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: 2013-10-31

Target version for DMTF Standard: 2.34.0
DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems
management and interoperability. Members and non-members may reproduce DMTF specifications and
documents, provided that correct attribution is given. As DMTF specifications may be revised from time to
time, the particular version and release date should always be noted.

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withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreword</td>
<td>7</td>
</tr>
<tr>
<td>Introduction</td>
<td>8</td>
</tr>
<tr>
<td>Document conventions</td>
<td>8</td>
</tr>
<tr>
<td>Scope</td>
<td>9</td>
</tr>
<tr>
<td>Normative references</td>
<td>9</td>
</tr>
<tr>
<td>Terms and definitions</td>
<td>10</td>
</tr>
<tr>
<td>Symbols and abbreviated terms</td>
<td>10</td>
</tr>
<tr>
<td>Synopsis</td>
<td>11</td>
</tr>
<tr>
<td>Description</td>
<td>12</td>
</tr>
<tr>
<td>Implementation</td>
<td>14</td>
</tr>
<tr>
<td>7.1 CIM_AffectedJobElement</td>
<td>14</td>
</tr>
<tr>
<td>7.1.1 CIM_AffectedJobElement.ElementEffects</td>
<td>14</td>
</tr>
<tr>
<td>7.2 CIM_ConcreteJob</td>
<td>14</td>
</tr>
<tr>
<td>7.2.1 CIM_ConcreteJob.InstanceID</td>
<td>14</td>
</tr>
<tr>
<td>7.2.2 CIM_ConcreteJob.Name</td>
<td>15</td>
</tr>
<tr>
<td>7.2.3 CIM_ConcreteJob.JobState</td>
<td>15</td>
</tr>
<tr>
<td>7.2.4 CIM_ConcreteJob.DeleteOnCompletion</td>
<td>15</td>
</tr>
<tr>
<td>7.2.5 CIM_ConcreteJob.TimeBeforeRemoval</td>
<td>15</td>
</tr>
<tr>
<td>7.2.6 CIM_ConcreteJob.StartTime</td>
<td>15</td>
</tr>
<tr>
<td>7.2.7 CIM_ConcreteJob.ElapsedTime</td>
<td>16</td>
</tr>
<tr>
<td>7.2.8 CIM_ConcreteJob.PercentComplete</td>
<td>16</td>
</tr>
<tr>
<td>7.2.9 CIM_ConcreteJob.TimeOfLastStateChange</td>
<td>16</td>
</tr>
<tr>
<td>7.3 CIM_DiagnosticServiceJobCapabilities</td>
<td>16</td>
</tr>
<tr>
<td>7.3.1 CIM_DiagnosticServiceJobCapabilities.InstanceID</td>
<td>17</td>
</tr>
<tr>
<td>7.3.2 CIM_DiagnosticServiceJobCapabilities.ElementName</td>
<td>17</td>
</tr>
<tr>
<td>7.3.3 CIM_DiagnosticServiceJobCapabilities.DeleteJobSupported</td>
<td>17</td>
</tr>
<tr>
<td>7.3.4 CIM_DiagnosticServiceJobCapabilities.RequestedStatesSupported</td>
<td>17</td>
</tr>
<tr>
<td>7.3.5 CIM_DiagnosticServiceJobCapabilities.InteractiveTimeoutMax</td>
<td>17</td>
</tr>
<tr>
<td>7.3.6 CIM_DiagnosticsServiceJobCapabilities.DefaultValuesSupported</td>
<td>18</td>
</tr>
<tr>
<td>7.3.7 CIM_DiagnosticsServiceJobCapabilities.ClientRetriesMax</td>
<td>18</td>
</tr>
<tr>
<td>7.3.8 CIM_DiagnosticsServiceJobCapabilities.CleanupInterval</td>
<td>18</td>
</tr>
<tr>
<td>7.3.9 CIM_DiagnosticsServiceJobCapabilities.SilentModeSupported</td>
<td>18</td>
</tr>
<tr>
<td>7.4 CIM_JobSettingData (Client)</td>
<td>18</td>
</tr>
<tr>
<td>7.4.1 CIM_JobSettingData.InstanceID</td>
<td>19</td>
</tr>
<tr>
<td>7.4.2 CIM_JobSettingData.DeleteOnCompletion</td>
<td>19</td>
</tr>
<tr>
<td>7.4.3 CIM_JobSettingData.InteractiveTimeout</td>
<td>19</td>
</tr>
<tr>
<td>7.4.4 CIM_JobSettingData.TerminateOnTimeout</td>
<td>19</td>
</tr>
<tr>
<td>7.4.5 CIM_JobSettingData.DefaultInputValues</td>
<td>20</td>
</tr>
<tr>
<td>7.4.6 CIM_JobSettingData.DefaultInputNames</td>
<td>20</td>
</tr>
<tr>
<td>7.4.7 CIM_JobSettingData.ClientRetries</td>
<td>20</td>
</tr>
<tr>
<td>7.4.8 CIM_JobSettingData.RunInSilentMode</td>
<td>20</td>
</tr>
<tr>
<td>7.5 CIM_JobSettingData (Default)</td>
<td>20</td>
</tr>
<tr>
<td>7.5.1 CIM_JobSettingData.InstanceID</td>
<td>21</td>
</tr>
<tr>
<td>7.5.2 CIM_JobSettingData.DeleteOnCompletion</td>
<td>21</td>
</tr>
<tr>
<td>7.5.3 CIM_JobSettingData.InteractiveTimeout</td>
<td>21</td>
</tr>
<tr>
<td>7.5.4 CIM_JobSettingData.TerminateOnTimeout</td>
<td>22</td>
</tr>
<tr>
<td>7.5.5 CIM_JobSettingData.DefaultInputValues</td>
<td>22</td>
</tr>
<tr>
<td>7.5.6 CIM_JobSettingData.DefaultInputNames</td>
<td>22</td>
</tr>
<tr>
<td>7.5.7 CIM_JobSettingData.ClientRetries</td>
<td>22</td>
</tr>
<tr>
<td>7.5.8 CIM_JobSettingData.RunInSilentMode</td>
<td>22</td>
</tr>
<tr>
<td>7.6 Interactive options</td>
<td>23</td>
</tr>
</tbody>
</table>
Diagnostic Job Control Profile

7.7 Job deletion options .................................................. 25
7.8 Diagnostic Job Control Profile indications support .................. 28
7.8.1 CIM_IndicationFilter (StaticIndicationFilter) .................... 28
7.8.2 CIM_FilterCollection (ProfileSpecificFilterCollection) ............ 28
7.8.3 CIM_MemberOfCollection (IndicationFilterInFilterCollection) ...... 28
7.8.4 CIM_OwningCollectionElement (IndicationServiceOfFilterCollection) . 29
7.9 Diagnostics job control alert indications and standard messages ...... 29
7.9.1 DIAG9 – Test continued after last interactive timeout using default values ........ 29
7.9.2 DIAG12 – Job could not be started ..................................... 29
7.9.3 DIAG19 – Test killed by client ......................................... 30
7.9.4 DIAG20 – Test terminated by client .................................... 30
7.9.5 DIAG21 – Test suspended by client ................................... 31
7.9.6 DIAG34 – Request for inputs ........................................... 32
7.9.7 DIAG35 – Request for action ........................................... 32
7.9.8 DIAG36 – Test killed by test ............................................. 33
7.9.9 DIAG37 – Test terminated by test ....................................... 33
7.9.10 DIAG38 – Test resumed by client ...................................... 34
7.9.11 DIAG39 – JobSettings reset ............................................. 34
7.9.12 DIAG40 – JobSettings defaults not used ............................. 35
7.9.13 DIAG48 – Test continued after an interim interactive timeout .......... 36
7.9.14 DIAG49 – Test terminated after an interactive timeout ............. 36
8 Methods ........................................................................... 37
8.1 Profile conventions for operations ....................................... 37
8.2 CIM_ConcreteJob ............................................................ 37
8.2.1 CIM_ConcreteJob.RequestStateChange( ) .......................... 37
8.2.2 CIM_ConcreteJob.ResumeWithInput( ) ............................. 38
8.2.3 CIM_ConcreteJob.ResumeWithAction( ) ................................ 39
8.3 CIM_DiagnosticServiceJobCapabilities ................................ 40
8.3.1 CreateGoalSettings( ) .................................................... 40
8.4 CIM_MethodResult ........................................................ 41
8.5 CIM_OwningJobElement .................................................. 41
8.6 CIM_AffectedJobElement .................................................. 41
8.7 CIM_AssociatedJobMethodResult ....................................... 41
8.8 CIM_HostedDependency ................................................... 41
8.9 CIM_RestrictedProfile ...................................................... 41
8.10 CIM_JobSettingData ...................................................... 41
8.11 CIM_ElementSettingData .................................................. 42
8.12 CIM_ElementCapabilities ................................................ 42
8.13 CIM_DiagnosticTest.RunDiagnosticService( ) ....................... 42
9 Use cases (informative) ..................................................... 43
9.1 Use case summary ....................................................... 43
9.2 User input required ........................................................ 45
9.2.1 Single prompt and response has a valid value .................... 45
9.2.2 Single prompt and response has multiple valid values ........... 46
9.2.3 Multiple prompts and responses required with partial test execution after each 46
9.2.4 Client does not respond to a prompt ................................... 47
9.2.5 Client responds with an invalid value ................................. 47
9.2.6 Client does not respond with enough valid values ............... 47
9.3 User action required ........................................................ 48
9.3.1 Single prompt and response required ................................. 48
9.3.2 Multiple prompts and responses required before running the test ........ 48
9.3.3 Multiple prompts and responses required with partial test execution after each 49
9.3.4 Client does not respond to a prompt ................................... 49
9.4 Silent mode operation ..................................................... 50
9.4.1 Running an Interactive Test in Silent Mode .......................... 50
9.4.2 Running Silent Mode with invalid default values .................. 50

