Document Number: DSP1054

Date: 2011-06-30

Version: 1.2.0

Indications Profile

Document Type: Specification

Document Status: DMTF Standard

Document Language: en-US

CONTENTS

Figures

Tables

Foreword

The *Indications Profile* (DSP1054) was prepared by the DMTF WBEM Infrastructure Modeling Working Group. Version 1.0 was prepared by the DMTF WBEM Infrastructure and Protocols Working Group.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about the DMTF, see http://www.dmtf.org.

Acknowledgments

DMTF acknowledges the following individuals for their contributions to this document:

Editor:

- Michael Johanssen (IBM)

Contributors:

- Jim Davis, WBEM Solutions (former editor)
- Steve Hand, Symantec (former editor)
- Jon Hass, Dell (former editor)
- David Judkovics, IBM (former editor)
- Andreas Maier, IBM (former editor)
- Aaron Merkin, IBM (former editor)
- Venkat Puvvada, IBM
- Karl Schopmeyer, DMTF Fellow
- Hemal Shah, Broadcom (former editor)

Introduction

The information in this specification should be sufficient for a provider or consumer of this data to unambiguously identify the classes, properties, methods, and values that shall be instantiated to subscribe, advertise, produce, or consume an indication using the DMTF Common Information Model (CIM) Schema.

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

Document conventions

Typographical conventions

Any text in this document is in normal text font, with the following exceptions:

- Document titles are marked in *italics*.
- Important terms that are used for the first time are marked in *italics*.
- Terms within the text contain a link to the term definition defined in the "Terms and definitions" clause, enabling easy navigation to the term definition.
- ABNF rules are in `monospaced font`.

ABNF usage conventions

Format definitions in this document are specified using ABNF (see RFC5234), with the following deviations:

- Literal strings are to be interpreted as case-sensitive Unicode characters, as opposed to the definition in RFC5234 that interprets literal strings as case-insensitive US-ASCII characters.

Deprecated material

Deprecated material is not recommended for use in new development efforts. Existing and new implementations may use this material, but they shall move to the newer approach as soon as possible. An implementation of this profile in a CIM server shall use any deprecated material as if it were not deprecated, in order to achieve backwards compatibility for clients. Although implementations of clients may use deprecated material, it is recommended that they use the newer approach instead.

The following typographical convention indicates deprecated material:

DEPRECATED

Deprecated material appears here.

DEPRECATED

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

Experimental material

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

The following typographical convention indicates experimental material:

EXPERIMENTAL

Experimental material appears here.

EXPERIMENTAL

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

Indications Profile

1 Scope

The *Indications Profile* defines the CIM elements that are used to subscribe for indications of unsolicited events, to advertise the possible indications, and to represent indications used to report events in a managed system.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated or versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies. For undated and unversioned references, the latest published edition of the referenced document (including any corrigenda or DMTF update versions) applies.

DMTF DSP0004, *CIM Infrastructure Specification 2.6*,
http://www.dmtf.org/standards/published_documents/DSP0004_2.6.pdf

DMTF DSP0202, *CIM Query Language Specification 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0202_1.0.pdf

DMTF DSP0207, *WBEM URI Mapping Specification 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0207_1.0.pdf

DMTF DSP0223, *Generic Operations 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0223_1.0.pdf

DMTF DSP0228, *Message Registry XML Schema 1.1*,
http://schemas.dmtf.org/wbem/messageregistry/1/dsp0228_1.1.xsd

DMTF DSP1001, *Management Profile Specification Usage Guide 1.1*,
http://www.dmtf.org/standards/published_documents/DSP1001_1.1.pdf

DMTF DSP1033, *Profile Registration Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1033_1.0.pdf

IETF RFC3986, *Uniform Resource Identifier (URI): Generic Syntax, January 2005*,
http://tools.ietf.org/html/rfc3986

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

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

3 Terms and definitions

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

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

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

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

The terms defined in DSP0004, DSP0223 and DSP1001 apply to this document. The following additional terms are used in this document.

3.1

alert indication

an indication that indicates an event related to the managed environment

For details, see 6.1.2.2.

3.2

client

a WBEM client that exploits applicable portions of this profile

For details, see DSP1001.

3.3

coverage

the set of indications that can pass an indication gate

For details, see 6.2.2 and 6.3.2.

3.4

defined coverage

the coverage specified by a profile for static filter collections through normative statements

For details, see 6.3.3.

3.5

dynamic indication filter

an indication filter whose lifecycle is controlled by a client

3.6

event

an observable occurrence of a phenomenon of interest

For details, see 6.1.

3.7

filter collection

an indication gate that may contain other indication gates such as indication filters or other filter collections

For details, see 6.3.

3.8

global indication filter

an indication filter that covers large sets of indications, such as all alert indications

For details, see 6.2.5.

3.9

global filter collection

a filter collection that covers large sets of indications, such as all lifecycle indications

For details, see 6.3.3.5.

3.10

implementation

a WBEM server that implements applicable portions of this profile and of referencing profiles

For details, see DSP1001.

3.11

indication

the notification about an event that occurred

For details, see 6.1.

3.12

indication delivery

the process of delivering indications from an implementation to a listener

indication filter

an indication gate whose coverage is defined through a query statement

For details, see 6.2

3.13

indication filtering

the process of selecting indications based on filtering rules applied by indication gates, such that only indications within the coverage of the indication gate pass the indication gate

3.14

indication gate

a managed element that filters indications such that only indications within its coverage pass. Indication gates can serve as targets for subscriptions, and control which indications are delivered to subscribed listeners.

3.15

indication generation

the process of creating an indication as the event that the indication is designed to report occurs

3.16

indication origin

the namespace out of that the indication originates

For details, see 6.1.2.4.

3.17

indication service

a component within a WBEM server for indication related processing, including handling of subscriptions and delivery of indications to a WBEM listener

3.18

indication system

a system that hosts a WBEM server with one or more indication services

For details, see 6.6.

3.19

indication-specific indication filter

a static indication filter that covers a particular indication specified in a profile

For details, see 6.2.4.

3.20

Interop namespace

a namespace containing CIM instances representing specific capabilities of a WBEM server

Examples include CIM_RegisteredProfile instances representing specific versions of profiles or CIM_IndicationFilter instances representing indication filters. For details, see DSP1033.

3.21

lifecycle indication

an indication indicating an event related to the lifecycle of CIM instances or CIM classes; for details, see 6.1.2.3.

3.22

listener

a WBEM listener that implements applicable portions of this profile

For details, see DSP1001.

3.23

listener destination

an entity that maintains a reference to a listener within an implementation; for details, see 6.4.5..

3.24

profile-specific filter collection

a static filter collection that covers all indications of a particular type defined in a profile

For details, see 6.3.3.4.

3.25

query statement

a statement expressed in a query language used to describe either (a part of) an event or the coverage of an indication filter

3.26

referencing profile

a profile referencing this profile

Note that DSP1001 requires each profile that defines indications to reference this profile.

3.27

reliable indication

an indication containing a sequence identifier enabling listeners to detect duplicate, missing, or out-of-order indications

For details, see 6.1.5 and 7.4.

3.28
repeated indication
an indication that reports the same event as a previous indication
For details, see 6.1.6.

3.29
repeated indication delivery
the delivery of repeated indications
Repeated indication delivery typically occurs if the reported event describes a persistent situation such as exceeding a threshold value.

3.30
sequence identifier
data element with a reliable indication that ensures unique identification of the reliable indication
A sequence identifier is composed of a sequence context and a sequence number
For details, see 7.4.2.

3.31
sequence identifier lifetime
a maximum time interval maintained by an implementation implementing reliable indications within which the implementation retries failed indication delivery attempts
For details, see 7.4.2.

3.32
static filter collection
a filter collection whose lifecycle is controlled by the implementation, that is uniquely identifiable and for which a defined coverage is established
For details, see 6.3.3.

3.33
static indication filter
an indication filter whose lifecycle is controlled by the implementation

3.34
subscription
the mechanism whereby a client registers a listener for the delivery of indications from an implementation

3.35
this profile
a short term for the Indications profile, the profile specified in this specification document (DSP1054)

3.36
WBEM client
a CIM client (see DSP0004) that supports a WBEM protocol
For details, see DSP1001.

3.37
WBEM listener
a CIM listener (see DSP0004) that supports a WBEM protocol
For details, see DSP1001.

3.38
WBEM server
a CIM server (see DSP0004) that supports a WBEM protocol
For details, see DSP1001.

4 Symbols and abbreviated terms

4.1
CQL
CIM Query Language

4.2
QoS
Quality of service

4.3
URI
Uniform Resource Identifier

4.4
WBEM
Web Based Enterprise Management

5 Synopsis

Profile name: Indications

Version: 1.2.0

Organization: DMTF

Profile type: Component

Schema version: 2.25

Central class adaptation: IndicationService (see 7.3.2)

Scoping class adaptation: IndicationSystem (see 7.3.3)

Scoping algorithm: HostedIndicationService (see 7.3.4)

This profile extends the management capabilities defined in referencing profiles by adding the capability to subscribe for indications of unsolicited events, and to notify about such events by means of sending indications from the implementation to a listener. This profile defines the required content of indications defined in referencing profiles.

514 Table 1 lists the profile references defined by this profile.

515 **Table 1 – Profile references**

Profile reference name	Profile name	Organi-zation	Version	Relationship	Description
ProfileRegistration	Profile Registration	DMTF	1.0	Mandatory	Registration of this profile; the central class profile advertisement methodology is mandated by this profile; for details, see 7.3.6.

516 Table 2 lists the class adaptations that are defined in this profile.

517 **Table 2 – Adaptations**

Adaptation	Elements	Requirement	Description
Instantiated and embedded class adaptations			
IndicationService	CIM_IndicationService	Mandatory	See 7.3.2.
IndicationSystem	CIM_System	Mandatory	See 7.3.3.
HostedIndicationService	CIM_HostedService	Mandatory	See 7.3.4.
IndicationsProfileRegistration	CIM_RegisteredProfile	Mandatory	See 7.3.5.
ElementConformsToProfile	CIM_ElementConformsToProfile	Mandatory	See 7.3.6.
IndicationServiceCapabilities	CIM_IndicationServiceCapabilities	Conditional	See 7.3.7.
CapabilitiesOfIndicationService	CIM_ElementCapabilities	Conditional	See 7.3.8.
IndicationServiceInitialSettings	CIM_IndicationServiceSettingData	Conditional	See 7.3.9.
InitialSettingsOfIndicationService	CIM_ElementSettingData	Conditional	See 7.3.10.
IndicationFilter	CIM_IndicationFilter	See derived adaptations	See 7.3.11.
StaticIndicationFilter	CIM_IndicationFilter	See derived adaptations	See 7.3.12.
DynamicIndicationFilter	CIM_IndicationFilter	Conditional	See 7.3.13.
IndicationServiceOfIndicationFilter	CIM_ServiceAffectsElement	Mandatory	See 7.3.14.
IndicationSpecificIndicationFilter	CIM_IndicationFilter	Optional	See 7.3.15.
GlobalIndicationFilter	CIM_IndicationFilter	Mandatory	See 7.3.16.
StaticFilterCollection	CIM_FilterCollection	See derived adaptations	See 7.3.17.
IndicationServiceOfFilterCollection	CIM_OwningCollectionElement	Mandatory	See 7.3.18.
IndicationFilterInFilterCollection	CIM_MemberOfCollection	Conditional	See 7.3.19.
FilterCollectionInFilterCollection	CIM_MemberOfCollection	Conditional	See 7.3.20.
ProfileSpecificFilterCollection	CIM_FilterCollection	Optional	See 7.3.21.
GlobalFilterCollection	CIM_FilterCollection	Mandatory	See 7.3.22.
ListenerDestination	CIM_ListenerDestination	Mandatory	See 7.3.23.
IndicationServiceOfListener-Destination	CIM_ServiceAffectsElement	Mandatory	See 7.3.24.
AbstractSubscription	CIM_AbstractIndication-Subscription	See derived adaptations	See 7.3.25.
FilterSubscription	CIM_IndicationSubscription	Conditional	See 7.3.26.

Adaptation	Elements	Requirement	Description
CollectionSubscription	CIM_FilterCollectionSubscription	Mandatory	See 7.3.27.
ProfileOfFilterCollection { D }	CIM_ConcreteDependency	Mandatory	See 7.3.28.
Indications and exceptions			
BasicIndication	CIM_Indication	See derived adaptations	See 7.3.29.
ReliableIndication	CIM_Indication	See derived adaptations	See 7.3.30.
AlertIndication	CIM_AlertIndication	See derived adaptations	See 7.3.31.
LifecycleIndication	CIM_InstIndication	See derived adaptations	See 7.3.32.
ListenerDestination-RemovalIndication	CIM_InstDeletion	Optional	See 7.3.33.
SubscriptionRemovalIndication	CIM_InstDeletion	Optional	See 7.3.34.

518 Table 3 lists the features that are defined in this profile.

519 **Table 3 – Features**

Feature name	Granularity	Requirement	Description
DynamicIndicationFilters	IndicationService instance	Optional	See 7.2.1.
IndicationServiceInitialSettingsExposed	IndicationService instance	Optional	See 7.2.2.
IndicationServiceModification	IndicationService instance	Optional	See 7.2.3.
ReliableIndications	IndicationService instance	Optional	See 7.2.4.
SuppressRepeatNotificationPolicy	Profile implementation	Optional	See 7.2.5.
DelayRepeatNotificationPolicy	Profile implementation	Optional	See 7.2.6.
IndividualFilterSubscription	IndicationFilter instance	Optional	See 7.2.7.
FilterCollectionCoverageExposure	StaticFilterCollection instance	Conditional	See 7.2.8.

6 Description

This profile defines the concept of indications as a means to notify listeners about events occurring in the managed environments addressed by referencing profiles. This profile establishes basic reusable elements enabling referencing profiles to specify indications that report events occurring in their managed environments. For example, this profile defines reusable adaptations of CIM classes by defining requirements or constraints on suitable properties and methods, by defining required relationships, and by defining the modeled object types in the managed environment.

Furthermore, this profile defines how clients can subscribe listeners for the delivery of indications, and how clients can monitor and control certain aspects of the behavior of implementations of this profile, such as the number of retry attempts or the retry delay when the implementation is unable to deliver indications.

This profile also defines mechanisms for the reliable delivery of indications.

6.1 Events and indications

6.1.1 Events

An event is the observable occurrence of a phenomenon of interest.

Events could be distinguished into root events and secondary events.

Root events are events directly related the managed environment; they may be related to a managed object.

Secondary events are events that are effected by or occur as a consequence of root events. For example, a root event could be the emergence of a fire on a house. Smoke or heat are both possible effects or, in other words, secondary events, caused by the fire.

Furthermore, if a managed object is represented in CIM, the model changes resulting from the change of a managed object may be visible through corresponding changes in its CIM representation.

6.1.2 Indications

6.1.2.1 General

An indication is a notification about an event. It is possible that an indication only reports an aspect of the event and not the entire event. Therefore, multiple indications may be reported in context of a particular event.

For example, an indication could directly report the root event that a house has caught fire. In addition, or alternatively, respective indications could separately report secondary events (or effects) caused by the fire, such as that smoke or heat are observed.

Accordingly, if a managed object is represented in CIM, an indication could directly report the root event related to the managed object. In addition, or alternatively, respective indications could separately report events (or effects) caused by the root event, such that a CIM instance representing an aspect of the managed object was created, modified or deleted.

Reporting events from the managed environment is typically facilitated by means of alert indications, whereas reporting events from the CIM model is typically facilitated by means of lifecycle indications.

6.1.2.2 Alert indications

Alert indications are indications that provide notification about root events (see 6.1.1). If a reported event relates to a managed object, that managed object may or may not have a representation in CIM. Some types of alert indications can also contain information about or refer to corresponding changes in the CIM representation where that is available.

6.1.2.3 Lifecycle indications

Lifecycle indications are indications that provide notification about events (see 6.1.1) related to the lifecycle of CIM instances and CIM classes, such as their creation, deletion or modification.

Only lifecycle events related to the creation, deletion, or modification of CIM instances are within the scope of this profile.

NOTE The CIM schema defines the CIM_InstIndication class as the base class for indications reporting lifecycle events and other model-related events, such as the execution of methods or the execution of read operations; reporting the latter kinds of events is not addressed in this profile.

Lifecycle events related to CIM instances are reported using instances of adaptations of the CIM_InstCreation, CIM_InstDeletion, or CIM_InstModification classes.

It is important to realize that lifecycle events are events (see 6.1.1) in the CIM model, reflecting corresponding events in the managed environment. This applies regardless of whether or not a change was requested by means of a CIM operation; CIM instances are required to always correctly represent (an aspect of) the actual state of a managed object, and thus can only change if the represented (aspect of the) managed object changed.

DSP1001 defines the existence of CIM instances as a logical concept that ties the existence of CIM instances to the existence of the represented managed object in the managed environment (instead of tying the existence of CIM instances to a physical representation such as a repository entry). By that definition the creation of a CIM instance logically occurs when the represented managed object is added to the managed environment, and the deletion of a CIM instance logically occurs when the represented managed object is removed from the managed environment.

With that definition, a CIM instance logically exists even if the WBEM server containing its implementation is inactive, or does temporarily not have access to the managed environment containing the represented managed object. If a WBEM server is inactive when a managed object is added to the managed environment, the CIM instance(s) representing (an aspect of) that managed object still are assumed to be "logically" created exactly at that point in time; however, because the WBEM server is inactive, no lifecycle indications are sent. Furthermore, when the WBEM server is started later on, sending lifecycle indications about lifecycle events occurring while the WBEM server was inactive is not to be made up for. Similarly, when a WBEM server is initially started, lifecycle indications about instances initially existing within that WBEM server are not to be sent. So the DSP1001 based definition of instance existence provides for not having to indicate the creation / deletion of CIM instances every time a WBEM server is activated or deactivated, and avoids requiring a WBEM server to determine which CIM instances were created / deleted / modified while it was inactive.

With the DSP1001 based definition of instance existence, clients may exploit lifecycle indications as a means to monitor the existence of the represented managed object in the managed environment. However, clients cannot rely on indications as the sole means to track the lifecycle of managed objects in the managed environment. At least initially, and after every WBEM server restart, clients actively need to inspect (by means of invoking respective operations) the CIM model of the managed environment for changes that occurred while the WBEM server was inactive. If reliable indications (see 6.1.5) are implemented, a change of the value of the SequenceContext property in the stream of indications arriving at a particular listener from a particular WBEM server may be used as an indicator that a WBEM server restart occurred; for details, see 7.3.30.2.2, and the CIM schema definition of the CIM_Indication class.

A CIM model can represent different aspects of a particular managed object through several instances of different CIM classes. Consequently, one event in the managed environment can be related to multiple events in the CIM model of the managed environment, such as changes in several CIM instances, each of which could be reported through a separate lifecycle indication.

As an example, consider a managed environment composed of systems and their components. If a component such as a fan is added to one of these systems, this would be constitute an event in the managed environment and could be reported by means of an alert indication. Alternatively, or in addition, if the added fan is represented by a CIM_Fan instance, the creation of that CIM_Fan instance could be reported by means of a lifecycle indication.

6.1.2.4 Origin of indications

The origin of an indication is defined as the local namespace in context of that the indication is generated; for details, see 7.3.29.3.

The CIM representation of an indication as defined by the CIM_Indication class does not reflect the origin namespace. Nevertheless, the process of indication filtering (see 6.1.4) is required to consider the origin namespace of an indication; for details, see 7.3.11.2.

6.1.3 Definition of events and indications in referencing profiles

Referencing profiles may define events separately through normative text, or as part of the definition of indication adaptations reporting the event.

NOTE Defining events separately is particularly useful if multiple indications reporting the same event are defined. However, if an event is only reported through one indication, the event definition as part of the definition of the indication adaptation is more compact.

This profile defines several basic indication adaptations for the use by referencing profiles that define indications:

- The BasicIndication adaptation requires the reported event to be specified by means of a query statement; for details, see 7.3.29.2.
- The AlertIndication adaptation refines the BasicIndication adaptation for alert indications. It refines the definition of the query statement, delegating the event definition to an alert message defined in a message registry. For details, see 7.3.31.
- The LifecycleIndication adaptation refines the BasicIndication adaptation for lifecycle indications. A lifecycle indication refers to the CIM instance for which it reports a lifecycle event. The profile defining the lifecycle indications defines for which class adaptations respective lifecycle indications are reported. For details, see 7.3.32.

6.1.4 Indication generation, indication filtering, and indication delivery

The indication related functionality within an implementation can be structured into indication generation, indication filtering and indication delivery. This is detailed in Figure 1.

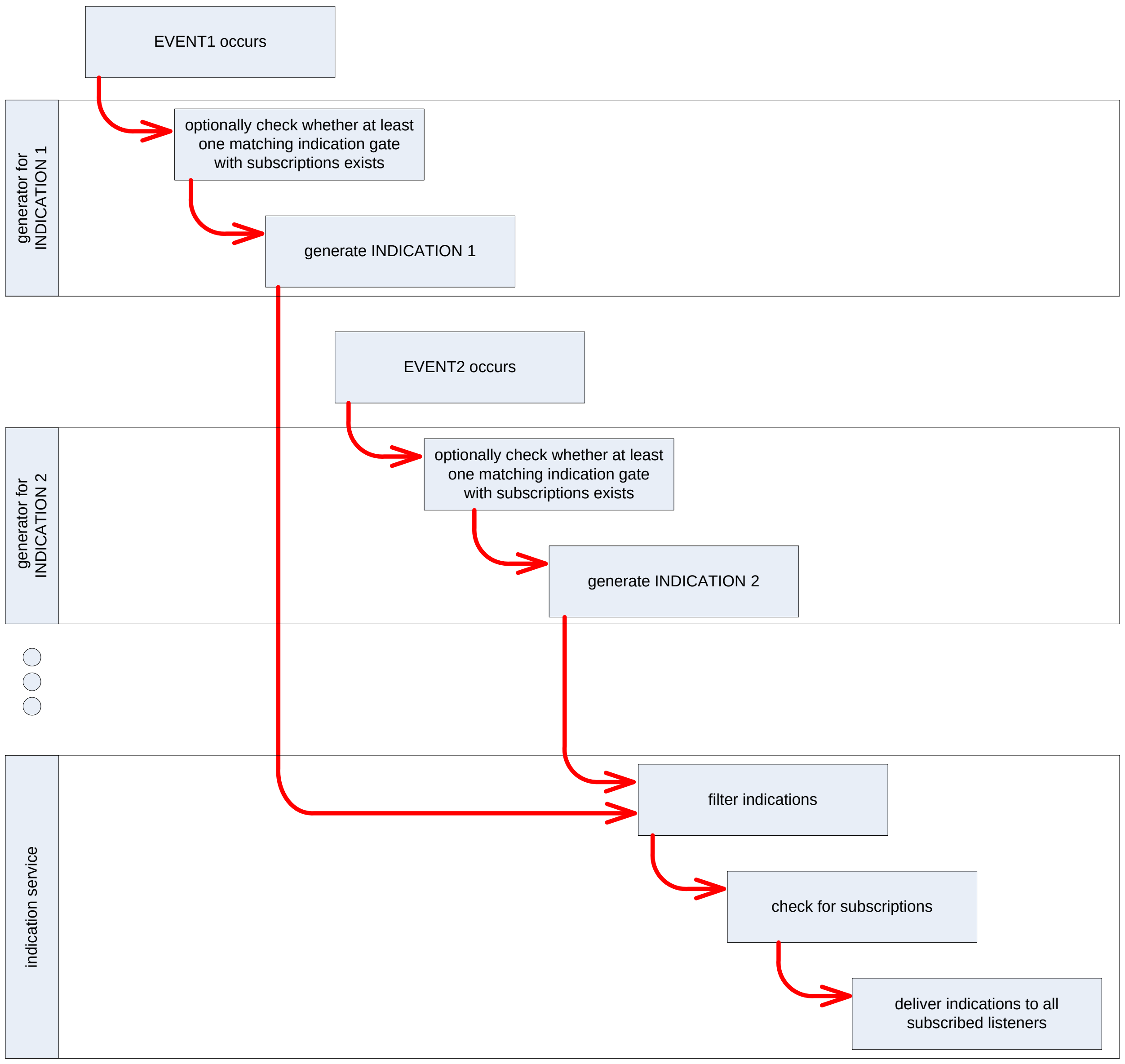

EVENT1 occurs
generator for INDICATION 1
optionally check whether at least one matching indication gate with subscriptions exists
generate INDICATION 1
EVENT2 occurs
generator for INDICATION 2
optionally check whether at least one matching indication gate with subscriptions exists
generate INDICATION 2
indication service
filter indications
check for subscriptions
deliver indications to all subscribed listeners

Figure 1 – Indication related functionality within an implementation

Indication generation is the process of creating an indication as the event that the indication is designed to report occurs. As shown in Figure 1, this functionality is typically implemented separately for each indication, because it depends on the distinct event reported through each particular indication.

Optionally, in order to avoid the generation of indications for which no listeners are subscribed, part of indication filtering can already occur at indication generation time, such that an indication is only generated if at least one indication gate exists that has a coverage covering the indication to be generated, and that has subscribed listeners; for details, see 7.3.29.5. However, even in this case (complete) indication filtering is still required in order to ensure that the generated indication is checked against *every* existing indication gate.

After an indication is generated it is subjected to indication filtering. Indication filtering is the process of selecting indications based on specific filtering rules applied by indication gates, such that only indications within the coverage of the indication gate pass. This functionality is typically implemented in common independent of the implementation of individual indications; however, it depends on indication gates that may be provided by implementations of referencing profiles. For details, see 7.3.11.2 and 7.3.17.2.

Indication delivery is the process of delivering filtered indications from an implementation to a listener. This profile defines rules for the delivery of indications as part of adaptations modeling indications themselves, as part of adaptations modeling indication gates such as indication filters or filter collections, and as part of adaptations modeling subscriptions and listener destinations. For details, see 7.3.23.2 and 7.3.25.2.

6.1.5 Reliable indication delivery

Reliable indication delivery is an optional extension of indication delivery that aims to

- enable implementations to discover and retry unsuccessful indication deliveries, and
- enable listeners to detect duplicate, missing, or out-of-order indications, and to re-order indications that arrive out of order. This includes the discovery of server restarts.

The ReliableIndication adaptation (see 7.3.30) models reliable indications, and additional requirements are specified in 7.4.

6.1.6 Avoidance of repeated indication delivery

6.1.6.1 General

This profile defines policies for the avoidance of repeated indication delivery (see 3.29). Policies for avoiding repeated indication delivery aim at preventing the implementation from flooding subscribed listeners with large amounts of repeated indications. This is a typical scenario if an event models a persistent situation, such as exceeding a threshold value.

For example, consider an indication modeled to report disk i/o errors. If a disk generates i/o errors at a high rate, the implementation would be required to generate a respective amount of indications and deliver them to subscribed listeners.

In order to avoid flooding subscribed listeners with such redundant indications, three policies are modeled in this profile, as detailed in 6.1.6.2, 6.1.6.3 and 6.1.6.4.

The effective policy for the suppression of repeated indication delivery is determined at the level of subscriptions (see 6.4.1). For a particular subscription, the determination whether an indication passing the indication gate referenced by that subscription is a repeated indication — that is, an indication reporting the same event — of a first indication is made as follows: The first indication starts a monitoring time interval. Any indication passing the referenced indication gate during that monitoring time interval is considered a repeated indication if it is equal with the first indication except for the identification and the generation time.

NOTE The identification of indications as modeled by the BasicIndication adaptation (see 7.3.29) is exposed by the value of the IndicationIdentifier property, and the generation time is exposed by the value of the IndicationTime property.
Version 1.1 of this profile also considered the values of the SequenceContext and the SequenceNumber properties (see 7.3.30.2.2 and 7.3.30.2.3) for the determination of repeated indications. However, the values of these properties are specific for listener destinations. Once these values were determined for a particular indication, that indication must be sent to the referenced listener in order to ensure a continuous and homogeneous stream of indications, thereby enabling reliable indication delivery. Thus, the suppression of repeated indication delivery needs to occur before reliable indication processing, and the determination of repeated indications needs to occur without considering these values.

6.1.6.2 No repeated indication delivery avoidance policy

With this policy in effect, no measures against repeated indication delivery are taken (see the CIM schema description of the value 2 (None) for the RepeatNotificationPolicy property of the CIM_AbstractIndicationSubscription class).

6.1.6.3 Suppress repeated indication delivery avoidance policy

This policy is modeled by means of the SuppressRepeatNotificationPolicy feature (see 7.2.5, and the CIM schema description of the value 3 (Suppress) for the RepeatNotificationPolicy property of the CIM_AbstractIndicationSubscription class).

With this policy in effect, the implementation with the delivery of a first indication starts a monitoring time interval. If during that monitoring time interval repeated indications of the first indication accrue, these are likewise delivered up to a predefined threshold. If the threshold is reached while the monitoring time interval is not expired, the delivery of further repeated indications is suppressed until the monitoring time interval expires. After the time interval has expired, the cycle is repeated with the next accruing repeated indication.

6.1.6.4 Delayed indication delivery avoidance policy

This policy is modeled by the DelayRepeatNotificationPolicy feature (see 7.2.6, and the CIM schema description of the value 4 (Delay) for the RepeatNotificationPolicy property of the CIM_AbstractIndicationSubscription class).

With this policy in effect, the implementation with a first accruing indication starts a specified monitoring time interval; however, the first indication is not delivered at that point in time. Only if during that monitoring time interval a specified number of repeated indications of the first indication accrue, the implementation delivers the first indication, but suppresses delivering the remaining accrued indications during the monitoring time interval, and then waits for a separately specified delay time interval. After that, or if the specified number of repeated indications did not accrue during the monitoring time interval, the cycle is repeated, using the next accruing repeated indication as the next first indication.

Note that with this policy it is possible that no indications are actually delivered if the specified number of repeated indications does not accrue during the monitoring time interval.

6.2 Indication filters

6.2.1 General

Indication filters are a special kind of indication gate. The main purposes of indication filters are as follows:

- Indication filters can serve as targets for subscriptions; for details on subscriptions, see 6.4.
- Indication filters filter indications such that only indications within the coverage of the indication filter pass for further processing; for details on defining and exposing the indication filter coverage, see 6.2.2.
- Dynamic indication filters enable clients to establish indication filters with client specified coverage within the implementation; for details, see 6.2.6.
- If defined in profiles, indication filters can represent an implementation's ability to generate respective indications. However, in general it is not possible to conclude from the existence of an indication filter that an implementation actually generates and delivers any indications covered by that indication filter.

The lifecycle of indication filters is controlled by the implementation. For static indication filters (see 6.2.3), this applies without restrictions; the concept of dynamic indication filters (see 6.2.6) provides for clients being able to prompt the implementation for the creation, modification or deletion of dynamic indication filters.

