,I **Specification** DSP0107

STATUS: Preliminary

COMMON INFORMATION MODEL (CIM) INDICATIONS (FINAL DRAFT)

Version 2.5 Final Draft

December 14, 2000

Abstract

The Common Information Model (CIM) is an object-oriented information model defined by the Distributed Management Task Force (DMTF) that provides a conceptual framework for describing management data [1].

This document describes CIM Indications and how they are used to communicate occurrences of events in the CIM. This document also describes the classes that enable clients to subscribe to CIM Indications including how to specify a desired mode of delivery. The Specification for CIM Operations over HTTP defines the XML encoding for CIM Indications over HTTP [3]. Other protocols may be defined in future releases.

Participants

Many thanks to the following companies and organizations whose generous participation and contribution in the Distributed Management Task Force Event Technical Work Group make this document possible:

- BMC Software
- Cisco Systems
- Compaq Computer Corp.

- EMC
  - Evidian
- IBM
- Intel Corp.
- LSI Corp.
- Microsoft Corp.
  - SNIA
- Smart Technology Enable
- Sun Microsystems
- Symantec Corp
- Tivoli Systems, Inc.

Change History

Version 1	1 Dec 1998	First internal review (J. Patrick Thompson, Microsoft)
Version 1.1	24 Jan 1999	Second internal review (Holger Dietrich, HP) incorporating SysDev event paper and additional feedback
Version 1.2	22 Feb 1999	Merging alternate proposals for event type hierarchies (Holger Dietrich, HP)
Version 1.3	26 Mar 1999	Refining Event type hierarchy according to feedback from cs-events. Insert more detailed explanation of meta model instantiation within the Core model. (Holger Dietrich, HP)
Version 1.4	12 April 1999	Switching back to event type hierarhy as proposed by WMI from Microsoft. Refining meta model (method parameter, …). (Holger Dietrich, HP)
Version 1.5	13 April 1999	Correcting typos, inserting corrected diagrams and MOF. (Holger Dietrich, HP)
Version 1.6 Draft	July 29 1999	Reflect output of Hillsboro meeting
Version 1.7 Draft	Aug 24 1999	Reflect working group discussions
Verison 1.8 Draft	Nov 10 1999	Incroporate Requirements analysis
Version 1.9 Draft	Nov 15 1999	Remove CIM_object
Version 1.9.1 Draft	Dec 8 1999	Outline reflecting 11/19 and 12/8 meetings

Version 1.10 Draft	Dec 29, 1999	Complete revision by J. Patrick Thompson, Rudyard Merriam, and Stephen Schleimer
Version 1.11 Draft	February 24, 2000	Embedding in new DMTF format. Editorial changes from January F2F and Teleconferences through February 17, 2000 by Schleimer
Version 1.12 Draft	March 24, 2000	Editorial changes from March F2F and Teleconferences through March 16, 2000 by Shaw
Version 1.13 Draft	April 12, 2000	Editorial changes by Schleimer
Version 1.14 Draft	April 18, 2000	Editorial changes by Shaw to add to reference and requirement sections
Version 1.15 Draft	April 21, 2000	Editorial changes by Shaw to add review comments from email and 4/20/00 teleconf; EventTime (not GenerationTime), CIM_InstIndication (not LifeCycle), CIM_ClassIndication (not MetaIndication), add discussion on surpressing event storms in Filter example, Subscriber identity property in CIM_Indication, CIM_IndSFDelivery, finish requirement section.
Version 1.16 Draft	May 9, 2000	Editorial changes by Schleimer: Add discussion of Filters and Namespaces; update subscription description; discuss multi-subscription to single Filter for single subscriber behavior; discuss ReturnName in Filter and CIM_Indication.
Version 1.17	July 26, 2000	Edited by Shaw to include changes resulting from Jun 27, 28 F2F at Intel in Portland: Alert hierarchy, DMI, TMN, SNMP mapping, refined subscription model and discussion. Including WG sessions thru 7/20/2000.
Version 1.18	August 7, 2000	Edited by Shaw to include changes to CIM_AlertIndication.Severity, created appendices A - Interop Issues, B - Requirements, C - Mapping other event systems, Added CIM_ClassModification.PreviousClass, changed CIM_ClassIndication.SourceDefinition to SourceClass, corrected sundry errors in text.
Version 1.19 Final Draft	October 24, 2000	Edited by Shaw to include clarification text for rational in model for ProcessIndication tree and clarification of the role of Providers with respect to Indications. Added Filter examples for AlertIndication and SNMPTrapIndication. Removed CIM_ preface from most class names in discussion text. Rrom 10/26/00 telecon vote: remove FilterPath and add SubscriptionID; Upon review & acceptance Version 1.19 will be progressed to Version 2.5 to synch up with CIM specification level.
Version 2.5 Company Review	November 6, 2000	Added changes agreed upon at 11/2/00 teleconf: Change AlertIndication.AlertType from "General." to "Primary classification ..., Updated text for

Draft		AlertIndication.AlertType "7", Added the Override of AlertType (to set the default = 7) to AlertInstIndication., Changed IndicationSubscription.SubscriptionID to string SubscriptionAlias.
Version 2.5 Company Review Final	November 9, 2000	Errata: 1.1 'filter is an SQL statement' s/b 'Filter contains a Query ..'; chg'd all occurrences of EventTime to IndicationTime; removed __PATH from InstModification filter example 2 and removed redundant clause to check previous state; remove [] from InstMethodCall.MethodParameters and added text to explain __MethodParameters ; CIM_InstMethodCall.ReturnValue is just a string (Removed EmbeddedObject qualifier); added text to specify what DMTF Query Language MUST support; add name, value, datatype arrays to SNMPTrapIndication; Glossary: add Handler, update Filter; add new visios. Add text to encoding for SubscriptionAlias.
Version 2.5g	November 16,2000	F version VISIO - to allow generalization of IndicationIdentifiers: changes the names of the AlertIdentifier and CorrelatedAlerts properties in CIM_AlertIndication to IndicationIdentifier, CorrelatedIndications. To allow possible future promotion of these properties higher in the object hierarchy, we must have more generic names. g version of VISIO to remove REQUIRED from SourceNamespace
Version 2.5f	December 14, 2000	Cshaw added Corrections and minor edits from Agbabian 11/20/00. Also edits to: Glossary, section 2.1.1 (CIM_InstMethodCall), 2.1.2, 2.2.1, Filter examples 3 and 5, section 2.2.3 (added Name, changed Owner). Chenged DMTF query language to WBEM query language. Removed reference to SubscriptionAlias as per vote at Santa Monica F2F. Added filter example to illustrate tagged filter in Query Select.

Editors

Stephen Schleimer, Cisco Systems, Incorporated
Christina Shaw, Compaq Computer Corporation
For the DMTF Events Working Group

This document can be located at: htwww.dmtf.org/members/tdc/wg-events/archive/Event_v25.doc

Glossary

Term	Definition
Alert / Event Service	A service that provides higher level alert or event Indication processing such as a persistent record of events of interest, unique Indication identity, guaranteed delivery, etc.
AlertIndication	A particular subclass of CIM_ProcessIndication that provides specific properties and may include CIM_InstIndication and CIM SourceObject data.
Client	A process that creates subscriptions and associates them with Indication filters and Handlers for the purpose of receiving indications.
Delivery	The process of transporting one or more indications to a subscriber. The method of delivery is determined by the subscription process via the association between the Indication Filter and the Handler instances.
Event	An occurrence of a phenomenon of interest. Events are not published in CIM nor can a client subscribe to them.
Filter	A Filter is an object that defines the conditions that detect to trigger the occurrence

	of indications as directed by the query property of the filter.
Handler	A Handler is an object that collects arbitrary Indications from Filters. A Handler may format Indications to a particular syntax, and resend them to a sink or it may send email, perform paging, or other associated actions like launching a process.
Indication	The active representation of the occurrence of an event in CIM for which a subscription exists. Specific delivery parameters for indications are included in the subscription process. Indications are published (see Publication). Indications are subscribed to through Filters that select them.
Projection	That set of data extracted from an object (or objects) defined as components of an indication.
Publication	Registration of Indication definition. Certain indications are defined as part of CIM others are defined by users. Querying the CIM repository allows clients to discover which phenomena may give rise to an indication.
Query	A description of the interesting characteristics of one or more indications. A query is a component of a Filter.
Subscription	A declaration of interest in one or more streams of indications
Trigger	A description of a change of interest. Also considered to be the proximal cause of an event. "The operational state of the line went from up to down. A line down event occurred. A line down Indication might be generated."

