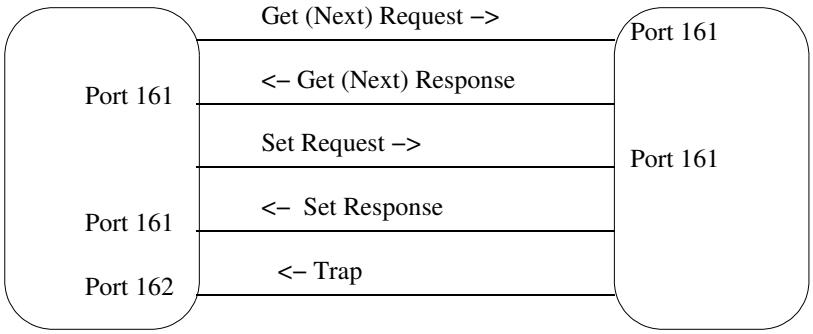
Chapter 25 – SNMP

SNMP = *Simple Network Management Protocol*

A distributed network management protocol

Management Station — Network Element

(Sysadmin's Workstation) — (Host, router, bridge)



SNMP Manager — *SNMP Agent*

The three components of network management with SNMP

1 – The management information base. The collection of state variables that SNMP agents are required to maintain. Specified as MIB – II in RFC 1213

2 – A common structures and an identification scheme used to reference variables in the MIB. Called the SMI (structure of management information), this data is specified in RFC 1155.

3 – A protocol for communications between managers and agents. The protocol is called SNMP and is specified in RFC 1157. (An updated version called SNMP–II adds requests for groups of variables).

All three elements are tied together via a formal language called ASN.1 or ASN.1–BER

ASN.1 – Abstract syntax notation 1
The syntax of the language
BER Basic encoding rules
The rules for encoding it for transmission.

The syntax of the ASN.1 language is Pascal like with basic types including

Integer
String
Object Identifier
Sequence (like a record or structure)
Boolean

Stucture of Management Information:

A tree representing *all* network management conceivably useful in whole world...

The root node itself is unlabeled, but has at least three children

label iso(1) is owned by the ISO
label ccitt(0) is owned by the CCITT
label joint−iso−ccitt(2) is jointly administerd.

The ISO has designated one subtree of iso(1) for use by other international organizations

It is called org(3).
Two of org(3)'s children were assigned to U.S.
One of these subtrees has been transferred by the NIST to the U.S. Department of Defense, dod(6).

RFC 1155 assumes: DoD will allocate a node to the Internet community, to be administered by the Internet Activities Board (IAB) as follows:

internet OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }

That is, the Internet subtree of OBJECT IDENTIFIERs starts with the prefix:

1.3.6.1.

The mgmt(2) subtree is used to identify objects which are defined in IAB−approved documents

The initial Internet standard MIB was assigned management document number 1

Within the standard MIB the following entities are defined

system	(1)
interfaces	(2)
at	(3)
ip	(4)
icmp	(5)
tcp	(6)
udp	(7)

The RFC also defines some derived types:

3.2.3.1. NetworkAddress

This CHOICE represents an address from one of possibly several protocol families. Currently, only one protocol family, the Internet family, is present in this CHOICE.

3.2.3.2. IpAddress

This application−wide type represents a 32−bit internet address. It is represented as an OCTET STRING of length 4, in network byte−order.

3.2.3.3. Counter

This application−wide type represents a non−negative integer which monotonically increases until it reaches a maximum value, when it wraps around and starts increasing again from zero. This memo specifies a maximum value of 2^32−1 (4294967295 decimal) for counters.

3.2.3.4. Gauge

This application−wide type represents a non−negative integer, which may increase or decrease, but which latches at a maximum value. This memo specifies a maximum value of 2^32−1 (4294967295 decimal) for gauges.

3.2.3.5. TimeTicks

This application−wide type represents a non−negative integer which counts the time in hundredths of a second since some epoch. When object types are defined in the MIB which use this ASN.1 type, the description of the object type identifies the reference epoch.

Some others appear in the book.

Basic encoding rules –

The basic encoding rules describe how elements in the language are laid out in messages

Objective –
No ambiguity
No dependence on specific hardware features
Integer length
Byte order
Mechanism –

Heavy use of self–defining fields

Result –

You need a compiler to parse an SNMP request or response!

