CIM Event Model White Paper

CIM Version 2.7

Document Version 2.1 June 10,2003

Abstract

The DMTF Common Information Model (CIM) is a conceptual information model for describing computing and business entities in enterprise and Internet environments. It provides a consistent definition and structure of data, using object-oriented techniques. The CIM Schema establishes a common conceptual framework that describes the managed environment.

The CIM Event Model defines the Event-related abstractions. It describes the CIM Indication hierarchy and the use of Indications to model Events. The Event Model also describes the use of subscriptions to register to receive Indications.

The Specification for CIM Operations over HTTP [1] describes the CIM-XML WBEM Standard for sending and receiving CIM Indications.

Notice

DSP107 ***Status: Preliminary***

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 for uses consistent with this purpose, 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patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a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.

Table of Contents

1 Introduction

1.1 Overview

The subject of Events is complex and covers a wide range of topics and scenarios. An Event can be defined as a change in state. For example, when a service is started, its state is changed from “Stopped” to “Started.” Or, when a plug-and-play device is added to a system, the state of the hardware configuration is changed. Alternatively, Events can be defined as the occurrence of a phenomenon of interest. For example, an Event can denote the occurrence of a disk write error, a failed authentication attempt, or even a mouse click. Both interpretations are supported by the CIM Event Model.

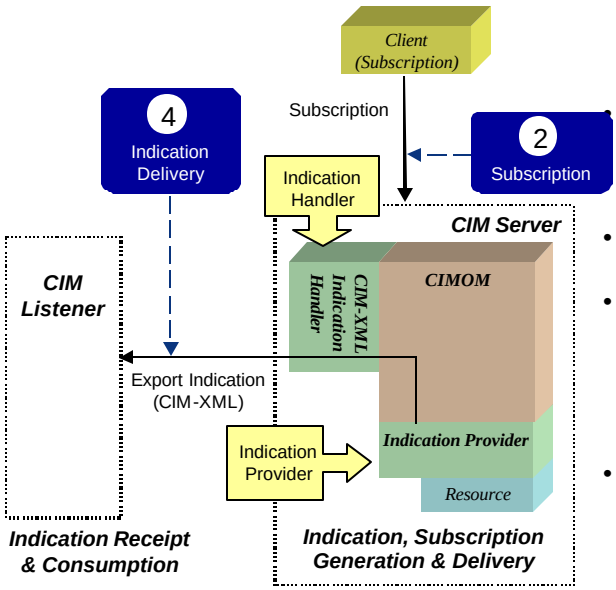
An Event may be a pervasive incident that occurs infrequently, such as a system re-boot, or it may reflect very small scale, frequently occurring incidents, like mouse-clicks.

The way Events are handled can vary enormously. Some Events may require immediate action on the part of the observer, while for other types of Events, the action may be deferred until a later time. For example, an ‘out of disk space’ Event on a Web server may require immediate action to make disk space available. But an Event indicating that an application has exceeded expected CPU utilization can be handled as part of the nightly billing reconciliation.

1.2 Terminology

This section contains a glossary of terms used in this white paper.

Term	Definition
Client	A process that creates subscriptions.
Delivery	The process of transporting one or more Indications to an Indication consumer. The delivery destination, encoding and transport protocol are defined as part of the definition of the subscription.
Event	An occurrence of a phenomenon of interest.
Filter	An instance of CIM_IndicationFilter that defines the set of Indications of interest to an Indication consumer.
Handler	An instance of CIM_IndicationHandler that describes the location, encoding and transport protocol of an Indication consumer.
Indication	The representation of the occurrence of an Event.
Indication Stream	A defined set of Indications.
Indication Consumer	A process that consumes Indications.
Query	A description of the interesting characteristics of one or more Indications. A query is a component of a Filter.
Subscription	The process of registering to receive Indications.



- **Indication.** The representation of the occurrence of an event.

Subscription. The process of registering to receive Indications.

- **Filter.** The definition of a set Indications of interest.
- **Delivery.** The process of transporting one or more indications to a designated destination. The intended protocol and destination are specified as part of the subscription definition.
- **Client.** A process that creates subscriptions.

Figure 1 Terminology

2 The Event Model

2.1 Model Overview

The CIM Event Model defines Event-related abstractions. It describes the CIM Indication hierarchy and the use of Indications to model Events. The Event Model also describes the use of Subscriptions to register to receive Indications.

The Event Model has been designed to address two potentially conflicting requirements. First, it has to be extensible, allowing schema designers to add new types of Indications (Events) to reflect 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 requiring the application to be aware of the full range of Event types implied by the first requirement.

2.1.1 CIM Indications

The CIM_Indication hierarchy is used to describe the type of Events that can be detected. An abstract class, CIM_Indication serves as the base class for all Indication classes.

In version 2.7 of the Event Schema, three types of Indications (representing different types of Events) are modeled as CIM_Indication subclasses. These subclasses include:

- **CIM_InstIndication** – used to report CIM Instance life cycle (i.e., change of state) Events. Types of Events include: Instance creation, deletion, modification, method invocation and read access.
- **CIM_ClassIndication** – used to report CIM class definition life cycle Events. Types of Events include: class creation, deletion and modification.
- **CIM_ProcessIndication** – used to report the occurrence of a "phenomenon of interest." This includes alert or notification type Events.