1. Introduction

The Methods and Events Working Group Charter:

> *The working group will review the syntax and meta schema for method declarations and ensure that it is correct and implementable in the light of the proposed XML mapping. It will use the XML mapping as a basis for proposing a method invocation based on or compatible with the XML transport protocol.*
>
> *The working group will also review the indications defined in the current meta schema. It will*

propose (or clarify) MOF extensions necessary to define indications and propose filtering and aggregation mechanisms consistent with these. An event registration mechanism compatible with the method invocation mechanism will also be provided.

1.1 Scope

This document describes CIM Indications and how they are used to communicate occurrences of events in the CIM. The CIM Indication Schema includes the Indication and Subscription class hierarchies. Indications carry salient event information for CIM objects as well as entities not modeled in CIM. The Subscription class hierarchy provides the mechanism for client applications to subscribe to specific indications (e.g. event registration). The associated Filter contains a Query parameter to specify the source instance parameters to be returned with the indication. Companion documents, "CIM Indication MOF" [2] and "Specification for CIM Operations over HTTP" [3] provide the MOF and xmlCIM encoding specifications.

This specification does not include event aggregation at this time. While it is recognized that this is a core issue, the working group felt it would be useful to deliver the base framework (described in this document) in CIM 2.5. In this way preliminary implementations that demonstrate the model's viability may be achieved sooner. It is intended that aggregation can be supported by the primitives defined herein and will be addressed in the next version of this specification.

1.2 Terminology

The key phrases and words, MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY and OPTIONAL in this document are to be interpreted as described in RFC 2119 [4]

1.3 Overview

The subject of events is complex as it covers a wide range of topics and scenarios. An event is typically assumed to be a change in the state of the environment or a record of the behavior of some component of the environment. For example, the state property for a service may go from Stopped to Started, indicating that the service is now started. Or a device may be added to a machine resulting in a plug and play Indication ultimately notifying the operating system that the device is present and should be configured with settings and drivers in order for it to be usable.

An event may be a pervasive incident that occurs infrequently such as a system re-boot or it may reflect very small scale, frequently occurring incidents, for example mouse-clicks. Many things can be affected. The consequences of an event can last a long time.

The way that events are dealt with may also vary enormously. Some events may require immediate action on the part of the observer. For example an 'out of disk space' event on a web server may require immediate action to make disk space available. Some events may only be of interest at a later time. An example of the interest at a later time is a 'bandwidth utilization on an interface' event that a billing application deals with only during nightly billing reconciliation.

1.4 Representation

In the context of this specification, the concrete Indication of the occurrence of an event is represented by an instance of the CIM_Indication class. To improve readability, the redundant use of the 'CIM_' preface will often be omitted in the text when discussing the members of the CIM_Indication class hierarchy.

Types of indications (representing different types of events) are denoted by Indication subclasses. These include:

- InstIndication for modeling CIM life cycle events; instance creation, deletion, modification, method invocation and read access
- ClassIndication for CIM schema life cycle events; class creation, deletion and modification.
- ProcessIndication for alert notifications associated with objects that may or may not be completely modeled in CIM or do not correspond to a simple life cycle event; like low-level instrumentation alerts, DMI alerts, SNMP traps and TMN events

Indication classes are described in section 2.

Instances of indications cannot be enumerated because they are transient objects (not guaranteed to have persistence). Indications are only received after subscribing to them. They cannot be retrieved through enumeration or ordinary query processing in a CIM Object Manager. This was considered to be a necessary design constraint to ensure lightweight Indication processing. It is intended that specialized applications will be devised which will add more sophisticated post processing computations to an Indication stream on behalf of its clients. Such post processing may include creating unique keys, persistence, aggregation and correlation. Aggregation and correlation will be modeled in a future CIM Indication specification version.

Thus indications in this model are not guaranteed to be uniquely identifiable. For example, if the time and date stamp provided is insufficient to distinguish between instances of, say, InstModification, then those instances are not identifiable unless the provider chooses to add an identifier or a logging entity exists which adds this as a post processing function.

1.5 Publication and Subscription

A fundamental idea underlying the CIM approach to the representation of indications is the separation of Indication publication and Indication subscription. The *publication* of an Indication is accomplished using the same mechanism used for the publication of any other data in CIM; that is, through the declaration of classes and properties. Publication of events also implies the creation of IndicationFilter instances.

A Subscription is expressed by the creation of an IndicationSubscription association instance that references an IndicationFilter (a Filter) instance, and an IndicationHandler (a Handler) instance. A Filter contains the query that selects an Indication class or classes. The size and complexity of the result delivered to the subscriber is dictated by the query.

The CIM Object Manager is designed to process queries on behalf of managed object providers. However, it is intended that CIM managed object providers may be designed (although not required) to handle ad hoc Filter queries directly. The precise mechanism whereby providers communicate their respective capabilities to an Object Manager is currently being defined (within the Interoperability work group) and targeted for CIM v2.6.

Notifications of Filtered events are delivered as instances of the Indication class. A Handler subclass instance is used to specify the destination that is to receive the associated Indication stream. This version of the CIM

Indication specification defines the IndicationHandlerXMLHTTP subclass that is used to deliver indications to clients over HTTP and encoded as cim/XML. Other protocols may be defined in the future to support point to point protocols, multi-cast delivery, email, paging, as well as associated actions like launching a process. Thus the intent in naming this class IndicationHandler (rather than IndicationDelivery) is meant to convey that handling an Indication can require more than delivery.

It is the intention of this specification that Indication instances are created only if there is an instance of IndicationSubsription which associates the event Filter that can generate the Indication with the Handler subclass that defines the precise handling mechanism.

If there is no provider capable of generating the requested Indication the instantiation of the IndicationSubscription SHOULD fail. Likewise, if there is no instance of the requested IndicationHandler the instantiation of the IndicationSubscription SHOULD fail.

The Modeling Events Section describes the properties and semantics of the CIM_Indication and Subscription class hierarchies. ____

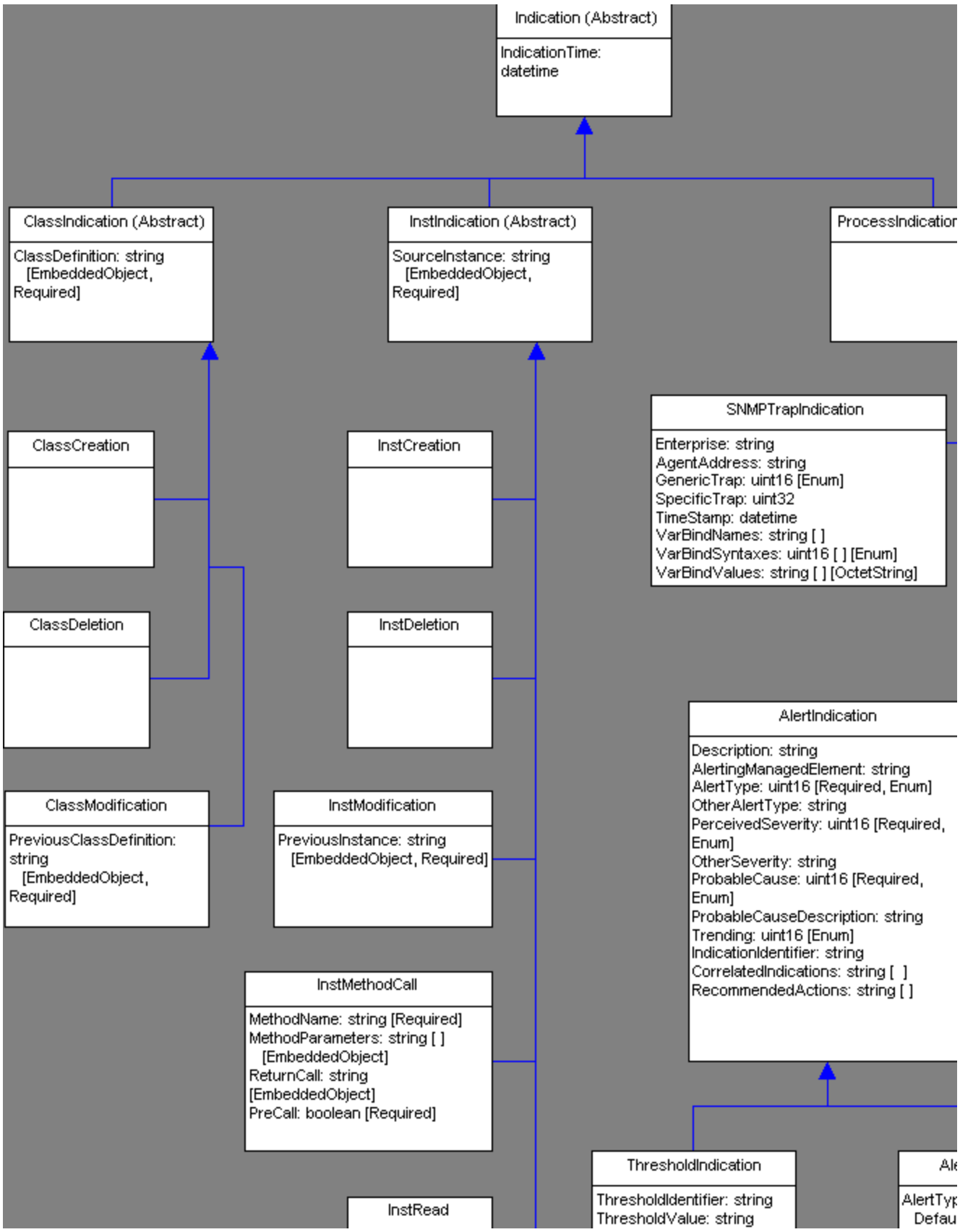
1.6 Namespace and Subscription Management

