Indication Standard
DMTF

Roger Kumpf
Hewlett-Packard
Module Content

CIM Indication Overview

- Terminology
- Indication Hierarchy
- Indication Subscription Schema
Terminology

- **Event.** The occurrence of a phenomenon of interest. For example, an Event can denote the occurrence of a disk write error, a failed authentication attempt, or even a mouse click.

- **Indication.** The representation of the occurrence of an Event.
Terminology

- **Indication Subscription.** The process of registering to receive Indications.
- **Indication Delivery.** The process of transporting one or more indications to a designated destination. The intended protocol and destination are specified as part of the subscription definition.
- **Subscription Client.** A CIM Client application that creates subscriptions.
Module Content

CIM Indication Overview
- Terminology
- **Indication Hierarchy**
- Indication Subscription Schema
The **CIM Indication Hierarchy** is used to describe the Events that can be represented as CIM Indications. An abstract class, **CIM_Indication**, serves as the base class for all Indication classes.

### What’s Available

- **CIM_Indication**
  - Abstract
  - IndicationIdentifier: string
  - CorrelatedIndications[]): string
  - IndicationTime: datetime

- **CIM_ClassIndication**
  - Abstract
  - ClassDefinition: string

- **CIM_InstIndication**
  - Abstract
  - SourceInstance: string

- **CIM_ProcessIndication**
  - Abstract
CIM_ProcessIndication
Model Extensions

What’s Available from ACME

ACME_UPSAlertIndication
- UPS tty Path: string
- UPS Status Code: uint16
- UPS Firmware Version: string
- Battery Voltage: uint16
- Remaining Runtime Reserve: unit16
- Unit Type: string
- Rated Input Voltage: uint16
- Nominal Frequency: unit16
- Rated VA: unit16
- Output Voltage: uint16
- Phase: uint16

The CIM Indication hierarchy is extensible. New subclasses can be added to capture vendor-specific properties and Event types.
Module Content

CIM Indication Overview

- Terminology
- Indication Hierarchy
- Indication Subscriptions
CIM_IndicationSubscription

**CIM_IndicationFilter**
What to Send

<table>
<thead>
<tr>
<th>IndicationFilter</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemCreationClassName: string {Key}</td>
</tr>
<tr>
<td>SystemName: string {Key}</td>
</tr>
<tr>
<td>CreationClassName: string {Key}</td>
</tr>
<tr>
<td>Name: string {Key}</td>
</tr>
<tr>
<td>SourceNamespace: string</td>
</tr>
<tr>
<td>Query: string {Required}</td>
</tr>
<tr>
<td>QueryLanguage: string {Required}</td>
</tr>
</tbody>
</table>

**CIM_IndicationHandler**
How & Where to Send

<table>
<thead>
<tr>
<th>IndicationHandler</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemCreationClassName: string {Key}</td>
</tr>
<tr>
<td>SystemName: string {Key}</td>
</tr>
<tr>
<td>CreationClassName: string {Key}</td>
</tr>
<tr>
<td>Name: string {Key}</td>
</tr>
<tr>
<td>Owner: string</td>
</tr>
<tr>
<td>PersistenceType: uint16 {Enum}</td>
</tr>
<tr>
<td>OtherPersistenceType: string</td>
</tr>
</tbody>
</table>

A CIM_IndicationSubscription instance defines an **association** between a **CIM_IndicationFilter** instance and a **CIM_IndicationHandler** instance.
CIM_IndicationFilter

- **SourceNamespace** defines the namespace for the Indication stream. In particular, all Indications in the generated Indication stream *must* be applicable to, and consistent with, the designated namespace.

- **Query** defines the Indication class, filter condition and property list of the Indication stream.

- **QueryLanguage** defines the Query Language used to define the Query.