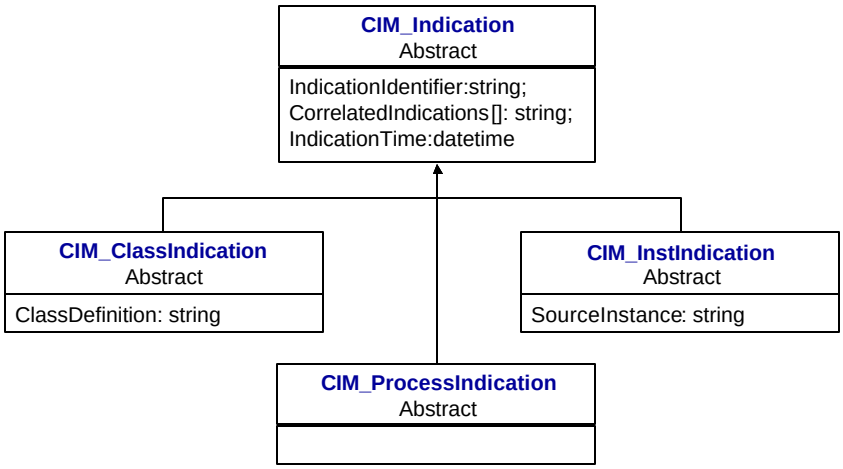


Figure 2 CIM_Indication Hierarchy

The CIM Indication hierarchy is extensible. New subclasses can be added to capture vendor-specific properties and Event types. The following figure illustrates extensions to the CIM Indication hierarchy to support UPS alerts from a hypothetical vendor, ACME.

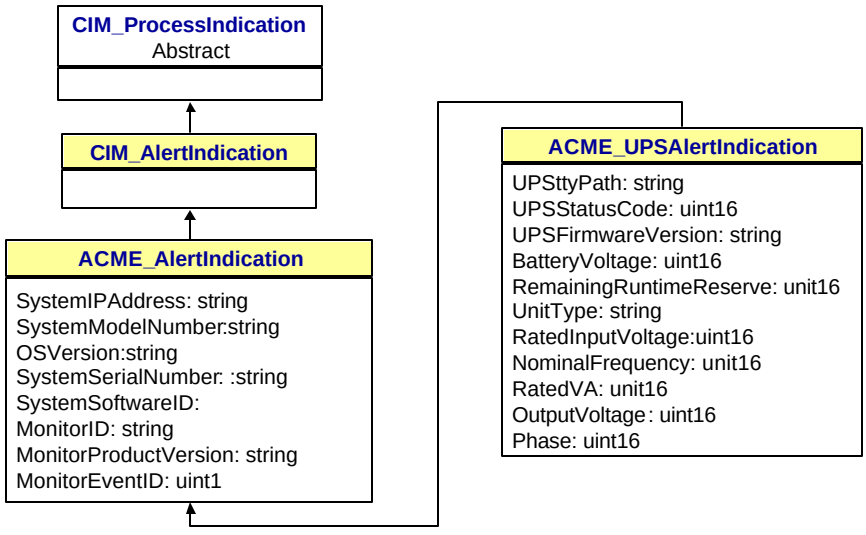


Figure 3 CIM Indication Hierarchy Extension

2.1.2 CIM Indication Subscriptions

Creating an instance of CIM_IndicationSubscription defines a subscription. Instances of the CIM_IndicationSubscription association class are used to register to receive Indications. In a subscription, the CIM_IndicationFilter instance describes the desired set of Indications (i.e., the desired Indication stream), and the CIM_IndicationHandler instance describes the target destination (i.e., the Indication consumer) and communication protocol. An instance of CIM_IndicationSubscription defines an association between a CIM_IndicationFilter instance and a CIM_IndicationHandler instance.

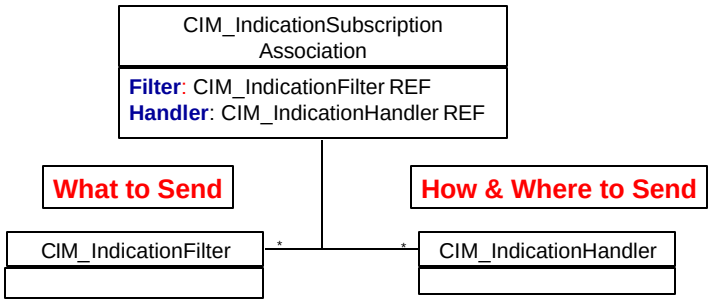


Figure 4 Subscription Hierarchy

2.1.3 Events, Indications and Correlations

There is a subtle, but fundamental, difference between an Event and an Indication. An Event is defined to be the "occurrence of a phenomenon of interest." Examples of Events could include: the failure of a hardware device, the exceeding of an acceptable resource utilization threshold, or the successful completion of the installation of a product. An Indication, on the other hand, is a record of the detection of an Event of interest. It is important to understand that there is not a one-to-one correspondence between Events and Indications. In particular, zero or more Indications can be generated for the same underlying Event.