Object identifiers from the standardard internet MIB:

```
sysDescr                1.3.6.1.2.1.1.1.0        display
sysObjectID             1.3.6.1.2.1.1.2.0        object
sysUpTime               1.3.6.1.2.1.1.3.0        ticks
sysContact              1.3.6.1.2.1.1.4.0        display
ifNumber                1.3.6.1.2.1.2.1.0        number
ifIndex                 1.3.6.1.2.1.2.2.1.1.    number
ifInOctets              1.3.6.1.2.1.2.2.1.10.   counter
ifInUcastPkts           1.3.6.1.2.1.2.2.1.11.   counter
ifInDiscards            1.3.6.1.2.1.2.2.1.13.   counter
ifInErrors              1.3.6.1.2.1.2.2.1.14.   counter
ifInUnknownProtos       1.3.6.1.2.1.2.2.1.15.   counter
ifOutOctets             1.3.6.1.2.1.2.2.1.16.   counter
ifMtu                   1.3.6.1.2.1.2.2.1.4.    number
ifSpeed                 1.3.6.1.2.1.2.2.1.5.    gauge
ifPhysAddress           1.3.6.1.2.1.2.2.1.6.    string
ifAdminStatus           1.3.6.1.2.1.2.2.1.7.    number
ifOperStatus            1.3.6.1.2.1.2.2.1.8.    number
ifLastChange            1.3.6.1.2.1.2.2.1.9.    ticks
atIfIndex               1.3.6.1.2.1.3.1.1.1.    number
atPhysAddress           1.3.6.1.2.1.3.1.1.2.    string
atNetAddress            1.3.6.1.2.1.3.1.1.3.    internet
ipForwarding            1.3.6.1.2.1.4.1.0       number
ipOutRequests           1.3.6.1.2.1.4.10.0      counter
ipOutDiscards           1.3.6.1.2.1.4.11.0      counter
```

Note: Items with trailing 0's are scalars.

To request a scalar, a .0 is appended to its name.

To request a table entry, the proper subscript is appended to its name.

Example of accessing MIB data from an agent:

Syntax varies by implementation.. standard parameters include

–h *hostname*	The name of the host, router, etc that you wish to contact
–c *community*	Basically a password that determines your privileges at that host
command	get, getnext, set,
object identifier	Either by name or by fully qualified id number

```
snmp –h jmw2 –c public get sysDescr.0

sysDescr.0 : OS/2 SNMP AGENT version 1.2, with DPI version 1.1.03
(Jun 2, 1993)

snmp –h hubcap.clemson.edu –c public get sysDescr.0

sysDescr.0 : hubcap DEC3000 – M500 DEC OSF/1 V3.2 (Rev. 214); Fri
Oct  6 15:36:52 EDT 1995  TCP/IP
```

Equivalently one could specify the actual name in the SMI tree of the sysDescr data

```
snmp –h jmw2 –c public get 1.3.6.1.2.1.1.1.0
sysDescr.0 : OS/2 SNMP AGENT version 1.2, with DPI version 1.1.03
(Jun 2, 1993)
```

SNMP Requests with Linux

```
==> snmpget glint3 public system.sysDescr.0
system.sysDescr.0 = "Linux glint3.cs.clemson.edu 2.2.9 #5 SMP Fri
Mar 3 17:12:44 EST 2000 i686"
==> snmpget glint3 public 1.1.0
system.sysDescr.0 = "Linux glint3.cs.clemson.edu 2.2.9 #5 SMP Fri
Mar 3 17:12:44 EST 2000 i686"
```

Transmission of SNMP messages:

UDP port 161
Request/Response
port 162
Trap (agent needs to inform management station of event).

Packet Layout of SMP Messages

IP header
UDP header
SNMP data (Actual packet format appears at variance with layout in book!)

```
---------------------- UDP HEADER ----------------------
 UDP:  Source Port: 1473     Dest Port: 161
 UDP:  Length: 51 (x33)
 UDP:  Checksum: AD38
--------------------------------- DATA ---------------------------------
0000 30 29 02 01 00 04 06 70    75 62 6C 69 63 A0 1C 02   0).....public...
0010 04 30 C3 2E 56 02 01 00    02 01 00 30 0E 30 0C 06   .0..V......0.0..
0020 08 2B 06 01 02 01 01 01    00 05 00                  .+.........
```

Elements of this request:

30 – The code for sequence
29 – The length of the sequence
02 – The code for integer
01 – The length of the integer (BER says 2's complement, big endian)
00 – The version number (actually 1) of SNMP
04 – The code for string
06 – The length of the string
70–63 – The string "public" (a community name = password)
A0 – The code for get request
1C – The length of the get request
02 – The code for integer
04 – The length of the integer
30–56 – The request id code (used for mapping requests to responses)
02 – The code for integer
01 – A one byte integer
00 – The error status
02 – The code for integer
01 – A one byte integer
00 – The error index
30 – The code for sequence
0C – The length of this sequence
06 – The code for object identifier
08 – The length of the object identifier
2B–00 – The object identifier for sysDescr (The 2B is a compressed version of 1.3)
(The encoding scheme is 40x + y where x and y are the 1st two fields)
05 – The code for a null object
00 – A length of 0

