matches the DiagnosticTest name and the value of the ElementSettingData.IsDefault property is 1 (“Is Default”). If so, the JobSettingData instance represents the default job settings. The name of this JobSettingData instance may also be retrieved from ElementSettingData.SettingData property.

9.6.2 Creating the job settings

A client may create their own job settings to pass as an argument to the DiagnosticTest.RunDiagnosticService( ) extrinsic method as follows. Assume that the client starts at a known DiagnosticTest instance.

4) The client can attempt to discover the default job settings of the DiagnosticTest instance. The GetDefaultJobSettings use case (subclause 9.6.1) describes the necessary steps.
5) If the client wants to not use the default job settings, the client can attempt to find the associated DiagnosticServiceJobCapabilities instance by calling the Associators operation using ElementCapabilities as the association class argument, DiagnosticServiceJobCapabilities as the result class, and Capabilities as the result role.
6) If Step 2 returns an instance, the client calls the CreateGoalSettings( ) extrinsic method of the returned DiagnosticServiceJobCapabilities instance. This operation returns an instance of JobSettingData containing default values. The client can modify any property values as desired. If a range of values is permitted for a property, the client should use only those values indicated in the DiagnosticServiceJobCapabilities instance.
7) If Step 2 does not return an instance because the implementation of DiagnosticServiceJobCapabilities is optional, the client should use the default JobSettingData.

9.7 Execute and control a job for a diagnostic test

The DiagnosticServiceJobCapabilities.RequestedStatesSupported property indicates the permitted values of the RequestedState input parameter for the ConcreteJob.RequestStateChange( ) extrinsic method. Since DiagnosticServiceJobCapabilities is an optional class, a client may not be able to examine an instance to determine which values of RequestedState to use. If a client invokes ConcreteJob.RequestStateChange( ) to change to an unsupported state, the extrinsic method shall return 4097 (“Invalid State Transition”).

9.7.1 Suspend a job for a diagnostic test

The client can suspend the execution of a test by invoking the ConcreteJob.RequestStateChange( ) extrinsic method on the ConcreteJob instance that is returned from the DiagnosticTest.RunDiagnosticService( ) extrinsic method. Assume that the client starts at a known ConcreteJob instance and that a DiagnosticServiceJobCapabilities instance exists.

1) From the ConcreteJob instance, the client calls the Associators operation using OwningJobElement as the association class argument, DiagnosticTest as the result class argument, and OwningElement as the result role. The associated DiagnosticTest instance is returned.

2) From the ConcreteJob instance, the client calls the Associators operation using ElementCapabilities as the association class argument, DiagnosticServiceJobCapabilities as the result class argument, and Capabilities as the result role. The associated DiagnosticServiceJobCapabilities instance is returned.

3) The client examines the DiagnosticServiceJobCapabilities.RequestedStatesSupported property. If it contains the value of 3 (“Suspend”), the ConcreteJob can be suspended.

4) The client invokes the ConcreteJob.RequestStateChange( ) extrinsic method where input parameter RequestedState has the value of 3 (“Suspend”).

5) When the transition completes successfully, the ConcreteJob.JobState property shall have the value of 5 (“Suspended”) and ConcreteJob.TimeOfLastStateChange property shall be set to the current time.

9.7.2 Resume a job for a diagnostic

The client can resume the execution of a test that was previously suspended by invoking the ConcreteJob.RequestStateChange( ) extrinsic method on the ConcreteJob instance that is returned from the DiagnosticTest.RunDiagnosticService( ) extrinsic method. Assume that the client starts at a known ConcreteJob instance and that a DiagnosticServiceJobCapabilities instance exists.

1) From the ConcreteJob instance, the client calls the Associators operation using OwningJobElement as the association class argument, DiagnosticTest as the result class argument, and OwningElement as the result role. The associated DiagnosticTest instance is returned.

2) From the ConcreteJob instance, the client calls the Associators operation using ElementCapabilities as the association class argument, DiagnosticServiceJobCapabilities as the result class argument, and Capabilities as the result role. The associated DiagnosticServiceJobCapabilities instance is returned.