Indications and their properties are to be interpreted in the context of a single namespace. The IndicationFilter.SourceNameSpace parameter is used to denote the namespace in which the event that triggered the Indication occurred. This allows creation of all subscriptions in a single CIM namespace even if the events of interest originate from a different namespace. This schema allows creating and examining all Filters in a single namespace regardless of the origin of the events. In addition, since Filter and Handler are subclasses of CIM_ManagedElement, Filters and Handlers can be managed by higher=level services.

2. Modeling Events

The event model has to meet two conflicting requirements. First, it has to be extensible allowing schema designers to add new types of indications (events) in arbitrary ways reflecting unforeseen Indication structure and usage. Second, it has to provide a basis for event analysis and applications that interpret the event flow for aggregation, correlation and throttling purposes without the application having to be aware of the full range of event types implied by the first requirement. As mentioned, while it is intended that this specification support aggregation, correlation and throttling, the exact mechanism for doing this is deferred to a future version.

In this proposal, the occurrence of an event is represented as an instance of the Indication class. Figure 1 illustrates the proposed Indication type hierarchy that addresses the requirements for event processing in CIM as enumerated in the requirements document [6].


Indication (Abstract)
IndicationTime: datetime
ClassIndication (Abstract)
ClassDefinition: string [EmbeddedObject, Required]
InstIndication (Abstract)
SourceInstance: string [EmbeddedObject, Required]
ProcessIndication
SNMPTrapIndication
Enterprise: string
AgentAddress: string
GenericTrap: uint16 [Enum]
SpecificTrap: uint32
TimeStamp: datetime
VarBindNames: string [ ]
VarBindSyntaxes: uint16 [ ] [Enum]
VarBindValues: string [ ] [OctetString]
ClassCreation
InstCreation
ClassDeletion
InstDeletion
AlertIndication
Description: string
AlertingManagedElement: string
AlertType: uint16 [Required, Enum]
OtherAlertType: string
PerceivedSeverity: uint16 [Required, Enum]
OtherSeverity: string
ProbableCause: uint16 [Required, Enum]
ProbableCauseDescription: string
Trending: uint16 [Enum]
IndicationIdentifier: string
CorrelatedIndications: string [ ]
RecommendedActions: string [ ]
ClassModification
PreviousClassDefinition: string [EmbeddedObject, Required]
InstModification
PreviousInstance: string [EmbeddedObject, Required]
InstMethodCall
MethodName: string [Required]
MethodParameters: string [ ] [EmbeddedObject]
ReturnCall: string [EmbeddedObject]
PreCall: boolean [Required]
ThresholdIndication
ThresholdIdentifier: string
ThresholdValue: string
InstRead

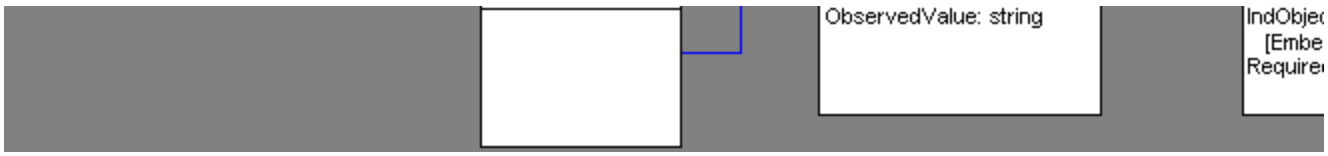


Figure 1 CIM_Indication type hierarchy

2.1 CIM_Indication Hierarchy

An abstract Class, CIM_Indication is the base class from which all other indications inherit. CIM_Indication has the following property:

- datetime IndicationTime

The value of the IndicationTime is as close as possible to the time of the underlying event. If the time of the event cannot be determined the IndicationTime SHALL be NULL.

Note: Due to implementation limitations IndicationTime is not guaranteed to be accurate enough to infer the order of events.

As mentioned above, Indication classes do not themselves contain any keys because they are by design not enumerable. It would be too much of a burden for the Object Manager to store and process queries over the vast number of Indication instances expected to be generated even in an average implementation (about 1500 objects). This level of processing can be accomplished more efficiently outside the Object Manager by an entity that subscribes to all or a particular subset of indications and then stores and processes them on behalf of its client entities.

The immediate subclasses of CIM_Indication are:

- CIM_ClassIndication

- CIM_ InstIndication

CIM_ProcessIndicationClassIndication and InstIndication (LifeCycle indications) provide a convenient intuitive mechanism for clients to subscribe to indications triggered by real world events that change the state of CIM Objects (InstIndication) or the CIM Schema (ClassIndication). That is events triggered by the creation, deletion, or modification of class definitions or instances, or events triggered by a read access or method invocation on instances. Using ad hoc queries in IndicationFilters, clients can set up any number of Indication scenarios that need not have been 'programmed' into the managed object instrumentation.

ProcessIndications allow clients to subscribe to a wide variety of operational notifications (AKA alerts and traps) in the same simple manner. While it can be successfully argued that almost any LifeCycle Indication can be used to provide operational notifications, there are two fundamental reasons for modeling operational notifications as a separate hierarchy.

Firstly, operational notifications tend to be the result of error, warning or other abnormal system conditions. As such, it is important to convey the associated conditions that prevailed that gave rise to the anomalous condition as well as the perceived severity and consequences. Thus parameters like, PerceivedSeverity, ProbableCause, AlertType, and RecommendedAction are an fundamental part of a standard notification scheme.

However, adding these parameters to the LifeCycle Indication hierarchy would burden the general case, normal LifeCycle transformations, with a specific case, anomalous potentially system threatening transformations. That is each LifeCycle class would need to contain the parameters: PerceivedSeverity, ProbableCause, AlertType, and RecommendedAction, even though they are only meaningful when the triggering event has potentially adverse or meaningful operational consequences.

The second reason to separate the two, is to give low level managed objects a simple mechanism to indicate an exceptional condition. Consider an operating system kernel that needs to indicate an exception, perhaps due to a bad parameter in a context switch table. While the operating system is certainly modeled as a CIM object, it is not possible at this level of execution (context switch) for the OperatingSystem provider to recognize the triggering event that would give rise to the LifeCycle Indication even if it modeled the parameter "ContextSwitchTable".

Operating systems and other managed systems (Storage, Network, Application, etc.) are typically instrumented to issue operational notifications. ProcessIndication classes make it very easy to convert the underlying system notifications into ProcessIndications. The consumer (client) for these indications might be the OperatingSystem provider or some other provider capable of correlating Process and LifeCycle Indications. Since the OperatingSystem provider has a greater view of the OperatingSystem object and its associated managed objects it is possible that it could provide more meaningful context than is possible with a single ProcessIndication.

The ability for instrumentation-level providers to publish clear, concise and relatively inflexible (perhaps even hard coded) notifications is a basic requirement [6]. General providers and some instrumentation can clearly do more like process Filters and support arbitrary queries. As noted, the precise mechanism whereby a provider communicates its ability to process Filters or support arbitrary queries will be specified by the Interoperability technical working group in a future CIM specification.