Work in Progress — Not a DMTF Standard
Version 1.0.0b
9.5 Finding diagnostic jobs ................................................................. 51
9.5.1 Finding all diagnostic tests executed on a system ................................. 51
9.5.2 Finding all diagnostic tests executed against a ManagedElement .......... 51
9.6 Configuring a diagnostic job ......................................................... 52
9.6.1 Getting the default job settings ................................................... 52
9.6.2 Creating the job settings ............................................................. 52
9.7 Execute and control a job for a diagnostic test ..................................... 52
9.7.1 Suspend a job for a diagnostic test .............................................. 53
9.7.2 Resume a job for a diagnostic test ............................................... 53
9.7.3 Terminate a job for a diagnostic test ............................................. 54
9.7.4 Kill a job for a diagnostic test .................................................... 54
9.8 Delete a job for a diagnostic test .................................................... 55
9.8.1 Client deletes a job for a diagnostic test ....................................... 55
9.8.2 Provider deletes a job ................................................................. 55
10 CIM elements .................................................................................. 56
10.1 CIM_AffectedJobElement ............................................................... 58
10.2 CIM_ConcreteJob ........................................................................... 58
10.3 CIM_DiagnosticServiceJobCapabilities ............................................ 59
10.4 CIM_ElementCapabilities (Job) ..................................................... 59
10.5 CIM_ElementSettingData (Default JobSettingData) ............................... 59
10.6 CIM_FilterCollection (ProfileSpecificFilterCollection) ......................... 60
10.7 CIM_HostedDependency ................................................................. 60
10.8 CIM_IndicationFilter (StaticIndicationFilter) ...................................... 60
10.9 CIM_JobSettingData (Client) ........................................................... 61
10.10 CIM_JobSettingData (Default) ......................................................... 61
10.11 CIM_MemberOfCollection (ProfileSpecificMemberOfCollection) .......... 62
10.12 CIM_OwningCollectionElement .................................................... 62
10.13 CIM_OwningJobElement ............................................................... 62
10.14 CIM_RegisteredProfile ................................................................ 63
ANNEX A (informative) Change log .......................................................... 64
Diagnostic Job Control Profile

Figures

Figure 1 – Diagnostic Job Control Profile: Profile class diagram ........................................ 13

Tables

Table 1 – Referenced profiles ............................................................................................... 12
Table 2 – OperationalStatus to JobState mapping ............................................................... 15
Table 3 – Interactive options ............................................................................................... 24
Table 4 – Job deletion options ............................................................................................. 27
Table 5 – ResumeWithInput( ) method: Return code values ............................................... 39
Table 6 – ResumeWithInput( ) method: Parameters .......................................................... 39
Table 7 – ResumeWithAction( ) method: Return code values ............................................ 40
Table 8 – CreateGoalSettings( ) method: Return code values ......................................... 40
Table 9 – CreateGoalSettings( ) method: Parameters ......................................................... 41
Table 10 – Operations: CIM_JobSettingData ..................................................................... 42
Table 11 – Operations: CIM_ElementSettingData .............................................................. 42
Table 12 – Operations: CIM_ElementCapabilities ............................................................. 42
Table 13 – Job settings options ................................................................................................
Table 14 – Diagnostic test use cases ....................................................................................
Table 15 – CIM Elements: Diagnostic Job Control Profile ..................................................
Table 16 – Class: CIM_AffectedJobElement ........................................................................
Table 17 – Class: CIM_ConcreteJob .....................................................................................
Table 18 – Class: CIM_DiagnosticServiceJobCapabilities ..................................................
Table 19 – Class: CIM_ElementCapabilities ....................................................................... 59
Table 20 – Class: CIM_ElementSettingData ....................................................................... 60
Table 21 – Class: CIM_FilterCollection .............................................................................. 60
Table 22 – Class: CIM_HostedDependency ......................................................................... 60
Table 23 – Class: CIM_IndicationFilter (StaticIndicationFilter) ........................................ 61
Table 24 – Class: CIM_JobSettingData (Client) ................................................................... 61
Table 25 – Class: CIM_JobSettingData (Default) ............................................................... 61
Table 26 – Class: CIM_MemberOfCollection ...................................................................... 62
Table 27 – Class: CIM_OwningCollectionElement ............................................................. 62
Table 28 – Class: CIM_OwningJobElement ........................................................................ 63
Table 29 – Class: CIM_RegisteredProfile ........................................................................... 63

6 Work in Progress — Not a DMTF Standard Version 1.0.0b
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.
1 Scope

The Diagnostic Job Control Profile is a specialization of the Job Control Profile (DSP1103) that extends the profile by defining the job control functions used to monitor and interact with diagnostic tests.

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

2 Normative references

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

DMTF DSP0004, CIM Infrastructure Specification 2.6,

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

DMTF DSP1001, Management Profile Specification Usage Guide 1.0,

DMTF DSP1002, Diagnostics Profile Specification 2.0,

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

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

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

DMTF DSP1104, Fibre Channel Host Bus Adapter Diagnostics Profile 1.0
http://dmtf.org/sites/default/files/standards/documents/DSP1104_1.0.0.pdf

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

IETF RFC5234, ABNF: Augmented BNF for Syntax Specifications, January 2008,

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

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

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

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

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

The terms defined in DSP0004, DSP0200, and DSP1001 apply to this document.

3.1 job

a task or thread or execution under which an operation will be run.

3.2 organization

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

3.3 test

a test is task that performs parameter and environment checking before launching a job to execute diagnostic operations against a target element.

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: 1.0.0b
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.
### 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 client application 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 it is controlled by the values of the properties in CIM_ConcreteJob and CIM_JobSettingData.

To start a diagnostic test, the client application calls CIM_DiagnosticTest.RunDiagnosticService() which returns 0 (Success) and the object path of a single CIM_ConcreteJob instance. The CIM_ConcreteJob manages the diagnostic test execution. Additionally, a client can monitor and interact 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.

After the diagnostic test is completed, 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 diagnostic component profile. For example, if the Referencing Profile is the Fibre Channel Host Bus Adapter Diagnostics Profile (DSP1104), the DiagnosticTest is a subclass called FCHBADiagnosticTest. The Referencing Profile for the ManagedElement is a profile that contains the ManagedElement and that profile should reference the profile with the DiagnosticTest. For the DSP1104, the ManagedElement is CIM_PortController in a system profile. That system profile would reference the Fibre Channel Host Bus Adapter Diagnostic Profile, which in turn would reference the Diagnostic Job Control Profile. The System in Figure 1 would be a system in a profile that references the ManagedElement profile.
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, which associates a CIM_ConcreteJob instance to the CIM_ManagedElement instances that are affected by the job.

Minimally, this shall contain the ManagedElement under test. However, it might also include other managed elements that could be affected. For example, an FC HBA self-test would affect the FC HBA, but might also affect the FC ports on the HBA. In addition, it might impact the system in which the FC HBA is a component.

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 separate test instances over an extended period of time. This uniqueness of the test execution identifier could include multiple executions of the same test or multiple executions of multiple tests. In practice, <LocalID> could be an incremented counter or a timestamp in combination with other test names and job identifiers or monikers.

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. In DSP1002, see Figure 5 and use case 9.8.4 GetDiagnosticExecutionFinalResults 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

<table>
<thead>
<tr>
<th>Operational Status</th>
<th>JobState</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (OK)</td>
<td>4 (Running)</td>
<td>Client has responded to the prompt</td>
</tr>
<tr>
<td>10 (Stopped)</td>
<td>12 (Query Pending)</td>
<td>Waiting for the client to respond to the prompt</td>
</tr>
</tbody>
</table>

### 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 is completed after a configurable time period. See subclause 7.2.5 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 is completed, 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 use a value that balances the time required to collect the results against the load on the system. For example, in lightly loaded environments it should be possible to choose a value close to test completion time whereas in heavily loaded environments it would be safer to configure a value a multiple of that time period.

### 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. It is recommended that it be synchronized with the updating of PercentComplete.

7.2.8 CIM_ConcreteJob.PercentComplete

In addition to the requirements specified in DSP1103 and CIM_ConcreteJob.ElapsedTime, 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. It is recommended that it be synchronized with the updating of ElapsedTime. If progress cannot be determined with that reasonable amount of time it should be set to 50 percent. 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.TimeOfLastStateChange

The date and time when the state of the Job last changed (via RequestedStateChange). If the state of the Job has not changed and this property is populated, then it shall be set to a 0 interval value. If a state change was requested, but rejected or not yet processed, the property shall not be updated. If a state change was requested, accepted and processed successfully, then the value shall be the date and time of the successful completion of the state change.

---

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_DiagnosticServiceJobCapabilities 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 JobSettings 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. The default CIM_JobSettingData.DeleteOnCompletion property shall have the value TRUE. In addition, a client shall not set the value of the DeleteOnCompletion property in the CIM_JobSettingData instance that it passes as the JobSettings 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. The default 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 JobSettings 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.RequestedStateChange( ) 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 JobSettings parameter to the CIM_DiagnosticTest.RunDiagnosticService() extrinsic method, and the value of CIM_JobSettingData.InteractiveTimeout exceeds the value of CIM_DiagnosticsServiceJobCapabilities.InteractiveTimeoutMax, the associated CIM_ConcreteJob instance shall use the value of CIM_DiagnosticsServiceJobCapabilities.InteractiveTimeoutMax.

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 have the value 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 job, after completion, will remain until it is explicitly deleted by the client. The CleanupInterval datetime property insures that the job will be deleted should the client fail to do so. It defines the time interval before the job is removed.

7.3.9 CIM_DiagnosticsServiceJobCapabilities.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_JobSettingData. 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 JobSettings 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_DiagnosticsServiceJobCapabilities is implemented, a client can create a CIM_JobSettingData (Client) instance containing the default values specified in the CIM_DiagnosticsServiceJobCapabilities...
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 have the value 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_DiagnosticServiceJobCapabilities.SilentModeSupported has the value of FALSE, this property is
ignored.

EXPERIMENTAL

7.5 CIM_JobSettingData (Default)

This subclause specifies 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, a single instance
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 as the class CIM_JobSettingData (Client) 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 (Client) 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:

\(<\text{OrgID}>:<\text{LocalID}>\)

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

The purpose for \(<\text{LocalID}>\) is to provide some form of uniqueness within the context of different JobSettingData instances within the system. In practice, \(<\text{LocalID}>\) 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 a DeleteInstance operation.

