Chapter 5

SNMPv1 Network Management:

Communication and Functional Models

Objectives

- Communication model: Administrative structure and messages
- Administrative structure
  - Community-based model
  - Access policy
  - MIB view
- Message PDU (Protocol data Unit)
- SNMP protocol specifications
- SNMP operations
- SNMP MIB
- SNMP functional model

Communication model

The SNMPv1 communication model defines specifications of four aspects of SNMP communication: architecture, administrative model that defines data access policy, SNMP protocol, and SNMP MIB.

Security in SNMP is managed by defining community, and only members belonging to the same community can communicate with each other. A manager can belong to multiple communities and can thus manage multiple domains. SNMP protocol specifications and messages are presented. SNMP entities are grouped into an SNMP MIB module.

SNMP Architecture

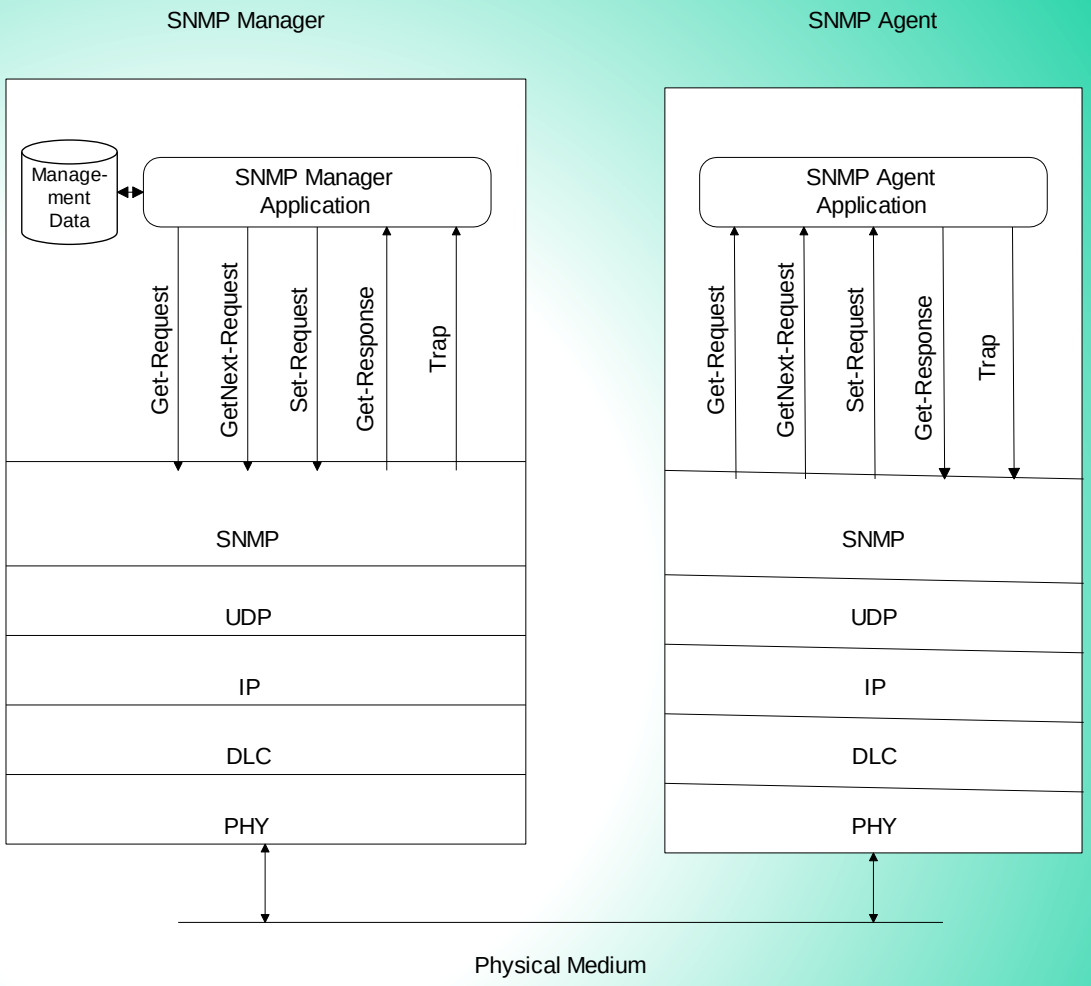


Figure 4.9 SNMP Network Management Architecture

Notes

- Truly s*imple* network management protocol
- Five messages, 3 from manager and 2 from agent

SNMP monitors the network with the five messages shown in Figure 4.9; and we discussed them in Section 4.6. They comprise three basic messages: set, get, and trap. Information about the network is primarily obtained by the management stations polling the agents. The get-request and get-next-request messages are generated by the manager to retrieve data from network elements using associated management agents. The set-request is used to initialize and edit network element parameters. The get-response-request is the response from the agent to get and set messages from the manager. The number of unsolicited messages in the form of traps is limited to make the architecture simple and to minimize traffic.

SNMP Messages

- Get-Request
- Get-Next-Request
- Set-Request
- Get-Response
- Trap
  - Generic trap
  - Specific trap
  - Time stamp

Notes

- Generic trap
  - coldStart
  - warmStart
  - linkDown
  - linkUp
  - authenticationfailure
  - egpNeighborLoss
  - enterpriseSpecific
- Specific trap
  - For special measurements such as statistics
- Time stamp: Time since last initialization

Details later

get-next-request and get-request – SNMP commands

- The get-request will only match exactly the OID (ObjectIDentifier) specified in the request.
- It will fail if no such OID exists on the target.
- The get-next-request will return the OID that is the "next" OID in the "tree".
    - The get-next-request can be used to "walk through" the OIDs in the MIB of a certain domain/system

Administrative Model

Although the topic of administrative models should normally be discussed as part of security and privacy under the functional model, at this point it helps to understand the administrative relationship among entities that participate in the communication protocol in SNMP.