3) The client examines the DiagnosticServiceJobCapabilities. RequestedStatesSupported property. If it contains the value of 2 (“Start”), the ConcreteJob can be resumed.

4) The client invokes the ConcreteJob.RequestStateChange( ) extrinsic method where input parameter RequestedState has the value of 2 (“Start”).

5) When the transition completes successfully, the ConcreteJob.JobState property shall have the value of property to 4 (“Running”) and ConcreteJob.TimeOfLastStateChange property shall be set to the current time.

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

9.7.3 Terminate a job for a diagnostic

The client can cleanly terminate the execution of a test by invoking the ConcreteJob.RequestStateChange( ) extrinsic method on the ConcreteJob instance that is returned from the DiagnosticTest.RunDiagnosticService( ) extrinsic method. Assume that the client starts at a known ConcreteJob instance and that a DiagnosticServiceJobCapabilities instance exists.

1) From the ConcreteJob instance, the client calls the Associators operation using OwningJobElement as the association class argument, DiagnosticTest as the result class argument, and OwningElement as the result role. The associated DiagnosticTest instance is returned.

2) From the ConcreteJob instance, the client calls the Associators operation using ElementCapabilities as the association class argument, DiagnosticServiceJobCapabilities as the result class argument, and Capabilities as the result role. The associated DiagnosticServiceJobCapabilities instance is returned.

3) The client examines the DiagnosticServiceJobCapabilities. RequestedStatesSupported property. If it contains the value of 4 (“Terminate”), the ConcreteJob can be terminated.

4) The client invokes the ConcreteJob.RequestStateChange( ) extrinsic method where input parameter RequestedState has the value of 4 (“Terminate”).

5) When the transition completes successfully, the ConcreteJob.JobState property shall have the value of property to 8 (“Terminated”) and ConcreteJob.TimeOfLastStateChange property shall be set to the current time.

NOTE The JobState property may transition to 7 (“Shutting Down”) before the final transition to 8 (“Terminated”).

9.7.4 Kill a job for a diagnostic

The client can immediately abort the execution of a test, with no attempt to perform a clean shutdown, by invoking the ConcreteJob.RequestStateChange( ) extrinsic method on the ConcreteJob instance that is returned from the DiagnosticTest.RunDiagnosticService( ) extrinsic method. Assume that the client starts at a known ConcreteJob instance and that a DiagnosticServiceJobCapabilities instance exists.

1) From the ConcreteJob instance, the client calls the Associators operation using OwningJobElement as the association class argument, DiagnosticTest as the result class argument, and OwningElement as the result role. The associated DiagnosticTest instance is returned.

2) From the ConcreteJob instance, the client calls the Associators operation using ElementCapabilities as the association class argument, DiagnosticServiceJobCapabilities as the result class argument, and Capabilities as the result role. The associated DiagnosticServiceJobCapabilities instance is returned.

3) The client examines the DiagnosticServiceJobCapabilities. RequestedStatesSupported property. If it contains the value of 5 (“Kill”), the ConcreteJob can be aborted.

4) The client invokes the ConcreteJob.RequestStateChange( ) extrinsic method where input parameter RequestedState has the value of 5 (“Kill”).

5) When the transition completes successfully, the ConcreteJob.JobState property shall have the value of property to 9 (“Killed”) and ConcreteJob.TimeOfLastStateChange property shall be set to the current time.

9.8 Delete a job

When the optional DiagnosticServiceJobCapabilities instance exists and its DeleteJobSupported property has the value of TRUE, a client can control how and when the ConcreteJob instance associated to a specific diagnostic test execution is deleted. Assume that the client has verified that DiagnosticServiceJobCapabilities.DeleteJobSupported has the value of TRUE. Also assume that the client starts at a known DiagnosticTest instance. See Table 4 for further information.

NOTE A client may remove a job by invoking the ConcreteJob.RequestStateChange extrinsic method with input parameter having a value of 5 (“Kill”) or 4 (“Terminate”). Also, a client may perform the DeleteInstance operation on the ConcreteJob instance when JobSettingData.DeleteOnCompletion property has the value of FALSE.