---

### CIM_IndicationFilter

**What to Send**

<table>
<thead>
<tr>
<th>IndicationFilter</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemCreationClassName: string (Key)</td>
</tr>
<tr>
<td>SystemName: string (Key)</td>
</tr>
<tr>
<td>CreationClassName: string (Key)</td>
</tr>
<tr>
<td>Name: string (Key)</td>
</tr>
<tr>
<td>SourceNamespace: string</td>
</tr>
<tr>
<td>Query: string (Required)</td>
</tr>
<tr>
<td>QueryLanguage: string (Required)</td>
</tr>
</tbody>
</table>
CIM_IndicationHandler

- **PersistenceType** and **OtherPersistenceType** – characterize the expected “lifetime” of the Indication consumer.

- **Owner** – this property is ill defined and will be proposed for “deprecation” in version 2.8 of the Event Schema.
A CIM Client application activates a subscription by creating an instance of CIM_IndicationSubscription.
The **Repeat Notification** properties define the desired frequency for notifying a consumer of the occurrence of an Event that satisfies the subscription.

The **Subscription State** properties allow a Client to monitor and control the state of the subscription.

The **Subscription Failure Handling** properties define the desired behavior when a fatal error occurs processing the subscription.

The **Subscription Duration** properties define the desired length of the subscription.

Refer to DMTF Event White Paper for further details.
**Subscription Example:** Create a subscription to send any critical UPS alert indications for device “tty0p1” on system “server001.acme.com” using CIM-XML to URL “server006.acme.com/cimom/EB”.

**Example Diagram:**

- **server006.acme.com**
  - Indication Receipt & Consumption
  - CIM Listener

- **server001.acme.com**
  - Indication Subscription, Generation & Delivery
  - Subscription Client

- **UPS Alert Export Indication (CIM-XML)**
  - Critical Problem with UPS Device Detected

- **CIMOM**
  - Indication Provider
  - UPS dev/tty0p1
## IndicationFilter Example

Filter used by ACME to generate critical UPS alerts for device ‘tty0p1’. This indication will return all properties associated with the class ACME_UPSAalertIndication.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CIM_IndicationFilter</strong></td>
<td>What to Send</td>
</tr>
<tr>
<td>Caption</td>
<td>Generate critical UPS alerts for /dev/tty0p1</td>
</tr>
<tr>
<td>Description</td>
<td>Filter used by ACME to generate critical UPS alerts for device ‘tty0p1’. This indication will return all properties associated with the class ACME_UPSAalertIndication.</td>
</tr>
<tr>
<td>Key</td>
<td>SystemCreationClassName</td>
</tr>
<tr>
<td>Key</td>
<td>SystemName</td>
</tr>
<tr>
<td>Key</td>
<td>CreationClassName</td>
</tr>
<tr>
<td>Key</td>
<td>Name</td>
</tr>
<tr>
<td>SourceNamespace</td>
<td>root/cimv2</td>
</tr>
<tr>
<td>Required</td>
<td>Query</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Required</td>
<td>QueryLanguage</td>
</tr>
</tbody>
</table>
# IndicationHandler Example

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caption</td>
<td>Acme Hardware Event Browser</td>
</tr>
<tr>
<td>Description</td>
<td>Location of Acme Event Browser responsible for monitoring hardware events for this system.</td>
</tr>
<tr>
<td>Key</td>
<td>SystemCreationClassName</td>
</tr>
<tr>
<td></td>
<td>CIM_UnitaryComputerSystem</td>
</tr>
<tr>
<td>Key</td>
<td>SystemName</td>
</tr>
<tr>
<td></td>
<td>server001.acme.com</td>
</tr>
<tr>
<td>Key</td>
<td>CreationClassName</td>
</tr>
<tr>
<td></td>
<td>CIM_IndicationHandlerCIMXML</td>
</tr>
<tr>
<td>Key</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>ACMESubscription12345</td>
</tr>
<tr>
<td>Owner</td>
<td>EventAdmin</td>
</tr>
<tr>
<td>Required</td>
<td>Destination</td>
</tr>
<tr>
<td></td>
<td>server006.acme.com/cimom/EB</td>
</tr>
</tbody>
</table>
IndicationSubscription Example