Notes

Administrative Model

The administrative model defines SNMP Entities, community profile and policy of communication

Peer processes, which implement SNMP, and thus support SNMP application entities, they are termed protocol entities

- SNMP Entities:
  - SNMP application entities
    - Reside in management stations and network elements
    - Manager and agent
  - SNMP protocol entities
    - Communication processes (PDU handlers)
    - Peer processes that support application entities

- community profile and policy
  - Community: Pairing of two application entities (SNMP Manager and SNMP agent)
  - Community name: String of octets
  - Two applications in the same community communicate with each other

A community name acts as a password that is shared, typically, by multiple SNMP agents and one or more SNMP managers. You configure the SNMP manager and the computers or devices that it manages as members of a single SNMP community. An SNMP agent only accepts requests from SNMP managers that are on the agent's list of acceptable community names.

SNMP Community

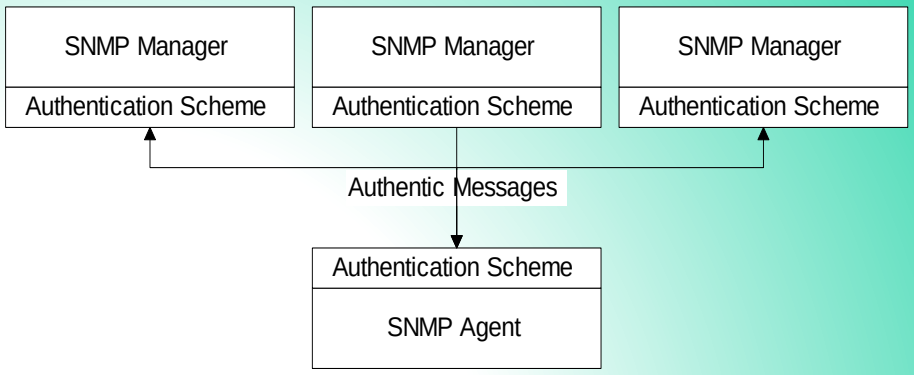


Figure 5.1 SNMP Community

Notes

- Security in SNMPv1 is community based
- Authentication scheme in manager and agent
- **Application** could have multiple community names
- Communication is not secured in SNMPv1 - no encryption

Multiple pairs can belong to the same community. Figure 5.1 shows multiple SNMP managers communicating with a single SNMP agent. While an SNMP manager is monitoring traffic on an element, another manager may be configuring some administrative information on it. A third manager can be monitoring it to perform some statistical study. We also have the analogous situation where a manager communicates with multiple agents.

Community Profile

The SNMP authorization is implemented as part of managed object MIB specifications.

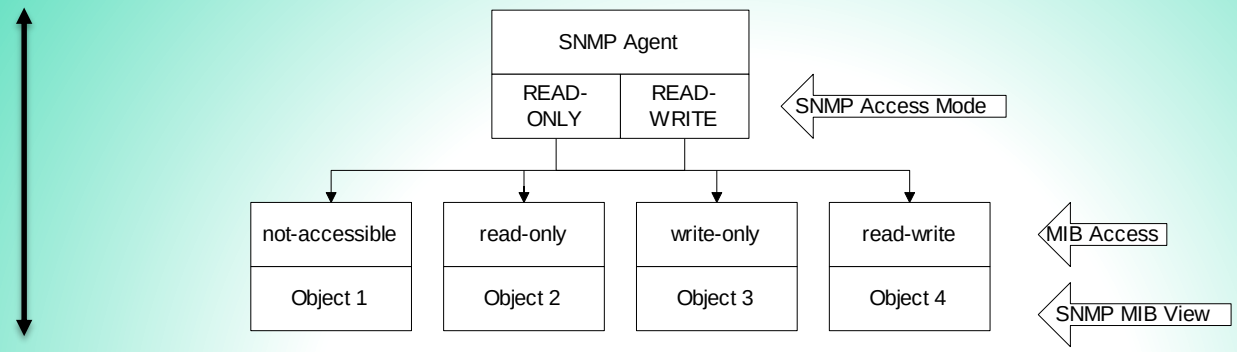


Figure 5.2 SNMP Community Profile

Notes

- MIB view
  - An agent is programmed to view only a subset of managed objects ("attributes", data types, object types) of a network element
- Access mode
  - Each community name is assigned an access mode: read-only and read-write
- **Community profile: MIB view + SNMP access mode (of the community)**
- Operations on an object determined by community profile and the access mode of the object
- Total of four access privileges
- Some objects, such as table and table entry are non-accessible

Administrative Model

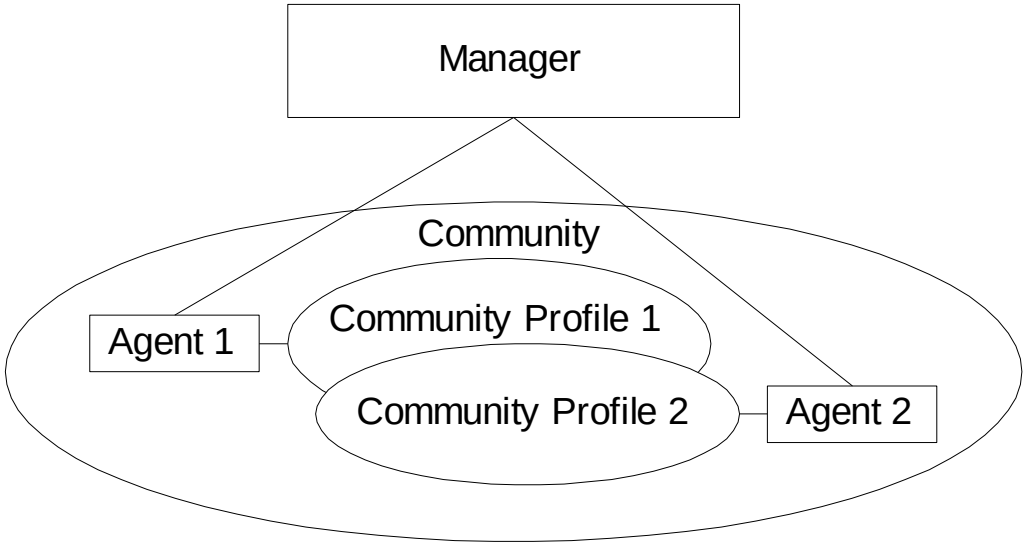
- Administrative model is SNMP access policy