NOTE When the value of the CIM_DiagnosticServiceJobCapabilities.DeleteJobSupported property is FALSE, the value of CIM_JobSettingData.DeleteOnCompletion shall have the value TRUE and the jobs created will always be removed after the TimeBeforeRemoval time interval.

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.

NOTE If JobSettings parameter of the RunDiagnosticService is NULL (not supplied), then the default JobSettingData value for InteractiveTimeout shall be used.

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
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 DefaultInputValues as the response and resume the test.

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

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 the specified CIM_JobSettingData.InteractiveTimeout period of time. A value of one for this property indicates that the diagnostic test will issue a second prompt for the same response.

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 values defined in default
CIM_JobSettingData.DefaultInputValues or the JobSettings parameter of the RunDiagnosticService.

When the value is FALSE, the interactive diagnostic test will prompt the client for a response.

If CIM_DiagnosticServiceJobCapabilities.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 JobSettings input parameter, which is an embedded instance of CIM_JobSettingData(Client), 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 JobSettings 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, CIM_JobSettingData.DefaultInputValues, 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 JobSettings 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
## Diagnostic Job Control Profile

### Table 3 – Interactive options

<table>
<thead>
<tr>
<th>DiagnosticServiceJob Capabilities</th>
<th>JobSettingData (Default)</th>
<th>JobSettings [JobSettingData (Client)]</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>InteractiveTimeoutMax</td>
<td>InteractiveTimeout &lt;= InteractiveTimeoutMax</td>
<td>InteractiveTimeout &lt;= InteractiveTimeoutMax</td>
<td>Use JobSettings value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>InteractiveTimeout &gt;= InteractiveTimeoutMax</td>
<td>Use DiagnosticServiceJob Capabilities value (2)</td>
</tr>
<tr>
<td>DefaultValuesSupported = TRUE</td>
<td>DefaultInputValues = non-NULL DefaultInputNames = non-NULL</td>
<td>DefaultInputValues = non-NULL DefaultInputNames = non-NULL</td>
<td>Use JobSettings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DefaultInputValues = NULL DefaultInputNames = NULL</td>
<td>Use JobSettingData (Default). (2)</td>
</tr>
<tr>
<td>DefaultValuesSupported = FALSE</td>
<td>DefaultInputValues = NULL DefaultInputNames = NULL</td>
<td>DefaultInputValues = NULL DefaultInputNames = NULL</td>
<td>JobSettings is ignored. (3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DefaultInputValues = non-NULL DefaultInputNames = non-NULL</td>
<td>The client shall supply input values.</td>
</tr>
<tr>
<td>DefaultValuesSupported = TRUE</td>
<td>DefaultInputValues = non-NULL DefaultInputNames = non-NULL</td>
<td>DefaultInputValues = non-NULL DefaultInputNames = non-NULL RunInSilentMode = TRUE</td>
<td>Run in silent mode using JobSettings default values.</td>
</tr>
<tr>
<td>SilentModeSupported = TRUE</td>
<td>DefaultInputValues = non-NULL DefaultInputNames = non-NULL</td>
<td>DefaultInputValues = NULL DefaultInputNames = NULL RunInSilentMode = TRUE</td>
<td>Run in silent mode using JobSettingData default values.</td>
</tr>
<tr>
<td>ClientRetriesMax = N</td>
<td>TerminateOnTimeout = TRUE</td>
<td>TerminateOnTimeout = TRUE</td>
<td>The Job will terminate after N tries to solicit input.</td>
</tr>
<tr>
<td>DefaultValuesSupported = FALSE</td>
<td></td>
<td>TerminateOnTimeout = FALSE</td>
<td>Use JobSettingData. TerminateOnTimeout (2)</td>
</tr>
<tr>
<td>ClientRetriesMax = N</td>
<td>ClientRetries = M where M &lt;= N</td>
<td>ClientRetries = R where R &lt;= N</td>
<td>After R retries, the job terminates or runs with defaults. (1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ClientRetries = NULL</td>
<td>After M retries, the job terminates or runs with defaults. (1) (2)</td>
</tr>
<tr>
<td>ClientRetriesMax = N</td>
<td>ClientRetries = M where M &lt;= N</td>
<td>ClientRetries = R where R &gt; N</td>
<td>After M retries, the job terminates or runs with defaults. (1) (2)</td>
</tr>
</tbody>
</table>
### Diagnostic Job Control Profile

<table>
<thead>
<tr>
<th>DiagnosticServiceJob Capabilities</th>
<th>JobSettingData (Default)</th>
<th>JobSettings [JobSettingData (Client)]</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>InteractiveTimeoutMax</td>
<td>InteractiveTimeout &gt; InteractiveTimeoutMax</td>
<td>Undefined</td>
<td></td>
</tr>
<tr>
<td>DefaultValuesSupported = TRUE</td>
<td>DefaultInputValues = NULL, DefaultInputNames = NULL</td>
<td>Undefined</td>
<td></td>
</tr>
<tr>
<td>DefaultValuesSupported = FALSE</td>
<td>DefaultInputValues = non-NULL, DefaultInputNames = non-NULL</td>
<td>Undefined</td>
<td></td>
</tr>
<tr>
<td>DefaultValuesSupported = TRUE, SilentModeSupported = TRUE</td>
<td>DefaultInputValues = NULL, DefaultInputNames = NULL, RunInSilentMode = TRUE</td>
<td>Undefined</td>
<td></td>
</tr>
<tr>
<td>DefaultValuesSupported = FALSE, SilentModeSupported = TRUE</td>
<td>TerminateOnTimeout = FALSE</td>
<td>Undefined</td>
<td></td>
</tr>
<tr>
<td>ClientRetriesMax = N</td>
<td>ClientRetries = M where M &gt; N</td>
<td>Undefined</td>
<td></td>
</tr>
</tbody>
</table>

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 JobSettings, which is an embedded instance of CIM_JobSettingData(Client), was reset. The overridden effective JobSettings should be logged.
3. An alert indication (DIAG40) is sent indicating that the default values were not used. The overridden effective JobSettings 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. After a diagnostic test is completed, 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 properties in the mandatory CIM_JobSettingData (Default) class, and the optional JobSettings input parameter, which is an embedded CIM_JobSettingData instance, used when the client invokes the CIM_DiagnosticTest.RunDiagnosticService( ) extrinsic method. Table 4 shows the behavior when these following properties have certain and possibly conflicting values.

- CIM_DiagnosticServiceJobCapabilities.DeleteJobSupported
- CIM_DiagnosticServiceJobCapabilities.CleanupInterval
- CIM_JobSettingData.DeleteOnCompletion
- CIM_JobSettingData.TimeBeforeRemoval

Since the JobSettings input parameter, which is an embedded instance of CIM_JobSettingData (Client), is optional, its value may by NULL. In this case, the CIM_DiagnosticTest.RunDiagnosticService extrinsic method shall use the values of the mandatory CIM_JobSettingData (Default) instance.
An instance of CIM_DiagnosticServiceJobCapabilities may not exist because its implementation is optional. In this case, the other class properties present will dictate behavior.
### Table 4 – Job deletion options

<table>
<thead>
<tr>
<th>DiagnosticServiceJob Capabilities</th>
<th>JobSettingData (Default)</th>
<th>JobSettings [JobSettingData (Client)]</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeleteJobSupported = TRUE AND CleanupInterval = non-NULL</td>
<td>DeleteOnCompletion = TRUE or FALSE</td>
<td>DeleteOnCompletion = TRUE</td>
<td>The provider deletes the job instance after the job is completed and an elapsed time of TimeBeforeRemoval</td>
</tr>
<tr>
<td>DeleteJobSupported = TRUE AND CleanupInterval = NULL</td>
<td>DeleteOnCompletion = TRUE</td>
<td>DeleteOnCompletion = TRUE</td>
<td>The provider deletes the job instance after the job is completed and an elapsed time of TimeBeforeRemoval</td>
</tr>
<tr>
<td>DeleteJobSupported = FALSE AND CleanupInterval = non-NULL</td>
<td>DeleteOnCompletion = TRUE</td>
<td>DeleteOnCompletion = TRUE</td>
<td>The provider deletes the job instance after the job is completed and an elapsed time of TimeBeforeRemoval</td>
</tr>
<tr>
<td>DeleteJobSupported = FALSE AND CleanupInterval = NULL</td>
<td>DeleteOnCompletion = TRUE</td>
<td>DeleteOnCompletion = TRUE</td>
<td>The provider deletes the job instance after the job is completed and an elapsed time of TimeBeforeRemoval (1)</td>
</tr>
<tr>
<td>DeleteJobSupported = FALSE AND CleanupInterval = non-NULL</td>
<td>DeleteOnCompletion = FALSE</td>
<td>DeleteOnCompletion = FALSE</td>
<td>The client should delete the job instance after the job is completed. The provider will not automatically delete the job. The CleanupInterval is ignored.</td>
</tr>
</tbody>
</table>

---

(1) An alert indication (DIAG39) is sent indicating that the JobSettings, which is an embedded instance of CIM_JobSettingData (Client), was reset. The overridden effective JobSettings should be logged.
7.8 Diagnostic Job Control Profile indications support

The Diagnostic Job Control Profile constrains certain elements in its support for the DMTF Indications Profile. This subclause identifies those constraints.

7.8.1 CIM_IndicationFilter (StaticIndicationFilter)

The Diagnostic Job Control Profile constrains some of the properties of the StaticIndicationFilter version of the CIM_IndicationFilter class and makes the class mandatory. The class is mandatory because some of the alert indication filters are mandatory and the Diagnostic Job Control profile requires that static versions of mandatory indication filters be populated.

7.8.1.1 CIM_IndicationFilter.Name

The Diagnostic Job Control Profile constrains names of the profile defined alert indication filters as prescribed by DSP1054. The names for the indication filters are identified in the entries for the indications in Table 15. The Name shall be formatted as defined by the following ABNF rule:

```
"DMTF:Diagnostic Job Control:" MessageID
```

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

7.8.1.2 CIM_IndicationFilter.Query

The Diagnostic Job Control Profile constrains the Query properties of the profile defined alert indication filters as prescribed by DSP1054. The Query properties for the indication filters are identified in the entries for the indications in Table 15.

