Pegasus Enhancement Proposal (PEP)

PEP #: 062

Title: SNMP Mapper Indication Handler

Version: 1.1

Created: 08 May 2003

Authors: Yi Zhou, Hewlett-Packard

Status: Accepted

Version History:

Version	Date	Author	Change Description
1.0	08 May 2003	Yi Zhoui, Denise Eckstein	Initial Submission
1.1	21 May 2003	Yi Zhoui, Denise Eckstein	Incorporated comments from Architecture Team review to create ballot version: • Replaced definition of SNMPCommunityName with SNMPSecurityName. SNMPSecurityName is used to support the use of SNMPCommunityName in versions SNMPv1 and SNMPv2C and the use of User Name in SNMPv3. • Changed the name of the property TargetHostName to TargetHost. Modified the description to be more generic. • Added three new properties to PG_IndicationHandlerSNMPMapper: TargetHostFormat, OtherTargetHostFormat and SNMPEngineID. • Updated Figure with PG_IndicationHandler Schema. • Updated SNMPVersion information to include format type.

Abstract: This PEP proposes the addition of a new Indication Handler to OpenPegasus release 2.3. The SNMP Mapper Indication Handler is being proposed to give CIM Indication Provider developers a simple mechanism to support event notification delivery over SNMP.

Definition of the Problem

Issue

The Simple Network Management Protocol (SNMP) is a simple, extensible industry standard protocol for managing network devices. SNMP has been widely adopted as an interoperable framework for managing network devices and is a component of virtually all multi-platform management solutions deployed today.While the new DMTF CIM-XML WBEM Indication Standard offers many advantages over SNMP, product vendors will, for the foreseeable future, need to continue delivering management instrumentation solutions that interoperate with existing SNMP-based management solutions.

Pegasus currently does not support event notification delivery over SNMP.

Constraints

The following constraints have been identified for a solution that maps CIM Indications to SNMP Traps:

- **Operational Efficiency:** The solution must offers developers a reduced cost option for developing instrumentation that supports delivery of both SNMP Traps and CIM Indications.
- **Backward Compatibility:** The solution must support the development of instrumentation that interoperates with existing SNMP-based management solutions. In particular, the mapping definition and algorithm must be sufficiently flexible to allow the format of the generated traps to be be consistent with that expected by the SNMP Management Application.

Proposed Solution

Background

Please refer to the document titled PEP#062 - SNMP Mapper Backgrounder for a brief overivew of the DMTF CIM-XML WBEM and IETF SNMP alert mechanisms.

Solution Description

The proposed solution defines a new type of Indication Handler, the SNMP Mapper Indication Handler. The SNMP Mapper Indication Handler supports a simple mechanism for mapping CIM Indications to SNMP Traps. This mapping is accomplished by:

- Defining a new subclass of CIM_IndicationHandler, PG_IndicationHandlerSNMPMapper.
- Defining a mechanism that allows the definition of a CIM Indication class to be extended to include information on how to map an instance of CIM Indication class into an SNMP Trap.
- Developing an SNMP Mapper Indication Handler that performs the defined mapping and sends the created SNMP Trap to the desired destination.

> **Implementation Note:** The implementation of the SNMP Mapper Indication Handler depends on, for each supported platform, the availability of a runtime library that supports the construction and delivery of SNMP Traps.

