CIM Common Diagnostic Model Technical Note
CIM Version 2.11
Version 0.3   (8/29/06)

1 Introduction

1.1 Overview

The CDM is an architecture and methodology for exposing system diagnostic instrumentation through the CIM standard interfaces. Standardization of these interfaces means that clients, providers, and tests gain a certain degree of portability and, in many cases, need only be written once to satisfy multiple environments and platforms. OEMs can differentiate their diagnostic offerings by how effectively their applications use the information and capabilities available through CIM to maintain and service their systems.

The purpose of this paper is to provide an overview of the model and its intended usage. The Common Diagnostic Model Profile should be referenced to obtain a detailed understanding of the model and its use cases.

1.2 Background Reference Material

DMTF DSP1002, Common Diagnostic Model Profile
　　　http://www.dmtf.org/standards/profiles/
DMTF DSP2000, CIM Diagnostic Model White Paper
　　　http://www.dmtf.org/standards/published_documents/
CIM MOFs and Visios
　　　http://www.dmtf.org/standards/cim/
DMTF Common Information Model (CIM) Specification V2.2
　　　http://www.dmtf.org/spec/cims.html

1.3 Terminology

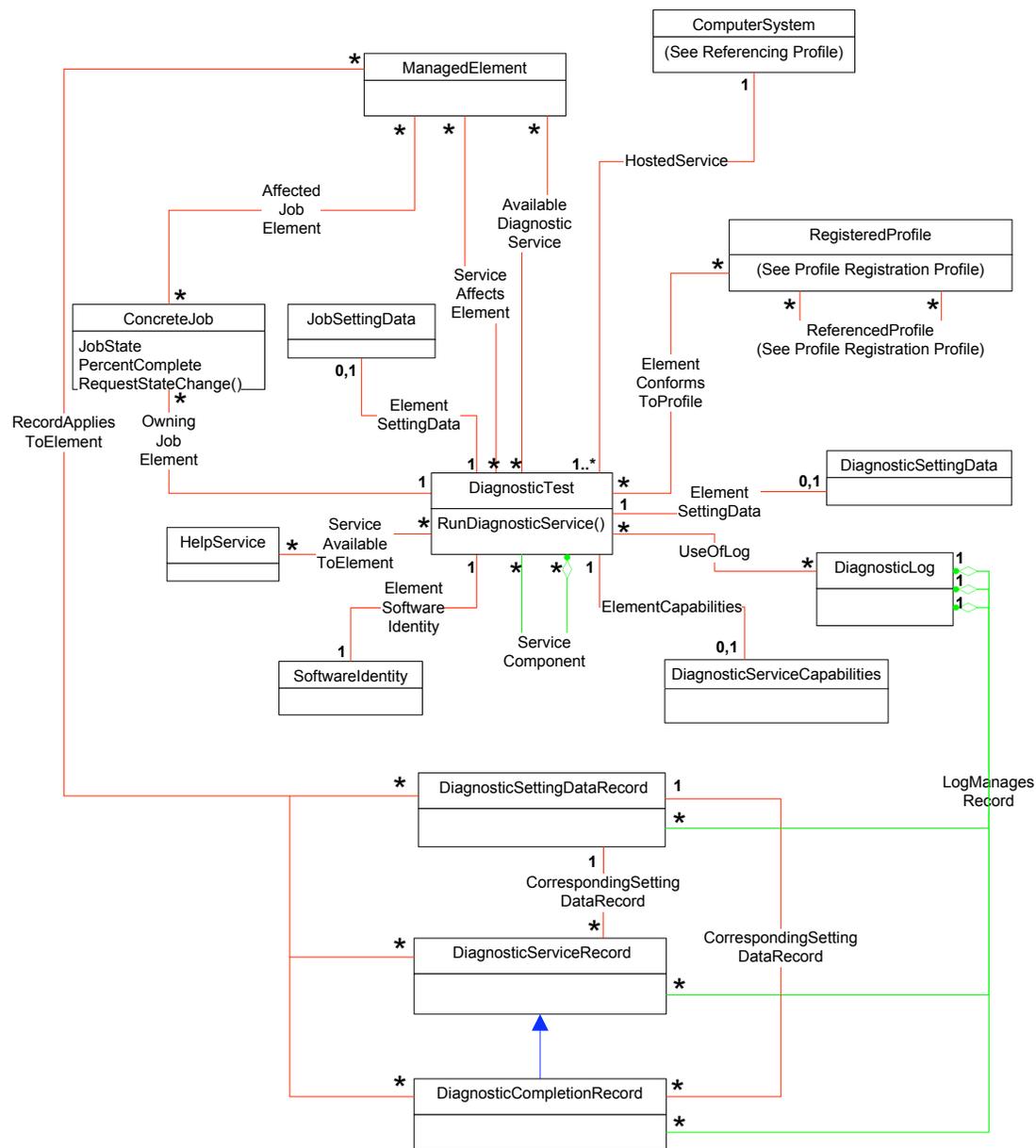| Term	Definition
CDM	Common Diagnostic Model
CIM	Common Information Model
DMTF	Distributed Management Task Force
Use Case	A description of a client application goal and how the model can be used to accomplish that goal.
MOF	Managed Object Format, the "language defining the CIM object classes, properties, methods and associations.

