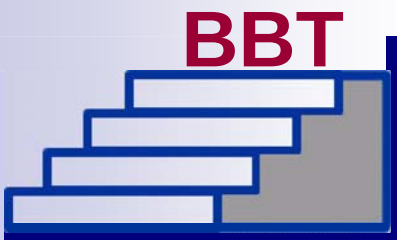


CPP

SNMP

SNMP

- The Simple Network Management Protocol (SNMP) is an application layer protocol that facilitates the exchange of management information between network devices. SNMP enables network administrators to manage network performance, find and solve network problems, and plan for network growth.

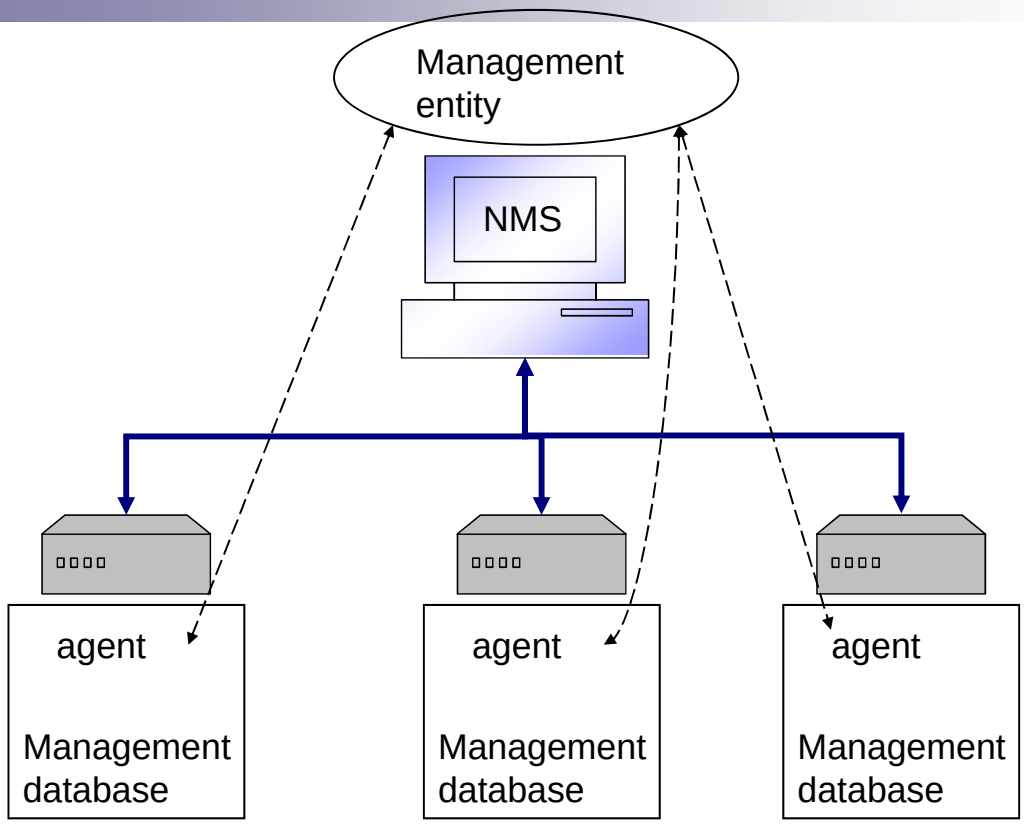
SNMP Basic Components

An SNMP-managed network consists of three key components:

- **managed device**
  - A network node that contains an SNMP agent and that resides on a managed network. Managed devices collect and store management information and make this information available to NMSs using SNMP. Managed devices, sometimes called network elements, can be routers and access servers, switches and bridges, hubs, computer hosts, or printers.