7.8.1.3 CIM_IndicationFilter.QueryLanguage

The Diagnostic Job Control Profile constrains the QueryLanguage properties of the profile defined alert indication filters as prescribed by DSP1054. The QueryLanguage properties for the indication filters are identified in the entries for the indications in Table 15.

7.8.2 CIM_FilterCollection (ProfileSpecificFilterCollection)

The Diagnostic Job Control Profile constrains the CollectionName property of the ProfileSpecificFilterCollection filter collection.

7.8.2.1 CIM_FilterCollection.CollectionName

The Diagnostic Job Control Profile constrains CollectionName of the profile defined ProfileSpecificFilterCollection filter collection as prescribed by DSP1054. The CollectionName for the filter collection shall be formatted as defined by the following ABNF rule:

```
"DMTF:Diagnostic Job Control:ProfileSpecifiedAlertIndicationFilterCollection"
```

7.8.3 CIM_MemberOfCollection (IndicationFilterInFilterCollection)

7.8.3.1 CIM_MemberOfCollection.Collection

The Diagnostic Job Control Profile constrains the Collection property to be the reference to the ProfileSpecificFilterCollection filter collection.

7.8.3.2 CIM_MemberOfCollection.Member

The Diagnostic Job Control Profile constrains the Member property to be a reference to one of the profile defined alert indication filters.
7.8.4  CIM_OwningCollectionElement (IndicationServiceOfFilterCollection)

7.8.4.1  CIM_OwningCollectionElement.OwnedElement

The Diagnostic Job Control Profile constrains OwnedElement property to be the reference to the ProfileSpecifiedFilterCollection filter collection.

7.9  Diagnostics job control alert indications and standard messages

7.9.1  DIAG9 – Test continued after last interactive timeout using default values

The interactive test experienced a timeout on its last query to the user and was resumed using default values.

This alert would only be sent if the test job was interactive (CIM_DiagnosticTest.Characteristics includes "3" (Is Interactive). The alert is sent when the JobSettings.TerminateOnTimeout = FALSE and the JobSettings.InteractiveTimeout and JobSettings.ClientRetries are exceeded before the client provides a response to an alert for either input or action (see DIAG34 in subclause 7.9.6 and DIAG35 in subclause 7.9.7). When the JobSettings.TerminateOnTimeout = FALSE and the JobSettings.ClientRetries have been exceeded, the test resumes with Default Values.

The variables in this message are:

- Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name property of the DiagnosticTest instance.
- Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive) that was specified.

This could be one of the following:
- The Object path of the element
- The ElementName of the element
- A unique user friendly name not in the model (such as, asset name)

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

With this alert the AlertType shall have the value 1 (Other). The OtherAlertType should be set to “Default Values Used”.

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

7.9.2  DIAG12 – Job could not be started

The test job could not be started. The test was not run.

This alert would be sent if conditions are such that the test could not be executed. This message shall be sent if the ReturnCode on RunDiagnosticService is non-zero. Some of the reasons that this might be true are:

- Element already under test. Too many jobs are currently running
- The test is disabled
- The element is disabled
- The element under test is in recovery
- Resources are inadequate to run job

The variables in this message are:
Diagnostic Job Control Profile

- Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name property of the DiagnosticTest instance.

- Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive) that was specified.

  This could be one of the following:
  - The Object path of the element
  - The ElementName of the element
  - A unique user friendly name not in the model (such as, asset name)

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

- Reason – Identifies the reason the job was not started.

  With this alert the AlertType shall have the value 4 (Processing Error)

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

7.9.3 DIAG19 – Test killed by client

The test was killed by the client using the RequestedStateChange method.

This alert would be sent if the client issued a RequestedStateChange method on the ConcreteJob for the test and the change was successfully executed.

NOTE If the RequestedStateChange failed for any reason, then this is indicated to the client that issued the request with a ReturnCode on the RequestedStateChange. No alert message is sent. The client should retry or take the appropriate remedial action on the test.

The variables in this message are:

- Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name property of the DiagnosticTest instance.

- Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive) that was specified.

  This could be one of the following:
  - The Object path of the element
  - The ElementName of the element
  - A unique user friendly name not in the model (such as, asset name)

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

  With this alert the AlertType shall have the value 1 (Other). OtherAlertType should be set to “Killed by Client”.

  With this alert the PerceivedSeverity shall have the value 2 (Information).

7.9.4 DIAG20 – Test terminated by client

The test was terminated by the client using the RequestedStateChange method.

This alert would be sent if the client issued a RequestedStateChange method on the ConcreteJob for the test and the change was successfully executed.
NOTE If the RequestedStateChange failed for any reason, then this is indicated to the client that issued the request with a ReturnCode on the RequestedStateChange. No alert message is sent. The client should retry or take the appropriate remedial action on the test.

The variables in this message are:

- Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name property of the DiagnosticTest instance.
- Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive) that was specified.

This could be one of the following:

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

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

With this alert the AlertType shall have the value 1 (Other). OtherAlertType should be set to “Terminated by Client”.

With this alert the PerceivedSeverity shall have the value 2 (Information).

7.9.5 DIAG21 – Test suspended by client

The test was suspended by a client that issued a RequestedStateChange setting the new state to suspended.

This alert would be sent if a client issues a RequestedStateChange on the ConcreteJob specifying the new state as Suspended, the implementation supports the state change and the implementation successfully executes the request.

NOTE If the RequestedStateChange failed for any reason, then this is indicated to the client that issued the request with a ReturnCode on the RequestedStateChange. No alert message is sent. The client should retry or take the appropriate remedial action on the test.

The variables in this message are:

- Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name property of the DiagnosticTest instance.
- Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive) that was specified.

This could be one of the following:

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

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

With this alert the AlertType shall have the value 1 (Other). The OtherAlertType should be set to “Suspended by Client”.

With this alert the PerceivedSeverity shall have the value 2 (Information).
7.9.6 DIAG34 – Request for inputs

This is an alert indication to solicit input to an interactive test from a client.

This alert would be sent if the test is interactive and requires input values to continue running when SilentMode is set to false.

The variables in this message are:

- Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name property of the DiagnosticTest instance.
- Element Moniker – Identifies a unique name for the element under test (e.g., Disk Drive) that was specified.
  
  This could be one of the following:
  - The Object path of the element
  - The ElementName of the element
  - A unique user friendly name not in the model (such as, asset name)

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

- InputNames – Identifies the names of values that the test is requesting.

With this alert the AlertType shall have the value 1 (Other). OtherAlertType should be set to “Request for Input”.

With this alert the PerceivedSeverity shall have the value 2 (Information).

7.9.7 DIAG35 – Request for action

This is an alert indication to solicit user action from a client on an interactive test.

This alert would be sent if the test is interactive and requires user actions to continue running when SilentMode is set to false.

The variables in this message are:

- Action String - Identifies the action being requested
- Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name property of the DiagnosticTest instance.
- Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive) that was specified.

  This could be one of the following:
  - The Object path of the element
  - The ElementName of the element
  - A unique user friendly name not in the model (such as, asset name)

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

With this alert the AlertType shall have the value 1 (Other). OtherAlertType should be set to “Request for Action”.

With this alert the PerceivedSeverity shall have the value 2 (Information).
With this alert the PerceivedSeverity shall have the value 2 (Information).

### 7.9.8 DIAG36 – Test killed by test

The test killed itself. The test was killed and limited or no clean up was done.

This alert would be sent if the test could not complete due to some error. There may be other (earlier) alerts that specify the specific error. Note this could happen as a result of subtests needed in executing the test. This alert may also be sent if the test that was killed was a subtest of a parent test.

**NOTE** With this alert message, the client may need to take action to effect clean up in order to rerun the test.

The variables in this message are:

- Diagnostic Test Name – Identifies the Diagnostic Test instance (possibly a subtest) that was killed. This is the Name property of the DiagnosticTest instance.
- Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive) that was killed.

This could be one of the following:

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

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

With this alert the AlertType shall have the value 4 (Processing Error). It would be a Processing Error if there was a test software error or a fault with the device that caused the test to be killed.

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

### 7.9.9 DIAG37 – Test terminated by test

The test terminated itself. The test was terminated and clean up was done.

This alert would be sent if the test could not complete and some error occurred. There may be other (earlier) alerts that specify the specific error. Note this could happen as a result of subtests needed to executing the test. This alert may also be sent if the test that was terminated was a subtest of a parent test.

**NOTE** With this alert message, the client need not take action to effect clean up in order to rerun the test.

The variables in this message are:

- Diagnostic Test Name – Identifies the Diagnostic Test instance that was terminated. This is the Name property of the DiagnosticTest instance.
- Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive) that was specified.

This could be one of the following:

- The Object path of the element
- The ElementName of the element
- A unique user friendly name not in the model (such as, asset name)
The Element Moniker can be any of these, but whichever one is used shall be used consistently for all managed elements of the same type within the scoping profile (such as, all disk drives in a system).

With this alert the AlertType shall have the value 4 (Processing Error). It would be a Processing Error if there was a test software error or a fault with the device that caused the test to be terminated, since it does not imply there is anything wrong with the device.

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

7.9.10 DIAG38 – Test resumed by client

The suspended test was resumed by a client that issued a RequestedStateChange setting the new state to start.

This alert would be sent if a client issues a RequestedStateChange on the ConcreteJob specifying the new state as Start when the current job state is Suspended, the implementation supports the state change, and the implementation successfully executes the request.

NOTE If the RequestedStateChange failed for any reason, then this is indicated to the client that issued the request with a ReturnCode on the RequestedStateChange. No alert message is sent. The client should retry or take the appropriate remedial action on the test.

The variables in this message are:

- Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name property of the DiagnosticTest instance.
- Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive) that was specified.

This could be one of the following:

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

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

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

With this alert the PerceivedSeverity shall have the value 2 (Information).