- **SNMP community paired with SNMP community profile is SNMP access policy**

Notes

Parameters:
- Community / communities
- Agent / Agents
- Manager / Managers

Access Policy



Notes

- Manager manages Community 1 and 2 network components via Agents 1 and 2
- Agent 1 has only view of Community Profile 1, e.g., Cisco components
- Agent 2 has only view of Community Profile 2, e.g., 3Com components
- Manager has total view of both Cisco and 3Com components

Generalized Administrative Model

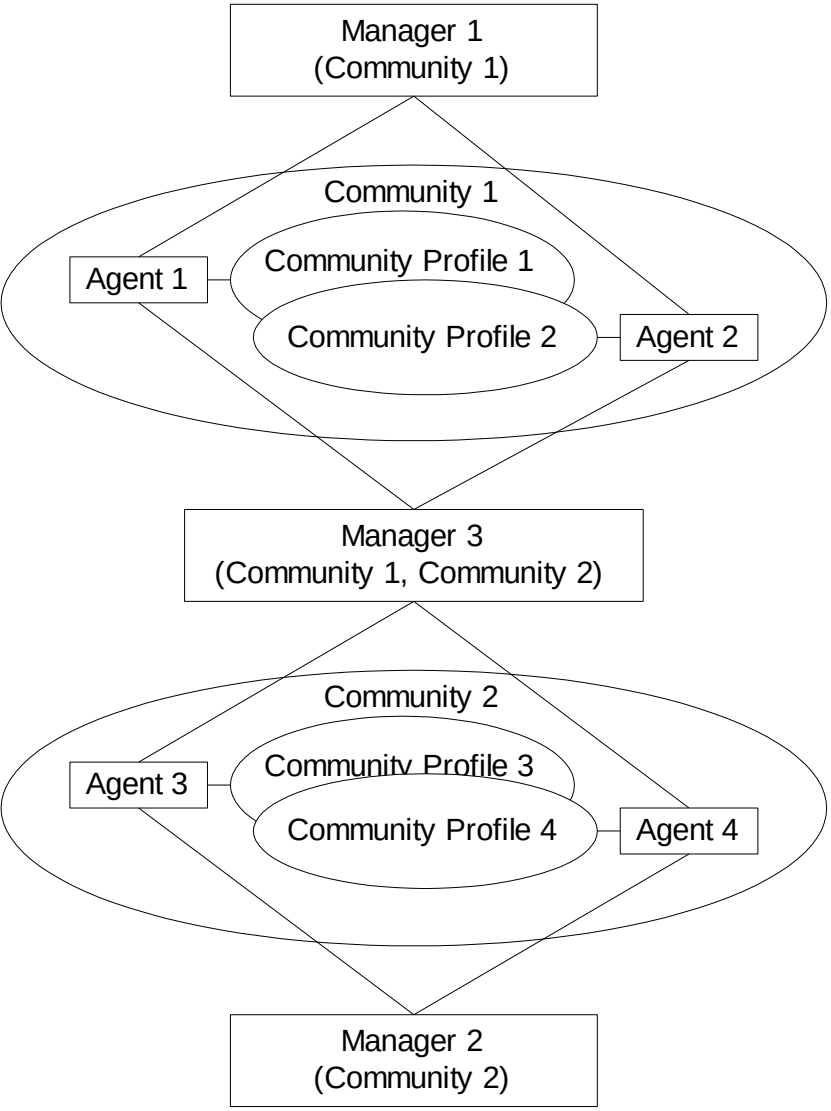


Figure 5.3 SNMP Access Policy

Notes

- Manager 1 manages community 1, manager 2 community 2, and manager 3 (MoM) both communities 1 and 2

Proxy Access Policy

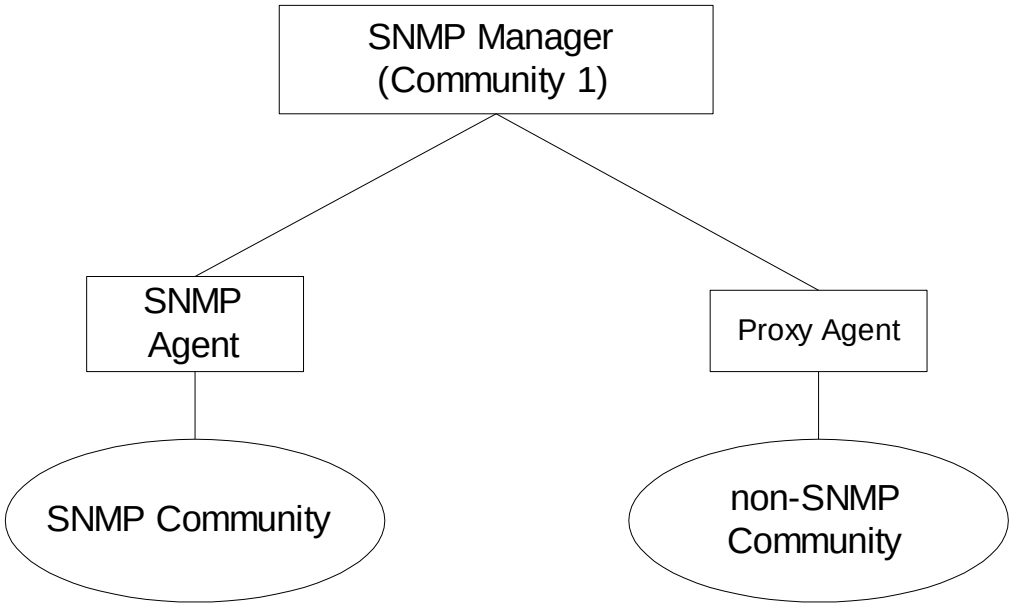


Figure 5.4 SNMP Proxy Access Policy

Notes

- **Proxy agent enables non-SNMP community elements to be managed by an SNMP manager**.

- An SNMP MIB is created to handle the non-SNMP objects.