Generally the existence of an indication filter does not imply that any of the indications covered by the indication filter is actually implemented. However, referencing profiles may define amended semantics for indication filters. For details, see 7.3.11.2.

Listeners subscribed to an indication gate must be prepared to process any indication within the coverage of the indication gate.

6.2.2 Indication filter coverage

The coverage of an indication filter is the set of indications that can pass the indication filter; it is specified through an indication filter query statement and a set of namespaces identifications that identify the namespaces out of which indications are filtered. In other words, only indications that originate (see 6.1.2.4) in one of the identified namespaces, and match the query statement pass the indication filter. For details, see 7.3.11.2.

A indication filter query statement identifies source classes, selects properties, and specifies logic that is used to combine instances of those classes containing the selected property values as part of generated indications.

A indication filter query statement is defined using the rules of a query language, for example the CIM Query Language (CQL) (see DSP0202). Profiles that define indication filters specify the exact string that defines the indication filter query statement.

Clients capable of inspecting query statements thereby can learn about the coverage of respective indication filters.

Following are examples of properly formatted CQL indication filter query statements:

EXAMPLE 1:

```
SELECT * FROM CIM_AlertIndication
```

This indication filter query statement covers all alert indications. The selection of all properties exposed by the CIM_AlertIndication class indicates that values of these properties are present in CIM_AlertIndication instances delivered to listeners. However, note that generally the value Null is admissible unless otherwise required.

EXAMPLE 2:

```
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA
CIM_StorageVolume
```

This indication filter query statement covers lifecycle indications reporting the creation of CIM_StorageVolume instances representing newly created storage volumes within the managed environment. This is because the schema definition of the CIM_InstCreation indication states that it indicates the creation of a new CIM instance (of any class), and the `WHERE` clause limits that to instances of the CIM_StorageVolume class.

The selection of all properties exposed by the CIM_InstCreation class indicates that values of these properties are present in CIM_InstCreation instances delivered to listeners. The schema definition of the CIM_InstCreation indication requires that the value of the SourceInstance property contains a copy of the new instance (the CIM_StorageVolume instance in this case). However, with respect to other property values, again note that generally the value Null is admissible unless otherwise required.

EXAMPLE 3:

```
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'DMTF' AND
MessageID = 'SVPC0123'
```

This indication filter query statement covers one alert indication. The related event is defined by an alert message defined in a message repository. The value of the `OwningEntity` property identifies DMTF as the organization owning the message registry. The value of the `MessageID` property allows identifying the alert message within the owning organization; for details, see 7.3.31.

EXAMPLE 4:

```
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'DMTF' AND
MessageID LIKE 'SVPC0123|SVPC0124|SVPC0125'
```

This indication filter query statement covers a closed set of alert indications. Note that the use of the `LIKE` expression implies "full like extended regular expressions" as defined in DSP0202.

6.2.3 Static indication filters

Static indication filters are provided by an implementation, that is, their lifecycle and coverage is controlled solely by the implementation, and clients are not able to create or delete static indication filters.

Profiles define the requirements for the CIM representation of static indication filters along with a requirement level, such as mandatory, conditional, or optional. In addition, WBEM servers may expose CIM_IndicationFilter instances representing static indication filters that are not defined by a profile.

Profiles define the coverage of static indication filters (that is, the set of covered indications) through a query statement (see 6.2.2). There is a certain degree of flexibility in defining the indication filter coverage by means of a query statement:

- Indication filters that cover more than one indication

  A referencing profile might require an indication filter of this kind in the case where one or more indications covered by that indication filter are implemented.

- Indication filters that cover exactly one indication

  This is achieved by specifying a "WHERE" clause as part of the indication filter query statement that restricts the selected indication class to one particular indication. A referencing profile might require an indication filter of this kind for the case "if and only if" the covered indication is implemented. Only in this very special case clients that are aware of that profile definition upon detection of the representation of that particular indication filter would know that the covered indication is actually implemented.

Static indication filters are uniquely identified by means of a naming convention that involves the name of the organization defining the profile, the name of this profile and a string that is required to be unique within the implementation of this profile; for details, see 7.3.12.

Filter collections provide a means for aggregating the coverage of indication filters and other filter collections; see 6.3.

6.2.4 Indication-specific indication filters

Indication-specific filters address the needs of clients requiring notifications about events reported by particular indications specified in a profile. Indication-specific indication filters are a specialization of static indication filters, and are designed to cover one or more of the indications specified in a referencing profile or in this profile. For details, see 7.3.15.

One central purpose of indication-specific indication filters is contributing to the defined coverage of profile-specific filter collections; see 6.3.3.

6.2.5 Global indication filters

Global indication filters address the needs of clients requiring notifications about large sets of events, irrespective of a profile context. Global indication filters are a specialization of static indication filters (see 6.2.3), and are designed to cover large sets of indications, such as:

- All alert indications
- All lifecycle indications reporting the creation of a CIM instance
- All lifecycle indications reporting the modification of a CIM instance
- All lifecycle indications reporting the deletion of a CIM instance

For details, see 7.3.16.

6.2.6 Dynamic indication filters

The creation, deletion and modification of dynamic indication filters can be requested by clients and is then performed by the implementation. If suitable static indication filters do not exist within an implementation, clients can request the creation of dynamic indication filters with a coverage that is specifically tailored to the notification requirements of one or more listeners. However, the implementation of dynamic indication filters is expensive. Not all implementations, especially footprint-sensitive implementations, will be able to implement dynamic indication filters. For that reason this profile models dynamic indication filters in the form of the optional DynamicIndicationFilters feature; for details, see 7.2.1

Even if dynamic indication filters are implemented, clients should first look for existing indication filters or filter collections that might satisfy listener notification requirements, before attempting to create a dynamic indication filter. Adding unnecessary dynamic indication filters may adversely affect the performance of indication delivery by the implementation.

6.3 Filter collections

6.3.1 General

Filter collections are a special kind of indication gate designed to contain other indication gates; the contained indication gates may or may not be represented in CIM.

This profile only models static filter collections (see 6.3.3). Dynamic filter collections, that is, filter collections that could be created, deleted and modified by clients, are not addressed by this profile.

The main purposes of filter collections are:

- Filter collections can serve as targets for subscriptions; for details on subscriptions, see 6.4.
- Filter collections filter indications according to their coverage; for details on defining and exposing the coverage of filter collections, see 6.3.2.
- If defined in profiles, filter collections can represent an implementation's ability to generate respective indications. However, in general it is not possible to conclude from the existence of a filter collection that an implementation actually generates and delivers any indications covered by that filter collection.

6.3.2 Filter collection coverage

The coverage of a filter collection determines the actual filtering rules for that filter collection; it is defined as the aggregated coverage of all contained indication gates. For details, see 7.3.17.2.

6.3.3 Static filter collections

6.3.3.1 General

Static filter collections are filter collections whose lifecycle is controlled by the implementation, that are uniquely identifiable, and for which a defined coverage can be established.

6.3.3.2 Unique identification

Unique identification of static filter collections is achieved through establishing a naming convention. The naming convention enables clients to identify static filter collections about which they have prior knowledge. For details on specifying the unique identification, see 7.3.17.4.2.

6.3.3.3 Defined coverage

The concept of the defined coverage addresses the need to reduce the memory footprint of embedded implementations. It allows defining the coverage of static filter collections by means of specification in profiles, but without requiring the CIM representation of contained indication gates. The knowledge about the defined coverages of static filter collections specified in profiles can be built into clients, such that the clients know the coverage of those static filter collections in advance, instead of determining the coverage through the inspection of the CIM representation of contained indication gates. For details on specifying the defined coverage of static filter collections, see 7.3.17.3.

6.3.3.4 Profile specific filter collections

Profile-specific filter collection address the needs of clients requiring notifications about events reported by the indications specified in a particular profile. Profile specific filter collections are a specialization of static filter collections. The defined coverage of a profile-specific filter collection covers all indications of a particular type (that is, all alert indications or all lifecycle indications) defined in a profile. For details, see 7.3.21.

6.3.3.5 Global filter collections

Global filter collections address the needs of clients requiring notifications about large sets of events. Global filter collections are a specialization of static filter collections.

The defined coverage of global filter collections covers large sets of indications, such as

- All alert indications
- All alert indications specified in profiles
- All lifecycle indications
- All indications specified in profiles
- All alert indications specified in profiles
- All lifecycle indications specified in profiles

For details, see 7.3.22.

6.4 Subscriptions, listeners, and listener destinations

6.4.1 Subscriptions

Subscriptions model a mechanism that enables clients to register listeners at an indication gate for the delivery of indications that are within the coverage of that indication gate.

Clients need to perform three steps in order to subscribe a listener for the delivery of indications:

1) Determine if there is an existing indication gate covering the desired indication set. If an appropriate indication gate does not exist, and the support for dynamic indication filters is implemented, the client could create dynamic indication filters (see 6.2.6).

2) Determine if a listener destination referencing the listener already exists within the implementation. If such a listener destination does not yet exist, and the support for creating or modifying listener destinations is implemented, the client could create a new listener destination or modify an existing listener destination.

3) Create a subscription that relates the listener destination with the indication gate.

After it is created, a subscription results in indications being delivered to the listener that is referenced by the listener destination for each event reported through any of the indications covered by the indication gate referenced by the subscription.

6.4.2 Overlapping coverages of subscriptions

This profile does not specify any rules prohibiting that a listener simultaneously is subscribed to several indication gates with overlapping coverages.

For example, a listener could simultaneously be subscribed to a filter collection and to an indication filter contained by that filter collection. As another example, a listener could simultaneously be subscribed to two or more unrelated indication filters that are defined in the same or in different profiles and where the coverages as defined by respective query statements overlap.

If separate subscriptions to indication gates with overlapping coverages exist, indications are independently delivered for each individual subscription. This can result in multiple indications being delivered to the listener for the same event. The semantical requirements pertaining to the delivery of indications to subscribed listener destinations are detailed in 7.3.23.2 and 7.3.25.2.

6.4.3 Subscription management authorization

This profile makes no explicit provisions for managing the permissions of a client with respect to its ability to create, modify, or delete subscriptions. Any coordination between clients, or between a client and access management, to govern the ability of one client to make changes that affect the delivery of indications delivered to a listener is outside the scope of this profile.

6.4.4 Listeners

A listener is a WBEM listener that implements applicable portions of this profile. Listeners can be subscribed at an implementation for the delivery of specific sets of indications as exposed by indication gates within that implementation. After a subscription is established within an implementation, indications are delivered to subscribed listeners as respective events occur, and the listeners need to receive and process these indications.

In general, a listener is different from the client that establishes its representation within the implementation in the form of a respective listener destination (see 6.4.5); however, clients that also implement listener functionality can establish themselves as listeners.

6.4.5 Listener destinations

A listener destination is an entity that maintains a reference to a listener within an implementation, including information about the protocol applicable to contact the listener; for details, see 7.3.23.

A free listener destination is a listener destination that does not currently reference a listener. Clients are enabled to establish a reference to a particular listener; for details, see 7.3.23.3.6.

The implementation is responsible for delivering the indications that are passed from any indication gate to any listener referenced by a listener destination that is subscribed to that indication gate. The semantical requirements pertaining to the delivery of indications to subscribed listener destinations are detailed in 7.3.23.2 and 7.3.25.2.

Implementations provide functionality enabling clients to control the lifecycle of listener destinations (for example, their creation and destruction), or provide a set of predefined listener destinations along with functionality enabling clients to modify these to refer to different listeners, or provide a combination of both approaches.

The second approach requiring the modification of predefined listener destinations is inherently unsafe because activities of different clients can overlap, and race conditions can occur; for that reason the create/delete based approach should be favored.

6.5 Indication service and implementation

6.5.1 Implementation

An implementation is the realization of applicable portions of this profile within a WBEM server. Within implementations, the functionality defined in this profile may be divided into common parts and referencing profile related parts; for details, see 7.1.

6.5.2 Indication service

An indication service is a component within an implementation that is responsible for delivering indications to listeners. An indication service manages elements such as listener destinations (see 6.4.3) and subscriptions (see 6.4.1), and it may provide support for reliable indication delivery (see 6.1.5) and for dynamic indication filters (see 6.2.6).

6.6 Indication system and referencing profiles

An indication system is a system that hosts a WBEM server with one or more indication services.

NOTE The current version of this profile allows only one indication service per indication system; the limitation may be raised in a future version of this profile.

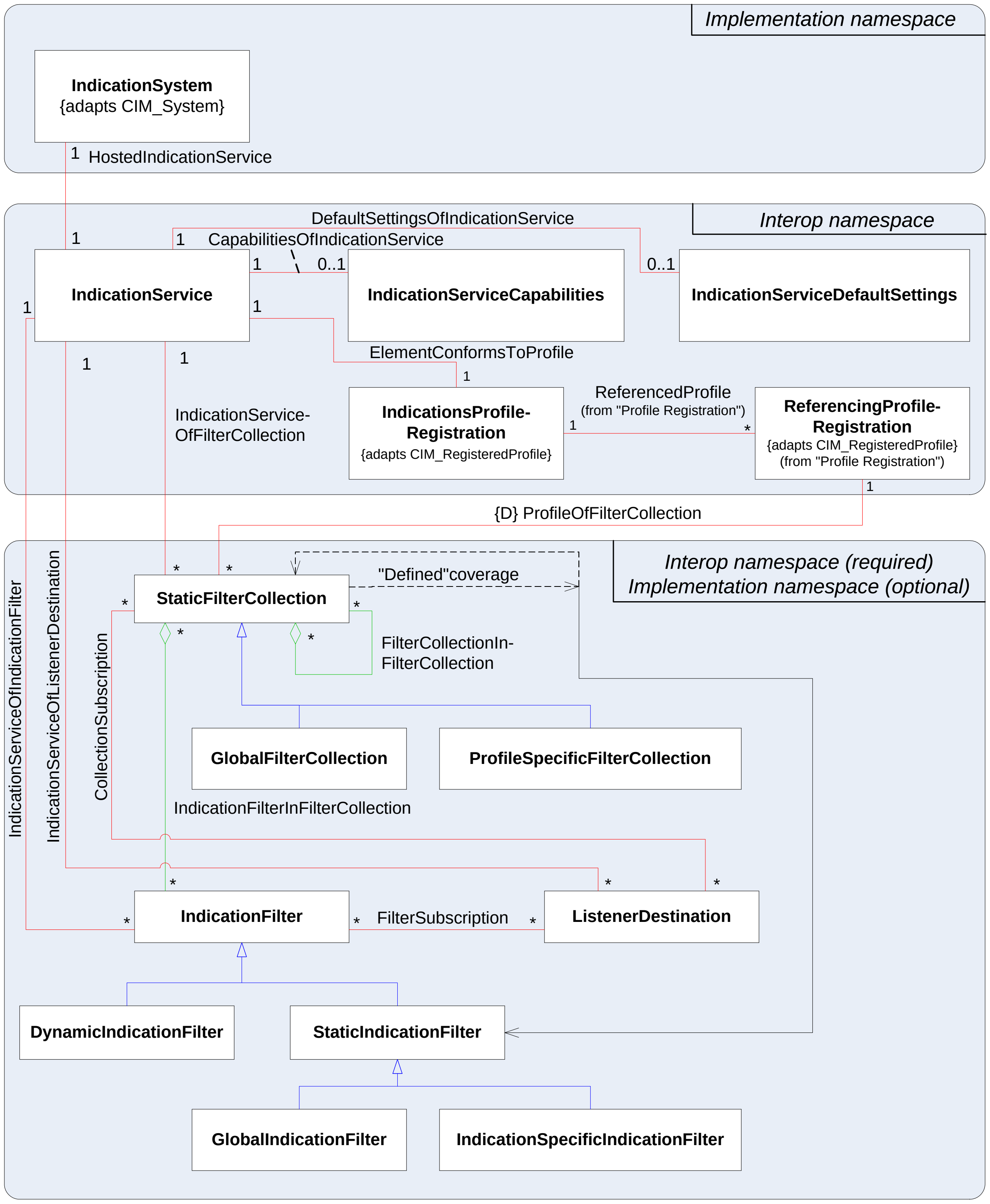
In the general case, the scoping systems of referencing profiles are different from the indication system, that is, they are different from the system hosting the WBEM server. In other words, referencing profiles are not required to provide the scope for the indication service, and the central class adaptation of a referencing profile is not required to model the system that hosts the indication service. For that reason, this profile requires that the central class profile advertisement methodology as defined in DSP1033 is applied for advertising this profile; for details, see 7.3.6.

For example, consider an Example Fan profile that defines a central Fan adaptation of the CIM_Fan class modeling fans and also defines indications reporting events related to fans and their related elements; in this case the systems containing the fans are not required to be indication systems; particularly, they are not required to host an indication service.

As a second example, consider an Example Virtual System profile that defines a central VirtualSystem adaptation of the CIM_ComputerSystem class modeling virtual systems and also defines indications reporting events related to virtual systems and their components; again, the virtual systems are not required to be indication systems, that is, they are not required to host an indication service.

980 6.7 CIM model

981 Figure 2 shows the DMTF adaptation diagram for this profile.


Implementation namespace
IndicationSystem {adapts CIM_System}
1 HostedIndicationService
Interop namespace
DefaultSettingsOfIndicationService
CapabilitiesOfIndicationService
IndicationService
IndicationServiceCapabilities
IndicationServiceDefaultSettings
0..1
ElementConformsToProfile
IndicationProfile-Registration {adapts CIM_RegisteredProfile}
ReferencedProfile (from "Profile Registration")
ReferencingProfile-Registration {adapts CIM_RegisteredProfile} (from "Profile Registration")
IndicationService-OfFilterCollection
{D} ProfileOfFilterCollection
Interop namespace (required)
Implementation namespace (optional)
StaticFilterCollection
"Defined"coverage
FilterCollectionIn-FilterCollection
GlobalFilterCollection
ProfileSpecificFilterCollection
IndicationFilterInFilterCollection
IndicationServiceOfIndicationFilter
IndicationServiceOfListenerDestination
CollectionSubscription
IndicationFilter
FilterSubscription
ListenerDestination
DynamicIndicationFilter
StaticIndicationFilter
GlobalIndicationFilter
IndicationSpecificIndicationFilter

982

983 **Figure 2 – Indications Profile: DMTF class adaptation diagram**

The most essential adaptations defined in this profile are listed below, along with their modeled managed object types:

- the IndicationService adaptation (see 7.3.2) models indication services as described in 6.5.2
- the IndicationFilter adaptation (see 7.3.11) models indication filters as described in 6.2
- the StaticFilterCollection adaptation (see 7.3.17) models static filter collections as described in 6.3
- the StaticIndicationFilter adaptation (see 7.3.17) models static indication filters as described in 6.2.3
- the ListenerDestination adaptation (see 7.3.23) models listener destinations as described in 6.4.3
- the AbstractSubscription adaptation (see 7.3.25) models subscriptions as described in 6.4.1

Instances of most of these adaptations are instantiated in the Interop namespace; the use of the Interop namespace (see DSP1033) makes it easier for clients to detect the CIM representations of respective managed objects.

DEPRECATED

The ProfileOfFilterCollection association adaptation models the relationship between filter collections and the registration of this profile.

NOTE The ProfileOfFilterCollection association adaptation (defined as the CIM_ConcreteDependency "profile class" in version 1.1 of this profile) is deprecated in version 1.2 of this profile in favor of a naming convention for static filter collections that enables their unique identification.

DEPRECATED

Figure 3 depicts the adaptations of indication classes defined by this profile along with the adapted indication classes.

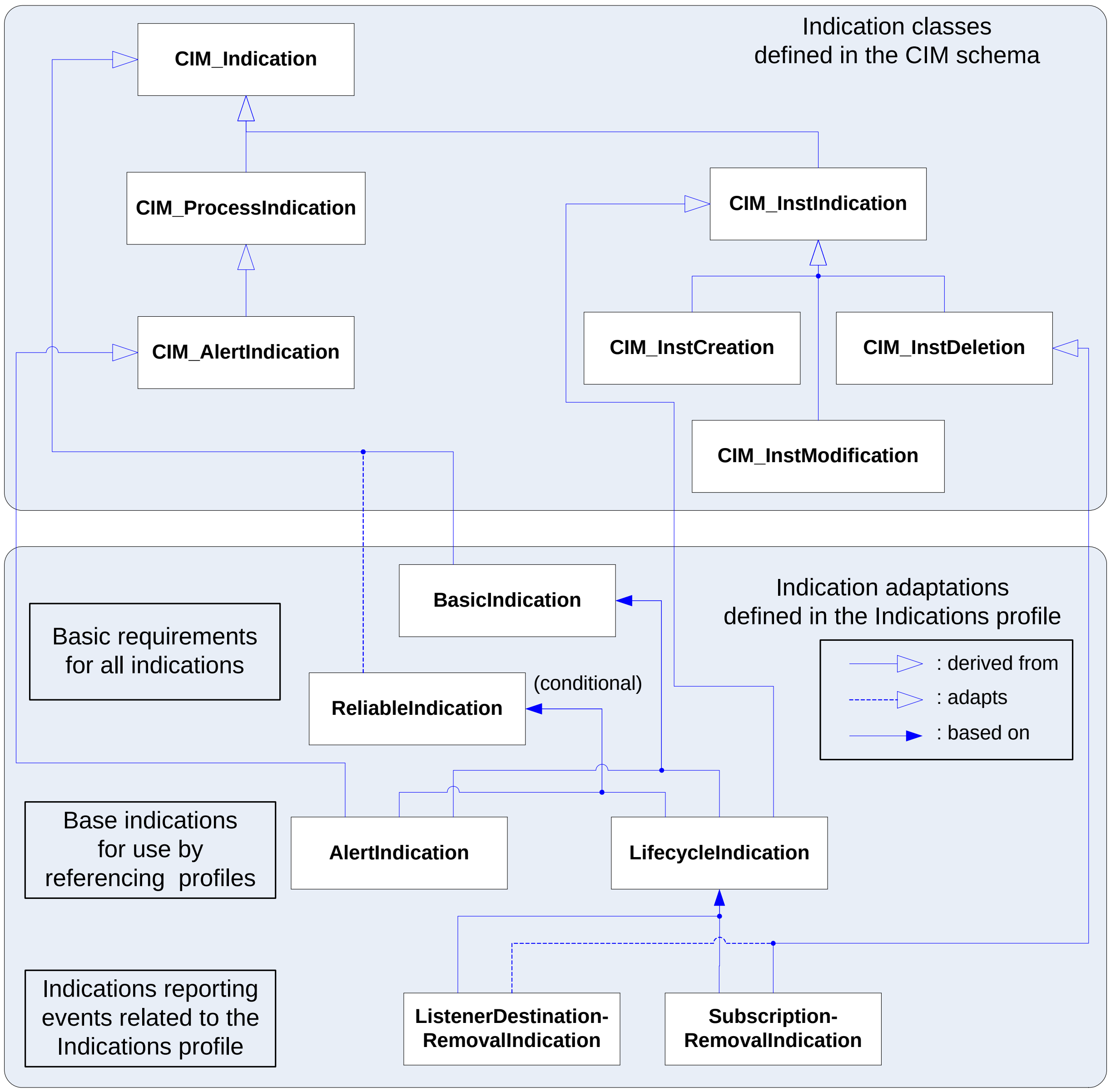

Indication classes defined in the CIM schema
CIM_Indication
CIM_ProcessIndication
CIM_AlertIndication
CIM_InstIndication
CIM_InstCreation
CIM_InstDeletion
CIM_InstModification
Indication adaptations defined in the Indications profile
Basic requirements for all indications
BasicIndication
ReliableIndication
(conditional)
: derived from
: adapts
: based on
Base indications for use by referencing profiles
AlertIndication
LifecycleIndication
Indications reporting events related to the Indications profile
ListenerDestination-RemovalIndication
Subscription-RemovalIndication

Figure 3 – Indications Profile: Indication adaptations and adapted indication classes

The most essential indication adaptations defined in this profile are listed below, along with their modeled indications:

- the BasicIndication adaptation (see 7.3.29) models indications as described in 6.1.2
- the ReliableIndication adaptation (see 7.3.30) models reliable indications as described in 6.1.5; this adaptation specifies additional optional requirements that can be implemented separately from the requirements of other indication adaptations.

- the AlertIndication adaptation (see 7.3.31) models alert indication as described in 6.1.2.2; it is an abstract adaptation available to referencing profiles in order to define their own alert indications
- the LifecycleIndication adaptation (see 7.3.32) models lifecycle indications as described in 6.1.2.3; it is an abstract adaptation available to referencing profiles in order to define their own lifecycle indications.

7 Implementation

7.1 Separation of requirements

This profile defines implementation requirements for implementations (for example, WBEM servers implementing this profile) and for listeners (for example, WBEM listeners implementing this profile).

The implementation requirements for implementations are further separated into WBEM server related requirements and referencing profile related requirements, as follows:

- Requirements that address the infrastructure for the delivery of indications (including the management of listener destinations and subscriptions) are WBEM server related requirements, and are typically implemented only once within an implementation.
- Requirements that address the generation of indications are related to the referencing profile defining those indications, and are typically implemented as part of the implementation of that referencing profile.
- Requirements that address functionality related to indication filters and filter collections are referencing profile related requirements.

  However, WBEM servers may contain other facilities allowing implementations of referencing profiles to delegate some of their implementation responsibilities to these facilities. For example, within WBEM servers providing a CIM instance repository the implementations of referencing profiles can delegate storing indication filters and filter collections to the CIM instance repository, such that in this case the implementation requirements for referencing profiles are effectively reduced to storing respective objects into the repository when the implementation of the referencing profile is installed.

In this profile WBEM server related implementation requirements are marked with a phrase such as the following:

"The requirements in this subclause are WBEM server related implementation requirements."

In this profile referencing profile related implementation requirements are marked with a phrase such as the following:

"The requirements in this subclause are referencing profile related implementation requirements."

This facilitates explicit distinction of WBEM server related implementation requirements as opposed to requirements related to the implementation of referencing profiles.

7.2 Features

7.2.1 DynamicIndicationFilters

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The implementation of the DynamicIndicationFilters feature provides functionality for dynamic indication filters; for a description of dynamic indication filters, see 6.2.6.

The granularity of the DynamicIndicationFilters feature is per IndicationService instance (see 7.3.2).

The requirement level of the DynamicIndicationFilters feature is optional.

The implementation of the DynamicIndicationFilters feature for a particular IndicationService instance is indicated by a value of True for the FilterCreationEnabled property.

7.2.2 IndicationServiceInitialSettingsExposed

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The implementation of the IndicationServiceInitialSettingsExposed feature provides information about the initial settings of an indication service.

The granularity of the IndicationServiceInitialSettingsExposed feature is per IndicationService instance (see 7.3.2).

The requirement level of the IndicationServiceInitialSettingsExposed feature is optional.

The availability of the IndicationServiceInitialSettingsExposed feature for a particular IndicationService instance is indicated by the presence of an IndicationServiceInitialSettings instance (see 7.3.9) associated through an InitialSettingsOfIndicationService instance (see 7.3.10).

7.2.3 IndicationServiceModification

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The implementation of the IndicationServiceModification feature provides functionality for client requested dynamic modification of an indication service.

The granularity of the IndicationServiceModification feature is per IndicationService instance (see 7.3.2).

The requirement level of the IndicationServiceModification feature is optional.

The availability of the IndicationServiceModification feature for a particular IndicationService instance is indicated if an IndicationServiceCapabilities (see 7.3.7) instance representing the capabilities of the represented indication service exists and is associated via the CapabilitiesOfIndicationService association (see 7.3.8), and in that instance the value True is set for any of the following properties: FilterCreationEnabledIsSettable, DeliveryRetryAttemptsIsSettable, DeliveryRetryIntervalIsSettable, SubscriptionRemovalActionIsSettable, or SubscriptionRemovalTimeIntervalIsSettable.

7.2.4 ReliableIndications

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The implementation of the ReliableIndications feature provides functionality for reliable indications as described in 6.1.5. For further details, see 7.3.30 and 7.4.

The granularity of the ReliableIndications feature is per IndicationService instance (see 7.3.2).

The requirement level of the ReliableIndications feature is optional. The implementation of the ReliableIndications feature is also optional for listeners; in this case, the granularity is once per listener, and the discovery mechanism does not apply.

The availability of the ReliableIndications feature for a particular IndicationService instance is indicated by a value larger than 0 for the DeliveryRetryAttempts property.

7.2.5 SuppressRepeatNotificationPolicy

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The implementation of the SuppressRepeatNotificationPolicy feature provides functionality for suppressing repeated indication delivery by implementing the "suppress repeated indication delivery avoidance policy", as described in 6.1.6.3.

The granularity of the SuppressRepeatNotificationPolicy feature is per implementation.

The requirement level of the SuppressRepeatNotificationPolicy feature is optional.

The availability of the SuppressRepeatNotificationPolicy feature is indicated by the value 3 (Suppress) for the RepeatNotificationPolicy property in AbstractSubscription instances (see 7.3.25) representing existing subscriptions.

NOTE The discovery mechanism specified here is only rudimentary because the feature presence can only be discovered if at least one exploiting subscription is discovered. A future version of this profile is expected to introduce a new property into the CIM_IndicationServiceCapabilities class that indicates the presence of the feature per indication service.

7.2.6 DelayRepeatNotificationPolicy

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The implementation of the DelayRepeatNotificationPolicy feature provides functionality for suppressing repeated indication delivery by implementing the "delayed indication delivery avoidance policy", as described in 6.1.6.4.

The granularity of the DelayRepeatNotificationPolicy feature is per implementation.

The requirement level of the DelayRepeatNotificationPolicy feature is optional.

The availability of the DelayRepeatNotificationPolicy feature is indicated by the value 4 (Delay) for the RepeatNotificationPolicy property in AbstractSubscription instances (see 7.3.25) representing existing subscriptions.

NOTE The discovery mechanism specified here is only rudimentary because the feature presence can only be discovered if at least one exploiting subscription is discovered. A future version of this profile is expected to introduce a new property into the CIM_IndicationServiceCapabilities class that indicates the presence of the feature per indication service.

7.2.7 IndividualFilterSubscription

The implementation of the IndividualFilterSubscription feature provides functionality for subscriptions to individual indication filters.

The granularity of the IndividualFilterSubscription feature is per IndicationFilter instance (see 7.3.11).

The requirement level of the IndividualFilterSubscription feature is optional.

The availability of the IndividualFilterSubscription feature for a particular IndicationFilter instance is indicated by the value True for the IndividualSubscriptionSupported property.

7.2.8 FilterCollectionCoverageExposure

The implementation of the FilterCollectionCoverageExposure feature provides functionality for exposing the coverage of static filter collections.

The granularity of the FilterCollectionCoverageExposure feature is per StaticFilterCollection instance (see 7.3.17).