Consider, for example, instrumentation designed to detect and report memory failures. A memory vendor could decide that, even though the hardware can detect and report failures at a finer granularity, the "phenomenon of interest" (i.e., the Event) is the failure of a replaceable memory component. Multiple errors can be detected, and consequently Indications generated, by the failure of various memory subcomponents, but each of these errors is associated with the same underlying Event (i.e., the failure of the replaceable memory component).

It is also important to note that the number of times an Event is detected may also be of interest. For example, a high number of "correctable single bit errors" reported against a

single memory component within a specified time interval could indicate that the component is about to fail.

Also included in the Event Model, Correlation is different. Two Indications are correlated if they do not represent the same underlying Event, but there is a relationship between the Events they represent. Consider, for example, instrumentation designed to detect and report software configuration changes as Indications. If the installation of a software package A automatically triggers the installation of dependent software packages B and C, then there is a correlation between the installation of software packages B and C and the installation of software package A.

2.2 CIM Indication Schema

The CIM_Indication class hierarchy is used to describe the type of Events that can be detected. An instance of CIM_Indication represents the occurrence of an Event. Typically, Indications are very short-lived objects (i.e., transient objects) used to communicate information from an Indication generator to zero or more Indication consumers. Indications are not required to have keys.

Although Indications are modeled using CIM classes, Indication classes are unique. In particular, standard CIM Instance Operations (e.g., CreateInstance, GetInstance, EnumerateInstance, or ExecQuery) cannot be used to manipulate or retrieve instances of CIM_Indication classes. Indications can only be received by subscription. However, CIM Class Operations (e.g., CreateClass or ModifyClass) can be used to manipulate CIM_Indication class definitions.

The CIM_Indication class is the base class for all Indication classes. It includes the following properties:

- **IndicationIdentifier** – Although Indications are not guaranteed to be uniquely identifiable, the IndicationIdentifier property can be used for identification. If Indication correlations are reported, this value should be unique.
- **IndicationTime** – describes, to the extent possible, the time of the underlying Event.
- **CorrelatedIndications –** contains a list of Indications, specified by IndicationIdentifer, that are related to (i.e., correlated with) this Indication.

Indications and their properties are to be interpreted in the context of a single namespace. However, there is no requirement for the subscription portion of the Event Schema to be located in that same namespace as the CIM Indication.

The Event Model defines three abstract subclasses of CIM_Indication:

- **CIM_InstIndication** – denotes the occurrence of an action on an instance. Types of CIM_InstIndication include: CIM_InstCreation, CIM_InstDeletion, CIM_InstModification, CIM_InstMethodCall and CIM_InstRead.
- **CIM_ClassIndication** – denotes the occurrence of an action on a class definition. Types of CIM_ClassIndication include: CIM_ClassCreation, CIM_ClassDeletion and CIM_ClassModification.
- **CIM_ProcessIndication** – denotes the occurrence of a phenomenon of interest. Types of CIM_ProcessIndication include: CIM_SNMPTrapIndication and CIM_AlertIndication.

2.2.1 CIM_InstIndication

An instance of CIM_InstIndication denotes the occurrence of an action on an instance. Types of actions include: creating an instance, deleting an instance, modifying an instance, reading an instance or invoking an extrinsic method on an instance.

An instance of CIM_InstIndication includes a copy (i.e., a current snapshot) of the instance, SourceInstance, on which the action occurred. Instances of CIM_InstModification also include a copy of the instance, PreviousInstance, before the modification occurred. Instances of CIM_InstMethodCall include information about the extrinsic method invocation.

2.2.2 CIM_ClassIndication

An instance of CIM_ClassIndication denotes the occurrence of an action on a class definition. Types of actions include: creating a class, deleting a class or modifying a class definition.

An instance of CIM_ClassIndication includes a copy (i.e., a current snapshot) of the class definition, ClassDefinition, on which the action occurred. Instances of CIM_ClassModification also include a copy of the class definition, PreviousClassDefinition, before the modification occurred.

2.2.3 CIM_ProcessIndication

An instance of CIM_ProcessIndication denotes the occurrence of an Event.

In version 2.7 of the Event Schema, the following two subclasses of CIM_ProcessIndication are defined:

- **CIM_AlertIndication –** denotes the occurrence of an alert or notification Event. Indications of this class contain information about an alerting situation such as PerceivedSeverity, ProbableCause, RecommendedAction and Trending. A subclass of CIM_AlertIndication, CIM_AlertInstIndication, provides a mechanism to integrate other domain notifications with CIM_InstIndications.

- **CIM_SNMPTrapIndication** - provides a general mechanism for modeling SNMP Traps as CIM Indications. The mapping of an SNMP Trap to a CIM Indication defined in this class is based on the IETF RFC 1157. Naturally, a complete understanding of any trap data received relies on the Indication consumer understanding the sender's MIB.

2.3 CIM Subscription Schema

The subscription portion of the Event Schema defines how Clients subscribe to receive Indications. A CIM Client application activates a subscription by creating an instance of CIM_IndicationSubscription. A CIM_IndicationSubscription instance defines an association between a CIM_IndicationFilter instance and a CIM_IndicationHandler instance. The CIM_IndicationFilter instance describes the desired Indication stream. The CIM_IndicationHandler instance describes "how and where" to deliver the Indication streams.