PG_IndicationHandlerSNMPMapper

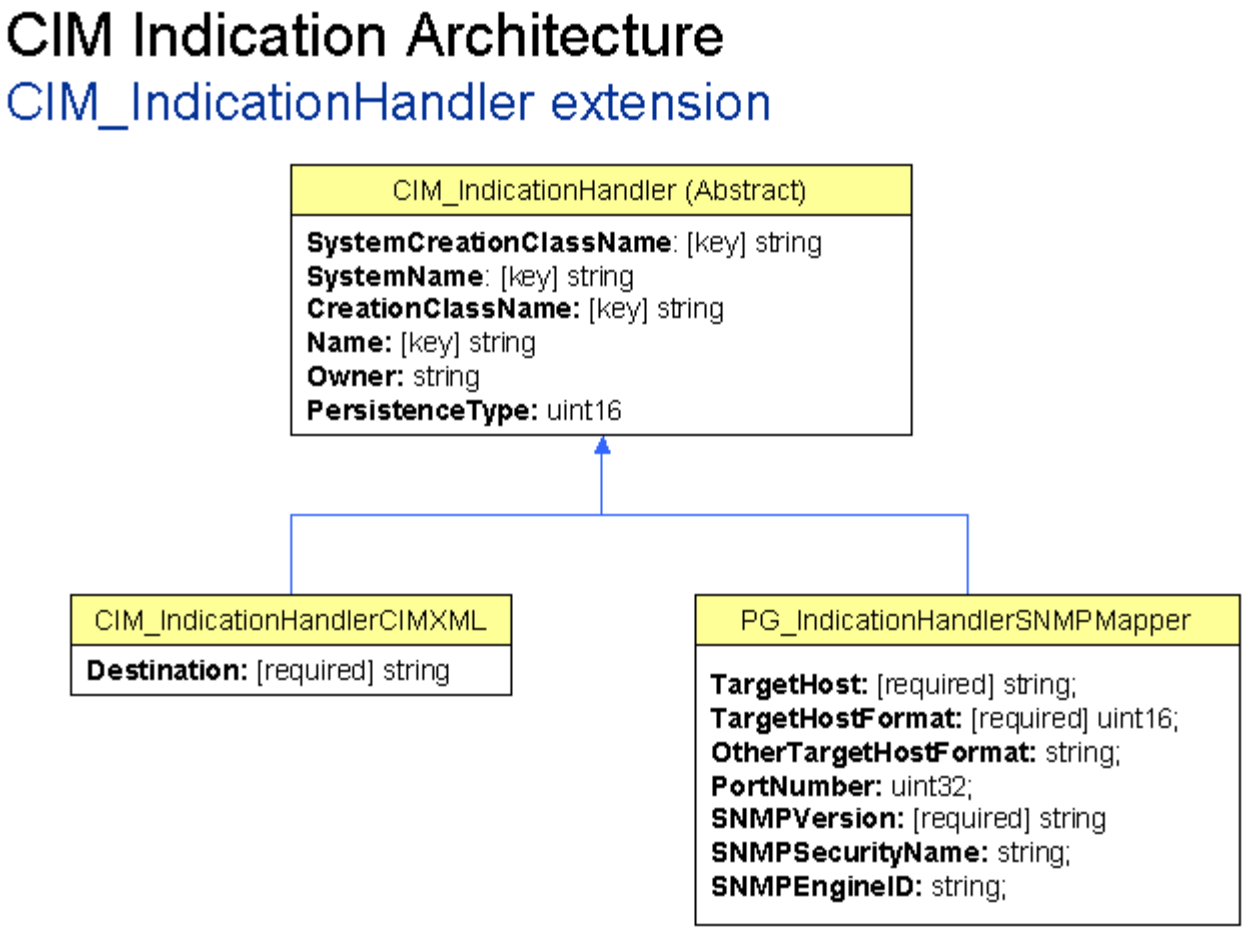
Defining a new CIM_IndicationHandler subclass, PG_IndicationHandlerSNMPMapper, allows an SNMP Trap Destination to be defined as the destination of a Subscription.

An instance of PG_IndicationHandlerSNMPMapper contains the following seven properties:

- The **TargetHost** property contains the trap/inform destination.
- The **TargetHostFormat** property describes the TargetHost format.
- The **OtherTargetHostFormat** property describes an "other" format type.
- The **PortNumber** property contains the UDP port number.
- The **SNMPSecurityName** property contains the name of the SNMP Community.
- The **SNMPVersion** property describes the desired SNMP protocol encoding.
- The **SNMPEngineID** property contains the SNMP EngineID of the Target Host.

```
//
================================================================
==
// PG_IndicationHandlerSNMPMapper
//
================================================================
==
   [Description (
     "PG_IndicationHandlerSNMPMapper describes the destination
for "
     "Indications to be delivered via SNMP trap/inform")]

class PG_IndicationHandlerSNMPMapper: CIM_IndicationHandler
{
    [Required, Description (
      "The address of the trap/inform destination.")]
    string TargetHost;

    [Required, Description (
      "An enumerated integer describing the format and "
      "interpretation of the TargetHost property."),
      ValueMap {"1", "2", "3", "4", "5..65535"},
      Values {"Other", "Host Name", "IPV4 Address",
              "IPV6 Address", "Pegasus Reserved"},
      ModelCorrespondence {
         "PG_IndicationHandlerSNMPMapper.OtherTargetHostFormat"}
]
    uint16 TargetHostFormat;

    [Description (
      "Describes the format when the value of TargetHostFormat "
      "is set to 1 (\"Other\")."),
      ModelCorrespondence {
         "PG_IndicationHandlerSNMPMapper.TargetHostFormat"} ]
    string OtherTargetHostFormat;

    [Description (
      "The UDP port number to send the trap/inform. "
      "The default is port 162.") ]
    uint32 PortNumber = 162;
```

```
    [Required, Description (
      "The SNMP version and format to use to send the "
      "Indication."),
      ValueMap {"2", "3", "4", "5", "6", "7..65535"},
      Values {"SNMPv1 Trap", "SNMPv2C Trap",
        "SNMPv2C Inform", "SNMPv3 Trap", "SNMPv3 Inform",
        "Pegasus Reserved"} ]
    uint16 SNMPVersion;

    [Description (
      "A human readable string that contains either "
      "an SNMPv1 or SNMPv2C community name or "
      "an SNMPv3 user name."),
      ModelCorrespondence {
         "PG_IndicationHandlerSNMPMapper.SNMPVersion"} ]
    string SNMPSecurityName;

    [Description (
      "The SNMP Engine ID used to create the SNMPv3 inform. "
      "The Engine ID of the Target Host is required when "
      "sending an SNMPV3 inform.") ]
    string SNMPEngineID;
};
```



MappingString Qualifier

The DMTF "Common Information Model (CIM) Specification", Version 2.2, defines a MappingStrings qualifier. The MappingStrings qualifier was defined as a mechanism to specify the mapping from CIM Objects to other types of management objects (e.g., DMI Attributes or SNMP variables).

