Indication Standard
DMTF

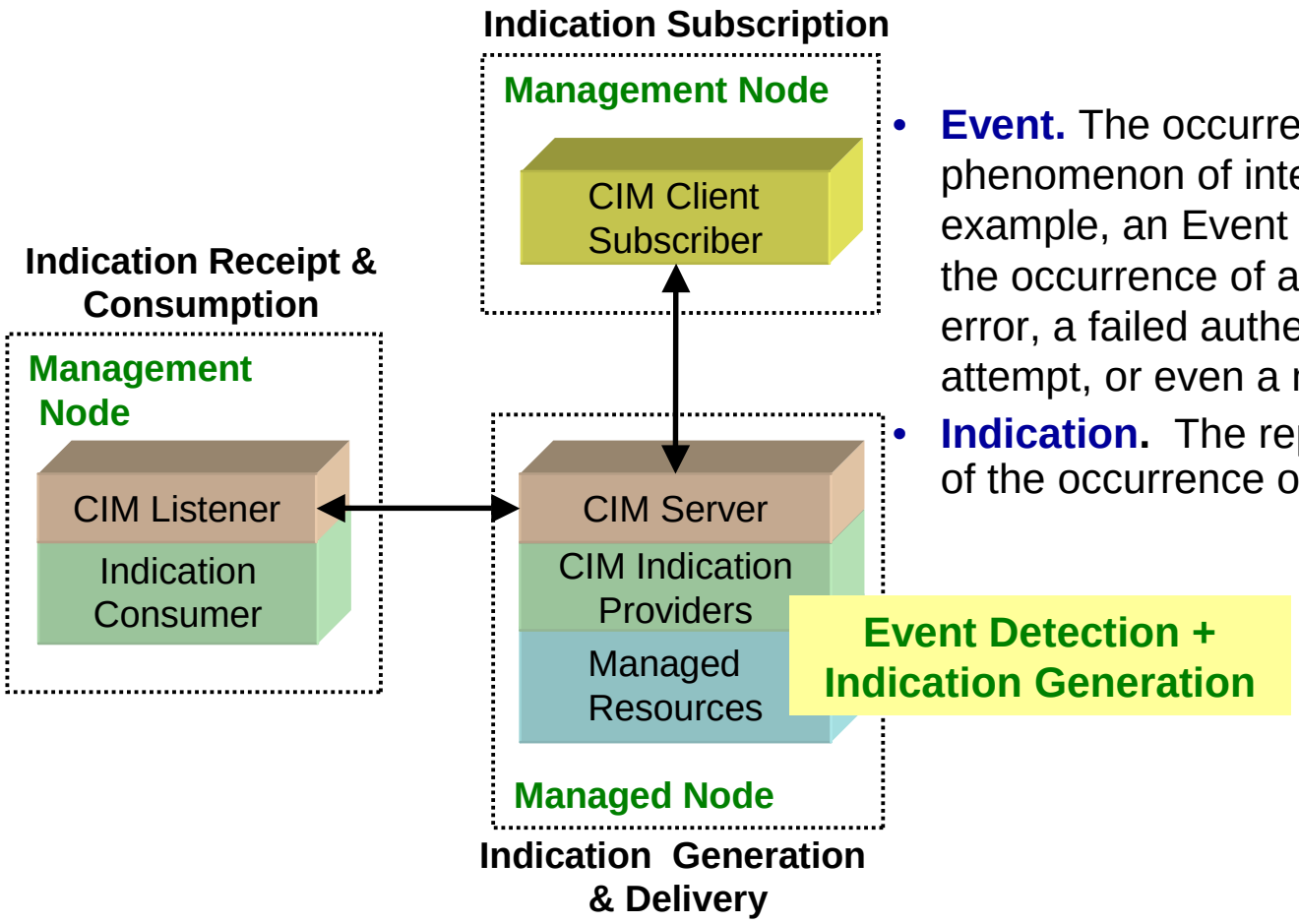
Roger Kumpf
Hewlett-Packard

Module Content

CIM Indication Overview

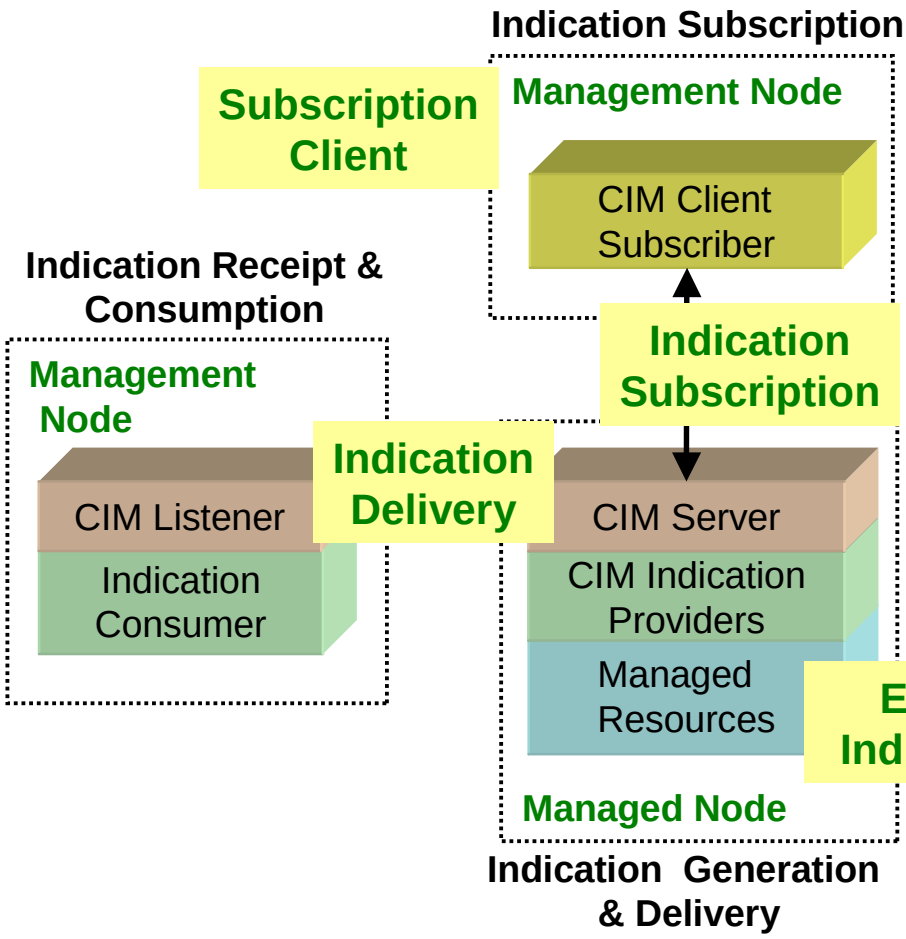
- **Terminology**
- Indication Hierarchy
- Indication Subscription Schema

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



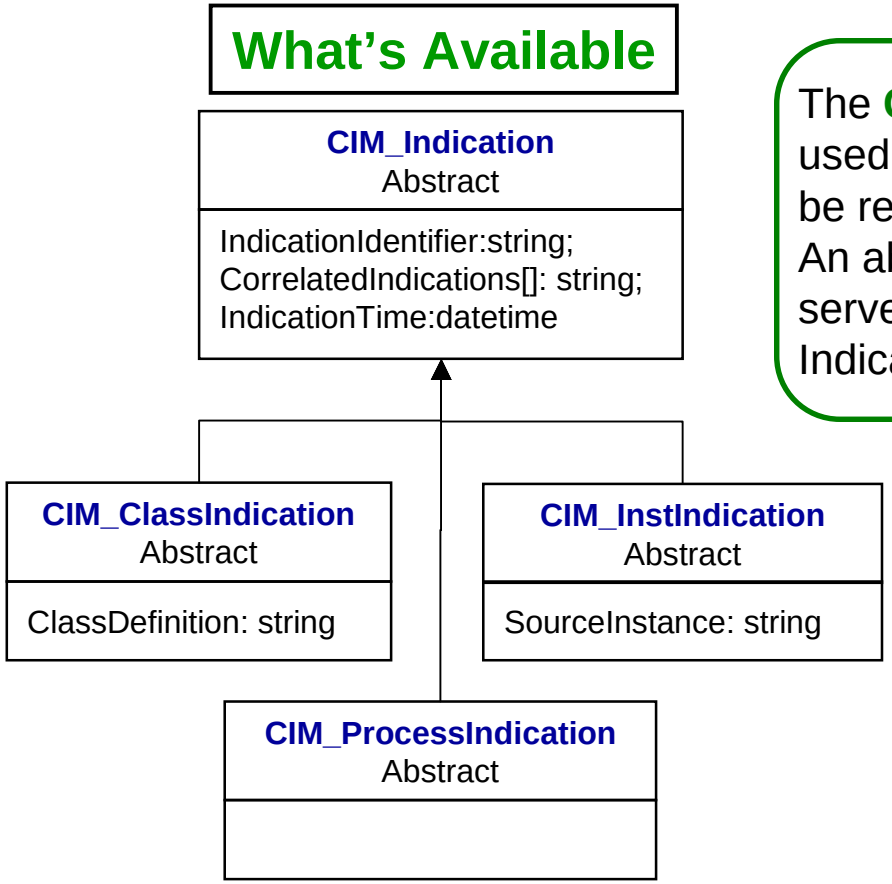
- **Indication Subscription.** The process of registering to receive Indications.
- **Indication Delivery.** The process of transporting one or more indications to a designated destination. The intended protocol and destination are specified as part of the subscription definition.
- **Subscription Client.**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

Module Content

CIM Indication Overview

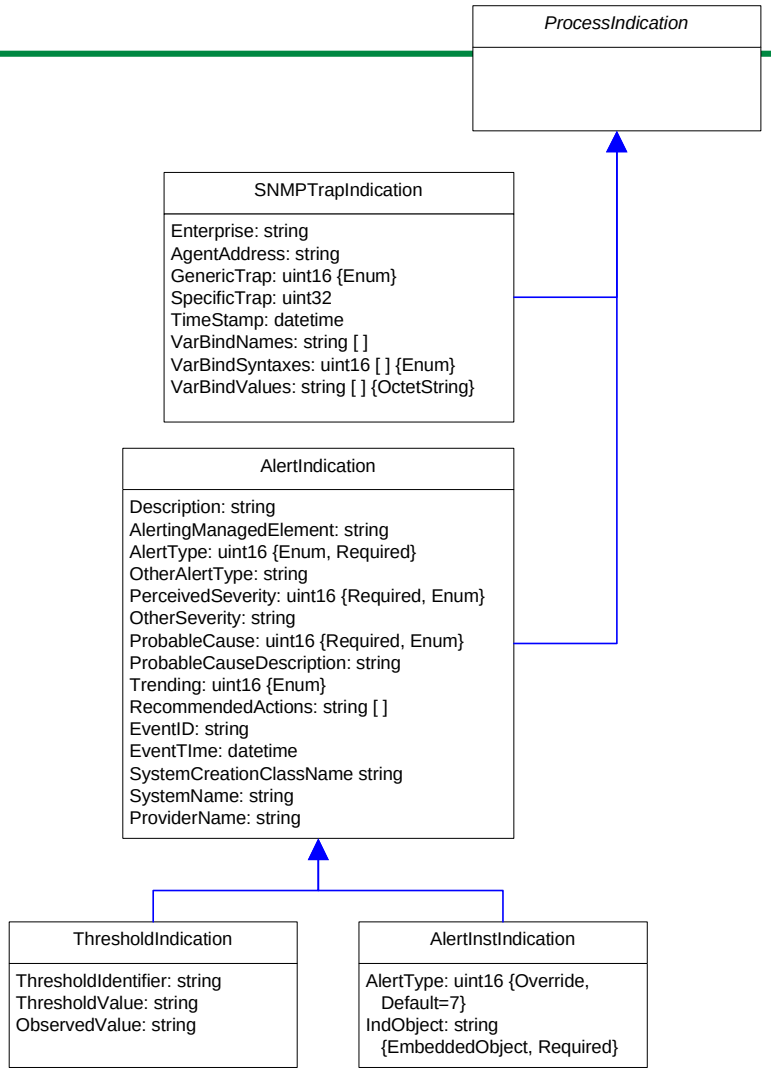
- Terminology
- **Indication Hierarchy**
- Indication Subscription Schema

Indication Hierarchy

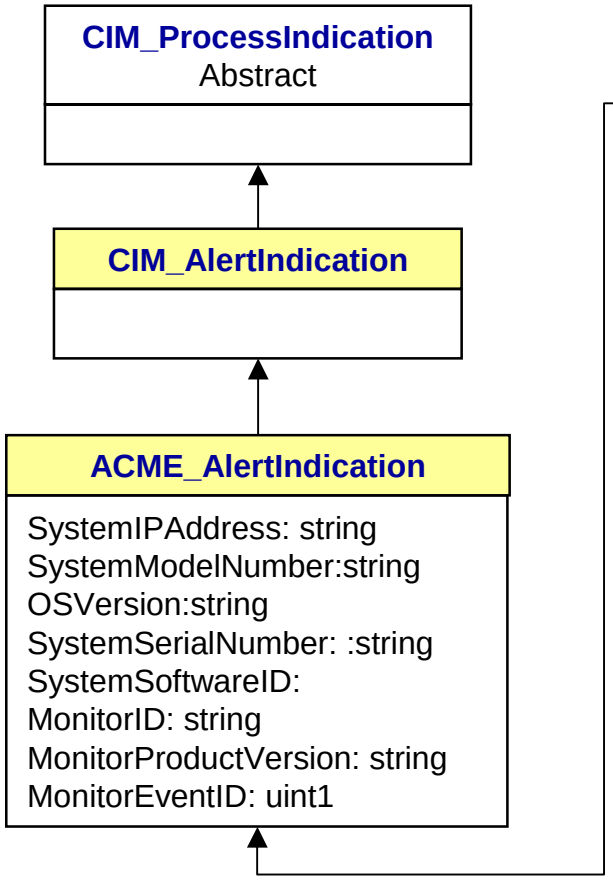


The **CIM Indication Hierarchy** is used to describe the Events that can be represented as CIM Indications. An abstract class, **CIM_Indication**, serves as the base class for all Indication classes.

CIM_ProcessIndication



Model Extensions



What's Available from ACME

ACME_UPSAlertIndication

UPSttyPath: string
UPSStatusCode: uint16
UPSFirmwareVersion: string
BatteryVoltage: uint16
RemainingRuntimeReserve: unit16
UnitType: string
RatedInputVoltage:uint16
NominalFrequency: unit16
RatedVA: unit16
OutputVoltage: uint16
Phase: uint16

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

Module Content

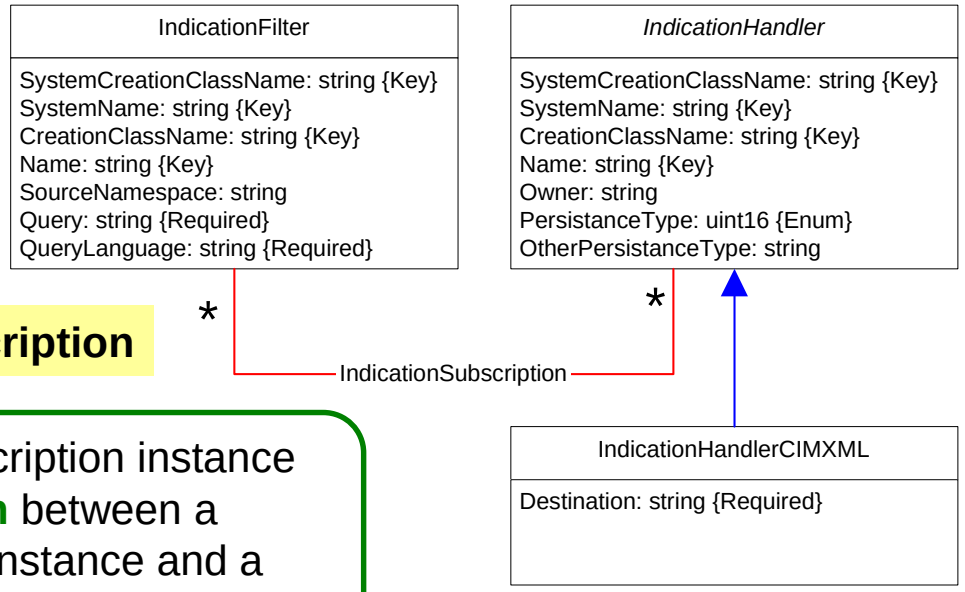
CIM Indication Overview

- Terminology
- Indication Hierarchy
- **Indication Subscriptions**

CIM_IndicationSubscription

CIM_IndicationFilter
What to Send

CIM_IndicationHandler
How & Where to Send



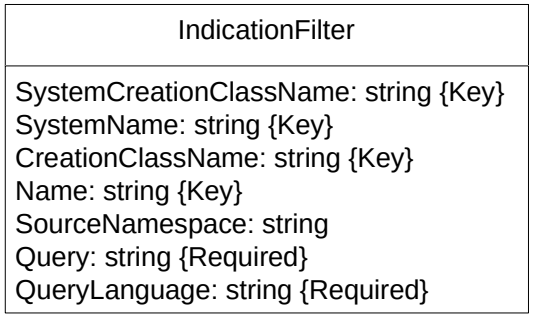
CIM_IndicationSubscription

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.