9.8.1 Client deletes a job

To configure the ConcreteJob instance to be deleted by the client rather than by the provider

1) The client creates an embedded instance of JobSettingData where its DeleteOnCompletion property has the value of FALSE.

2) The client invokes the DiagnosticTest.RunDiagnosticService( ) extrinsic method where the JobSetting input parameter has the value of the embedded instance of JobSettingData created in the previous step.

3) After the diagnostic test completes or otherwise terminates, the ConcreteJob instance shall remain indefinitely until the client performs the DeleteInstance operation on it.

NOTE Assume that DiagnosticServiceJobCapabilities.CleanupInterval is NULL. Otherwise, the provider deletes the ConcreteJob instance..

9.8.2 Provider deletes a job

To configure the ConcreteJob instance to be deleted by the provider rather than by the client

1) The client creates an embedded instance of JobSettingData where its DeleteOnCompletion property has the value of TRUE.

2) The client may set the TimeBeforeRemoval property of the embedded JobSettingData instance. If set, the ConcreteJob instance is deleted based on the value of TimeBeforeRemoval after the job completes.

3) The client invokes the DiagnosticTest.RunDiagnosticService( ) extrinsic method where the JobSetting input parameter has the value of the embedded instance of JobSettingData created in the previous steps.

4) The provider removes the ConcreteJob instance at the time indicated by the value of the TimeBeforeRemoval property.

EXPERIMENTAL

10 CIM elements

Table 15 shows the instances of CIM elements for this profile. Instances of the CIM elements shall be implemented as described in Table 15. Clause 7 (“Implementation”) and Clause 8 (“Methods”) may impose additional requirements on these elements. See DSP1103 for other mandatory elements that must be implemented.

Table 15 – CIM Elements: Diagnostic Job Control Profile

Element Name	Requirement	Description
Classes		
CIM_AffectedJobElement	Optional	See 10.1.
CIM_ConcreteJob	Mandatory	See 10.2.
CIM_DiagnosticServiceJobCapabilities	Optional	See 10.3.
CIM_ElementCapabilities (Job)	Optional	See 10.4.
CIM_ElementSettingData (Default Job)	Optional	See 10.5.
CIM_HostedDependency	Mandatory	See 10.6.
CIM_JobSettingData (Client)	Optional	See 10.7.
CIM_JobSettingData (Default)	Mandatory	See 10.8.
CIM_OwningJobElement	Mandatory	See 10.9.
CIM_RegisteredProfile	Mandatory	See 10.10.
Indications		
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG9"	Conditional	This indication is mandatory if CIM_DiagnosticTest.Characteristics has the value of 3 (Is Interactive). Query Language:DMTF:CQL The test continued execution using a default response because a query timeout occurred.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG12"	Optional	Query Language: DMTF:CQL The test did not run because the job could not be started.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG19"	Mandatory	Query Language: DMTF:CQL The test was killed by the client
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG20"	Mandatory	Query Language: DMTF:CQL The test was terminated by the client
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG21"	Optional	Query Language: DMTF:CQL The test was suspended by the client

Element Name	Requirement	Description
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG34"	Optional	Query Language: DMTF:CQL The test solicits input from a client.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG35"	Optional	Query Language: DMTF:CQL The test solicits an action from a user.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG36"	Optional	Query Language: DMTF:CQL The test was killed by the test (provider).
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG37"	Optional	Query Language: DMTF:CQL The test was terminated by the test (provider).
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG38"	Optional	Query Language: DMTF:CQL The test was resumed by the client
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG39"	Conditional	This indication is mandatory if CIM_DiagnosticTest.Characteristics has the value of 3 (Is Interactive). Query Language:DMTF:CQL A JobSetting parameter was reset by the job.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG40"	Conditional	This indication is mandatory if CIM_DiagnosticTest.Characteristics has the value of 3 (Is Interactive). Query Language:DMTF:CQL JobSetting parameter was ignored by the job.