2 The Common Diagnostic Model

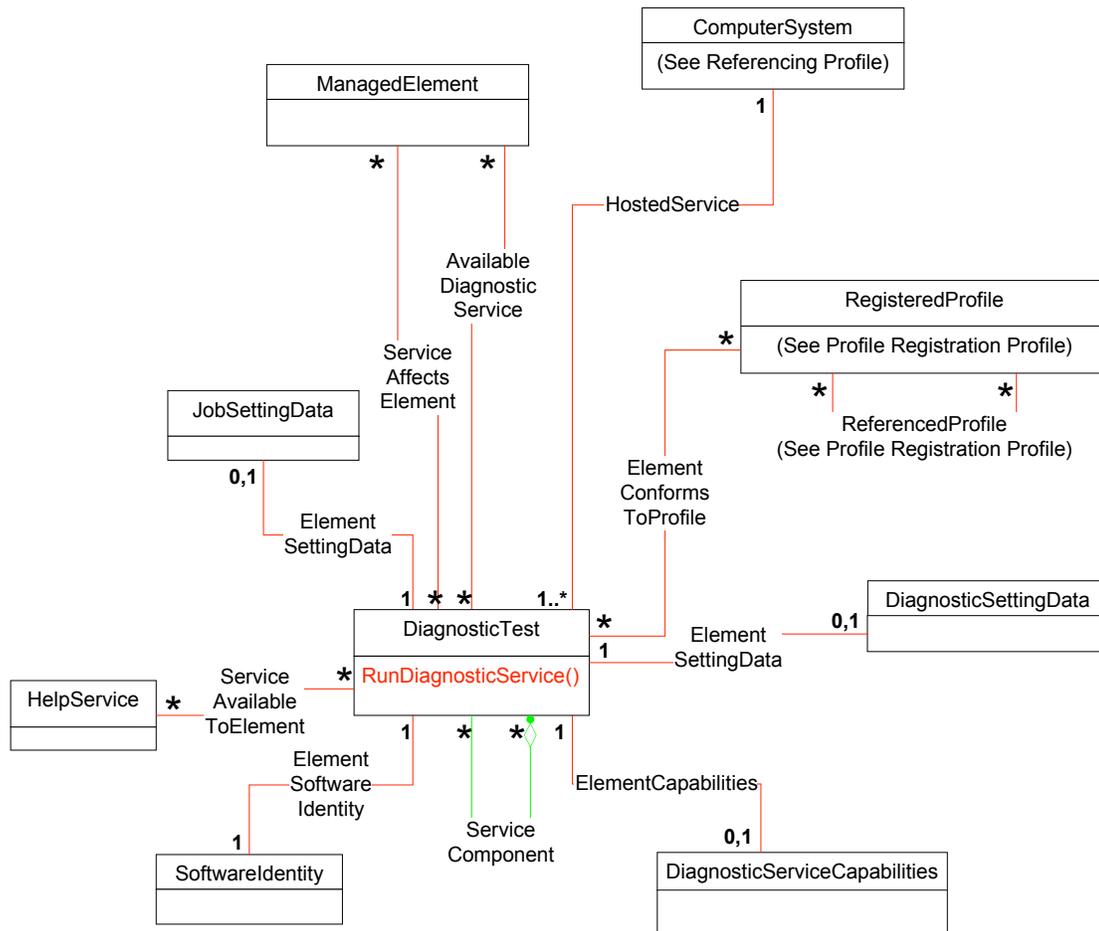## 2.1 Conceptual Areas Addressed by the Model

CDM allows Diagnostic Client applications to:
- discover, configure and execute diagnostic tests
- view progress and control test execution
- view and manage test execution results

The figure below shows the entire model. The description of the model will be broken into three sections, one for each of the client activities listed above. Each section will include a figure for the portion of the model related to that activity and provide a description of the classes, associations, properties and methods in the figure.