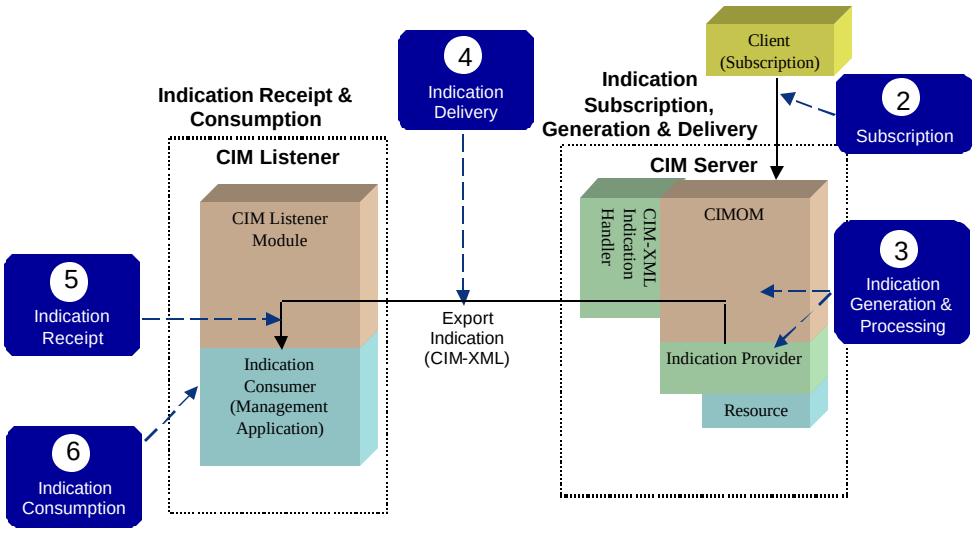


Figure 5 Indication Flow

Indications are sent only if there is an instance of CIM_IndicationSubscription that requests the delivery of the Indication. That is, the Indication matches the condition defined by the CIM_IndicationFilter instance and is dispatched only to the destination defined by the CIM_IndicationHandler instance.

2.3.1 CIM_IndicationFilter

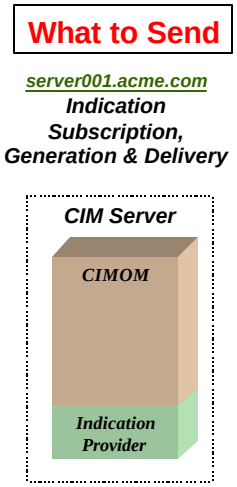
An instance of CIM_IndicationFilter defines a desired set of Indications. In particular, the CIM_IndicationFilter instance defines the namespace, Indication class, filter criteria and data project list of the desired Indication stream.

The CIM_IndicationFilter class includes the following set of properties that define an Indication steam:

- **SourceNamespace** – defines the namespace for the Indication stream. In particular, all Indications in the generated Indication stream *must* be applicable to, and consistent with, the designated namespace.
- **Query** – defines the Indication class, filter condition and property list of the Indication stream.
- **QueryLanguage** – defines the Query Language used to define the Query.

The following figure shows an example of a CIM_IndicationFilter instance.

Indication Filter



CIM_IndicationFilter		
	Property Name	**Value**
	Caption	Generate critical UPS alerts for /dev/tty0p1
	Description	Filter used by ACME to generate critical UPS alerts for device 'tty0p1'. This indication will return all properties associated with the class ACME_UPSAlertIndication.
Key	SystemCreationClassName	CIM_UnitaryComputerSystem
Key	SystemName	server001.acme.com
Key	CreationClassName	CIM_IndicationFilter
Key	Name	ACMESubscription12345
	SourceNamespace	root/cimv2
Required	Query	SELECT * FROM ACME_UPSAlertIndication WHERE PerceivedSeverity = Critical AND UPSttyPath = "/dev/tty0p1"
Required	QueryLanguage	WQL

Figure 6 CIM_IndicationFilter Example

2.3.1.1 Query Examples

Queries are used to define the Indication stream. The following are examples of queries for various CIM_Indication classes. A Query Specification is not yet approved by the DMTF. The syntax used in these examples is representative of the current proposal, but is still subject to change.

2.3.1.1.1 CIM_AlertIndication Filter Example

Subscribe to Events where PerceivedSeverity is "Major" (5) and the “Quality of Service Alert” (3) of the managed object is “Trending Up” (2).

```
SELECT      AlertingManagedElement, ProbableCause,
            ProbableCauseDescription, RecommendedActions
FROM        CIM_AlertIndication
WHERE       PerceivedSeverity = 5 AND
            AlertType = 3 AND
            Trending = 2
```

This query returns the AlertingManagedElement containing the path of the alerting CIM object (or other Domain-identifying string, if it is not a CIM object), as well as 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 IndicationIdentifier (used to identify this instance), and the CorrelatedIndication property, which contains an array of IndicationIdentifiers for other Indications related to this one, thus allowing subscribers to correlate multiple CIM_AlertIndications.