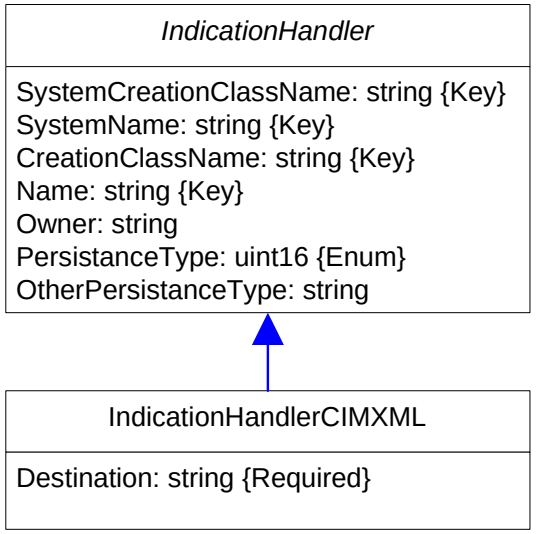
CIM_IndicationFilter
What to Send



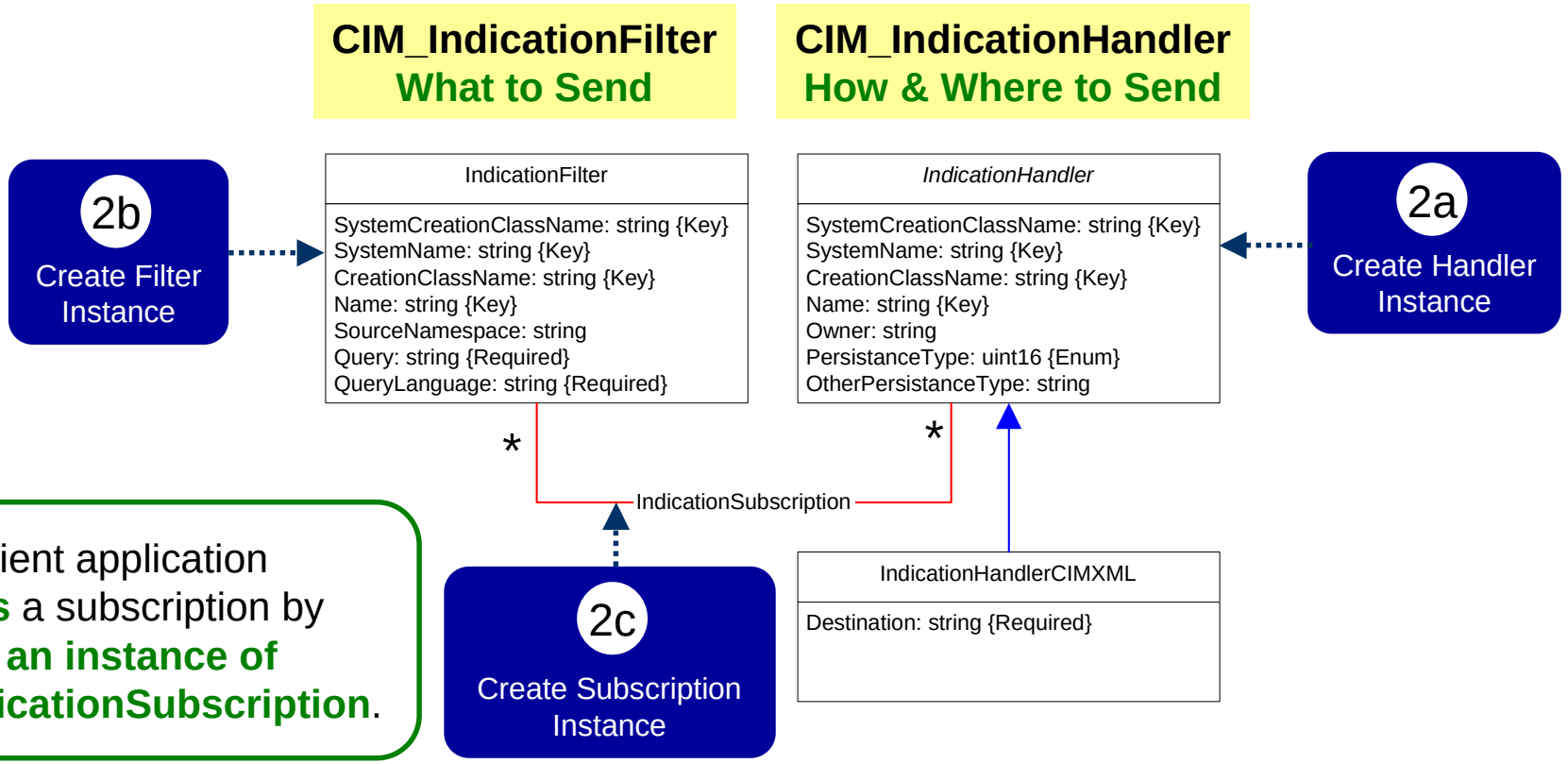
CIM_IndicationHandler

- **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.

CIM_IndicationHandler
How & Where to Send



CIM_IndicationSubscription



A CIM Client application **activates** a subscription by **creating an instance of CIM_IndicationSubscription**.

CIM_IndicationSubscription

CIM_IndicationSubscription

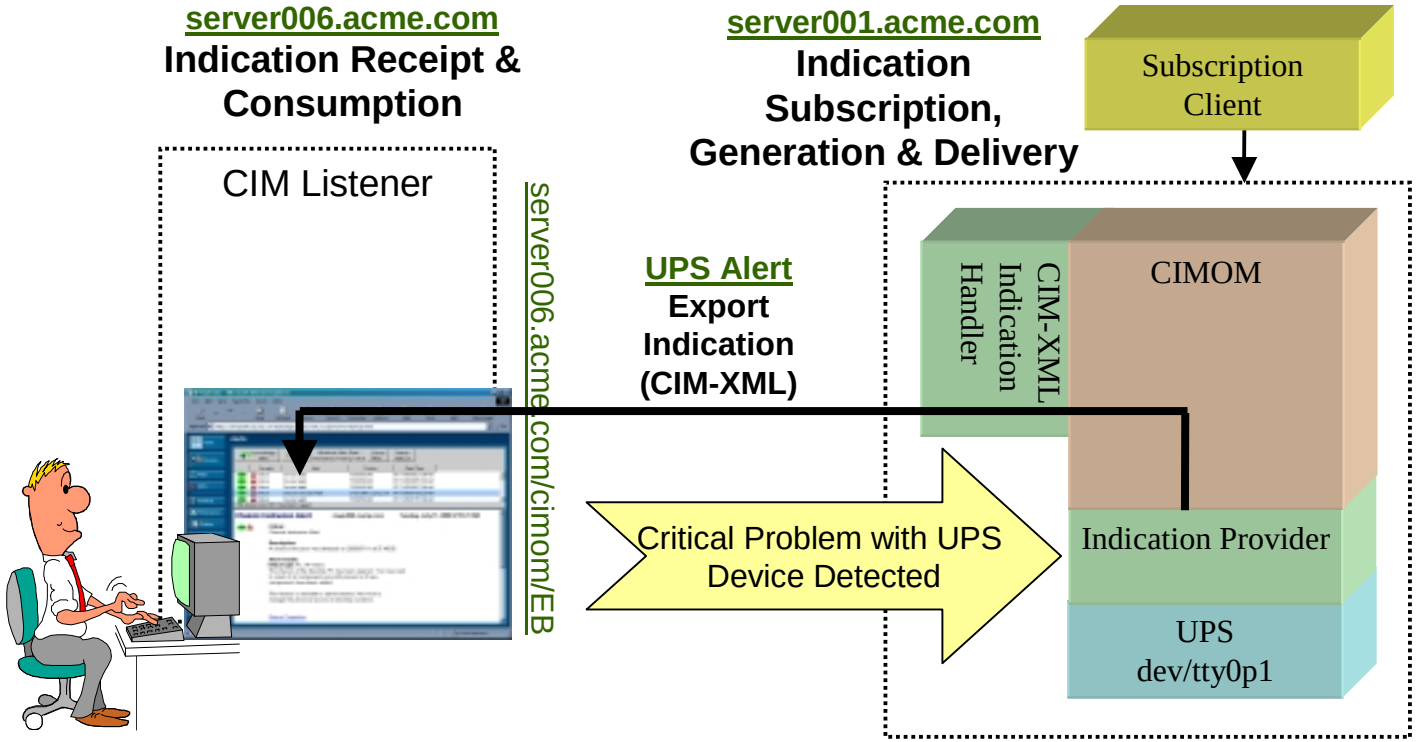
IndicationSubscription
IndicationFilter: ref Filter {Key} FilterIndicationHandler: ref Handler {Key} OnFatalErrorPolicy: uint16 {Enum} OtherOnFalalErrorPolicy: string FailureTriggerTimeInterval: uint64 SubscriptionState: uint16 {Enum, Write} OtherSubscriptionState: string TimeOfLastStateChange: datetime SubscriptionDuration: uint64 {Write} SubscriptionStartTime: datetime SubscriptionTimeRemaining: uint64 {Write} RepeatNotificationPolicy: uint16 {Enum} OtherRepeatNotificationPolicy: string RepeatNotificationInterval: uint64 RepeatNotificationGap: uint64 RepeatNotificationCount: uint16

- 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.

Example

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”.



IndicationFilter Example

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

CIM_IndicationFilter
What to Send

IndicationHandler Example

CIM_IndicationHandler
How & Where to Send

CIM_IndicationHandlerCIMXML		
	Property Name	**Value**
	Caption	Acme Hardware Event Browser
	Description	Location of Acme Event Browser responsible for monitoring hardware events for this system.
Key	SystemCreationClassName	CIM_UnitaryComputerSystem
Key	SystemName	server001.acme.com
Key	CreationClassName	CIM_IndicationHandlerCIMXML
Key	Name	ACMESubscription12345
	Owner	EventAdmin
Required	Destination	server006.acme.com/cimom/EB

IndicationSubscription Example

CIM_IndicationSubscription
Activate Subscription

CIM_IndicationSubscription		
	Property Name	**Value <instanceName)**
key	Filter	Class Name = CIM_IndicationFilter Key Binding = SystemCreateClassName = “CIM_UnitaryComputerSystem”, SystemName = “server001.acme.com”, CreationClassName = “CIM_IndicationFilter, Name = “ACMESubscription12345”
key	Handler	Class Name = CIM_IndicationHanlderCIMXML Key Binding = SystemCreateClassName = “CIM_UnitaryComputerSystem”, SystemName = “server001.acme.com”, CreationClassName = “CIM_IndicationHandlerCIMXML, Name = “ACMESubscription12345”

Indication Architecture OpenPegasus

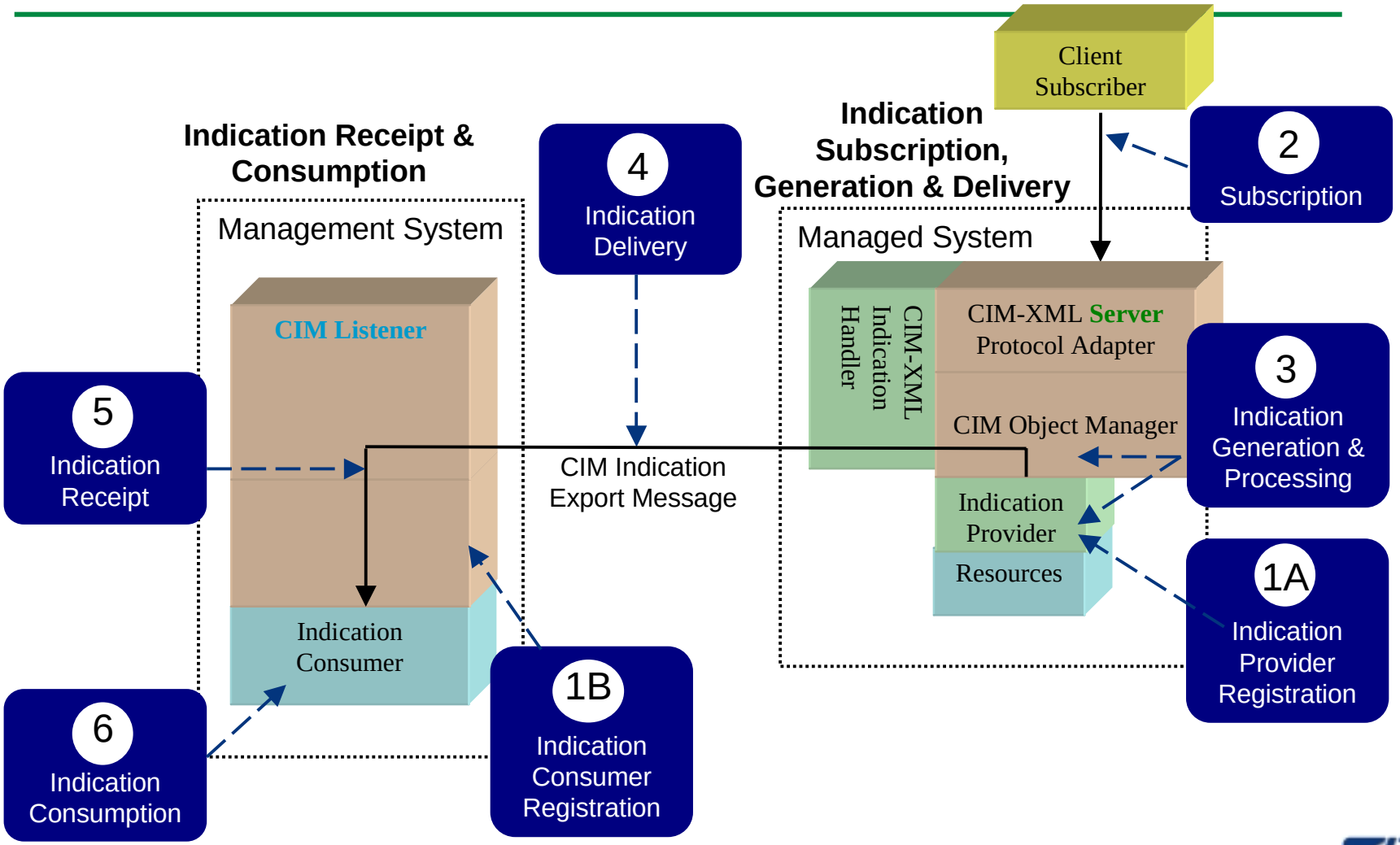
Roger Kumpf
Hewlett-Packard

Module Content

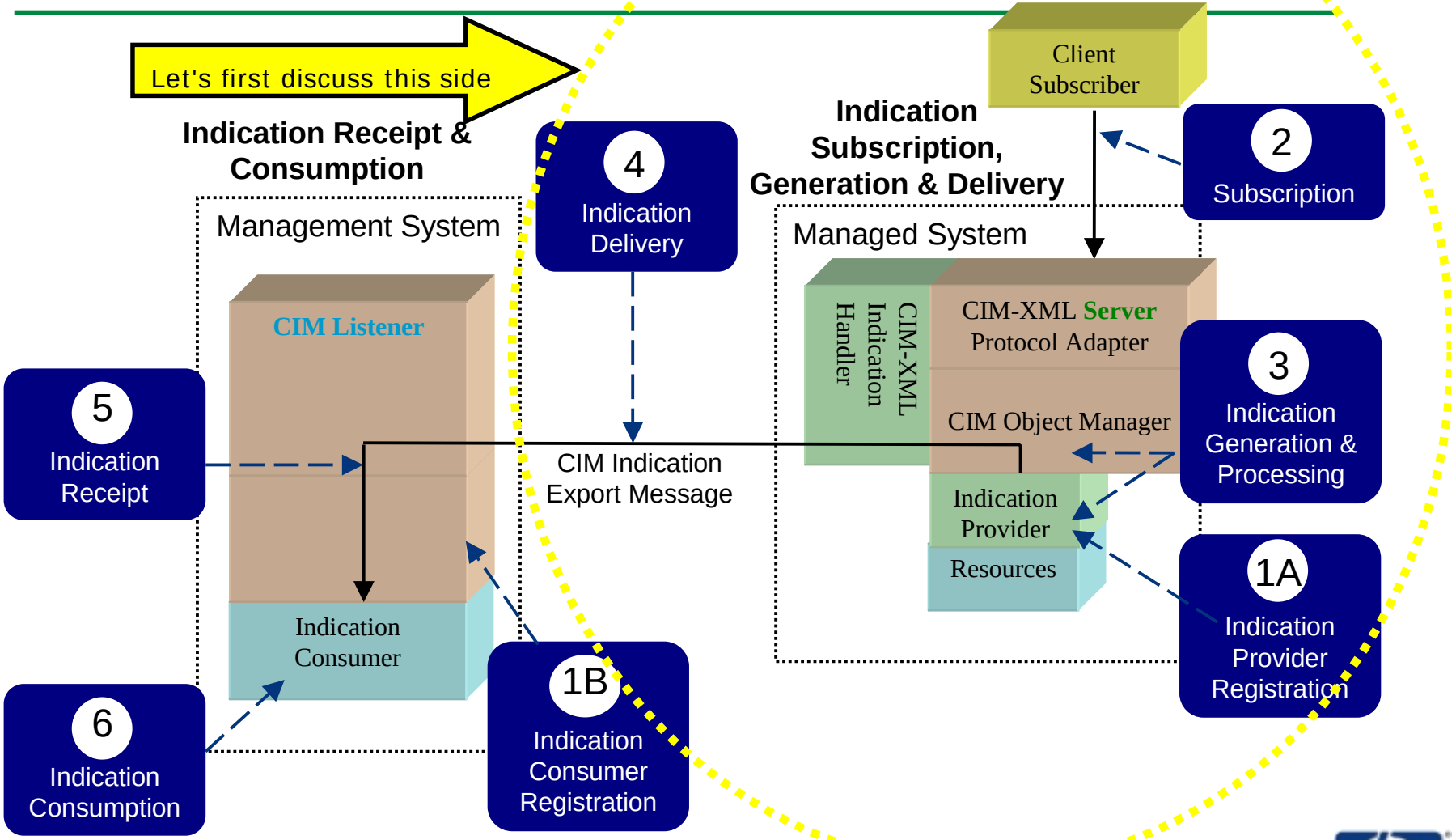
HP WBEM Services Indication Architecture