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Value <code>&lt;instanceName)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>Filter</td>
</tr>
<tr>
<td>Class Name = CIM_IndicationFilter</td>
<td></td>
</tr>
<tr>
<td>Key Binding = SystemCreateClassName = “CIM_UnitaryComputerSystem”, SystemName = “server001.acme.com”, CreationClassName = “CIM_IndicationFilter”, Name = “ACMESubscription12345”</td>
<td></td>
</tr>
<tr>
<td>key</td>
<td>Handler</td>
</tr>
<tr>
<td>Class Name = CIM_IndicationHandlerCIMXML</td>
<td></td>
</tr>
<tr>
<td>Key Binding = SystemCreateClassName = “CIM_UnitaryComputerSystem”, SystemName = “server001.acme.com”, CreationClassName = “CIM_IndicationHandlerCIMXML”, Name = “ACMESubscription12345”</td>
<td></td>
</tr>
</tbody>
</table>
Indication Architecture
OpenPegasus

Roger Kumpf
Hewlett-Packard
Module Content

HP WBEM Services Indication Architecture

- **Indication Components**
  - Indication Generation
  - Indication Subscription
  - Indication Delivery
  - Indication Consumption
Indication Components

Indication Receipt & Consumption

Management System

1A Indication Provider Registration

CIM Object Manager

1B Indication Consumer Registration

CIM Listener

2 Subscription

Indication Subscription, Generation & Delivery

Managed System

3 Indication Generation & Processing

CIM-XML Server

Protocol Adapter

CIM-XML

Indication Handler

Indication Provider

Resources

Indication Receipt & Consumption

Indication Delivery

CIM Indication Export Message

5 Indication Receipt

Indication Consumption

6

Client Subscriber
Let's first discuss this side.
Indication Architecture

- **CIM Server**
  - CIM-XML Server Protocol Adapter
  - CIM Operation Processor
  - Indication Service

- **Subscription**
  - Provider Manager
  - Indication Provider
  - Resource

- **Indication Export Message**
  - Indication Handler Manager
  - CIM-XML Indication Handler

- **Indication Delivery**
  - Indication Handler Manager
  - Indication Handler
Module Content

HP WBEM Services Indication Architecture

- Indication Components
  - Indication Generation
  - Indication Subscription
  - Indication Delivery
  - Indication Consumption
Indication Generation

Simply defined, an **Event** is the occurrence of a phenomenon of interest. For example, an Event can denote the occurrence of a disk write error, a failed authentication attempt, or even a mouse click.

An **Indication** is the representation of the occurrence of an Event.

A **CIM Indication Provider** translates the detection of an Event into a CIM Indication and sends the Indication to the CIM Object Manager for further processing and delivery.
What’s Available

**CIM_Indication**
- Abstract
  - IndicationIdentifier: string
  - CorrelatedIndications[]: string
  - IndicationTime: datetime

**CIM_ClassIndication**
- Abstract
  - ClassDefinition: string

**CIM_InstIndication**
- Abstract
  - SourceInstance: string

**CIM_ProcessIndication**
- Abstract

The **CIM Indication Hierarchy** is used to describe the Events that can be represented as CIM Indications. An abstract class, **CIM_Indication**, serves as the base class for all Indication classes.
A **CIM Indication Provider** registers with the CIM Server to generate events for one or more Indication classes.

**CIM Indication Hierarchy**

- **CIM_ProcessIndication**
  - **Abstract**
    - **CIM_AlertIndication**
      - **ACME_AlertIndication**

**Indication Generation**

**ACME_AlertIndication Provider**

1. **ACME_Resource**
2. **Acme_AlertIndication Provider**
3. **CIM Object Manager**
4. **CIM-XML Server Protocol Adapter**

**Provider Registration**

- SystemIPAddress: string
- SystemModelNumber: string
- OSVersion: string
- SystemSerialNumber: string
- SystemSoftwareID:
- MonitorID: string
- MonitorProductVersion: string
- MonitorEventID: uint1
Indication Provider

<table>
<thead>
<tr>
<th>CIM Export</th>
<th>Implementation Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExportIndication</td>
<td>Indication Providers</td>
</tr>
</tbody>
</table>

```c
handle->deliver();
```
CIM Provider Logic