Qualifier	Default	Applies To	Type	Meaning
MAPPINGSTRINGS	NULL	Class, Property, Association, Indication, Reference	STRING ARRAY	Mapping strings for one or more management data providers or agents.

The OpenPegasus SNMPMapper Indication Handler uses the value of this Qualifier to determine how to map an instance of a CIM Indication class into an SNMP Trap. When applied to the definition of a class, the MappingStrings value is interpreted as the value of the SNMP TrapOID.

The following is an example of a MappingStrings value for the Class HP_NSAParityError.

```
[Indication,
    MappingStrings {"OID.IETF | SNMP.1.3.6.1.4.1.11.2.23.20528"},
    Description ("Parity Error has been detected in parity
memory.")]
class HP_NSAParityError: HP_NSAAlertIndication
```

Implementation Note: This PEP proposes the addition of a new OperationContext container class, SNMPTrapOidContainer. This class will allow an Indication Provider to specify the TrapOID for an Indication, effectively overriding the TrapOID defined for the Indication class. If a TrapOID is not specified by the Indication Provider, the value of the MappingsString Qualifier for the Indication class will be used as the TrapOID.

When applied to a Property, the OID value of MappingStrings is interpreted as the OID for the property. This value is used to generate the OID for the value.

The following is an example of a MappingStrings value for the Property LocalDateTime.

```
[Description( "OperatingSystem's notion of the local date and
time of day"),
     MappingStrings {"OID.IETF | SNMP.1.3.6.1.2.1.25.1.2"}]
datetime LocalDateTime;
```

A MappingStrings value must contain the following two elements.

- The value of the Object Identifier Field for the CIM Property. The OID element MUST be formatted as follows:

  OID.IETF | <Content>, where <CONTENT> contains the OID

- The SNMP Data Type for the CIM Property. The DataType element MUST be formatted as follows:

  DataType.IETF | <DataTypeName>, where <DataTypeName> is one of the following values:

  - **Integer** (signed integers in the range of -2,147,483,648 to 2,147,483,647),
  - **OctetString** (an ordered sequence of zero to 65,535 octets), or
  - **OID** (an identifier defined according to the rules specified in ASN.1).

  Note: The **DataType.IETF** format is an extension to the DMTF Mapping String definition.

Implementation Note: The values of OID.IETF and DataType.IETF defined by the MappingStrings Qualifier are used by the SNMP Mapper Indication Handler to construct the corresponding VarBind value in the SNMP trap.

SNMP Mapper Indication Handler

The SNMP Mapper Indication Handler is responsible for mapping a CIM Indication into an SNMP Trap and sending the Trap to the desired destination.

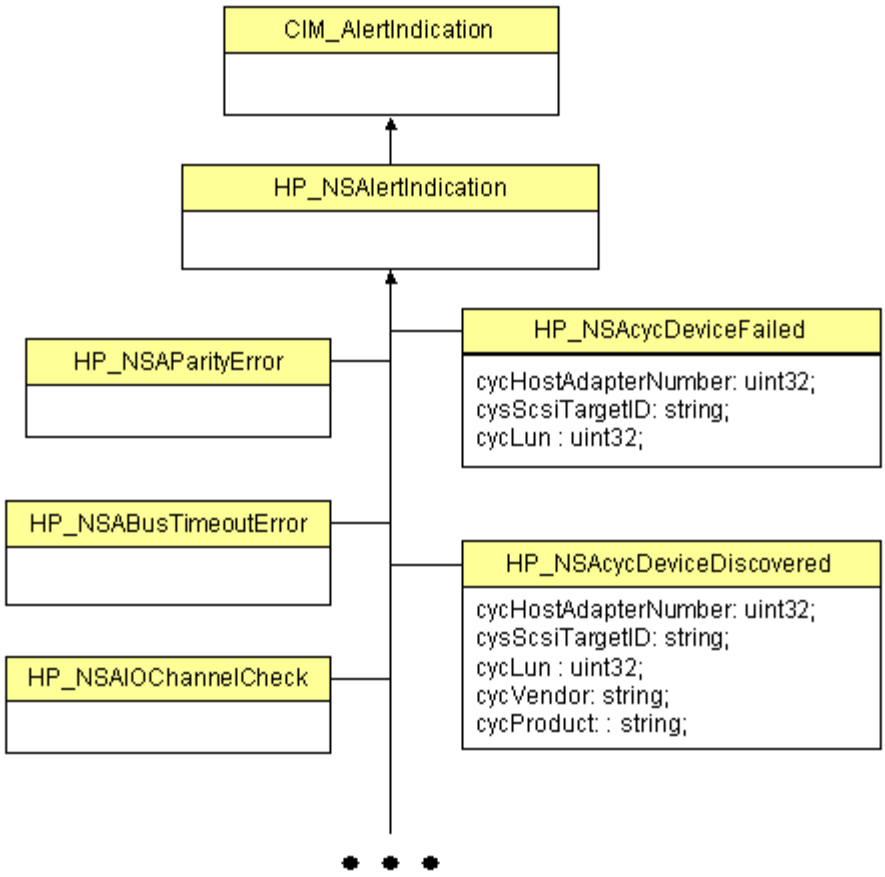
Implementation Note: The implementation of the SNMP Mapper Indication Handler depends on, for each supported platform, the availability of a runtime library that supports the construction and delivery of SNMP traps.