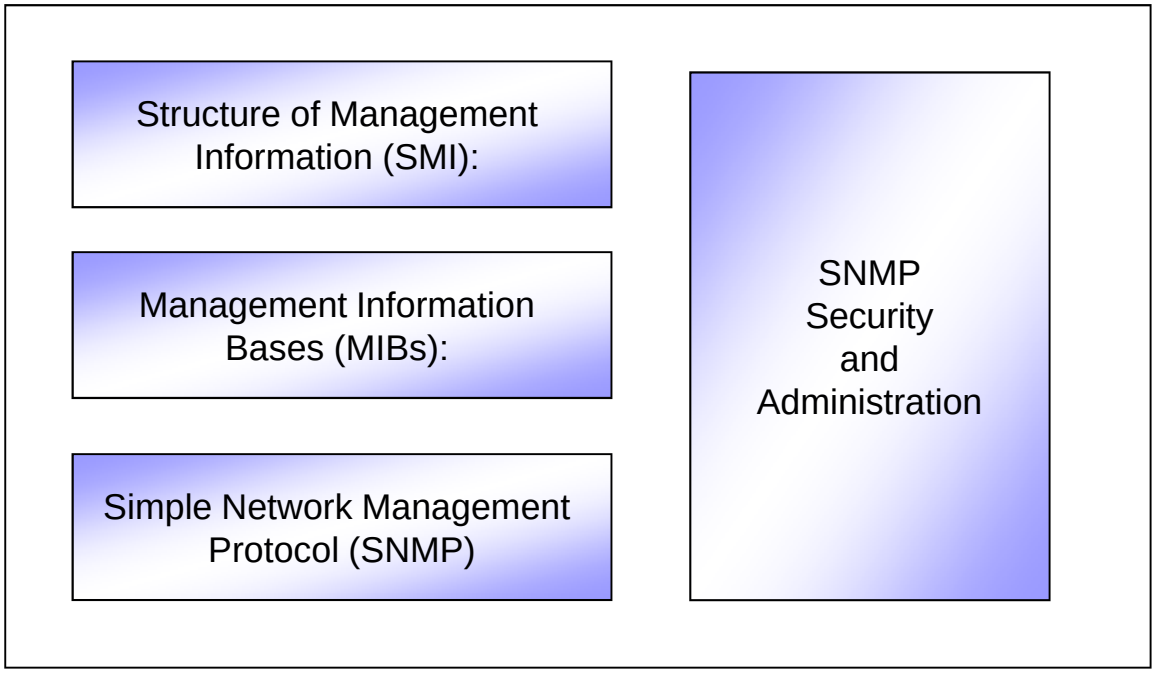
Components –cont.

- **agent**
  - a network-management software module that resides in a managed device. An agent has local knowledge of management information and translates that information into a form compatible with SNMP.
- **Network Management Station (NMS)**
  - executes applications that monitor and control managed devices. NMSs provide the bulk of the processing and memory resources required for network management. One or more NMSs must exist on any managed network.



SNMP-Managed Network Consists

SNMP Framework



Structure of Management Information (SMI):

- The Structure of Management Information (SMI) is a standard that defines the structure, syntax and characteristics of management information in SNMP.

Management Information Bases (MIBs)

- Each managed device contains a set of variables that is used to manage it. The management information base (MIB) is the full set of these variables that describe the management characteristics of a particular type of device.
- Each variable in a MIB is called a MIB object, and is defined using the SMI data description language. A device may have many objects, corresponding to the different hardware and software elements it contains.
- Two types of managed objects exist: scalar and tabular.
  - Scalar objects define a single object instance.
  - Tabular objects define multiple related object instances that are grouped in MIB tables.

Simple Network Management Protocol (SNMP)

- This is the actual SNMP protocol itself. It defines how information is exchanged between SNMP agents and network management stations.

Security and Administration

- These provide enhancements to the operation of the SNMP protocol for security, and address issues related to SNMP implementation, version transition and other administrative issues.

SMI overview

- The definition of managed objects can be broken down into three attributes:
- **Name**
  - The name, or object identifier (OID), uniquely defines a managed object. Names commonly appear in two forms: numeric and "human readable."
- **Type and syntax**
  - A managed object's datatype is defined using a subset of Abstract Syntax Notation One (ASN.1). ASN.1 is a way of specifying how data is represented and transmitted between managers and agents, within the context of SNMP.

SMI overview –cont.