Indication Service

Commands to Control Behavior

CIM Indication Provider

Event Occurred

Instrumentation to Monitor Resource

Indication Delivery

Indication Service
Indication Provider API

createSubscription
void createSubscription(
    const OperationContext & context,
    const CIMObjectPath & subscriptionName,
    const Array<CIMObjectPath> & classNames,
    const CIMPropertyList & propertyList,
    const Uint16 repeatNotificationPolicy)

modifySubscription
const OperationContext & context,
    const CIMObjectPath & subscriptionName,
    const Array<CIMObjectPath> & classNames,
    const CIMPropertyList & propertyList,
    const Uint16 repeatNotificationPolicy) = 0;

deleteSubscription
void deleteSubscription(
    const OperationContext & context,
    const CIMObjectPath & subscriptionName,
    const Array<CIMObjectPath> & classNames)
CIM Provider Logic

Indication Service

Process createInstance of CIM_IndicationSubscription

createSubscription

void createSubscription(
    const OperationContext & context,
    const CIMObjectPath & subscriptionName,
    const Array<CIMObjectPath> & classNames,
    const CIMPropertyList & propertyList,
    const Uint16 repeatNotificationPolicy)

modifySubscription

const OperationContext & context,
const CIMObjectPath & subscriptionName,
const Array<CIMObjectPath> & classNames,
const CIMPropertyList & propertyList,
const Uint16 repeatNotificationPolicy) = 0;

deleteSubscription

void deleteSubscription(
    const OperationContext & context,
    const CIMObjectPath & subscriptionName,
    const Array<CIMObjectPath> & classNames)

CIM Indication Provider

Process modifyInstance of CIM_IndicationSubscription

Process deleteInstance of CIM_IndicationSubscription

OpenPegasus C++ Provider API

OpenPegasus C++ Provider API
Indication Provider API

CIM Indication Provider

**enableIndications**

```cpp
void enableIndications(IndicationResponseHandler & handler)
```

**disableIndications**

```cpp
void disableIndications(void)
```
CIM Provider Logic

Begin Processing Event

Validate Input

Build Indication

Deliver Indication

End Processing Event

CIM Indication Provider

Indication Delivery

Event Occurred

Instrumentation to Monitor Resource

Indication Service
Indication Provider Example

RT_Indication

Select an item to view its description.

See also:
- My Documents
- My Network Places
- My Computer
void _generateIndication (  
    IndicationResponseHandler * handler,  
    const CIMName methodName)  
{  
    if (_enabled)  
    {  
        CIMInstance indicationInstance (CIMName("RT_TestIndication"));  
        CIMObjectPath path;  
        path.setNameSpace("root/SampleProvider");  
        path.setClass Name("RT_TestIndication");  
        indicationInstance.setPath(path);  
        char buffer[32];  
        sprintf(buffer, "%d", _nextUID++);  
        indicationInstance.addProperty  
            (CIMProperty ("IndicationIdentifier", String(buffer)));
        CIMDateTime currentTime = CIMDateTime::getCurrentDateTime ();  
        indicationInstance.addProperty  
            (CIMProperty ("IndicationTime", currentTime));  
        Array<String> correlatedIndications;  
        indicationInstance.addProperty  
            (CIMProperty ("CorrelatedIndications", correlatedIndications));  
        indicationInstance.addProperty  
            (CIMProperty ("MethodName", CIMValue(methodName.getString())));  
        CIMIndication cimIndication (indicationInstance);  
        handler->deliver (cimIndication);  
    }  
}  
handler->deliver();
In the RT Indication Example we've implemented an extrinsic method SendTestIndication to allow us to "trigger" the generation of an Indication.
void RT_IndicationProvider::invokeMethod(
    const OperationContext & context,
    const CIMObjectPath & objectReference,
    const CIMName & methodName,
    const Array<CIMParamValue> & inParameters,
    MethodResultResponseHandler & handler)
{
    Boolean sendIndication = false;
    handler.processing();

    if (objectReference.getClassName().equal("RT_TestIndication") &&
        _enabled)
    {
        if (methodName.equal("SendTestIndication"))
        {
            sendIndication = true;
            handler.deliver( CIMValue( 0 ) );
        }
    }
    else
    {
        handler.deliver( CIMValue( 1 ) );
        PEGASUS_STD(cout) << "Provider is not enabled." << PEGASUS_STDendl;
    }

    handler.complete();

    if (sendIndication)
        _generateIndication(_handler,"generateIndication");
}
Indication Provider Registration