Solution Example

The following figure shows a segment from HP NetServer Assistant Extended Trap Definitions Management Information Base (MIB) for SNMP Network Management. This MIB will be used throughout the remainder of this section to illustrate the mapping of CIM Indications to SNMP traps.

```
hpnsaParityError TRAP-TYPE
  ENTERPRISE hpnsa
  DESCRIPTION     "Parity error has been detected in parity
memory"
  --#TYPE         "Parity error"
  --#SUMMARY      "%s"
  --#ARGUMENTS    {1}
  --#SEVERITY     MAJOR
  --#TIMEINDEX    100
  --#HELP         "nsa.hlp"
  --#HELPTAG      2042
  --#STATE        OPERATIONAL

  -- FIXED STRING  : NOT PRESENT
  -- CUSTOM STRING : PRESENT
  -- SEVERITY      : MAJOR
  ::= 20528

cycHostAdapterDiscovered TRAP-TYPE
  ENTERPRISE      cyclone
  VARIABLES {
     cycHostAdapterNumber,
     cycHostAdapterID,
     cycManagerID }
  DESCRIPTION "The HostAdapter# %d with HostAdapter Id %s
               and Manager Id %s is discovered"
  --#TYPE "Host Adapter Discovered"
  --#SUMMARY  "The HostAdapter# %d with HostAdapter Id %s
               and Manager Id %s is discovered"
  --#ARGUMENTS {0,1,2}
  --#SEVERITY INFORMATIONAL
  --#TIMEINDEX  100
  --#STATE OPERATIONAL
  --#HELP "scsismrt.hlp"
  --#HELPTAG 108
  ::= 108
```

The following figure shows one possible representation of the NSA Traps and CIM Indications.



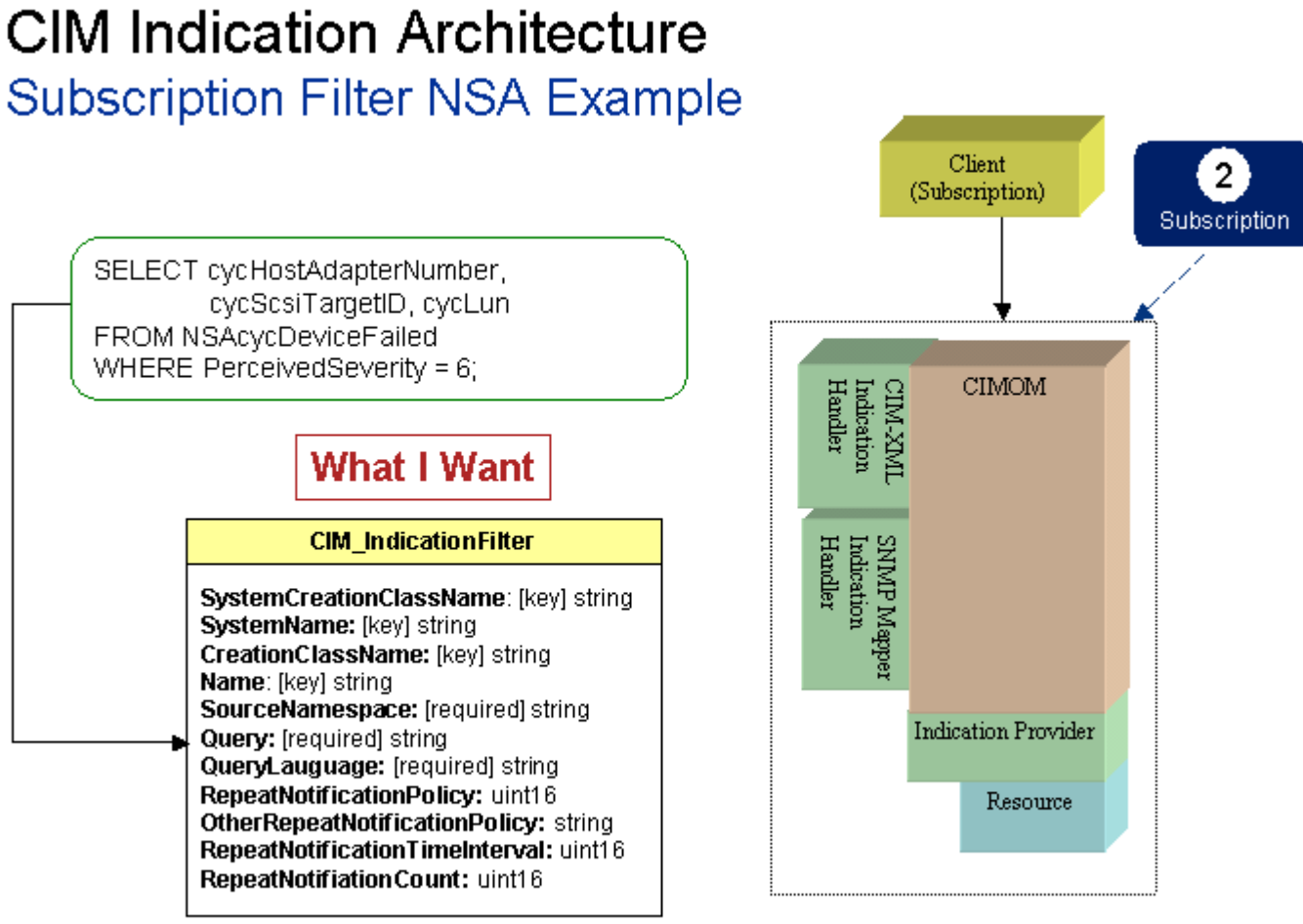
The MOF representation is shown below. Note that the MappingStrings Qualifier is use to defined the mapping between the CIM Indication definition and the SNMP MIB OIDs. The Trap OID that corresponds with Instances of a class (i.e., SNMP Trap) is specified using the MappingStrings Qualifier on the class definition.

```
//
==========================================================================
=
// HP NetServer Assistance Indication Definitions
//
==========================================================================
=

[Indication, Description ("HP NetServer Assistance Indications")]
class HP_NSAAlertIndication: CIM_AlertIndication {
};

[Indication,
    MappingStrings {"OID.IETF | SNMP.1.3.6.1.4.1.11.2.23.20528"},
```