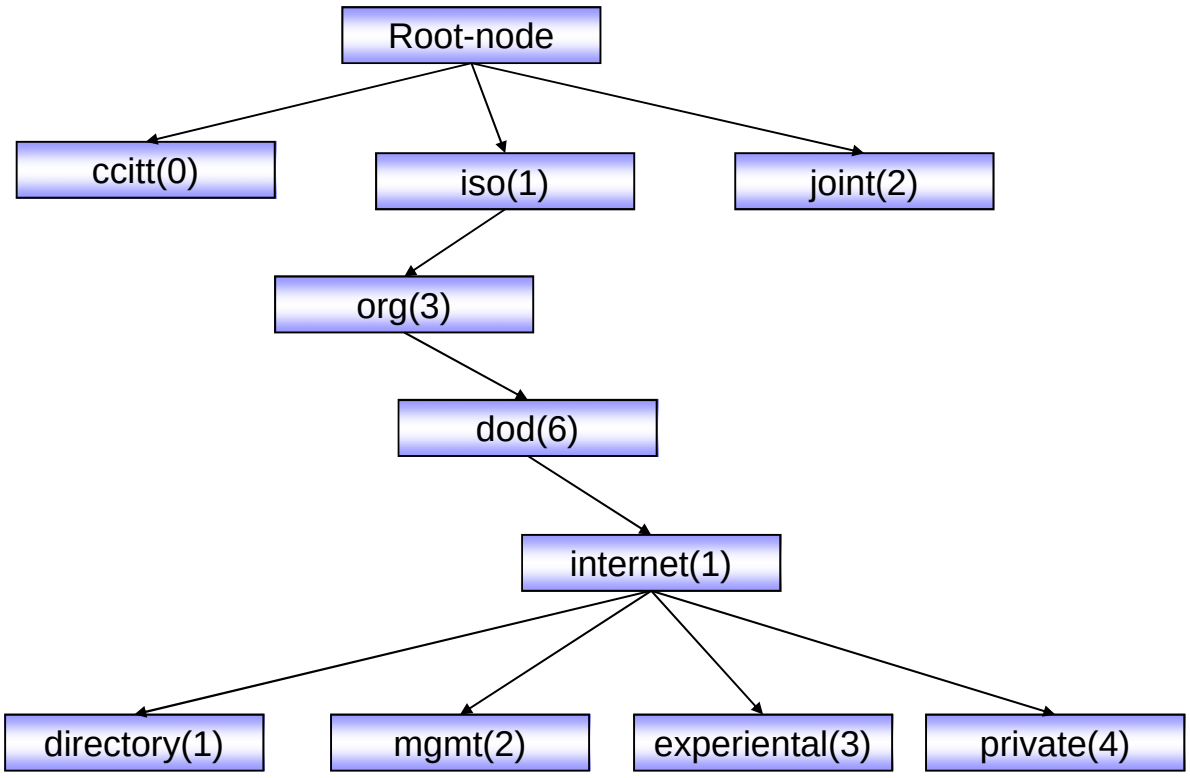
- **Encoding**
  - A single instance of a managed object is encoded into a string of octets using the Basic Encoding Rules (BER). BER defines how the objects are encoded and decoded so that they can be transmitted over a transport medium such as Ethernet.

Naming OIDs

- Managed objects are organized into a treelike hierarchy . This structure is the basis for SNMP's naming scheme. An object ID is made up of a series of integers based on the nodes in the tree, separated by dots (.)

- Example: iso.org.dod.internet is 1.3.6.1

SMI objects tree



SMIv1 datatypes

- **INTEGER**
  - A 32-bit number often used to specify enumerated types within the context of a single managed object. For example, the operational status of a router interface can be up, down, or testing. With enumerated types, 1 would represent up, 2 down, and 3 testing. The value zero (0) must not be used as an enumerated type, according to RFC 1155.
- **OCTET STRING**
  - A string of zero or more octets (more commonly known as bytes) generally used to represent text strings, but also sometimes used to represent physical addresses.
- **Counter**
  - A 32-bit number with minimum value 0 and maximum value 2^32 - 1 (4,294,967,295). When the maximum value is reached, it wraps back to zero and starts over.

SMIv1 datatypes –cont.

- **OBJECT IDENTIFIER**
  - A dotted-decimal string that represents a managed object within the object tree.
- **SEQUENCE**
  - Defines lists that contain zero or more other ASN.1 datatypes.
- **SEQUENCE OF**
  - Defines a managed object that is made up of a SEQUENCE of ASN.1 types.
- **IpAddress**
  - Represents a 32-bit IPv4 address. Neither SMIv1 nor SMIv2 discusses 128-bit IPv6 addresses .
- **NetworkAddress**
  - Same as the IpAddress type, but can represent different network address types.

SMIv1 datatypes –cont.

- **Gauge**
  - A 32-bit number with minimum value 0 and maximum value 2^32 - 1 (4,294,967,295). Unlike a Counter, a Gauge can increase and decrease at will, but it can never exceed its maximum value. The interface speed on a router is measured with a Gauge.
- **TimeTicks**
  - A 32-bit number with minimum value 0 and maximum value 2^32 - 1 (4,294,967,295). TimeTicks measures time in hundredths of a second. Uptime on a device is measured using this datatype.
- **Opaque**
  - Allows any other ASN.1 encoding to be stuffed into an OCTET STRING.

Understand MIB files (rfc1213)

```
RFC1213-MIB DEFINITIONS ::= BEGIN
        IMPORTS
                mgmt, NetworkAddress, IpAddress, Counter, Gauge,
                TimeTicks
                        FROM RFC1155-SMI
                OBJECT-TYPE
                        FROM RFC 1212;
        mib-2      OBJECT IDENTIFIER ::= { mgmt 1 }
-- groups in MIB-II
        system       OBJECT IDENTIFIER ::= { mib-2 1 }
        interfaces   OBJECT IDENTIFIER ::= { mib-2 2 }
        at           OBJECT IDENTIFIER ::= { mib-2 3 }
        ip           OBJECT IDENTIFIER ::= { mib-2 4 }
...
```

Understand MIB files –cont.

```
-- the Interfaces table
        ifTable OBJECT-TYPE
            SYNTAX  SEQUENCE OF IfEntry
            ACCESS  not-accessible
            STATUS  mandatory
            DESCRIPTION
                "A list of interface entries. The number of entries is
                 given by the value of ifNumber."
            ::= { interfaces 2 }
```