2.1.1 CIM_InstIndication

An abstract subclass of CIM_Indication denoting life cycle changes of instances (creation, deletion, modification) or method calls to instances defined in the CIM or reads of instances. InstIndication has the

following property:

- [EmbeddedObject, Required[1]] string SourceInstance

SourceInstance provides a snap shot of the properties of the source instance for which the event occurred. This content is the information provided by projection during the filtering process (described below). The representation of SourceInstance is described in [8]. See each of the subclasses for further discussion.

SourceInstance (and PreviousInstance of CIM_InstModification) represent two distinct concepts. When considered in the context of a Filter, SourceInstance (and PreviousInstance) represent states of the object that participated in the determination of the occurrence of an event. Thus, it makes sense, in the context of the Filter (specifically the query) to operate on SourceInstance (PreviousInstance) as though it were an object.

When considered in the context of an Indication (as a subscriber might see it), SourceInstance (PreviousInstance) is actually a set of properties as determined by the projection part of the query in the Filter (for example, the select clause of an SQL query). It is not an object and should not be thought of as an object instance. It is important to realize that both SourceInstance and PreviousInstance are known by the entity that recognizes the fact of the filtered event.

CIM_ InstIndication provides the following subclasses (representing five instance life cycle events):

- CIM_InstCreation
  - CIM_InstDeletion
- CIM_InstModification
- CIM_InstMethodCall
- CIM_InstRead

CIM_InstCreation

A concrete class. When an instance of an object is created, a creation event occurs. If appropriate (see 2.2), a CIM_InstCreation Indication is created. The created instance is recorded in the inherited attribute SourceInstance.

Note: Indications are not treated like other objects for the purpose of Indication generation. Thus, creation of an instance of CIM_InstCreation is not an event and can not be filtered for the purpose of creating an Indication stream.

CIM_InstDeletion

A concrete class. When an instance of an object is deleted, a deletion event occurs. If appropriate (see 2.2) a CIM_InstDeletion Indication is created. A projection of the deleted instance is recorded in the inherited attribute SourceInstance.

CIM_InstModification

A concrete class. When an instance of an object is modified, a modification event occurs. If appropriate (see 2.2), a CIM_InstModification Indication is created. A projection of the modified state is recorded in the inherited attribute SourceInstance. InstModification has the following property:

- [EmbeddedObject, Required] string PreviousInstance

The PreviousInstance property has the same properties as that of the inherited property, SourceInstance. PreviousInstance records a projection of the appropriate properties of the previous state of the modified object (as defined by the Filter criteria).

CIM_InstMethodCall

A concrete class. When an instance's method is called, a method call event occurs. If appropriate (see 2.2), a CIM_InstMethodCall Indication is created. A projection of the instance for which the method call is made is recorded in the inherited attribute SourceInstance. InstMethodCall has the following properties:

- [Required] string MethodName
- [EmbeddedObject] string MethodParameters
- string ReturnValue
    - [Required] Boolean PreCall

MethodName is the name of the method invoked.

MethodParameters are the parameters of the method, formatted as an EmbeddedObject (with a predefined class name of __MethodParameters.ReturnValue may be NULL, depending on the value of the PreCall property. When PreCall is TRUE, this property is NULL describing that there is no method return value (since the method has not yet executed). When PreCall is FALSE, ReturnValue contains a string representation of the method's return value.PreCall is a Boolean indicating whether the Indication is sent before the method begins executing (TRUE) or when the method completes (FALSE).

CIM_InstRead

A concrete class. When an instance of an object is read, a read event occurs. If appropriate (see 2.2), a CIM_InstRead Indication is created. The read instance is recorded in the inherited attribute SourceInstance. Of course, it is not usual to record read events in general. However, in certain security contexts, it may be necessary to note each time a certain object is read to provide an appropriate level of control. Recall that indications are generated for events only if appropriate.

2.1.2 CIM_ClassIndication

An abstract class. When schema changes are made, schema change events occur. The model provides Indication of these events with the CIM_ClassIndication class hierarchy. In parallel with the components of the InstIndication tree, ClassIndication has the three subclasses:

- CIM_ClassCreation

- CIM_ClassModification
- CIM_ClassDeletion

These correspond to schema creation, schema modification, and schema deletion events respectively. CIM_ClassIndication has a single property:

- [EmbeddedObject, Required] ClassDefinition

ClassDefinition is the definition of the class for which the Indication was created.

ClassModification has a single property:

- [EmbeddedObject, Required] PreviousClassDefinition

PreviousClassDefinition is the definition of the class before it was modified.

The Interoperability work group will define the representation of a ClassDefinition and PreviousClassDefinition.

2.1.3 CIM_ProcessIndication

An abstract superclass for specialized Indication classes, addressing specific changes and alerts published by instrumentation. ProcessIndications support instrumentation-level providers' requirement to publish clear, concise and (sometimes) relatively inflexible notifications. In many cases, this must be done using minimum system requirements.

It is expected and desired that clients will use the InstIndication classes for most CIM instance related events; e.g. lifecycle events create, delete, modify, method call and read.However, it is not always convenient or appropriate to model events in a system through life cycle events even though it may always be possible to do so. For example, when multiple independent finite state machines are communicating state transitions (as in a distributed PBX, for example) it is preferable to state explicitly that a particular state change has occurred rather than implicitly as the consequence of the change of an attribute of an object. CIM_ProcessIndication is created as a super class for such explicit indications.

NOTE: Triggers causing ProcessIndications may be outside of the CIM. It is generally recommended, to the extent possible, that the event behavior of a system be explicitly modeled using CIM instance and instance property changes.

CIM_ProcessIndication has the following subclasses:

- SNMPTrapIndication
- AlertIndication

As noted, rather than burden the general purpose InstIndications lifecycle classes with information that pertains only in special circumstances it was decided to model alerts as a subclass of ProcessIndication.

Thus, CIM_AlertIndication is explicitly designed to model alert and or error notifications in a CIM system. A subclass of CIM_AlertIndication, CIM_AlertInstIndication, provides a mechanism to integrate other domain notifications with CIM_InstIndications.

ProcessIndications are described in the following sections.

CIM_SNMPTrapIndication

A concrete class for mapping an SNMP Trap to CIM based on the IETF RFC 1157. The usefulness of this class is to describe common trap semantics. Naturally, a complete understanding of any trap data received relies on the Indication recipient having access to the sender's MIB. Understanding can be improved by mapping the SNMP managed object domain to CIM.

CIM_SNMPTrapIndication contains the following parameters:

- string Enterprise
- string AgentAddress
- [Enum] uint16 GenericTrap
- uint32 SpecificTrap
- datetime TimeStamp
- string VarBindNames[]
- uint16 VarBindSyntaxes[]
- string VarBindValues[]

Enterprise (based on PDU.IETF|RFC1157-TRAP-PDU.enterprise) describes the type of object generating the trap.

AgentAddress (based on PDU.IETF|RFC1157-TRAP-PDU.agent-addr) provides the address of the object generating the trap.

GenericTrap (based on PDU.IETF|RFC1157-TRAP-PDU. generic-trap) is an enumerated value that describes the generic trap type. The following values are defined:

- 0 - coldStart trap signifies that the sending protocol entity is reinitializing itself such that the agent's configuration or the protocol entity implementation may be altered.
- 1 - warmStart trap signifies that the sending protocol entity is reinitializing itself such that neither the agent configuration nor the protocol entity implementation is altered.
- 2 - linkDown trap signifies that the sending protocol recognizes a failure in one of the communication links represented in the agent's configuration. The Trap-PDU of type linkDown contains as the first element of its variable-bindings the name and value of the ifIndex instance for the affected interface.
- 3 - linkUp trap signifies that the sending protocol entity recognizes that one of the communication links represented in the agent's configuration has come up. The Trap-PDU of type linkUp contains as the first element of its variable-bindings, the name and value of the ifIndex instance for the affected interface.
- 4 - authenticationFailure trap signifies that the sending protocol entity is the adressee of a protocol

message that was not properly authenticated. While implementations of SNMP must be capable of generating this trap, they must also be capable of suppressing the emission of such traps via an implementation-specific mechanism

- 5 - egpNeighborLoss trap signifies that an EGP neighbor for whom the sending protocol entity was an EGP peer has been marked as down and the peer relationship no longer pertains. The Trap-PDU of type egpNeighborLoss contains as the first element of its variable-bindings, the name and value of the egpNeighAddr instance for the affected neighbor.
- 6 - ? enterpriseSpecific? trap signifies that the sending protocol entity recognizes that som enterprise-specific event has occurred. The specific-trap field identifies the particular trap which occurred.