7.9.11 DIAG39 – JobSettings reset

The test was run with the specified JobSettings parameter on RunDiagnosticService reset to match what the test is capable of supporting.

This alert would be sent if a JobSettings supplied on the invocation of RunDiagnosticService cannot be supported and an element of the supplied JobSettingData is replaced. The Alert message identifies the JobSettings property that was reset and the value it was reset to. If multiple JobSettings properties are reset then there would be multiple instances of this alert (each identifying one property that was reset).

The variables in this message are:

- Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name property of the DiagnosticTest instance.
• Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive) that was specified.

This could be one of the following:
– The Object path of the element
– The ElementName of the element
– A unique user friendly name not in the model (such as, asset name)

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

• JobSettings Property – Identifies the JobSettings property that was reset.
• JobSettings Value – Identifies the JobSettings property value that was used

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

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

For more information on this alert see subclause 7.6 and note 2 for entries in the table.

7.9.12 DIAG40 – JobSettings defaults not used

The test ran, but the default values for interactive input as specified in the JobSettings parameter were not used.

This alert would be sent if the client provides default values in JobSettings on invocation of the RunDiagnosticService method but the implementation does not support default values as input (as defined in the DiagnosticServiceCapabilities.DefaultValuesSupported property).

The variables in this message are:

• Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name property of the DiagnosticTest instance.

• Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive) that was specified.

This could be one of the following:
– The Object path of the element
– The ElementName of the element
– A unique user friendly name not in the model (such as, asset name)

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

With this alert the AlertType shall have the value 1 (Other). The OtherAlertType should be set to “Defaults Not Used”.

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

For more information on this alert see subclause 7.6 and note 3 for entries in the table.
7.9.13 DIAG48 – Test continued after an interim interactive timeout

The interactive test experienced a timeout on one of its queries (but not the last) to the user. The test reissued the query for inputs or actions because the number of retries has not been exhausted.

This alert would only be sent if the test job was interactive (CIM_DiagnosticTest.Characteristics includes “3” (Is Interactive) and the JobSettings.InteractiveTimeout is exceeded before the client provides a response to an alert for either input or action (see DIAG34 in subclause 7.9.6 and DIAG35 in subclause 7.9.7). If JobSettings.TerminateOnTimeout = FALSE was specified and the JobSettings.ClientRetries have not been exceeded, the test will reissue the query (DIAG34 or DIAG35).

A separate test completion status message will be sent later.

The variables in this message are:

- Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name property of the DiagnosticTest instance.
- Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive) that was specified.

This could be one of the following:

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

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

With this alert the AlertType shall have the value 1 (Other). The OtherAlertType should be set to “Interim Interactive Timeout”.

With this alert the PerceivedSeverity shall have the value 2 (Information).

7.9.14 DIAG49 – Test terminated after an interactive timeout

The interactive test experienced a timeout on one of its queries to the user. The test execution is terminated because JobSettings.TerminateOnTimeout was set to TRUE and the number of retries has been exhausted.

This alert would only be sent if the test job was interactive (CIM_DiagnosticTest.Characteristics includes “3” (Is Interactive) and the JobSettings.InteractiveTimeout is exceeded before the client provides a response to an alert for either input or action (see DIAG34 in subclause 7.9.6 and DIAG35 in subclause 7.9.7). If JobSettings.TerminateOnTimeout = TRUE was specified on the request, then the Job (and test) is terminated with a JobState of 8 (Terminated).

A separate test completion status message (DIAG45) will be sent later.

The variables in this message are:

- Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name property of the DiagnosticTest instance.
- Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive) that was specified.
This could be one of the following:
- The Object path of the element
- The ElementName of the element
- A unique user friendly name not in the model (such as, asset name)

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

With this alert the AlertType shall have the value 4 (Processing Error). The test job is terminated.

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

**8 Methods**

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

**8.1 Profile conventions for operations**

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

Support for operations for each profile class (including associations) shall be as mandated in DSP1103, 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 successful 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_DiagnosticServiceJobCapabilitiesRequestedStatesSupported 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.

If the RequestedStateChange is successful (ReturnCode = 0), the CIM_ConcreteJob.TimeOfLastStateChange shall be set to the date and time of the successful completion.

EXPERIMENTAL

If the RequestedStateChange is successful (ReturnCode = 0) and the client has subscribed to the following indications, then the following AlertIndications will be sent to the client (depending on the state change requested):
Diagnostic Job Control Profile

- DIAG19 – Test Killed by client (see subclause 7.9.3)
- DIAG20 – Test Terminated by client (see subclause 7.9.4)
- DIAG21 – Test Suspended by client (see subclause 7.9.5)
- DIAG38 - Test Resumed by client (see subclause 7.9.10)

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 (see 7.9.6), 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. Instead the client can use a default value from CIM_JobSettingData (Default). 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.

If this method is supported, then DIAG34 (see 7.9.6) shall also be supported. In addition, the following alert messages may need to be supported:

- DIAG9 (Test continued after last interactive timeout using Default Values) – This shall be supported if CIM_DiagnosticServiceJobCapabilities.DefaultValuesSupported has the value of TRUE.
- DIAG48 (Test continued after an interim interactive timeout) – This shall be supported if CIM_DiagnosticServiceJobCapabilities.ClientRetriesMax has a value greater than one.
- DIAG49 (Test terminated after an interactive test timeout) – This shall be supported if CIM_JobSettingData.TerminateOnTimeout has the value of TRUE.

If the client is running an interactive test (CIM_DiagnosticTest.Characteristics includes "Is Interactive"), then the client should be subscribing for the interactive alert indications (such as, DIAG34). If the client does not subscribe to the interactive alert indications, it will never be notified that inputs are required and interactive timeouts will occur. Eventually the test will either run with default values or be terminated.
Table 5 – ResumeWithInput() method: Return code values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Completed with No Error</td>
</tr>
<tr>
<td>2</td>
<td>Unknown/Unspecified Error</td>
</tr>
<tr>
<td>3</td>
<td>Can NOT complete within Timeout Period</td>
</tr>
<tr>
<td>4</td>
<td>Failed</td>
</tr>
<tr>
<td>5</td>
<td>Invalid Parameter</td>
</tr>
<tr>
<td>6</td>
<td>JobState not Query Pending</td>
</tr>
<tr>
<td>32768..65535</td>
<td>Vendor specific</td>
</tr>
</tbody>
</table>

Table 6 – ResumeWithInput() method: Parameters

<table>
<thead>
<tr>
<th>Qualifiers</th>
<th>Name</th>
<th>Type</th>
<th>Description/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>Inputs</td>
<td>String[]</td>
<td>The client inputs being requested by the job when its state changed to 12 (Query Pending).</td>
</tr>
</tbody>
</table>

8.2.3 CIM_ConcreteJob.ResumeWithAction()

The CIM_ConcreteJob.ResumeWithAction() 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 7. No input or output parameters are defined. No standard messages are defined.

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

If this method is supported, then DIAG35 (see 7.9.7) shall also be supported. In addition, the following alert messages may need to be supported:

- DIAG48 (Test continued after an interim interactive timeout) – This shall be supported if CIM_DiagnosticServiceJobCapabilities.ClientRetriesMax has a value greater than one.
- DIAG49 (Test terminated after an interactive test timeout) – This shall be supported if CIM_JobSettingData.TerminateOnTimeout has the value of TRUE.

If the client is running an interactive test (CIM_DiagnosticTest.Characteristics includes “Is Interactive”), then the client should be subscribing for the interactive alert indications (such as, DIAG35). If the client does not subscribe to the interactive alert indications, it will never be notified that an action required and interactive timeouts will occur. Eventually the test will be terminated.
Table 7 – ResumeWithAction() method: Return code values

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

8.3 CIM_DiagnosticServiceJobCapabilities

8.3.1 CreateGoalSettings()

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

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

CIM_DiagnosticServiceJobCapabilities is optional. If not specified the client can create JobSettings based on the CIM_JobSettingData (Default) instance. CIM_JobSettingData (Default) is mandatory and will be supplied by the provider.

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

Table 8 – CreateGoalSettings( ) method: Return code values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Completed with No Error</td>
</tr>
<tr>
<td>1</td>
<td>Not supported</td>
</tr>
<tr>
<td>2</td>
<td>Unknown</td>
</tr>
<tr>
<td>3</td>
<td>Timeout</td>
</tr>
<tr>
<td>4</td>
<td>Failed</td>
</tr>
<tr>
<td>5</td>
<td>Invalid Parameter</td>
</tr>
<tr>
<td>6</td>
<td>Alternative Proposed</td>
</tr>
<tr>
<td>32768..65535</td>
<td>Vendor specific</td>
</tr>
</tbody>
</table>
Table 9 – CreateGoalSettings() method: Parameters

<table>
<thead>
<tr>
<th>Qualifiers</th>
<th>Name</th>
<th>Type</th>
<th>Description/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>TemplateGoalSettings[ ]</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>IN/OUT</td>
<td>SupportedGoalSettings[ ]</td>
<td>String</td>
<td>An array of CIM_JobSettingData embedded instances that are consistent with the CIM_DiagnosticServiceJobCapabilities and are closest matches to the input TemplateGoalSettings.</td>
</tr>
</tbody>
</table>

**EXPERIMENTAL**

8.4 CIM_MethodResult

All operations are supported as for CIM_MethodResult in [DSP1103](#).

8.5 CIM_OwningJobElement

All operations are supported as for CIM_OwningJobElement in [DSP1103](#).

8.6 CIM_AffectedJobElement

All operations are supported as for CIM_AffectedJobElement in [DSP1103](#).

8.7 CIM_AssociatedJobMethodResult

All operations are supported as for CIM_AssociatedJobMethodResult in [DSP1103](#).

8.8 CIM_HostedDependency

All operations are supported as for CIM_HostedDependency in [DSP1103](#).

8.9 CIM_RegistredProfile

All operations are supported as for CIM_RegistredProfile in [DSP1033](#).