Understand MIB files –cont.

```
ifEntry OBJECT-TYPE
            SYNTAX  IfEntry
            ACCESS  not-accessible
            STATUS  mandatory
            DESCRIPTION
                "An interface entry containing objects at the subnetwork
                 layer and below for a particular interface."
            INDEX   { ifIndex }
            ::= { ifTable 1 }
```

Understand MIB files –cont.

```
IfEntry ::=
        SEQUENCE {
            ifIndex
                INTEGER,
            ifDescr
                DisplayString,
            .
            :
            ifOutQLen
                Gauge,
            ifSpecific
                OBJECT IDENTIFIER
        }
```

Understand MIB files –cont.

```
ifIndex OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "A unique value for each interface. Its value ranges
             between 1 and the value of ifNumber. The value for
             each interface must remain constant at least from one
             reinitialization of the entity's network management
             system to the next reinitialization."

        ::= { ifEntry 1 }
```

Understand MIB files –cont.

```
ifDescr OBJECT-TYPE
            SYNTAX  DisplayString (SIZE (0..255))
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                "A textual string containing information about the
                 interface. This string should include the name of
                 the manufacturer, the product name, and the version
                 of the hardware interface."
            ::= { ifEntry 2 }

END
```

Definition format:

```
<name> OBJECT-TYPE
    SYNTAX <datatype>
    ACCESS <either read-only, read-write, write-only, or not-accessible>
    STATUS <either mandatory, optional, or obsolete>
    DESCRIPTION
        "Textual description describing this particular managed object."
    ::= { <Unique OID that defines this object> }
```

New datatypes for SMIv2 (rfc2578)

- **Integer32**
  - Same as an INTEGER.
- **Counter32**
  - Same as a Counter.
- **Gauge32**
  - Same as a Gauge.
- **Unsigned32**
  - Represents decimal values in the range of 0 to 2^32 - 1, inclusive.
- **Counter64**
  - Similar to Counter32, but its maximum value is 18,446,744,073,709,551,615. Counter64 is ideal for situations in which a Counter32 may wrap back to 0 in a short amount of time.
- **BITS**
  - An enumeration of nonnegative named bits.

SMIv2 Definition format:

<name> OBJECT-TYPE
SYNTAX <datatype>
UnitsParts <Optional, see below>
MAX-ACCESS <See below>
STATUS <See below>
DESCRIPTION
"Textual description describing this particular managed object."
AUGMENTS { <name of table> }
::= { <Unique OID that defines this object> }

SMIv2 object definition enhancements

- **UnitsParts**
  - A textual description of the units (i.e., seconds, milliseconds, etc.) used to represent the object.
- **MAX-ACCESS**
  - An OBJECT-TYPE's ACCESS can be MAX-ACCESS in SNMPv2 . The valid options for MAX-ACCESS are read-only, read-write, read-create, not-accessible, and accessible-for-notify.
- **STATUS**
  - This clause has been extended to allow the current, obsolete, and deprecated keywords. current in SNMPv2 is the same as mandatory in an SNMPv1 MIB.
- **AUGMENTS**
  - In some cases, it is useful to add a column to an existing table. The AUGMENTS clause allows you to extend a table by adding one or more columns, represented by some other object. This clause requires the name of the table the object will augment.

Textual conventions for SMIv2 (rfc2579)

- **DisplayString**
  - A string of NVT ASCII characters. A DisplayString can be no more than 255 characters in length.
- **PhysAddress**
  - A media- or physical-level address, represented as an OCTET STRING.
- **MacAddress**
  - Defines the media-access address for IEEE 802 (the standard for LANs) in canonical[*] order. (In everyday language, this means the Ethernet address.) This address is represented as six octets.
- **TruthValue**
  - Defines both true and false Boolean values.
- **TestAndIncr**
  - Used to keep two management stations from modifying the same managed object at the same time.

Textual conventions –cont.

- **AutonomousType**
  - An OID used to define a subtree with additional MIB-related definitions.
- **VariablePointer**
  - A pointer to a particular object instance, such as ifDescr for interface 3. In this case, the VariablePointer would be the OID ifDescr.3.
- **RowPointer**
  - A pointer to a row in a table. For example, ifIndex.3 points to the third row in ifTable.
- **RowStatus**
  - Used to manage the creation and deletion of rows in a table, since SNMP has no way of doing this via the protocol itself. RowStatus can keep track of the state of a row in a table as well as receive commands for creation and deletion of rows. This textual convention is designed to promote table integrity when more than one manager is updating rows. The following enumerated types define the commands and state variables: active(1), notInService(2), notReady(3), createAndGo(4), createAndWait(5), and anddestroy(6).