ComputerSystem
(See Referencing Profile)

ManagedElement

HostedService

RegisteredProfile
(See Profile Registration Profile)

ReferencedProfile
(See Profile Registration Profile)

Affected
Job
Element

Available
Diagnostic
Service

Service
Affects
Element

ConcreteJob
JobState
PercentComplete
RequestStateChange()

JobSettingData

Element
Conforms
ToProfile

Element
SettingData

RecordApplies
ToElement

Owning
Job
Element

DiagnosticTest
RunDiagnosticService()

DiagnosticSettingData

Element
SettingData

HelpService

Service
Available
ToElement

UseOfLog

DiagnosticLog

Element
Software
Identity

ElementCapabilities

Service
Component

SoftwareIdentity

DiagnosticServiceCapabilities

LogManages
Record

DiagnosticSettingDataRecord

CorrespondingSetting
DataRecord

DiagnosticServiceRecord

CorrespondingSetting
DataRecord

DiagnosticCompletionRecord

2.1.1 Discover, Configure and Execute Diagnostic Tests

DiagnosticTest is the central class of the model. This class represents the tests available on a system. This class contains the RunDiagnosticService() method used to initiate a diagnostic test. The method takes as input a ManagedElement to execute against and optionally a JobSettingData and DiagnosticSettingData to control the execution of the test. It returns a reference to a ConcreteJob to be used to view progress and control the execution of the test. See section 2.1.2

RegisteredProfile contains the information about the profile the diagnostic conforms to.

ElementConformsToProfile associates the diagnostic to the conforming profile. Compliant diagnostics are discovered by following this association from the diagnostic to the profile.

AvailableDiagnosticService associates the diagnostic to the ManagedElement it tests. Diagnostics for ManagedElements are discovered by following this association from the ManagedElement to the diagnostics.

HostedService associates the diagnostic to the ComputerSystem where the diagnostic is hosted. Diagnostics available on a system are discovered by following this association to the diagnostics.

DiagnosticServiceCapabilities contains information about the capabilities of a diagnostic test, such as the level of logging available, the type of execution control available, etc.

DiagnosticSettingData contains the settings that are available for the diagnostic to be used when the diagnostic is initiated. This can either be the default settings, or a new setting created by the client and passed into the RunDiagnosticService() method.

JobSettingData contains the settings to be used for Job creation when the diagnostic test is initiated. This can either be the default settings, or a new setting created by the client and passed into the RunDiagnosticService() method.

ElementSettingData associates the settings to the diagnostic test. When ElementSettingData.IsDefault is TRUE, the setting is the default setting that will be used when a diagnostic test is initiated and no other setting is provided.

HelpService describes the nature of the available help documents and a method to request needed documents.

ServiceAvailableToElement associates the diagnostic to its help information.

## 2.1.2 View Progress and Control Test Execution
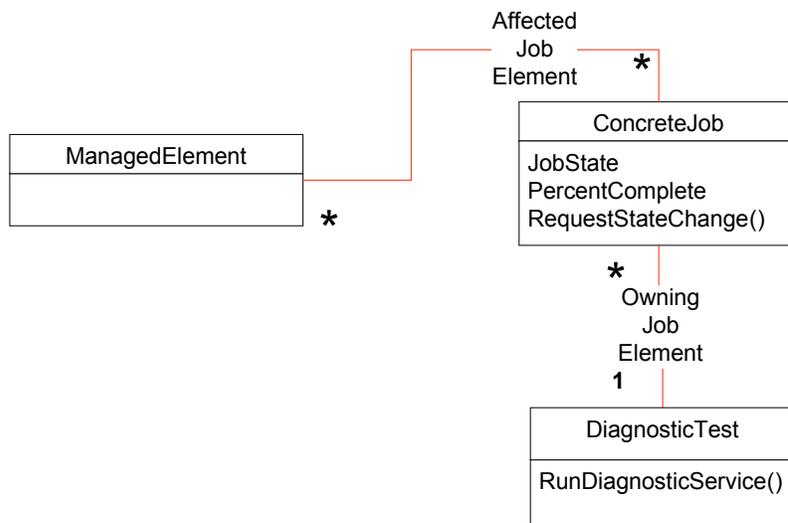
ConcreteJob represents the diagnostic execution. It contains information about the progress of the test execution, such as JobState and PercentComplete. ConcreteJob.RequestStateChange() method changes the state of the ConcreteJob. This method is used to suspend, resume and abort a diagnostic execution.

AffectedJobElement associates the job to the ManagedElement affected by the diagnostic execution. Diagnostics executing on ManagedElements can be discovered by following this association from the ManagedElement to the job.

OwningJobElement associates the Job to the diagnostic execution represented by the job. Diagnostic executions can be discovered by following this association from the diagnostic to the job.