Protocol Entities

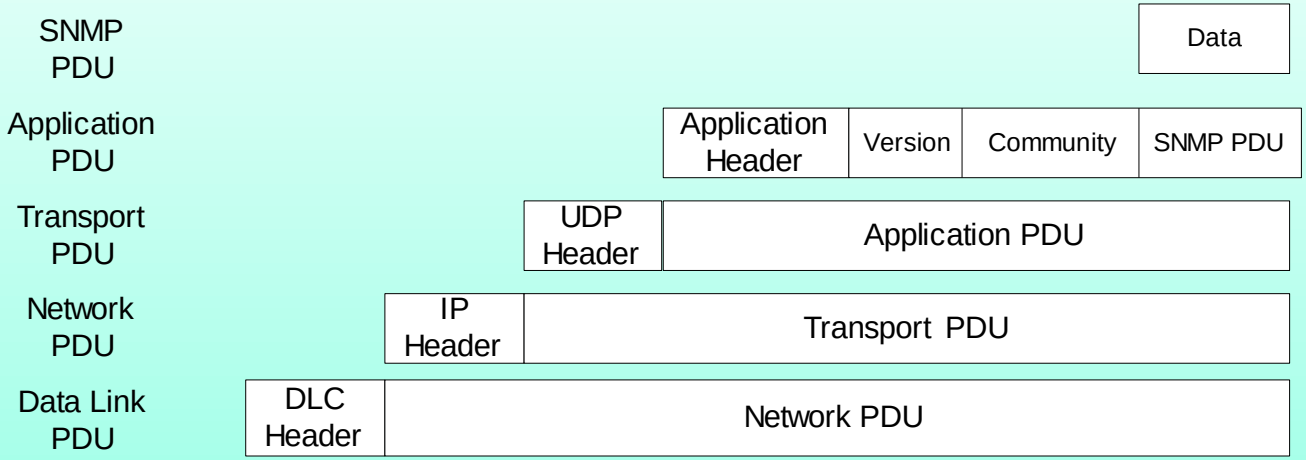


Figure 5.5 Encapsulated SNMP Message

Notes

- Protocol entities support application entities
- Communication between remote **peer processes**
- Message consists of:
  - Version identifier
  - Community name
  - Protocol Data Unit
- Message encapsulated and transmitted

Get and Set PDU

PDU Type	RequestID	Error Status	Error Index	VarBind 1 name	VarBind 1 value	...	VarBind n name	VarBind n value

Figure 5.8 Get and Set Type PDUs

Notes

- VarBindList: multiple instances of VarBind pairs

```
PDUs ::=
        CHOICE {
            get-request          GetRequest-PDU,
            get-next-request     GetNextRequest-PDU,
            get-response         GetResponse-PDU,
            set-request          SetRequest-PDU,
            trap              Trap-PDU
            }
```

PDU Types: enumerated INTEGER

get-request	[0]
get-next-request	[1]
set-request	[2]
get-response	[3]
trap	[4]

- **A managed object is represented in SNMP by a scalar variable and is simply called a variable.**
- **Associated with the variable is its value. The pairing of the variable and value is called variable binding or VarBind.**
- **The data PDU in the message contains a VarBind pair. For efficiency sake, a list of VarBind pairs can be sent in a message.**

Error in Response

```
ErrorStatus ::=
    INTEGER {
        noError(0)
        tooBig(1)
        noSuchName(2)
        bad value(3)
        readOnly(4)
        genErr(5)
    }
```

Error Index: No. of VarBind that the first error occurred

Notes

PDU Type	RequestID	Error Status	Error Index	VarBind 1 name	VarBind 1 value	...	VarBind n name	VarBind n value

Figure 5.8 Get and Set Type PDUs

Trap PDU

PDU Type	Enterprise	Agent Address	Generic Trap Type	Specific Trap Type	Timestamp	VarBind 1 name	VarBind 1 value	...	VarBind n name	VarBind n value

Figure 5.8 Get and Set Type PDUs

Table 5.1 Generic Traps

Generic Trap Type	Description (brief)
coldStart(0)	Sending protocol entity is reinitializing itself; agent's configuration or protocol entity implementation may be altered
warmStart(1)	Sending protocol entity is reinitializing itself; agent configuration or protocol entity implementation not altered
linkDown(2)	Failure of one of the communication links
linkUp(3)	One of the links has come up
authenticationFailure(4)	Authentication failure
egpNeighborLoss(5)	Loss of EGP neighbor
enterpriseSpecific(6)	Enterprise-specific trap

The agent is reporting that the peer relationship between an External Gateway Protocol (EGP) neighbor and an EGP peer no longer exists.

Notes

- Enterprise and agent address pertain to the system generating the trap
- Seven generic traps specified by enumerated INTEGER
- Specific trap is a trap not covered by enterprise specific trap
- Timestamp indicates elapsed time since last re-initialization

SNMP Operations

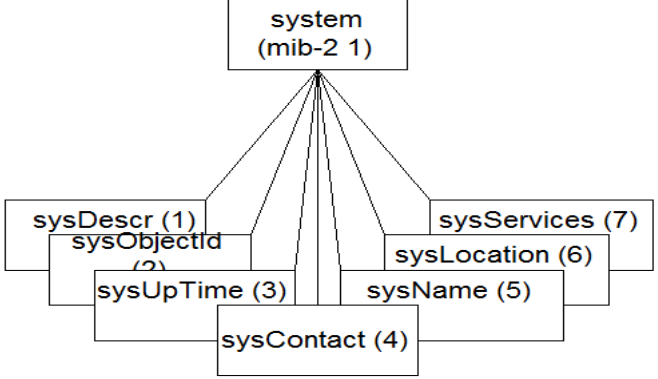
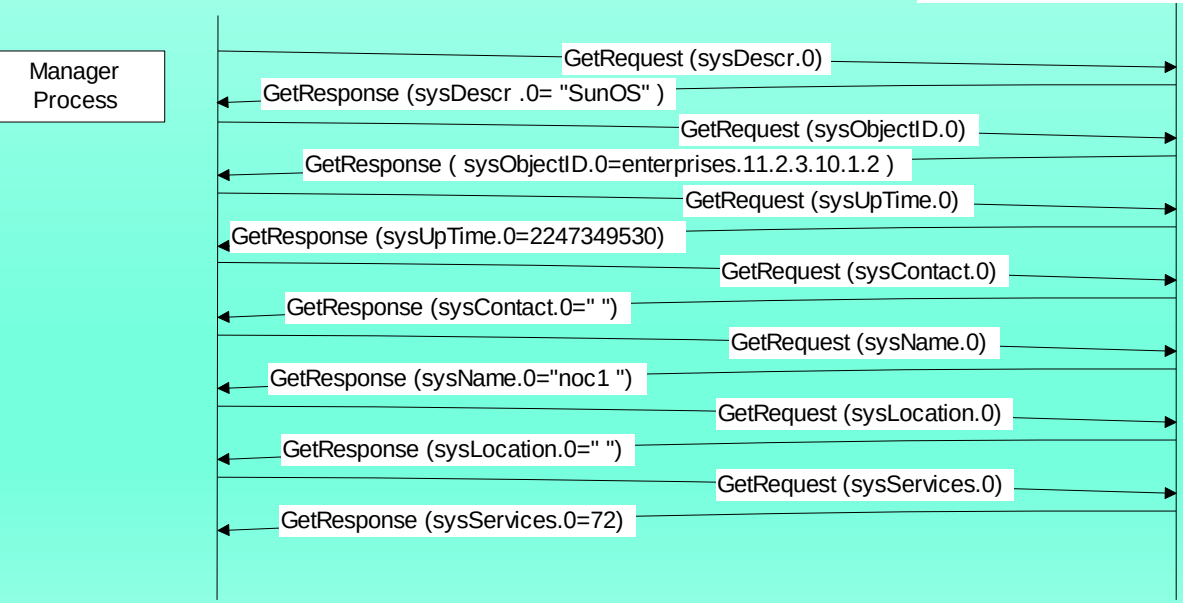