The requirement level of the FilterCollectionCoverageExposure feature is optional.

The availability of the FilterCollectionCoverageExposure feature for a particular StaticFilterCollection instance is indicated through at least one instance of either the IndicationFilterInFilterCollection association adaptation (see 7.3.19) or the FilterCollectionInFilterCollection association adaptation (see 7.3.20) referencing the StaticFilterCollection instance.

7.3 Adaptations

7.3.1 Conventions

This profile repeats the effective values of certain Boolean qualifiers as part of property requirements, or of method parameter requirements. The following convention is established: If the name of a qualifier is listed, its effective value is True; if the qualifier name is not listed, its effective value is False. The convention is applied in the following cases:

- In: indicates that the parameter is an input parameter
- Out: indicates that the parameter is an output parameter
- Key: indicates that the property is a key (that is, its value is part of the instance part)
- Required: indicates that the element value shall be non-Null

This profile defines operation requirements based on DSP0223.

For adaptations of ordinary classes and of associations the implementation requirements for operations are specified in adaptation-specific subclauses of 7.3.

7.3.2 IndicationService: CIM_IndicationService

7.3.2.1 General

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The IndicationService adaptation models indication services; indication services are described in 6.5.2.

The implementation type of the IndicationService adaptation is: "instantiated".

The IndicationService adaptation shall conform to the requirements for "central classes" defined in the Profile Registration profile; for details, see DSP1033.

7.3.2.2 Initial behavior

If the IndicationServiceInitialSettingsExposed feature (see 7.2.2) is implemented, the initial behavior of an indication service shall be as exposed by the IndicationServiceInitialSettings instance (see 7.3.9) that is associated with the IndicationService instance representing that indication service through an InitialSettingsOfIndicationService instance (see 7.3.10).

If the IndicationServiceInitialSettingsExposed feature (see 7.2.2) is not implemented, then the initial behavior of the indication service shall be as follows:

- Retry the delivery of an indication after a delivery failure three additional times, each time waiting 20 seconds before the retry, and indicate this behavior with a value of 3 for the DeliveryRetryAttempts property (see 7.3.2.3.3) and the value 20 for the DeliveryRetryInterval property (see 7.3.2.3.4) in the IndicationService instance representing the indication service

- Remove affected subscriptions after 30 days, and indicate this behavior with a value of 2 (Remove) for the SubscriptionRemovalAction property (see 7.3.2.3.5), and a value of 2,592,000 seconds (30 days) for the SubscriptionRemovalTimeInterval property (see 7.3.2.3.6) in the IndicationService instance representing the indication service

NOTE With respect to the availability of DynamicIndicationFilters feature (see 7.2.1) as indicated by the value of the FilterCreationEnabled property an recommended initial behavior is not established; instead the implementation is required to always expose the available behavior; see 7.3.2.3.2.

7.3.2.3 Element requirements

7.3.2.3.1 General

Table 4 lists the element requirements for the IndicationService adaptation.

Table 4 – IndicationService: Element requirements

Elements	Requirement	Description
Properties		
Name	Mandatory	**Key**: See CIM schema definition.
CreationClassName	Mandatory	**Key**: See CIM schema definition.
SystemName	Mandatory	**Key**: See CIM schema definition.
SystemCreationClassName	Mandatory	**Key**: See CIM schema definition.
FilterCreationEnabled	Mandatory	See 7.3.2.3.2.
DeliveryRetryAttempts	Mandatory	See 7.3.2.3.3.
DeliveryRetryInterval	Mandatory	See 7.3.2.3.4.
SubscriptionRemovalAction	Mandatory	See 7.3.2.3.5.
SubscriptionRemovalTimeInterval	Mandatory	See 7.3.2.3.6.
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.
GetAssociatedInstancesWithPath( )	Mandatory	See DSP0223.
GetAssociatedInstancePaths( )	Mandatory	See DSP0223.
GetReferencingInstancesWithPath( )	Mandatory	See DSP0223.
GetReferencingInstancePaths( )	Mandatory	See DSP0223.
ModifyInstance( )	Conditional	See 7.3.2.3.7 and DSP0223.

If the ModifyInstance( ) operation is implemented (see 7.3.2.3.7), the values of some properties might be modifiable through client requests; see 7.3.7 for details on indicating those properties whose values are actually modifiable.

7.3.2.3.2 Property: FilterCreationEnabled

The value of the FilterCreationEnabled property shall reflect whether the DynamicIndicationFilters feature (see 7.2.1) is available for the IndicationService instance. A value of False indicates that the feature is not available; a value of True indicates that the feature is available.

7.3.2.3.3 Property: DeliveryRetryAttempts

The value of the DeliveryRetryAttempts property shall reflect the number of times that the implementation is going to retry the delivery of an indication to a particular listener in the case of delivery failures. This value does not include the initial delivery attempt.

A value larger than 0 indicates that the ReliableIndications feature (see 7.2.4) is available. The value 0 indicates that the ReliableIndications feature is not available.

7.3.2.3.4 Property: DeliveryRetryInterval

The value of the DeliveryRetryInterval property shall reflect the minimal time interval in seconds that the implementation waits before delivering an indication to a particular listener destination after a previous delivery failure.

7.3.2.3.5 Property: SubscriptionRemovalAction

The value of the SubscriptionRemovalAction property shall reflect the removal action for subscriptions after two failed indication deliveries where the time interval between the failed deliveries, without any intermediate successful indication delivery, exceeds the timeout reflected by the value of the SubscriptionRemovalTimeInterval property.

7.3.2.3.6 Property: SubscriptionRemovalTimeInterval

The value of the SubscriptionRemovalTimeInterval property shall reflect the minimum time interval that implementations shall wait after two failed indication deliveries without any intermediate successful indication delivery, before performing the activity designated by the value of the SubscriptionRemovalAction property.

7.3.2.3.7 Method: ModifyInstance( )

The implementation of the ModifyInstance( ) operation enables clients to modify aspects of the behavior of the represented indication service.

The requirement level of the ModifyInstance( ) operation is conditional.

Condition: The IndicationServiceModification feature is implemented; for a description, see 7.2.3.

Information about which properties are modifiable is provided by an IndicationServiceCapabilities instance that is associated to the IndicationService instance representing the indication service; see 7.3.7 and 7.3.8.

Table 5 lists the error reporting requirements for the ModifyInstance( ) operation on IndicationService instances. If any of the error situations described in the Description column of Table 5 matches, the operation shall fail and the corresponding CIM status code shall be returned. In addition, the error reporting requirements defined in DSP0223 for the ModifyInstance( ) operation apply.

Table 5 – ModifyInstance( ): Error reporting requirements

Reporting mechanism	Requirement level	Description
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the behavior requested by the value of the FilterCreationEnabled property in the input IndicationService instance, as described in 7.3.2.3.2.
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the behavior requested by the value of the DeliveryRetryAttempts property in the input IndicationService instance, as described in 7.3.2.3.3.

CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the delivery retry interval requested by the value of the DeliveryRetryInterval property, as described in 7.3.2.3.4.
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the subscription removal action requested by the value of the SubscriptionRemovalAction property in the input IndicationService instance, as described in 7.3.2.3.5.
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the subscription removal time interval requested by the value of the SubscriptionRemovalTimeInterval property in the input IndicationService instance, as described in 7.3.2.3.6.
CIM_ERR_NOT_SUPPORTED	Mandatory	The IndicationServiceModification feature is not implemented; see 7.2.3 and 7.3.7.
CIM_ERR_FAILED	Mandatory	The IndicationServiceModification feature is not available for the IndicationService instance; see 7.2.3 and 7.3.7.

If the ModifyInstance( ) operation is successful, the requested modification on the indication service shall be applied, and — as a consequence — shall be reflected in all IndicationService instances that represent the modified indication service and are exposed by the implementation.

If the ModifyInstance( ) operation fails, the requested modification on the indication service shall not be applied, and — as a consequence — all IndicationService instances that represent the indication service shall remain unchanged.

7.3.2.4 Instance requirements

Within an implementation there shall be exactly one indication service. That indication service shall be represented by an IndicationService instance in the Interop namespace.

NOTE 1 The reasons for requiring exactly one indication service are a) other elements defined in this profile (such as subscriptions, listener destinations, or dynamic indication filters) require a relationship to the indication service, and b) the modeled use of the CreateInstance( ) operation does not provide for expressing that required relationship at creation time. For these reasons an indication service must be implied at creation time, and the simplest approach for that is allowing just one indication service. Future versions of this profile might lift the single instance restriction, for example by modeling respective creation methods with parameters that enable establishing the required relationship to a specifiable indication service.

NOTE 2 In some places in this profile multiple indication services are mentioned. This is not meant to lift the restriction established in this subclause, but to accommodate the future introduction of multiple indication services.

7.3.3 IndicationSystem: CIM_System

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The IndicationSystem adaptation models indication systems; indication systems are described in 6.6.

The implementation type of the IndicationSystem adaptation is: "instantiated".

The IndicationSystem adaptation shall conform to the requirements for "scoping classes" defined in the Profile Registration profile; for details, see DSP1033.

Table 6 lists the element requirements of the IndicationSystem adaptation.

Table 6 – IndicationSystem: Element requirements

Elements	Requirement	Description
Properties		
Name	Mandatory	**Key**: See CIM schema definition.
CreationClassName	Mandatory	**Key**: See CIM schema definition.
Operations		
Associators( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.
GetAssociatedInstancesWithPath( )	Mandatory	See DSP0223.
GetAssociatedInstancePaths( )	Mandatory	See DSP0223.
GetReferencingInstancesWithPath( )	Mandatory	See DSP0223.
GetReferencingInstancePaths( )	Mandatory	See DSP0223.

7.3.4 HostedIndicationService: CIM_HostedService

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The HostedIndicationService adaptation models the relationship between an indication service and its hosting indication system.

The implementation type of the HostedIndicationService association adaptation is: "instantiated".

Table 7 lists the element requirements for the HostedIndicationService association adaptation.

Table 7 – HostedIndicationService: Element requirements

Elements	Requirement	Description
Properties		
Antecedent	Mandatory	**Key**: Value shall reference the IndicationSystem instance **Multiplicity**: 1
Dependent	Mandatory	**Key**: Value shall reference the IndicationService instance **Multiplicity**: 1
Operations		
GetInstance( )	Mandatory	See DSP0223.

Elements	Requirement	Description
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.

Each IndicationSystem instance (see 7.3.3) shall be associated through a HostedIndicationService instance with the IndicationService instance (see 7.3.2) representing the indication service hosted by the indication system represented by the IndicationSystem instance.

7.3.5 IndicationsProfileRegistration: CIM_RegisteredProfile

7.3.5.1 General

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The IndicationsProfileRegistration adaptation models the profile registration of this profile, that is, the representation of the specific implemented version 1.2.0 of this profile.

The implementation type of the IndicationsProfileRegistration adaptation is: "instantiated".

The specific implemented version of this profile shall be represented by IndicationsProfileRegistration instances in the Interop namespace.

NOTE The existence of an instance of this adaptation indicates that version 1.2.0 of this profile is implemented at least once within the WBEM server.

Table 8 lists the element requirements for the IndicationsProfileRegistration adaptation.

Table 8 – IndicationsProfileRegistration: Element requirements

Elements	Requirement	Description
Base adaptations		
ProfileRegistration::CIM_RegisteredProfile		The IndicationsProfileRegistration adaptation shall conform to the requirements for the CIM_RegisteredProfile "profile class" defined in the Profile Registration profile; see DSP1033.
Properties		
InstanceID	Mandatory	**Key**: See CIM schema definition.
RegisteredName	Mandatory	Value shall be "Indications".
RegisteredVersion	Mandatory	Value shall be "1.2.0".
RegisteredOrganization	Mandatory	Value shall be 2 (DMTF).

NOTE Operation requirements are defined by the base "profile class" CIM_RegisteredProfile defined in DSP1033.

7.3.6 ElementConformsToProfile: CIM_ElementConformsToProfile

The ElementConformsToProfile adaptation models the relationship between an indication service and the profile registration of this profile (see 7.3.5).

The implementation type of the ElementConformsToProfile association adaptation is: "instantiated".

Table 9 lists the element requirements for the ElementConformsToProfile association adaptation.

Table 9 – ElementConformsToProfile: Element requirements

Elements	Requirement	Description
Base adaptations		
Profile Registration::CIM_Element-ConformsToProfile	Mandatory	The ElementConformsToProfile association adaptation shall conform to the requirements for the CIM_ElementConformsToProfile "profile class" defined in the Profile Registration profile; see DSP1033.
Properties		
ConformantStandard	Mandatory	**Key**: Value shall reference the IndicationsProfileRegistration instance **Multiplicity**: 1
ManagedElement	Mandatory	**Key**: Value shall reference the IndicationService instance. **Multiplicity**: 1
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.

Each IndicationService instance (see 7.3.2) shall be associated through an ElementConformsToProfile instance with an IndicationsProfileRegistration instance (see 7.3.5).

NOTE By requiring the implementation of the ElementConformsToProfile adaptation, this profile in fact requires the central class profile advertisement methodology defined in DSP1033. The scoping class profile advertisement methodology is not applicable because the central instances of implementations of referencing profiles will in almost all cases not be identical with the central instance of this profile, that is, the IndicationSystem instance required by 7.3.3. Note that this does not restrict referencing profiles from choosing a different methodology for their profile advertisement.

7.3.7 IndicationServiceCapabilities: CIM_IndicationServiceCapabilities

7.3.7.1 General

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The IndicationServiceCapabilities adaptation models the capabilities of indication services; indication services are described in 6.5.2.

The requirement level of the IndicationServiceCapabilities adaptation is conditional.

Condition: The IndicationServiceModification feature is implemented; see 7.2.3.

The implementation type of the IndicationServiceCapabilities adaptation is: "instantiated".

7.3.7.2 Element requirements

7.3.7.2.1 General

Table 10 lists the element requirements for the IndicationServiceCapabilities adaptation.

Table 10 – IndicationServiceCapabilities: Element requirements

Element	Requirement	Description
Properties		
InstanceID	Mandatory	**Key**: See CIM schema definition.
FilterCreationEnabledIsSettable	Mandatory	See 7.3.7.2.2
DeliveryRetryAttemptsIsSettable	Mandatory	Value shall indicate whether the implementation supports modification of the DeliveryRetryAttempts property of the associated IndicationService instance
DeliveryRetryIntervalIsSettable	Mandatory	Value shall indicate whether the implementation supports modification of the DeliveryRetryInterval property of the associated IndicationService instance
SubscriptionRemovalActionIsSettable	Mandatory	Value shall indicate whether the implementation supports modification of the SubscriptionRemovalAction property of the associated IndicationService instance
SubscriptionRemovalTimeIntervalIs-Settable	Mandatory	Value shall indicate whether the implementation supports modification of the SubscriptionRemovalTimeInterval property of the associated IndicationService instance
MaxListenerDestinations	Mandatory	Value shall indicate the maximum number of listener destinations
MaxActiveSubscriptions	Mandatory	Value shall indicate the maximum number of active subscriptions
SubscriptionsPersisted	Mandatory	Value shall indicate whether subscriptions are persisted across restarts of the indication service
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.
GetAssociatedInstancesWithPath( )	Mandatory	See DSP0223.
GetAssociatedInstancePaths( )	Mandatory	See DSP0223.
GetReferencingInstancesWithPath( )	Mandatory	See DSP0223.
GetReferencingInstancePaths( )	Mandatory	See DSP0223.

7.3.7.2.2 Property: FilterCreationEnabledIsSettable

DEPRECATED

The value of the FilterCreationEnabledIsSettable property shall indicate whether the implementation supports modification of the FilterCreationEnabled property of the associated IndicationService instance.

NOTE Values other than False are deprecated because it does not make sense enabling clients to set values of properties that represent functionality that is either implemented or not implemented.

DEPRECATED

The value of the FilterCreationEnabledIsSettable property should be False, indicating that the implementation does not support the modification of the FilterCreationEnabled property of the associated IndicationService instance.

7.3.8 CapabilitiesOfIndicationService: CIM_ElementCapabilities

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The CapabilitiesOfIndicationService adaptation models the relationship between an indication service and its capabilities.

The requirement level of the CapabilitiesOfIndicationService adaptation is conditional.

Condition: The IndicationServiceModification feature is implemented; see 7.2.3.

The implementation type of the CapabilitiesOfIndicationService association adaptation is: "instantiated".

Table 11 lists the element requirements for the CapabilitiesOfIndicationService association adaptation.

Table 11 – CapabilitiesOfIndicationService: Element requirements

Elements	Requirement	Description
Properties		
ManagedElement	Mandatory	**Key**: Value shall reference the IndicationService instance **Multiplicity**: 1
Capabilities	Mandatory	**Key**: Value shall reference the IndicationServiceCapabilities instance **Multiplicity**: 0..1
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.

Each IndicationService instance (see 7.3.2) shall be associated through a CapabilitiesOfIndicationService instance with at most one IndicationServiceCapabilities instance (see 7.3.7) representing the capabilities of the indication service represented by the IndicationService instance.

7.3.9 IndicationServiceInitialSettings: CIM_IndicationServiceSettingData

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The IndicationServiceInitialSettings adaptation models initial settings for indication services; indication services are described in 6.5.2. The initial settings of an indication service are the settings that apply at the point in time when the WBEM server hosting the indication service initially starts up the indication service.

The requirement level of the IndicationServiceInitialSettings adaptation is conditional.

Condition: The IndicationServiceInitialSettingsExposed feature is implemented; see 7.2.2.

The implementation type of the IndicationServiceInitialSettings adaptation is: "instantiated".

Table 12 lists the element requirements for the IndicationServiceInitialSettings adaptation.

Table 12 – IndicationServiceInitialSettings: Element requirements

Elements	Requirement	Description
Properties		
InstanceID	Mandatory	**Key**: See CIM schema definition.
FilterCreationEnabled	Mandatory	Value shall be the initial value for the FilterCreationEnabled property in the associated IndicationService instance; the requirements of 7.3.2.3.3 apply.
DeliveryRetryAttempts	Mandatory	Value shall be the initial value for the DeliveryRetryAttempts property in the associated IndicationService instance; the requirements of 7.3.2.3.4 apply.
SubscriptionRemovalAction	Mandatory	Value shall be the initial value for the SubscriptionRemovalAction property in the associated IndicationService instance; the requirements of 7.3.2.3.5 apply.
SubscriptionRemovalTimeInterval	Mandatory	Value shall be the initial value for the SubscriptionRemovalTimeInterval property in the associated IndicationService instance; the requirements of 7.3.2.3.5 apply.
SubscriptionRemovalTimeInterval	Mandatory	Value shall be the initial value for the SubscriptionRemovalTimeInterval property (see 7.3.2.3.6) in the associated IndicationService instance
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.
GetAssociatedInstancesWithPath( )	Mandatory	See DSP0223.
GetAssociatedInstancePaths( )	Mandatory	See DSP0223.
GetReferencingInstancesWithPath( )	Mandatory	See DSP0223.
GetReferencingInstancePaths( )	Mandatory	See DSP0223.

The initial settings of an indication service shall be represented by an IndicationServiceInitialSettings instance in the Interop namespace.

7.3.10 InitialSettingsOfIndicationService: CIM_ElementSettingData

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The InitialSettingsOfIndicationService association adaptation models the relationship between an indication service and its initial settings; indication services are described in 6.5.2.

The requirement level of the InitialSettingsOfIndicationService association adaptation is conditional.

Condition: The IndicationServiceInitialSettingsExposed feature is implemented; see 7.2.2.

The implementation type of the InitialSettingsOfIndicationService association adaptation is: "instantiated".

Table 13 lists the element requirements for the InitialSettingsOfIndicationService association adaptation.

Table 13 – InitialSettingsOfIndicationService: Element requirements

Elements	Requirement	Description
Properties		
ManagedElement	Mandatory	**Key**: Value shall reference an IndicationService instance **Multiplicity**: 1
SettingData	Mandatory	**Key**: Value shall reference the IndicationServiceInitialSettings instance **Multiplicity**: 0..1
IsDefault	Mandatory	Value shall be 1 (Is Default)
IsNext	Mandatory	Value shall be 1 (Is Next)
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.

Each IndicationService instance (see 7.3.2) shall be associated through a InitialSettingsOfIndicationService instance with at most one IndicationServiceInitialSettings instance (see 7.3.9) representing the initial settings of the indication service represented by the IndicationService instance.

7.3.11 IndicationFilter: CIM_IndicationFilter

7.3.11.1 General

The requirements in this subclause are referencing profile and WBEM server related implementation requirements.

The IndicationFilter adaptation models indication filters; indication filters are described in 6.2.

The implementation type of the IndicationFilter adaptation is: "abstract".

7.3.11.2 Semantical requirements

For a particular indication filter the implementation shall filter any indication generated by (indication-specific parts of) the implementation that is within the coverage of the indication filter, that is, that meets both of the following requirements:

- it matches the query statement (see 7.3.11.3.5) given by the value of the Query property in the IndicationFilter instance representing the indication filter
- its indication origin (see 6.1.2.4) is one of the local namespaces identified by the value of the SourceNamespaces[ ] array property in that instance, or, in case that value is NULL, is the local namespace in which the IndicationFilter instance representing the indication filter resides

For the particular indication filter the implementation shall ignore any generated indication that does not meet these requirements.

Indications that passed an indication filter need to be further processed; see the requirements on the IndicationFilterName property defined in 7.3.29.4.2, and the semantical requirements on listener destinations defined in 7.3.23.2, and on subscriptions defined in 7.3.25.2. If implemented, the requirements for reliable indications as defined in 7.3.30 and 7.4 may apply.

Note that the indication filter semantics apply regardless of which profile specified the indications and indication filters; thus an indication specified in one referencing profile is required to be considered by indication filters specified in that referencing profile, but also by those specified in any other referencing profile or in this profile and by those not specified in any profile.

The indication filter semantics defined in this subclause do not require that an implementation implements any of the indications within the coverage of an indication filter. However, referencing profiles may define additional semantics for indication filters they define, including the case that the existence of a particular IndicationFilter instance indicates that one or all indications within the coverage of the represented indication filter are implemented. Of course, this approach is only feasible if the coverage covers one or just a few indications.

7.3.11.3 Element requirements

7.3.11.3.1 General

Table 14 lists the element requirements for the IndicationFilter adaptation.

Table 14 – IndicationFilter: Element requirements

Elements	Requirement	Description
Properties		
Name	Mandatory	**Key**: See 7.3.11.3.2.
CreationClassName	Mandatory	**Key**: See CIM schema definition.
SystemName	Mandatory	**Key**: See CIM schema definition.
SystemCreationClassName	Mandatory	**Key**: See CIM schema definition.
SourceNamespaces[ ]	Mandatory	See 7.3.11.3.3.
IndividualSubscriptionSupported	Mandatory	See 7.3.11.3.4.
Query	Mandatory	See 7.3.11.3.5.
QueryLanguage	Mandatory	See 7.3.11.3.6.
Operations		
Associators( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.
GetAssociatedInstancesWithPath( )	Mandatory	See DSP0223.
GetAssociatedInstancePaths( )	Mandatory	See DSP0223.
GetReferencingInstancesWithPath( )	Mandatory	See DSP0223.
GetReferencingInstancePaths( )	Mandatory	See DSP0223.

7.3.11.3.2 Property: Name

The value of the Name property shall be the name of the indication filter; it shall be formatted as defined by the following ABNF rule:

```
    OrgID ":" RegisteredName ":" UniqueID
```

`OrgID` shall identify the business entity owning the referencing profile. `OrgID` shall include a copyrighted, trademarked, or otherwise unique name that is owned by that business entity or that is a registered ID assigned to that business entity by a recognized global authority. In addition, to ensure uniqueness, `OrgID` shall not contain a colon (`:`). For referencing profiles owned by DMTF, `OrgID` shall match `"DMTF"`.

`RegisteredName` shall be the registered name of the referencing profile, as defined by the value of the RegisteredName property in the RegisteredProfile instance representing the implemented version of that profile.

`UniqueID` shall uniquely identify the represented indication filter within the referencing profile.

DEPRECATED

For compatibility with version 1.0 of this profile, referencing profiles owned by business entities other than DMTF may in addition define values for the Name property that are formatted as defined by the following ABNF rule:

```
    OrgID ":" UniqueID
```

Where:

`OrgID` is defined above in this subclause.

`UniqueID` shall uniquely identify the instance within the business entity owning the referencing profile.

Version 1.1 of this profile has deprecated this additional format.

DEPRECATED

7.3.11.3.3 Property: SourceNamespaces

A non-Null value of this property is required for IndicationFilter instances in the Interop namespace; for IndicationFilter instances in other namespaces it is optional.

If not Null, the value of the SourceNamespaces[ ] array property shall contain the names of local namespaces that are considered as potential indication origin namespaces (see 6.1.2.4) during indication filtering; see 7.3.11.2. The value shall not be an empty array.

It is not required that the local namespaces identified by elements of value of the SourceNamespaces[ ] array property exist. If a non-existing local namespace is identified, no indications can originate out of that non-existing namespace; consequently, that element does not have an effect on indication filtering. However, if the identified namespace is added to the implementation at a later point in time, per the requirements of 7.3.11.2 indications originating out of that namespace are to be considered for indication filtering from then on.

The value elements of the SourceNamespaces[ ] array property shall be formatted using the format that the implementation uses for value of the Name property in instances of the CIM_Namespace class that represent namespaces.

7.3.11.3.4 Property: IndividualSubscriptionSupported

The value of the IndividualSubscriptionSupported property shall be True if the IndividualFilterSubscription feature (see 7.2.7) is available for the IndicationFilter instance; otherwise, the value shall be False.

7.3.11.3.5 Property: Query

The value of the Query property shall be a properly formed query statement that is conformant to the requirements of the query language identified by the value of the QueryLanguage property, and that states the coverage of the indication filter.

7.3.11.3.6 Property: QueryLanguage

The value of the QueryLanguage property shall identify the query language in which the query statement exposed by the value of the Query property is expressed.

NOTE This profile presently does not define a straight forward mechanism enabling clients to discover the set of query languages supported by an implementation. A future version of this profile is expected to introduce such a mechanism. For now, a rudimentary workaround may be inspecting the CIM representation of existing indication filters, thereby discovery a lower boundary for the set of supported query languages.

7.3.11.4 Instance requirements

Indication filters (see 6.2) shall be represented by IndicationFilter instances in the Interop namespace.

The representation in namespaces other than the Interop namespace should be avoided. However, if additional IndicationFilter instances represent an indication filter also in implementation namespaces, these instances shall have the same key property values as the one in the Interop namespace.

7.3.12 StaticIndicationFilter: CIM_IndicationFilter

7.3.12.1 General

The requirements in this subclause are referencing profile and WBEM server related implementation requirements.

The StaticIndicationFilter adaptation models static indication filters; static indication filters are described in 6.2.3.

The implementation type of the StaticIndicationFilter adaptation is: "abstract".

7.3.12.2 Element requirements

7.3.12.2.1 General

Table 15 lists the element requirements for the StaticIndicationFilter adaptation.

Table 15 – StaticIndicationFilter: Element requirements

Elements	Requirement	Description
Base adaptations		
IndicationFilter	Mandatory	See 7.3.11.
Properties		
QueryLanguage	Mandatory	See 7.3.12.2.2.
Operations		
CreateInstance( )	Prohibited	The implementation shall return the CIM status code CIM_ERR_NOT_IMPLEMENTED.
DeleteInstance( )	Prohibited	The implementation shall return the CIM status code CIM_ERR_NOT_IMPLEMENTED.
ModifyInstance( )	Prohibited	The implementation shall return the CIM status code CIM_ERR_NOT_IMPLEMENTED.

7.3.12.2.2 Property: QueryLanguage

In adaptations based on the StaticIndicationFilter adaptation in referencing profiles owned by DMTF, the value shall be "CQL", thereby requiring CQL as the query language.

7.3.13 DynamicIndicationFilter: CIM_IndicationFilter

7.3.13.1 General

The requirements in this subclause are WBEM server related implementation requirements.

The DynamicIndicationFilter adaptation models dynamic indication filters; dynamic indication filters are described in 6.2.6.

The requirement level of the DynamicIndicationFilter adaptation is conditional.

Condition: The DynamicIndicationFilters feature is implemented; see 7.2.1.

The implementation type of the DynamicIndicationFilter adaptation is: "instantiated".

7.3.13.2 Element requirements

7.3.13.2.1 General

Table 16 lists the element requirements for the DynamicIndicationFilter adaptation.

Table 16 – DynamicIndicationFilter: Element requirements

Elements	Requirement	Description
Base adaptations		
IndicationFilter	Mandatory	See 7.3.11.
Operations		
CreateInstance( )	Mandatory	See 7.3.13.2.2.
DeleteInstance( )	Mandatory	See 7.3.13.2.3.
ModifyInstance( )	Optional	See 7.3.13.2.4.

7.3.13.2.2 Operation: CreateInstance( )

Table 17 lists the error reporting requirements for the CreateInstance( ) operation on DynamicIndicationFilter instances. If any of the error situations described in the Description column of Table 17 matches, the operation shall fail and the corresponding CIM status code shall be returned. In addition, the error reporting requirements defined in DSP0223 for the CreateInstance( ) operation apply.

Table 17 – CreateInstance( ): Error reporting requirements

Reporting mechanism	Requirement level	Description
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the query language requested by the value of the Name property, as described in 7.3.11.3.2.
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the query language requested by the value of the SourceNamespaces[ ] array property, as described in 7.3.11.3.3. Note that the identified local namespaces do not have to exist.
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the query language requested by the value of the QueryLanguage property, as described in 7.3.11.3.6.

CIM_ERR_INVALID_PARAMETER	Mandatory	The value of the Query property in the embedded CIM_IndicationFilter instance is not a well formed query statement in the implemented subset of the query language expressed by the value of the QueryLanguage property.
CIM_ERR_INVALID_PARAMETER	Mandatory	The value of the Query property in the embedded CIM_IndicationFilter instance covers lifecycle indications, but does not contain a `WHERE` clause.
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the behavior requested by the value of the Query property, as described in 7.3.11.3.5.
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the behavior requested by the value of the IndividualSubscriptionSupported property, as described in 7.3.11.3.4.
CIM_ERR_FAILED	Mandatory	The implementation is unable to create the requested dynamic indication filter for other unspecified reasons.

If the CreateInstance( ) operation is successful, the requested dynamic indication filter shall be created, and — as a consequence — shall be represented by a DynamicIndicationFilter instance in the requested namespace.

Clients should abstain from requesting the creation of DynamicIndicationFilter instances in namespaces other than the Interop namespace. However, if the requested namespace is not the Interop namespace, the implementation shall expose an additional DynamicIndicationFilter instance representing the dynamic indication filter in the Interop namespace. That instance shall have identical values for all properties except for the SourceNamespaces[ ] array property for which the provisions of 7.3.11.3.3 apply.