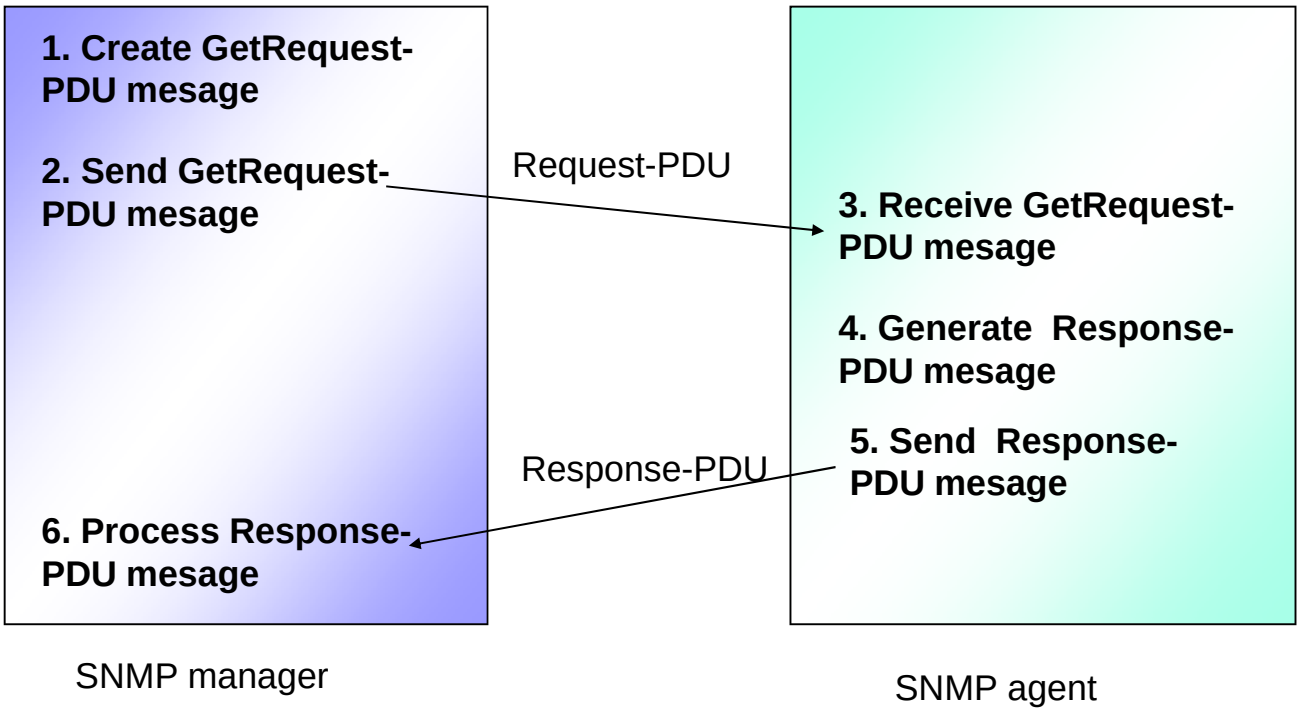
Textual conventions –cont.

- **TimeStamp**
  - Measures the amount of time elapsed between the device's system uptime and some event or occurrence.
- **TimeInterval**
  - Measures a period of time in 0.01second. TimeInterval can take any integer value from 0-2147483647.
- **DateAndTime**
  - An OCTET STRING used to represent date and time information.
- **StorageType**
  - Defines the type of memory an agent uses. The possible values are other(1), volatile(2), nonVolatile(3), permanent(4), and readOnly(5).
- **TDomain**
  - Denotes a kind of transport service.
- **TAddress**
  - Denotes the transport service address. TAddress is defined to be from 1-255 octets in length.