Description of Physical Package (e.g., Shared Library)

PG_ProviderModule

Description of Provider within Physical Package

PG_Provider

Description of Provider Capabilities

PG_ProviderCapabilities

PG_ProviderModule

ProviderModuleName: string;
ProviderName: string;
CapabilityID: string;
ClassName: string;
Namespaces: string;
ProviderType[]: uint16;
SupportedProperties[]: string;
SupportedMethods[]: string;

PG_Provider

Vendor: string;
Version: string;
InterfaceType: string;
InterfaceVersion: string;
Location: string;
OperationalStatus: uint16(enum);
OtherStatusDescription: string;
uint32 start();
uint32 stop();

PG_ProviderCapabilities

ProviderModuleName: string;
ProviderName: string;
CapabilityID: string;
ClassName: string;
Namespaces: string;
ProviderType[]: uint16;
SupportedProperties[]: string;
SupportedMethods[]: string;

CIM_ManagedElement

Description: string;
Caption: string;
ElementName: string;

CIM_ManagedSystemElement

InstallDate: datetime;
Name: string;
Status: string;
OperationalStatus: uint16;
StatusDescriptions[]: string;

CIM_LogicalElement

PG_ProviderCapabilitiesElements

1
### CIM Provider Types

<table>
<thead>
<tr>
<th>CIM Operation</th>
<th>Implementation Owner¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetClass, CreateClass, ModifyClass, DeleteClass, GetQualifier, SetQualifier, DeleteQualifier, EnumerateQualifier, EnumerateClasses, EnumerateClassNames,</td>
<td>CIMOM</td>
</tr>
<tr>
<td>GetInstance, EnumerateInstances, EnumerateInstanceNames, GetProperty, SetProperty, CreateInstance, ModifyInstance, DeleteInstance</td>
<td>Instance Provider</td>
</tr>
<tr>
<td>InvokeMethod</td>
<td>Method Provider</td>
</tr>
<tr>
<td>References, ReferenceNames, Associators, AssociatorNames</td>
<td>Association Providers</td>
</tr>
<tr>
<td>ExportIndication</td>
<td>Indication Providers</td>
</tr>
</tbody>
</table>

¹ Details in this column vary by implementation.

A **CIM Indication Provider** translates the occurrence of an Event into a CIM Indication and sends the Indication to the CIM Object Manager for further processing and delivery.
Provider Capabilities

**Acme_AlertIndication Provider**

<table>
<thead>
<tr>
<th>Set of Operations</th>
<th>Indication Provider Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>ACME_AlertIndication</td>
</tr>
<tr>
<td>Namespace</td>
<td>root/cimv2</td>
</tr>
</tbody>
</table>

**PG_ProviderCapabilities**

```c
instance of PG_ProviderCapabilities
{
    ProviderModuleName = "ACME_IndicationModule";
    ProviderName = "Acme_AlertIndicationProvider";
    CapabilityID = "1";
    ClassName = "ACME_AlertIndication";
    Namespaces = {"root/cimv2"};
    ProviderType = { 4 }; // Indication
    SupportedProperties = NULL; // All properties
    SupportedMethods = NULL; // All methods
}
```
Module Content

HP WBEM Services Indication Architecture

- Indication Components
  - Indication Generation
  - Indication Subscription
  - Indication Delivery
  - Indication Consumption
Indication Subscription

• An "Indication Subscriber" is a **CIM Client** that issues CIM Operation requests to create instances of the CIM_IndicationSubscription class.