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

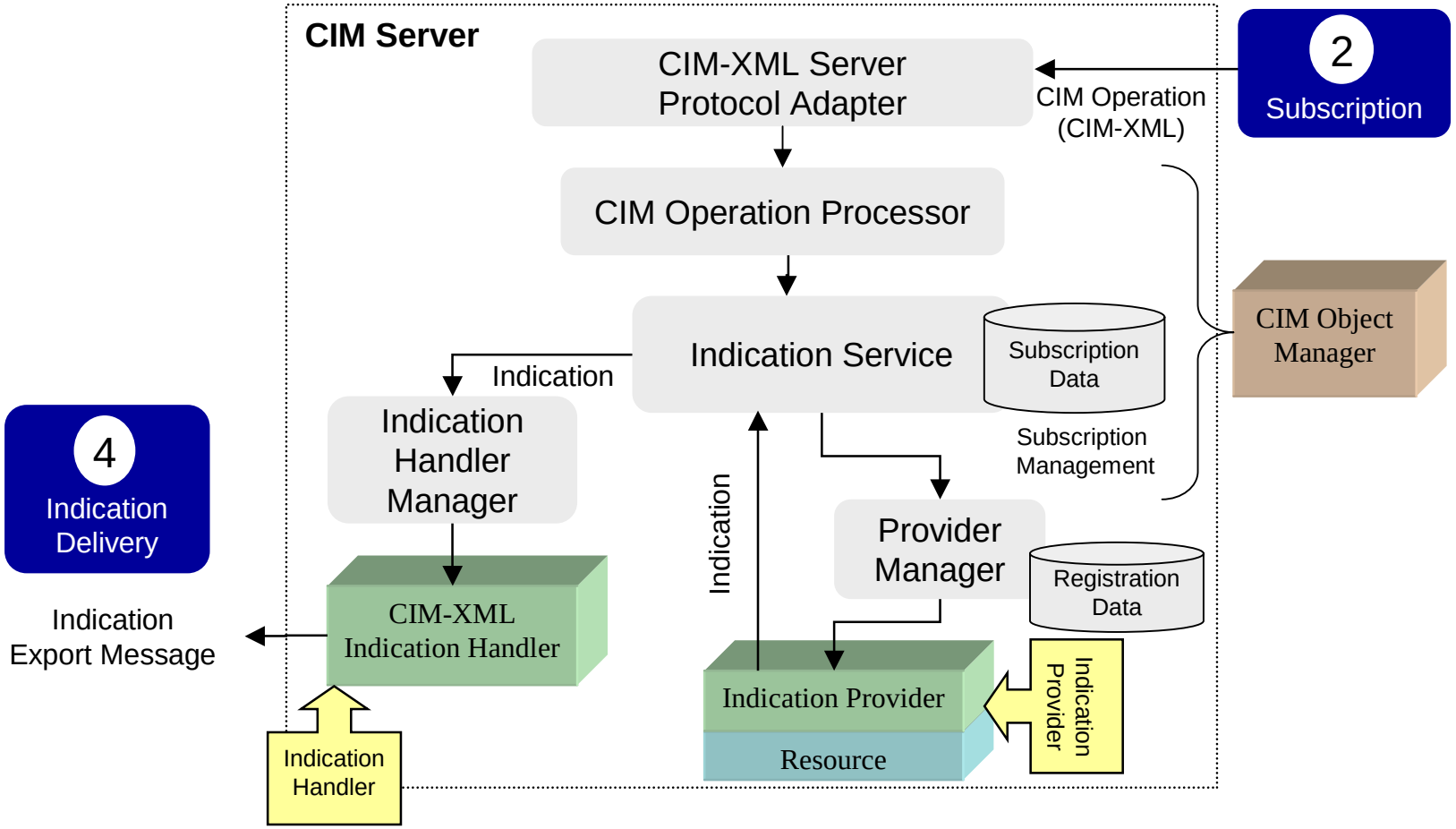
Indication Components



Indication Components



Indication Architecture



Module Content

HP WBEM Services Indication Architecture

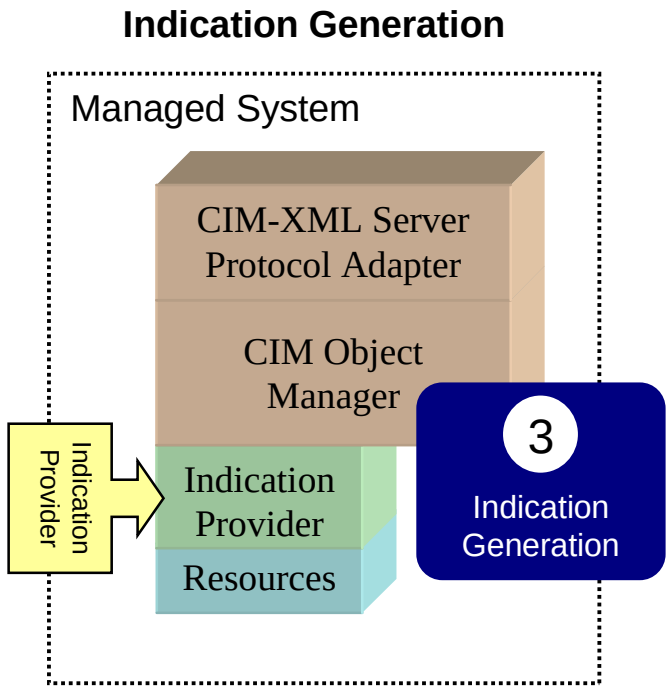
- Indication Components
  - **Indication Generation**
  - Indication Subscription
  - Indication Delivery
  - Indication Consumption

Indication Generation

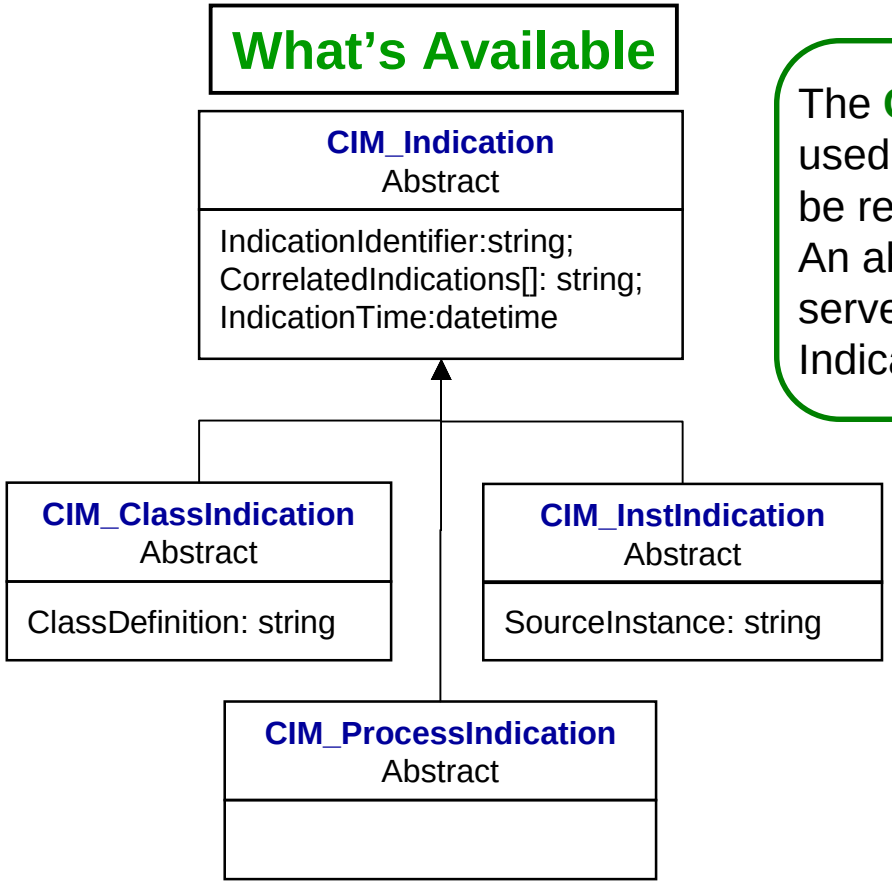
Simply defined, an **Event** is the occurrence of a phenomenon of interest. For example, an Event can denote the occurrence of a disk write error, a failed authentication attempt, or even a mouse click.

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.



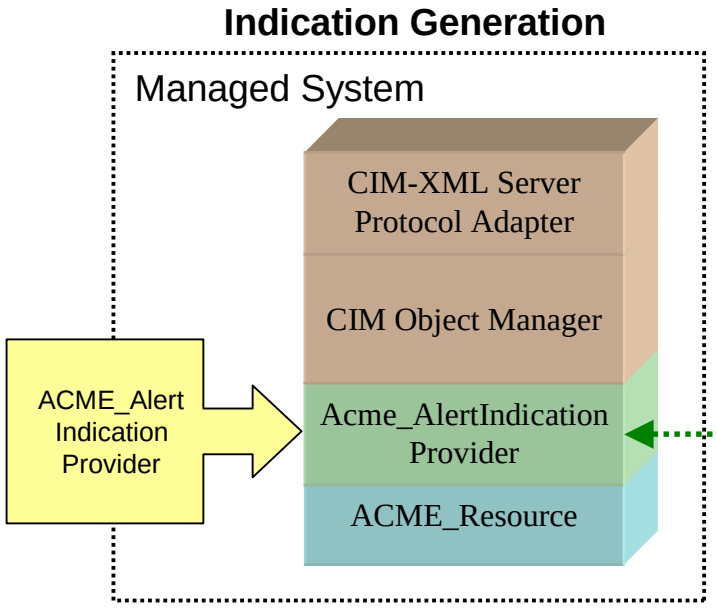
Indication Hierarchy



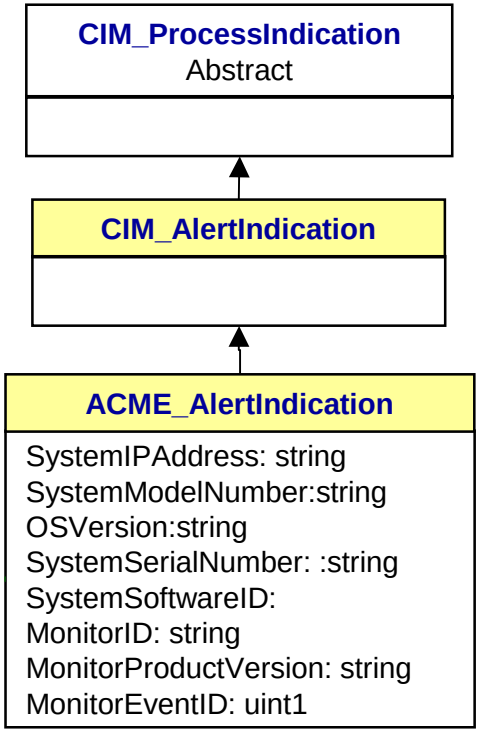
The **CIM Indication Hierarchy** is used to describe the Events that can be represented as CIM Indications. An abstract class, **CIM_Indication**, serves as the base class for all Indication classes.

Indication Generation

A **CIM Indication Provider** registers with the CIM Server to generate events for one or more Indication classes.

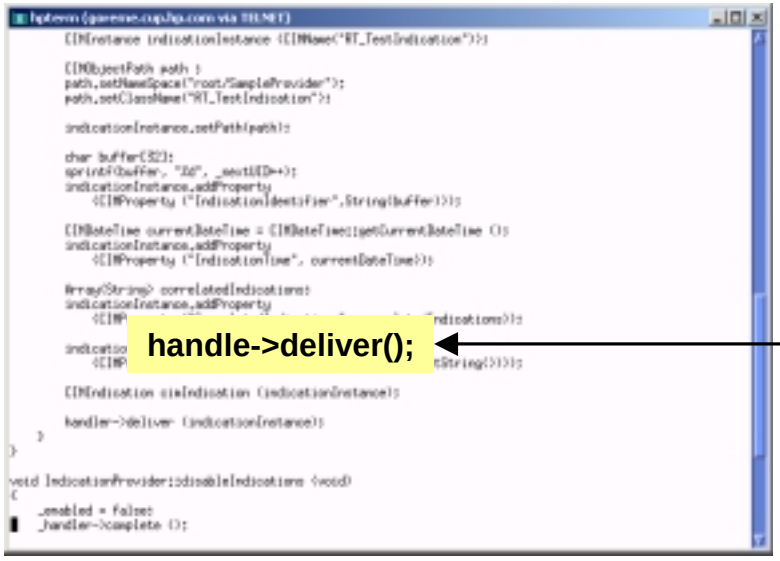


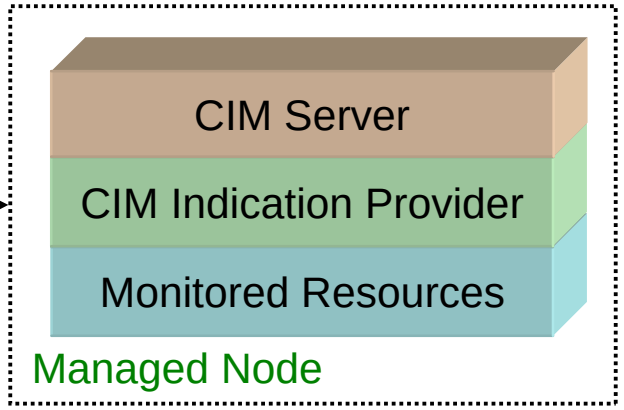
CIM Indication Hierarchy



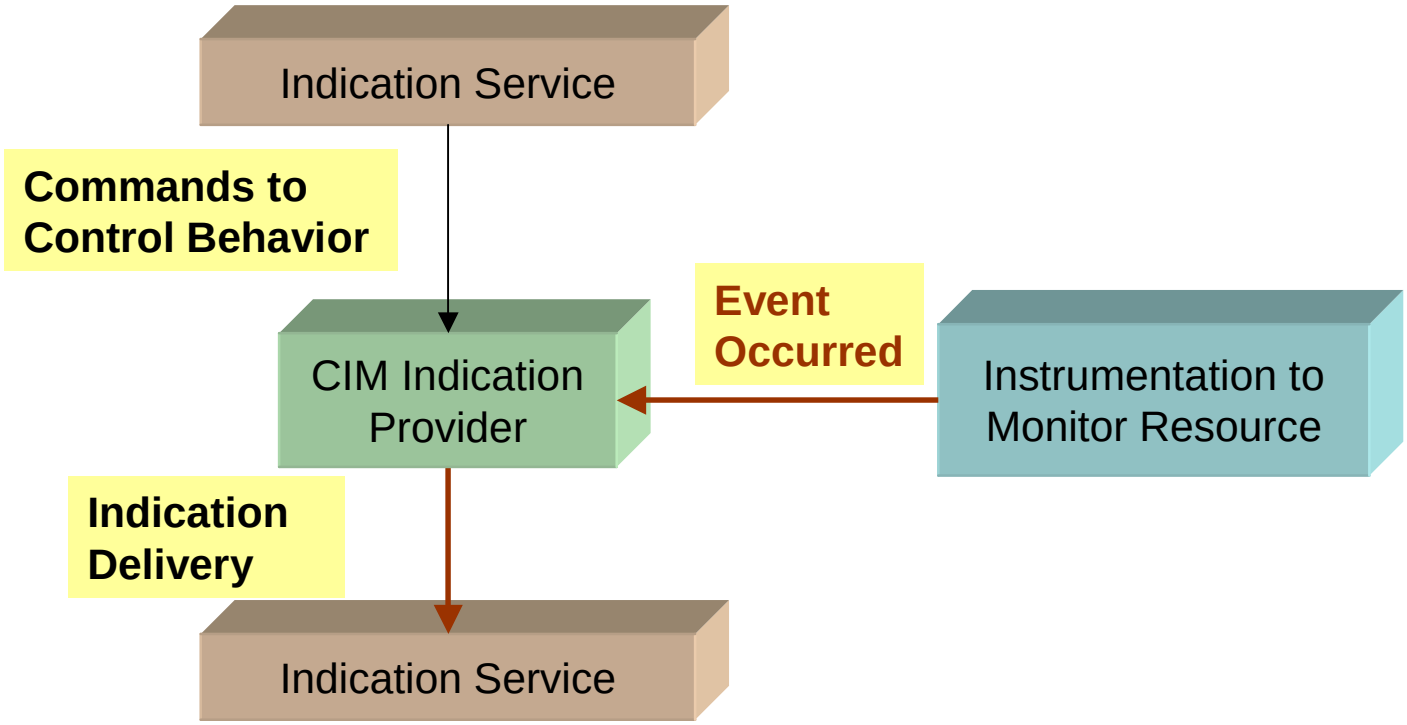
Indication Provider

CIM Export	Implementation Owner
ExportIndication	**Indication Providers**

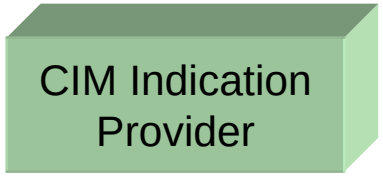




CIM Provider Logic



Indication Provider API



createSubscription

OpenPegasus C++ Provider API

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

modifySubscription

OpenPegasus C++ Provider API

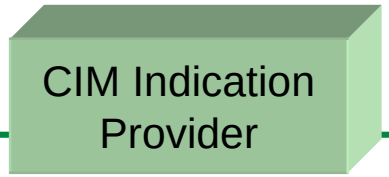
```
 const OperationContext & context,
        const CIMObjectPath & subscriptionName,
        const Array<CIMObjectPath> & classNames,
        const CIMPropertyList & propertyList,
        const Uint16 repeatNotificationPolicy) = 0;
```

deleteSubscription

OpenPegasus C++ Provider API

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

CIM Provider Logic



Process createInstance of CIM_IndicationSubscritpion

Process modifyInstance of CIM_IndicationSubscritpion

Process deleteInstance of CIM_IndicationSubscritpion

OpenPegasus C++ Provider API

createSubscription

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

OpenPegasus C++ Provider API

modifySubscription

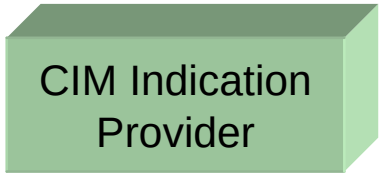
```
 const OperationContext & context,
        const CIMObjectPath & subscriptionName,
        const Array<CIMObjectPath> & classNames,
        const CIMPropertyList & propertyList,
        const Uint16 repeatNotificationPolicy) = 0;
```