Figure 4.27 System Group

Notes

Entity	OID	Description (brief)
sysDescr	system 1	Textual description
sysObjectID	system 2	OBJECT IDENTIFIER of the entity
sysUpTime	system 3	Time (in hundredths of a second since last reset)
sysContact	system 4	Contact person for the node
sysName	system 5	Administrative name of the system
sysLocation	system 6	Physical location of the node
sysServices	system 7	Value designating the layer services provided by the entity



The agent here is integrated in the NE since in the GetRequest, no MO is specified.

Figure 5.10 Get-Request Operation for System Group

System Group

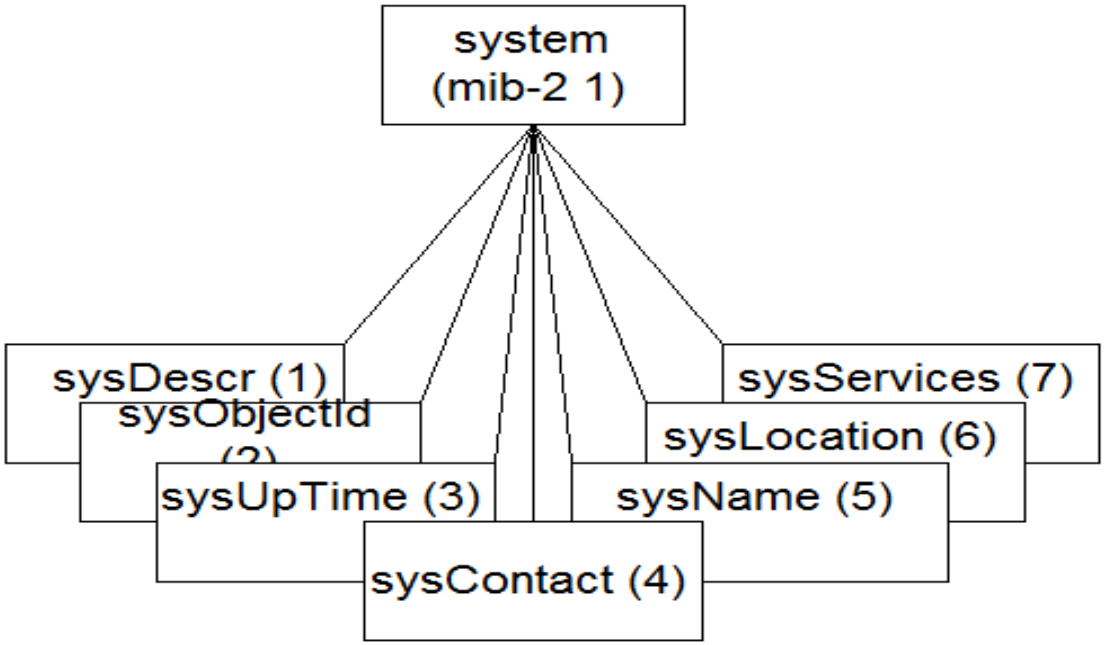


Figure 4.27 System Group

Notes

Entity	OID	Description (brief)
sysDescr	system 1	Textual description
sysObjectID	system 2	OBJECT IDENTIFIER of the entity
sysUpTime	system 3	Time (in hundredths of a second since last reset)
sysContact	system 4	Contact person for the node
sysName	system 5	Administrative name of the system
sysLocation	system 6	Physical location of the node
sysServices	system 7	Value designating the layer services provided by the entity

MIB for Get-Next-Request

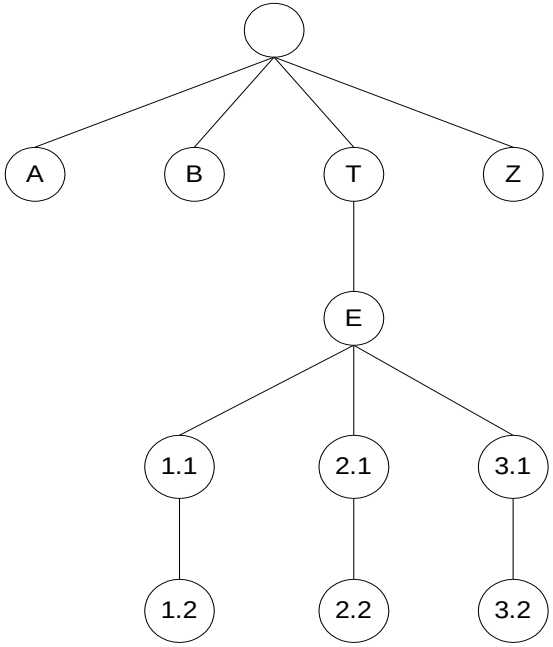


Figure 5.12 MIB for Operation Examples in Figures 5.13 and 5.