2.3.1.1.2 CIM_InstCreation Filter Example

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_InstCreation
WHERE       SourceInstance ISA CIM_LogicalDevice
```

The application would subscribe by creating a CIM_IndicationSubcription instance containing the reference to the Filter (with the above query,) and the desired Handler (i.e., Indication consumer).

2.3.1.1.3 CIM_InstModification Filter Example

An application interested in receiving an Indication when the status of a network port changes to "Degraded" (3) might create a Filter with the following query:

SELECT SourceInstance.Name, SourceInstance.Description

FROM CIM_InstModification

WHERE SourceInstance ISA CIM_NetworkPort AND
SourceInstance.OperationalStatus = 3 AND
PreviousInstance.Status <> SourceInstance.Status

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

The "PreviousInstance.status <> SourceInstance.status" clause is used here to illustrate how a client may suppress unwanted repetitive Events, after a threshold condition has been triggered.

2.3.1.1.4 SNMPTrapIndication Filter Example

Subscribe to all traps from Compaq Computer Corporation (i.e., Enterprise = 1.3.6.1.4.1.232) devices signaling a link down error.

SELECT AgentAddress

FROM CIM_SNMPTrapIndication

WHERE Enterprise = "1.3.6.1.4.1.232" AND
GenericTrap = "Link Down"

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 general, and the Compaq Computer Corporation MIBs in particular, could have subscribed to the unprocessed contents of the trap by including the CIM_SNMPTrapIndication.VariableBindings property in the **SELECT** list.

2.3.2 CIM_IndicationHandler

An Instance of CIM_IndicationHandler describes "how and where" to send an Indication. In particular, the CIM_IndicationHandler instance defines the desired Indication destination, encoding and protocol for delivery of the Indication stream.

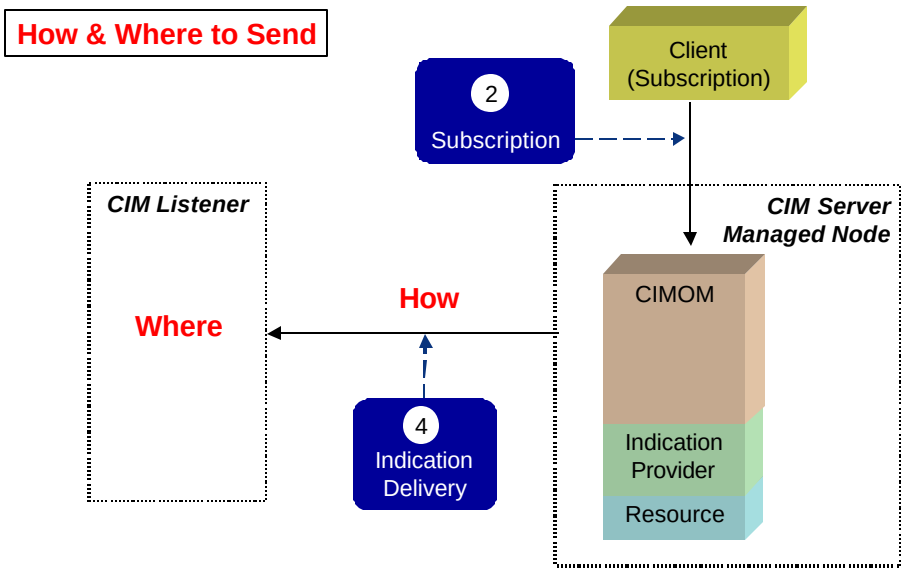


Figure 7 Indication Handler

The 2.7 version of the Event Model defines a single, concrete subclass of CIM_IndicationHandler:

- **CIM_IndicationHandlerCIMXML** – this class is used to describe Indication consumers that comply with the DMTF WBEM CIM-XML standard.

The CIM_IndicationHandler class hierarchy can be extended to allow the definition of alternative Indication handling mechanisms, such as point-to-point protocols, multi-cast delivery, email, paging or logging.

The CIM_IndicationHandler class is the base class from all CIM_IndicationHandler classes. It includes the following non-key properties:

- **PersistenceType** and **OtherPersistenceType** – characterize the expected "lifetime" of the Indication consumer.
- **Owner** – this property is ill defined and will be proposed for "deprecation" in version 2.8 of the Event Schema.

2.3.2.1 Persistence Properties

The **PersistenceType** and **OtherPersistenceType** properties allow a client to characterize the expected "lifetime" of the Indication consumer.

Two values are defined for the **PersistenceType** property: **Permanent** and **Transient**.

- A **Permanent** destination (i.e., Indication consumer) is always expected to be available (e.g., system log file). Inability to access a **Permanent** Indication consumer *must* be treated as an error condition.