OpenPegasus C++ Provider API

deleteSubscription

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

Indication Provider API



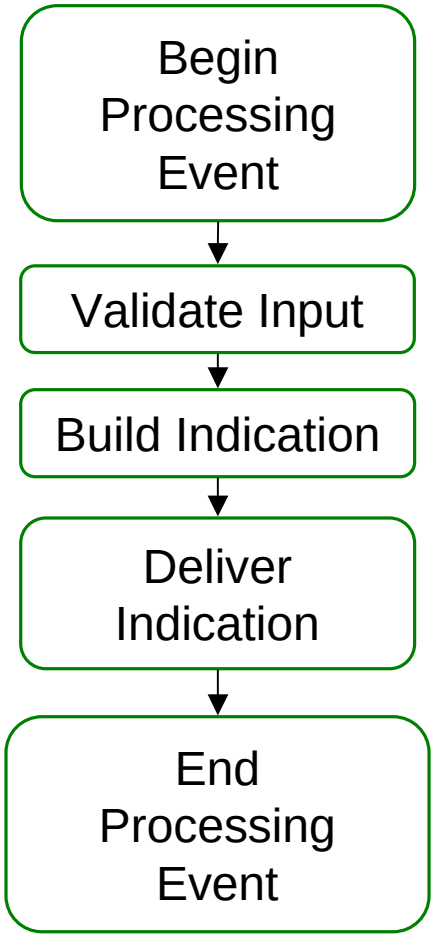
enableIndications — **OpenPegasus C++ Provider API**

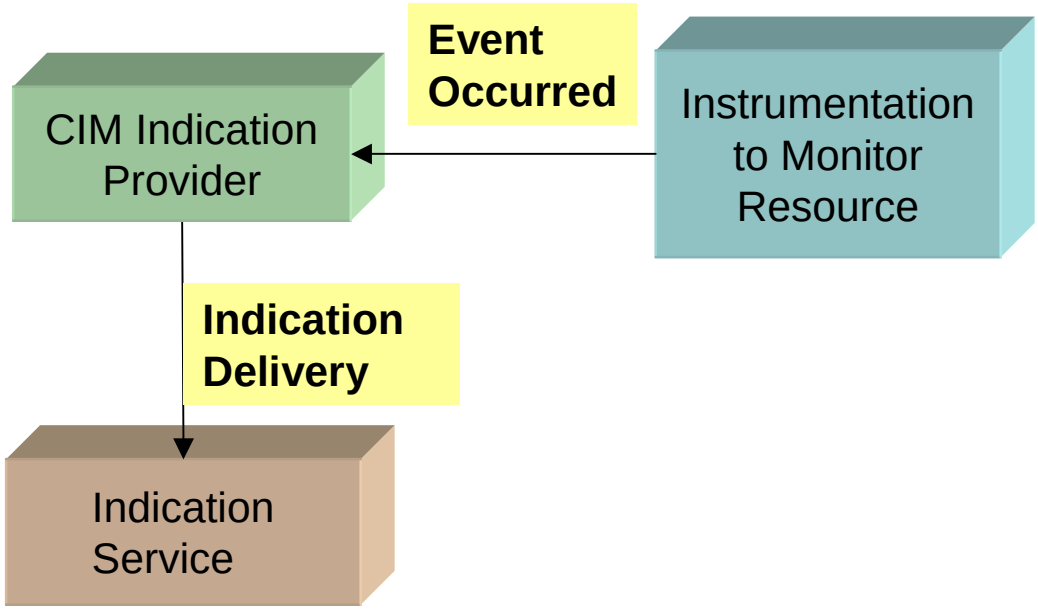
```
void enableIndications(IndicationResponseHandler & handler)
```

disableIndications — **OpenPegasus C++ Provider API**

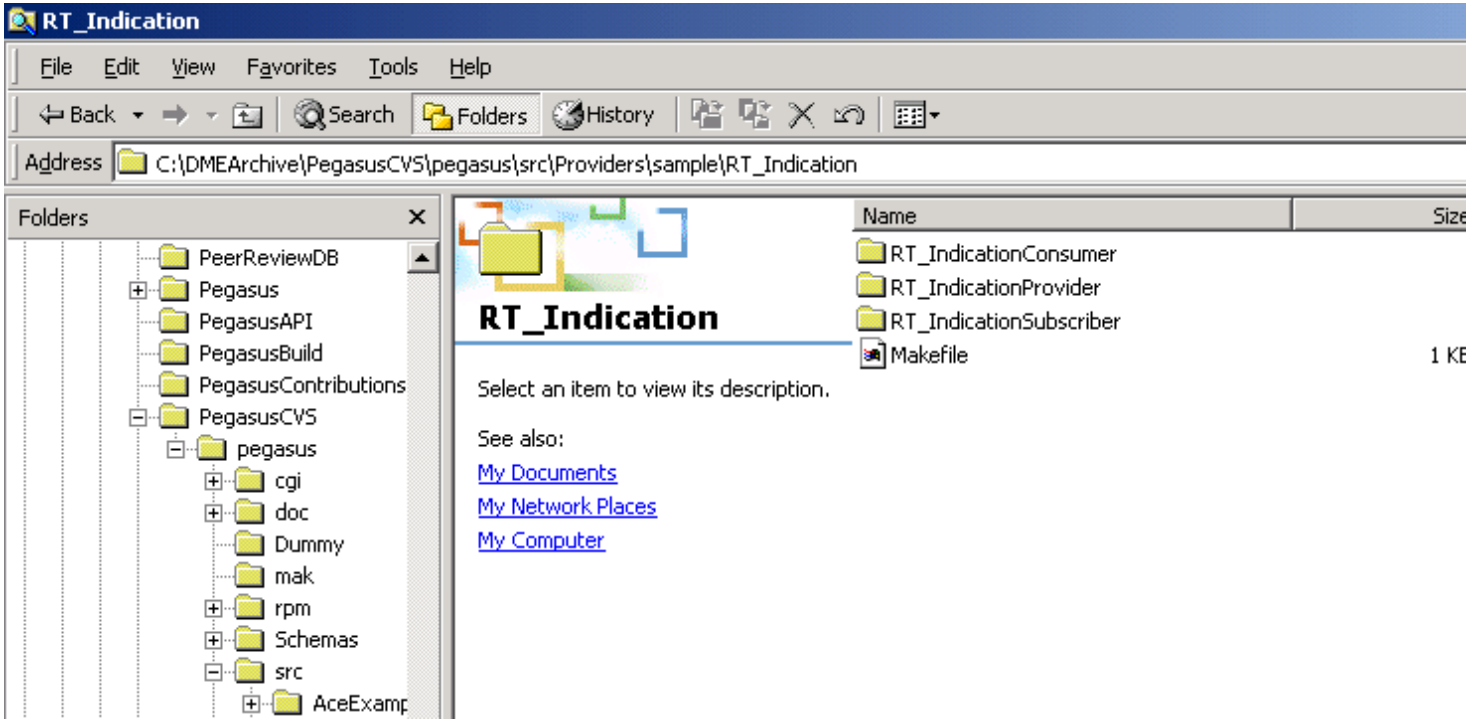
```
void disableIndications(void)
```

CIM Provider Logic





Indication Provider Example

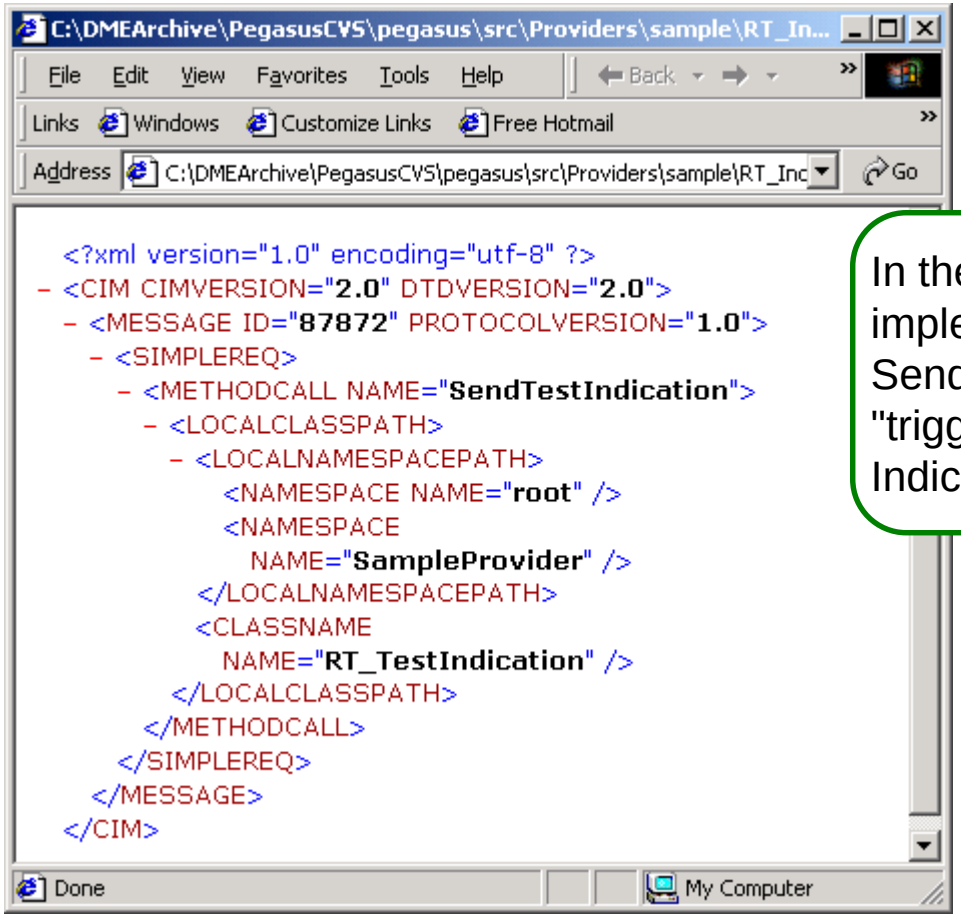


hpterm (goreme.cup.hp.com via TELNET)

_generateIndication

```
void _generateIndication (
    IndicationResponseHandler * handler,
    const CIMName methodName)
{
    if (_enabled)
    {
        CIMInstance indicationInstance (CIMName("RT_TestIndication"));

        CIMObjectPath path ;
        path.setNameSpace("root/SampleProvider");
        path.setClassName("RT_TestIndication");

        indicationInstance.setPath(path);

        char buffer[32];
        sprintf(buffer, "%d", _nextUID++);
        indicationInstance.addProperty
            (CIMProperty ("IndicationIdentifier",String(buffer)));

        CIMDateTime currentDateTime = CIMDateTime::getCurrentDateTime ();
        indicationInstance.addProperty
            (CIMProperty ("IndicationTime", currentDateTime));

        Array<String> correlatedIndications;
        indicationInstance.addProperty
            (CIMProperty ("CorrelatedIndications", correlatedIndications));

        indicationInstance.addProperty
            (CIMProperty ("MethodName", CIMValue(methodName.getString())));

        CIMIndication cimIndication (indicationInstance);

        handler->deliver (indicationInstance);
    }
}
```

handler->deliver()



In the RT Indication Example we've implemented an extrinsic method SendTestIndication to allow us to "trigger" the generation of an Indication.

RT_IndicationProvider.cpp - WordPad

File Edit View Insert Format Help

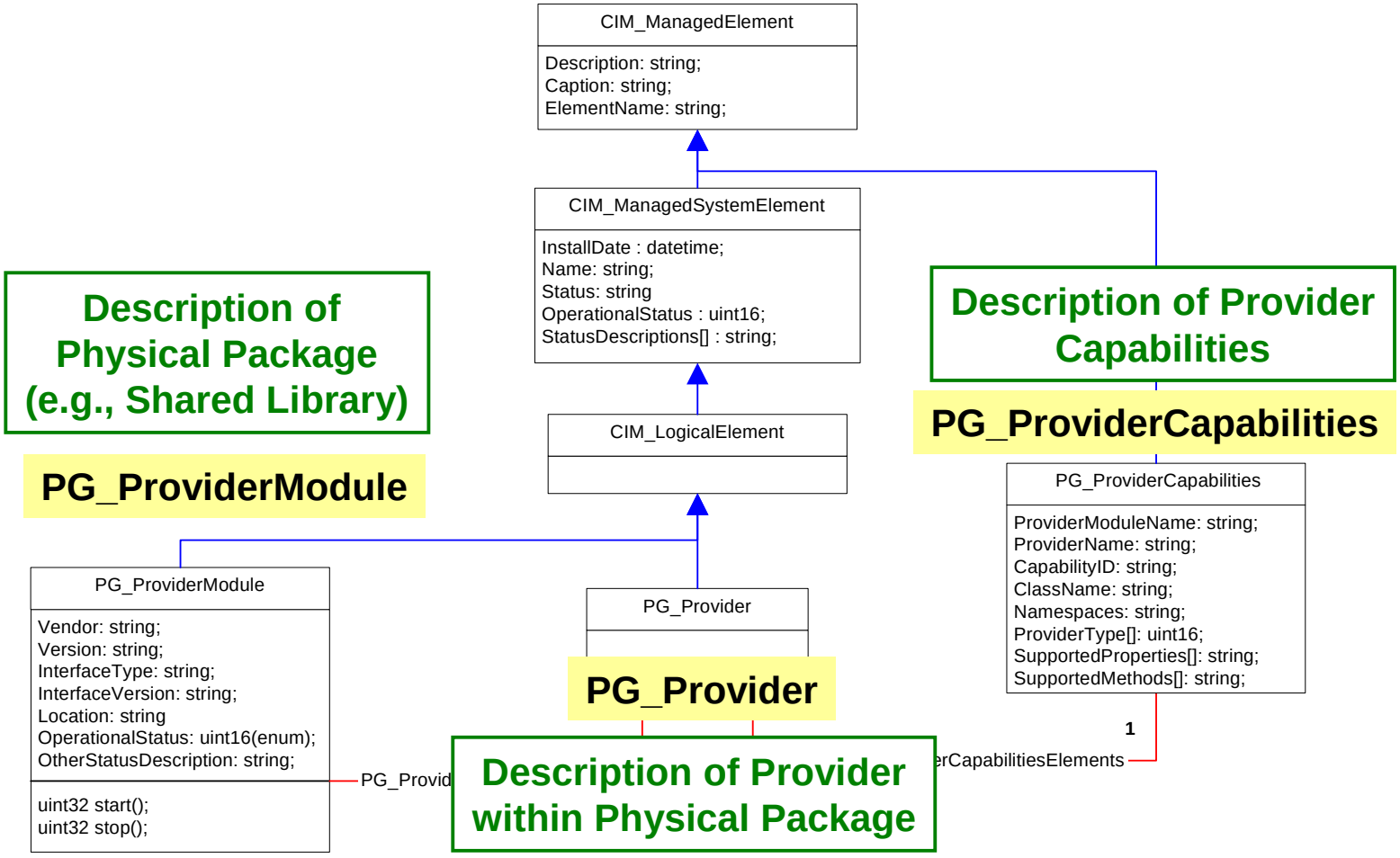
RT_IndicationProvider::invokeMethod

```
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_STD(endl);
        }

        handler.complete();

        if (sendIndication)
           _generateIndication(_handler,"generateIndication");
}
```

_generateIndication

For Help, press F1

Indication Provider Registration

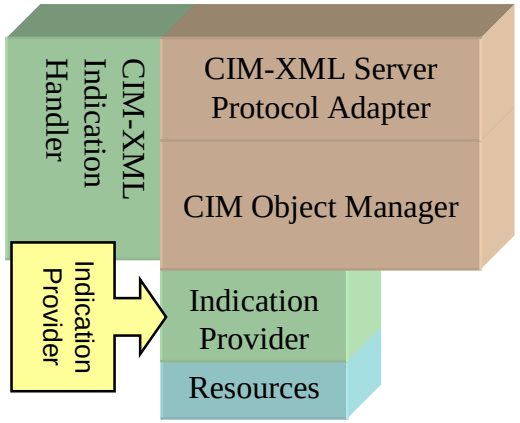


CIM Provider Types

CIM Operation	Implementation Owner[1]
GetClass, CreateClass, ModifyClass, DeleteClass, GetQualifier, SetQualifier, DeleteQualifier, EnumerateQualifier, EnumerateClasses, EnumerateClassNames,	CIMOM
GetInstance, EnumerateInstances, EnumerateInstanceNames, GetProperty, SetProperty, CreateInstance, ModifyInstance, DeleteInstance	**Instance Provider**
InvokeMethod	**Method Provider**
References, ReferenceNames, Associators, AssociatorNames	**Association Providers**
ExportIndication	**Indication Providers**

Indication Provider

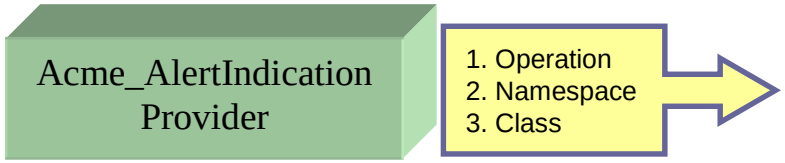
A **CIM Indication Provider** translates the occurrence of an Event into a CIM Indication and sends the Indication to the CIM Object Manager for further processing and delivery.



[1] Details in this column vary by implementation.