If the CreateInstance( ) operation is fails, no dynamic indication filter shall be created, and — as a consequence — no representing DynamicIndicationFilter instances shall be exposed in any namespace.

DEPRECATED

If the returned CIM status code is CIM_ERR_FAILED because an indication filter with the same coverage as that requested already exists, the object path of the CIM_IndicationFilter instance representing the existing indication filter in the Interop namespace shall be returned as the value of the ErrorSource property in the CIM_Error instance accompanying the CIM status code.

NOTE Only this specific ad-hoc use of CIM_Error is deprecated. It is intended that a future version of this profile introduces extended error handling based on standard error messages.

DEPRECATED

With respect to input values for key properties the rules defined in DSP1001 apply, namely that implementation may ignore any input value for non-reference key properties, and that clients should abstain from providing input values for key properties.

7.3.13.2.3 Operation: DeleteInstance( )

Table 18 lists the error reporting requirements for the DeleteInstance( ) operation on DynamicIndicationFilter instances, and related CIM status codes. If any of the error situations described in the Description column of Table 18 matches, the operation shall fail and the corresponding CIM status code shall be returned. In addition, the error reporting requirements defined in DSP0223 for the DeleteInstance( ) operation apply.

Table 18 – DeleteInstance( ): Error reporting requirements

Reporting mechanism	Requirement level	Description
CIM_ERR_FAILED	Mandatory	The represented dynamic indication filter is referenced by subscription(s).

If the DeleteInstance( ) operation succeeds, the represented dynamic indication filter shall be deleted and — as a consequence — no longer be represented by any DynamicIndicationFilter instances in any namespace exposed by the implementation.

NOTE The instance requirements of associations representing relationships of the deleted dynamic indication filter imply that respective association instances in any namespace exposed by the implementation cease to exist; in this case this applies to IndicationServiceOfIndicationFilter instances (see 7.3.14). However, note that the DeleteInstance( ) operation for the dynamic indication filter is required to fail if subscriptions exist.

If the DeleteInstance( ) operation fails, the dynamic indication filter shall not be deleted, and — as a consequence — any representing DynamicIndicationFilter instances shall continue to exist as before.

7.3.13.2.4 Operation: ModifyInstance( )

The implementation of the ModifyInstance( ) operation enables clients to modify aspects of the behavior of the represented indication filter.

The requirement level of the ModifyInstance( ) operation is optional.

Table 19 lists the error reporting requirements for the ModifyInstance( ) operation on DynamicIndicationFilter instances. If any of the error situations described in the Description column of Table 19 matches, the operation shall fail and the corresponding CIM status code shall be returned. In addition, the error reporting requirements defined in DSP0223 for the ModifyInstance( ) operation apply.

Table 19 – ModifyInstance( ): Error reporting requirements

Reporting mechanism	Requirement level	Description
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the query language requested by the value of the Name property, as described in 7.3.11.3.2.
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the query language requested by the value of the SourceNamespaces[ ] array property, as described in 7.3.11.3.3. Note that the identified local namespaces do not have to exist.
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the query language requested by the value of the QueryLanguage property, as described in 7.3.11.3.6.
CIM_ERR_INVALID_PARAMETER	Mandatory	The value of the Query property in the embedded CIM_IndicationFilter instance is not a well formed query statement in the query language expressed by the value of the QueryLanguage property.
CIM_ERR_INVALID_PARAMETER	Mandatory	The value of the Query property in the embedded CIM_IndicationFilter instance covers lifecycle indications, but does not contain a WHERE clause.
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the behavior requested by the value of the Query property, as described in 7.3.11.3.5.

Reporting mechanism	Requirement level	Description
CIM_ERR_INVALID_PARAMETER	Mandatory	The implementation is unable to support the behavior requested by the value of the IndividualSubscriptionSupported property, as described in 7.3.11.3.4.
CIM_ERR_FAILED	Mandatory	The implementation is unable to apply the requested changes on the dynamic indication filter for other unspecified reasons.

If the ModifyInstance( ) operation is successful, the requested modification on the dynamic indication filter shall be applied, and — as a consequence — shall be reflected in all DynamicIndicationFilter instances that represent the modified dynamic indication filter and are exposed by the implementation.

If the ModifyInstance( ) operation is fails, the requested modification on the dynamic indication filter shall not be applied, and — as a consequence — all DynamicIndicationFilter instances that represent the dynamic indication filter shall remain unchanged.

7.3.13.3 Instance requirements

Dynamic indication filters shall be represented by DynamicIndicationFilter instances; the additional requirements of 7.3.11.4 apply.

7.3.14 IndicationServiceOfIndicationFilter: CIM_ServiceAffectsElement

The requirements in this subclause are referencing profile and WBEM server related implementation requirements.

The IndicationServiceOfIndicationFilter adaptation models the relationship between indication services and the indication filters they manage.

The implementation type of the IndicationServiceOfIndicationFilter association adaptation is: "instantiated".

Table 20 lists the element requirements for the IndicationServiceOfIndicationFilter association adaptation.

Table 20 – IndicationServiceOfIndicationFilter: Element requirements

Elements	Requirement	Description
Properties		
AffectingElement	Mandatory	**Key**: Value shall reference the IndicationService instance **Multiplicity**: 1
AffectedElement	Mandatory	**Key**: Value shall reference an IndicationFilter instance **Multiplicity**: *
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.

Each IndicationService instance (see 7.3.2) shall be associated through an IndicationServiceOfIndicationFilter instance with each IndicationFilter instance (see 7.3.11) representing an indication filter managed by the indication service represented by the IndicationService instance.

7.3.15 IndicationSpecificIndicationFilter: CIM_IndicationFilter

7.3.15.1 General

The requirements in this subclause are referencing profile and WBEM server related implementation requirements.

The IndicationSpecificIndicationFilter adaptation models indication-specific indication filters for indications defined in referencing profiles or in this profile; indication-specific indication filters are described in 6.2.4.

The requirement level of the IndicationSpecificIndicationFilter adaptation is optional.

The IndicationSpecificIndicationFilter adaptation should be implemented if indications defined in a referencing profile or in this profile are implemented.

The implementation type of the IndicationSpecificIndicationFilter adaptation is: "instantiated".

7.3.15.2 Element requirements

7.3.15.2.1 General

Table 21 lists the element requirements for the IndicationSpecificIndicationFilter adaptation.

Table 21 – IndicationSpecificIndicationFilter: Element requirements

Element	Requirement	Description
Base adaptations		
StaticIndicationFilter	Mandatory	See 7.3.12.
Properties		
Name	Mandatory	See 7.3.15.2.2.
Query	Mandatory	See 7.3.15.2.3.

7.3.15.2.2 Property: Name

The value of the Name property shall be formatted as defined by the following ABNF rule:

```
OrgID ":" RegisteredName ":" IndicationAdaptationName "Filter" [ "/"
MessageIdentification ]
```

`OrgID` and `RegisteredName` shall be specified as detailed in 7.3.11.3.2.

`IndicationAdaptationName` shall be the name of the indication adaptation defined in the profile identified by the `RegisteredName` rule. If the indication adaptation defines more than one possible indication.

The `MessageIdentification` suffix only applies for the representation of indication-specific indication filters covering alert indications modeled by an adaptation based on the AlertIndication adaptation (see 7.3.31); in this case for each alert indication defined by an alert message reference in the profile, a specific IndicationSpecificIndicationFilter instance is defined, where `MessageIdentification` shall be set as defined in 7.3.31.2 for the CIM representation of the alert indication. Thus, for alert indications, there is a one-to-one relationship between defined referenced alert messages and possible corresponding IndicationSpecificIndicationFilter instances.

For lifecycle indications the suffix is not necessary because adaptations based on the LifecycleIndication adaptation (see 7.3.32) only can address one event, as defined by a (constant) query statement. Thus, for lifecycle indications, there is a one-to-one relationship between defined lifecycle indications and possible corresponding IndicationSpecificIndicationFilter instances.

7.3.15.2.3 Property: Query

The value of the Query property shall be identical with the event definition query statement (see 7.3.29.2) of the indication adaptation defined in the referencing profile or in this profile that is covered by the represented indication-specific indication filter. In the case IndicationSpecificIndicationFilter instances covering alert indications modeled by an adaptation based on the AlertIndication adaptation, the value of the Query property shall apply the ABNF rule named `EventQuerySingle` (see 7.3.31.2); that way for alert indication adaptation referencing more than one alert message, separate IndicationSpecificIndicationFilter instances are defined for each referenced alert message.

7.3.15.3 Instance requirements

If a profile defines an indication adaptation based on the AlertIndication adaptation (see 7.3.31) or the Lifecycle adaptation (see 7.3.32), a corresponding indication-specific indication filter may be represented by an IndicationSpecificIndicationFilter instance, with respective values of the Name and Query properties.

NOTE As with any indication filter (see 6.2.1), the existence of an indication-specific indication filter and its representation by an IndicationSpecificIndicationFilter instance does not imply that the covered indication is actually implemented. Furthermore, in the case where multiple implementations of the referencing profile exist in a WBEM server, multiple IndicationSpecificIndicationFilter instances with identical values for Name and Query properties may result.

This profile leaves the decision whether or not to represent indication-specific indication filters as IndicationSpecificIndicationFilter instances to the implementation; however, referencing profiles can define an adaptation based on IndicationSpecificIndicationFilter adaptation that state more strict instance requirements.

In any case, if an implementation decides to represent indication-specific indication filters, these are to be represented as required by the IndicationSpecificIndicationFilter adaptation. In addition, the requirements of related adaptations such as the ProfileSpecificFilterCollection adaptation (see 7.3.21) or the IndicationFilterInFilterCollection associations adaptation (see 7.3.19) apply.

7.3.16 GlobalIndicationFilter: CIM_IndicationFilter

7.3.16.1 General

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The GlobalIndicationFilter adaptation models a global indication filters; global indication filters are described in 6.2.5.

The implementation type of the GlobalIndicationFilter adaptation is: "instantiated".

7.3.16.2 Element requirements

Table 22 lists the element requirements for the GlobalIndicationFilter adaptation.

Table 22 – GlobalIndicationFilter: Element requirements

Element	Requirement	Description
Base adaptations		
StaticIndicationFilter	Mandatory	See 7.3.12.

7.3.16.3 Instance requirements

7.3.16.3.1 Instance requirements related to alert indications

Table 23 lists the property value requirements for GlobalIndicationFilter instances covering all alert indications.

Table 23 – GlobalIndicationFilter: Instance requirements for instances covering all alert indications

Value of Name property	Value of Query property
"DMTF:Indications:GlobalAlertIndicationFilter"	"SELECT * FROM CIM_AlertIndication"

If the implementation supports the delivery of alert indications, it shall expose a GlobalIndicationFilter instance in the Interop namespace that complies with the value constraints defined in Table 23.

7.3.16.3.2 Instance requirements related to lifecycle indications

Table 24 lists the property value requirements for GlobalIndicationFilter instances covering all lifecycle indications of a particular subtype.

Table 24 – GlobalIndicationFilter: Instance requirements for instances covering all lifecycle indications

Value of Name property	Value of Query property
"DMTF:Indications:GlobalInstCreationIndicationFilter"	"SELECT * FROM CIM_InstCreation"
"DMTF:Indications:GlobalInstDeletionIndicationFilter"	"SELECT * FROM CIM_InstDeletion"
"DMTF:Indications:GlobalInstModificationIndicationFilter"	"SELECT * FROM CIM_InstModification"

If the implementation supports the delivery of lifecycle indications, it shall expose a GlobalIndicationFilter instance in the Interop namespace for each row listed in Table 24 that complies with the value constraints defined in that row.

7.3.17 StaticFilterCollection: CIM_FilterCollection

7.3.17.1 General

The requirements in this subclause are referencing profile and WBEM server related implementation requirements.

The StaticFilterCollection adaptation models static filter collections; static filter collections are described in 6.3.

The implementation type of the StaticFilterCollection adaptation is: "abstract".

7.3.17.2 Semantical requirements

The coverage of a filter collection shall be the aggregated coverage of all the indication gates contained by the filter collection. This definition applies recursively to contained filter collections.

NOTE Since filter collections aggregate the coverages of contained indication filters and contained other filter collections, and do not specify a filter query statement on their own, the defined coverage of a static filter collection is finally described by the set of query statements of its (directly or indirectly) aggregated indication filters.

The implementation shall filter all indications generated by (indication-specific parts of) the implementation that are within the coverage of a filter collection.

The implementation shall ignore any generated indication that is outside the coverage of the filter collection.

If a particular indication is within the coverage of more than one indication gate contained by a filter collection, that indication shall pass the filter collection only once, and shall not be replicated for every matching contained indication gate.

Indications that passed a filter collection need to be further processed; see the requirements on the IndicationFilterName property defined in 7.3.29.4.2, and the semantical requirements on listener destinations defined in 7.3.23.2, and on subscriptions defined in 7.3.25.2. If implemented, the requirements for reliable indications as defined in 7.3.30 and 7.4 may apply.

These semantics apply regardless of whether all, some or no contained indication gates are represented as collection members in CIM. Thus clients and listeners need to be aware of the fact that the coverage of a static filter collection may be larger than that observable through inspection of CIM represented members of that static filter collection. In other words, indications could be delivered to subscribed listeners that are within the coverage of members of the static filter collection that are not currently represented in CIM; in the extreme case no members at all are CIM represented. On the other hand, even if the coverage of a static filter collection is not represented through CIM, clients may have a priori knowledge about the defined coverage of that static filter collection, for example by means of built-in program code or data; see 7.3.17.3.

NOTE During runtime, the set of members of a static filter collection and the extent to which such members are represented in CIM may change. For example, consider the global filter collection with a defined coverage covering all alert indications defined in referencing profiles, as defined in 7.3.22.4.1. Its member set might grow or shrink over time as implementations of referencing profiles are installed in or removed from the implementation; however, the conceptual defined coverage of "all alert indications defined in referencing profile" remains constant.

7.3.17.3 Requirements pertaining to the defined coverage

For concrete adaptations based (directly or indirectly) on the StaticFilterCollection adaptation, profiles shall specify a defined coverage (see 6.3.3.3) through normative text that identifies indication filters and/or other filter collections as the *contained members* of the static filter collection, and thereby — because of 7.3.17.2 — as contributors to the coverage of the static filter collection.

NOTE If in a chain of (abstract and concrete) adaptations based on the StaticFilterCollection adaptation the defined coverage is defined as part of an intermediate (abstract or concrete) adaptation, that definition propagates into adaptations (directly or indirectly) based on that intermediate adaptation.

The defined coverage or a static filter collection always applies regardless of whether any members are represented in CIM. For contained static filter collections the specification of a defined coverage is likewise required.

The definition of the defined coverage may be specified at the level of adaptations, or may be broken down to individual adaptation instances, or both.

For examples of how to specify a defined coverage, see 7.3.21.3 and 7.3.22.

7.3.17.4 Element requirements

7.3.17.4.1 General

Table 25 lists the element requirements for the StaticFilterCollection adaptation.

Table 25 – StaticFilterCollection: Element requirements

Element	Requirement	Description
Properties		
InstanceID	Mandatory	**Key**: See CIM schema definition.
CollectionName	Mandatory	See 7.3.17.4.2.
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.
GetAssociatedInstancesWithPath( )	Mandatory	See DSP0223.
GetAssociatedInstancePaths( )	Mandatory	See DSP0223.
GetReferencingInstancesWithPath( )	Mandatory	See DSP0223.
GetReferencingInstancePaths( )	Mandatory	See DSP0223.

7.3.17.4.2 Property: CollectionName

The value of the CollectionName property shall be formatted as defined by the following ABNF rule:

```
OrgID ":" RegisteredName ":" UniqueID
```

`OrgID` shall identify the business entity owning the referencing profile. `OrgID` shall include a copyrighted, trademarked, or otherwise unique name that is owned by that business entity or that is a registered ID assigned to that business entity by a recognized global authority. In addition, to ensure uniqueness, `OrgID` shall not contain a colon (`:`).

For referencing profiles owned by DMTF, `OrgID` shall match `"DMTF"`.

`RegisteredName` shall be the registered name of the referencing profile, as defined by the value of the RegisteredName property in the RegisteredProfile instance representing the implemented version of the referencing profile.

`UniqueID` shall uniquely identify the instance within the implementation of the referencing profile.

DEPRECATED

For compatibility with version 1.0 of this profile, referencing profiles owned by business entities other than DMTF may in addition define values for the Name property that are formatted as defined by the following ABNF rule:

```
OrgID ":" UniqueID
```

Where:

`OrgID` is defined above in this subclause.

`UniqueID` shall uniquely identify the instance within the business entity owning the referencing profile.

Version 1.1 of this profile has deprecated this additional format.

DEPRECATED

7.3.17.5 Instance requirements

Static filter collections (see 6.3.3) shall be represented by StaticFilterCollection instances in the Interop namespace.

The representation in namespaces other than the Interop namespace should be avoided. However, if additional StaticFilterCollection instances represent a static filter collection in implementation namespaces, these StaticFilterCollection instances shall have the same key property values as the one in the Interop namespace.

If the FilterCollectionCoverageExposure feature (see 7.2.8) is available for a particular StaticFilterCollection instance, the contained members of the represented static filter collection (see 7.3.17.3), and their containment relationship to the static filter collection are required to be represented in CIM; see 7.3.12 for the representation of contained static indication filters, see 7.3.17 for the representation of contained static filter collections, and see 7.3.19 and 7.3.20 for the representation of the containment relationship.

7.3.18 IndicationServiceOfFilterCollection: CIM_OwningCollectionElement

The requirements in this subclause are referencing profile and WBEM server related implementation requirements.

The IndicationServiceOfFilterCollection adaptation models the relationship between a filter collection and the indication service that owns the filter collection.

The implementation type of the IndicationServiceOfFilterCollection association adaptation is: "instantiated".

Table 26 lists the element requirements for the IndicationServiceOfFilterCollection adaptation.

Table 26 – IndicationServiceOfFilterCollection: Element requirements

Elements	Requirement	Description
Properties		
OwningElement	Mandatory	**Key**: Value shall reference the IndicationService instance **Multiplicity**: 1
OwnedElement	Mandatory	**Key**: Value shall reference the StaticFilterCollection instance **Multiplicity**: *
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.

Each IndicationService instance (see 7.3.2.4) shall be associated through an IndicationServiceOfFilterCollection instance to every StaticFilterCollection instance (see 7.3.17) representing a static filter collection managed by the indication service represented by the IndicationService instance.

7.3.19 IndicationFilterInFilterCollection: CIM_MemberOfCollection

The IndicationFilterInFilterCollection adaptation models the relationship between a filter collection and its contained indication filters.

The requirement level of the IndicationFilterInFilterCollection adaptation is conditional.

Condition: The FilterCollectionCoverageExposure feature (see 7.2.8) is implemented.

The implementation type of the IndicationFilterInFilterCollection association adaptation is: "instantiated".

Table 27 lists the element requirements for the IndicationFilterInFilterCollection adaptation.

Table 27 – IndicationFilterInFilterCollection: Element requirements

Elements	Requirement	Description
Properties		
Collection	Mandatory	**Key**: Value shall reference a StaticFilterCollection instance representing a filter collection containing indication filters **Multiplicity**: *
Member	Mandatory	**Key**: Value shall reference an StaticIndicationFilter instance representing a contained static indication filter **Multiplicity**: *
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.

Each StaticFilterCollection (see 7.3.17) instance shall be associated through an IndicationFilterInFilterCollection instance with each of the IndicationFilter (see 7.3.11) instances representing contained indication filters.

7.3.20 FilterCollectionInFilterCollection: CIM_MemberOfCollection

The requirements in this subclause are referencing profile and WBEM server related implementation requirements.

The FilterCollectionInFilterCollection adaptation models the relationship between a filter collection and its contained other filter collections.

The requirement level of the FilterCollectionInFilterCollection adaptation is conditional.

Condition: All of the following:

- The static filter collections in the managed environment are capable of containing other static filter collections
- The FilterCollectionCoverageExposure feature (see 7.2.8) is implemented.

The implementation type of the FilterCollectionInFilterCollection association adaptation is: "instantiated".

Table 28 lists the element requirements for the FilterCollectionInFilterCollection adaptation.

Table 28 – FilterCollectionInFilterCollection: Element requirements

Elements	Requirement	Description
Properties		
Collection	Mandatory	**Key**: Value shall reference a StaticFilterCollection instance representing a filter collection containing other filter collections **Multiplicity**: *
Member	Mandatory	**Key**: Value shall reference a StaticFilterCollection instance representing a contained filter collection **Multiplicity**: *
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.

Each StaticFilterCollection instance (see 7.3.17) representing a static filter collection that contains other static filter collections shall be associated through a FilterCollectionInFilterCollection instance with each of the StaticFilterCollection instances (see 7.3.17) representing a contained static filter collection.

7.3.21 ProfileSpecificFilterCollection: CIM_FilterCollection

7.3.21.1 General

The requirements in this subclause are referencing profile and WBEM server related implementation requirements.

The ProfileSpecificFilterCollection adaptation models profile-specific filter collections; profile-specific filter collections are described in 6.3.3.4.

The requirement level of the ProfileSpecificFilterCollection adaptation is optional.

The ProfileSpecificFilterCollection adaptation should be implemented.

The implementation type of the ProfileSpecificFilterCollection adaptation is: "instantiated".

7.3.21.2 Element requirements

7.3.21.2.1 General

Table 29 lists the element requirements for the ProfileSpecificFilterCollection adaptation.

Table 29 – ProfileSpecificFilterCollection: Element requirements

Element	Requirement	Description
Base adaptations		
StaticFilterCollection	Mandatory	See 7.3.17.
Properties		

CollectionName	Mandatory	See 7.3.21.2.2.

7.3.21.2.2 Property: CollectionName

The value of the CollectionName property shall be formatted as defined by the following ABNF rule:

```
    "DMTF:" RegisteredName ":"
    "ProfileSpecified" Type "IndicationFilterCollection"
```

`OrgID` and `RegisteredName` shall be specified as detailed in 7.3.17.4.2.

`Type` shall be "Alert" in case the represented profile-specific filter collection covers all alert indications, and shall be "Lifecycle" in case the represented profile-specific filter collection covers all lifecycle indications defined in the referencing profile identified by `RegisteredName`.

NOTE This requirement does not preclude more than one instance in the Interop namespace from having identical values for the CollectionName property, because, for example, the referencing profile could be implemented more than once.

7.3.21.3 Requirements pertaining to the defined coverage

Requirements pertaining to the defined coverage are specified on a per instance basis; see 7.3.21.4 and 7.3.21.4.2.

7.3.21.4 Instance requirements

7.3.21.4.1 Instance requirements for profile-specific filter collections covering all alert indications specified in a profile

If and only if a referencing profile defines alert indications, the implementation may expose a ProfileSpecificFilterCollection instance in the Interop namespace that covers all alert indications defined in that profile. The element requirements defined in 7.3.21.2 apply.

NOTE The existence of that ProfileSpecificFilterCollection instance does not imply that any alert indications are actually implemented. Furthermore, in the case where multiple implementations of the referencing profile exist in a WBEM server, multiple ProfileSpecificFilterCollection instances may result.

The members of a profile-specific filter collection covering all alert indications defined in a referencing profile shall be all indication-specific indication filters covering the alert indications defined in that referencing profile; see 7.3.15. This definition in effect defines the defined coverage as all alert indications defined in the referencing profile.

NOTE For existing ProfileSpecificFilterCollection instances the instance requirements of association instances representing relationships of the represented profile-specific filter collection apply; for example, see 7.3.18, 7.3.19 or 7.3.20.

7.3.21.4.2 Instance requirements for profile-specific filter collections covering all lifecycle indications specified in a profile

If and only if a referencing profile defines lifecycle indications, the implementation may expose a ProfileSpecificFilterCollection instance in the Interop namespace that covers all lifecycle indications defined in that profile. The element requirements defined in 7.3.21.2 apply.

NOTE The existence of such a ProfileSpecificFilterCollection instance does not imply that any lifecycle indications are actually implemented. Furthermore, in the case where multiple implementations of the referencing profile exist in a WBEM server, multiple ProfileSpecificFilterCollection instances may result.

The members of a profile-specific filter collection covering all lifecycle indications defined in a referencing profile shall be all indication-specific indication filters covering the lifecycle indications defined in that referencing profile or in this profile; see 7.3.15. This definition in effect defines the defined coverage as all lifecycle indications defined in the referencing profile.

NOTE For existing ProfileSpecificFilterCollection instances the instance requirements of association instances representing relationships of the represented profile-specific filter collection apply; for example, see 7.3.18, 7.3.19 or 7.3.20.

The requirements specified in this subclause for lifecycle indications defined in referencing profiles shall also apply for the lifecycle indications defined in this profile; see 7.3.33 and 7.3.34.

7.3.22 GlobalFilterCollection: CIM_FilterCollection

7.3.22.1 General

The requirements in this subclause are referencing profile and WBEM server related implementation requirements; see 7.1.

The GlobalFilterCollection adaptation models global filter collection; global filter collections are described in 6.3.3.5.

The implementation type of the GlobalFilterCollection adaptation is: "instantiated".

7.3.22.2 Element requirements

Table 30 lists the element requirements for the ProfileSpecificFilterCollection adaptation.

Table 30 – GlobalFilterCollection: Element requirements

Element	Requirement	Description
Base adaptations		
StaticFilterCollection	Mandatory	See 7.3.17.

7.3.22.3 Requirements pertaining to the defined coverage

Requirements pertaining to the defined coverage are specified on a per instance basis; see 7.3.22.4.1, 7.3.22.4.2, 7.3.22.4.3 and 7.3.22.4.4.

7.3.22.4 Instance requirements

7.3.22.4.1 Instance requirements for the global filter collection covering all alert indications specified in profiles

If any alert indications specified in referencing profiles or in this profile are implemented, the implementation may expose a GlobalFilterCollection instance in the Interop namespace that covers all alert indications defined in profiles. In implementations where it is not possible to determine whether alert indications specified in referencing profiles are implemented, the instance may be exposed if the delivery of alert indications is implemented in general.

In the GlobalFilterCollection instance the value of the CollectionName property shall be as defined by the following ABNF rule:

```
"DMTF:Indications:"
"GlobalProfileSpecifiedAlertIndicationFilterCollection".
```

In this case the members of the represented global filter collection shall be all profile-specific filter collections covering the alert indications defined in any implemented referencing profile or in this profile; see 7.3.21.4. This definition in effect specifies the defined coverage as all alert indications defined in referencing profiles and in this profile; if instantiated by an implementation, the coverage would be all implemented alert indications out of that set.

NOTE For existing GlobalFilterCollection instances the instance requirements of association instances representing relationships of the represented global filter collection apply; for example, see 7.3.18, 7.3.19 or 7.3.20.

7.3.22.4.2 Instance requirements for the global filter collection covering all lifecycle indications specified in profiles

If any lifecycle indications specified in referencing profiles or in this profile are implemented, the implementation may expose a GlobalFilterCollection instance in the Interop namespace that covers all lifecycle indications defined in profiles. In implementations where it is not possible to determine whether lifecycle indications specified in referencing profiles are implemented, the instance may be exposed if the delivery of lifecycle indications is implemented in general.

In GlobalFilterCollection instance the value of the CollectionName property shall be as defined by the following ABNF rule:

```
"DMTF:Indications:"
"GlobalProfileSpecifiedLifecycleIndicationFilterCollection".
```

The members of the represented global filter collection shall be all profile-specific filter collections covering the lifecycle indications defined in any implemented referencing profile or in this profile; see 7.3.21.4.2. This definition in effect specifies the defined coverage as all lifecycle indications defined in referencing profiles and in this profile; if instantiated by an implementation, the coverage would be all implemented lifecycle indications out of that set.

NOTE For existing GlobalFilterCollection instances the instance requirements of association instances representing relationships of the represented global filter collection apply; for example, see 7.3.18, 7.3.19 or 7.3.20.

7.3.22.4.3 Instance requirements for the global filter collection covering all indications specified in profiles

If any indications specified in referencing profiles or in this profile are implemented, the implementation may expose a GlobalFilterCollection instance in the Interop namespace that covers all indications defined in profiles. In implementations where it is not possible to determine whether indications specified in referencing profiles are implemented, the instance may be exposed if the delivery of indications is implemented in general.

In the GlobalFilterCollection instance, the value of the CollectionName property shall be as defined by the following ABNF rule:

```
"DMTF:Indications:"
"GlobalProfileSpecifiedIndicationFilterCollection"
```

The members of the represented global filter collection shall be the following global filter collections (if existing):

- the global filter collection covering all alert indications defined in any implemented referencing profile, as required in 7.3.22.4.1
- the global filter collection covering all lifecycle indications defined in any implemented referencing profile, as required in 7.3.22.4.2

This definition in effect specifies the defined coverage as all indications defined in referencing profiles and in this profile; if instantiated by an implementation, the coverage would be all implemented indications out of that set.

NOTE For existing GlobalFilterCollection instances the instance requirements of association instances representing relationships of the represented global filter collection apply; for example, see 7.3.18, 7.3.19 or 7.3.20.

7.3.22.4.4 Instance requirements for the global filter collection covering all lifecycle indications

If the implementation supports the delivery of lifecycle indications, the implementation shall expose a GlobalFilterCollection instance in the Interop namespace that covers all lifecycle indications defined in profiles.

In GlobalFilterCollection instance the value of the CollectionName property shall be as defined by the following ABNF rule:

```
    "DMTF:Indications:GlobalLifecycleIndicationFilterCollection".
```

The members of the represented global filter collection shall be all profile-specific filter collections covering the global indication filters that each cover all indications of one of the three subtypes of lifecycle indications (CIM_InstCreation, CIM_InstDeletion and CIM_InstModification); see 7.3.16.3.2.

This definition in effect specifies the defined coverage as all lifecycle indications defined in referencing profiles and in this profile.

NOTE For existing GlobalFilterCollection instances the instance requirements of association instances representing relationships of the represented global filter collection apply; for example, see 7.3.18, 7.3.19 or 7.3.20.

7.3.23 ListenerDestination: CIM_ListenerDestination

7.3.23.1 General

The ListenerDestination adaptation models listener destinations; listener destinations are described in 6.4.5.

The implementation type of the ListenerDestination adaptation is: "instantiated".

7.3.23.2 Semantical requirements

For a particular listener destination, an implementation shall deliver any indication that passed the indication gate (see 6.2 or 6.3) referenced by any subscription (see 6.4.1) that also references the listener destination, to the listener referenced by that listener destination. See also the semantical requirements on indication filters defined in 7.3.11.2, on filter collections defined in 7.3.17.2, and on subscriptions defined in 7.3.25.2.