? SpecificTrap? (based on PDU.IETF|RFC1157-TRAP-PDU.specific-trap) identifies the specific trap code

? TimeStamp? (based on PDU.IETF|RFC1157-TRAP-PDU. time-stamp) is the time elapsed between the las (re) initialization of the managed entity and the generation of the trap.

? VarBindNames? array (based on PDU.IETF|RFC1157-TRAP-PDU.variable-bindings) contains Object namin information (an ? OID?) from the 'variable binding' portion of the Trap. This array is correlated with th ? VarBindSyntaxes? and ? VarBindValues? arrays. Each entry is related to the entries in the other arrays, that located at the same index. In this way, the variable binding's name/syntax/value ? tuple? can be constructe

? VarBindSyntaxes? array (based on PDU.IETF|RFC1157-TRAP-PDU.variable-bindings) contains the Objec syntax information (defined as an enumerated value) from the 'variable binding' portion of the Trap. This array is correlated with the ? VarBindNames? and ? VarBindValues? arrays. Each entry is related to the entries in other arrays, that are located at the same index. In this way, the variable binding's name/syntax/value ? tuple? c be constructed.

? VarBindValues? array (based on PDU.IETF|RFC1157-TRAP-PDU.variable-bindings) contains an ? OctetSt representing object value information from the 'variable binding' portion of the Trap. This array is correlated with the ? VarBindNames? and ? VarBindSyntaxes? arrays. Each entry is related to the entries in the other ar that are located at the same index. In this way, the variable binding's name/syntax/value ? tuple? can b constructed.

?CIM_AlertIndicatio

A concrete class for CIM alert indication. ? AlertIndication? is a standard notification Indication that contain information about an alerting situation (like ? PerceivedSeverity? , ? ProbableCause? , ? RecommendedAction Trending) that may or may not be modeled by CIM instances. ? AlertIndications? that are not dependent on CIM instance or do not wish to include ? InstIndication? data SHOULD use ? AlertIndication? or subclass it h is expected that ? DMI? recast indications will be subclassed from he See Appendix C, Mapping Existing Models Into CIM.

? CIM_AlertIndication? contains the following parameter

- string Description
- string ? AlertingManagedElemen
- [Required, ? Enum?] uint16 ? AlertT
- string ? OtherAlertType
- [Required, ? Enum?] uint16 ? PerceivedSever
    - string ? OtherSeverit
- [Required, ? Enum?] uint16 ? ProbableCa
- string ? ProbableCauseDescriptio
- [? Enum?] uint16 Trendi
- string ? IndicationIdentifie
- string ? CorrelatedIndications?
- string ? RecommendedActions?

Description (based on Recommendation.ITU|X733.Additional text) is a string that provides a short description of this alert.

? AlertingManagedElement? provides identifying information about the entity (instance) for which this Indicatio was generated. If the entity is modeled in the CIM Schema, this property contains the path of the instance encoded as a string parameter. If the entity is not a CIM instance, the property contains some identifying string that names the entity for which the Alert was generated.

? AlertType? (based on Recommendation.ITU|X733.Event type) is an enumeration that provides a primar classification of the indication. The following values are defined:

- 1 - Other State Change. The indication's ? OtherAlertType? property conveys its classification
    - 2 - Communications Alert. An Indication of this type is principally associated with the procedures and/or processes required to convey information from one point to another.
- 3 - Quality of Service Alert. An Indication of this type is principally associated with a degradation or errors in the performance or function of an entity.
- 4 - Processing Error. An Indication of this type is principally associated with a software or processing fault.
- 5 - Device Alert. An Indication of this type is principally associated with an equipment or hardware fault.
- 6 - Environmental Alert. An Indication of this type is principally associated with a condition relating to an enclosure in which the hardware resides, or other environmental considerations.
- 7 - Model Change. The Indication addresses changes in the Information Model and not the managed environment.
- 8 - Security Alert. An Indication of this type is associated with security violations, detection of viruses, and similar issues.

? OtherAlertType? is a string describing the Alert type when the ? AlertType? property value is set to 1,Other Change.

? PerceivedSeverity? (based on Recommendation.ITU|X733.Perceived severity) is an enumerated value tha describes the severity of the ? AlertIndication? from the ? notifier's? point of vThe following values are defined:

- 0 - Unknown SHOULD be used when the perceived severity is unknown.
- 1 - Other, by CIM convention, SHOULD be used to indicate that the perceived severity value can be

found in the ? OtherSeverity? property

- 2 - Information SHOULD be used when ? AlertIndication? is purely informationa
- 3 - Warning SHOULD be used when it's appropriate to let the user decide if action is needed.
- 4 - Minor SHOULD be used to indicate action is needed, but the situation is not serious at this time.
- 5 - Major SHOULD be used to indicate action is needed NOW.
- 6 - Critical SHOULD be used to indicate action is needed NOW and the scope is broad (perhaps an imminent outage to a critical resource will result).
- 7 - Fatal SHOULD be used to indicate an error occurred, but it's too late to take remedial action.

? OtherSeverity? is a string describing the ? PerceivedSeverity? when the ? PerceivedSeverity? property valu 1,Other.

? ProbableCause? (based on Recommendation.ITU|X733.Probable cause) is an enumerated value that describe the probable cause of the situation which resulted in the ? AlertIndication? . The following values are defined

- Unknown
- Other
- Adapter Error
- Application Subsystem Failure
- Bandwidth Reduced
- Connection Establishment Error
- Communications Protocol Error
- Communications Subsystem Failure
- Configuration/Customization Error
- Congestion
- Corrupt Data
- CPU Cycles Limit Exceeded
- Dataset/Modem Error
- Degraded Signal
- Denial of Service Detected
- ? DTE-DCE? Interface Err
- Enclosure Door Open
- Equipment Malfunction
- Excessive Vibration
- File Format Error
- Fire Detected
- Flood Detected
- Framing Error
- Hardware Security Breached
- Humidity Unacceptable
- HVAC Problem
- I/O Device Error
- Input Device Error

- Invalid Access of Data Detected
- LAN Error
- Non-Toxic Leak Detected
- Local Node Transmission Error
- Login Attempts Failed
- Loss of Frame
- Loss of Signal
- Material Supply Exhausted
- ? Multiplexer? Probl
- Out of Memory
- Output Device Error
- Performance Degraded
- Power Problem
- Pressure Unacceptable
- Previous Alert Cleared
- Processor Problem (Internal Machine Error)
- Pump Failure
- Queue Size Exceeded
- Receive Failure
- Receiver Failure
- Remote Node Transmission Error
- Resource at or Nearing Capacity
- Response Time Excessive
- Retransmission Rate Excessive
- Security Credential ? MisMatc
- Software Error
- Software Program Abnormally Terminated
- Software Program Error (Incorrect Results)
- Software Virus Detected
- Storage Capacity Problem
- Temperature Unacceptable
- Threshold Crossed
- Timing Problem
- Toxic Leak Detected
- Transmit Failure
- Transmitter Failure
- Underlying Resource Unavailable
- Version ? MisMatch

? ProbableCauseDescription? is a string that provides additional information related to the ? ProbableCau

Trending (based on Recommendation.ITU|X733.TrendIndication) is an enumeration (unit 16) that provides information on trending. The following values are defined:

- Unknown
- Not Applicable
- Trending Up
- Trending Down
- No Change

? IndicationIdentifier? (based on Recommendation.ITU|X733.Notification identifier) is an identifier for th
? AlertIndication? . This property is similar to a key value in that it can be used for identification, when correlatin
? AlertIndications? (see the ? CorrelatedIndicatons? array). Its value should be unique as long as correla
? AlertIndications? are reported, but may be reused or left NULL if no future ? AlertIndications? will referenc
their ? CorrelatedIndicatons? arra

? CorrelatedIndicatons? (based on Recommendation.ITU|X733.Correlated notifications) is list o
? IndicationIdentifiers? whose notifications are correlated with (related to) this on

? RecommendedActions? (based on Recommendation.ITU|X733.Proposed repair ? actions)is? an array of fre
descriptions of the recommended actions to take to resolve the cause of the notification.

?CIM_ThresholdIndicatio

A concrete subclass of ? CIM_AlertIndication? carrying additional threshold information related to th
notification. This subclass is used when one of the ? ProbableCauses? is set to 53, Threshold Crosse
? CIM_ThresholdIndication? has the following propertie

- string ? ThresholdIdentifie
- string ? ThresholdValu
- string ? ObservedValu

? ThresholdIdentifier? (based on Recommendation.ITU|X733.Threshold information) is a string describing th
threshold or naming the property that represents the threshold, if modeled in the CIM hierarchy. In the latter
case, the value should be written as <schema name>_<class name>.<property name>.