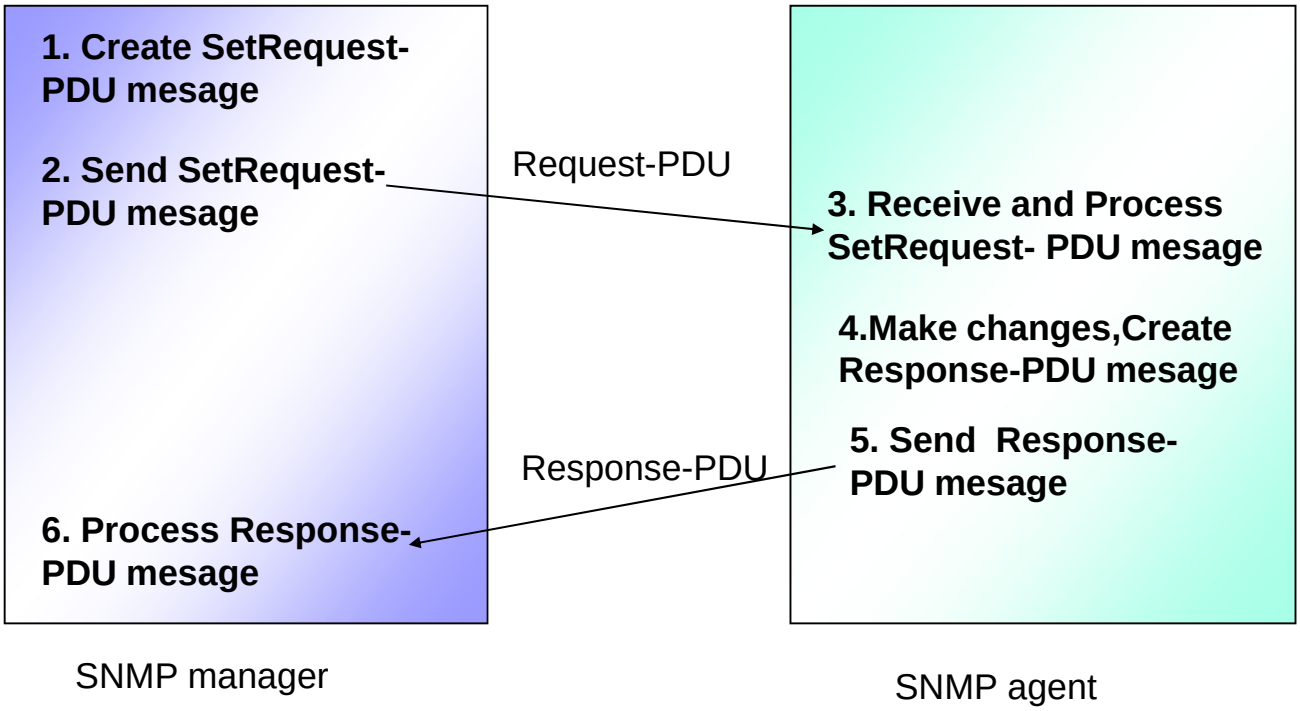
SNMP Operations

- get
- getnext
- getbulk (SNMPv2 and SNMPv3)
- set
- getresponse
- trap
- notification (SNMPv2 and SNMPv3)
- inform (SNMPv2 and SNMPv3)
- report (SNMPv2 and SNMPv3)

SNMP Information Poll Process



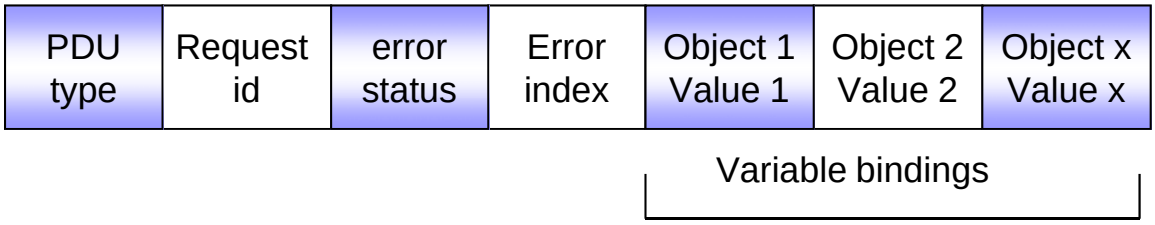
SNMP Object Modification Process



SNMP Version 1 (SNMPv1) General Message Format

- SNMPv1 message headers contain two fields: Version Number and Community Name.
  - **Version number** —Specifies the version of SNMP used.
  - **Community name** —Defines an access environment for a group of NMSs. NMSs within the community are said to exist within the same administrative domain. Community names serve as a weak form of authentication because devices that do not know the proper community name are precluded from SNMP operations.

Protocol Data Unit (PDU)



- **PDU type** —Specifies the type of PDU transmitted.
- **Request ID** —Associates SNMP requests with responses.
- **Error status** —Indicates one of a number of errors and error types. Only the response operation sets this field. Other operations set this field to zero.
- **Error index** —Associates an error with a particular object instance. Only the response operation sets this field. Other operations set this field to zero.
- **Variable bindings** —Serves as the data field of the SNMPv1 PDU. Each variable binding associates a particular object instance with its current value (with the exception of Get and GetNext requests, for which the value is ignored).

SNMPv1 error status

- **noError(0)** — There was no problem performing the request.
- **tooBig(1)** — The response to your request was too big to fit into one response.
- **noSuchName(2)** — An agent was asked to get or set an OID that it can't find; i.e., the OID doesn't exist.
- **badValue(3)** — A read-write or write-only object was set to an inconsistent value.
- **readOnly(4)** — This error is generally not used. The noSuchName error is equivalent to this one.
- **genErr(5)** —This is a catchall error. If an error occurs for which none of the previous messages is appropriate, a genErr is issued.