The nodes can be ordered **Lexicographic ally**

Notes

Lexicographic Order

Table 5.2 Lexicographic-Order Number Example

Notes

- Procedure for ordering:
  - Start with leftmost digit as first position
  - Before increasing the order in the first position, select the lowest digit in the second position
  - Continue the process till the lowest digit in the last position is captured
  - Increase the order in the last position until all the digits in the last position are captured
  - Move back to the last but one position and repeat the process
  - Continue advancing to the first position until all the numbers are ordered
- Tree structure for the above process

Numerical Order	Lexicographic order
1	1
2	1118
3	115
9	126
15	15
22	2
34	22
115	250
126	2509
250	3
321	321
1118	34
2509	9

MIB Lexicographic Order

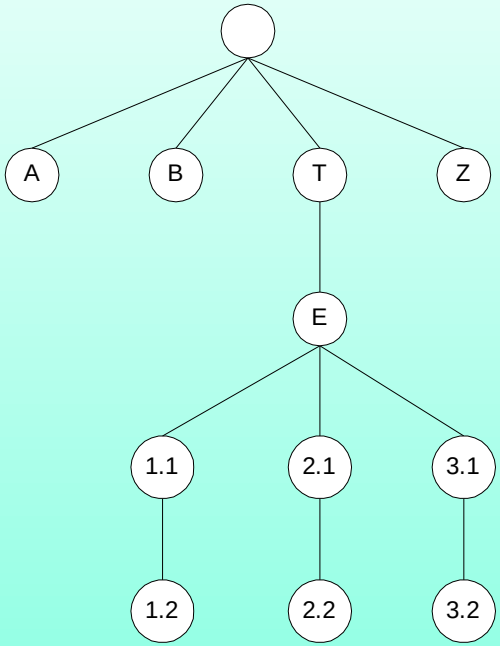


Figure 5.12 MIB for Operation Examples in Figures 5.13 and 5.15

The System group example we just looked al is a simple case where all the objects are single-valued scalar objects.

Let us now consider a more complex scenario of a MlB that contains both scalar and aggregate objects.

A generalized case of a conceptual MIB comprising three scalar objects and a table is shown in Figure 5.12. contains two objects A and B which arc single- valued scalar objects. They are followed by an aggregate object represented by the table T with an entry E and **two rows of three columnar objects**, T.E.1.l. through T.E.3.2.

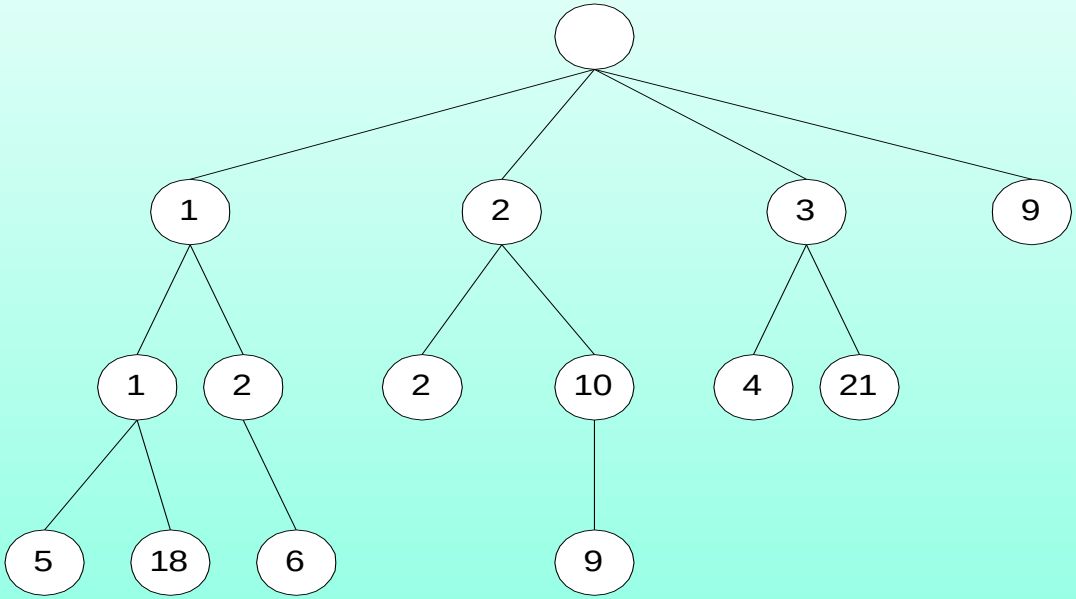
The MIB group ends wiih n scalar object Z.

Notes

A
B
T
E
1.1
1.2
2.1
2.2
3.1
3.2
Z

N.B: 1.1 and 1.2 are objects of the first columnar object…

A More Complex MIB Example



1
1.1
1.1.5
1.1.18
1.2
1.2.6
2
2.2
2.10
2.10.9
3
3.4
3.21
9

Figure 5.14 MIB Example for Lexicographic Ordering

The MIB associated with this example is shown in Figure 5.14. It can be noticed that the lexicographically increasing order of node traces the traversal of the tree starting from the leftmost node 1. We traverse down the path all the way to the leftmost leaf 1.1.5, keeping to the right whenever a fork is encountered. We then move up the tree and take a right on the first fork. This leads us to the leaf node 1.1.18. Thus, the rule at a forked node is to always keep to the right while traversing down and while going up. Thus, we are always keeping to the right if you imagine ourselves walking along the tree path and looking in the forward direction. We turn around when we reach a leaf.