8.10 CIM_JobSettingData

Table 10 lists operations that either have special requirements beyond those from [DSP0200](#) or shall not be supported.
Table 10 – Operations: CIM_JobSettingData

<table>
<thead>
<tr>
<th>Operation</th>
<th>Requirement</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetInstance</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>EnumerateInstances</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>ExecQuery</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>Associates</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>AssociatorNames</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>References</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>ReferenceNames</td>
<td>Optional</td>
<td>None</td>
</tr>
</tbody>
</table>

8.11 CIM_ElementSettingData

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

Table 11 – Operations: CIM_ElementSettingData

<table>
<thead>
<tr>
<th>Operation</th>
<th>Requirement</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetInstance</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>EnumerateInstances</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>Mandatory</td>
<td>None</td>
</tr>
</tbody>
</table>

8.12 CIM_ElementCapabilities

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

Table 12 – Operations: CIM_ElementCapabilities

<table>
<thead>
<tr>
<th>Operation</th>
<th>Requirement</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetInstance</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>EnumerateInstances</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>Mandatory</td>
<td>None</td>
</tr>
</tbody>
</table>

8.13 CIM_DiagnosticTest.RunDiagnosticService()

DSP1002 describes this extrinsic method. This subclause describes how the CIM_DiagnosticServiceJobCapabilities, the CIM_JobSettingData (Default), and the JobSettings input parameter affects the execution of this extrinsic method.

CIM_DiagnosticServiceJobCapabilities is implemented, the client application may specify values in the JobSettings 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 JobSettingData (Default) or the maximum allowed by the
CIM_DiagnosticServiceJobCapabilities. If the client invokes this extrinsic method and the JobSettings
parameter has the value of NULL, the CIM_JobSettingData (Default) will be used. In either case, if any
value was changed, an alert message (DIAG39, see 7.9.11) will be issued. Whether or not a value was
changed, the effective JobSettings 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 JobSettings parameter.

<table>
<thead>
<tr>
<th>DiagnosticServiceJobCapabilities</th>
<th>JobSettingData (Default)</th>
<th>JobSettings [CIM_JobSettingData (Client)]</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absent</td>
<td>Present</td>
<td>NULL</td>
<td>Use JobSettingData (Default).</td>
</tr>
<tr>
<td>Absent</td>
<td>Present</td>
<td>No conflict in values exist</td>
<td>Use JobSettings (Client input)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conflict in values exist</td>
<td>Use JobSettingData (Default). An alert is issued.</td>
</tr>
<tr>
<td>Present</td>
<td>Present</td>
<td>NULL</td>
<td>Use JobSettingData (Default).</td>
</tr>
<tr>
<td>Present</td>
<td>Present</td>
<td>No conflict in values exist</td>
<td>Use JobSettings (Client input)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conflict in values exist</td>
<td>JobSettings (Client input) is modified to conform to the capabilities. One or more alerts are issued.</td>
</tr>
<tr>
<td>Absent</td>
<td></td>
<td></td>
<td>Undefined.</td>
</tr>
<tr>
<td>Present</td>
<td>Present but conflicting</td>
<td></td>
<td>Undefined.</td>
</tr>
</tbody>
</table>

The effective JobSettings 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.
### Table 14 – Diagnostic test use cases

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

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

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 object path 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 The DiagnosticTest.RunDiagnosticService( ) always returns a reference to a job when the method returns a 0 (Success). That is, there is no case where the test is executed without a job being created.

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 object path 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. ResumeWithInput() returns 0 ("Completed with No Error") and test execution starts.

7) After the test is completed 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 object path 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. ResumeWithInput() returns 0 ("Completed with No Error") and test execution starts.

7) After the test is completed 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 object path 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. ResumeWithInput() returns 0 ("Completed with No Error") and test execution resumes.

7) Test execution finishes for 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.
9.2.4 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 object path 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]

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

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

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

- The ConcreteJob provider detects that Inputs[0]="-1" is invalid.

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

- ConcreteJob provider sets ConcreteJob.JobState= 10 (Exception) if this was the last retry. Otherwise, the provider changes the JobState to 12 (Query Pending) and reissues the DIAG34.
1) Client calls DiagnosticTest.RunDiagnosticService( ) which returns 0 (Success) and the object path 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 ConcreteJob 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) if this was the last retry. Otherwise, the provider changes the JobState to 12 (Query Pending) and reissues the DIAG34.

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 object path 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 resumes.

7) After the test is completed 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 object path 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 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 inserting the CD into the drive, ConcreteJob.ResumeWithAction( ) returns 0 ("Completed with No Error") and test execution resumes with the ConcreteJob.JobState set to 4 (Running).

10) After test is completed 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 object path 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") and test execution resumes with ConcreteJob.JobState set to 4 (Running) and ConcreteJob.PercentComplete set to 25.

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 verifying the LED is ON.

9) ConcreteJob.ResumeWithAction( ) returns 0 ("Completed with No Error") and test execution resumes with ConcreteJob.JobState set to 4 (Running) and ConcreteJob.PercentComplete set to 50.

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

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

12) Client calls ConcreteJob.ResumeWithAction( ) after verifying the LED is OFF.

13) ConcreteJob.ResumeWithAction( ) returns 0 ("Completed with No Error") and test execution resumes with ConcreteJob.JobState set to 4 (Running) and ConcreteJob.PercentComplete set to 75.

14) After test is completed 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 object path 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 CIM_JobSettingData (Default) values 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 DiagnosticService.JobCapabilities.SilentModeSupported = "TRUE". To run in silent mode, the client sets JobSettingData.RunInSilentMode = "TRUE" before invoking the DiagnosticTest.RunDiagnosticService( ) extrinsic method.

9.4.1  Running an Interactive Test in Silent Mode

The DiagnosticService.JobCapabilities.SilentModeSupported is TRUE and the default JobSettingData has RunInSilentMode set to TRUE and non-empty arrays for DefaultInputNames and DefaultInputValues.

1) Client calls DiagnosticTest.RunDiagnosticService( ) with the JobSettings parameter NULL and the method returns 0 (Success) and the object path of the ConcreteJob instance and starts the diagnostic test. ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 0.

2) When the test requires input, the ConcreteJob provider sees that the test is to run in Silent Mode (RunInSilentMode = TRUE) so the ConcreteJob gets the DefaultInputValues for the DefaultInputNames the test is looking for and supplies them to the test

- No DIAG34 message is issued to the client

3) ConcreteJob provider leaves ConcreteJob.JobState = 4 (Running) and the test continues to execute.

4) If the test requests additional inputs, steps 2 and 3 are repeated.

5) After the test is completed, the appropriate test completion status message is sent (indicating success or failure).

9.4.2  Running Silent Mode with invalid default values

The DiagnosticService.JobCapabilities.SilentModeSupported is TRUE and the client supplies its own JobSettingData (in the JobSettings parameter) with RunInSilentMode set to TRUE and non-empty arrays
for DefaultInputNames and DefaultInputValues, but the default values are invalid. (The
DefaultInputNames are valid, but the DefaultInputValues are invalid.)

1) Client calls DiagnosticTest.RunDiagnosticService( ) with the JobSettings parameter and the
method returns 0 (Success) and the object path of the ConcreteJob instance and starts the
diagnostic test. ConcreteJob.JobState = 4 (Running) and ConcreteJob.PercentComplete = 0.

2) When the test requires input, the ConcreteJob provider sees that the test is to run in Silent Mode
(RunInSilentMode = TRUE) so the ConcreteJob gets the DefaultInputValues for the
DefaultInputNames the test is looking for and supplies them to the test

- No DIAG34 message is issued to the client

3) ConcreteJob provider leaves ConcreteJob.JobState = 4 (Running) and the test continues to
execute.

4) The test cannot process the input values, so the job issues a DIAG39 message indicating that the
Default Values were not used

5) The job then gets the Default values in the Default JobSettingData and supplies these to the test

6) If the test requests additional inputs, steps 2, 3, 4 and 5 are repeated.

7) After the test is completed, the appropriate test completion status message is sent (indicating
success or failure).

NOTE If the DefaultInputValues are valid, but the DefaultInputNames are invalid, the process would be the same.
The test will not attempt to use the invalid name until it has a need for the input. If it only accesses names that are
valid, then it does not matter that one (or more) of the names are invalid.

9.5 Finding diagnostic jobs

9.5.1 Finding all diagnostic tests executed on a system

A client can find all the diagnostic tests executed 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 instances.

9.5.2 Finding all diagnostic tests executed against a ManagedElement

A client can find all diagnostic tests executed against 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 instances.
9.6 Configuring a diagnostic job

To run a diagnostic test, the client invokes the RunDiagnosticService( ) extrinsic method of DiagnosticTest. The JobSettings parameter may be an empty string, NULL or a string representing an embedded instance of JobSettingData. When JobSettings is an empty string or NULL, 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 (Default) 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 test 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 as the result role argument. 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.IsDefault property is 1 ("Is Default"). If so, the JobSettingData instance represents the default job settings.

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.

1) 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.

2) 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.

3) 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.

4) 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.ReqestedStatesSupported 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 DiagnosticTest 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) After the transition is completed successfully, the ConcreteJob.JobState property shall have the value of 5 ("Suspended") 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").

9.7.2 Resume a job for a diagnostic test

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 DiagnosticTest 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 DiagnosticServiceJobCapabilitiesRequestedStatesSupported 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) After the transition is completed successfully, the ConcreteJob.JobState property shall have the value of 4 ("Running") and ConcreteJob.TimeOfLastStateChange property shall be set to the current time.
9.7.3 Terminate a job for a diagnostic test

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 DiagnosticTest 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) After the transition is completed successfully, the ConcreteJob.JobState property shall have the value of 8 ("Terminated") and ConcreteJob.TimeOfLastStateChange property shall be set to the current time.

NOTE The JobState property may transition to 6 ("Shutting Down") before the final transition to 8 ("Terminated").

9.7.4 Kill a job for a diagnostic test

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 DiagnosticTest 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) After the transition is completed successfully, the ConcreteJob.JobState property shall have the value of 9 ("Killed") and ConcreteJob.TimeOfLastStateChange property shall be set to the current time.
9.8 Delete a job for a diagnostic test

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.

9.8.1 Client deletes a job for a diagnostic test

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 JobSettings input parameter has the value of the embedded instance of JobSettingData created in the previous step.