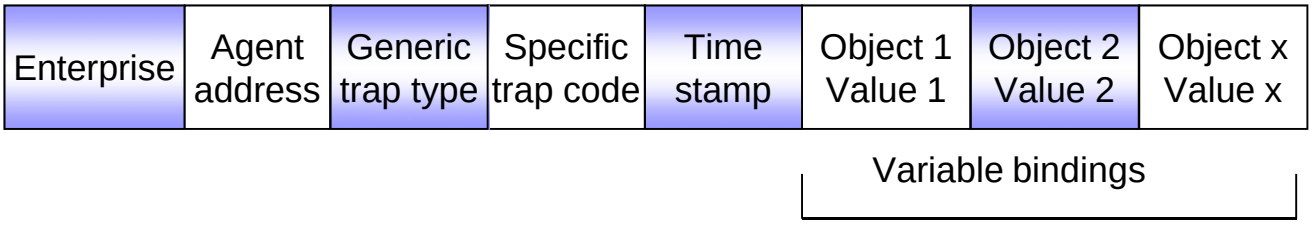
SNMPv2 error messages

- **noAccess(6)** — A set to an inaccessible variable was attempted. This typically occurs when the variable has an ACCESS type of not-accessible.
- **wrongType(7)** — An object was set to a type that is different from its definition. This error will occur if you try to set an object that is of type INTEGER to a string, for example.
- **wrongLength(8)** — An object's value was set to something other than what it calls for. For instance, a string can be defined to have a maximum character size. This error occurs if you try to set a string object to a value that exceeds its maximum length.
- **wrongEncoding(9)** — A set operation was attempted using the wrong encoding for the object being set.
- **wrongValue(10)** — A variable was set to a value it doesn't understand. This can occur when a read-write is defined as an enumeration, and you try to set it to a value that is not one of the enumerated types.

SNMPv2 error messages –cont.

- **noCreation(11)** — You tried to set a nonexistent variable or create a variable that doesn't exist in the MIB.
- **inconsistentValue (12)—** A MIB variable is in an inconsistent state and is not accepting any set requests.
- **resourceUnavailable(13)** — No system resources are available to perform a set.
- **commitFailed(14)** — This error is a catchall for set failures.
- **undoFailed(15)** — A set failed and the agent was unable to roll back all the previous sets up until the point of failure.
- **authorizationError(16)** — An SNMP command could not be authenticated; in other words, someone has supplied an incorrect community string.
- **notWritable(17)** — A variable will not accept a set, even though it is supposed to.
- **inconsistentName(18)** — You attempted to set a variable, but that attempt failed because the variable was in some kind of inconsistent state.

SNMPv1 Trap PDU



- **Enterprise** —Identifies the type of managed object generating the trap.
- **Agent address** —Provides the address of the managed object generating the trap.
- **Generic trap type** —Indicates one of a number of generic trap types.
- **Specific trap code** —Indicates one of a number of specific trap codes.
- **Time stamp** —Provides the amount of time that has elapsed between the last network reinitialization and generation of the trap.

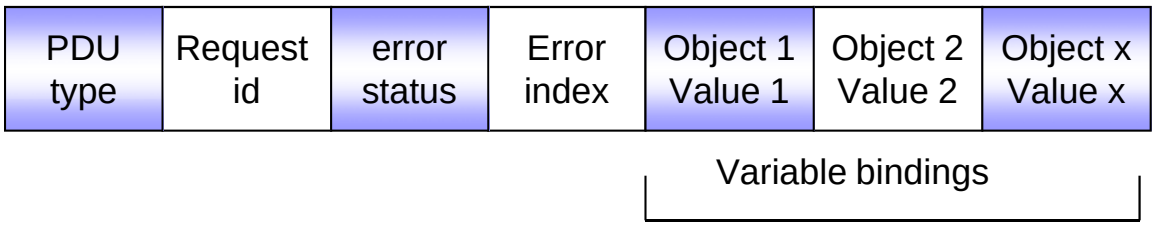
Generic traps

- **coldStart (0)**
  - Indicates that the agent has rebooted. All management variables will be reset; specifically, Counters and Gauges will be reset to zero (0). One nice thing about the coldStart trap is that it can be used to determine when new hardware is added to the network. When a device is powered on, it sends this trap to its trap destination. If the trap destination is set correctly (i.e., to the IP address of your NMS), the NMS can receive the trap and determine whether it needs to manage the device.
- **warmStart (1)**
  - Indicates that the agent has reinitialized itself. None of the management variables will be reset.
- **linkDown (2)**
  - Sent when an interface on a device goes down. The first variable binding identifies the index in the interfaces table for the interface that went down.

Generic traps –cont.