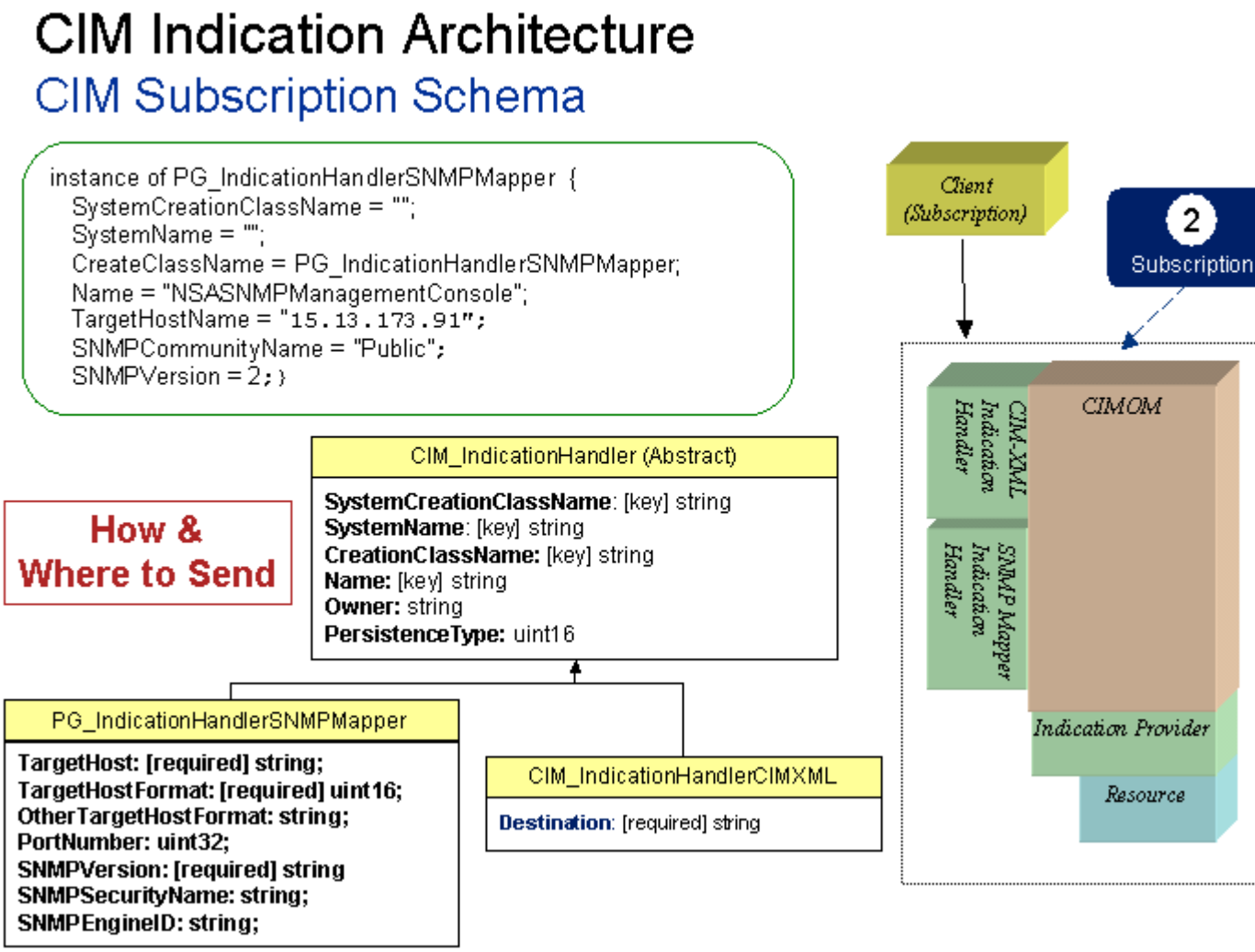
```
    Description ("Parity Error has been detected in parity
memory.")]
class HP_NSAParityError: HP_NSAAlertIndication {
};

[Indication,
    MappingStrings {"OID.IETF | SNMP.1.3.6.1.4.1.11.2.23.20544"},
    Description ("One of the bus masters caused a bus timeout.")]
class HP_NSABusTimeout: HP_NSAAlertIndication {
};

[Indication,
    MappingStrings {"OID.IETF | SNMP.1.3.6.1.4.1.11.2.23.20560"},
    Description ("I/O channel check activated by device in EISA "
                 "slot caused an error.")]
class HP_NSAIOChannelCheck : HP_NSAAlertIndication {
};

[Indication,
    MappingStrings {"OID.IETF |
SNMP.1.3.6.1.4.1.795.2.5.9000.112"},
    Description ("The Host Adapater, cycHostAdapterNumber, with "
        "TargetId and LunId has failed.")]
class HP_NSAcycDeviceFailed : HP_NSAAlertIndication {
    [Description ("The unique Host Adapter number"),
        MappingStrings {"OID.IETF |
SNMP.1.3.6.1.4.1.795.2.5.9000.9003",
            "DataType.IETF | Integer"}]
    uint32 cycHostAdapterNumber;
    [Description ("The SCSI Target ID"),
        MappingStrings {"OID.IETF |
SNMP.1.3.6.1.4.1.795.2.5.9000.9009",
            "DataType.IETF | Integer"}]
    uint32 cycScsiTargetID;
    [Description ("The LUN of the device ID"),
        MappingStrings {"OID.IETF |
SNMP.1.3.6.1.4.1.795.2.5.9000.9010",
            "DataType.IETF | Integer"}]
    uint32 cycLun;
};

[Indication,
    MappingStrings {"OID.IETF |
SNMP.1.3.6.1.4.1.795.2.5.9000.113"},
    Description ("The Host Adapater, cycHostAdapterNumber, with "
        "TargetId and LunId has been discovered.") ]
class HP_NSAcycDeviceDiscovered : HP_NSAAlertIndication {
    [Description ("The unique Host Adapter number"),
        MappingStrings {"OID.IETF |
SNMP.1.3.6.1.4.1.795.2.5.9000.9003",
            "DataType.IETF | Integer"}]
    uint32 cycHostAdapterNumber;
    [Description ("The SCSI Target ID"),
        MappingStrings {"OID.IETF |
SNMP.1.3.6.1.4.1.795.2.5.9000.9009",
            "DataType.IETF | Integer"}]
    uint32 cycScsiTargetID;
```

```
    [Description ("The LUN of the device ID"),
        MappingStrings {"OID.IETF |
SNMP.1.3.6.1.4.1.795.2.5.9000.9010",
            "DataType.IETF | Integer"}]
    uint32 cycLun;
    [Description ("This indicates the Name of the Vendor."),
        MappingStrings {"OID.IETF |
SNMP.1.3.6.1.4.1.795.2.5.9000.9004",
            "DataType.IETF | Octet"}]
    String cycVendor;
    [Description ("This indicates the Name of the Product."),
        MappingStrings {"OID.IETF |
SNMP.1.3.6.1.4.1.795.2.5.9000.9005",
            "DataType.IETF | Octet"}]
    String cycProduct;
};
```