NOTE It is possible that a particular indication is delivered more than once to a particular listener for various reasons, such as that the listener is referenced by more than one listener destination, or that the indication is within the coverage of more than one indication gate, each of which is referenced by a subscription referencing the listener destination referencing the listener.

7.3.23.3 Element requirements

7.3.23.3.1 General

Table 31 lists the element requirements of the ListenerDestination adaptation.

Table 31 – ListenerDestination Element requirements

Element	Requirement	Description
Properties		
Name	Mandatory	**Key**: See CIM schema definition.
CreationClassName	Mandatory	**Key**: See CIM schema definition.
SystemName	Mandatory	**Key**: See CIM schema definition.
SystemCreationClassName	Mandatory	**Key**: See CIM schema definition.
ElementName	Mandatory	See CIM schema description.
Destination	Mandatory	See 7.3.23.3.2.
PersistenceType	Mandatory	See 7.3.23.3.3.
Protocol	Mandatory	See CIM schema description.
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.
GetAssociatedInstancesWithPath( )	Mandatory	See DSP0223.
GetAssociatedInstancePaths( )	Mandatory	See DSP0223.
GetReferencingInstancesWithPath( )	Mandatory	See DSP0223.
GetReferencingInstancePaths( )	Mandatory	See DSP0223.
CreateInstance( )	Optional	See 7.3.23.3.4 and DSP0223.
DeleteInstance( )	Optional	See 7.3.23.3.5 and DSP0223.
ModifyInstance( )	Optional	See 7.3.23.3.6 and DSP0223.

7.3.23.3.2 Property: Destination

The value of the Destination property shall identify the listener referenced by the listener destination.

A value of Null for the Destination property indicates a free listener destination (see 6.4.5).

If the value of the Destination property is not Null, it shall be a valid IETF Uniform Resource Identifier value (as defined in RFC3986) including the scheme, host and port as part of the URI Location.

7.3.23.3.3 Property: PersistenceType

The value of the PersistenceType property shall describe the durability of the represented listener destination.

The property values shall be constrained to 3 (Transient), 2 (Permanent), and Null.

If the listener destination is permanent, then the value of the PersistenceType property shall be either Null or 2 (Permanent). Permanent listener destinations are long-lived and are expected to be available for indication delivery. For example, a typical listener referenced by a permanent listener destination would be a system log file. The inability of an implementation to deliver indications to a listener referenced by a permanent listener destination will be treated as an error condition by the implementation, as defined in 7.4.3.5.

If the listener destination is transient, then the value of the PersistenceType property shall be 3 (Transient). Transient listener destinations are short-lived and have less strong requirements (than permanent listener destinations) regarding their availability for indication delivery. For example, a typical listener referenced by a transient listener destination would be a task progress meter in a graphical management application. The inability of an implementation to deliver indications to a listener described by a transient listener destination will be handled by removing the listener destination and its subscriptions from the implementation, as defined in 7.4.3.6.

7.3.23.3.4 Operation: CreateInstance( )

Table 32 lists the error reporting requirements for the CreateInstance( ) operation on ListenerDestination instances. If any of the error situations described in the Description column of Table 32 matches, the operation shall fail and the corresponding CIM status code shall be returned. In addition, the error reporting requirements defined in DSP0223 for the CreateInstance( ) operation apply.

Table 32 – CreateInstance( ): Error reporting requirements

Reporting mechanism	Requirement level	Description
CIM_ERR_INVALID_PARAMETER	Mandatory	The behavior requested by the value of the PersistenceType/OtherPersistenceType properties in the embedded CIM_ListenerDestination instance request a persistence type that is not implemented by the implementation.
CIM_ERR_INVALID_PARAMETER	Mandatory	The value of the Destination property in the embedded CIM_ListenerDestination instance does not constitute a valid URI as required in 7.3.23.3.2.
CIM_ERR_INVALID_PARAMETER	Mandatory	The behavior requested by the value of the Protocol/OtherProtocol properties in the embedded CIM_ListenerDestination instance request a protocol that is not implemented by the implementation.
CIM_ERR_FAILED	Mandatory	The number of listener destinations managed by the implementation would exceed the maximum number of listener destinations supported by the implementation; also see the description of the MaxListenerDestination property in 7.3.7.

If the CreateInstance( ) operation is successful, the requested listener destination shall be created, and — as a consequence — shall be represented by a ListenerDestination instance in the requested namespace. In addition, if the requested namespace is not the Interop namespace, the implementation shall expose an additional ListenerDestination instance representing the listener destination in the Interop namespace (see 7.3.23.4).

If the CreateInstance( ) operation fails, no listener destination shall be created, and — as a consequence — no representing ListenerDestination instances shall be exposed in any namespace.

The implementation may ignore the values of key properties in the embedded CIM_ListenerDestination instance passed as the value of the NewInstance parameter.

Clients should abstain from providing the values of key properties in the embedded CIM_ListenerDestination instance passed as the value of the NewInstance parameter.

Clients should abstain from requesting the creation of ListenerDestination instances in namespaces other than the Interop namespace.

Clients should favor the re-use of an existing listener destination referencing a particular listener over the creation of a new listener destination referencing the same listener.

7.3.23.3.5 Operation: DeleteInstance( )

Table 33 lists the error reporting requirements for the DeleteInstance( ) operation on ListenerDestination instances, and related CIM status codes. If any of the error situations described in the Description column of Table 33 matches, the operation shall fail and the corresponding CIM status code shall be returned. In addition, the error reporting requirements defined in DSP0223 for the DeleteInstance( ) operation apply.

Table 33 – ListenerDestination.DeleteInstance( ): Error reporting requirements

Reporting mechanism	Requirement level	Description
CIM_ERR_FAILED	Mandatory	The represented listener destination is referenced by subscription(s).

If the DeleteInstance( ) operation is successful, the represented listener destination shall be deleted and — as a consequence — shall no longer be represented by ListenerDestination instances in any namespace exposed by the implementation.

NOTE The instance requirements of associations representing relationships of the deleted listener destination imply that respective association instances in any namespace exposed by the implementation cease to exist; in this case this applies to IndicationServiceOfListenerDestination instances (see 7.3.24). However, note that the DeleteInstance( ) operation for the listener destination is required to fail if subscriptions exist.

If the DeleteInstace( ) operations fails, the listener destination shall not be deleted, and — as a consequence — any representing ListenerDestination instances shall continue to exist as before.

7.3.23.3.6 Operation: ModifyInstance( )

The ModifyInstance operation may be available for an instance of CIM_ListenerDestination.

The implementation of the ModifyInstance( ) operation enables clients to modify existing listener destinations.

The requirement level of the ModifyInstance( ) operation is optional.

Table 34 lists the error reporting requirements for the ModifyInstance( ) operation on ListenerDestination instances. If any of the error situations described in the Description column of Table 34 matches, the operation shall fail and the corresponding CIM status code shall be returned. In addition, the error reporting requirements defined in DSP0223 for the ModifyInstance( ) operation apply.

Table 34 – ModifyInstance( ): Error reporting requirements

Reporting mechanism	Requirement level	Description
CIM_ERR_INVALID_PARAMETER	Mandatory	The behavior requested by the value of the PersistenceType/OtherPersistenceType properties in the embedded CIM_ListenerDestination instance request a persistence type that is not implemented by the implementation.
CIM_ERR_INVALID_PARAMETER	Mandatory	The value of the Destination property in the embedded CIM_ListenerDestination instance does not constitute a valid URI as required in 7.3.23.3.2.
CIM_ERR_INVALID_PARAMETER	Mandatory	The behavior requested by the value of the Protocol/OtherProtocol properties in the embedded CIM_ListenerDestination instance requests a protocol that is not implemented by the implementation.
CIM_ERR_FAILED	Mandatory	A modification of the Destination and/or the Protocol/OtherProtocol properties was requested, but the represented listener destination is still referenced by subscription(s).

If the ModifyInstance( ) operation is successful, the requested modification on the listener destination shall be applied, and — as a consequence — shall be reflected in all ListenerDestination instances that represent the modified listener destination and are exposed by the implementation.

If the ModifyInstance( ) operation fails, the requested modification on the listener destination shall not be applied, and — as a consequence — all ListenerDestination instances that represent the listener destination shall remain unchanged.

7.3.23.4 Instance requirements

Listener destinations (see 6.4.5) shall be represented by ListenerDestination instances in the Interop namespace.

The representation in namespaces other than the Interop namespace should be avoided. However, if additional ListenerDestination instances represent the listener destination in implementation namespaces, these ListenerDestination instances shall have the same key property values as the one in the Interop namespace.

7.3.24 IndicationServiceOfListenerDestination: CIM_ServiceAffectsElement

The IndicationServiceOfListenerDestination adaptation models the relationship between indication services and the listener destinations they manage. Indication services are described in 6.5.2; listener destinations are described in 6.4.5.

The implementation type of the IndicationServiceOfListenerDestination association adaptation is: "instantiated".

Table 35 lists the elements requirements of the IndicationServiceOfListenerDestination adaptation.

Table 35 – IndicationServiceOfListenerDestination: Element requirements

Elements	Requirement	Description
Properties		
AffectingElement	Mandatory	**Key**: Value shall reference the IndicationService instance **Multiplicity**: 1
AffectedElement	Mandatory	**Key**: Value shall reference a ListenerDestination instance **Multiplicity**: *
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.

Each IndicationService (see 7.3.2) instance shall be associated through an IndicationServiceOfListenerDestination instance with each ListenerDestination (see 7.3.23) instance representing a listener destination managed by the indication service represented by the IndicationService instance.

7.3.25 AbstractSubscription: CIM_AbstractIndicationSubscription

7.3.25.1 General

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The AbstractSubscription adaptation models subscriptions for the delivery of indications from an indication gate to a listener referenced by a listener destination; subscriptions are described in 6.4.

The implementation type of the AbstractSubscription association adaptation is: "abstract".

7.3.25.2 Semantical requirements

An implementation shall deliver any indication that passed the indication gate referenced by the subscription (that is, any indication generated by the implementation that is within the coverage of the indication gate) to the listener referenced by the listener destination referenced by the subscription.

A listener that is referenced by the listener destination referenced by a subscription needs to be prepared to receive any indication that is within the coverage of the indication gate referenced by that subscription. Of course, listeners may ignore received indications.

7.3.25.3 Element requirements

Table 36 lists the element requirements for the AbstractSubscription adaptation.

Table 36 – AbstractSubscription: Element requirements

Elements	Requirement	Description
Properties		
Filter	Mandatory	**Key**: Value shall reference the IndicationFilter instance or the StaticFilterCollection instance
Handler	Mandatory	**Key**: Value shall reference the ListenerDestination instance
OnFatalErrorPolicy	Mandatory	See 7.3.25.3.1.
OtherOnFatalErrorPolicy	Conditional	Condition: The OnFatalErrorPolicy property can have the value 1 (Other). Pattern (".+") Value shall be non-Null if the value of the OnFatalErrorPolicy property is 1 (Other).
FailureTriggerTimeInterval	Mandatory	Value shall be the minimum delay before the policy indicated by the value of the OnFatalErrorPolicy property is applied
SubscriptionState	Mandatory	See CIM schema definition.

OtherSubscriptionState	Conditional	Condition: The SubscriptionState property can have the value 1 (Other). Pattern (".+") Value shall be non-Null if the value of the SubscriptionState property is 1 (Other).
RepeatNotificationPolicy	Mandatory	See 7.3.25.3.2.
RepeatNotificationInterval	Conditional exclusive	See 7.3.25.3.3.
RepeatNotificationGap	Conditional exclusive	See 7.3.25.3.4.
RepeatNotificationCount	Conditional exclusive	See 7.3.25.3.5.
Operations		
DeleteInstance( )	Mandatory	See 7.3.25.3.6 and DSP0223.
ModifyInstance( )	Optional	See 7.3.25.3.7 and DSP0223.
NOTE The CreateInstance( ) operation is defined in adaptations based on the AbstractSubscription adaptation; see 7.3.26 and 7.3.27.		

7.3.25.3.1 Property: OnFatalErrorPolicy

The value of the OnFatalErrorPolicy property shall indicate the behavior that the implementation exposes with respect to represented subscriptions in case of failures that imply that some aspect of indication generation processing or indication delivery is no longer functioning and indications may be lost.

A value of 4 (Remove) shall indicate that the implementation performs implicit subscription removal as detailed in 7.4.3.6; this shall be the default behavior.

7.3.25.3.2 Property: RepeatNotificationPolicy

The value of the RepeatNotificationPolicy property shall indicate the policy that the implementation applies with respect to the avoidance of repeated indication delivery of repeated indications as described in 6.1.6.

Table 37 lists constraints for the value of the RepeatNotificationPolicy property.

Table 37 – RepeatNotificationPolicy: Value constraints

Subscription behavior for the avoidance of repeated indication delivery	Required value
No avoidance of repeated indication delivery	2 (None)
The implementation applies the policy of suppressing the repeated indication delivery for the represented subscription, as described in 6.1.6.	3 (Suppress)
The implementation applies the policy of delaying the repeated indication delivery for the represented subscription, as described in 6.1.6 .	4 (Delay)

7.3.25.3.3 Property: RepeatNotificationInterval

The requirement level of the RepeatNotificationInterval property is conditional exclusive.

Condition: Either the SuppressRepeatNotificationPolicy feature (see 7.2.5) or the DelayRepeatNotificationPolicy feature (see 7.2.6) is available.

If the implementation applies the SuppressRepeatNotificationPolicy feature (see 7.2.5) for the represented subscription, as indicated by the value 3 (Suppress) for the RepeatNotification property, the value of the RepeatNotificationInterval property shall be the length of the time interval in seconds that the implementation waits after initial delivery of a number of repeated indications as indicated by the value of the RepeatNotificationCount property before delivering the next repeated indication.

If the implementation applies the DelayRepeatNotificationPolicy feature (see 7.2.6) for the represented subscription, as indicated by the value 4 (Delay) for the RepeatNotification property, the value of the RepeatNotificationInterval property shall be the length of the monitoring time interval in seconds during which the implementation monitors the indication gate referenced by the subscription for a number of additional repeated indications. Furthermore, only if during that monitoring interval at least the number of repeated indications as indicated by the value of the RepeatNotificationCount accrue, delivers only the first indication as a substitute for all the repeated indications accrued during the monitoring time interval.

7.3.25.3.4 Property: RepeatNotificationGap

The requirement level of the RepeatNotificationGap property is conditional exclusive.

Condition: The DelayRepeatNotificationPolicy feature (see 7.2.6) is implemented.

The value of the RepeatNotificationGap property shall be the length of the delay time interval in seconds that the implementation waits after delivering the first of a number of repeated indications that accrued during the monitoring time interval, before starting another monitoring time interval, as described in 7.3.25.3.5 with respect to implementations of the DelayRepeatNotificationPolicy feature.

7.3.25.3.5 Property: RepeatNotificationCount

The requirement level of the RepeatNotificationCount property is conditional exclusive.

Condition: Either the SuppressRepeatNotificationPolicy feature (see 7.2.5) or the DelayRepeatNotificationPolicy feature (see 7.2.6) is implemented.

If the implementation applies the SuppressRepeatNotificationPolicy feature (see 7.2.5) for the represented subscription, as indicated by the value 3 (Suppress) for the RepeatNotification property, the value of the RepeatNotificationCount property shall be the number of repeated indications that the implementation delivers before suppressing the delivery of further repeated indications within the time interval exposed by the value of the RepeatNotificationInterval property.

If the implementation applies the DelayRepeatNotificationPolicy feature (see 7.2.6) for the represented subscription, as indicated by the value 4 (Delay) for the RepeatNotification property, the value of the RepeatNotificationCount property shall be the number of repeated indications that the implementation is required to monitor and delay during the monitoring time interval exposed by the value of the RepeatNotificationInterval property. Only if during that monitoring time interval the number of accrued repeated indications reaches that number, the implementation shall deliver the first of repeated indication as a substitute for the accrued repeated indications. In other words, the quotient of the values of the RepeatNotificationCount and the RepeatNotificationInterval properties expresses a rate of repeated indications that must have been reached or exceeded during the monitoring time interval before one indication is delivered at the end of the monitoring time interval.

7.3.25.3.6 Operation: DeleteInstance( )

The error situations and CIM status codes defined in DSP0223 for the DeleteInstance( ) operation apply.

If the DeleteInstance( ) operation succeeds, the represented subscription shall be deleted and — as a consequence — shall no longer be represented by any AbstractSubscription instances in any namespace exposed by the implementation.

If the DeleteInstance( ) operation fails, the subscription shall not be deleted, and — as a consequence — any representing AbstractSubscription instances shall continue to exist as before.

7.3.25.3.7 Operation: ModifyInstance( )

The requirement level of the ModifyInstance( ) operation is optional.

Table 38 lists the error reporting requirements for the ModifyInstance( ) operation on AbstractSubscription instances, and related CIM status codes. If any of the error situations described in the Description column of Table 38 matches, the operation shall fail and the corresponding CIM status code shall be returned. In addition, the error reporting requirements defined in DSP0223 for the ModifyInstance( ) operation are applicable.

Table 38 – ModifyInstance( ): Error reporting requirements

Reporting mechanism	Requirement level	Description
CIM_ERR_INVALID_PARAMETER	Mandatory	The value of the OnFatalErrorPolicy/OtherOnFatalErrorPolicy properties (see 7.3.25.3.1) in the embedded CIM_AbstractSubscription instance request a fatal error policy that is not supported by the implementation, or the implementation does not support client-initiated changes of the fatal error policy.
CIM_ERR_INVALID_PARAMETER	Mandatory	The value of the FailureTriggerTimeInterval property in the embedded CIM_AbstractSubscription instance requests a time interval that is not supported by the implementation, or the implementation does not support client-initiated changes of the failure trigger time interval.
CIM_ERR_INVALID_PARAMETER	Mandatory	The value of the RepeatNotificationPolicy/RepeatNotificationInterval-/RepeatNotificationGap/RepeatNotificationCount properties in the embedded CIM_AbstractSubscription instance request a change in the repeat notification behavior of the represented subscription state that is not supported by the implementation, or the implementation does not support client-initiated changes of the repeat notification behavior.
CIM_ERR_INVALID_PARAMETER	Mandatory	The embedded CIM_AbstractSubscription instance has non-Null values for properties for which the implementation does not support client-initiated modifications.

If the ModifyInstance( ) operation is successful, the requested modification on the represented subscription shall be applied, and — as a consequence — shall be reflected in all AbstractSubscription instances that represent the modified subscription.

If the ModifyInstance( ) operation fails, the requested modification on the subscription shall not be applied, and — as a consequence — all AbstractSubscription instances that represent the subscription shall remain unchanged.

7.3.25.4 Instance requirements

Subscriptions (see 6.4.1) shall be represented by AbstractSubscription instances in the Interop namespace that relate either IndicationFilter instances (see 7.3.11) or StaticFilterCollection instances (see 7.3.17) with ListenerDestination instances (see 7.3.23).

The representation in namespaces other than the Interop namespace should be avoided. However, if both the indication filter/filter collection and the related listener destination represented by the referenced instances in the Interop namespace are also represented by additional instances in other namespaces, respective AbstractSubscription instances shall represent the subscription in these other namespaces as well.

7.3.26 FilterSubscription: CIM_IndicationSubscription

7.3.26.1 General

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The FilterSubscription adaptation models subscriptions for the delivery of indications from an indication filter to a listener referenced by a listener destination; subscriptions are described in 6.4.

The requirement level of the FilterSubscription adaptation is conditional.

Condition: The IndividualFilterSubscription feature (see 7.2.7) is implemented.

The implementation type of the FilterSubscription association adaptation is: "instantiated".

7.3.26.2 Semantical requirements

The semantical requirements of 7.3.25.2 apply respectively for the FilterSubscription adaptation.

7.3.26.3 Element requirements

7.3.26.3.1 General

Table 39 lists the element requirements for the FilterSubscription adaptation.

Table 39 – FilterSubscription: Element requirements

Elements	Requirement	Description
Base adaptations		
AbstractSubscription	Mandatory	See 7.3.25.
Properties		
Filter	Mandatory	**Key**: Value shall reference the IndicationFilter instance **Multiplicity**: *
Handler	Mandatory	**Key**: Value shall reference the ListenerDestination instance **Multiplicity**: *
Operations		
CreateInstance( )	Mandatory	See 7.3.26.3.2 and DSP0223.

7.3.26.3.2 Operation: CreateInstance( )

Table 40 lists the error reporting requirements for the CreateInstance( ) operation on FilterSubscription instances. If any of the error situations described in the Description column of Table 40 matches, the operation shall fail and the corresponding CIM status code shall be returned. In addition, the error reporting requirements defined in DSP0223 for the CreateInstance( ) operation apply.

Table 40 – CreateInstance( ): Error reporting requirements

Reporting mechanism	Requirement level	Description

Reporting mechanism	Requirement level	Description
CIM_ERR_INVALID_PARAMETER	Mandatory	The value of the Filter property in the embedded CIM_IndicationSubscription instance references an instance that does not exist, or is not an IndicationFilter instance (see 7.3.11).
CIM_ERR_INVALID_PARAMETER	Mandatory	The value of the Handler property in the embedded CIM_IndicationSubscription instance references an instance that does not exist, or is not ListenerDestination instance (see 7.3.23).
CIM_ERR_FAILED	Mandatory	The IndividualFilterSubscription feature (see 7.2.7) is not available for the indication filter represented by the IndicationFilter instance referenced by the value of the IndicationFilter property in the embedded CIM_IndicationSubscription instance.
CIM_ERR_FAILED	Mandatory	The number of subscriptions managed by the implementation would exceed the maximum number of subscriptions supported by the implementation; also see the description of the MaxSubscriptions property in 7.3.7.
NOTE With version 1.2 of this profile the requirements for CIM status code values were refined, fixing the incorrect requirement for a value named CIM_ERROR_NOT_SUPPORTED mandated by previous versions.		

If the CreateInstance( ) operation is successful, the requested filter subscription was created, and consequently — as required by 7.3.26.4 — shall be represented by a FilterSubscription instance in the requested namespace. In addition, if the requested namespace is not the Interop namespace, the implementation shall expose an additional FilterSubscription instance representing the subscription in the Interop namespace (see 7.3.26.4).

If the CreateInstance( ) operation fails, no subscription shall be created, and — as a consequence — no representing FilterSubscription instances shall be exposed in any namespace.

Clients should abstain from requesting the creation of FilterSubscription instances in namespaces other than the Interop namespace.

7.3.26.4 Instance requirements

The requirements of 7.3.25.4 apply respectively for FilterSubscription instances.

7.3.27 CollectionSubscription: CIM_FilterCollectionSubscription

7.3.27.1 General

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The CollectionSubscription adaptation models subscriptions for the delivery of indications from a filter collection to a listener referenced by a listener destination; subscriptions are described in 6.4.

The implementation type of the FilterCollectionSubscription association adaptation is: "instantiated".

7.3.27.2 Semantical requirements

The semantical requirements of 7.3.25.2 apply respectively for the CollectionSubscription adaptation.

7.3.27.3 Element requirements

7.3.27.3.1 General

Table 41 lists the element requirements for the CollectionSubscription adaptation.

Table 41 – CollectionSubscription: Element requirements

Elements	Requirement	Description
Base adaptations		
AbstractSubscription	Mandatory	See 7.3.25.
Properties		
Filter	Mandatory	**Key**: Value shall reference the StaticFilterCollection instance **Multiplicity**: *
Handler	Mandatory	**Key**: Value shall reference the ListenerDestination instance **Multiplicity**: *
Operations		
CreateInstance( )	Mandatory	See 7.3.27.3.2 and DSP0223.

7.3.27.3.2 Operation: CreateInstance( )

Table 42 lists the error reporting requirements for the CreateInstance( ) operation on CollectionSubscription instances. If any of the error situations described in the Description column of Table 42 matches, the operation shall fail and the corresponding CIM status code shall be returned. In addition, the error reporting requirements defined in DSP0223 for the CreateInstance( ) operation apply.

Table 42 – CreateInstance( ): Error reporting requirements

Reporting mechanism	Requirement level	Description
CIM_ERR_INVALID_PARAMETER	Mandatory	The value of the Collection property in the embedded CIM_FilterCollectionSubscription instance references an instance that does not exist, or is not a StaticFilterCollection instance (see 7.3.17).
CIM_ERR_INVALID_PARAMETER	Mandatory	The value of the Handler property in the embedded CIM_FilterCollectionSubscription instance references an instance that does not exist, or is not a ListenerDestination instance (see 7.3.23).
CIM_ERR_FAILED	Mandatory	The number of subscriptions managed by the implementation would exceed the maximum number of subscriptions supported by the implementation; also see the description of the MaxSubscriptions property in 7.3.7.
NOTE With version 1.2 of this profile the requirements for CIM status code values were refined, fixing the incorrect requirement for a value named CIM_ERROR_NOT_SUPPORTED mandated by previous versions.		

If the CreateInstance( ) operations is successful, the requested filter subscription was created, and consequently — as required by 7.3.27.4 — shall be represented by a CollectionSubscription instance in the requested namespace. In addition, if the requested namespace is not the Interop namespace, the implementation shall expose an additional CollectionSubscription instance representing the subscription in the Interop namespace (see 7.3.27.4).

If the CreateInstance( ) operation fails, no subscription shall be created, and — as a consequence — no representing CollectionSubscription instances shall be exposed in any namespace.

Clients should abstain from requesting the creation of CollectionSubscription instances in namespaces other than the Interop namespace.

7.3.27.4 Instance requirements

The instance requirements of 7.3.25.4 apply respectively for CollectionSubscription instances.

DEPRECATED

7.3.28 ProfileOfFilterCollection: CIM_ConcreteDependency

The ProfileOfFilterCollection adaptation models the relationship between a filter collection defined in a referencing profile and the profile registration of that referencing profile.

The implementation type of the ProfileOfFilterCollection association adaptation is: "instantiated".

Each StaticFilterCollection instance (see 7.3.17) representing a filter collection defined in a referencing profile shall be associated through a ProfileOfFilterCollection instance with the ProfileRegistration instance (see DSP1033) representing the implemented version of the referencing profile.

NOTE This profile assumes that a future version of the Profile Registration profile (see DSP1033) will be based on version 1.1 of the Profile Usage Guide (see DSP1001), and define the ProfileRegistration adaptation; until then, substitute that by the definition of the CIM_RegisteredProfile "profile class" defined in version 1.0 of DSP1033.

Table 43 lists the element requirements for the ProfileOfFilterCollection adaptation.

Table 43 – ProfileOfFilterCollection: Element requirements

Elements	Requirement	Description
Properties		
Antecedent	Mandatory	**Key**: Value shall reference the ProfileRegistration instance **Multiplicity**: 1
Dependent	Mandatory	**Key**: Value shall reference the StaticFilterCollection instance **Multiplicity**: *
Operations		
GetInstance( )	Mandatory	See DSP0223.
GetClassInstancesWithPath( )	Mandatory	See DSP0223.
GetClassInstancePaths( )	Mandatory	See DSP0223.

DEPRECATED

7.3.29 BasicIndication: CIM_Indication

7.3.29.1 General

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The BasicIndication adaptation models indications; indications are described in 6.1.

The implementation type of the BasicIndication indication adaptation is: "abstract".

7.3.29.2 Event definition requirements

Referencing profiles that model indications through adaptations based on the BasicIndication adaptation shall define event that the indication is designed to report. This event definition shall be accomplished by means of an event definition query statement stated in CQL (see DSP0202).

The purpose of an event definition query statement is to formally define the event(s) that an indication adaptation is designed to report, such that by inspecting the event definition query statements an implementer knows how to implement the indication adaptation. A CIM representation of event definition query statements is not defined, thus there is no requirement for implementations or clients to be able to programmatically interpret event definition query statements.

NOTE Event definition query statements are different from indication filter query statements. An indication filter query statement (see 7.3.11.3.5) defines the coverage of an indication filter, and is exposed to clients by the value of the Query property in the IndicationFilter instance representing the indication filter. The IndicationSpecificIndicationFilter adaptation (see 7.3.15) models indication-specific indication filters (see 6.2.4) and addresses the needs of clients requiring notifications about events reported by particular indications specified in a profile.

The CQL query statement defining the event shall comply with the following ABNF rule:

```
"SELECT" WS PropertySet WS "FROM" WS IndicationClass WS "WHERE" WS
SelectionExpression
```

`PropertySet` shall be `"*"`, or a comma-separated list of property names.

`IndicationClass` shall be the adapted indication class, that is, CIM_Indication or a subclass thereof.

`SelectionExpression` shall be a constant string that defines a selection expression conformant with the rules for selection expressions defined by DSP0202.

`WS` represents one or more whitespace characters.

The requirements in this subclause may be refined by requirements defined in adaptations based on the BasicIndication adaptation, including the case that a refined query statement references an external element (such as an alert message definition in a message registry) that defines the event.

7.3.29.3 Indication origin

Each indication shall be assigned an origin namespace (see 6.1.2.4).

In general, an implementation is free to select any local namespace as the origin namespace for a generated indication; however, adaptations based on the BasicIndication adaptation such as the AlertIndication adaptation (see 7.3.31) and the LifecycleIndication (see 7.3.32) establish additional constraints.

The indication origin is not represented in the CIM representation of an indication as defined by the CIM_Indication class.

The implementation class of the indication is required to reside in the origin namespace.

NOTE As with any implementation class, the existence of an indication implementation class within a namespace is does not sufficiently indicate that the indication is really implemented. Additional requirements — such as the presence and integration of functional code implementing the indication — apply, but are outside of the scope of this profile.

The indication origin is required to be considered during indication filtering; see 6.1.4 and 7.3.11.2.

7.3.29.4 Element requirements

7.3.29.4.1 General

Table 44 lists the element requirements for the BasicIndication adaptation.

Table 44 – BasicIndication: Element requirements

Elements	Requirement	Description
Properties		
IndicationFilterName	Mandatory	See 7.3.29.4.2.
IndicationIdentifier	Mandatory	See CIM schema definition.
IndicationTime	Mandatory	See CIM schema definition.

7.3.29.4.2 Property: IndicationFilterName

The value of the IndicationFilterName property shall contain the name of the indication gate that the indication passed before being delivered to the listeners subscribed to that indication gate. For indication filters, the name is exposed by the value of the Name property in representing IndicationFilter instances (see 7.3.11). For filter collections, the name is exposed by the value of the CollectionName property in representing StaticFilterCollection instances (see 7.3.17).

Because an indication is generated independently and before it is subjected to filtering, the name of the filtering indication gate is not known at indication-generation time. Instead, a generated indication might match a large number of indication gates. During indication filtering (see 6.1.4 and 7.3.11.2), each time a generated indication matches an indication gate with existing subscriptions, and before delivering that indication to subscribed listeners, the implementation shall set the value of the IndicationFilterName property in the BasicIndication instance representing the indication to the identification of that indication gate, as follows:

- in case of indication filters, the identification shall be the value of the Name property of the IndicationFilter instance representing the indication filter
- in case of filter collections, the identification shall be the value of the CollectionName property of the StaticFilterCollection instance representing the filter collection.