- The availability of a **Transient** destination is expected to be short-lived. Inability to access a Transient Indication consumer *may* be treated as a normal termination. Subscriptions with "Transient" Indication consumers may be deleted when the destination is no longer available.

2.3.3 CIM_IndicationSubscription

In addition to describing the association between a CIM_IndicationFilter instance and a CIM_IndicationHandler instance, Instances of CIM_IndicationSubscription include a set of properties that describe various aspects of the Subscription life cycle.

- The **Repeat Notification** properties define the desired frequency for notifying a consumer of the occurrence of an Event that satisfies the subscription.
- The **Subscription State** properties allow a Client to monitor and control the state of the subscription.
- The **Subscription Failure Handling** properties define the desired behavior when a fatal error occurs processing the subscription.
- The **Subscription Duration** properties define the desired length of the subscription.

2.3.3.1 Repeat Notification Properties

The Repeat Notification properties, **RepeatNotificationPolicy**, **OtherRepeatNotificationPolicy**, **RepeatNotificationInterval**, **RepeatNoficationGap** and **RepeatNotificationCount**, allow a client to control how frequently repeat Event notifications are sent to the Indication consumer. These properties give Clients greater control over resource utilization (e.g., network bandwidth), by allowing the repeat notification frequency to be configured at the subscription level.

At a conceptual level, a Repeat Notification refers to an Indication that describes an Event that has already been reported. By definition, all Repeat Notifications *must* match the same filter condition as the initial notification.

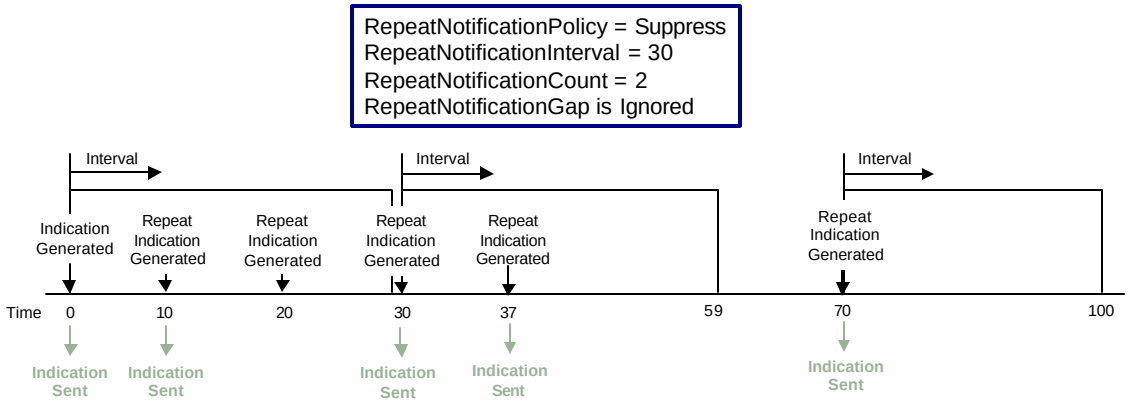
The interpretation of the RepeatNotificationInterval, RepeatNotificationGap and RepeatNotificationCount properties will vary depending on the value of RepeatNotificationPolicy.

The RepeatNotificationPolicy property defines two types of Repeat Notification polices: **Suppress** and **Delay**.

- The **Suppress** policy is useful for suppressing the notification of critical conditions that are detected frequently, but may take time to repair (e.g., failure of a replaceable hardware component).

Repeat Notification

If the value is '**Suppress**' the first '**RepeatNotificationCount**' indications, describing the same event, must be sent and all subsequent indications for this event suppressed for the remainder of the time interval '**RepeatNotificationInterval**'. A new interval starts when the next indication for this event is received.



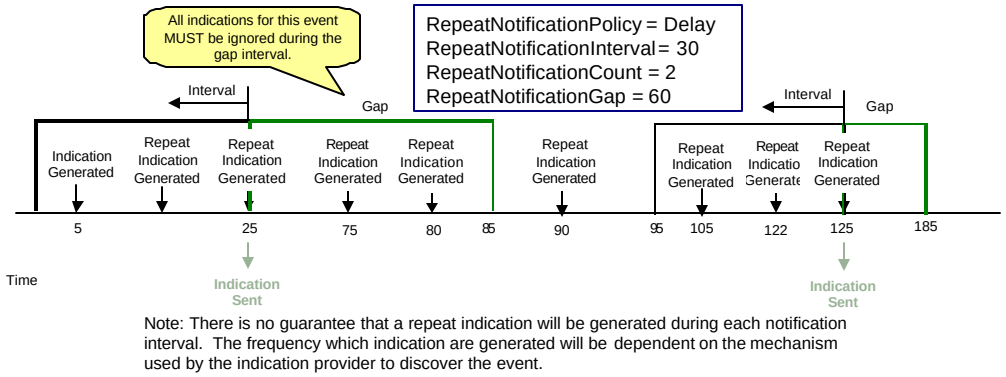
Note: There is no guarantee that a repeat indication will be generated at a consistent interval. The frequency which indication are generated will be dependent on the mechanism used by the indication provider to discover the event.

Figure 8 Suppress Repeat Notification Policy

- The **Delay** policy is useful for suppressing early notification of conditions that may "self repair" (e.g., a transient spike in CPU utilization).

Repeat Notification