? ThresholdValue? (based on Recommendation.ITU|X733.Threshold information) is a string holding the curren
value of the threshold. This is modeled as a string for universal mapping, similar to the ? CIM_Sensor? propertie
in the Device Model.
? ObservedValue? (based on Recommendation.ITU|X733.Threshold information) is a string holding the curren
reading value that exceeds the threshold. This is modeled as a string for universal mapping, similar to the
? CIM_Sensor? properties in the Device Mode

?CIM_AlertInstIndicatio

A concrete subclass of ? CIM_AlertIndication? that can return a ? CIM_InstIndication? instance when reques
The embedded ? InstIndication? instance provides the connection between the ? AlertIndication? instance and
? InstIndication? instance containing the projection of the target CIM object instance that gave rise to th
? AlertIndication? . CIM domain mapped models (e.g. ? DMI? and ? TMN?) may use this class to r
? AlertIndications? with target CIM instance Providers capable of high level system analysis can use this
subclass to provide additional information (? PerceivedSeverity? , ? ProbableCause? , ? RecommendedActions?

with CIM object life cycle events.

? CIM_AlertInstIndication? has the following propertie

- [Override ("? AlertType? ")] uint16 ? AlertType?
- [? EmbeddedObject? , Required] ? IndObj

The ? AlertTyp is always set to 7, “Model Change”, for ? AlertInstIndicationThis is done because: 1) the primary purpose of ? AlertInstIndication? is to add Alert data to a ? LifeCycle? Indication; and 2) ? LifeC Indications deal with 'model changes'.

? IndObject? contains the ? CIM_InstIndication? that is part of this indicAs always, only the properties selected by the Indication Filter are included. The ? InstIndication? elaborates the meaning of the ? AlertIndica with change type (? CIM_InstIndication? class) and object properties (? SourceInstance? and ? PreviousInst values, depending upon the type of the embedded indication).

2.2 Subscription Hierarchy

The Subscription hierarchy specifies how clients subscribe to and receive indications.

Two classes and an association comprise the Subscription hierarchy:

- ? CIM_IndicationFilte
- CIM_ ? IndicationHandle
- ? CIM_IndicationSubscription? (associatio

? CIM_IndicationFilter? defines a stream of indications, ? CIM_IndicationHandler? defines how and where deliver the Indication stream, and ? CIM_IndicationSubscription? associates a Filter instance with a Handle instance for a particular Indication stream.

Clients instantiate a Handler to express interest in receiving indications from one or more Filters. Clients create an instance of ? IndicationSubscription? to associate a Filter to a HandleThis is done for each Filter from which indications are to be delivered. The Filter Query property allows the client to specify only those properties in the source object that it desires to be included in the Indication when the event occurs.

NOTE: CIM event indications are WYSIWYG – What you subscribe (to) is what you get.

Figure 2 illustrates the Subscription hierarchy.

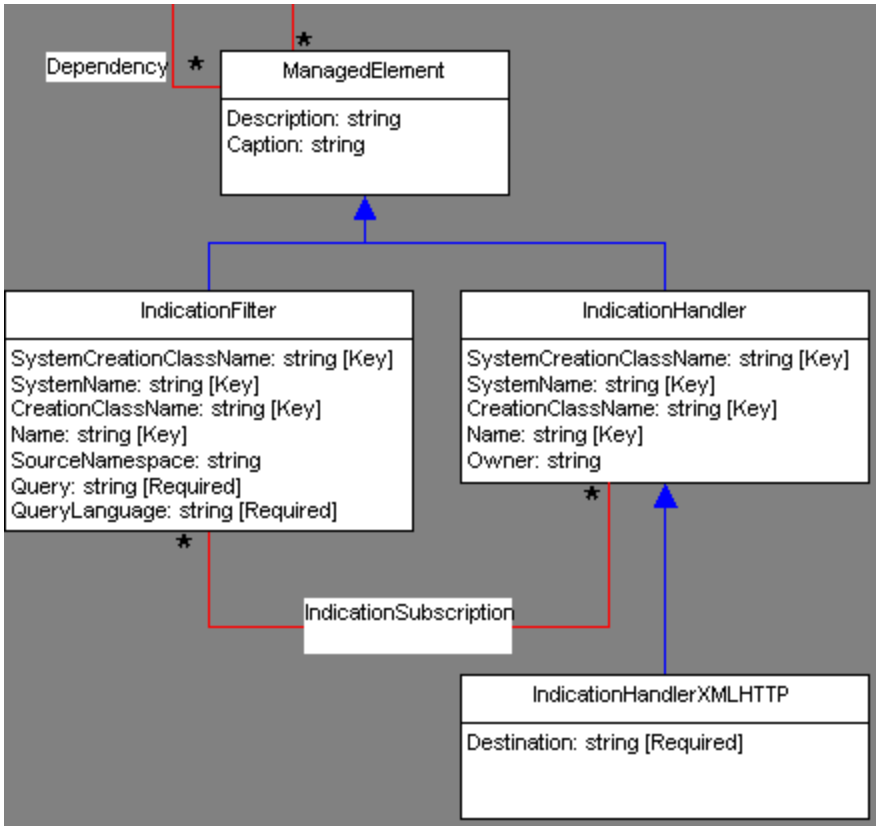


Figure 2 Subscription Hierarchy

Circumstances under which indications are created

Events occur continuously. Indications SHOULD only be created when there are instances of a Filter and a Handler associated by a subscription. For example, if there is no Filter selecting read indications for a particular class OR there are no Handlers bound to Filters for read indications for a particular class, then no indications for read access to instances of that class SHOULD NOT be generated. It is recognized that some providers will not have the ability to so distinguish.

The Interoperability technical working group will define the mechanism whereby the fact of Filter publication and/or subscriber binding may be communicated to providers to aid providers in adhering to this requirement.

Precision

There is no implication regarding when, in the real world the event took place. There is not even any implication that other similar events did not take place before, during, or after the detection of this particular event. For example, consider a Filter that recognizes status changes from "ok" to "degraded". Now suppose that for some real world object status changes from ok to degraded and back again in 100 milliseconds.

Consider various detection strategies. First, imagine that the process that changes the value of the status variable from ok to degraded also evaluates the Filter criterion. Then we may assume that the generation of indications would be reliable. Now, imagine that there is a monitor of the status variable that polls the value every 50 milliseconds and performs the Filter criterion evaluation. We may assume in this case as well that the generation of indications would be reliable. Now, suppose that we again have an external monitor, but that the polling interval is 100 seconds. In this case we would expect that Indication generation would be unreliable. We assume that the reliability of the generation of indications is a function of the resolution of the observation of the events for which indications are generated.

The Interoperability technical work group will provide the mechanism that allows a provider to communicate its precision capabilities to an Object Manager and allows an Object Manager to pass this information along to an interested client. Similarly, as noted above, the Interoperability technical work group will define the mechanism that enables providers to specify which indications they are capable of providing including whether or not they are capable of processing Filters or parsing queries.

2.2.1 ?CIM_IndicationSubscriptio

A concrete association class. A client creates a ? CIM_IndicationSubscription? to direct a flow of indication from a particular Filter and Handler. The flow is directed from the referenced Filter to the referenced destination or process in the Handler. A client may have many subscriptions linking many Filters to many Handlers.

? CIM_IndicationSubscription? contains the following properties

- [Key] ? CIM_IndicationFilter? REF Filt
- [Key] ? CIM_IndicationHandler? REF Handl

Filter defines the criteria and data of the possible Indications of this subscription.

Handler defines how possible Indications of this Subscription are handled; e.g. HTML delivery mechanism, email, pager, process invocation, etc.

2.2.2 ?CIM_IndicationFilte

A concrete subclass of ? CIM_ManagedElement? . ? CIM_IndicationFilter? defines the criteria for generating Indication and what data should be returned in the indication. It is derived from ? CIM_ManagedElement? t allow modeling the dependency of the Filter on a specific service.