• A **CIM Server** receives and processes CIM Operation requests and issues CIM Operation responses.

• The **CIM Indication Service** is a component of the CIM Object Manager. It is responsible for the processing of CIM Operations on the classes in the CIM Subscription Schema.
Indication Subscriptions

CIM Subscription Schema

A CIM Client application activates a subscription by creating an instance of CIM_IndicationSubscription.
**IndicationFilter**
- SystemCreationClassName: string (Key)
- SystemName: string (Key)
- CreationClassName: string (Key)
- Name: string (Key)
- SourceNamespace: string
- Query: string (Required)
- QueryLanguage: string (Required)

**IndicationHandler**
- SystemCreationClassName: string (Key)
- SystemName: string (Key)
- CreationClassName: string (Key)
- Name: string (Key)
- Owner: string
- PersistanceType: uint16 (Enum)
- OtherPersistanceType: string

**IndicationSubscription**

```xml
<?xml version="1.0" encoding="utf-8" ?>
< CIM CIMVERSION="2.0" DTDVERSION="2.0">  
  < MESSAGE ID="020001" PROTOCOLVERSION="1.0">  
    < SIMPLEREQ > 
      - <IMETHODCALL NAME="CreateInstance"> 
        < LOCALNAMESPACEPATH > 
          <NAMESPACE NAME="root" /> 
          <NAMESPACE NAME="PG_InterOp" /> 
        </LOCALNAMESPACEPATH> 
        - <IPARAMVALUE NAME="NewInstance"> 
          - <INSTANCE CLASSNAME="#_IndicationHandlerCIMXML"> 
            - <PROPERTY NAME="#_SystemCreationClassName" TYPE="string"> 
              <VALUE> </VALUE> 
            </PROPERTY> 
            - <PROPERTY NAME="#_SystemName" TYPE="string"> 
              <VALUE> </VALUE> 
            </PROPERTY> 
            - <PROPERTY NAME="#_CreationClassName" TYPE="string"> 
              <VALUE> </VALUE> 
            </PROPERTY> 
            - <PROPERTY NAME="#_Name" TYPE="string"> 
              <VALUE> </VALUE> 
            </PROPERTY> 
            - <PROPERTY NAME="#_Owner" TYPE="string"> 
              <VALUE> </VALUE> 
            </PROPERTY> 
            - <PROPERTY NAME="#_PersistanceType" TYPE="uint16"> 
              <VALUE> </VALUE> 
            </PROPERTY> 
            - <PROPERTY NAME="#_OtherPersistanceType" TYPE="string"> 
              <VALUE> </VALUE> 
            </PROPERTY> 
        </INSTANCE> 
      </IMETHODCALL> 
    </MESSAGE> 
  </SIMPLEREQ> 
</MESSAGE>
```

**Create Handler Instance**
IndicationFilter
- SystemCreationClassName: string (Key)
- SystemName: string (Key)
- CreationClassName: string (Key)
- Name: string (Key)
- SourceNamespace: string
- Query: string (Required)
- QueryLanguage: string (Required)

IndicationHandler
- SystemCreationClassName: string (Key)
- SystemName: string (Key)
- CreationClassName: string (Key)
- Name: string (Key)
- Owner: string
- PersistanceType: uint16 (Enum)
- OtherPersistanceType: string

Create Subscription Instance
Indication Service Role

CIM Indication Schema

Send me all CIM_AlertIndications.

ACME_AlertIndication

SystemIPAddress: string
SystemModelNumber: string
OSVersion: string
SystemSerialNumber: string
SystemSoftwareID:
MonitorID: string
MonitorProductVersion: string
MonitorEventID: uint1

HW_AlertIndication
Indication Service Role

The CIM Indication Service coordinates the handling of subscriptions among the Indication Providers.

- CIM Client
  - Send me ACME_AlertIndications.
  - Send me HW_AlertIndications.
  - Send me CIM_AlertIndications.

- ACME_AlertIndication Provider
  - Send me ACME_AlertIndications.