Provider Capabilities

Acme_AlertIndication Provider	
Set of Operations	Indication Provider Operations
Class	ACME_AlertIndication
Namespace	root/cimv2



PG_ProviderCapabilities

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

```
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
};
```

Description of Provider Capabilities

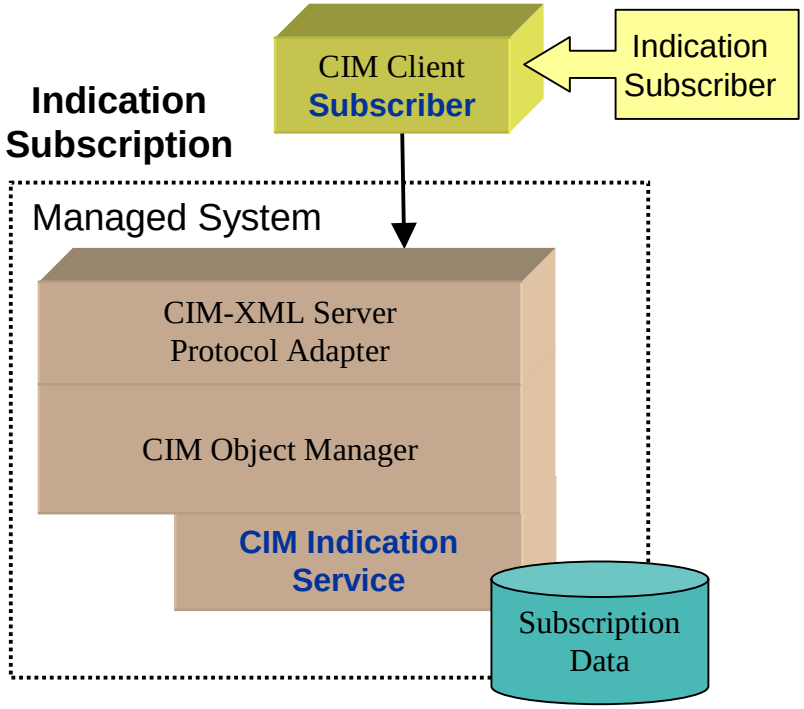
Module Content

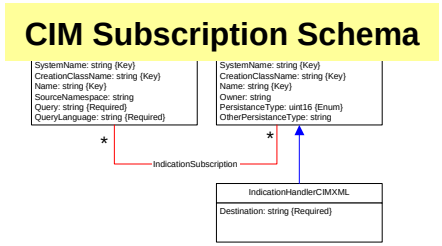
HP WBEM Services Indication Architecture

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

Indication Subscription

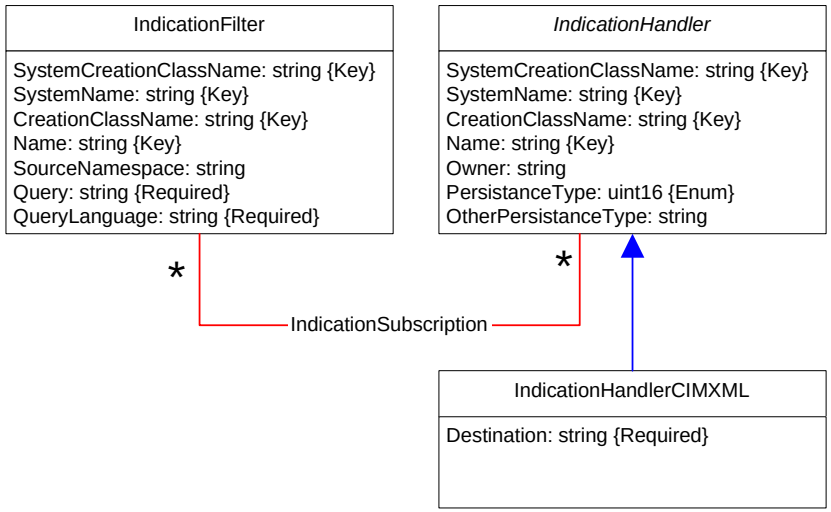
- An **"Indication Subscriber"** is a **CIM Client** that issues CIM Operation requests to create instances of the CIM_IndicationSubscription class.
- A **CIM Server** receives and processes CIM Operation requests and issues CIM Operation responses.
- The **CIM Indication Service** is a component of the CIM Object Manager. It is responsible for the processing of CIM Operations on the classes in the CIM Subscription Schema.

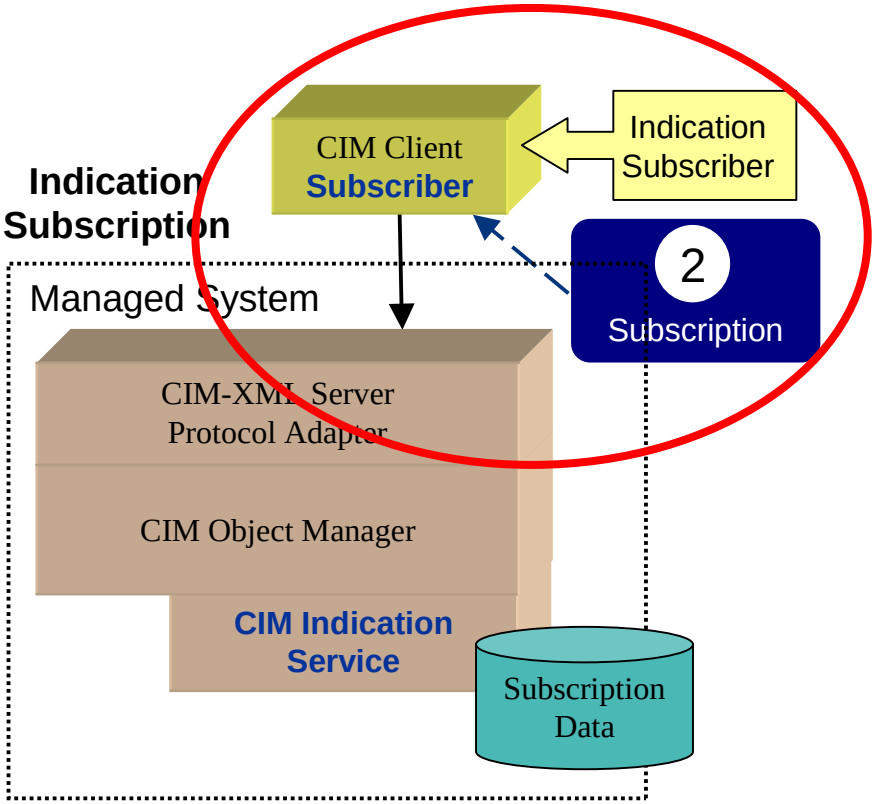




Indication Subscriptions

CIM Subscription Schema



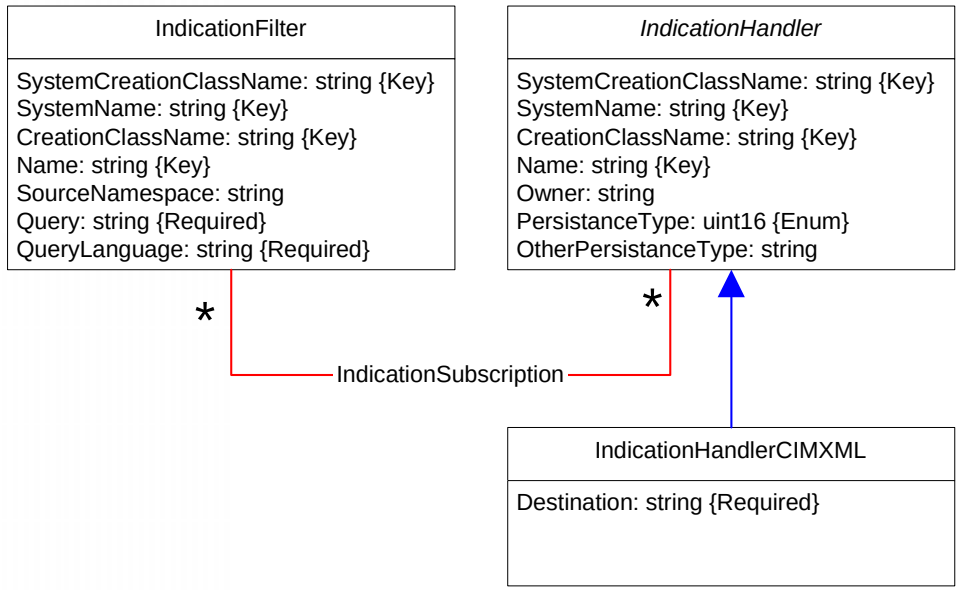


A CIM Client application **activates** a subscription by **creating an instance of CIM_IndicationSubscription**.

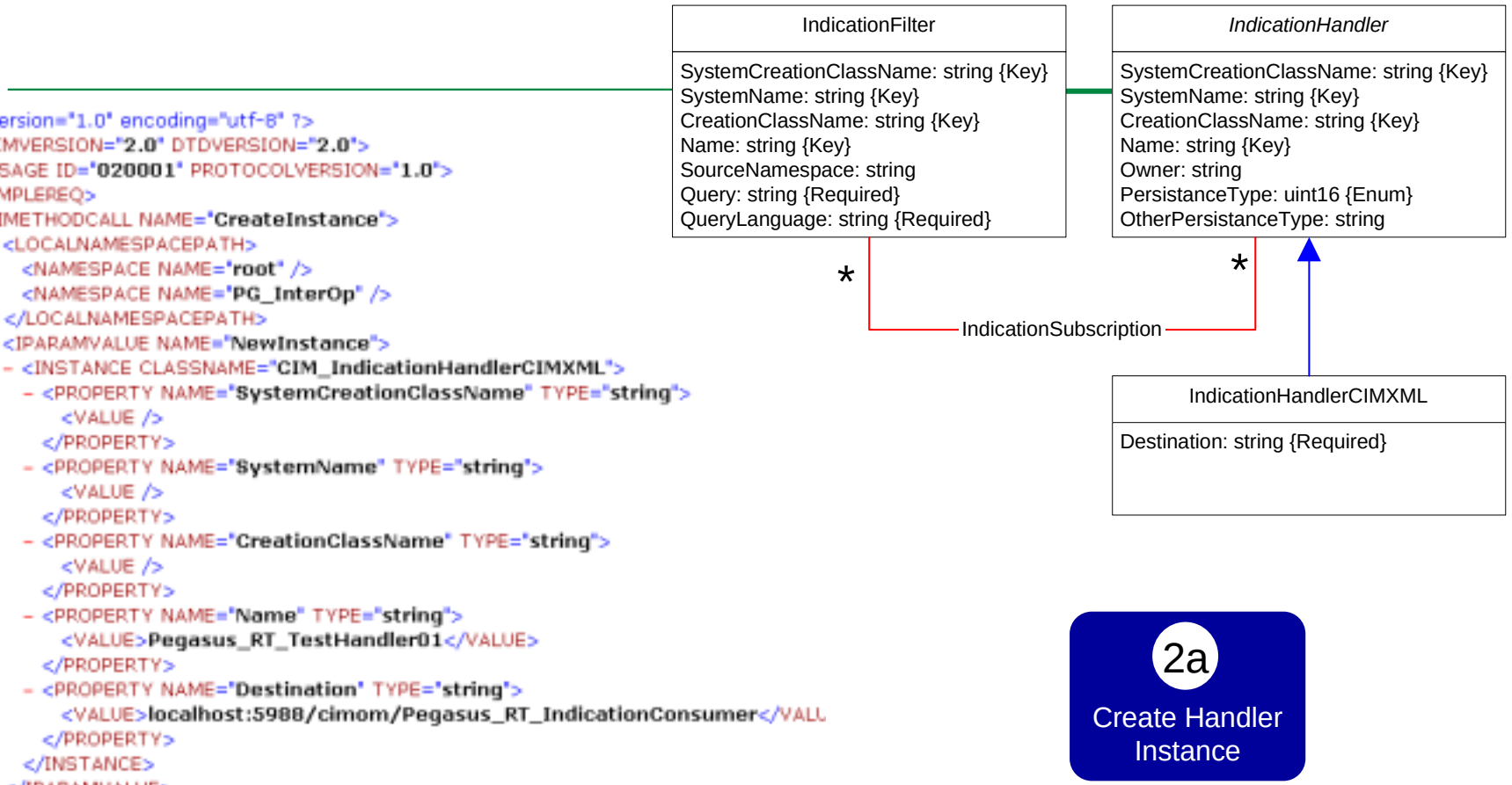
```
<?xml version="1.0" encoding="utf-8" ?>
- <CIM CIMVERSION="2.0" DTDVERSION="2.0">
  - <MESSAGE ID="010001" PROTOCOLVERSION="1.0">
    - <SIMPLEREQ>
      - <IMETHODCALL NAME="CreateInstance">
        - <LOCALNAMESPACEPATH>
            <NAMESPACE NAME="root" />
            <NAMESPACE NAME="PG_InterOp" />
          </LOCALNAMESPACEPATH>
        - <IPARAMVALUE NAME="NewInstance">
          - <INSTANCE CLASSNAME="CIM_IndicationFilter">
            - <PROPERTY NAME="SystemCreationClassName" TYPE="string">
                <VALUE />
              </PROPERTY>
            - <PROPERTY NAME="SystemName" TYPE="string">
                <VALUE />
              </PROPERTY>
            - <PROPERTY NAME="CreationClassName" TYPE="string">
                <VALUE />
              </PROPERTY>
            - <PROPERTY NAME="Name" TYPE="string">
                <VALUE>Pegasus_RT_TestFilter01</VALUE>
              </PROPERTY>
            - <PROPERTY NAME="Query" TYPE="string">
                <VALUE>SELECT * FROM RT_TestIndication</VALUE>
              </PROPERTY>
            - <PROPERTY NAME="QueryLanguage" TYPE="string">
                <VALUE>WQL1</VALUE>
              </PROPERTY>
            - <PROPERTY NAME="SourceNamespace" TYPE="string">
                <VALUE>root/SampleProvider</VALUE>
              </PROPERTY>
            </INSTANCE>
          </IPARAMVALUE>
        </IMETHODCALL>
      </SIMPLEREQ>
    </MESSAGE>
</CIM>
```

2b

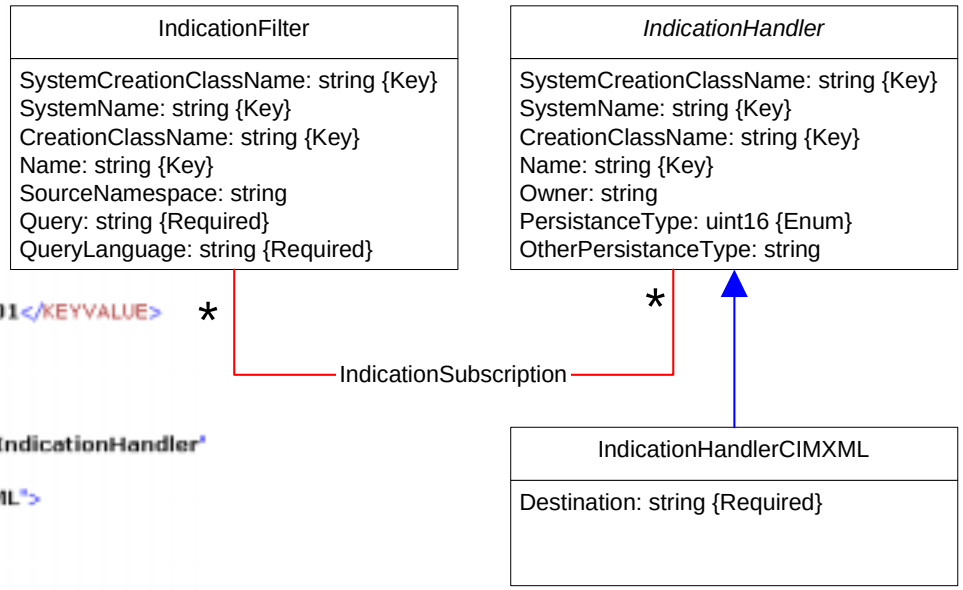
Create Filter Instance



```
<?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="CIM_IndicationHandlerCIMXML">
            - <PROPERTY NAME="SystemCreationClassName" TYPE="string">
                <VALUE />
              </PROPERTY>
            - <PROPERTY NAME="SystemName" TYPE="string">
                <VALUE />
              </PROPERTY>
            - <PROPERTY NAME="CreationClassName" TYPE="string">
                <VALUE />
              </PROPERTY>
            - <PROPERTY NAME="Name" TYPE="string">
                <VALUE>Pegasus_RT_TestHandler01</VALUE>
              </PROPERTY>
            - <PROPERTY NAME="Destination" TYPE="string">
                <VALUE>localhost:5988/cimom/Pegasus_RT_IndicationConsumer</VALU
              </PROPERTY>
            </INSTANCE>
          </IPARAMVALUE>
        </IMETHODCALL>
      </SIMPLEREQ>
    </MESSAGE>
  </CIM>
```