2.1.3 View and Manage Test Execution Results
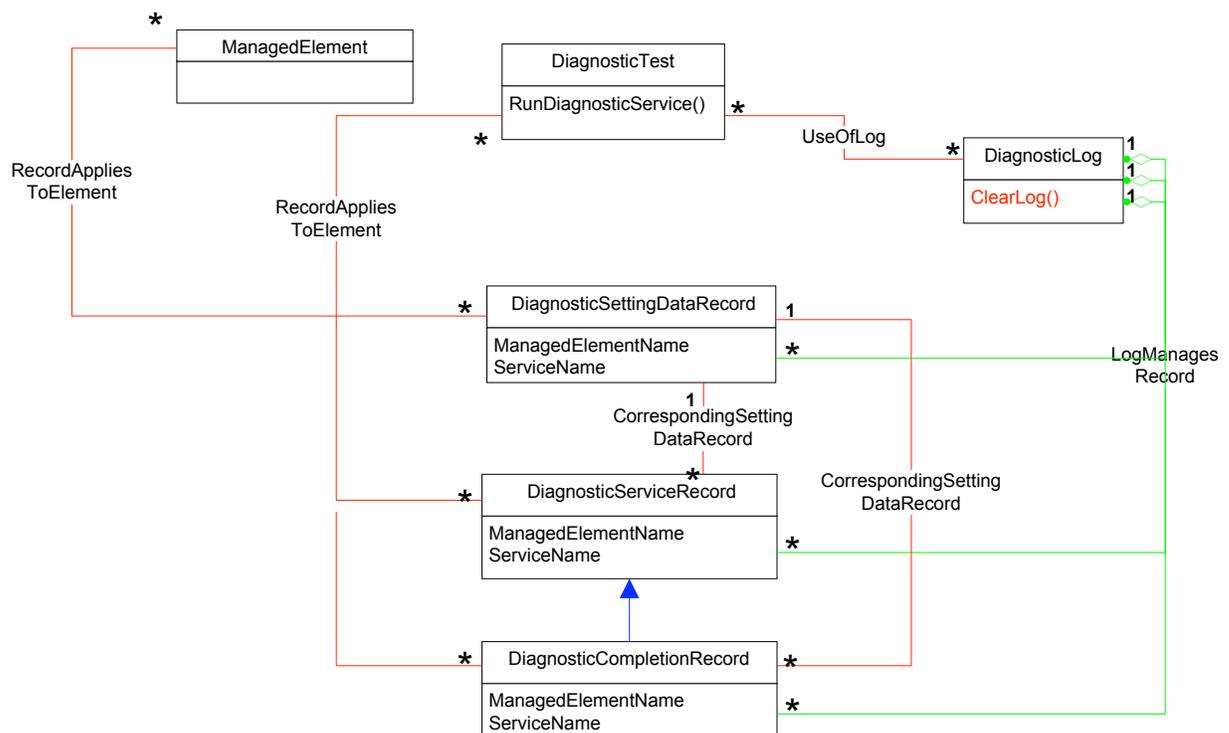
DiagnosticLog represent the log of the information logged by the diagnostic. DiagnosticLog.ClearLog() method clears the log of all records.

UseOfLog associates the log to the diagnostic. Logged information for a diagnostic can be discovered by following this association from the diagnostic to the log.

DiagnosticSettingDataRecord contains the settings used to initiate the diagnostic. Settings used for a previous execution of a diagnostic can be discovered by finding the DiagnosticSettingDataRecord in the log with the desired ServiceName and ManagedElementName.

DiagnosticCompletionRecord contains the final results of the diagnostic execution. The final results of a previous execution of a diagnostic can be discovered by finding the

DiagnosticCompletionRecord in the log with the desired ServiceName and ManagedElementName.

DiagnosticServiceRecord contains all other logged information, progress, errors, warnings, etc. The information for a previous execution of a diagnostic can be discovered by finding the DiagnosticCompletionRecord in the log with the desired ServiceName and ManagedElementName.

LogManagesRecord aggregates each record in the log. Information about the previous execution of a diagnostic and the final results can be discovered by following this association to the records in the log.

RecordAppliesToElement associates each record to the ManagedElement and the DiagnosticTest. Information about diagnostics that have been executed against a ManagedElement can be discovered by following this association to the records in a log. Information about diagnostics that have been executed can be discovered by following this association to the records in a log.

CorrespondingSettingDataRecord associates each non-setting record to the setting record for that execution of the diagnostic. Settings used for an execution of a diagnostic that generated a particular record can be discovered by following this association to the setting record.

Notice