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

NOTE This assumes that DiagnosticServiceJobCapabilities.CleanupInterval is NULL; otherwise, the provider deletes the ConcreteJob instance after the configured interval.

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

3) 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

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Classes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIMAffectedJobElement</td>
<td>Optional</td>
<td>See 10.1.</td>
</tr>
<tr>
<td>CIM_ConcreteJob</td>
<td>Mandatory</td>
<td>See 10.2.</td>
</tr>
<tr>
<td>CIM_DiagnosticServiceJobCapabilities</td>
<td>Optional</td>
<td>See 10.3.</td>
</tr>
<tr>
<td>CIM_ElementCapabilities(Job)</td>
<td>Optional</td>
<td>See 10.4.</td>
</tr>
<tr>
<td>CIM_ElementSettingData(DefaultJobSettingData)</td>
<td>Mandatory</td>
<td>See 10.5.</td>
</tr>
<tr>
<td>CIM_FilterCollection</td>
<td>Optional</td>
<td>See 10.6.</td>
</tr>
<tr>
<td>CIM_HostedDependency</td>
<td>Mandatory</td>
<td>See 10.7.</td>
</tr>
<tr>
<td>CIM_IndicationFilter</td>
<td>Mandatory</td>
<td>See 10.8.</td>
</tr>
<tr>
<td>CIM_JobSettingData(Client)</td>
<td>Optional</td>
<td>See 10.9.</td>
</tr>
<tr>
<td>CIM_JobSettingData(DFault)</td>
<td>Mandatory</td>
<td>See 10.10.</td>
</tr>
<tr>
<td>CIM_MemberOfCollection</td>
<td>Optional</td>
<td>See 10.11.</td>
</tr>
<tr>
<td>CIM_OwningCollectionElement</td>
<td>Optional</td>
<td>See 10.12.</td>
</tr>
<tr>
<td>CIM_OwningJobElement</td>
<td>Mandatory</td>
<td>See 10.13.</td>
</tr>
<tr>
<td>CIM_RegisteredProfile</td>
<td>Mandatory</td>
<td>See 10.14.</td>
</tr>
<tr>
<td><strong>Indications</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELECT * FROM CIM_AlertIndication WHERE OwningEntity=&quot;DMTF&quot; and MessageID=&quot;DIAG9&quot;</td>
<td>Conditional</td>
<td>This indication is mandatory if CIM_DiagnosticTest.Characteristics has the value of 3 (Is Interactive) and DiagnosticTest.ResumeWithInput is supported. Query Language = &quot;DMTF:CQL&quot; Name = &quot;DMTF:Diagnostic Job Control:DIAG9&quot; See 7.9.1</td>
</tr>
<tr>
<td>SELECT * FROM CIM_AlertIndication WHERE OwningEntity=&quot;DMTF&quot; and MessageID=&quot;DIAG12&quot;</td>
<td>Optional</td>
<td>Query Language = &quot;DMTF:CQL&quot; Name = &quot;DMTF:Diagnostic Job Control:DIAG12&quot; See 7.9.2</td>
</tr>
<tr>
<td>SELECT * FROM CIM_AlertIndication WHERE OwningEntity=&quot;DMTF&quot; and MessageID=&quot;DIAG19&quot;</td>
<td>Mandatory</td>
<td>Query Language = &quot;DMTF:CQL&quot; Name = &quot;DMTF:Diagnostic Job Control:DIAG19&quot; See 7.9.3</td>
</tr>
<tr>
<td>Element Name</td>
<td>Requirement</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| SELECT * FROM CIM_ALERTINDICATION WHERE OwningEntity="DMTF" and MessageID="DIAG20" | Mandatory   | Query Language = “DMTF:CQL”  
Name = “DMTF:Diagnostic Job Control:DIAG20”  
See 7.9.4                                                                                   |
| SELECT * FROM CIM_ALERTINDICATION WHERE OwningEntity="DMTF" and MessageID="DIAG21" | Optional    | Query Language = “DMTF:CQL”  
Name = “DMTF:Diagnostic Job Control:DIAG21”  
See 7.9.5                                                                                   |
| SELECT * FROM CIM_ALERTINDICATION WHERE OwningEntity="DMTF" and MessageID="DIAG34" | Conditional | This indication is mandatory if CIM_DiagnosticTest.Characteristics has the value of 3 (Is Interactive) and DiagnosticTest.ResumeWithInput is supported.  
Query Language = “DMTF:CQL”  
Name = “DMTF:Diagnostic Job Control:DIAG34”  
See 7.9.6                                                                                   |
| SELECT * FROM CIM_ALERTINDICATION WHERE OwningEntity="DMTF" and MessageID="DIAG35" | Conditional | This indication is mandatory if CIM_DiagnosticTest.Characteristics has the value of 3 (Is Interactive) and DiagnosticTest.ResumeWithAction is supported.  
Query Language = “DMTF:CQL”  
Name = “DMTF:Diagnostic Job Control:DIAG35”  
See 7.9.7                                                                                   |
| SELECT * FROM CIM_ALERTINDICATION WHERE OwningEntity="DMTF" and MessageID="DIAG36" | Optional    | Query Language = “DMTF:CQL”  
Name = “DMTF:Diagnostic Job Control:DIAG36”  
See 7.9.8                                                                                   |
| SELECT * FROM CIM_ALERTINDICATION WHERE OwningEntity="DMTF" and MessageID="DIAG37" | Optional    | Query Language = “DMTF:CQL”  
Name = “DMTF:Diagnostic Job Control:DIAG37”  
See 7.9.9                                                                                   |
| SELECT * FROM CIM_ALERTINDICATION WHERE OwningEntity="DMTF" and MessageID="DIAG38" | Optional    | Query Language = “DMTF:CQL”  
Name = “DMTF:Diagnostic Job Control:DIAG38”  
See 7.9.10                                                                                   |
| 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”  
Name = “DMTF:Diagnostic Job Control:DIAG39”  
See 7.9.11                                                                                   |
| 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”  
Name = “DMTF:Diagnostic Job Control:DIAG40”  
See 7.9.12                                                                                   |
### 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.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AffectedElement (overridden)</td>
<td>Mandatory</td>
<td>Key. The property shall be a reference to an instance of the CIM_ManagedElement being tested.</td>
</tr>
<tr>
<td>AffectingElement</td>
<td>Mandatory</td>
<td>Key. The property shall be a reference to an instance of CIM_ConcreteJob executing the test.</td>
</tr>
<tr>
<td>ElementEffects</td>
<td>Optional</td>
<td>See 7.1.1.</td>
</tr>
<tr>
<td>OtherElementEffectsDescriptions</td>
<td>Optional</td>
<td>If ElementEffects contains the value 1 (Other), this property is Mandatory.</td>
</tr>
</tbody>
</table>

### 10.2 CIM_ConcreteJob

Each successful invocation of the CIM_DiagnosticTest.RunDiagnosticService() extrinsic method returns a CIM_ConcreteJob instance. Each CIM_ConcreteJob instance represents a diagnostic test execution.

This class specializes CIM_ConcreteJob as defined in [DSP1103](#). The constraints listed in Table 17 are in addition to those specified in [DSP1103](#). See [DSP1103](#) for other mandatory elements that must be implemented.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstanceID</td>
<td>Mandatory</td>
<td>Key. See 7.2.1.</td>
</tr>
<tr>
<td>Name</td>
<td>Mandatory</td>
<td>See 7.2.2.</td>
</tr>
<tr>
<td>JobState</td>
<td>Mandatory</td>
<td>See 7.2.3.</td>
</tr>
<tr>
<td>DeleteOnCompletion</td>
<td>Mandatory</td>
<td>See 7.2.4.</td>
</tr>
<tr>
<td>TimeBeforeRemoval</td>
<td>Mandatory</td>
<td>See 7.2.5.</td>
</tr>
<tr>
<td>StartTime</td>
<td>Mandatory</td>
<td>See 7.2.6.</td>
</tr>
</tbody>
</table>
### 10.3 CIM_DiagnosticServiceJobCapabilities

The CIM_DiagnosticServiceJobCapabilities is used to provide information about on the capabilities of the job that is used to run the diagnostic test. Table 18 contains the requirements for the elements of this class.

#### Table 18 – Class: CIM_DiagnosticServiceJobCapabilities

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ElapsedTime</td>
<td>Mandatory</td>
<td>See 7.2.7.</td>
</tr>
<tr>
<td>RequestedState</td>
<td>Mandatory</td>
<td></td>
</tr>
<tr>
<td>PercentComplete</td>
<td>Mandatory</td>
<td>See 7.2.8.</td>
</tr>
<tr>
<td>TimeOfLastStateChange</td>
<td>Mandatory</td>
<td>See 7.2.9.</td>
</tr>
<tr>
<td>RequestStateChange()</td>
<td>Mandatory</td>
<td>See 8.2.1.</td>
</tr>
<tr>
<td>ResumeWithInput()</td>
<td>Conditional</td>
<td>See 8.2.2.</td>
</tr>
<tr>
<td>ResumeWithAction()</td>
<td>Conditional</td>
<td>See 8.2.3.</td>
</tr>
</tbody>
</table>

### 10.4 CIM_ElementCapabilities (Job)

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

#### Table 19 – Class: CIM_ElementCapabilities

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManagedElement</td>
<td>Mandatory</td>
<td>Key. The property shall be a reference to an instance of CIM_DiagnosticTest.</td>
</tr>
<tr>
<td>Capabilities</td>
<td>Mandatory</td>
<td>Key. The property shall be a reference to an instance of CIM_DiagnosticServiceJobCapabilities.</td>
</tr>
</tbody>
</table>

### 10.5 CIM_ElementSettingData (Default JobSettingData)

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

**Table 20 – Class: CIM_ElementSettingData**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManagedElement</td>
<td>Mandatory</td>
<td>Key. The property shall be a reference to an instance of CIM_DiagnosticTest.</td>
</tr>
<tr>
<td>SettingData</td>
<td>Mandatory</td>
<td>Key. The property shall be a reference to an instance of CIM_JobSettingData (Default).</td>
</tr>
<tr>
<td>IsDefault</td>
<td>Mandatory</td>
<td>The value of the property shall have the value TRUE.</td>
</tr>
</tbody>
</table>

**10.6 CIM_FilterCollection (ProfileSpecificFilterCollection)**

CIM_FilterCollection represents a ProfileSpecificFilterCollection as defined in DSP1054. It defines the collection of all the alert indications of the Diagnostic Job Control profile. Table 21 contains the requirements for elements of this class.