NOTE 1 The requirement for referencing filter collections was added with version 1.2. of this profile.

NOTE 2 A listener may use the value of the IndicationFilterName property to determine which indication gate was passed by the indication before being delivered to the listener.

7.3.29.5 Indication generation requirements

Adaptations based on the BasicIndication adaptation are required to define the event that the modeled indication is designed to report; see 7.3.29.2.

If the event defined by such an adaptation occurs, and if subscriptions exist for any indication gate covering the modeled indication, an instance of the indication adaptation based on the BasicIndication shall be generated.

NOTE The way this requirement is stated it provides for the optimized approach of checking for the presence of matching indication gate with subscriptions already at indication generation time; however, even in this case indication filtering is required as a subsequent step (see 6.1.4) in order to ensure that all matching indication gates are considered, and indication delivery occurs to all listeners subscribed to any of the indication gates covering the indication.

7.3.30 ReliableIndication: CIM_Indication

7.3.30.1 General

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The ReliableIndication adaptation models reliable indications; the concept of reliable indications is introduced in 6.1.5. Additional requirements for reliable indication delivery are specified in 7.4.

The implementation type of the ReliableIndication indication adaptation is: "abstract".

NOTE The ReliableIndications adaptation is intentionally not based on the BasicIndication adaptation, such that it can be implemented independently as a separate option. Reliable indication delivery is typically implemented centrally once for the delivery of all indications implemented by an implementation.

7.3.30.2 Element requirements

7.3.30.2.1 General

Table 45 lists the element requirements for the ReliableIndication adaptation.

Table 45 – ReliableIndication: Element requirements

Elements	Requirement	Description
Properties		
SequenceContext	Mandatory	See 7.3.30.2.2.
SequenceNumber	Mandatory	See 7.3.30.2.3.

7.3.30.2.2 Property: SequenceContext

The value of the SequenceContext property shall contain the sequence context portion of the sequence identifier (see 3.30 and 7.4.2). See the CIM schema description for additional constraints and the required semantics, and see 7.4 for additional requirements on reliable indication delivery.

NOTE 1 The CIM schema definition of the CIM_Indication class requires for the SequenceContext property that the implementation maintains the context for this property separately for each registered listener destination, and that restarts of the WBEM server cause the value to change. This requirement enables a listener to detect WBEM server restarts, and to differentiate the indication streams from a particular WBEM server that were processed (within that WBEM server) through different listener destinations referring to the listener.

NOTE 2 Indications can be lost when a listener fails and restarts, with the WBEM server continuing to send indications while the listener is inactive. In that case, upon restart of the listener, if does not persist the last received sequence identifier, the listener would establish the sequence identifier of the first received indication after the restart as check value, failing to notice that while it was inactive additional indications were sent (and lost). One approach for discovering an actual loss of indications might be to persist the latest sequence identifier as part of a listener termination routine, and upon restart use the persisted value as a check value (instead of that taken from the first arriving indication after the restart).

7.3.30.2.3 Property: SequenceNumber

The value of the SequenceNumber property shall contain the sequence number portion of the sequence identifier (see 3.30 and 7.4.2). See the CIM schema description for additional constraints and the required semantics, and see 7.4 for additional requirements on reliable indication delivery.

NOTE The CIM schema definition of CIM_Indication class requires for the SequenceNumber property in the stream of instances processed through a particular listener destination, that the value starts at 0 whenever the value of the SequenceContext property changes.

7.3.31 AlertIndication: CIM_AlertIndication

7.3.31.1 General

The AlertIndication adaptation models alert indications; alert indications are described in 6.1.3.

The implementation type of the AlertIndication indication adaptation is: "abstract".

It is expected that the AlertIndication adaptation is used as a base adaptation for modeling alert indications in referencing profiles.

7.3.31.2 Event definition requirements

This subclause refines the event definition requirements established by the BasicIndication adaptation; see 7.3.29.2.

The query statement defined by the following ABNF rules define the event(s) that are reported by AlertIndication instances:

- If the AlertIndication adaptation identifies only one related alert message (see 7.3.31.3), the event query statement is defined as follows:

```
EventQuerySingle = "SELECT" WS PropertySet WS "FROM" WS
AlertIndicationClass WS "WHERE" WS "OwningEntity='" OwningEntity "'"
WS "AND" WS "MessageID=" MessageId WS AdditionalWhereElements
```

- If the AlertIndication adaptation identifies more than one related alert message (see 7.3.31.3), the event query statement is defined as follows:

```
EventQueryMulti = "SELECT" WS PropertySet WS "FROM" WS
AlertIndicationClass WS "WHERE" WS "OwningEntity='" OwningEntity "'"
WS "AND" WS "MessageID LIKE" WS "'" MessageSet "'" [ WS
AdditionalSelectionExpression ]

MessageSet = MessageIdentification [ "|" MessageSet ]
```

NOTE Recall that the purpose of the event definition query statement is to formally define the event(s) that an indication is designed to report; see 7.3.29.2. Event definition query statements are not represented in CIM; thus there is no requirement for implementations or clients to interpret event definition query statements.

`PropertySet` shall be `"*"`, or a comma-separated list of property names.

`AlertIndicationClass` shall be `CIM_AlertIndication`, or, if adaptations based on the AlertIndication adaptation adapt a class derived from CIM_AlertIndication, shall be replaced by the name of the adapted alert indication class.

`OwningEntity` shall be the name of the organization defining the alert indication. In profiles owned by DMTF, the value shall be "DMTF".

`MessageIdentification` shall identify each referenced alert message, as required by 7.3.31.3.

Referencing profiles in their adaptations based on the AlertIndication adaptation may refine the event definition; however, such refinements shall remain within the constraints established by the query statement specified in this subclause.

If a referencing profile defining an adaptation based on the AlertIndication adaptation does not require refining the query statement specified in this subclause, then a repetition of the query statement is not required as part of the adaptation in the referencing profile, and compliance with this subclause is achieved through designating a related alert message as required in 7.3.31.3.

AdditionalSelectionExpression shall be a constant string that defines a selection expression conformant with the rules for selection expressions defined by DSP0202. For example, the value of the PerceivedSeverity property could be constrained to specific values.

7.3.31.3 Related alert messages

Referencing profiles defining adaptations based on the AlertIndication adaptation as part of their alert indication adaptation shall reference one or more related CIM alert message(s) that are defined in a message registry conformant to DSP0228.

The formal requirements for referencing alert messages through message identifications as part of adaptation definitions are detailed in DSP1001; as defined there, the main elements of a message identification are the name of the registry reference referring to the registry defining the alert message, and the message id as the concatenation of the value of the PREFIX attribute and the SEQUENCE_NUMBER attribute from the MESSAGE_ID element that defines the message within the message registry.

CIM alert messages provide for a formalized and widely self-contained approach to define alert indications. CIM alert messages are defined in message registries. A message registry is an XML document that contains message definitions. DSP0228 defines an XML schema for message registries. The schema defines the XML elements that can be used for message definitions. Each element is formally defined using the XML schema language. Each of these element definitions is annotated with documentation that may define formal requirements for the use of the message element.

Each message definition in a message registry consists of a standard message identifier and a description of static and dynamic message elements and of other message components; for details, see DSP0228.

The MESSAGE_ID element within the message definition identifies the message within the scope of the message registry through a prefix and a sequence number.

The MESSAGE_DESCRIPTION element within an alert message definition contains a plain text description of the event that is reported by the defined alert message. A profile modeling an alert indication shall rely on the event definition provided in the alert message description. In case the alert-message-based definition of the event is insufficient in the context of the profile, the profile may augment the event definition within its definition of the alert indication; however, the amendments to the event definition stated in a profile shall remain within the constraints defined by the event definition in the alert message definition in the message repository.

The <MESSAGE_COMPONENTS> element within an alert message definition defines a sequence of static and dynamic elements that together compose the message. The static elements define constant text parts of the message. The dynamic elements reference property values in identified CIM instances, such that the property values become dynamic parts of the alert message.

7.3.31.4 Indication origin

If the alert indication is related to a managed object, and the CIM representation of that managed object is referenced by the value of the AlertingManagedElement property in the CIM representation of the alert indication, then the indication origin as required by 7.3.29.3 should be the namespace in which the CIM representation of that managed object exists.

7.3.31.5 Element requirements

7.3.31.5.1 General

Table 46 lists the element requirements for the AlertIndication adaptation.

Table 46 – AlertIndication: Element requirements

Elements	Requirement	Description
Base adaptations		
BasicIndication	Mandatory	See 7.3.29.
ReliableIndication	Conditional	Condition: The ReliableIndications feature (see 7.2.4) is implemented. See 7.3.30; note that this is a WBEM server related implementation requirement; see 7.1.
Properties		
AlertingElementFormat	Mandatory	Value shall match 2 (CIMObjectPath)
AlertingManagedElement	Mandatory	See 7.3.31.5.2.
AlertType	Mandatory	See 7.3.31.5.3.
Message	Optional	See 7.3.31.5.4.
MessageID	Mandatory	See 7.3.31.5.5.
OtherAlertType	Conditional	Condition: The AlertType property can have the value 1 (Other). Value shall be non-Null if the value of the AlertType property is 1 (Other).
OwningEntity	Mandatory	See 7.3.31.5.6.
PerceivedSeverity	Mandatory	See 7.3.31.5.7.
ProbableCause	Mandatory	See CIM schema definition.
ProbableCauseDescription	Conditional	Condition: The ProbableCause property can have the value 1 (Other). Value shall be non-Null if the value of the ProbableCause property is 1 (Other).
SystemName	Mandatory	See 7.3.31.5.8.
MessageArguments[ ]	Mandatory	See 7.3.31.5.9.

7.3.31.5.2 Property: AlertingManagedElement

If the managed element for which the alert indication is reported is represented by one or more CIM instances within the implementation, then the value of the AlertingManagedElement property shall identify the most prominent of these CIM instances, using the format of a WBEM-URI-UntypedInstancePath (as defined in DSP0207); otherwise the value of the AlertingManagedElement property shall be Null.

7.3.31.5.3 Property: AlertType

The requirements of DSP0228 apply. Note that DSP0228 requires the value of the AlertType property in CIM_AlertIndication instances conveying an alert message from a message registry to be set to the content of the ALERT_TYPE element from the alert message definition in the message registry.

7.3.31.5.4 Property: Message

The requirement level of the Message property is optional.

The Message property may contain the formatted alert message from the registry.

7.3.31.5.5 Property: MessageID

The requirements of DSP0228 apply. Note that DSP0228 requires the value of the MessageID property in CIM_AlertIndication instances conveying an alert message from a message registry to be set to the concatenation of the `PREFIX` and `SEQUENCE_NUMBER` attribute values from the alert message definition in the message registry (that is, no further padding or adjustment of these values takes place).

NOTE The `SEQUENCE_NUMBER` attribute value is not to be confused with the sequence number within a sequence identifier that enables unique identification of the indications originating from a particular WBEM server to a particular WBEM listener; see 7.4.2.

7.3.31.5.6 Property: OwningEntity

The requirements of DSP0228 apply. Note that DSP0228 requires the value of the OwningEntity property in CIM_AlertIndication instances conveying an alert message from a message registry to be set to the content of the `OWNING_ENTITY` element from the alert message definition in the message registry.

7.3.31.5.7 Property: PerceivedSeverity

The requirements of DSP0228 apply. Note that DSP0228 requires the value of the PerceivedSeverity property in CIM_AlertIndication instances conveying an alert message from a message registry to be set to the content of the `PERCEIVED_SEVERITY` element from the alert message definition in the message registry.

7.3.31.5.8 Property: SystemName

If the managed element for which the alert indication is reported is represented by a CIM instance within the implementation, and the managed element is a component of a system that is represented by a CIM_System instance, then the value of the SystemName property in the AlertIndication instance shall be identical with the value of the Name property in the CIM_System instance; otherwise, the value of the SystemName property shall be Null.

7.3.31.5.9 Property: MessageArguments[ ]

The requirements of DSP0228 apply. Note that DSP0228 requires the (string typed) MessageArguments array property in CIM_AlertIndication instances conveying an alert message from a message registry to contain one array entry for each dynamic element defined in the alert message, in the order specified by the alert message definition in the message registry, where the value of the array element provides the value of the dynamic element.

If for a particular alert indication defined by a referencing profile the definition of a dynamic element (including its description) within an alert message definition in a message registry is not sufficient to identify a particular CIM instance and property as required by the referencing profile, then the referencing profile shall specify augmenting provisions that explicitly identify an instance and a property that are compatible with the definition of the dynamic element within the alert message.

For example, assume that an alert message is defined in a message repository, as follows:

```
<MESSAGE NAME="System state change">
  <MESSAGE_ID PREFIX="SVPC" SEQUENCE_NUMBER="0123"/>
  <MESSAGE_DESCRIPTION>
    This message describes a system state change.
  </MESSAGE_DESCRIPTION>
  <MESSAGE_COMPONENTS>
    <STATIC_ELEMENT>The system </STATIC_ELEMENT>
    <DYNAMIC_ELEMENT NAME="SystemElementName"
      SOURCE_PROPERTY="CIM_System.ElementName" DATATYPE="string"/>
    <STATIC_ELEMENT> changed its state to </STATIC_ELEMENT>
```

```
        <DYNAMIC_ELEMENT NAME="SystemState"
          SOURCE_PROPERTY="CIM_System.EnabledState" DATATYPE="string"/>
        <STATIC_ELEMENT> .</STATIC_ELEMENT>
      </MESSAGE_COMPONENTS>
      <FIXED_MESSAGE_INSTANCE_VALUES TYPE="ALERT">
        <!-- . . . -->
      </FIXED_MESSAGE_INSTANCE_VALUES>
      <!-- . . . -->
    </MESSAGE>
```

An Example System Virtualization profile might model an indication reporting state changes of both host systems and virtual systems. In both cases the SVPC0123 alert message would be used, but the identification of affected instances would need to be specialized separately for each case.

Assuming that the profile defines a HostSystem adaptation of the CIM_System class for the representation of host systems, and defines a HostStateChange indication adaptation in order to report state changes of host systems, the requirements for the MessageArguments[] array property as part of the HostStateChange indication adaptation would need to augment the alert message definition from the message registry, as follows:

- The value of MessageArguments[0] shall be the value of the ElementName property of the HostSystem instance representing the host system that changed its state.
- The value of MessageArguments[1] shall be the new value of the EnabledState property of the HostSystem instance representing the host system that changed its state.

7.3.31.6 Indication generation requirements

The indication generation requirements of 7.3.29.5 apply respectively for the AlertIndication adaptation.

7.3.32 LifecycleIndication: CIM_InstIndication

7.3.32.1 General

The LifecycleIndication adaptation models lifecycle indications of CIM instances; lifecycle indications are described in 6.1.2.3.

The LifecycleIndication adaptation adapts the CIM_InstIndication class and is based on the BasicIndication adaptation (see 7.3.29); in addition, if the ReliableIndications feature (see 7.2.4) is implemented, it is also based on the ReliableIndication adaptation (see 7.3.30).

The implementation type of the LifecycleIndication indication adaptation is: "abstract".

It is expected that the LifecycleIndication adaptation is used as a base adaptation for modeling lifecycle indications in referencing profiles.

7.3.32.2 Event definition requirements

This subclause refines the event definition requirements established by the BasicIndication adaptation (see 7.3.29.2) for the LifecycleIndication adaptation.

Recall that lifecycle indication reports secondary events (see 6.1.1). The secondary event that is reported by LifecycleIndication instances shall be described by an event definition query statement that conforms to the following ABNF rule:

```
"SELECT" WS PropertySet WS "FROM" WS LifecycleIndicationClass WS
"WHERE" WS "ISA" WS ModelElement [ WS "WHERE" SelectionExpression ]
```

`PropertySet` shall be `"*"`, or a comma-separated list of property names.

`LifecycleIndicationClass` shall be one of `CIM_InstCreation`, `CIM_InstDeletion`, or `CIM_InstModification`, or a subclass of these indication classes.

`ModelElement` shall identify a class for that the referencing profile defines a class adaptation, and for which the modeled lifecycle indication reports secondary events. The class adaptation of that class shall be stated as part of the description of the lifecycle indication adaptation in the referencing profile.

NOTE For examples that comply with this requirement, see 7.3.33 and 7.3.34.

`SelectionExpression` shall be a constant string that defines a selection expression conformant with the rules for selection expressions defined by DSP0202.

NOTE These rules provide for referencing profiles being able to define one lifecycle indication for one target adaptation per lifecycle indication adaptation. If for a particular target adaption a referencing profile intends to model lifecycle indications for different lifecycle events (such as the creation, destruction or modification of instances of the target adaptation), for each of these lifecycle events separate lifecycle indication adaptations are required. Furthermore, if lifecycle indications are to be modeled for different target adaptations, for each target adaptation separate lifecycle indication adaptations are required. As usual, if common requirements exist for such lifecycle indication adaptations, these can be defined in a common abstract base adaptation that is used as a base for the specific lifecycle indication adaptations, thereby avoiding the repetition of the commonalities.

7.3.32.3 Indication origin

The indication origin as required by 7.3.29.3 shall be the namespace of the CIM instance referenced by the value of the SourceInstanceModelPath property (see 7.3.32.4.3).

7.3.32.4 Element requirements

7.3.32.4.1 General

Table 47 lists the element requirements for the LifecycleIndication adaptation.

Table 47 – LifecycleIndication: Element requirements

Elements	Requirement	Description
Base adaptations		
BasicIndication	Mandatory	See 7.3.29.
ReliableIndication	Conditional	Condition: The ReliableIndications feature (see 7.2.4) is implemented. See 7.3.30; note that this is a WBEM server related implementation requirement; see 7.1.
Properties		
SourceInstance	Mandatory	See 7.3.32.4.2.
SourceInstanceModelPath	Mandatory	See 7.3.32.4.3.

7.3.32.4.2 Property: SourceInstance

The value of the SourceInstance property shall be an embedded instance of the class selected in the query statement defining the event. The embedded instance shall be a copy of the instance for which the lifecycle indication is reported. If the query statement specifies a specific selection of properties (other than `"*"`), then the set of properties contained in the embedded instance shall be limited to those selected; otherwise, the embedded instance shall at least contain values for each of the properties required by the related adaptation of the selected class in the same referencing profile; see 7.3.29.2.

7.3.32.4.3 Property: SourceInstanceModelPath

The value of the SourceInstanceModelPath property shall refer to the same instance that is copied as an embedded instance through the value of the SourceInstance property.

7.3.32.5 Indication generation requirements

The indication generation requirements of 7.3.29.5 apply respectively for the LifecycleIndication adaptation.

7.3.33 ListenerDestinationRemovalIndication: CIM_InstDeletion

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The ListenerDestinationRemovalIndication adaptation models a lifecycle indication that reports the destruction of a CIM_ListenerDestination instance, as modeled in this profile by the ListenerDestination adaptation (see 7.3.23). The destruction of a ListenerDestination instance is a secondary event caused by the destruction of the represented listener destination; see 6.4.5.

The requirement level of the ListenerDestinationRemovalIndication indication adaptation is optional.

The implementation type of the ListenerDestinationRemovalIndication indication adaptation is: "indication".

Table 48 lists the element requirements for the ListenerDestinationRemovalIndication adaptation.

Table 48 – ListenerDestinationRemovalIndication: Element requirements

Elements	Requirement	Description
Base adaptations		
LifecycleIndication	Mandatory	See 7.3.32.

The requirement level of the ListenerDestinationRemovalIndication adaptation is optional.

The event reported by the ListenerDestinationRemovalIndication adaptation is defined by the following event definition query statement:

```
    SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA
    CIM_ListenerDestination
```

7.3.34 SubscriptionRemovalIndication: CIM_InstDeletion

The requirements in this subclause are WBEM server related implementation requirements; see 7.1.

The SubscriptionRemovalIndication adaptation models a lifecycle indication that reports the destruction of a CIM_AbstractIndicationSubscription instance, as modeled in this profile by the AbstractSubscription adaptation (see 7.3.25). The destruction of a CIM_AbstractIndicationSubscription instance is a secondary event caused by the destruction of the represented subscription; see 6.1.1.

The requirement level of the SubscriptionRemovalIndication indication adaptation is optional.

The implementation type of the SubscriptionRemovalIndication indication adaptation is: "indication".

Table 49 lists the element requirements for the SubscriptionRemovalIndication adaptation.

Table 49 – SubscriptionRemovalIndication: Element requirements

Elements	Requirement	Description
Base adaptations		
LifecycleIndication	Mandatory	See 7.3.32.

The requirement level of the SubscriptionRemovalIndication adaptation is optional.

The event reported by the SubscriptionRemovalIndication adaptation is defined by the following query statement:

```
    SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA
    CIM_AbstractIndicationSubscription
```

7.4 Reliable indication delivery

7.4.1 General

This subclause defines mechanisms for the reliable delivery of indications from an implementation to a listener as described in 6.1.5.

Implementations implementing the ReliableIndications feature (see 7.2.4) shall comply with the requirements specified in 7.4.3; note that in addition the requirements of the ReliableIndications adaptation (see 7.3.30) apply.

Implementations not implementing the ReliableIndications feature are not required to comply with the provisions in this subclause or those in 7.3.30.

Listeners implementing the ReliableIndications feature (see 7.2.4) shall comply with the provisions stated in 7.4.4. Listeners not implementing the ReliableIndications feature are not required to comply with these provisions and may ignore the sequence identifiers in received indications, as exposed by the values of the SequenceContext and SequenceNumber properties in any received CIM_Indication instances.

7.4.2 Sequence identifier and sequence identifier lifetime

This subclause defines the concepts of *sequence identifier* and *sequence identifier lifetime*.

The *sequence identifier* within an indication enables unique identification of the indications originating from a particular WBEM server to a particular WBEM listener.

A sequence identifier is composed of a sequence context and a sequence number.

NOTE The sequence number within a sequence identifier is not to be confused with the `SEQUENCE_NUMBER` attribute value that is part of the identification of the alert message that defines an alert indication; see 7.3.31.5.5.

The sequence context is required to be unique for each listener destination maintained by the indication service within a WBEM server; within that context the sequence number is required to be unique for each indication delivered from the WBEM server to the listener referenced by the listener destination. The requirements for the CIM representation of the sequence identifier in reliable indications are defined in 7.3.30.

The *sequence identifier lifetime* maintained by an implementation is a duration defined as follows:

sequence-identifier-lifetime = number-of-retry-attempts * delivery-retry-interval * 10

In this formula the number-of-retry-attempts is the number of retry attempts as indicated by the value of the DeliveryRetryAttempts property (see 7.3.2.3.3) in the IndicationService instance representing the indication service within the implementation, and the delivery-retry-interval is the duration of the delivery retry interval as indicated by the value of the DeliveryRetryInterval property (see 7.3.2.3.4) in the same instance.

Within the sequence identifier lifetime an implementation that is implementing reliable indications may attempt to retry failed indication delivery attempts, as detailed in 7.4.3, and a listener implementing reliable indications may expect the delivery of anticipated indications, as detailed in 7.4.4.

7.4.3 WBEM server requirements

7.4.3.1 General

Indication delivery is based on a publish/subscribe event paradigm, where an implementation delivers indications to subscribed listeners. The indication delivery may fail for various reasons, including unavailability of the listener or network issues. This subclause describes the requirements for the implementation that are related to reliable indication delivery. The mechanisms to deliver indications and to determine success or failure of indication delivery are protocol dependent; see the specifications of applicable protocols that specify mechanisms for indication delivery.

7.4.3.2 Prohibition of indication delivery for disabled or removed subscriptions

If a subscription is disabled or has been removed, the implementation should discard any undelivered indications for that subscription. For example, this applies if the implementation has queued indications for delivery retry, and the subscription is removed by a client before the delivery retry is executed.

7.4.3.3 Prohibition of repeated indication delivery

After an implementation has successfully delivered an indication to a listener, it shall not deliver that indication again to that same listener.

7.4.3.4 Requirements for the retry of failed indication deliveries

If the attempt to deliver an indication to a particular listener fails, the implementation shall retry the indication delivery as detailed in this subclause.

1) The implementation shall wait for the duration of the delivery retry interval, as exposed by the value of the DeliveryRetryInterval property in the IndicationService instance (see 7.3.2) representing the indication service within the implementation.
2) If the actual number of retry attempts is less than the maximum number of retry attempts as exposed by the value of the DeliveryRetryAttempts property in the IndicationService instance representing the indication service within the implementation, and the elapsed time after the first delivery is less than the sequence identifier lifetime as defined in 7.4.2, the implementation shall retry the failed indication delivery.
   - If the retry is successful, delivery of that indication to the particular listener is complete.
   - If the retry is not successful, and preconditions of step 2) still apply, then the implementation shall re-iterate starting with step 1).
   - Otherwise, the indication shall be considered as not deliverable to the particular listener, and the requirements defined in 7.4.3.5 apply.

7.4.3.5 Requirements for undeliverable indications

This subclause defines the implementation behavior if an indication has been considered unable to be delivered to a listener, as described in 7.4.3.4.

If the listener destination referencing that listener is permanent (see 7.3.23.3.3), the implementation shall record an error and shall no longer attempt to deliver that indication to that listener (that is, the implementation shall discard it). This action does not modify the listener destination and any of its subscriptions.

If the listener destination referencing that listener is transient (see 7.3.23.3.3), the implementation shall record an error and shall no longer attempt to deliver that indication to that listener (that is, the implementation shall discard it). In addition, the listener destination and its subscriptions may be removed from the implementation as described in 7.4.3.6.

7.4.3.6 Requirements for the implicit removal of subscriptions and listener destinations

An implementation may remove a subscription and the referenced listener destination if the delivery of one or more indications to the represented listener failed as described in 7.4.3.4 and 7.4.3.5.

The implementation behavior with respect to the implicit removal of subscriptions and listener destinations shall be exposed by the value of the SubscriptionRemovalAction property in the IndicationService instance representing the responsible indication service; see 7.3.2.3.5.

7.4.3.7 Behavior related to WBEM server restarts

Indications that have been generated but not yet delivered may get lost during a WBEM server crash or restart because there is not requirement to persist such indications.

If the implementation chooses an algorithm for the construction of the sequence context part of the sequence identifier (see 7.4.2) that includes the WBEM server startup time, the potential re-use of the same sequence identifier is implicitly avoided. That way listeners can deal with indication delivery failures caused by WBEM server restarts in the same way they deal with other kinds of indication delivery failures.

7.4.4 WBEM listener requirements

7.4.4.1 General

A listener shall keep track of each distinct sequence identifier of any indications received from a particular indication service for the duration of the sequence identifier lifetime maintained by that indication service, counting from the last time that sequence identifier was detected in a received indication from that indication service. If the same sequence identifier is used by two different indication services (for example, in two different implementations), the listener shall keep track of them independently.

After the lifetime of a sequence identifier expires, the listener should discard the knowledge about that sequence identifier from that indication service. After the knowledge about a sequence identifier for an indication service has been discarded by the listener, a new usage of that sequence identifier in an indication from that indication service shall be treated by the listener like a new, unknown sequence identifier from that indication service.

Keeping track of sequence identifiers in listeners enables the detection of lost and duplicate deliveries, and the detection and re-ordering of indications arriving out of order, as described in 7.4.4.5. Discarding the knowledge about sequence identifiers minimizes the resource requirements of the listener.

7.4.4.2 Determination of the expected sequence identifier of the next indication

From the sequence identifier of the last indication received from a particular implementation, a listener shall infer the expected sequence identifier of the next indication by incrementing the sequence number by 1, wrapping to an initial value of 0 if the maximum limit has been reached, and maintaining the sequence context.

7.4.4.3 Lost indications

If the sequence identifier of the next received indication sent from the same implementation does not match the expected value as described in 7.4.4.2, the listener shall consider the expected indication as a candidate for a lost indication. After waiting for the sequence identifier lifetime period as maintained by the implementation sending that indication, the listener shall conclude that the expected indication is lost.

7.4.4.4 Duplicate indications

Any additional indications received from the same implementation with the same sequence identifier shall be considered duplicates. In this case, the lifetime for the sequence identifier shall be adjusted starting with the delivery time of the most recently received duplicate indication, and adding the sequence identifier lifetime period as maintained by the implementation sending that indication.

7.4.4.5 Out-of-order indications

A listener that intends to re-establish the original order of indications before processing them needs to defer the processing of any prematurely arriving indication that does not have the expected sequence number, until the decision can be made as to whether the expected indications are lost.

If the sequence identifier of the next received indication does not match the expected sequence identifier as described in 7.4.4.2, the listener shall cache such prematurely arriving indications and wait for delivery of the indication with the expected sequence identifier for a period of time defined by the sequence identifier lifetime (as defined in 7.4.4.1) of the last received indication from the same implementation.

If the indication with the expected sequence identifier is not received during that period, the expected indication should be considered lost (see 7.4.4.3).

If the indication with the expected sequence identifier is received during that period, the indication order shall be re-ordered using their sequence numbers, such that the indications are processed in the order they were sent by the implementation.

8 Use Cases

8.1 Object Diagrams

Figure 4 depicts a DMTF object diagram. It shows CIM instances exposed by the implementation of an Example Fan profile that defines some indications (not shown in the diagram), and thus is required by DSP1001 to reference this profile, implying the implementation of respective elements defined in this profile.