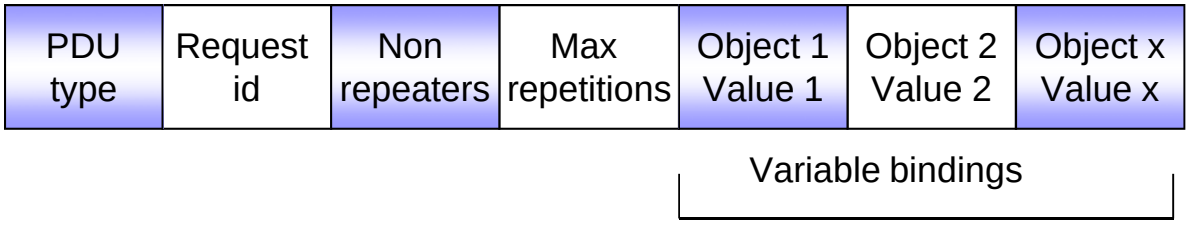
- **linkUp (3)**
  - Sent when an interface on a device comes back up. The first variable binding identifies which interface came back up.
- **authenticationFailure (4)**
  - Indicates that someone has tried to query your agent with an incorrect community string; useful in determining if someone is trying to gain unauthorized access to one of your devices.
- **egpNeighborLoss (5)**
  - Indicates that an EGP neighbor has gone down.
- **enterpriseSpecific (6)**
  - Indicates that the trap is enterprise-specific. SNMP vendors and users define their own traps under the private-enterprise branch of the SMI object tree. To process this trap properly, the NMS has to decode the specific trap number that is part of the SNMP message.

SNMPv2 Protocol Data Unit (PDU)



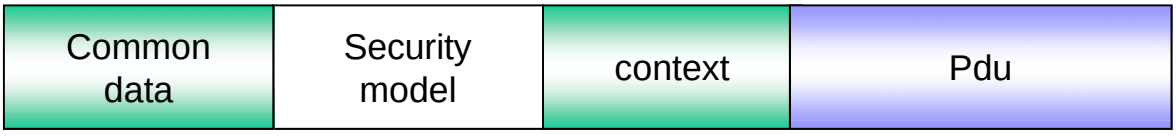
- Get, GetNext, Inform, Response, Set, and Notification PDUs Contain the Same Fields

SNMPv2 GetBulk PDU



- **Non repeaters** —Specifies the number of object instances in the variable bindings field that should be retrieved no more than once from the beginning of the request. This field is used when some of the instances are scalar objects with only one variable.
- **Max repetitions** —Defines the maximum number of times that other variables beyond those specified by the Non repeaters field should be retrieved.

SNMPv3 message format



- **Common data** —These fields occur in all SNMPv3 messages.
- **Security model data** —This area has three subsections—one general, one for authentication, and one for privacy data.
- **Context** —These two fields are used to provide the correct context in which the protocol data unit (PDU) should be processed.
- **PDU** —This area contains an SNMPv2c PDU.

Common data

- **MessageVersion**
  - The first field in the message is the SNMP version. This provides for backwards and forwards compatibility. A value of 3 in this field indicates an SNMPv3 message.
- **MessageID**
  - The MessageID is a number used between two entities for message correlation. So, if a manager sends a GetRequest with MessageID x, then it is important that the manager does not re-use x until the outstanding message is answered or timed out.
- **MaxMessageSize**
  - The MaxMessageSize is the maximum message size supported by the sender of the message.

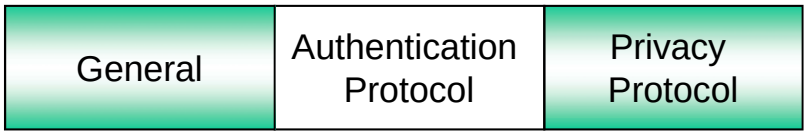
Common data –cont.

- **MessageFlags**
  - The MessageFlags object is 1-byte long and determines the authentication and privacy settings for the message. It also indicates if this message requires a (report) response from the receiver. The three right-most bit positions are used when encoding this object, and the following are the allowed combinations:
    - No authentication and no privacy (bit values 000)
    - Authentication and no privacy (bit values 001)
    - Authentication and privacy (bit values 011)
  - All three of the above may have the report option set. This indicates that a response is required.

Common data –cont.

- **MessageSecurity**
  - The MessageSecurity is an integer object that indicates the security setting associated with the message. The range of values supported is as follows:
    - 0 is reserved for "any."
    - 1 is reserved for SNMPv1.
    - 2 is reserved for SNMPv2c.
    - 3 is reserved for USM.
    - 4–255 is reserved for standards-track security models.

Security Model Data



Security Model Data: General

- **EngineID**: unique identification of an SNMPv3 engine
- **EngineBoots:** the number of times an SNMP engine has either been started up or reset since the value of EngineID was last modified
- **EngineTime:** the number of seconds that have passed since the value of EngineBoots was last modified
- **UserName:** the name of a user

Security Model Data: Authentication Protocol

- Two authentication protocols are supported in SNMPv3, namely MD5 and SHA

Security Model Data: Privacy Protocol

- the privacy protocol field is an 8-byte octet string used for the Data Encryption Standard (DES) algorithm

Context

- Contexts were invented to allow multiple instances of the same MIB table within the same SNMP agent in order to handle same special cases.

END