**Table 21 – Class: CIM_FilterCollection**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstanceID</td>
<td>Mandatory</td>
<td>Key: See DSP1054</td>
</tr>
<tr>
<td>CollectionName</td>
<td>Mandatory</td>
<td>The property shall have the value “DMTF:Diagnostic Job Control: ProfileSpecifiedAlertIndicationFilterCollection”</td>
</tr>
</tbody>
</table>

**10.7 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 22 contains the requirements for elements of this class.

**Table 22 – Class: CIM_HostedDependency**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antecedent</td>
<td>Mandatory</td>
<td>Key. The property shall be a reference to an instance of CIM_System.</td>
</tr>
<tr>
<td>Dependent</td>
<td>Mandatory</td>
<td>Key. The property shall be a reference to an instance of CIM_ConcreteJob executing the test.</td>
</tr>
</tbody>
</table>

**10.8 CIM_IndicationFilter (StaticIndicationFilter)**

CIM_IndicationFilter represents a StaticIndicationFilter as defined in DSP1054. It defines the format of all the alert indication filters in the Diagnostic Job Control profile. Table 23 contains the requirements for elements of this class.
Table 23 – Class: CIM_IndicationFilter (StaticIndicationFilter)

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Mandatory</td>
<td>Key: See the Name values as identified in Table 15.</td>
</tr>
<tr>
<td>CreationClassName</td>
<td>Mandatory</td>
<td>Key: See DSP1054.</td>
</tr>
<tr>
<td>SystemName</td>
<td>Mandatory</td>
<td>Key: See DSP1054.</td>
</tr>
<tr>
<td>SystemCreationClassName</td>
<td>Mandatory</td>
<td>Key: See DSP1054.</td>
</tr>
<tr>
<td>SourceNamespaces[ ]</td>
<td>Mandatory</td>
<td>See DSP1054.</td>
</tr>
<tr>
<td>IndividualSubscriptionSupported</td>
<td>Mandatory</td>
<td>See DSP1054.</td>
</tr>
<tr>
<td>Query</td>
<td>Mandatory</td>
<td>See the Query values as identified in Table 15.</td>
</tr>
<tr>
<td>QueryLanguage</td>
<td>Mandatory</td>
<td>See the QueryLanguage values as identified in Table 15.</td>
</tr>
</tbody>
</table>

10.9 CIM_JobSettingData (Client)

This CIM_JobSettingData definition represents the JobSettings parameter the client passes when it invokes the RunDiagnosticService( ) extrinsic method of the DiagnosticTest. Table 24 contains the requirements for elements of this class.

Table 24 – Class: CIM_JobSettingData (Client)

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstanceID</td>
<td>Mandatory</td>
<td>Key. See 7.4.1.</td>
</tr>
<tr>
<td>DeleteOnCompletion</td>
<td>Conditional</td>
<td>See 7.4.2.</td>
</tr>
<tr>
<td>InteractiveTimeout</td>
<td>Conditional</td>
<td>See 7.4.3.</td>
</tr>
<tr>
<td>TerminateOnTimeout</td>
<td>Conditional</td>
<td>See 7.4.4.</td>
</tr>
<tr>
<td>DefaultInputsValues</td>
<td>Conditional</td>
<td>See 7.4.5.</td>
</tr>
<tr>
<td>DefaultInputNames</td>
<td>Conditional</td>
<td>See 7.4.6.</td>
</tr>
<tr>
<td>ClientRetries</td>
<td>Conditional</td>
<td>See 7.4.7.</td>
</tr>
<tr>
<td>RunInSilentMode</td>
<td>Conditional</td>
<td>See 7.4.8.</td>
</tr>
<tr>
<td>ElementName</td>
<td>Mandatory</td>
<td>This property is a free-form string of variable length. (pattern &quot;.*&quot;)</td>
</tr>
</tbody>
</table>

10.10 CIM_JobSettingData (Default)

This CIM_JobSettingData definition represents the default settings for the job used to run the diagnostic test. Table 25 contains the requirements for elements of this class. Each CIM_DiagnosticTest (or a subclass) shall have one associated instance of this class.

Table 25 – Class: CIM_JobSettingData (Default)

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstanceID</td>
<td>Mandatory</td>
<td>Key. See 7.5.1.</td>
</tr>
<tr>
<td>DeleteOnCompletion</td>
<td>Conditional</td>
<td>See 7.5.2.</td>
</tr>
<tr>
<td>InteractiveTimeout</td>
<td>Conditional</td>
<td>See 7.5.3.</td>
</tr>
<tr>
<td>TerminateOnTimeout</td>
<td>Conditional</td>
<td>See 7.5.4.</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultInputValues</td>
<td>Conditional</td>
<td>See 7.5.5.</td>
</tr>
<tr>
<td>DefaultInputNames</td>
<td>Conditional</td>
<td>See 7.5.6.</td>
</tr>
<tr>
<td>ClientRetries</td>
<td>Conditional</td>
<td>See 7.5.7.</td>
</tr>
<tr>
<td>RunInSilentMode</td>
<td>Conditional</td>
<td>See 7.5.8.</td>
</tr>
<tr>
<td>ElementName</td>
<td>Mandatory</td>
<td>This property is a free-form string of variable length. (pattern &quot;.*&quot;)</td>
</tr>
</tbody>
</table>

### 10.11 CIM_MemberOfCollection (ProfileSpecificMemberOfCollection)

CIM_MemberOfCollection represents an association between the profile specific FilterCollection and the CIM_IndicationFilters for the alert indications. Table 26 contains the requirements for elements of this class.

#### Table 26 – Class: CIM_MemberOfCollection

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection</td>
<td>Mandatory</td>
<td>Key. Value shall reference the profile specific FilterCollection instance representing a filter collection containing the alert indication filters.</td>
</tr>
<tr>
<td>Member</td>
<td>Mandatory</td>
<td>Key. Value shall reference an Alert IndicationFilter instance representing a contained alert indication filter.</td>
</tr>
</tbody>
</table>

### 10.12 CIM_OwningCollectionElement

CIM_OwningCollectionElement represents an association between the IndicationService that controls the profile specific FilterCollection and the profile specific CIM_FilterCollection for the alert indication filters. Table 27 contains the requirements for elements of this class.

#### Table 27 – Class: CIM_OwningCollectionElement

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OwningElement</td>
<td>Mandatory</td>
<td>Key. See DSP1054</td>
</tr>
<tr>
<td>OwnedElement</td>
<td>Mandatory</td>
<td>Key. Value shall be a reference to the DMTF profile specific Alert Indication FilterCollection instance</td>
</tr>
</tbody>
</table>

### 10.13 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_ManagedElement. The constraints listed in Table 28 are in addition to those specified in DSP1103. See DSP1103 for other mandatory properties of CIM_HostedService that must be implemented.
Table 28 – Class: CIM_OwningJobElement

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OwningElement</td>
<td>Mandatory</td>
<td>Key. The property shall be a reference to an instance of CIM_DiagnosticTest.</td>
</tr>
<tr>
<td>OwnedElement</td>
<td>Mandatory</td>
<td>Key. The property shall be a reference to an instance of CIM_ConcreteJob executing the test.</td>
</tr>
</tbody>
</table>

10.14 CIM_RegisteredProfile

The CIM_RegisteredProfile class is defined in DSP1033. The constraints listed in Table 29 are in addition to those specified in DSP1033. See DSP1033 for other mandatory properties of CIM_RegisteredProfile that must be implemented.

Table 29 – Class: CIM_RegisteredProfile

<table>
<thead>
<tr>
<th>Properties</th>
<th>Requirement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegisteredName</td>
<td>Mandatory</td>
<td>The value of the property shall have the value “Diagnostic Job Control”</td>
</tr>
<tr>
<td>RegisteredVersion</td>
<td>Mandatory</td>
<td>The value of the property shall have the value “1.0.0”</td>
</tr>
<tr>
<td>RegisteredOrganization</td>
<td>Mandatory</td>
<td>The value of the property shall be 2 (DMTF)</td>
</tr>
</tbody>
</table>
## ANNEX A
(informative)

### Change log

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2010-12-05</td>
<td>Initial Version</td>
</tr>
<tr>
<td>1.0.0a</td>
<td>2012-09-11</td>
<td>Work In Progress version</td>
</tr>
<tr>
<td>1.0.0b</td>
<td>2012-10-27</td>
<td>Changed the version to 1.0.0b</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 Terms and definitions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Modified the definition of Job</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added a definition of test</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 Synopsis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Changed the version to 1.0.0b</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 Implementation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added a subclause &quot;7.8 Diagnostics Job Control Profile Indications Support&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added a subclause &quot;7.9 Diagnostics Job Control Alert Indications and Standard Messages&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 CIM elements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Edited the Indications entries</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- changed the format of QueryLanguage entries to avoid confusion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added a Name element to the Description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Replaced the text with a reference to the appropriate Implementation subclause</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added a class table for CIM_FilterCollection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added a class table for CIM_IndicationFilter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added a class table for CIM_MemberOfCollection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added a class table for CIM_OwningCollectionElement</td>
</tr>
<tr>
<td></td>
<td>2012-11-03</td>
<td>Changed “section” to “subclause”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed “JobSetting” to “JobSettings”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Edited Methods section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Edited Requirements on Indications, particularly the Conditional statements</td>
</tr>
<tr>
<td></td>
<td>2012-12-17</td>
<td>Integrated comments from Peter and updated the figure</td>
</tr>
<tr>
<td></td>
<td>2013-05-20</td>
<td>Work in Progress version</td>
</tr>
</tbody>
</table>

**NEED TO FIX THE TABLES**