10.1 CIM_AffectedJobElement

Although defined in DSP1103, the CIM_AffectedJobElement class is listed here because the AffectedElement reference is scoped down to CIM_DiagnosticTest. The constraints listed in Table 16 are in addition to those specified in DSP1103. See DSP1103 for other mandatory elements that must be implemented.

1239 **Table 16 – Class: CIM_AffectedJobElement**

Properties	Requirement	Notes
AffectedElement (overridden)	Mandatory	Key. The property shall be a reference to an instance of the CIM_ManagedElement being tested..
AffectingElement	Mandatory	Key. The property shall be a reference to an instance of CIM_ConcreteJob.
ElementEffects	Optional	See 7.1.1.
OtherElementEffectsDescriptions	Optional	If ElementEffects contains the value 1, this property is Mandatory.

1240 ## 10.2 CIM_ConcreteJob

1241 Each successful invocation of the CIM_DiagnosticTest.RunDiagnosticService( ) extrinsic method returns
1242 a CIM_ConcreteJob instance. Each CIM_ConcreteJob instance represents a diagnostic test execution.
1243 This class specializes CIM_ConcreteJob as defined in DSP1103. The constraints listed in Table 17 are in
1244 addition to those specified in DSP1103. See DSP1103 for other mandatory elements that must be
1245 implemented.

1246 **Table 17 – Class: CIM_ConcreteJob**

Properties	Requirement	Notes
InstanceID	Mandatory	Key. See 7.2.1.
Name	Mandatory	See 7.2.2.
JobState	Mandatory	See 7.2.3.
DeleteOnCompletion	Mandatory	See 7.2.4.
TimeBeforeRemoval	Mandatory	See 7.2.5.
StartTime	Mandatory	See 7.2.6.
ElapsedTime	Mandatory	See 7.2.7.
RequestedState	Mandatory	
PercentComplete	Mandatory	See 7.2.8.
ElementName	Mandatory	See 7.2.9.
RequestStateChange( )	Mandatory	See 8.2.1.
ResumeWithInput( )	Conditional	See 8.2.2.
ResumeWithAction( )	Conditional	See 8.2.3.

1247 ## 10.3 CIM_DiagnosticServiceJobCapabilities

1248 The CIM_DiagnosticServiceJobCapabilities is used to provide information about on the capabilities of the
1249 job that is used to run the diagnostic test.

1250 **Table 18 – Class: CIM_DiagnosticServiceJobCapabilities**

Properties	Requirement	Notes
InstanceID	Mandatory	Key. See 7.3.1.
ElementName	Mandatory	See 7.3.2.
DeleteJobSupported	Mandatory	See 7.3.3.
RequestedStatesSupported	Mandatory	See 7.3.4.
InteractiveTimeoutMax	Conditional	See 7.3.5.
DefaultValuesSupported	Conditional	See 7.3.6.
ClientRetriesMax	Conditional	See 7.3.7.
CleanupInterval	Optional	See 7.3.8.
SilentModeSupported	Conditional	See 7.3.9.
CreateGoalSettings( )	Mandatory	See 8.3.1.

1251 ## 10.4 CIM_ElementCapabilities (Job)

1252 CIM_ElementCapabilities represents an association between a test and its capabilities to use a job to run
1253 the test. Table 19 contains the requirements for elements of this class.

1254 **Table 19 – Class: CIM_ElementCapabilities**

Properties	Requirement	Notes
ManagedElement	Mandatory	Key. The property shall be a reference to an instance of CIM_DiagnosticTest.
Capabilities	Mandatory	Key. The property shall be a reference to an instance of CIM_DiagnosticServiceJobCapabilities.

1255 ## 10.5 CIM_ElementSettingData (Default JobSettingData)

1256 CIM_ElementSettingData represents an association between a test and the default setting data for the
1257 job used to run the test. ManagedElement is responsible for the creation of the job. Table 20 contains the
1258 requirements for elements of this class. If CIM_DiagnosticTest (or a subclass) has more than one type of
1259 test, a separate default job setting may be defined for each. However, all instances of the same type of
1260 test shall reference the same default job setting.