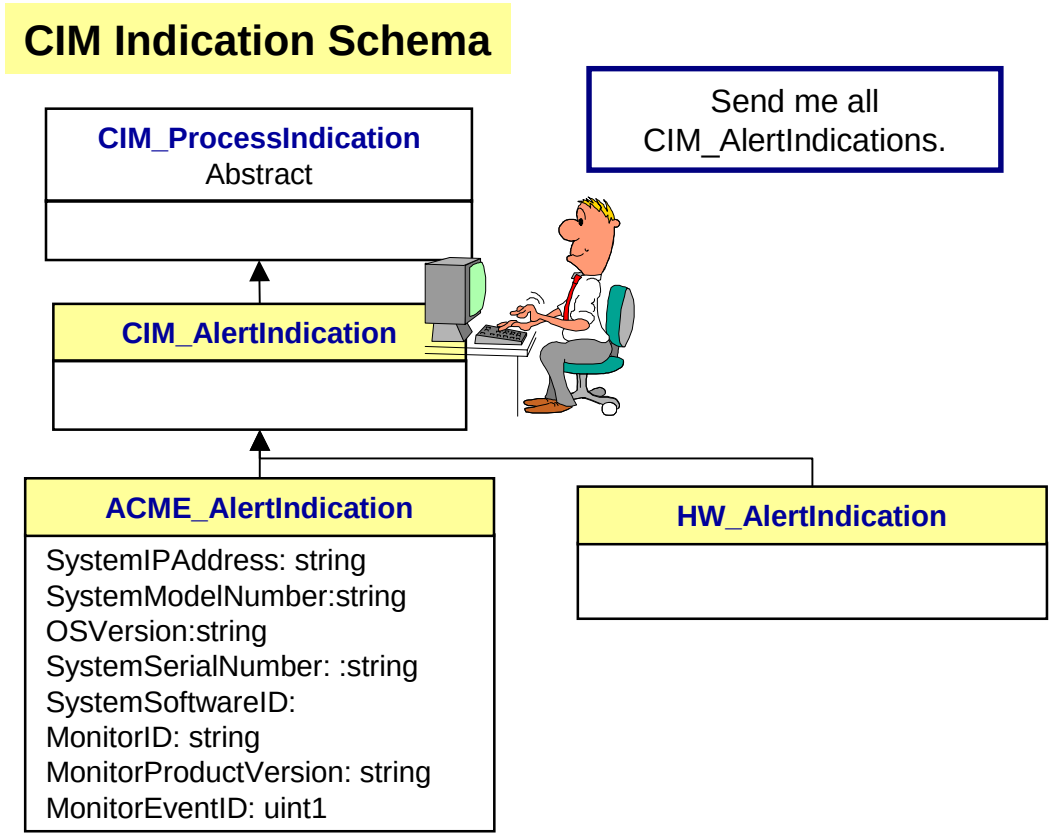
```
- <IMETHODCALL NAME="CreateInstance">
  - <LOCALNAMESPACEPATH>
      <NAMESPACE NAME="root" />
      <NAMESPACE NAME="PG_InterOp" />
    </LOCALNAMESPACEPATH>
  - <IPARAMVALUE NAME="NewInstance">
    - <INSTANCE CLASSNAME="CIM_IndicationSubscription">
      - <PROPERTY.REFERENCE NAME="Filter" REFERENCECLASS="CIM_IndicationFilter">
        - <VALUE.REFERENCE>
          - <INSTANCENAME CLASSNAME="CIM_IndicationFilter">
            - <KEYBINDING NAME="SystemCreationClassName">
                <KEYVALUE VALUETYPE="string" />
              </KEYBINDING>
            - <KEYBINDING NAME="SystemName">
                <KEYVALUE VALUETYPE="string" />
              </KEYBINDING>
            - <KEYBINDING NAME="CreationClassName">
                <KEYVALUE VALUETYPE="string" />
              </KEYBINDING>
            - <KEYBINDING NAME="Name">
                <KEYVALUE VALUETYPE="string">Pegasus_RT_TestFilter01</KEYVALUE>
              </KEYBINDING>
            </INSTANCENAME>
          </VALUE.REFERENCE>
        </PROPERTY.REFERENCE>
      - <PROPERTY.REFERENCE NAME="Handler" REFERENCECLASS="CIM_IndicationHandler"
        - <VALUE.REFERENCE>
          - <INSTANCENAME CLASSNAME="CIM_IndicationHandlerCIMXML">
            - <KEYBINDING NAME="SystemCreationClassName">
                <KEYVALUE VALUETYPE="string" />
              </KEYBINDING>
            - <KEYBINDING NAME="SystemName">
                <KEYVALUE VALUETYPE="string" />
              </KEYBINDING>
            - <KEYBINDING NAME="CreationClassName">
                <KEYVALUE VALUETYPE="string" />
              </KEYBINDING>
            - <KEYBINDING NAME="Name">
                <KEYVALUE VALUETYPE="string">Pegasus_RT_TestHandler01</KEYVALUE>
              </KEYBINDING>
            </INSTANCENAME>
          </VALUE.REFERENCE>
        </PROPERTY.REFERENCE>
```



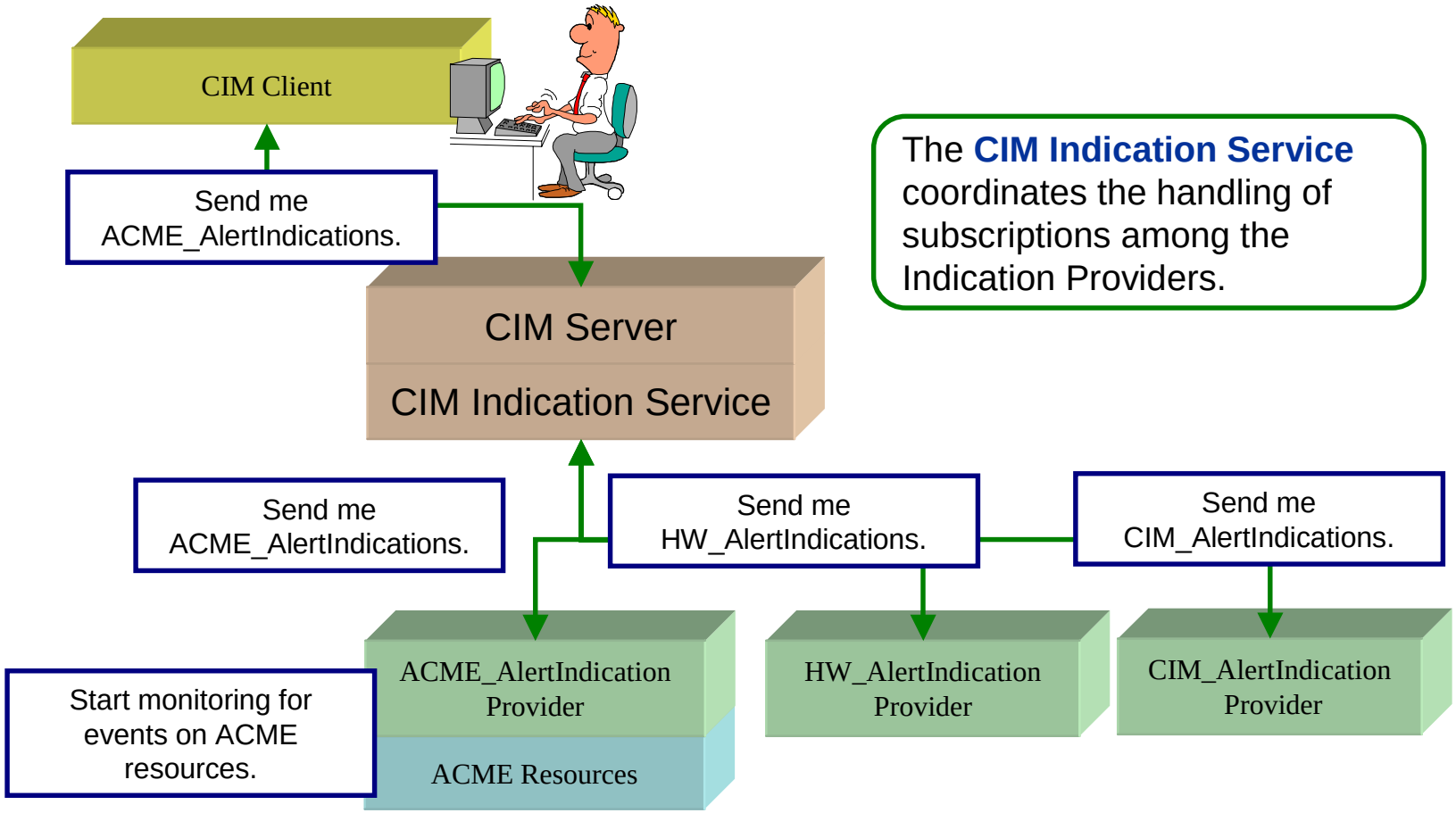
2c

Create Subscription Instance

Indication Service Role



Indication Service Role

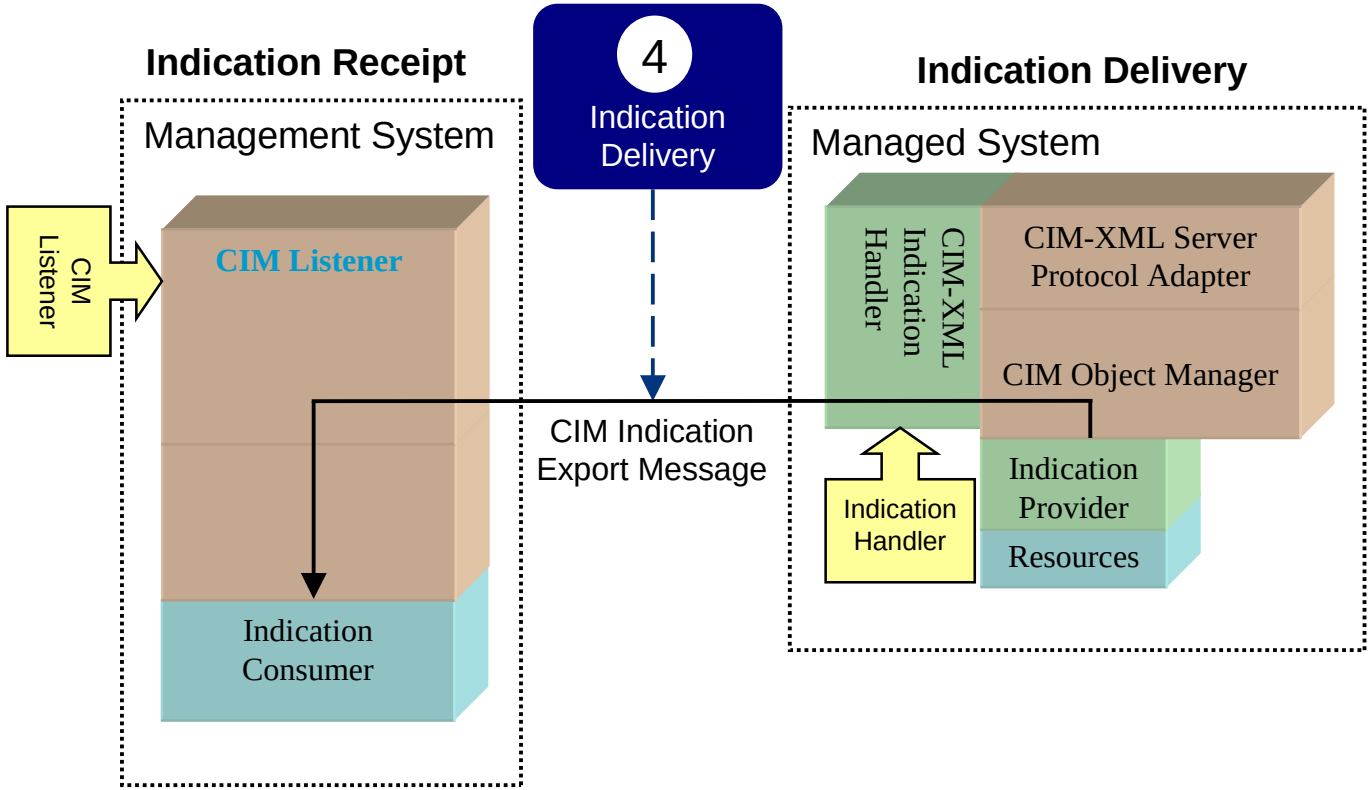


Module Content

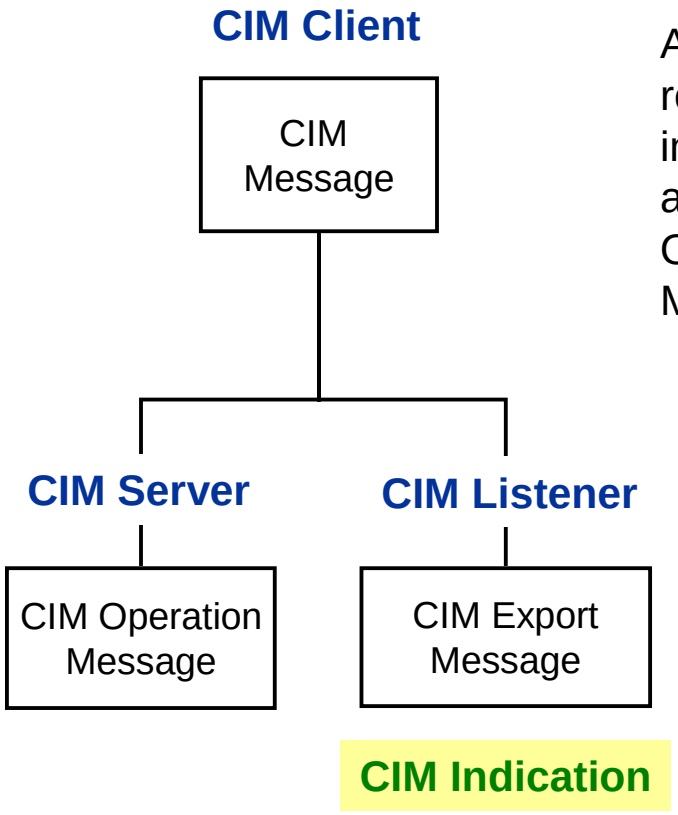
HP WBEM Services Indication Architecture

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

Indication Delivery



Indication Delivery

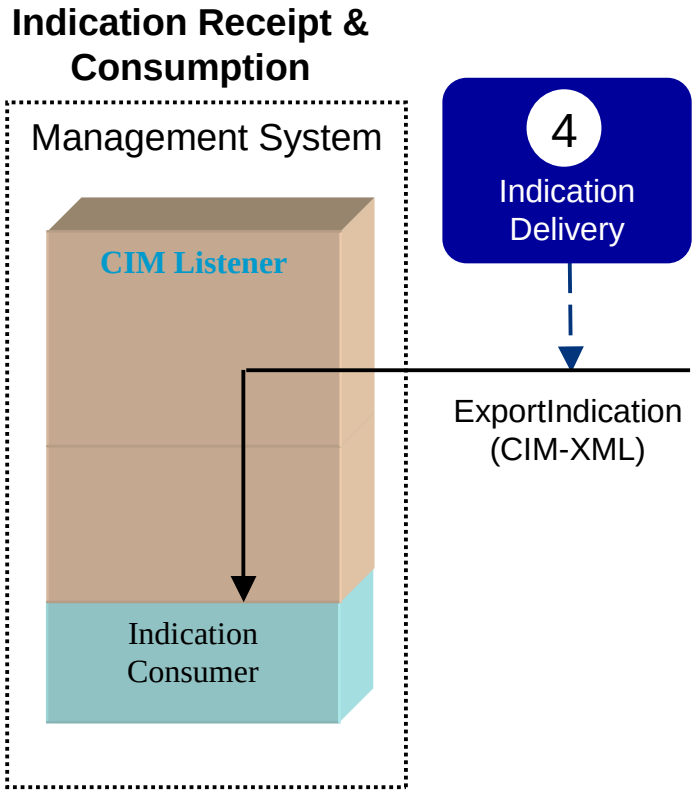


A **CIM Message** is a well-defined request or response data packet used to exchange information between CIM Applications. There are two types of CIM Messages, CIM Operation Messages and CIM Export Messages.

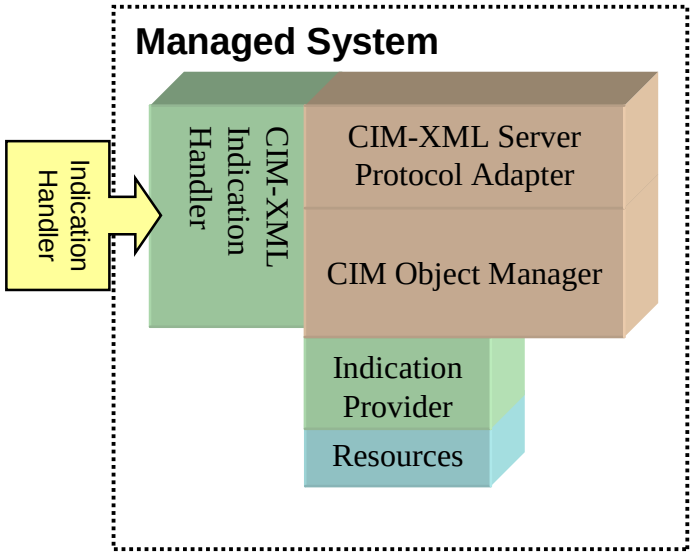
- A **CIM Operation Message** is a CIM Message used to invoke an operation on the target CIM namespace.
- A **CIM Export Message** is a CIM Message used to communicate information about a CIM namespace or element that is foreign to the target. A CIM Export Message is informational only and does not define an operation on the target CIM namespace or even imply the existence of a target namespace.

Indication Delivery

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.

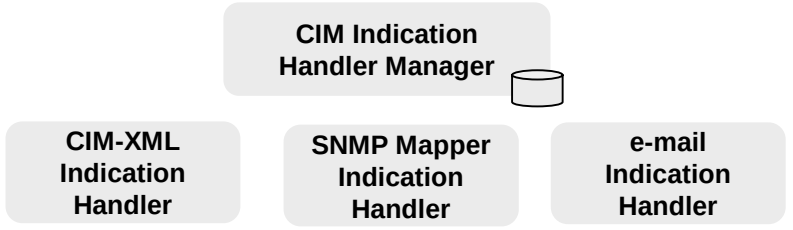


Indication Delivery

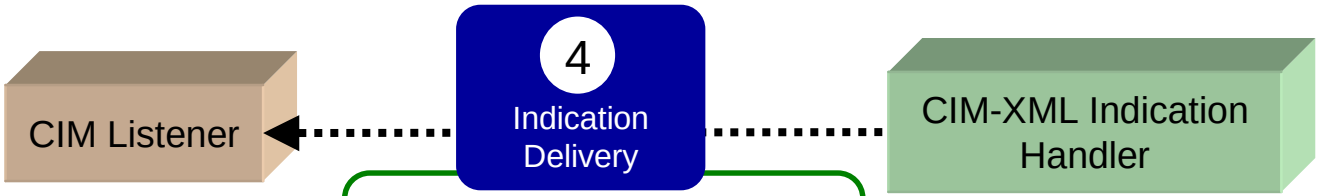


A **CIM Indication Handler** receives Indications, performs the mapping between the internal representation of a CIM Indication and the desired format and protocol, and sends the Indication to the designated target.