Get-Next-Request Operation

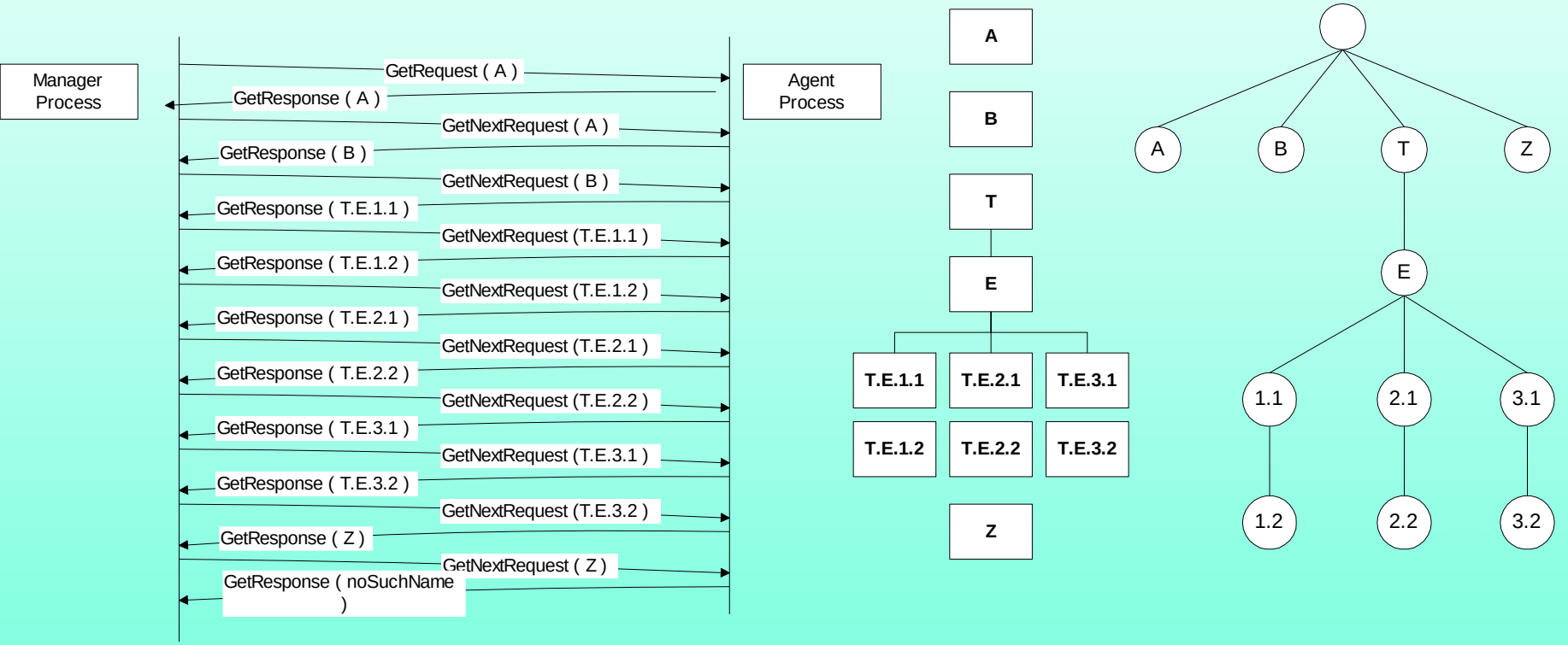


Figure 5.15 Get-Next-Request Operation for MIB in Figure 5.12

GetNextRequest PDU Operation. A get-next-request operation is very similar to get-request, except that the requested record is the next one to the OBJECT IDENTIFIER specified in the request.

Get-Next-Request Operation

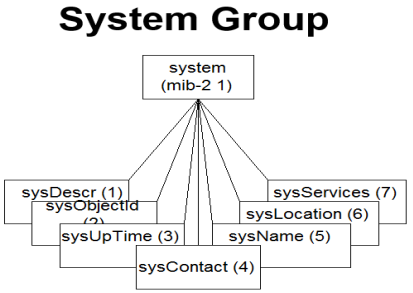
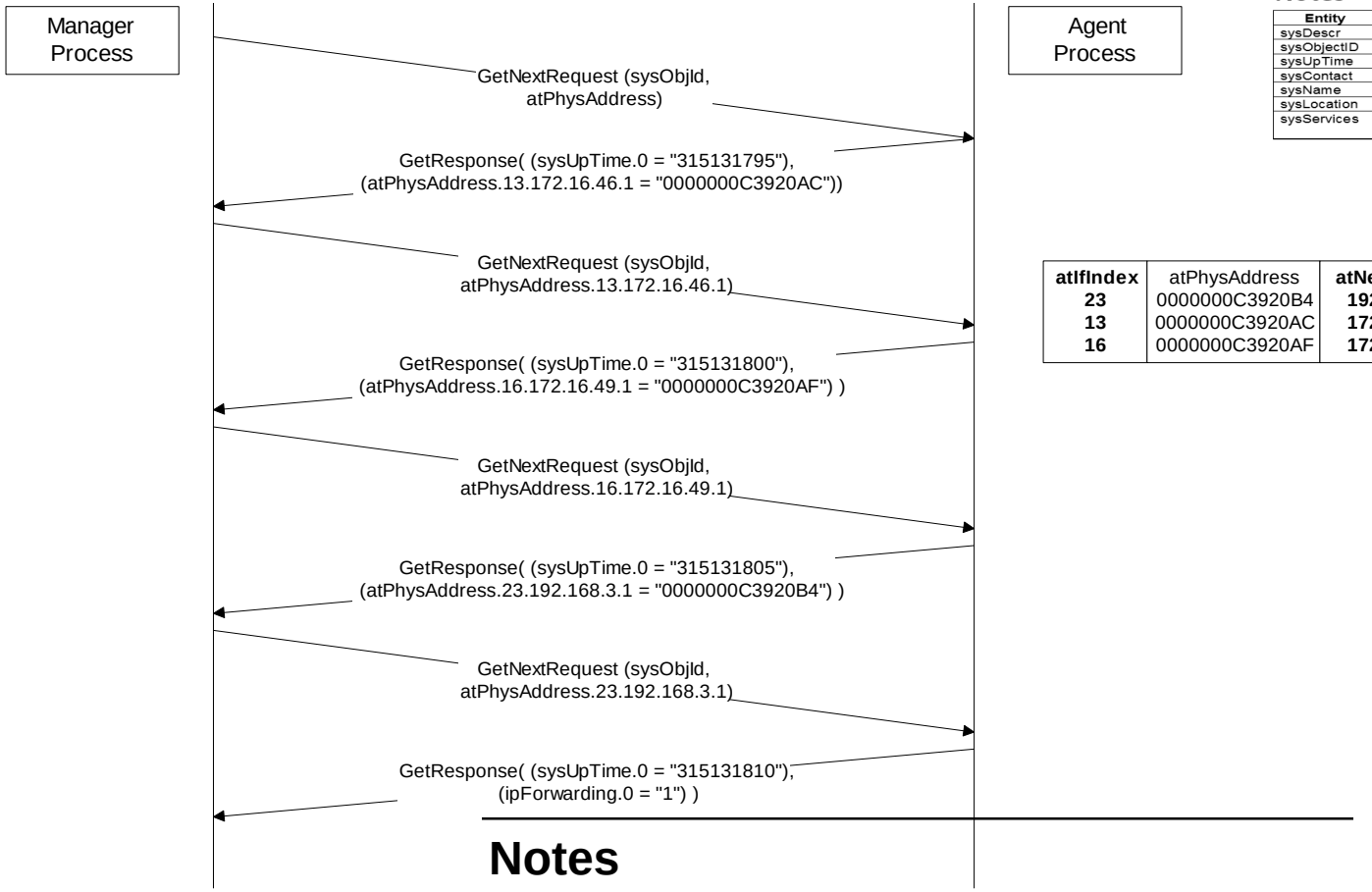