Filter publication occurs in the same fashion as any CIM object; that is through the existence of instances of ? IndicationFilters? in the CIM repositorAny entity (Indication producers or consumers) may create an instance of a ? CIM_IndicationFilter? , informing CIM clients of the kind of indications to which they ma subscribe. An Indication SHOULD **only** be generated when the Filter Query condition evaluates to TRUE AND there is a subscription associated with the Filter. ? FilterIndication? has the following propertie

- [Key] String ? SystemCreationClassNam
- [Key] String ? SystemNam
- [Key] String ? CreationClassNam
- [Key] String Name
- [Required] ? SourceNameSpac
- [Required] String Query
- [Required] String ? QueryLanguag

? SystemCreationClassName? provides a System's ? CreationClassName? . A Filter is defined in the context o ? CIM_System? , where it is hosted or to which it applies. In a future release, a weak relationship will be explicit added to the model. This is not being done now to allow further refinement of the Filter definition and its inheritance tree. Keys are being defined now to allow the class to be instantiated.

? SystemName? provides a System's Name. A Filter is defined in the context of a ? CIM_System? , where it hosted or to which it applies. In a future release, a weak relationship will be explicitly added to the model. This is not being done now to allow further refinement of the Filter definition and its inheritance tree. Keys are being defined now to allow the class to be instantiated.

? CreationClassName? indicates the name of the class or the subclass used in the creation of an instance. Whe used with the other key properties of this class, it allows all instances of this class and its subclasses to be uniquely identified.

Name is the name of the Filter.

? SourceNamespace? is the path to a local namespace where the indications originate. If NULL, the namespac of the Filter registration is assumed.

The Query property defines what indications ranging over what objects should return which properties to a subscriber. The Query expression defines the ? condition(s?) under which Indications will be generated. For some Indication classes, the query expression may also define the instance properties to be copied to the ? CIM_InstIndication's? ? SourceInstance? and ? PreviousInstance? properties. Query language semantics in projection (e.g., Select), range (e.g., From) and predicate (e.g., Where).

The ? QueryLanguage? parameter is the language in which the query is expresse? QueryLanguage? indicate how to interpret the Query. ? QueryLanguage? semantics MUST include projection (e.g. Select), range (e.g From) and predicate (e.g. Where). The WBEM Query Language [7] MUST support instance property projection, that is a mechanism to select particular properties defined in the indication class to be included in the indication object. The projection may include static values that can be for the purpose of tagging the indication objects. In addition the WBEM Query Language MUST support the ability to project meta data such as object path and instance class. The WBEM Query Language MUST also support the ISA operator.

Range specifies the Indication class to which a Filter applies. This implies a potential stream of indications. The predicate defines the conditions under which indications may be generated. All the clauses of the predicate are qualified by Indication properties. For example, ? SourceInstance? , ? PreviousInstance? , ? IndicationTime? all may be used in the predicate of the query. ? QueryLanguage? property includes vers information.

The specification of the query language is outside the scope of the Indication specification. The Indication specification does not restrict the ? CIM_IndicationFilter? to a single query language, however an implementatio

of the Indication specification MUST support the DMTF Query Language [7].

Filter Examples

Queries are used in Filters to refine Indication selection. The following are some examples of queries and their projections:

Filter Example 1 for ? InstIndicatio

An application interested in creating a Filter to track the creation of devices might create a Filter whose query was the following:

```
SELECT      *
FROM        ? CIM_InstCreatio? ?
WHERE       ? SourceInstance?  ISA ? CIM_LogicalDev
```

The application would subscribe by creating a ? CIM_IndicationSubcription? instance containing the reference t the Filter (with the above query) and the desired Handler (for ? IndicationHandlerXMLHTT the Destination parameter contains the desired URL for delivery.)

The Interoperability technical work group will define how result is returned; format and content [3].

Filter Example 2 for ? InstIndicatio

An application interested in receiving an Indication when the managed device object, "? FooBar_LogicalNetworkDevice? ", status property changes to "degraded" might create a Filter with th following query:

```
SELECT      ? SourceInstance.Name? , ? SourceInstance.Descript
FROM        ? CIM_InstModificatio
WHERE       ? SourceInstance?  ISA ? FooBar_LogicalNetworkDevice?
            ? SourceInstance.Status?  == "Degraded" A

            ? PreviousInstance.Status?  <> ? SourceInstance.Sta
```

Because the application is looking for modifications to the state of a CIM object, it Filters on ? CIM_InstModification? Indication Because the application is only interested in status changes to 'Degraded' in devices modeled using the "? FooBar_LogicalNetworkDevice? " object, the **WHERE** clause restricts the object to that type with Status value of 'Degraded'. This query returns only the instance Name and Description of the network device that has become degraded.

The ‘PreviousInstance.status <> ? SourceInstance.statu clause is used here to illustrate how a client may suppress unwanted repetitive events after a threshold condition has been triggered.

The Interoperability technical work group will define how often and under what circumstances the condition is evaluated. The assumption that would usually be made is that it is evaluated when the ? CIM_InstIndication condition (modified, created, etc.) is satisfied. This would then deal with the problem of an unchanging state causing multiple Indications [3].

Filter Example 3 for ?AlertIndicatio

Subscribe to events where ?PerceivedSeverity? is "Major" and the quality of service of the managed object i trending toward critical.

```
SELECT    ? AlertingManagedElement?  , ? ProbableCause? , ? ProbableCauseDescripti
          ? RecommendedAction
FROM      ? CIM_AlertInstIndicatio
WHERE     ? PerceivedSeverity? “Major” AND
          ? AlertType?  == "Quality of Service" A
          Trending  == "Trending Up"
```

This query returns the ?AlertingManagedElement? containing the path of the alerting Cobject (or other Domain identifying string if it is not a CIM object), an enumerated value of the probable cause of the alerting situation to support automated programmatic handling and a textual description of the probable cause and the recommended actions for the human operator to view. Other variations of this query might contain the ?AlertIdentifier? (used to identify this instance) and the ?CorrelatedAlerts? property that contains an array of ot ?AlertIndications? related to this one, thus allowing subscribers to correlate multiple ?AlertIndicatio

Filter Example 4 for ?SNMPTrapIndicatio

Subscribe to all traps from Compaq Computer Corporation devices signaling a link down error.

```
SELECT    ? AgentAddres
FROM      ? CIM_SNMPTrapIndicatio
WHERE     Enterprise == “1.3.6.1.4.1.232” AND
          ? GenericTrap?  == "Link Dow
```

The query returns the address of the object generating the trap alert (as defined by ? IETF RFC1157-TRAP-PDU.agent-addr). A subscriber capable of processing ? SNMP? ? MIBs? in genetal Compaq Computer Corporation ? MIBs? in particular could have subscribed to the unprocessed contents of th trap by Selecting the ? SNMPTrapIndication.VariableBindings? propert

Filter Example 5 for ?AlertIndication? with embedded ?InstIndicat

Subscribe to events where severity is "Critical" due to a change in the temperature in any logical device.

SELECT *

FROM ? CIM_AlertInstIndicatio

WHERE ? PerceivedSeverity? "€ritical" AND

? IndObject? ISA ? CIM_InstModificaAND

? IndObject.SourceInstance.Temperature? > 80 A

? IndObject.SourceInstance? ISA ? CIM_LogicalNetworkDevice?
? IndObject.PreviousInstance.Temperature? <=

This query triggers when a CIM ? LogicalNetworkDevice? temperature exceeds 80 and the condition has "Critical" severity. The WHERE clause phrase 'IndObject ISA ? CIM_InstModificatio' asserts that alert indications from this Filter will only be generated for ? CIM_InstModificationThe second ISA asserts that only modifications to ? CIM_LogicalNetworkDevices? will be examine'? IndObject.PreviousInstance.Temperature <= 80' ensures that this Indication is only triggered when the temperature of the device had been less than or equal to 80 and is now greater than 80. This clause also insures that any additional changes to temperature above the 80 will NOT cause additional indications of this kind. The projection, Select *, returns all properties of the ? AlertInstIndication? , all the properties of the embedded ? InstModification? and all the properties (bef and after) of the CIM object which triggered the ? InstModification? IndicatioAs usual, the Select clause can be changed to restrict the return parameters as desired. Any ? InstIndication? can be substituted fo
? InstModification

2.2.3 ?CIM_IndicationHandle

An abstract super class for classes that represent how an event is to be handled. This may define a destination for delivering indications and a delivery protocol or it may define a process to invoke.
? CIM_IndicationHandler? is a subclass of ? CIM_ManagedElement? to allow modeling the dependency of Handler on a specific service. ? IndicationHandler? has the following propertie

- [key] string ? SystemCreationClassNam
- [key] string ? SystemNam
- [key] string ? ClassCreationNam
- [key] string Name
- string Owner

? SystemCreationClassName? is a System's ? CreationClassName? . The Handler is defined in the context o

? CIM_System? , where it is hosted or to which it applies. In a future release, a weak relationship will be explicit added to the model. This is not being done now to allow further refinement of the Handler definition and its inheritance tree. Keys are defined now to allow the class to be instantiated.