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



Indication Delivery



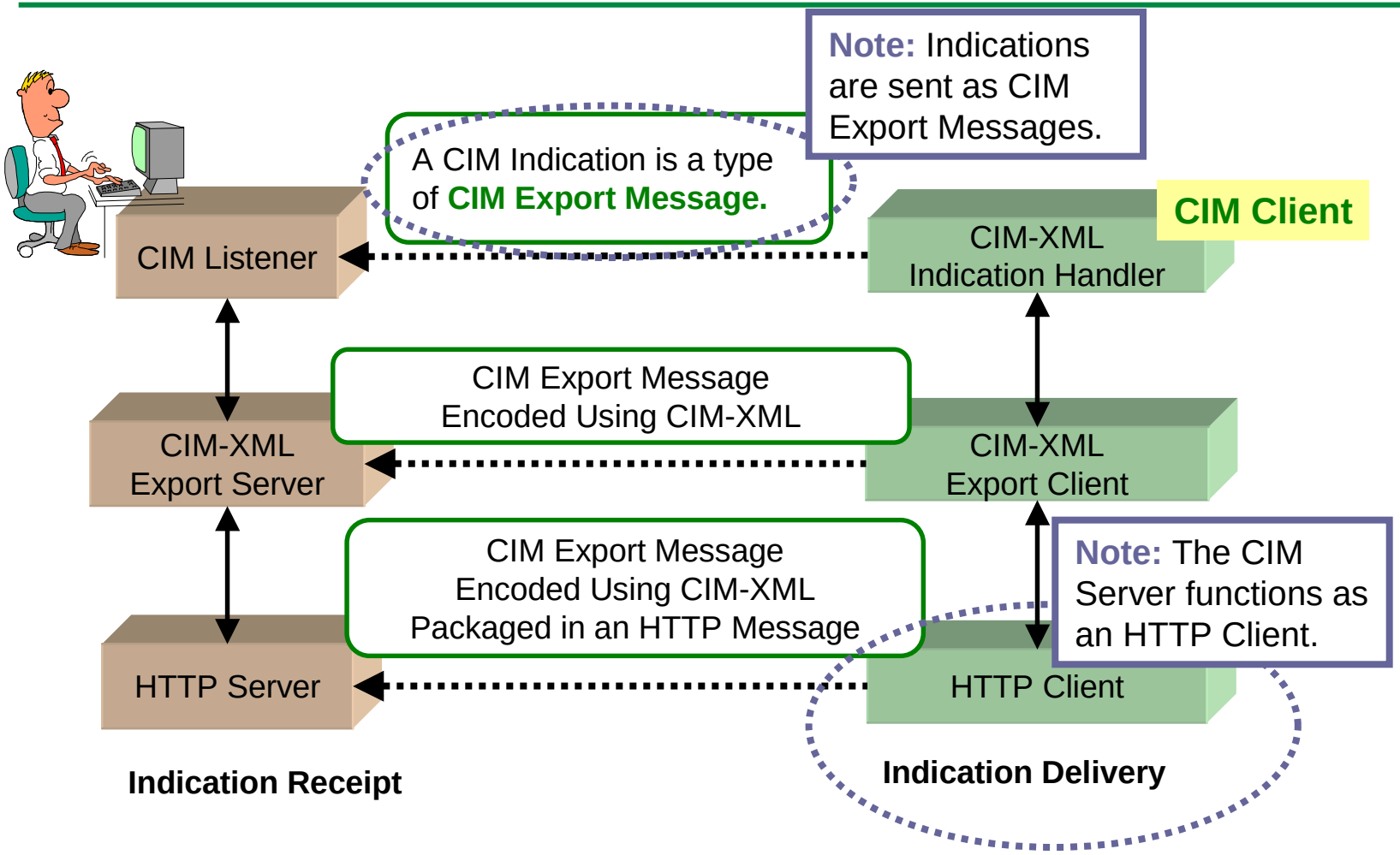
A CIM Indication is a type of **CIM Export Message.**

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



Indication Delivery

ExportIndication

void **ExportIndication** (
    [IN] <instance> NewIndication
)

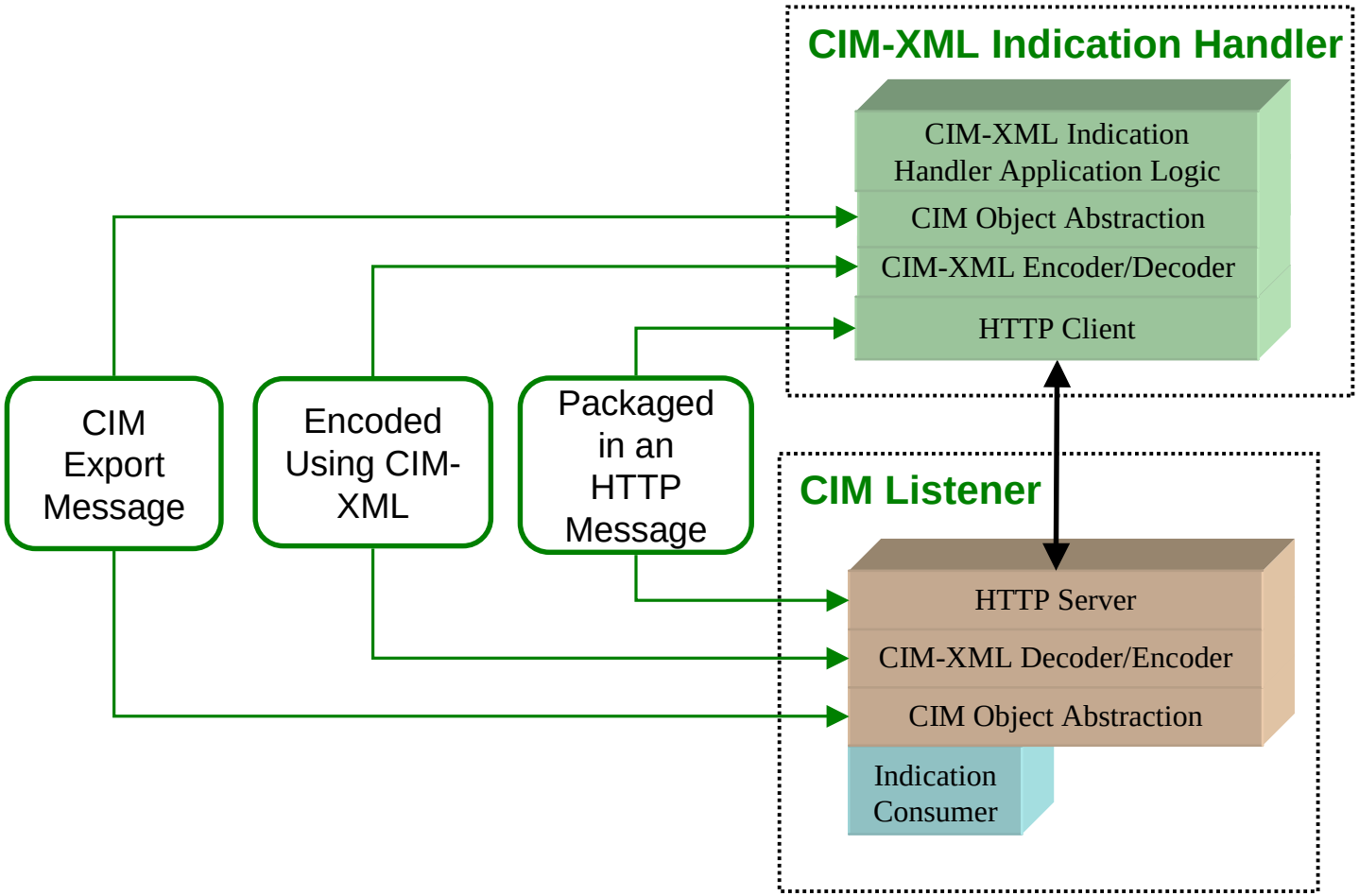
```
M-POST /cimlistener/browser HTTP/1.1
HOST: http://www.acme.com/
Content-Type: application/xml; charset="utf-8"
Content-Length: XXX
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=40
40-CIMExport: MethodRequest
40-CIMExportMethod: ExportIndication

<?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 NAME="IndicationTime" TYPE="datetime">
      <VALUE>20010515104354.000000:000</VALUE>
     </PROPERTY>
    </INSTANCE>
   </IPARAMVALUE>
  </EXPMETHODCALL>
 </SIMPLEEXPREQ>
</MESSAGE>
</CIM>
```

Export Request

Operation

Indication

CIM-XML Protocol Adapter

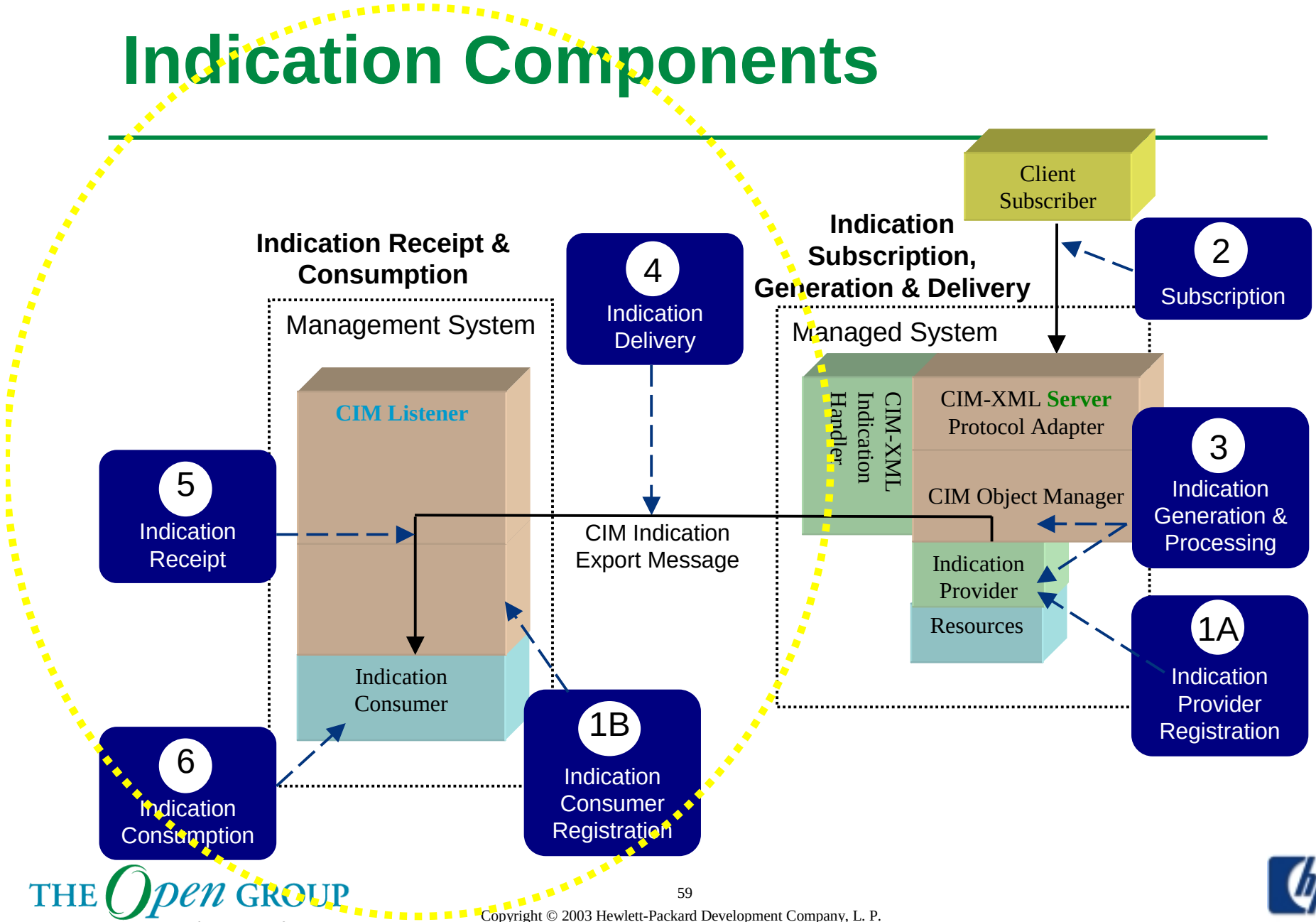


Module Content

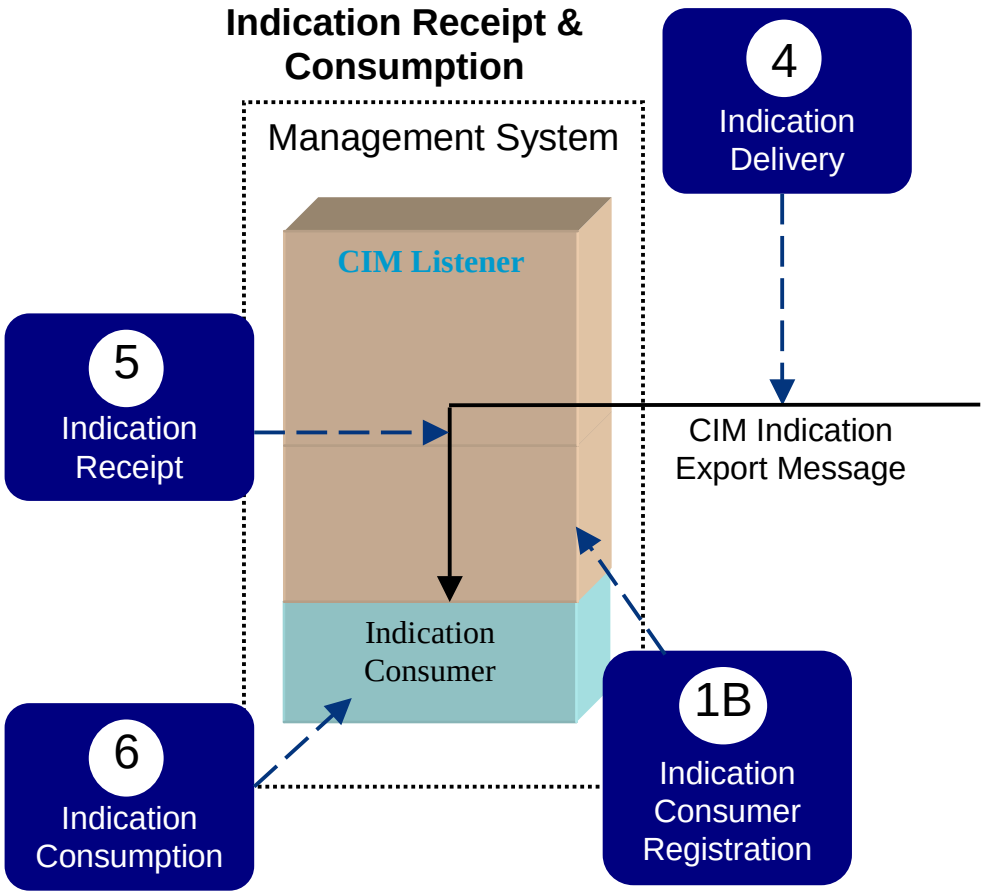
HP WBEM Services Indication Architecture