1261 **Table 20 – Class: CIM_ElementSettingData**

Properties	Requirement	Notes
ManagedElement	Mandatory	Key. The property shall be a reference to an instance of CIM_DiagnosticTest.
SettingData	Mandatory	Key. The property shall be a reference to an instance of CIM_JobSettingData (Default).
IsDefault	Mandatory	The value of the property shall be TRUE.

10.6 CIM_HostedDependency

CIM_HostedDependency represents an association between the system on which a test is run and the CIM_ConcreteJob that is used to run the test. Table 21 contains the requirements for elements of this class.

Table 21 – Class: CIM_HostedDependency

Properties	Requirement	Notes
Antecedent	Mandatory	Key. The property shall be a reference to an instance of CIM_System.
Dependent	Mandatory	Key. The property shall be a reference to an instance of CIM_ConcreteJob.

10.7 CIM_JobSettingData (Client)

This CIM_JobSettingData definition represents the settings the client uses for the job used to run the diagnostic test. Table 22 contains the requirements for elements of this class.

Table 22 – Class: CIM_JobSettingData (Client)

Properties	Requirement	Notes
InstanceID	Mandatory	Key. See 7.4.1.
DeleteOnCompletion	Conditional	See 7.4.2.
InteractiveTimeout	Conditional	See 7.4.3.
TerminateOnTimeout	Conditional	See 7.4.4.
DefaultInputsValues	Conditional	See 7.4.5.
DefaultInputNames	Conditional	See 7.4.6.
ClientRetries	Conditional	See 7.4.7.
RunInSilentMode	Conditional	See 7.4.8.
ElementName	Mandatory	This property is a free-form string of variable length. (pattern ".*")

10.8 CIM_JobSettingData (Default)

This CIM_JobSettingData definition represents the default settings for the job used to run the diagnostic test. Table 23 contains the requirements for elements of this class. If CIM_DiagnosticTest (or a subclass) has more than one type of test, a separate default job setting may be defined for each.

Table 23 – Class: CIM_JobSettingData (Default)

Properties	Requirement	Notes
InstanceID	Mandatory	Key. See 7.5.1.
DeleteOnCompletion	Conditional	See 7.5.2.
InteractiveTimeout	Conditional	See 7.5.3.
TerminateOnTimeout	Conditional	See 7.5.4.
DefaultInputValues	Conditional	See 7.5.5.

Properties	Requirement	Notes
DefaultInputNames	Conditional	See 7.5.6.
ClientRetries	Conditional	See 7.5.7.
RunInSilentMode	Conditional	See 7.5.8.
ElementName	Mandatory	This property is a free-form string of variable length. (pattern ".*")

10.9 CIM_OwningJobElement

Although defined in DSP1103, the CIM_OwningJobElement class is listed here because the OwningElement reference is scoped down to CIM_DiagnosticTest, which is a subclass of CIM_ManagedElementt. The constraints listed in Table 24 are in addition to those specified in DSP1103. See DSP1103 for other mandatory properties of CIM_HostedService that must be implemented.

Table 24 – Class: CIM_OwningJobElement

Properties	Requirement	Notes
OwningElement	Mandatory	Key. The property shall be a reference to an instance of CIM_DiagnosticTest.
OwnedElement	Mandatory	Key. The property shall be a reference to an instance of CIM_ConcreteJob.

10.10 CIM_RegisteredProfile

The CIM_RegisteredProfile class is defined in DSP1033. The constraints listed in Table 25 are in addition to those specified in DSP1033. See DSP1033 for other mandatory properties of CIM_RegisteredProfile that must be implemented.

Table 25 – Class: CIM_RegisteredProfile

Properties	Requirement	Notes
RegisteredName	Mandatory	The value of the property shall be “Diagnostic Job Control”
RegisteredVersion	Mandatory	The value of the property shall be “1.0.0”
RegisteredOrganization	Mandatory	The value of the property shall be 2 (DMTF)

Annex A
(informative)

Change log

Version	Date	Description
0.1	2010-12-05	Initial Version
1.0.0a	2012-09-12	Work In Progress version