- HW_AlertIndication Provider
  - Send me HW_AlertIndications.

- CIM_AlertIndication Provider
  - Send me CIM_AlertIndications.

ACME Resources

Start monitoring for events on ACME resources.
Module Content

HP WBEM Services Indication Architecture

- Indication Components
  - Indication Generation
  - Indication Subscription
  - Indication Processing
  - **Indication Delivery**
  - Indication Consumption
Indication Delivery

Indication Receipt

Management System

CIM Listener

Indication Consumer

CIM Indication Export Message

Indication Delivery

Managed System

CIM XML Server Protocol Adapter

CIM Object Manager

Indication Handler

Indication Provider Resources

Indication Handler
Indication Delivery

A **CIM Message** is a well-defined request or response data packet used to exchange information between CIM Applications. There are two types of CIM Messages, CIM Operation Messages and CIM Export Messages.

- **A CIM Operation Message** is a CIM Message used to invoke an operation on the target CIM namespace.
- **A CIM Export Message** is a CIM Message used to communicate information about a CIM namespace or element that is foreign to the target. A CIM Export Message is informational only and does not define an operation on the target CIM namespace or even imply the existence of a target namespace.
A **CIM Listener** receives CIM Exports (e.g., Indications) requests, coordinates the distribution of requests among one or more Consumers and sends CIM Export responses.
Indication Delivery

A **CIM Indication Handler** receives Indications, performs the mapping between the internal representation of a CIM Indication and the desired format and protocol, and sends the Indication to the designated target.

Note: A CIM Server may support multiple "indication handler interfaces".

---

**Managed System**

- Indication Handler
- CIM-XML Indication Handler
- CIM Object Manager
- Indication Provider
- Resources

**Protocols**

- CIM-XML Server Protocol Adapter
- SNMP Mapper Indication Handler
- e-mail Indication Handler

---

OpenPegasus Developer Conference
Indication Delivery

A CIM Indication is a type of CIM Export Message.

A CIM Listener receives CIM Exports (e.g., Indications) requests, coordinates the distribution of requests among one or more Consumers and sends CIM Export responses.

A CIM-XML Indication Handler, functioning as a CIM Client, uses the CIM-XML DMTF protocol to send Indication to the designated target.

Note: With CIM Export Messages, the "interesting" data (e.g., the Indication) is based as part of the request.
Indication Delivery

A CIM Indication is a type of CIM Export Message.

Note: Indications are sent as CIM Export Messages.

Note: The CIM Server functions as an HTTP Client.

CIM Listener

CIM-XML Export Server

HTTP Server

CIM-XML Export Client

HTTP Client

CIM Client

CIM-XML Indication Handler

CIM Export Message Encoded Using CIM-XML

CIM Export Message Encoded Using CIM-XML

Packaged in an HTTP Message

Indication Receipt

Indication Delivery
Indication Delivery

```xml
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
    <MESSAGE ID="1007" PROTOCOLVERSION="1.0">
        <SIMPLEEXPREQ>
            <EXPMETHODCALL NAME="ExportIndication">
                <IPARAMVALUE NAME="NewIndication">
                    <INSTANCE CLASSNAME="CIM_AlertIndication">
                        <PROPERTY NAME="Description" TYPE="string">
                            <VALUE>Sample CIM_AlertIndication indication</VALUE>
                        </PROPERTY>
                        <PROPERTY NAME="AlertType" TYPE="uint16">
                            <VALUE>1</VALUE>
                        </PROPERTY>
                        <PROPERTY NAME="IndicationTime" TYPE="datetime">
                            <VALUE>20010515104354.000000:000</VALUE>
                        </PROPERTY>
                    </INSTANCE>
                </IPARAMVALUE>
            </EXPMETHODCALL>
        </SIMPLEEXPREQ>
    </MESSAGE>
</CIM>
```

ExportIndication

```c
void ExportIndication ( [IN] <instance> NewIndication )
```
CIM-XML Protocol Adapter
Module Content

HP WBEM Services Indication Architecture
- Indication Components
  - Indication Generation
  - Indication Subscription
  - Indication Processing
  - Indication Delivery
- Indication Consumption
Indication Consumer

A **CIM Listener** receives CIM Exports requests, coordinates the distribution of requests among 1 or more Consumers and sends CIM Export responses.