A sample filter condition for the NSA example is shown below. It is important to note that the language used to specify the filter condition is “delivery mechanism” independent.



Creating an Instance of the appropriate CIM_IndicationHandler subclass specifies the delivery protocol and destination.



A sample a CIM Indication and its associated SNMP Trap.

```
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=53
53-CIMExport: MethodRequest
53-CIMExportMethod: ExportIndication
<?xml version="1.0" encoding="utf-8"?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="53000" PROTOCOLVERSION="1.0">
    <SIMPLEEXPREQ>
      <EXPMETHODCALL NAME="ExportIndication">
        <EXPPARAMVALUE NAME="NewIndication">
         <INSTANCE CLASSNAME="HP_NSAcycDeviceFailed">
            <PROPERTY NAME="cycHostAdapterNumber" TYPE="uint32">
```

```
              <VALUE>3</VALUE>
             </PROPERTY>
             <PROPERTY NAME="cycScsiTargetID" TYPE="uint32">
              <VALUE>4</VALUE>
             </PROPERTY>
             <PROPERTY NAME="cycLun" TYPE="uint32">
              <VALUE>5</VALUE>
             </PROPERTY>
           </INSTANCE>
         </EXPPARAMVALUE>
       </EXPMETHODCALL>
     </SIMPLEEXPREQ>
  </MESSAGE>
</CIM>
```

```
SNMP: --- Simple Network Management Protocol ---
SNMP:
SNMP: Version = 0
SNMP: Community = public
SNMP: Command = Trap
SNMP: Enterprise = {1.3.6.1.4.1.795.2.5.9000.112}
SNMP: Network address = [XXX.XXX.XXX.XXX]
SNMP: Generic trap = 3 (Link up)
SNMP: Specific trap = 0
SNMP: Time ticks = 797
SNMP: Object = {1.3.6.1.4.1.795.2.5.9000.9003.0}
(cycHostAdapterNumber.0)
SNMP: Value = 3
SNMP: Object = {1.3.6.1.4.1.795.2.5.9000.9009.0}
(cycScsiTargetID.0)
SNMP: Value = 4
SNMP: Object = {1.3.6.1.4.1.795.2.5.9000.9010.0) (cycLun.0)
SNMP: Value = 5
```