? SystemName? is a System's Name. The Handler is defined in the context of a ? CIM_System? , where it is h or to which it applies. In a future release, a weak relationship will be explicitly added to the model. This is not done now to allow further refinement of the Handler definition and its inheritance tree. Keys are defined now to allow the class to be instantiated.

? CreationClassName? indicates the name of the class or the subclass used in the creation of an instance. Whe used with the other key properties of this class, it allows all instances of this class and its subclasses to be uniquely identified.

Owner may be assigned the name of the creating entity of this Handler.

?CIM_IndicationHandlerXMLHTT

There is a single CIM_ ? IndicationHandler? subclass, ? CIM_IndicationHandlerXMLHTo provide XML encoded delivery over HTTP. The XML over HTTP Handler has a single property:

- string Destination

Destination is a URL (Universal Resource Locator) to which HTTP/? cimXML? Indication messages are to b delivered. The scheme prefix is implied and is not required, but must be 'http:' if specified.

The Interoperability technical work group will specify the actual transport and encoding for delivery. However, the following should be minimally provided:

- Asynchronous transport and encoding mechanism for Indication delivery; minimally including HTTP/XML, but not excluding other kinds of encoding and transport.
- Security defining who may register for what. To support security, it may be appropriate to bind subscriber to user (CIM 2.3).
    - Quality of delivery attributes (guaranteed, best effort, etc)
- Specify the encoding of the [? EmbeddedObject?] Strin

3. References

[1] "Common Information Model (CIM) Specification", Version 2.2, DMTF, March 2000 http://www.dmtf.org/spec/cims.html

[2] "CIM Indication MOF", Version 2.5, DMTF, October 2000, http://www.dmtf.org/members/tdc/wg-events/archive/Event_v25.mof

[3] "Specification for CIM Operations over HTTP", Version 1.1 in progress, DMTF, Date TBD

[4] "Key words for use in ? RFCs? to Indicate Requirement Lev? Scott ? Bradner? , RFC 21? IET

[5] "CIM Indication Visio Diagram", Version 2.5, DMTF, October 2000

http://www.dmtf.org/members/tdc/wg-events/.admin/sitemgr.cgi/members/tdc/wg-events/archive/Event_v25.vsd
[6] "Events Requirements document", Version 6.0, DMTF, Dec. 2, 1999
http://www.dmtf.org/members/tdc/wg-events/.admin/sitemgr.cgi/members/tdc/wg-events/archive/EvtReqsMaster.
[7] "WBEM Query Language Draft", Version 2.4, DMTF, June 14, 2000,
http://www.dmtf.org/members/review/wip/DMTF-query/DSP0104.htm
[8] "EmbeddedObject Qualifier", CR526, DMTF, October 26, 2000
http://www.dmtf.org/members/tdc/wg-events/.admin/sitemgr.cgi/members/tdc/wg-events/archive/cr526.html

APPENDIX A
SUMMARY ? INTEROP? ? WG? ISS

Legend: S = status; Open, Closed
P = requirement priority; Base, Future
?Req?# = corresponding requirement item numb

Event Model Component	Issue / Discussion	?Req
M_Indicatio	Providers can be competent enough to create indications. The API between CIM OM and providers must allow providers to indicate what they are capable of providing.	61
M_IndicationFilter	How is precision information surfaced in the model; to clients, between OM and provider? Precision and efficient (polling) is dependent on OM – Provider interface. How does provider signal an indication?	28
M_IndicationSubscriptio	Subscription association between Filter and Handler classes. What is delivery protocol for indications?	52
scription Model	Filter creation will fail if OM or agents cannot support it. The binding protocol defines the mechanism for recognizing a query language. Model permits multiple query languages. Mechanism depends on OM – Client IF (? interOp If there is no provider for a Filter, then the subscription association instance creation should fail.	8.f
scription Model	SQL 97 Query language may be parsed into XML (SQL 97) representation. Desire an XML rendering of the query such that it does not need to be parsed again.- content is process ready by an XML parser.	2.m
scription Model	Spec needs to address how subscriber sets up subscription – What is the API to OM? ?InterOp? dependency. Examp Cisco creates collection of classes; Filters – buffer overflow; ?tivoli? mgt app displays buffer overflow indicatio Some	8.d

	mechanism creates a protocol request to CIM OM to request a Cisco buffer overflow (bind subscriber to Filter and provide destination). It is undefined which entity (OM or subscriber) creates objects. How? Method? Service? A Subscriber sets up a subscription by instantiating an instance of ?CIM_IndicationSubscription? with references to (instances of the desired Filter and Handler.	
scription Model	Define the mechanisms whereby the fact of Filter publication and/or subscription binding may be communicated to providers and subscribers.	Core
scription Model	Model does not permit multiple destinations in a single association. Requirement for OM to merge Filters and/or push Filter down to provider. Mechanism depends on OM - Provider IF ? Sol'n? may have Event Model implications Is this still and issue? The current model does support multicast.	Core
overy	How are ?Oms? discovered by clients and provider	Core
overy	How are providers discovered by ?Oms? and client	Core
very	MUST support distinct modes of delivery; minimally including HTTP. The transport of indications should not be restricted to HTTP/XML. Other transport options may be possible (such as ?channelized? transport through ?Tibco? Rendez which might not pass through the ?CIMOM?This would have to be handled via management and registration of providers with Object Managers.	
very	Different delivery guarantees such as assured, best effort, etc.	
esentation	How are class definitions represented so that a representation of a class may be selectively included in a ?CIM_ClassIndication? and its subclasse	
ing	Not every Indication may be generated by a CIM Object Manager. We should not require that disambiguating naming be ONLY a function of the CIM Object Manager. A provider may be capable of handling this. For interoperability group: how is the protocol specified such that this is possible? The ?CIM_IndicationFilter.SourceNamespac contains the path to a local namespace where the events originate. If NULL the namespace of the Filter registration is assumed.	

APPENDIX B
REQUIREMENTS ANALYSIS

This section summarizes the features proposed versus the requirements stated by the Events ?WG?. Thi proposal has avoided architectural issues where possible and has avoided a specification of a correlation or aggregation engine or any way of specifying correlation and aggregation. Most of the correlation and aggregation scenarios are deliberately not addressed on the grounds that they are a) very poorly understood and b) would

make the specification extremely complex for even a limited set of correlation and aggregation requirements. Please refer to [6] for the list of requirements to be satisfied by the event model. The following discussion is restricted to those requirements that were considered to be a MUST for the base specification.

This specification addresses the following requirements:

- All group 1 requirements concerning when a specific event is created and triggered
- All group 2 requirements concerning the event filter mechanism
- All group 3 requirements concerning event properties
- Requirement 4 concerning best effort event data capture
- All group 7 requirements concerning what the model supports
- All group 8 requirements concerning subscription

APPENDIX C
MAPPING EXISTING MODELS INTO CIM

For the convenience of the reader, the following excerpts are reprinted from the CIM V2 specification concerning mapping other systems to CIM. For a full treatment of this subject the reader is referred to chapter 6 of that document. [1]

A recast mapping provides a mapping of the sources' meta constructs into the targeted meta constructs, so that a model expressed in the source can be translated into the target. The major design work is to develop a mapping between the sources' meta model and the CIM meta model. Once this is done, the source expressions are recast.

A domain mapping takes a source expressed in a particular technique and maps its content into either the core or common models, or extension sub-schemas of the CIM. This mapping does not rely heavily on a meta-to-meta mapping; it is primarily a content-to-content mapping. In one case, the mapping is actually a re-expression of content in a more common way using a more expressive technique.

This is an example of how CIM properties can be supplied by ? DMI? , using information from the ? DMI? di group ("DMTF|Disks|002"). For a hypothetical CIM disk class, the CIM properties are expressed as:

CIM "Disk" property	Can be sourced from ?DMI? group/attribu
?StorageTyp	"MIF.DMTF|Disks|002.1"
?StorageInterfac	"MIF.DMTF|Disks|002.3"
?RemovableDriv	"MIF.DMTF|Disks|002.6"
?RemovableMedi	"MIF.DMTF|Disks|002.7"
?DiskSiz	"MIF.DMTF|Disks|002.16"

[1] A Required property means that a provider for this instance MUST be capable of returning this property . The