If the value of RepeatNotificationPolicy is '**Delay**' and an indication is received, this indication MUST be suppressed if, including this indication, **RepeatNoticationCount** or fewer indications for this event have been received during the prior time interval defined by **RepeatNotificationInterval**. If this indication is the **RepeatNotificationCount + 1** indication, this indication MUST be sent and all subsequent indications for this event ignored until the **RepeatNotificationGap** has elapsed. A R**epeatNotificationInterval** MAY NOT overlap a **RepeatNotificationGap** time interval.



Note: There is no guarantee that a repeat indication will be generated during each notification interval. The frequency which indication are generated will be dependent on the mechanism used by the indication provider to discover the event.

Figure 9 Delay Repeat Notification Policy

2.3.3.2 Subscription State Properties

The Subscription State properties, **SubscriptionState**, **OtherSubscriptionState**, and **TimeOfLastStateChange,** allow a client to monitor and control the state of a Subscription. Possible values of SubscriptionState include: **Enable, EnableDegraded** and **Disabled**.

2.3.3.3 Subscription Failure Handling Properties

The Subscription Failure Handling properties, **OnFatalErrorPolicy**, **OtherOnFatalErrorPolicy** and **FailureTriggerTimeInterval,** allow a client to define the desired error handling behavior. Possible failure conditions include:

- Failure of an Indication Handler when trying to deliver an Indication.
- Failure by an Indication Service when trying to process an Indication.
- Failure of an Indication Provider. Note that this type of failure would not necessarily be associated with the generation of an Indication.

Not all types of failures are detectable or considered as error conditions. For example, delivery of an Indication over a specific protocol may not be guaranteed,meaning this type of delivery failure may not be detectable and would not be treated as an error.

The OnFatalErrorPolicy property defines three types of polices:

- **Ignore** – If the Ignore policy is set, processing of the subscription will continue in a "best effort" mode.
- **Disable** – If the Disable policy is set, the subscription is disabled. No new Indications will be generated or delivered until the subscription is explicitly enabled.
- **Remove** – If the **Remove** policy is set, the subscription is deleted. Selecting this policy has the same effect as issuing a DeleteInstance operation on all CIM_IndicationSubscription instances affected by the failure. This policy is useful in situations where subscriptions are expected to "exist" only for the duration of an "end user" session. For example, when creating ad-hoc subscriptions associated with an interactive monitoring session.

2.3.3.4 Subscription Duration Properties

The Subscription Duration properties, **SubscriptionDuration**, **SubscriptionStartTime**, and **SubscriptionTimeRemaining,** allow a client to monitor and control the duration of a Subscription.

The following table shows the relationship between the CIM_IndicationSubscription.SubscriptionDuration property and the CIM_IndicationHandler.PersistenceType property:

Indication Consumer Type	Subscription Expiration Date	Behavior
Persistent	NULL	Subscription must be explicitly deleted.
Transient	NULL	Subscription should be automatically deleted when the Indication consumer terminates.