A **CIM Indication Consumer** "consumes" the CIM data (e.g., an Indication) encapsulated in the CIM Export Message.
Indication Consumer Example

Select an item to view its description.

See also:
- My Documents
- My Network Places
- My Computer
CIM Indication Consumer

A Indication has been received.

Store the Indication in a file for further processing.
## CIM Indication Consumer

<table>
<thead>
<tr>
<th>CIM Export</th>
<th>Implementation Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExportIndication</td>
<td>Indication Consumer</td>
</tr>
</tbody>
</table>

```java
handleIndication();
```

**Management Node**

- CIM Listener
- Indication Consumer
PEGASUS_USING_PEGASUS;
PEGASUS_USING_STD;

#include "ClassExampleIndicationConsumer.h"

#define LOGFILE "/opt/wbem/sample/ClassProviders/LogIndicationConsumer/logFile"

ClassExampleIndicationConsumer::ClassExampleIndicationConsumer()
{
}

ClassExampleIndicationConsumer::~ClassExampleIndicationConsumer()
{
}

void ClassExampleIndicationConsumer::initialize()
{
}

void ClassExampleIndicationConsumer::terminate()
{
    delete this;
}

void ClassExampleIndicationConsumer::handleIndication(
    const OperationContext & context,
    const String & url,
    const CIMInstance & indicationInstance)
{
    try
    {
        FILE *logFileHandle = fopen(LOGFILE, "a+");
        fprintf(logFileHandle, "Received Indication\n");
        fclose(logFileHandle);
    }
    catch (...){}
    
}
Standalone CIM Listener

Indication Consumer "registers" with the CIM Listener to receive Indications.

Note: CIM Listeners waits at an "application specific" port to receive Indications (i.e., CIM Export Messages)
CIM Listener with CIM Server

**Note:** Allows Indication Consumers to "share" ports 5988 and 5989 with CIM Server

4 Indication Delivery

5 Indication Receipt

6 Indication Consumption

 Managed & Management System

HTTP Client

CIM-XML Encoder/Decoder

CIM Object Manager

Indication Consumer Manager

Provider Manager

Provider

Provider

Resources

Resources
Provider Registration Schema

CIM_ManagedElement
- Description: string
- Caption: string
- ElementName: string

CIM_ManagedSystemElement
- InstallDate: datetime
- Name: string
- Status: string
- OperationalStatus: uint16
- StatusDescriptions: string

CIM_LogicalElement

PG_Provider

PG_ProviderModule
- Vendor: string
- Version: string
- InterfaceType: string
- InterfaceVersion: string
- Location: string
- OperationalStatus: uint16
- OtherStatusDescription: string
- uint32 start();
- uint32 stop();

PG_ProviderModuleElements

PG_ProviderCapabilities
- ClassName: string
- Namespaces: string
- SupportedProperties: string
- SupportedMethods: string

PG_ProviderCapabilitiesElements

PG_CapabilitiesRegistration
- ProviderModuleName: string
- ProviderName: string
- CapabilityID: string
- ProviderType: uint16

PG_ConsumerCapabilities
- IndicationDestinations: string

PG_ConsumerCapabilitiesElements

The diagram illustrates the provider registration schema with various classes and their attributes.
## Provider Capabilities

### ACME Indication Provider

<table>
<thead>
<tr>
<th>Set of Operations</th>
<th>ExportIndication</th>
</tr>
</thead>
</table>

### PG_ConsumerCapabilities

```cpp
instance of PG_ProviderCapabilities
{
    ProviderModuleName = "SampleConsumerModule";
    ProviderName = "LogIndicationConsumer";
    CapabilityID = "1";
    ProviderType = { 6 }; // Indication Consumer
    Destinations = { PGLogger };
};
```

### Description of Consumer Capabilities