Figure 4.27 System Group

Notes

Entity	OID	Description (brief)
sysDescr	system 1	Textual description
sysObjectID	system 2	OBJECT IDENTIFIER of the entity
sysUpTime	system 3	Time (in hundredths of a second since last reset)
sysContact	system 4	Contact person for the node
sysName	system 5	Administrative name of the system
sysLocation	system 6	Physical location of the node
sysServices	system 7	Value designating the layer services provided by the entity



atIfIndex	atPhysAddress	atNetAddress
23	0000000C3920B4	192.168.3.1
13	0000000C3920AC	172.16.46.1
16	0000000C3920AF	172.16.49.1

Figure 5.16 GetNextRequest Example with Indices

Sniffer Data

```
14:03:36.788270 noc3.btc.gatech.edu.164 >
noc1.btc.gatech.edu.snmp:
Community = public
GetRequest(111)
Request ID = 4
system.sysDescr.0
system.sysObjectID.0
system.sysUpTime.0
system.sysContact.0
system.sysName.0
system.sysLocation.0
system.sysServices.0
```

Figure 5.17(a) Get-Request Message from Manager-to-Agent

```
14:03:36.798269 noc1.btc.gatech.edu.snmp >
noc3.btc.gatech.edu.164:
Community = public
GetResponse(196)
Request ID = 4
system.sysDescr.0 = "SunOS noc1 5.5.1 Generic_103640-08
sun4u"
system.sysObjectID.0 = E:hp.2.3.10.1.2
system.sysUpTime.0 = 247396453
system.sysContact.0 = "Brandon Rhodes"
system.sysName.0 = "noc1"
system.sysLocation.0 = "BTC NM Lab"
system.sysServices.0 = 72
```

Figure 5.17(b) Get-Response Message from Agent-to-Manager (After)

SNMP MIB

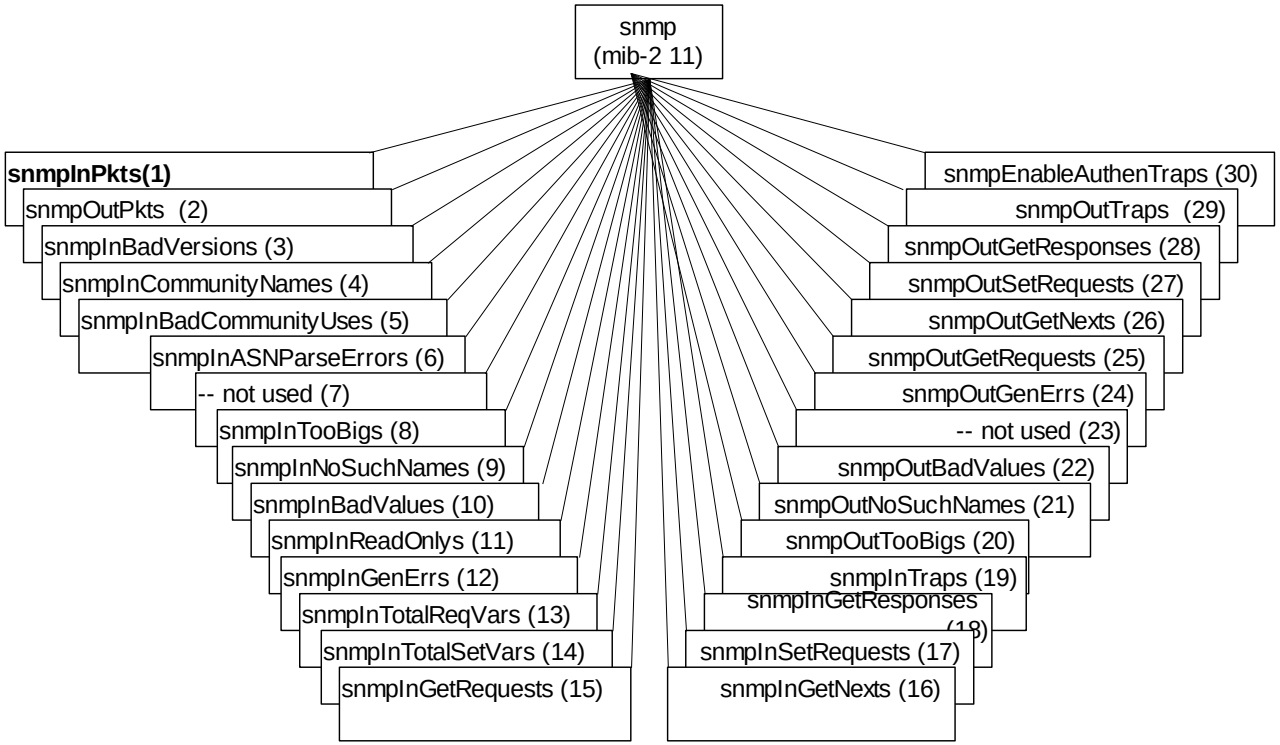


Figure 5.21 SNMP Group

Notes

- SNMPv1 MIB has too many objects that are not used
- SNMPv2 obsoleted a large number of them

Note: Most of the MIB objects were not used and hence deprecated in SNMPv2