Work Items

There are some works already done in the current Pegasus. The following is a list of the work items need to be done for this PEP:

- **Add Support for the SnmpTrapOidContainer class:** Create new OperationContext container class SnmpTrapOidContainer. Modify OperationResponseHandler, IndicationService, CIMHandler, and snmpIndicationHandler to use this class.

  Implementation Note:

  To add SNMPTrapOID Container class, the interface between provider and indication service does not need to be changed, they are communicate by using message CIMProcessIndicationRequestMessage, so this message need to be changed by adding snmpTrapOid. Also, IndicationService and handlerService are communicate by using message CIMHandleIndicationRequestMessage, so this message need to be

changed by adding snmpTrapOid too.

The interface between HandlerService and Handler need to be changed:

- **Old interface:** handleIndication(CIMInstance& indicationHandlerInstance, CIMInstance& indicationInstance, String nameSpace);
- **New Interface:** handleIndication(const OperationContext & context, CIMInstance& indicationHandlerInstance, CIMInstance& indicationInstance, String nameSpace);

- **Performance Improvements:** In the current implementation, the SNMP Mapper Indication Handler calls the Repository to get the class definition every time an Indication is received. As part of this PEP we will consider adding a hash table to improve performance.

- **Defects Fixes:** This PEP proposes fixing the following class of defects in the existing implementation.
  - Fix compile problem for snmpDeliverTrap_emanate.cpp.
  - Add port of destination information into snmpIndicationHandler.
  - Change all the cout error message, either write them to the log or trace file.

- **Add new test cases:** Tests will be enhanced to increase code coverage.

Schedule

Action	Planned	Actual	Comment
PEP Submitted	09 May 2003	08 May 2003	
PEP Reviewed	16 May 2003		
PEP Approved	30 May 2003		
Code Committed	01 August 2003		

Appendix A

This entire section has been extracted directly from RFC 2576, "Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework" (http://www.ietf.org/rfc/rfc2576.txt).

This section describes how parameters used for generating notifications are translated between the format used for SNMPv1 notification protocol operations and the format used for

SNMPv2 notification protocol operations. The parameters used to generate a notification are called 'notification parameters'. The of parameters used for SNMPv1 notification protocol operations is referred to in this document as 'SNMPv1 notification parameters'. The format of parameters used for SNMPv2 notification protocol operations is referred to in this document as 'SNMPv2 notification parameters'.

The situations where notification parameters MUST be translated are:

- When an entity generates a set of notification parameters in a particular format, and the configuration of the entity indicates that the notification must be sent using an SNMP message version that requires the other format for notification parameters.
- When a proxy receives a notification that was sent using an SNMP message version that requires one format of notification parameters, and must forward the notification using an SNMP message version that requires the other format of notification parameters.

In addition, it MAY be desirable to translate notification parameters in a notification receiver application in order to present notifications to the end user in a consistent format.

Note that for the purposes of this section, the set of notification parameters is independent of whether the notification is to be sent as a trap or an inform.

SNMPv1 notification parameters consist of:

- An enterprise parameter (OBJECT IDENTIFIER).
- An agent-addr parameter (NetworkAddress).
- A generic-trap parameter (INTEGER).
- A specific-trap parameter (INTEGER).
- A time-stamp parameter (TimeTicks).
- A list of variable-bindings (VarBindList).

SNMPv2 notification parameters consist of:

- A sysUpTime parameter (TimeTicks). This appears in the first variable-binding in an SNMPv2-Trap-PDU or InformRequest-PDU.
- An snmpTrapOID parameter (OBJECT IDENTIFIER). This appears the second variable-binding in an SNMPv2-Trap-PDU or InformRequest-PDU.
- A list of variable-bindings (VarBindList). This refers to all but the first two variable-bindings in an SNMPv2-Trap-PDU or InformRequest-PDU

Translating SNMPv1 Notification Parameters to SNMPv2 Notification Parameters

The following procedure describes how to translate SNMPv1 notification parameters into SNMPv2 notification parameters:

- The SNMPv2 sysUpTime parameter SHALL be taken directly from the SNMPv1 time-stamp parameter.

- If the SNMPv1 generic-trap parameter is 'enterpriseSpecific(6)', the SNMPv2 snmpTrapOID parameter SHALL be the concatentation of the SNMPv1 enterprise parameter and two additional sub- identifiers, '0', and the SNMPv1 specific-trap parameter.