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

Indication Components



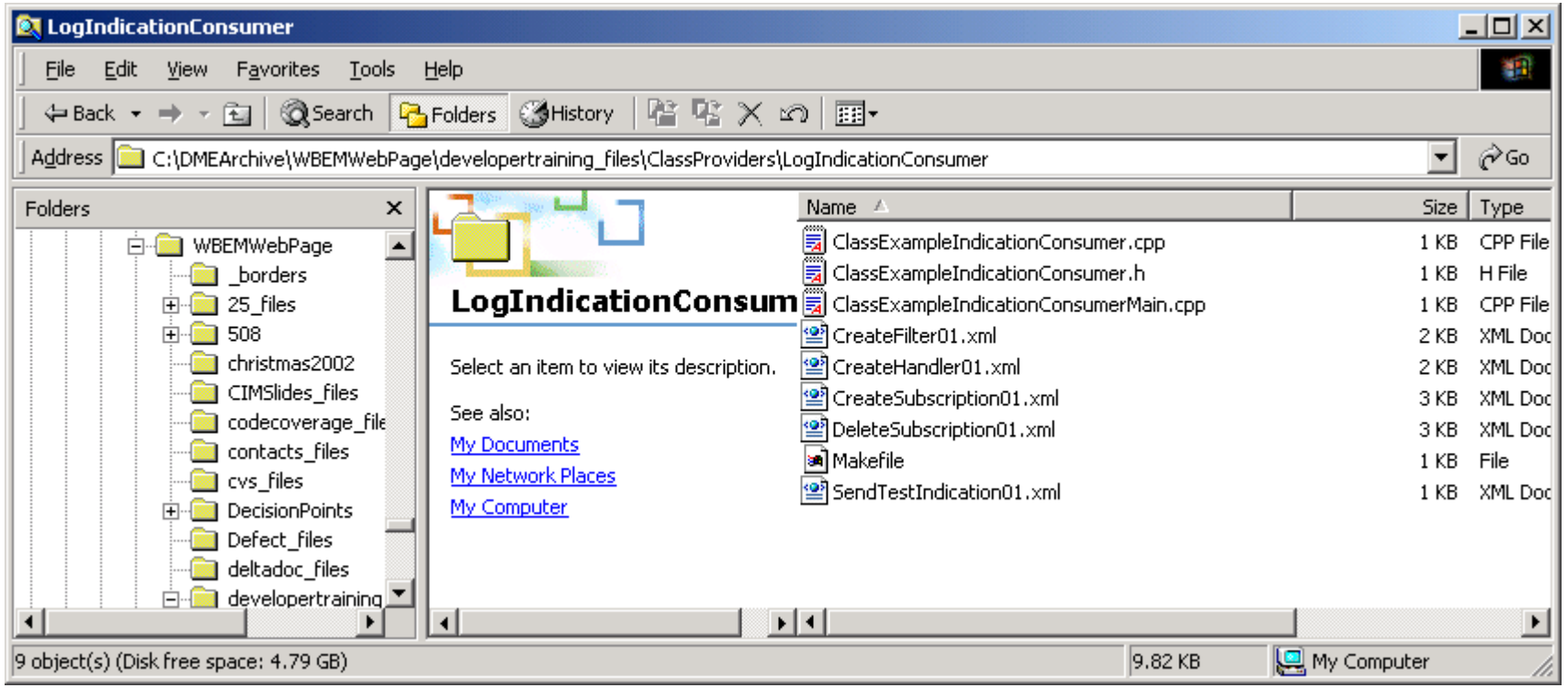
Indication Consumer



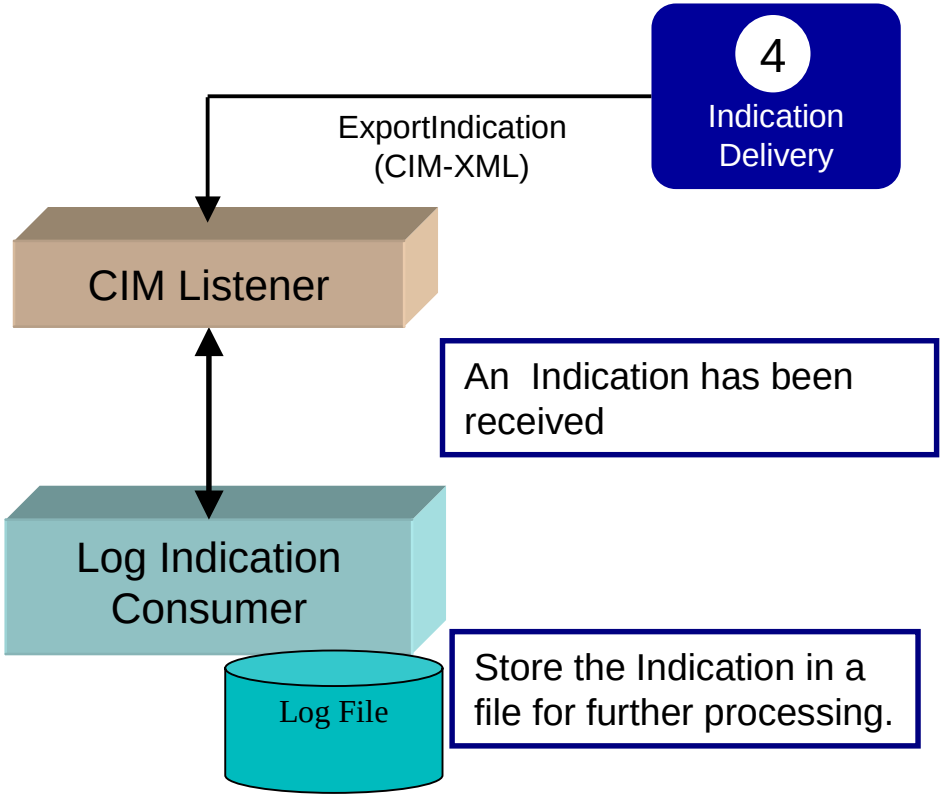
A **CIM Listener** receives CIM Exports requests, coordinates the distribution of requests among 1 or more Consumers and sends CIM Export responses.

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

Indication Consumer Example

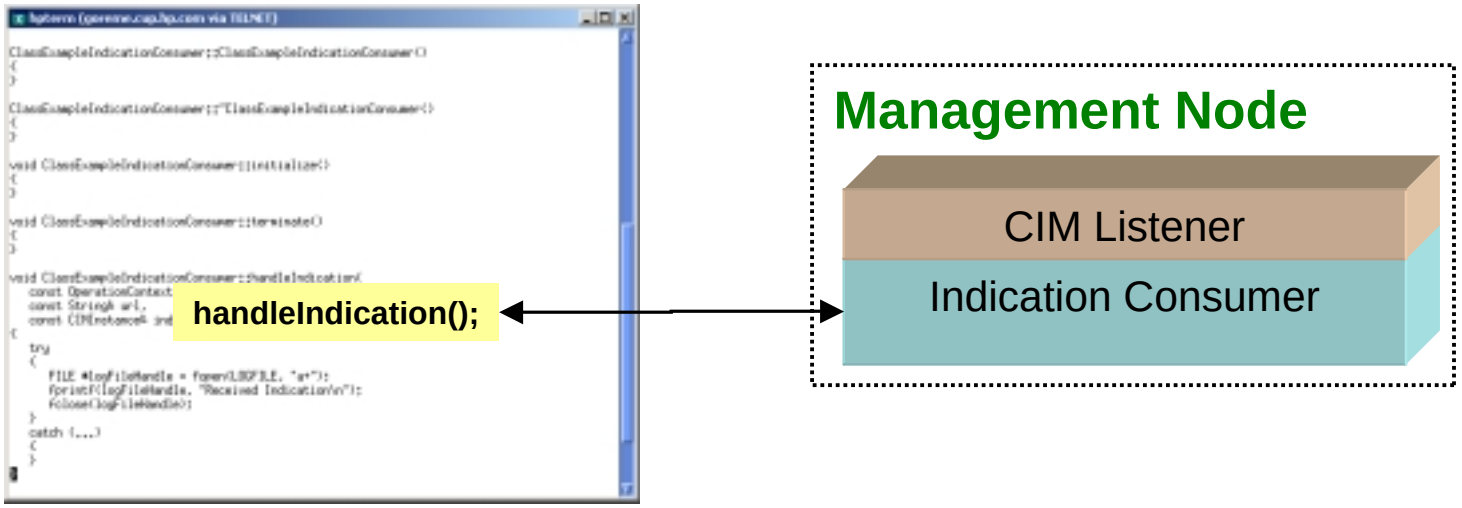


CIM Indication Consumer



CIM Indication Consumer

CIM Export	Implementation Owner
ExportIndication	**Indication Consumer**



ClassExampleIndicationConsumer.cpp - WordPad

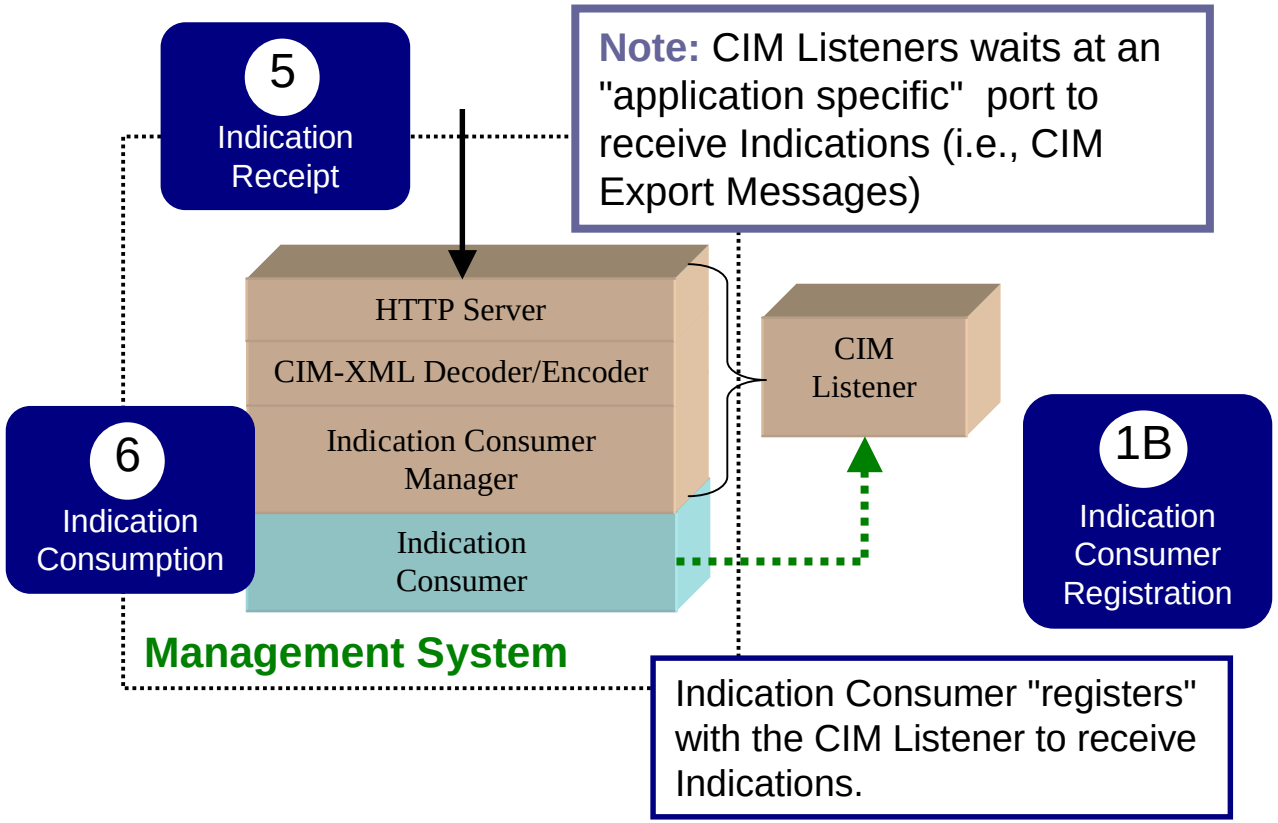
File Edit View Insert Format Help

```
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 (...)
   {
   }
}
```

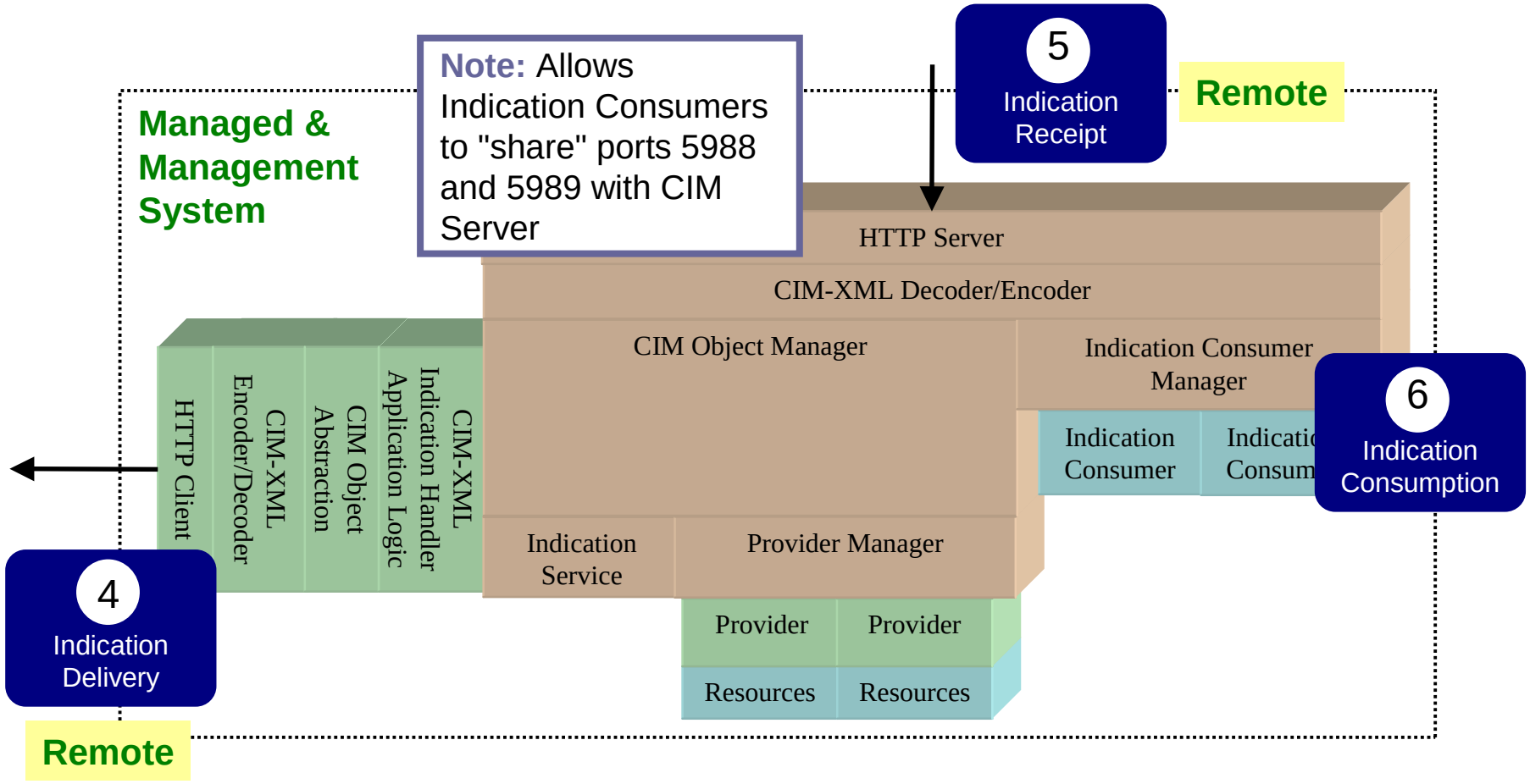
handleIndication

For Help, press F1

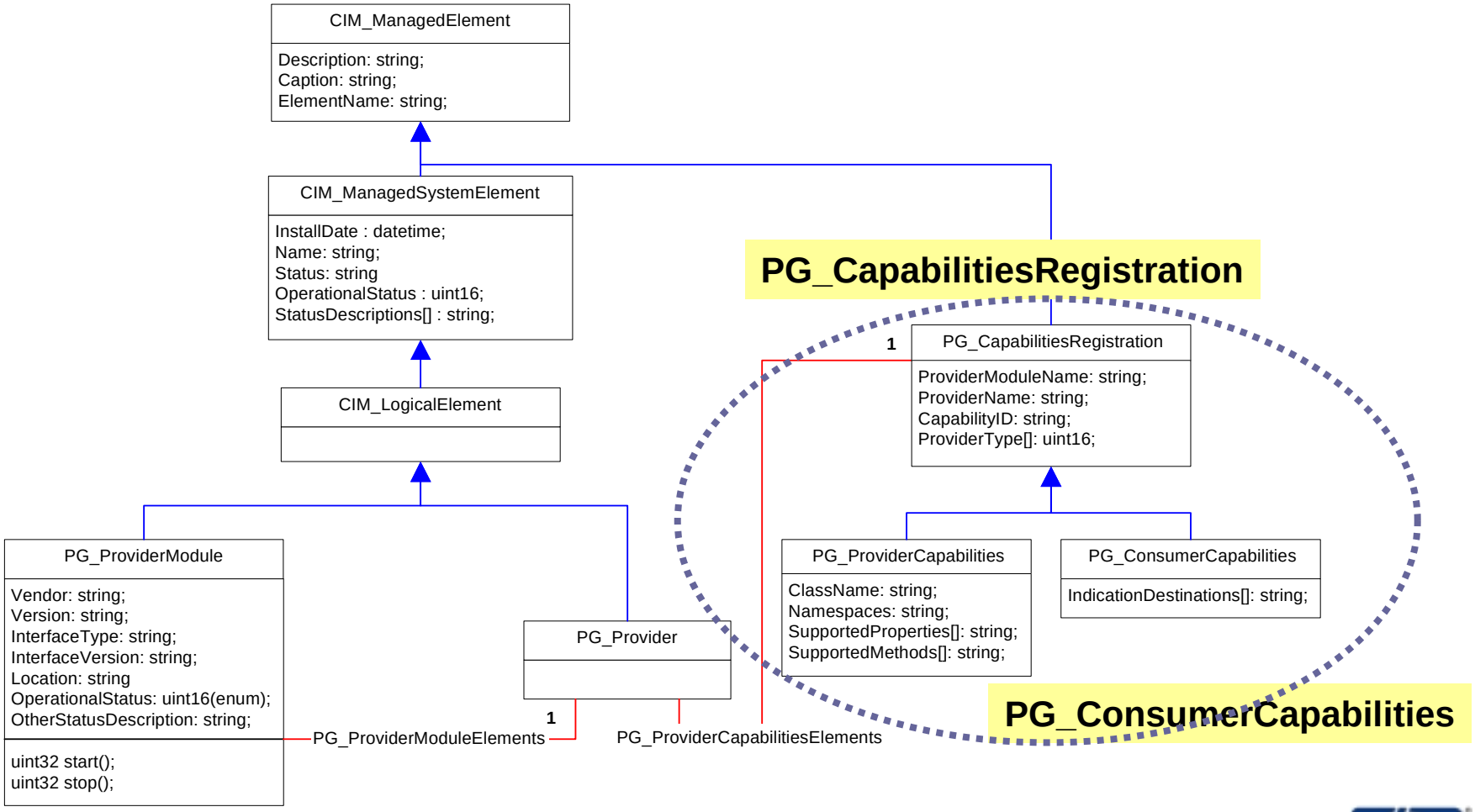
Standalone CIM Listener



CIM Listener with CIM Server

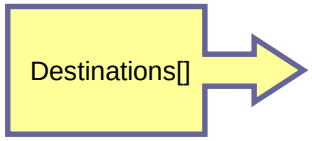


Provider Registration Schema



Provider Capabilities

ACME Indication Provider	
Set of Operations	ExportIndication



PG_ConsumerCapabilities

PG_ConsumerCapabilities
ProviderModuleName: string; ProviderName: string; CapabilityID: string; ProviderType[]: uint16; Destinations[]: string;

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

Description of Consumer Capabilities