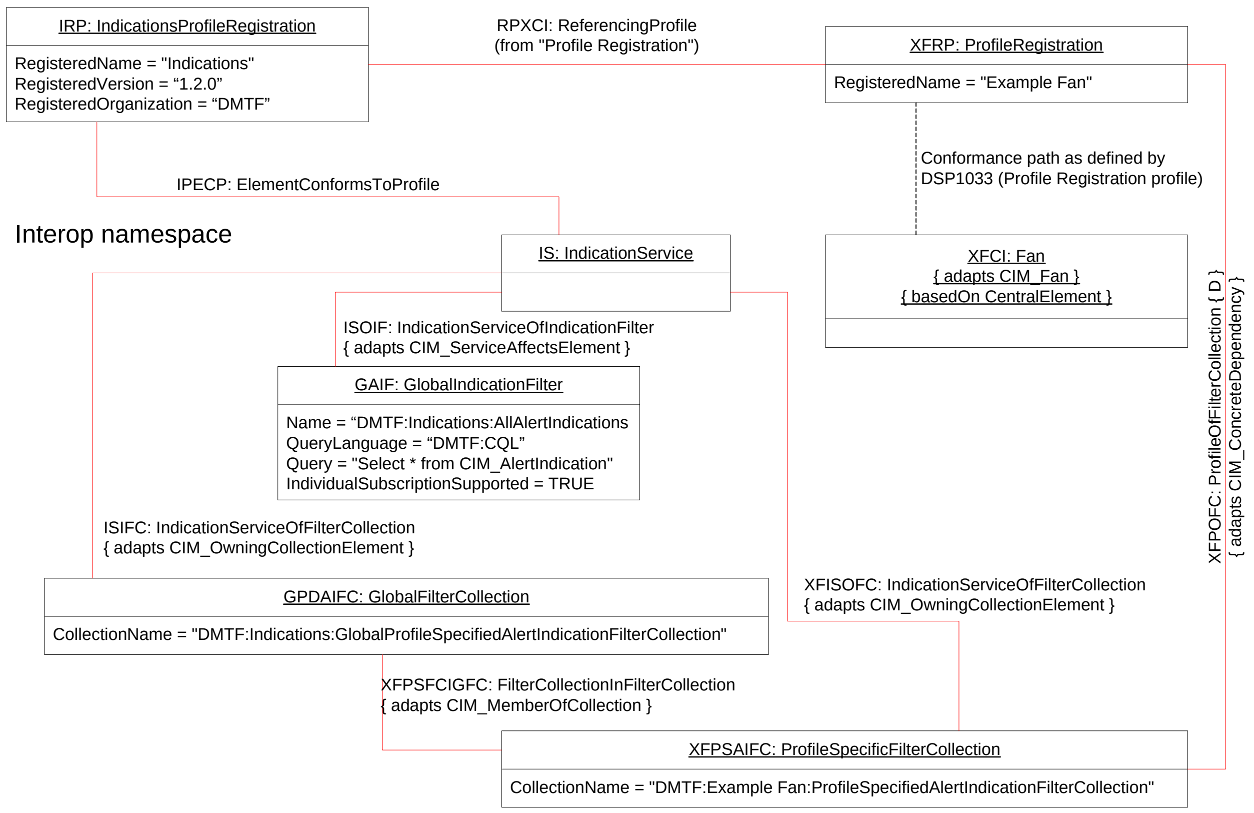

IRP: IndicationsProfileRegistration
RegisteredName = "Indications"
RegisteredVersion = “1.2.0”
RegisteredOrganization = “DMTF”
RPXCI: ReferencingProfile
(from "Profile Registration")
XFRP: ProfileRegistration
RegisteredName = "Example Fan"
Conformance path as defined by
DSP1033 (Profile Registration profile)
IPECP: ElementConformsToProfile
Interop namespace
IS: IndicationService
XFCI: Fan
{ adapts CIM_Fan }
{ basedOn CentralElement }
ISOIF: IndicationServiceOfIndicationFilter
{ adapts CIM_ServiceAffectsElement }
GAIF: GlobalIndicationFilter
Name = “DMTF:Indications:AllAlertIndications
QueryLanguage = “DMTF:CQL”
Query = "Select * from CIM_AlertIndication"
IndividualSubscriptionSupported = TRUE
XFPOFC: ProfileOfFilterCollection { D }
{ adapts CIM_ConcreteDependency }
ISIFC: IndicationServiceOfFilterCollection
{ adapts CIM_OwningCollectionElement }
GPDAIFC: GlobalFilterCollection
CollectionName = "DMTF:Indications:GlobalProfileSpecifiedAlertIndicationFilterCollection"
XFISOFC: IndicationServiceOfFilterCollection
{ adapts CIM_OwningCollectionElement }
XFPSFCIGFC: FilterCollectionInFilterCollection
{ adapts CIM_MemberOfCollection }
XFPSAIFC: ProfileSpecificFilterCollection
CollectionName = "DMTF:Example Fan:ProfileSpecifiedAlertIndicationFilterCollection"

Figure 4 – DMTF object diagram: Global and profile-specific filter collections

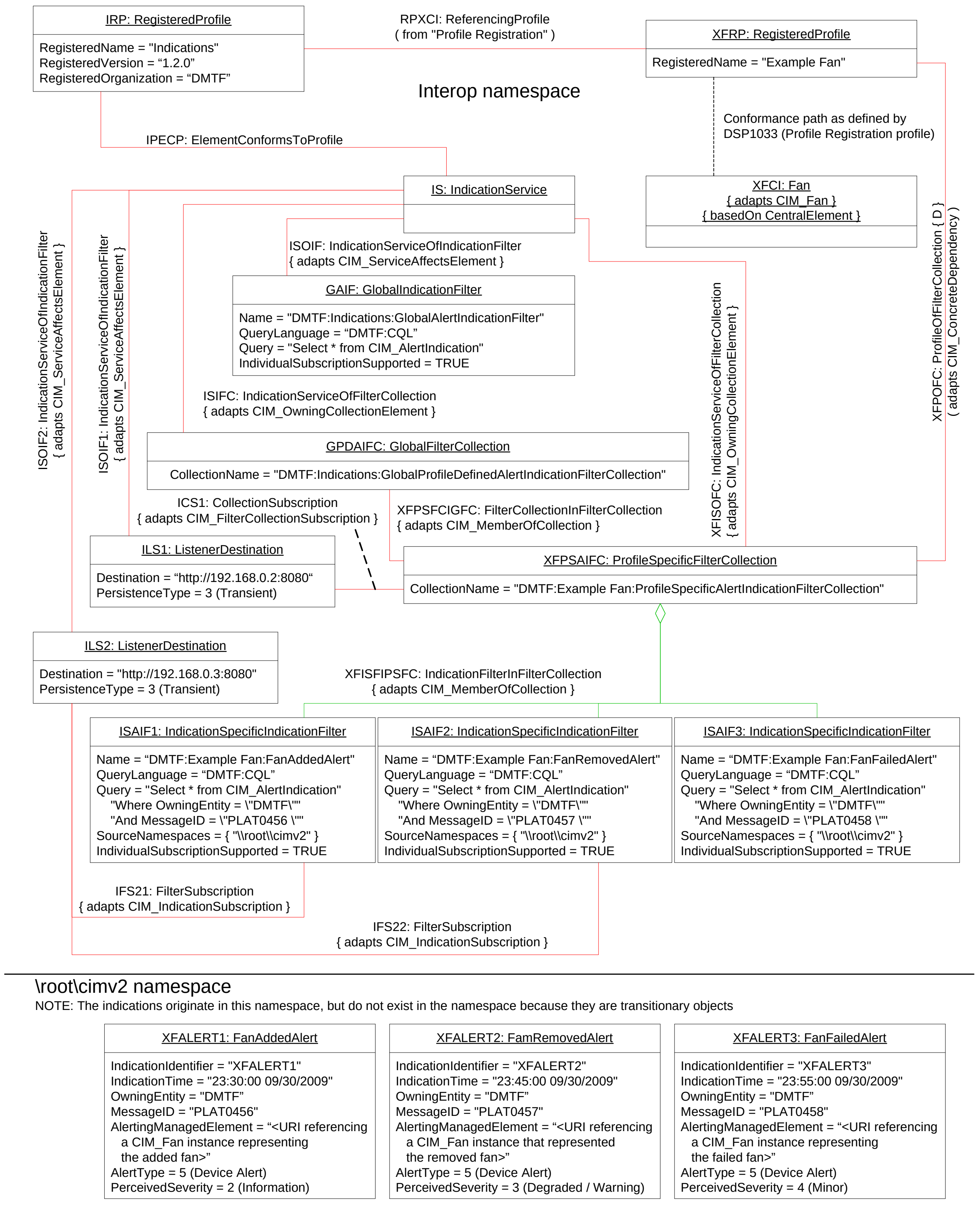
The implemented version of this profile is represented by the RegisteredProfile instance IRP, the implemented version of the Example Fan profile is represented by RegisteredProfile instance XFRP, and the reference relationship is shown by the ReferencingProfile association instance RPXCI.

The implementation of this profile exposes the IndicationService (see 7.3.2) instance IS representing the implemented indication service. It also exposes the GlobalIndicationFilter (see 7.3.16) instance GAIF representing the global indication filter covering all alert indications.

Furthermore, the implementation of this profile exposes the GlobalFilterCollection (see 7.3.22) instance GPDAIFC representing the global filter collection for alert indications with a defined coverage covering all profile defined alert indications. The implementation of the Example Fan profile exposes the ProfileSpecificFilterCollection (see 7.3.21) instance XFPSAIFC representing the related profile-specific filter collection for alert indications with a defined coverage covering all alert indications defined in the Example Fan profile.

The global filter collection for alert indications represented by GPDAIFC contains the profile-specific filter collection for alert indications represented by XFPSAIFC; this containment relationship is represented by the FilterCollectionInFilterCollection (see 7.3.20) instance XFPSFCIGFC. Because the coverage of the global filter collection is explicitly represented by containment, in this case its coverage is inspectable by clients. However, the CIM representation of the contained profile-specific filter collection for alert indications represented by XFPSAIFC does not expose any contained elements. In that case clients would require prior knowledge of the defined coverage, that is, all alert indications defined in the Example Fan profile, which (because of the explicitly represented containment relationship) is in this example also the coverage of the global filter collection for alert indications represented by GPDAIFC.

2724 Figure 5 depicts a DMTF object diagram. It shows a variant of the situation illustrated in Figure 4.


IRP: RegisteredProfile
RegisteredName = "Indications"
RegisteredVersion = "1.2.0"
RegisteredOrganization = "DMTF"
RPXCI: ReferencingProfile (from "Profile Registration")
XFRP: RegisteredProfile
RegisteredName = "Example Fan"
Interop namespace
Conformance path as defined by DSP1033 (Profile Registration profile)
IPECP: ElementConformsToProfile
IS: IndicationService
XFCI: Fan { adapts CIM_Fan } { basedOn CentralElement }
ISOIF2: IndicationServiceOfIndicationFilter { adapts CIM_ServiceAffectsElement }
ISOIF1: IndicationServiceOfIndicationFilter { adapts CIM_ServiceAffectsElement }
ISOIF: IndicationServiceOfIndicationFilter { adapts CIM_ServiceAffectsElement }
GAIF: GlobalIndicationFilter
Name = "DMTF:Indications:GlobalAlertIndicationFilter"
QueryLanguage = "DMTF:CQL"
Query = "Select * from CIM_AlertIndication"
IndividualSubscriptionSupported = TRUE
XFPOFC: ProfileOfFilterCollection { D } { adapts CIM_ConcreteDependency }
XFISOFC: IndicationServiceOfFilterCollection { adapts CIM_OwningCollectionElement }
ISIFC: IndicationServiceOfFilterCollection { adapts CIM_OwningCollectionElement }
GPDAIFC: GlobalFilterCollection
CollectionName = "DMTF:Indications:GlobalProfileDefinedAlertIndicationFilterCollection"
ICS1: CollectionSubscription { adapts CIM_FilterCollectionSubscription }
XFPSFCIGFC: FilterCollectionInFilterCollection { adapts CIM_MemberOfCollection }
ILS1: ListenerDestination
Destination = "http://192.168.0.2:8080"
PersistenceType = 3 (Transient)
XFPSAIFC: ProfileSpecificFilterCollection
CollectionName = "DMTF:Example Fan:ProfileSpecificAlertIndicationFilterCollection"
ILS2: ListenerDestination
Destination = "http://192.168.0.3:8080"
PersistenceType = 3 (Transient)
XFISFIPSFC: IndicationFilterInFilterCollection { adapts CIM_MemberOfCollection }
ISAIF1: IndicationSpecificIndicationFilter
Name = "DMTF:Example Fan:FanAddedAlert"
QueryLanguage = "DMTF:CQL"
Query = "Select * from CIM_AlertIndication" "Where OwningEntity = \"DMTF\"" "And MessageID = \"PLAT0456 \""
SourceNamespaces = { "\\root\\cimv2" }
IndividualSubscriptionSupported = TRUE
ISAIF2: IndicationSpecificIndicationFilter
Name = "DMTF:Example Fan:FanRemovedAlert"
QueryLanguage = "DMTF:CQL"
Query = "Select * from CIM_AlertIndication" "Where OwningEntity = \"DMTF\"" "And MessageID = \"PLAT0457 \""
SourceNamespaces = { "\\root\\cimv2" }
IndividualSubscriptionSupported = TRUE
ISAIF3: IndicationSpecificIndicationFilter
Name = "DMTF:Example Fan:FanFailedAlert"
QueryLanguage = "DMTF:CQL"
Query = "Select * from CIM_AlertIndication" "Where OwningEntity = \"DMTF\"" "And MessageID = \"PLAT0458 \""
SourceNamespaces = { "\\root\\cimv2" }
IndividualSubscriptionSupported = TRUE
IFS21: FilterSubscription { adapts CIM_IndicationSubscription }
IFS22: FilterSubscription { adapts CIM_IndicationSubscription }
\root\cimv2 namespace
NOTE: The indications originate in this namespace, but do not exist in the namespace because they are transitionary objects
XFALERT1: FanAddedAlert
IndicationIdentifier = "XFALERT1"
IndicationTime = "23:30:00 09/30/2009"
OwningEntity = "DMTF"
MessageID = "PLAT0456"
AlertingManagedElement = "<URI referencing a CIM_Fan instance representing the added fan>"
AlertType = 5 (Device Alert)
PerceivedSeverity = 2 (Information)
XFALERT2: FamRemovedAlert
IndicationIdentifier = "XFALERT2"
IndicationTime = "23:45:00 09/30/2009"
OwningEntity = "DMTF"
MessageID = "PLAT0457"
AlertingManagedElement = "<URI referencing a CIM_Fan instance that represented the removed fan>"
AlertType = 5 (Device Alert)
PerceivedSeverity = 3 (Degraded / Warning)
XFALERT3: FanFailedAlert
IndicationIdentifier = "XFALERT3"
IndicationTime = "23:55:00 09/30/2009"
OwningEntity = "DMTF"
MessageID = "PLAT0458"
AlertingManagedElement = "<URI referencing a CIM_Fan instance representing the failed fan>"
AlertType = 5 (Device Alert)
PerceivedSeverity = 4 (Minor)

2725

2726 **Figure 5 – DMTF object diagram: Filter collections and contained indication filters**

The first difference from the situation shown in Figure 4 is that in Figure 5 the profile-specific filter collection for alert indications represented by XFISAIFC contains three indication filters, represented by the IndicationSpecificIndicationFilter instances ISAIF1, ISAIF2 and ISAIF3. Hence the coverage of the profile-specific filter collection for alert indications represented by XFPSAIFC is now defined by the contained indication filters, that is, it covers the three alert indications described by the alert messages with the IDs PLAT0456, PLAT0457, and PLAT0458.

It is important to recapture that — as with any indication gate — the presence of the CIM representation of specific indication filters does not indicate that the covered indications are actually implemented. The semantics of indication gates are defined with respect to *filtering*, but *not* with respect to *generating*, indications (see 7.3.11.2 and 7.3.17.2). Thus, a subscribed listener is guaranteed only to be delivered any *generated* indication that is within the coverage of the indication gate, but the *generation* of the indication is not guaranteed. For that reason referencing profiles need to model other elements — such as capabilities — for the purpose of conveying the information about which indications defined in the referencing profile are actually implemented and thus generated when the respective event occurs; the definition of such mechanisms is outside the scope of this profile.

The second difference between Figure 4 and Figure 5 is that in Figure 5 listener destinations are represented by the ListenerDestination instances ILS1 and ILS2. The listener referenced by ILS1 is subscribed to the profile-specific filter collection represented by XFPSAIFC, and the listener referenced by ILS1 is subscribed to the indication-specific indication filters represented by ISAIF1 and ISAIF2.

Lastly, the representations of three indications are shown at the bottom of Figure 5, along with their origin namespace. Each of these indications is within the coverage of the indication filter represented directly above it. Thus, the alert indications represented by XFALERT1 and XFALERT2 are delivered to both the listeners represented by ILS1 and ILS2, whereas XFALERT3 is only delivered to ILS1.

Figure 6 depicts the DMTF object diagram for an implementation that supports a fixed number of listener destinations.

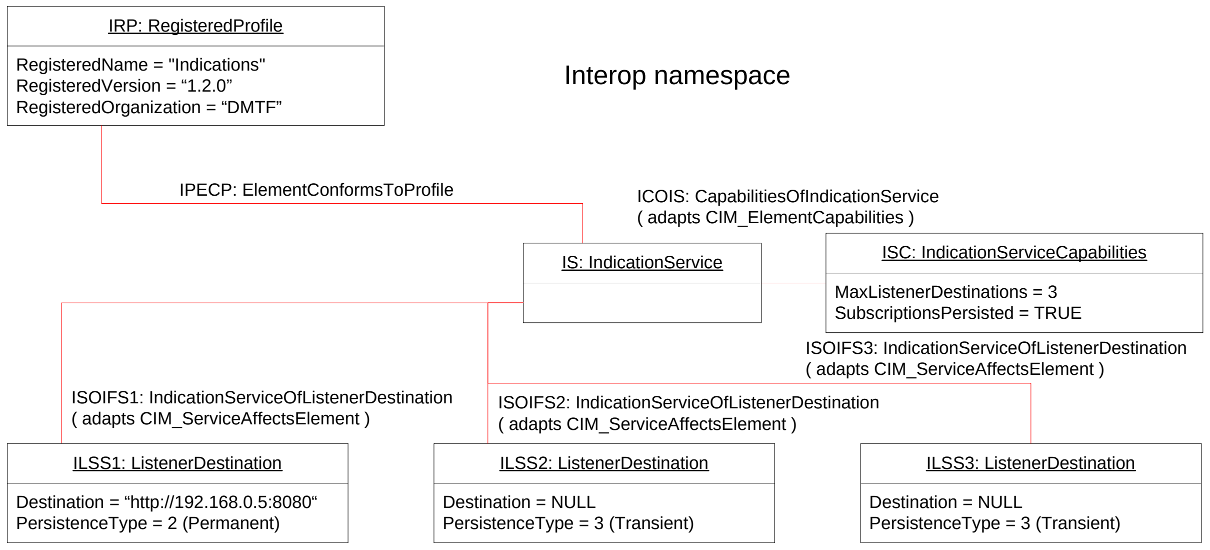

IRP: RegisteredProfile
RegisteredName = "Indications"
RegisteredVersion = "1.2.0"
RegisteredOrganization = "DMTF"
Interop namespace
IPECP: ElementConformsToProfile
ICOIS: CapabilitiesOfIndicationService
(adapts CIM_ElementCapabilities)
IS: IndicationService
ISC: IndicationServiceCapabilities
MaxListenerDestinations = 3
SubscriptionsPersisted = TRUE
ISOIFS3: IndicationServiceOfListenerDestination
(adapts CIM_ServiceAffectsElement)
ISOIFS1: IndicationServiceOfListenerDestination
(adapts CIM_ServiceAffectsElement)
ISOIFS2: IndicationServiceOfListenerDestination
(adapts CIM_ServiceAffectsElement)
ILSS1: ListenerDestination
Destination = "http://192.168.0.5:8080"
PersistenceType = 2 (Permanent)
ILSS2: ListenerDestination
Destination = NULL
PersistenceType = 3 (Transient)
ILSS3: ListenerDestination
Destination = NULL
PersistenceType = 3 (Transient)

Figure 6 – DMTF object diagram: Static listener destinations

In the example shown in Figure 6, an implementation supports a maximum of three listener destinations, indicated by the value of the MaxListenerDestinations property in the IndicationServiceCapabilities instance ISC that describes the capabilities of the indication service within the implementation. The three listener destinations are represented by the three respective ListenerDestination instances ILSS1, ILSS2, and ILSS3. The listener destination represented by ILSS1 is currently configured as a permanent listener destination, referencing the listener reachable under URI "http://192.168.0.5:8080". The listener destinations represented by ILSS2 and ILSS3 currently are free listener destinations as indicated by the value Null for the Destination property, that is, they are not currently configured for a specific listener. A client can request modifications of any of the listener destinations in order to reference a desired listener for indication delivery by modifying the representing ListenerDestination instances.

8.2 LocateIndicationService: Locate the indication service provided by an implementation of this profile

8.2.1 Preconditions

The client knows the following:

- The identifying information of a WBEM server (for example, its IP address and the port number if the WBEM server implements CIM operations over http as described in DSP0223)
- Name, required version, and registered organization of this profile as stated in 7.3.5

8.2.2 Flow of activities

1) The client obtains all IndicationsProfileRegistration instances (see 7.3.5), applying respective use cases described in DSP1033 to locate CIM_RegisteredProfile instances representing profile registrations of particular profiles and selecting those instances where the values of the RegisteredName, RegisteredVersion, and RegisteredOrganization properties match the required input values.

   The result is zero or more IndicationsProfileRegistration instances (see 7.3.23).

NOTE 1 Typically only one instance is returned, but if this profile is implemented more than once within the identified WBEM server, more than one instance may be returned.

   If no instance was detected, this use case is complete and the client knows that the required version of this profile is not implemented within the WBEM server. If one or more instances were detected, any of them represents the required version of this profile, and the client can select any of these for further processing.

2) The client applies use cases described in DSP1033 in order to locate instances of the IndicationService adaptation that is the central class adaptation defined in this profile.

   The result is zero or one IndicationService instances (see 7.3.2).

NOTE 2 Technically, more than one instance could be returned, but that would indicate a non-compliant implementation of this profile.

   If no instance was detected, this use case is complete and the client knows that an indication service is not presently active within the identified WBEM server. If one or more instances were detected, any of them represents an indication service compliant to the requirements specified in this profile, and the client can select any of these for further processing.

8.2.3 Postconditions

Unless errors occurred, the client either knows an IndicationService instance (including its object path) representing an indication service within the identified WBEM server with a behavior compliant to the requirements specified in this profile or knows that either this profile is not implemented within the identified WBEM server or that no indication service is presently active within the identified WBEM server.

8.3 LocateProfileIndicationService: Locate the indication service responsible for delivering indications defined by a referencing profile

8.3.1 Preconditions

The client knows the following:

- The ProfileRegistration instance (including its object path) representing the profile registration of the referencing profile

8.3.2 Flow of activities

1) For the input ProfileRegistration instance, find the IndicationsProfileRegistration instances (see 7.3.5) associated through ReferencedProfile instances (see DSP1033) (for example, using the GetAssociatedInstancesWithPath( ) operation).

   The result is zero or one IndicationsProfileRegistration instances (see 7.3.5).

NOTE 1 Technically, more than one instance could be returned, but that would indicate a non-compliant implementation of the referencing profile.

   If no instance was detected, this use case is complete and the client knows that the implementation of the referencing profile did not implement indications.

2) For the IndicationsProfileRegistration instance obtained in step 1), find the IndicationService instances (see 7.3.2) associated through ElementConformsToProfile instances (see 7.3.6) (for example, using the GetAssociatedInstancesWithPath( ) operation).

   The result is zero or one IndicationService instances (see 7.3.2).

NOTE 2 Technically, more than one instance could be returned, but that would indicate a non-compliant implementation of this profile.

8.3.3 Postconditions

Unless errors occurred, the client knows an IndicationService instance (including its object path) representing an indication service that is responsible for delivering indications defined by the referencing profile.

8.4 DetermineIndicationServiceCapabilities: Determine the capabilities of an indication service

8.4.1 Preconditions

The client knows all of the following:

- a copy of the IndicationService instance (including its object path) representing the indication service within the implementation

NOTE For example, that IndicationService instance could be obtained by applying the LocateIndicationService use case (see 8.2) or the LocateProfileIndicationService use case (see 8.3).

8.4.2 Flow of activities

1) Inspecting property values of the IndicationService instance (see 7.3.2.3), the client can already determine some aspects of the behavior of the represented indication service.

For example, the value of the FilterCreationEnabled property indicates whether the support for dynamic indication filters as modeled by the DynamicIndicationFilters feature (see 7.2.1) is available.

The values of the DeliveryRetryAttempts, the DeliveryRetryInterval, the SubscriptionRemovalAction, and the SubscriptionRemovalTimeInterval indicate if and to what extent the support for reliable indications as modeled by the ReliableIndications feature (see 7.2.4) is available.

2) Find the IndicationsServiceCapabilities instance (see 7.3.7) representing the capabilities of the input indication service, by traversing the CIM_ServiceAffectsElement association modeled by the CapabilitiesOfIndicationService association adaptation (see 7.3.8) by invoking the GetAssociatedInstancesWithPath( ) operation with the following actual values for the input parameters:

   – InstanceName: the object path to the input IndicationService instance

   – AssocClass: "CIM_ElementCapabilities", the adapted class of the CapabilitiesOfIndicationService association adaptation

   – ResultClass: "CIM_IndicationServiceCapabilities", the adapted class of the IndicationServiceCapabilities adaptation

   The result is zero or one IndicationServiceCapabilities instance.

NOTE Technically, more than one instance could be returned, but that would indicate a non-compliant implementation of this profile.

   If an IndicationServiceCapabilities instance was returned, the use case continues with step 3); otherwise, it continues with step 4).

3) Inspect the property values of the returned IndicationServiceCapabilities instance (see 7.3.7). The values of those properties with names ending with "IsSettable" enable the client to determine whether client modification of respective aspects of the behavior of the input indication service is possible. The values of the MaxListenerDestinations and the MaxActiveSubscriptions properties expose the upper limits for the number of listener destinations and for the number of subscriptions supported by the indication service, and the value of the SubscriptionsPersisted property exposes whether subscriptions are persisted over restarts of the input indication service. This step completes this use case.

4) Continue here after step 2) if no IndicationServiceCapabilities instance was returned. In this case, client modification of the indication service is not supported, and the upper limits for the number of supported listener destinations and subscriptions is not exposed by the implementation; in addition, whether subscriptions are persisted over indication service restarts is not exposed.

8.4.3 Postconditions

Unless errors occurred, the client knows the capabilities of the input indication service as far as it is exposed by the representing IndicationService instance, by the related IndicationServiceCapabilities instance, and by initial behavior specified in this profile.

8.5 ModifyIndicationService: Modify functional aspects of an indication service

The client knows all of the following:

- a copy of the IndicationService instance (including its object path) (see 7.3.2) representing the indication service within the implementation (see the LocateIndicationService use case in 8.2)

- a copy of the IndicationServiceCapabilities instance (including its object path) (see 7.3.7) representing the capabilities of the indication service within the implementation (See the DetermineIndicationServiceCapabilities use case in 8.4.)

8.5.1 Flow of activities

1) Inspect the property values in the input IndicationsServiceCapabilities instance (see 7.3.7) representing the capabilities of the input indication service to determine which properties in the IndicationService instance are modifiable. (See step 3) in the DetermineIndicationServiceCapabilities use case in 8.4.)
2) If admissible by the determination of step 1), in the input local copy of the input IndicationService instance, modify property values as desired. For example, if the value of the DeliveryRetryAttemptsIsSettable property in the IndicationServiceCapabilities instance is True, a modification of the corresponding DeliveryRetryAttempts property in the IndicationService instance is admissible.
3) Use the ModifyInstance( ) operation to request the desired change in the behavior of the indication service, providing the modified copy of the IndicationService instance as the actual value of the ModifiedInstance parameter.

8.5.2 Postconditions

Unless errors occurred, the desired change of functional aspects of the input indication service is effective.

8.6 ListListenerDestinations: List all listener destinations exposed by an implementation

8.6.1 Preconditions

The client knows all of the following:

- the object path to the IndicationService instance representing the indication service within the implementation (see 8.2)

8.6.2 Flow of activities

1) Find all listener destinations within the responsibility of the indication service by traversing the CIM_ServiceAffectsElement association modeled by the IndicationServiceOfListenerDestination adaptation (see 7.3.24) by invoking the GetAssociatedInstancesWithPath( ) operation with the following actual values for the input parameters:
   – InstanceName: the object path to the input IndicationService instance
   – AssocClass: "CIM_ServiceAffectsElement", the adapted class of the IndicationServiceOfListenerDestination adaptation
   – ResultClass: "CIM_ListenerDestination", the adapted class of the ListenerDestination adaptation

   The result is a set of ListenerDestination instances (see 7.3.23).

8.6.3 Postconditions

Unless errors occurred, the client knows all ListenerDestination instances (including their object paths) representing all the listener destinations maintained by the implementation.

8.7 SelectListenerDestination: Select an existing listener destination referencing a desired listener

8.7.1 Preconditions

The client knows all of the following:

- the object path to the IndicationService instance representing the indication service within the implementation (see 8.2)
- the URI exposed by the desired listener
- the particular protocol to be applied when delivering these indications

8.7.2 Flow of activities

1) Execute the ListListenerDestinations use case (see 8.6).

   The result is a set of ListenerDestination instances (see 7.3.23).

2) Inspect each ListenerDestination instance resulting from step 1) by checking the value of the Destination property against the input URI, and by checking whether the value of the Protocol property matches the particular protocol for this use case.

   If both conditions are met, the located ListenerDestination represents a listener destination that within the implementation represents the particular listener, and this use case is complete; otherwise, the client needs to repeat step 2), inspecting further ListenerDestination instances from the result of step 1).

3) If all result elements from step 1) checked in step 2) did not yield a ListenerDestination instance referencing the listener, then this use case is complete and the client knows that the listener is not presently represented by a listener destination within the implementation.

8.7.3 Postconditions

Unless errors occurred, the client either knows a ListenerDestination instance (including its object path) representing a listener destination within the implementation that references the particular listener, or knows that the listener is not referenced by any listener destination within the implementation.

In the latter case, and if the implementation has also implemented the dynamic creation of listener destinations, the client could apply the CreateListenerDestination use case (see 8.8) to dynamically create a respective listener destination within the implementation that represents the desired listener.

8.8 CreateListenerDestination: Create a new listener destination

8.8.1 Preconditions

The client knows all of the following:

- The same as for the SelectListenerDestination use case; see 8.7.1.

8.8.2 Flow of activities

1) Execute the SelectIndicationFilter use case (see 8.7).

   If a listener destination referencing the desired listener is found, use that; in this case, this use case is complete.

2) Prepare a local instance of the CIM_ListenerDestination class that complies with the requirements of the ListenerDestination adaptation (see 7.3.23), inserting property values as follows:

   – Destination: the identification of the listener that the new listener destination is to reference, using the format required in 7.3.23.3.2. The format needs to be compatible with the requested protocol.

   – PersistenceType: the durability requested for the new listener destination, using the format required in 7.3.23.3.3.

   – Protocol: the protocol to used for the communication with the listener, using the format required by the CIM schema definition of the CIM_ListenerDestination class.

3) Request the creation of the new listener destination in the implementation by invoking the CreateInstance( ) operation, providing the CIM_ListenerDestination instance prepared in step 2) as the actual value of the NewInstance parameter.

   If successful, the operation returns the object path of the ListenerDestination instance representing the newly created listener destination.

   If not successful, the operation returns a CIM status code providing details about the failure (see 7.3.23.3.4).

8.8.3 Postconditions

Unless errors occurred, the client knows the object path of a ListenerDestination instance representing a listener destination referencing the desired listener that either preexisted or was created; otherwise, the client knows details about why it was not possible to find or dynamically create the respective listener destination.