- If the SNMPv1 generic-trap parameter is not 'enterpriseSpecific(6)', the SNMPv2 snmpTrapOID parameter SHALL be the corresponding trap as defined in section 2 of RFC1907 draft:
-
- `generic-trap parameter  snmpTrapOID.0`
- `=======================  =============`
- `0  1.3.6.1.6.3.1.1.5.1 (coldStart)`
- `1  1.3.6.1.6.3.1.1.5.2 (warmStart)`
- `2  1.3.6.1.6.3.1.1.5.3 (linkDown)`
- `3  1.3.6.1.6.3.1.1.5.4 (linkUp)`
- `4  1.3.6.1.6.3.1.1.5.5 (authenticationFailure)`
- `5  1.3.6.1.6.3.1.1.5.6 (egpNeighborLoss)`

- The SNMPv2 variable-bindings SHALL be the SNMPv1 variable- bindings. In addition, if the translation is being performed by a proxy in order to forward a received trap, three additional variable-bindings will be appended, if these three additional variable-bindings do not already exist in the SNMPv1 variable- bindings. The name portion of the first additional variable binding SHALL contain snmpTrapAddress.0, and the value SHALL contain the SNMPv1 agent-addr parameter. The name portion of the second additional variable binding SHALL contain snmpTrapCommunity.0, and the value SHALL contain the value of the community-string field from the received SNMPv1 message which contained the SNMPv1 Trap-PDU. The name portion of the third additional variable binding SHALL contain snmpTrapEnterprise.0, and the value SHALL be the SNMPv1 enterprise parameter.

Translating SNMPv2 Notification Parameters to SNMPv1 Notification Parameters

The following procedure describes how to translate SNMPv2 notification parameters into SNMPv1 notification parameters:

- The SNMPv1 enterprise parameter SHALL be determined as follows:
    - If the SNMPv2 snmpTrapOID parameter is one of the standard traps as defined in RFC1907, then the SNMPv1 enterprise parameter SHALL be set to the value of the variable-binding in the SNMPv2 variable-bindings whose name is snmpTrapEnterprise, if that variable-binding exists. If it does not exist, the SNMPv1 enterprise parameter SHALL be set to the value ' snmpTraps' as defined in RFC1907.
    - If the SNMPv2 snmpTrapOID parameter is not one of the standard traps as defined in RFC1907, then the SNMPv1 enterprise parameter SHALL be determined from the SNMPv2 snmpTrapOID parameter as follows:
    - If the next-to-last sub-identifier of the snmpTrapOID is zero, then the SNMPv1 enterprise SHALL be the SNMPv2 snmpTrapOID with the last 2 sub-identifiers removed, otherwise
    - If the next-to-last sub-identifier of the snmpTrapOID is non-zero, then the SNMPv1 enterprise SHALL be the SNMPv2 snmpTrapOID with the last sub-identifier removed.

- The SNMPv1 agent-addr parameter SHALL be determined based on the situation in which the translation occurs.

    - If the translation occurs within a notification originator application, and the notification is to be sent over IP, the SNMPv1 agent-addr parameter SHALL be set to the IP address of the SNMP entity in which the notification originator resides. If the notification is to be sent over some other transport, the SNMPv1 agent-addr parameter SHALL be set to 0.0.0.0.

    - If the translation occurs within a proxy application, the proxy must attempt to extract the original source of the notification from the variable-bindings. If the SNMPv2 variable-bindings contains a variable binding whose name is snmpTrapAddress.0, the agent-addr parameter SHALL be set to the value of that variable binding. Otherwise, the SNMPv1 agent-addr parameter SHALL be set to 0.0.0.0.

- If the SNMPv2 snmpTrapOID parameter is one of the standard traps as defined in RFC1907, the SNMPv1 generic-trap parameter SHALL be set as follows:
-
-
-
-
-
-
-
-
-

snmpTrapOID.0 parameter	generic-trap
1.3.6.1.6.3.1.1.5.1 (coldStart)	0
1.3.6.1.6.3.1.1.5.2 (warmStart)	1
1.3.6.1.6.3.1.1.5.3 (linkDown)	2
1.3.6.1.6.3.1.1.5.4 (linkUp)	3
1.3.6.1.6.3.1.1.5.5 (authenticationFailure)	4
1.3.6.1.6.3.1.1.5.6 (egpNeighborLoss)	5

  Otherwise, the SNMPv1 generic-trap parameter SHALL be set to 6.

- If the SNMPv2 snmpTrapOID parameter is one of the standard traps as defined in RFC1907, the SNMPv1 specific-trap parameter SHALL be set to zero. Otherwise, the SNMPv1 specific-trap parameter SHALL be set to the last sub-identifier of the SNMPv2 snmpTrapOID parameter.

- The SNMPv1 time-stamp parameter SHALL be taken directly from the SNMPv2 sysUpTime parameter.

- The SNMPv1 variable-bindings SHALL be the SNMPv2 variable- bindings. Note, however, that if the SNMPv2 variable-bindings contain any objects whose type is Counter64, the translation to SNMPv1 notification parameters cannot be performed. In this case, the notification cannot be encoded in an SNMPv1 packet (and so the notification cannot be sent using SNMPv1, see section 4.1.3 and section 4.2).

SUBSTANTIAL PORTIONS OF THE SOFTWARE. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.