Persistent	NON-NULL	Subscription should be automatically deleted when the expiration date is passed.
Transient	NON-NULL	Subscription should be automatically deleted when either the Indication consumer terminates or the expiration date is passed.
NULL	NULL	Subscription must be explicitly deleted.
NULL	NON-NULL	Subscription should be automatically deleted when the expiration date is passed.

3 Event Model Use Case

3.1 Subscriptions

Use Model 1: Subscriptions are created when the system is deployed. Indication delivery is viewed as a critical component of a remote support service that offers a guaranteed level of availability. The expectation is that the subscriptions "exist" for the life of the system. Subscriptions must be explicitly deleted.

Use Model 2: Subscriptions are defined and removed as part of an interactive monitoring session with an "end user." The client application creates the subscriptions, and also serves as the Indication consumer. The expectation is that the subscriptions "exist" only while the client session is active. A subscription should be automatically deleted when the client application terminates.

Use Model 3: Subscriptions provide a service and are created, renewed and deleted based on the terms of a lease. The expectation is that a subscription "exists" for the duration of time specified by the lease. An "expiration date" is associated with each subscription and a "renewal" would consist of updating the expiration date. A subscription should be automatically deleted when the expiration date has passed.

3.2 Indication Delivery

Use Model 1: Indications are received, acted on by the management application, but not stored (e.g., "mouse-click," "plug-n-play" device added, statistics gathering data).

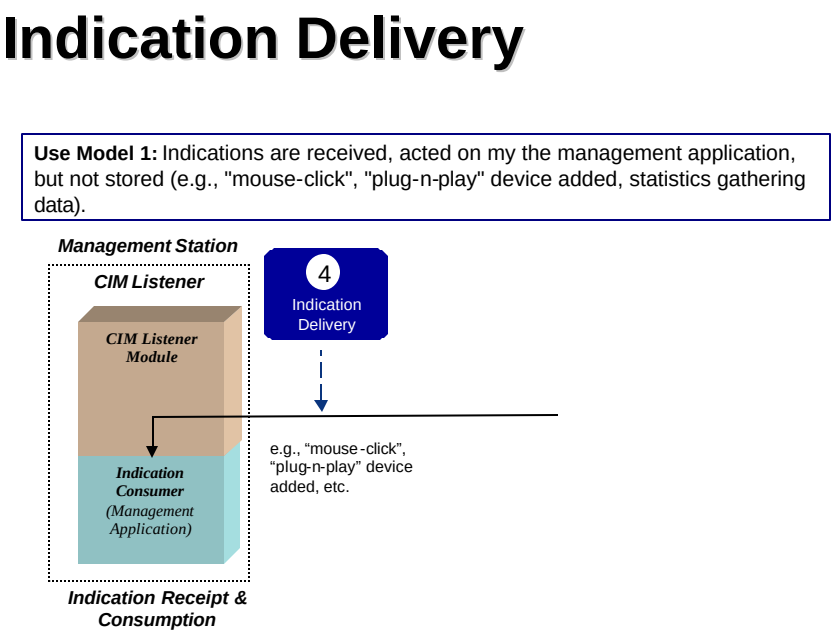


Figure 10 Indication Delivery - Use Model 1

Use Model 2: Indications are received, acted on by the management application, and stored in an application-specific repository.

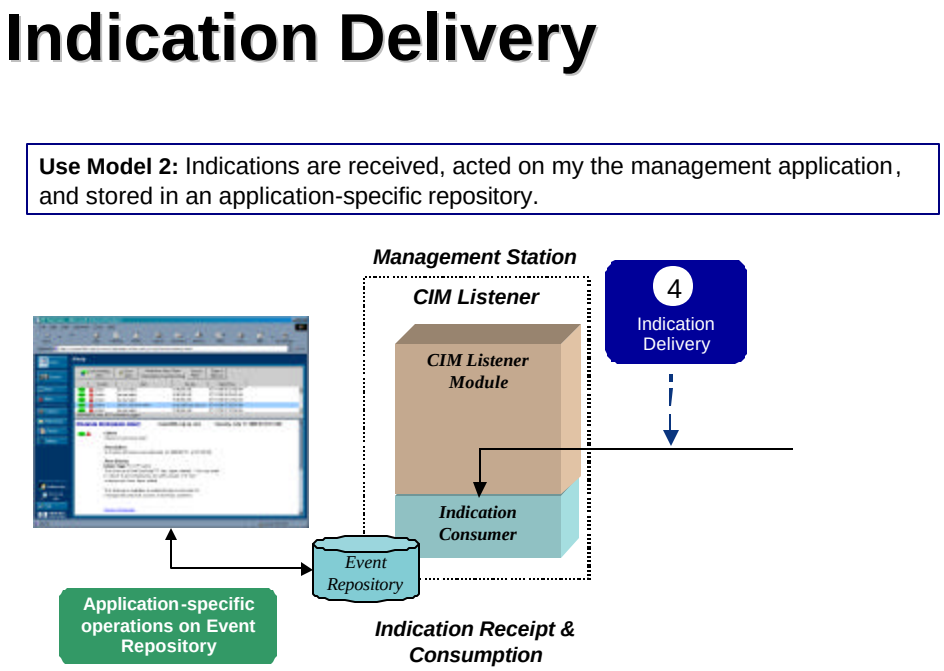


Figure 11 Indication Delivery - Use Model 2

Use Model 3: Indications are received, acted on by the management application, and stored in a repository that is accessible through CIM.

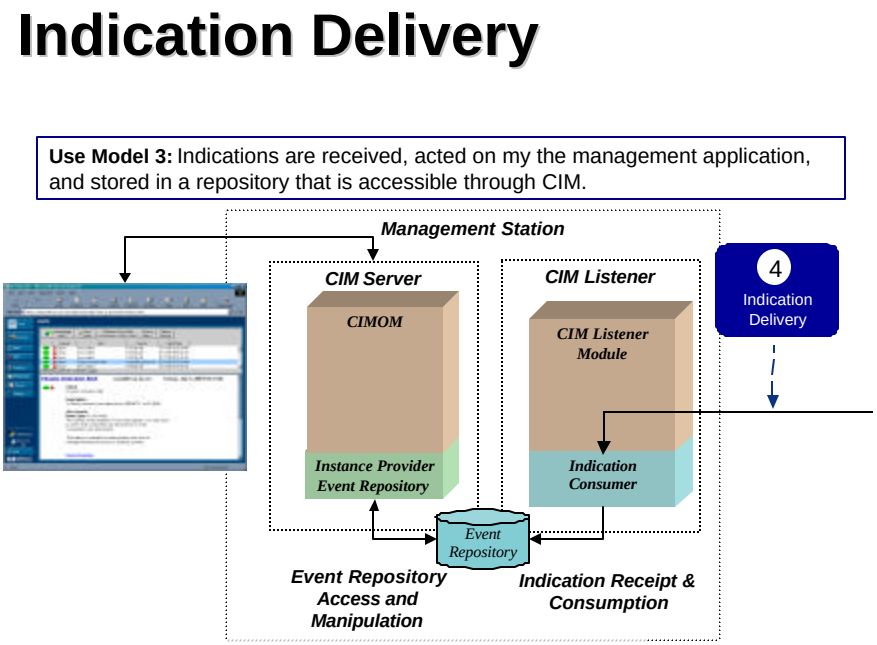


Figure 12 Indication Delivery - Use Model 3

Appendix A – 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 Draft	November 6, 2000	Added changes agreed upon at 11/2/00 teleconf: Change AlertIndication.AlertType from "General." to "Primary classification ..., Updated text for 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). Changed 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.
Version 2.6	May 23, 2001	Change name of IndicationHandlerXMLHTTP to IndicationHandlerCIM-XML to reflect DMTF naming convention; CIM-XML denotes xml encoded CIM operations over HTTP.
Version 2.7	August 27, 2001	Change name of IndicationHandlerCIM-XML to IndicationHandlerCIMXML. '-' is an illegal character in MOF syntax.
Version 2.7	September 14, 2001	Corrections to IndicationHandler.Owner description, AlertIndication.AlertingManagedObject Descriptions.
Version 2.7	June 7, 2003	Updates to reflect version 2.7 changes.

Appendix B – References

[1] CIM Operations over HTTP, v1.1, DSP0200 - Downloadable from http://www.dmtf.org/standards/standard_wbem.php