8.9 FindFreeListenerDestination: Find a free listener destination

8.9.1 Preconditions

The client knows all of the following:

- the object path to the IndicationService instance representing the indication service within the implementation (see 8.2)

8.9.2 Flow of activities

1) Execute the ListListenerDestinations use case (see 8.6).

   The result of this step is the set of ListenerDestination instances (including their object paths) representing all the listener destinations within the implementation.

2) From the result of step 1), select a free listener destination; free listener destinations are represented by those ListenerDestination instances where the value of the Destination property is Null.

8.9.3 Postconditions

Unless errors occurred, the client knows a free listener destination, or knows that presently no free listener destinations exist within the implementation.

8.10 ModifyListenerDestination: Modify an existing listener destination

8.10.1 Preconditions

The client knows all of the following:

- a local copy of a ListenerDestination instance (see 7.3.23)

NOTE For example, the listener destination and its representing ListenerDestination instance might have been obtained by executing the FindFreeListenerDestination use case described in 8.9.

8.10.2 Flow of activities

1) Modify the local copy of the ListenerDestination instance, maintaining compliance with the requirements of the ListenerDestination adaptation (see 7.3.23).
2) Modify the listener destination maintained by the implementation by invoking the ModifyInstance( ) operation, providing the CIM_ListenerDestination instance prepared in step 1) as the actual value of the ModifiedInstance parameter.

   If successful, the operation returns without error; otherwise, the operation returns a CIM status code providing details about the failure (see 7.3.23.3.6).

8.10.3 Postconditions

Unless errors occurred, the listener destination represented by the input ListenerDestination instance was modified; otherwise, the client knows details about why it was not possible to modify the represented listener destination.

8.11 DeleteListenerDestination: Delete an existing listener destination

8.11.1 Preconditions

The client knows all of the following:

- the object path to a ListenerDestination instance (see 7.3.23)

8.11.2 Flow of activities

1) For the input ListenerDestination instance, find all AbstractSubscription instances (see 7.3.25) referencing the ListenerDestination instance (for example, using the GetReferencingInstancePaths( ) operation).
2) Delete all subscriptions referencing the input listener destination by executing the DeleteSubscription use case (see 8.21) for each AbstractSubscription instance returned by step 1).
3) Invoke the DeleteInstance( ) operation on the input ListenerDestination instance, effecting the deletion of the referenced listener destination.

8.11.3 Postconditions

Unless errors occurred, the input listener destination is deleted and no longer represented by any ListenerDestination instances.

8.12 FindIndicationFilter: Find an indication filter covering a particular indication

8.12.1 Preconditions

The client knows all of the following:

- the object path to the IndicationService instance representing the indication service within the implementation (see 7.3.2)
- an implemented indication. Knowledge about whether or not a particular indication is actually implemented could for example be obtained by inspecting respective capabilities exposed by an implementation of a referencing profile that defines an adaptation of the particular indication.

8.12.2 Flow of activities

1) Find all indication filters within the responsibility of the indication service by traversing the CIM_ServiceAffectsElement association modeled by the IndicationServiceOfIndicationFilter association adaptation (see 7.3.14) by invoking the GetAssociatedInstancesWithPath( ) operation with the following actual values for the input parameters:

   – InstanceName: the object path to the input IndicationService instance

   – AssocClass: "CIM_ServiceAffectsElement", the adapted class of the IndicationServiceOfIndicationFilter association adaptation

   – ResultClass: "CIM_IndicationFilter", the adapted class of the IndicationFilter adaptation

   The result of this step is a set of IndicationFilter instances (see 7.3.11).

2) Inspect each IndicationFilter instance resulting from step 1) by first checking the value of the QueryLanguage property. If the query language indicated by that value is interpretable by the client, interpret the query statement presented by the value of the Query property; otherwise, continue inspecting the next IndicationFilter instance returned by step 1).

   If the desired indication is not within the coverage as expressed by the query statement, then continue inspecting the next IndicationFilter instance returned by step 1).

3) If the client desires to subscribe to the indication filter, continue by inspecting the IndicationFilter instance resulting from step 1) by checking whether the value of the IndividualSubscriptionSupported property is True. If so, this use case is complete; otherwise, continue with step 2) inspecting the next IndicationFilter instance returned by step 1); otherwise, this use case is complete.

8.12.3 Postconditions

Unless errors occurred, and if step 3) produced a suitable IndicationFilter instance, the client by that instance (including its object path) knows an indication filter that covers the desired indication and that supports individual subscriptions; otherwise, the client knows that within the responsibility of the indication service no such indication filter exists.

8.13 DetermineQueryLanguages: Determine the set of query languages supported for query statements

8.13.1 Preconditions

The client knows all of the following:

- The same as for the FindIndicationFilter use case described in 8.12.1.

NOTE The procedure outlined in this use case is only an auxiliary approach to be pursued if preliminary knowledge about the query languages supported by an implementation is not available to the client.

8.13.2 Flow of activities

1) Execute steps 1) and 2) of the FindIndicationFilter use case (see 8.9), but vary step 2) to collect the query languages applied by all the inspected indication filters.

8.13.3 Postconditions

Unless errors occurred, the client knows all the query languages in use by existing indication filters.

NOTE Because not all query languages supported by an implementation might be in use by indication filters, the set of query languages obtained by executing this use case is actually an open subset of the set of supported query languages.

8.14 CreateIndicationFilter: Create a dynamic indication filter covering a particular indication

8.14.1 Preconditions

The client knows all of the following:

- The same as for the FindIndicationFilter use case described in 8.12.1.

8.14.2 Flow of activities

1) Execute the FindIndicationFilter use case (see 8.9).

   If a suitable indication filter covering the desired indication is found, use that; in this case, this use case is complete.

2) If not already done previously, execute step 1) of the DetermineIndicationServiceCapabilities use case (see 8.4) and determine by the value of the FilterCreationEnabled property whether the support for dynamic indication filters as modeled by the DynamicIndicationFilters feature (see 7.2.1) is available.

3) If the set of query languages supported by the implementation is not known a priori, execute the DetermineQueryLanguages use case (see 8.13).

4) Prepare a local instance of the CIM_IndicationFilter class that complies with the requirements of the DynamicIndicationFilter adaptation (see 7.3.13), inserting property values as follows:

   – QueryLanguage: a query language supported by the implementation; see 7.3.11.3.6.

   – Query: the query statement covering the desired set of indications; see 7.3.11.3.5.

     NOTE Additional constraints on properties of the CIM_Indication class selected by the query statement may be specified through the WHERE clause; however, if the implementation is unable to comply with these constraints, the operation will fail.

   – SourceNamespaces[]: a list of local namespace names identifying the namespaces considered as ; see 7.3.11.3.3.

5) Request the creation of the new dynamic indication filter in the implementation by invoking the CreateInstance( ) operation, providing the CIM_IndicationFilter instance prepared in step 4) as the actual value of the NewInstance parameter.

   If successful, the operation returns the object path of the DynamicIndicationFilter instance representing the newly created dynamic indication filter.

   If not successful, the operation returns a CIM status code providing details about the failure (see 7.3.13.2.2).

8.14.3 Postconditions

Unless errors occurred, the client knows the object path of an IndicationFilter instance representing an indication filter covering the desired indication that either preexisted or was dynamically created; otherwise, the client knows details about why it was not possible to find or dynamically create the respective indication filter.

8.15 ModifyIndicationFilter: Modify a dynamic indication filter

8.15.1 Preconditions

The client knows all of the following:

- a local copy of an DynamicIndicationFilter instance (see 7.3.13)

NOTE For example, that dynamic indication filter and its representing DynamicIndicationFilter instance might have been created by executing the CreateIndicationFilter use case; see 8.14.

8.15.2 Flow of activities

1) Modify the local copy of the DynamicIndicationFilter instance, maintaining compliance with the requirements of the DynamicIndicationFilter adaptation (see 7.3.13).
2) Modify the dynamic indication filter maintained by the implementation by invoking the ModifyInstance( ) operation, providing the DynamicIndicationFilter instance prepared in step 1) as the actual value of the ModifiedInstance parameter.
3) If successful, the operation returns without error; otherwise, the operation returns a CIM status code providing details about the failure (see 7.3.13.2.4).

8.15.3 Postconditions

Unless errors occurred, the dynamic indication filter represented by the input DynamicIndicationFilter instance was modified; otherwise, the client knows details about why it was not possible to modify the represented dynamic indication filter.

8.16 DeleteIndicationFilter: Delete a dynamic indication filter

8.16.1 Preconditions

The client knows all of the following:

- the object path to a DynamicIndicationFilter instance (see 7.3.13)

8.16.2 Flow of activities

1) For the input DynamicIndicationFilter instance, find all AbstractSubscription instances (see 7.3.25) referencing the DynamicIndicationFilter instance (for example, using the GetReferencingInstancePaths( ) operation).
2) Delete all subscriptions referencing the input listener destination, by executing the DeleteSubscription use case (see 8.21) for each AbstractSubscription instance returned by step 1).
3) Invoke the DeleteInstance( ) operation on the input DynamicIndicationFilter instance, effecting the deletion of the referenced dynamic indication filter.

8.16.3 Postconditions

Unless errors occurred, the input dynamic indication filter is deleted and no longer represented by any DynamicIndicationFilter instances.

8.17 CheckCollectionCoverage: Check the coverage of a filter collection

8.17.1 Preconditions

The client knows all of the following:

- a local copy of a StaticFilterCollection instance (see 7.3.17), and the object path referencing the original StaticFilterCollection instance within the implementation

8.17.2 Flow of activities

1) Check whether the input filter collection contains any elements by resolving — from the StaticFilterCollection instance — the CIM_ConcreteComponent association as modeled by the IndicationFilterInFilterCollection association adaptation (see 7.3.19) and the FilterCollectionInFilterCollection association adaptation (see 7.3.20).

   If no contained elements are discovered, a defined coverage may apply as the coverage; in this case, skip to step 4).

2) For each of the contained elements found in step 1), determine the contributed coverage and add that to the resulting aggregated coverage of the input filter collection.

   In the case of a contained indication filter, the contributed coverage is determined by inspecting the values of the QueryLanguage property and that of the Query property containing the query statement.

   In the case of a contained filter collection, the contributed coverage is determined by recursively applying this use case (8.17).

3) Aggregate the contributed coverage of each contained element as determined in step 2) into the resulting aggregated coverage of the input filter collection. After completing this step the client knows the aggregated coverage of the input filter collection, and this use case is complete.

4) This step applies if no contained elements were discovered in steps 2) and 3).

   Check the value of the CollectionName property in the StaticFilterCollection instance for the pattern required for the name the global filter collection covering all instance lifecycle indications, as detailed in 7.3.22.4.4.

   If the pattern matches, the client knows that the represented filter collection is the global filter collection covering all instance lifecycle indications; in this case, the client knows that the coverage of the input filter collection is all instance lifecycle indications and this use case is complete.

5) Check the value of the CollectionName property in the StaticFilterCollection instance for the pattern required for the name of global filter collections for profile defined indications, as defined in 7.3.22.

   If the pattern matches, the client knows that the represented filter collection is a global filter collection for profile defined indications with a defined coverage as detailed in 7.3.22. The client needs to have a priori knowledge about the defined coverage of each referencing profile, and this use case is complete.

6) Check the value of the CollectionName property in the StaticFilterCollection instance for the pattern required for the name of profile-specific filter collections as defined in 7.3.21.2.2.

If the pattern matches, the client knows that the input filter collection is a profile-specific filter collection with a defined coverage as detailed in 7.3.21.3. The client needs to have a priori knowledge about the defined coverage of the identified referencing profile, and this use case is complete.

7) If the input filter collection does not match any of the types determined in steps 4), 5), and 6), then no defined coverage applies. Furthermore, because no contained elements were discovered in step 2), the coverage of the input filter collection is empty (that is, it does not cover any indications).

8.17.3 Postconditions

Unless errors occurred, or in the cases determined in steps 5) and 6) above the client does not have a priori knowledge about the defined coverage(s), the client knows the coverage of the input filter collection.

8.18 ObtainNamedCollection: Obtain a named filter collection

8.18.1 Preconditions

The client knows all of the following:

- the object path to the IndicationService instance representing the indication service within the implementation (see 7.3.2)
- the name of the named filter collection, for example, the name of a global filter collection or of a profile-specific filter collection

8.18.2 Flow of activities

1) Find all filter collections within the responsibility of the indication service by traversing the CIM_ServiceAffectsElement association modeled by the IndicationServiceOfFilterCollection association adaptation (see 7.3.18) by invoking the GetAssociatedInstancesWithPath( ) operation with the following actual values for the input parameters:
   – InstanceName: the object path to the input IndicationService instance
   – AssocClass: "CIM_ServiceAffectsElement", the adapted class of the IndicationServiceOfFilterCollection association adaptation
   – ResultClass: "CIM_FilterCollection", the adapted class of the StaticFilterCollection adaptation

   The result of this step is a set of StaticFilterCollection instances (see 7.3.17).

2) Inspect each StaticFilterCollection instance resulting from step 1) by checking the value of the CollectionName property. If the name of the static filter collection as indicated by that value matches the desired name, this use case is complete; otherwise, continue inspecting the next IndicationFilter instance returned by step 1).

8.18.3 Postconditions

Unless errors occurred, the client knows the named filter collection by means of the representing StaticFilterCollection instance (including its object path).

8.19 CreateSubscription: Create a subscription

8.19.1 Preconditions

The client knows all of the following:

- the object path to the IndicationService instance representing the indication service within the implementation (see 7.3.2)
- an object path to an IndicationFilter instance representing an indication filter covering the desired indication or set of indications

  For example, see the FindIndicationFilter (8.12) or CreateIndicationFilter (8.14) use cases about how to obtain that object path.
- Alternatively, an object path to a StaticFilterCollection instance representing a filter collection covering the desired indication or set of indications. For example, see the ObtainNamedCollection use case (8.18) about how to obtain the object path to a StaticFilterCollection instance representing a global filter collection or a profile-specific filter collection.
- an object path to a ListenerDestination instance representing a listener destination that represents the desired listener within the implementation. For example, see the SelectListenerDestination use case (8.7) about how to obtain that object path.

8.19.2 Flow of activities

1) Prepare a local instance of the CIM_IndicationSubscription class (or the CIM_FilterCollectionSubscription for a subscription to a filter collection) that complies with the requirements of the FilterSubscription adaptation (see 7.3.26) or the CollectionSubscription adaptation (see 7.3.27), inserting property values as follows:
   – Filter: input object path to the indication filter (or to the filter collection)
   – Handler: input object path to the listener destination

   The values of other properties should be specified in conformance with the capabilities of the implementation as exposed by instances of the IndicationService adaptation and the IndicationServiceCapabilities adaptation; see the DetermineIndicationServiceCapabilities use case (8.4) to obtain knowledge about these capabilities.

   Values not described through these adaptations may or may not be respected by the implementation; in this case it is implementation dependent whether in step 2) the implementation imposes a respective default behavior, or whether it fails in creating the new subscription.
2) Define the new subscription to the implementation by invoking the CreateInstance( ) operation, providing the CIM_IndicationSubscription (or CIM_FilterCollectionSubscription) instance prepared in step 1) as the actual value of the NewInstance parameter.

   If successful, the operation returns the object path of the DynamicIndicationFilter instance representing the newly created subscription.

   If not successful, the operation returns a CIM status code providing details about the failure (see 7.3.26.3.2 or 7.3.27.3.2).

8.19.3 Postconditions

Unless errors occurred, the client knows the object path of an AbstractSubscription instance representing the newly created subscription; otherwise, the client knows details about why it was not possible to create the subscription.

8.20 CheckSubscriptions: Determine whether subscriptions exist for a given indication and listener

8.20.1 Preconditions

The client knows all of the following:

- the object path to the IndicationService instance representing the indication service within the implementation (see 8.2)
- the URI exposed by the desired listener

8.20.2 Flow of activities

1) Execute the ListListenerDestinations use case (see 8.6). The result is a set of ListenerDestination instances (including their object paths) representing all the listener destinations within the implementation.

2) From the result of step 1), drop all ListenerDestination instances not referencing the desired listener. The result is a set of ListenerDestination instances (including their object paths) representing all the listener destinations referencing the desired listener.

3) For each ListenerDestination instance resulting from step 2), find all IndicationFilter instances (see 7.3.11) associated with the ListenerDestination instance (see 7.3.23) through a FilterSubscription instance (see 7.3.26).The result of this step is a set of IndicationFilter instances representing indication filters to which the desired listener is subscribed.

4) Inspect each IndicationFilter instance resulting from step 3) by checking the values of the QueryLanguage and the Query properties. Interpret the query statement expressed by the value of the Query property and check whether the input indication is covered. If the input indication is covered, add the identification of the represented listener destination to a filter result list, and continue inspecting the next IndicationFilter instance returned by step 3).

5) For each ListenerDestination instance resulting from step 2), find all StaticFilterCollection instances (see 7.3.17) associated through a CollectionSubscription instance (see 7.3.27). The result of this step is a set of StaticFilterCollection instances representing static filter collections to which the desired listener is subscribed.

6) For each StaticFilterCollection instance resulting from step 5), apply the CheckCollectionCoverage use case (see 8.17).

   If the input indication is covered, add the identification of the represented static filter collection to a collection result list, and continue inspecting the next StaticFilterCollection instance returned by step 5).

8.20.3 Postconditions

Unless errors occurred, the client knows (the identifications of) all listener destinations and filter collections to which the desired listener is subscribed.

8.21 DeleteSubscription: Delete a subscription

8.21.1 Preconditions

The client knows all of the following:

- the object path to the AbstractSubscription instance (see 7.3.25) representing a subscription within the implementation

8.21.2 Flow of activities

1) Invoke the DeleteInstance( ) operation on the AbstractSubscription instance, effecting the deletion of the represented subscription.

NOTE If the subscription referenced a dynamic indication filter, and no other subscriptions reference it, and the client does not plan to create a new subscription for this filter, the client can delete the dynamic indication filter using the DeleteFilter use case (see 8.16); likewise, unless referenced by other subscriptions, the client can delete the listener destination that was referenced by the deleted subscription, using the DeleteListenerDestination use case (see 8.11).

8.21.3 Postconditions

Unless errors occurred, the subscription is deleted and no longer represented by any AbstractSubscription instance.

8.22 FindAlertingSystem: Find the system containing a component causing an alert indication

8.22.1 Preconditions

The client knows all of the following:

- an AlertIndication instance representing an alert indication that references the alerting managed element

8.22.2 Flow of activities

1) Obtain the CIM element referenced by the value of the AlertingManagedElement in the input AlertIndication instance.
2) Determine the profile with which the CIM element is conformant and where the central class adaption adapts the CIM_System class.

   NOTE This step implies client knowledge about profiles defining adaptations of the class of the CIM element obtained in step 1). More than one profile could impact the CIM element, but the scoping CIM_System instance should be the same in all cases.

3) Use the scoping algorithm defined by the profile determined in step 2) to find the related instance of the scoping class adaptation of that profile.

8.22.3 Postconditions

Unless errors occurred, the client knows the CIM_System instance representing the system containing a component causing the generation of the input alert indication.

8.23 DetermineIndicationGate: Determine the indication gate of an indication

8.23.1 Preconditions

The client knows all of the following:

- an AlertIndication instance representing an alert indication that references the alerting managed element

In addition, subscriptions for the listener that received the input alert indication should have been established such that within the set of subscribed to indication gates within a particular implementation each is uniquely identified with a name as exposed by the value of the Name property in representing

IndicationFilter instances (see 7.3.11), or as exposed by the value of the CollectionName property in representing StaticFilterCollection instances (see 7.3.17).

NOTE This policy ensures that indication gate names are unique with respect to one implementation; implementations are unable to (and not required to) maintain that uniqueness, but clients can ensure it through carefully applying the subscription policy stated above for each listener that a client controls.

8.23.2 Flow of activities

1) Extract the value of the IndicationFilterName from the input AlertIndication instance as the name of the sought-after indication gate.

   If the input alert indication originates from an implementation that is known to the client by reference to its representing IndicationFilter instance, skip to step 8); otherwise, continue with step 2).

2) Inspect the value of the AlertingManagedElement property of the input AlertIndication instance.

   If that value is Null, then the indication gate cannot be determined, and this use case is complete without success; this is also the case of the value is a URI that does not reference a CIM instance that represents the alerting managed element. In subsequent steps it is assumed that the value is a URI that references a CIM instance that represents the alerting managed element.

3) Determine the ProfileRegistration instance that is providing the CIM instance referenced by the URI found in step 2), using one of the algorithms described in DSP1033 for that purpose.

4) Apply the LocateProfileIndicationService use case (see 8.3) in order to determine the IndicationService instance (see 7.3.2) that represents the indication service from which the input alert indication originated.

5) Find all IndicationFilter instances (see 7.3.11) associated with the IndicationFilter instance (see 7.3.23) found in step 4) through an IndicationServiceOfIndicationFilter instance (see 7.3.14), for example by executing the GetAssociatedInstancesWithPath( ) operation.

6) For each IndicationFilter instance obtained in step 5), determine if the value of the Name property matches the name of the sought-after indication gate determined in step 1).

   If it matches, and the subscription policy mentioned in the preconditions was maintained, then the indication filter represented by the IndicationFilter instance is the sought-after indication gate.

   If the name matches, and the subscription policy was not maintained, then all IndicationFilter instances determined in step 5) need to be checked with step 6) in order to ensure that the name as exposed by the value of the Name property is not used more than once. If this is the case, the sought-after indication gate cannot be exactly determined; however, at least it can be limited to the set of indication filters using the name as determined in step 1).

   If a name does match, continue with step 8).

   If the name does not match, the next instance from the set determined in step 5) needs to be checked with step 6); if no additional instances remain, continue with step 7).

7) Repeat steps 5) and 6) for filter collections, searching for StaticFilterCollection instances (see 7.3.17) associated through an IndicationServiceOfFilterCollection instance (see 7.3.18) in step 5), and checking the value of the CollectionName property in step 6).

8) If an indication filter was determined as the sought-after indication gate in steps 1), 6), or 7), the client can check the query statement exposed by the value of the Query property in the representing IndicationFilter instance (or — in case the alert indication was received through a filter collection — in at least one of the contained IndicationFilter instances), and verify that the

input alert indication is indeed within the coverage of the identified indication filter or filter collection.

8.23.3 Postconditions

Unless errors occurred, the client knows the indication gate emitting the input alert indication by means of its representing IndicationFilter or StaticFilterCollection instance.

8.24 SubscribeForProfileIndications: Subscribe for all of the indications defined in a referencing profile

8.24.1 Preconditions

The client knows the following:

- the registered name of the referencing profile
- the object path to the IndicationService instance representing the indication service within the implementation (see 7.3.2)
- the object path to the ListenerDestination instance (see 7.3.23) representing the desired listener destination

8.24.2 Flow of activities

1) Construct the name for the profile-specific filter collection for alert indications, applying the pattern defined in 7.3.21.2.2.
2) Execute the ObtainNamedCollection use case (see 8.18), providing the name constructed in step 1) as input; the result is either Null or the object path referencing the ProfileSpecificAlertIndicationFilterCollection instance (see 7.3.21) representing the profile-specific filter collection for alert indications of the referencing profile.
3) If an object path was returned on step 2), execute the CreateSubscription use case (see 8.19), providing that object path and the input object path to the ListenerDestination instance as input.
4) Perform steps 1), 2) and 3) analogously for lifecycle indications.

8.24.3 Postconditions

Unless errors occurred, the desired listener destination is subscribed for all alert indications and all lifecycle indications defined by the referencing profile.

ANNEX A
(informative)

Profiles defining indications

Referencing profiles define indications and related requirements in the following ways:

- Reference this profile as a mandatory or conditional profile
- Define lifecycle indications and/or alert indications by defining adaptations based on the LifecycleIndication adaptation (see 7.3.32) and/or the AlertIndication adaptation (see 7.3.31). This requires but is not limited to defining the requirement level, the reported event, and the query statement; however, the latter two may be implied by the respective base adaptation.
- Optionally, define indication filters by defining adaptations based on the StaticIndicationFilter adaptation (see 7.3.11). The definition of indication-specific indication filters covering each lifecycle indication and each alert indication defined in a referencing profile is implied by this profile through the IndicationSpecificIndicationFilter adaptation (see 7.3.15), but may be refined by referencing profiles.
- Optionally, define filter collections by defining adaptations based on the StaticFilterCollection adaptation (see 7.3.17). The definition of profile-specific filter collections covering all lifecycle indications and/or alert indications defined in a referencing profile is implied by this profile through the ProfileSpecificFilterCollection adaptation (see 7.3.21), but may be refined by referencing profiles.

ANNEX B (informative)

Change Log

Version	Date	Description
1.0.0a	2007-06-04	Preliminary Standard
1.0.0	2008-12-05	Final Standard
1.0.1	2009-09-07	Released as DMTF Standard, with the following changes: • Updated profile conventions for operations and their usage • Fixed incorrect CIM Schema version (from 2.16 to 2.22)
1.1.0a	2009-12-02	Released as Work in Progress, with the following changes: • Increased CIM Schema version to 2.23(exp). • Added support for reliable indications (delivery retry, detection of lost indications, reconstruction of original order): – Description of reliable indications concept in 7.10 (Indication Delivery). – Clarifications in description of CIM_ListenerDestination.PersistenceType. • Refined the format for CIM_FilterCollection.CollectionName in 7.6. • Refined the format for CIM_IndicationFilter.Name in 7.4. • Cleaned up terminology clause by removing most terms that are defined in DSP0004, DSP0200 or DSP1001. • Added "Document conventions" clause and consolidated existing text into that. • Updated profile conventions for operations to match DSP1001 1.0.1. • Fixed incorrect pattern value "WBEMURI" for CIM_AlertIndication.AlertingElementFormat.
1.1.0	2010-05-20	Released as DMTF Standard, with the following changes: • Clarified and added some terms in clause 3. • Clarified that there is only one indication service in a WBEM server, but added a recommendation for clients to expect more than one in the future. • Fixed incorrect verbiage of sending indications to clients, to sending indications to listeners. • Changed ambiguous "conditional/optional" requirement to "conditional or optional" in all cases but one. • Clarified that listeners that intend to re-establish the original order of indications need to buffer indications that do not have the predicted sequence number until decision about loss can be made. • Lowered the requirement not to interpret sequence numbers in case of not implementing them, to a permission to ignore them. • Fixed inconsistencies in several diagrams.

Version	Date	Description
1.2.0a	2010-06-16	Released as Work in Progress, with the following changes: • Increased CIM Schema version to 2.25 • Converted to PUG 1.1 "Condensed Format": – The semantics of the definitions from the previous version was maintained, except the semantical changes detailed – Introduced separation between managed environment and CIM model – Defined many new terms precisely capturing concepts only vaguely defined in the previous version – Introduced features – Introduced adaptations, integrating the content of the Methods and the "CIM elements" clauses defined in the previous version into the "Implementation" clause of this version – Modified existing use cases using adaptations, and introduced new use cases • Introduced the following new concepts: – Global filters – Global filter collections – Profile-specific filters – Profile-specific filter collections • Deprecated the use of the CIM_ConcreteDependency association for modeling the relationship between filter collections and profile representations of referencing profiles (CIM_RegisteredProfile) • Changed the requirement level for the ElementConformsToProfile association adaptation to mandatory • Fixed incorrect property name: CIM_ListenerDestination.Protocol was incorrectly named ProtocolType. • Many clarifications of existing concepts, such as the following: – Established indication emitters as the super type of indication filters and filter collections – Clarified that the purpose of indication emitters is filtering indications, and is not the representation of indication implementations – Restructured the specification of reliable indications using a feature and adaptations – Consistently use the terms "sequence identifier" and "sequence identifier lifetime", as established by the CIM schema (quit using the term "sequence identifier value") – Suppression of repeated indication delivery

Version	Date	Description
1.2.0b	2010-09-15	Released as Work in Progress, with the following changes: • Included cPubs major scrub • Renamed indication emitter -> indication gate • Renamed profile-specific filter -> indication-specific filter • Removed specializations of these (introduced in the 1.2.0a version) • Deprecate requiring a CIM_Error instance in case of IndicationFilter.CreateInstance( ) error • Recommending the Interop namespace as the only namespace for IndicationFilter instances, StaticFilterCollection instances, ListenerDestination instances and AbstractSubscription instances • Many clarifications of existing concepts and addition of new concepts, such as the following: – Require that all kinds of filters have one or more related namespaces, either those identified by the value of the SourceNamespaces[] property, or – if the value is Null - the namespace where the filter representation resides; version 1.1 left that open for dynamic filters. – Prohibit empty array as possible SourceNamespaces[] value, as that would be semantically useless because no indication would be allowed to pass in this case. – Requiring that indications have an origin namespace – Requiring that the origin namespace is taken into consideration during indication filtering, i.e., is subject to filtering. This is done by extending the concept of the filter coverage such that both query statement and the namespace list span the filter coverage – Correcting the prohibition of providing any key properties when creating dynamic indication filters by exempting the Name property, along with a recommended naming convention

Version	Date	Description
1.2.0c	2011-04-05	Released as a DMTF Draft Standard, with the following changes: • Adjusted to the DMTF Draft Standard version of DSP1001 1.1 – Moved base elements from the table of class adaptations to the individual element requirements tables of the adaptations – Adopted the format for error reporting requirements – Require Key properties to be listed when used the first time in a chain of adaptations – Introduction of the implementation type – Restructured the error reporting requirement tables • Minor corrections resulting from reviews of version 1.2.0b • Adjust the use of the Profile Registration profile to DSP1033 1.0 • Specify operation requirements in terms of DSP0223 (as required by DSP1001, after Architecture workgroup decision) • Rephrased the policies for the avoidance of repeated indication delivery, synchronizing it with the phraseology used in the schema description of the CIM_AbstractIndicationSubscription class • Resolved various comments from 2nd SNIA review • Changed the requirement level of IndicationServiceOfIndicationFilter, IndicationServiceOfFilterCollection, CollectionSubscription and ProfileOfFilterCollection from conditional to mandatory because the condition was always true (the GlobalFilter and GlobalFilterCollection adaptations are mandatory derived adaptation of the IndicationFilter and StaticFilterCollection adaptations) • Reinforced the version 1.1 requirement that key properties on the creation of DynamicIndicationFilter instances are to be ignored, and should not be provided by clients • Extended the AlertIndication adaptation to allow for referencing more than one alert message • Extended the IndicationSpecificIndicationFilter adaptation to provide for multiple instances for the coverage of multi-message AlertIndication adaptations
1.2.0	2011-06-30	Released as a DMTF Standard, with the following changes: • Confirmed the CIM schema definition of CIM_Indication wrt. that a sequence identifier needs to be maintained on a per listener destination basis (and not on a per listener basis)

3435