The response from the sysDescr query

```
---------------------- UDP HEADER ----------------------
 UDP:  Source Port: 161     Dest Port: 1473
 UDP:  Length: 117 (x75)
 UDP:  Checksum: F2CA
--------------------------------- DATA ---------------------------------
0000 30 6B 02 01 00 04 06 70    75 62 6C 69 63 A2 5E 02   0k.....public.^.
0010 04 30 C3 2E 56 02 01 00    02 01 00 30 50 30 4E 06   .0..V......0P0N.
0020 08 2B 06 01 02 01 01 01    00 04 42 4F 53 2F 32 20   .+........BOS/2
0030 53 4E 4D 50 20 41 47 45    4E 54 20 76 65 72 73 69   SNMP AGENT versi
0040 6F 6E 20 31 2E 32 2C 20    77 69 74 68 20 44 50 49   on 1.2, with DPI
0050 20 76 65 72 73 69 6F 6E    20 31 2E 31 2E 30 33 20    version 1.1.03
```

Processing tables of values:

Some items are naturally stored as variable size tables

The ARP cache
The IP routing table

These tables of values can be retrieved using *getnext*
You simply supply the last value you received at each new request.

Example:
The ARP cache at *jmw3* displayed by *Arp*

```
[L:\] ==> arp -a
          ARP table contents:

interface     hardware address          IP address   minutes since
                                                        last use
    0           0800202195dd          130.127.48.1       0
    0           02608c2adf8c          130.127.48.113     7
    0           080020182467          130.127.48.24     12
    0           0800207353b3          130.127.48.144     17
```

The network address part of *jmw3's* arp cache displayed by snmp
Note that the "answer" is always part of the question.

```
snmp -h jmw3 -c public getnext atNetAddress
atNetAddress.1.1.130.127.48.1 : 130.127.48.1
snmp -h jmw3 -c public getnext atNetAddress.1.1.130.127.48.1
atNetAddress.1.1.130.127.48.24 : 130.127.48.24
snmp -h jmw3 -c public getnext atNetAddress.1.1.130.127.48.24
atNetAddress.1.1.130.127.48.113 : 130.127.48.113
snmp -h jmw3 -c public getnext atNetAddress.1.1.130.127.48.113
atNetAddress.1.1.130.127.48.144 : 130.127.48.144
snmp -h jmw3 -c public getnext atNetAddress.1.1.130.127.48.144
ipForwarding.0 : 1   <--- Oops, end of table!
```

The physical address part of *jmw's* arp cache displayed by snmp

```
snmp -h jmw -c public getnext atPhysAddress
atPhysAddress.1.1.130.127.48.1 : 08:00:20:21:95:dd
snmp -h jmw -c public getnext atPhysAddress.1.1.130.127.48.1
atPhysAddress.1.1.130.127.48.2 : 08:00:20:73:f9:a8
snmp -h jmw -c public getnext atPhysAddress.1.1.130.127.48.2
atPhysAddress.1.1.130.127.48.87 : 08:00:69:07:e5:fd
snmp -h jmw -c public getnext atPhysAddress.1.1.130.127.48.87
atPhysAddress.1.1.130.127.48.118 : 00:20:af:0f:6f:3c
snmp -h jmw -c public getnext atPhysAddress.1.1.130.127.48.118
atPhysAddress.1.1.130.127.48.137 : 08:00:20:04:19:f7
snmp -h jmw -c public getnext atPhysAddress.1.1.130.127.48.137
atPhysAddress.1.1.130.127.48.144 : 08:00:20:73:53:b3
snmp -h jmw -c public getnext atPhysAddress.1.1.130.127.48.144
atNetAddress.1.1.130.127.48.1 : 130.127.48.1
```

jmw's route table dumped via SNMP

```
[L:\] ==> snmp    -h jmw -c public getnext ipRouteDest
ipRouteDest.0.0.0.0 : 0.0.0.0
[L:\] ==> snmp    -h jmw -c public getnext ipRouteDest.0
ipRouteDest.0.0.0.0 : 0.0.0.0
[L:\] ==> snmp    -h jmw -c public getnext ipRouteDest.0.0.0.0
ipRouteDest.130.127.48.0 : 130.127.48.0
[L:\] ==> snmp    -h jmw -c public getnext ipRouteDest.130.127.48.0
ipRouteDest.130.127.66.2 : 130.127.66.2
[L:\] ==> snmp    -h jmw -c public getnext ipRouteDest.130.127.66.2
ipRouteDest.130.127.66.3 : 130.127.66.3
[L:\] ==> snmp    -h jmw -c public getnext ipRouteDest.130.127.66.3
ipRouteDest.130.127.66.9 : 130.127.66.9
[L:\] ==> snmp    -h jmw -c public getnext ipRouteDest.130.127.66.9
ipRouteDest.130.127.66.10 : 130.127.66.10
[L:\] ==> snmp    -h jmw -c public getnext ipRouteDest.130.127.66.10
ipRouteDest.130.127.66.11 : 130.127.66.11
[L:\] ==> snmp    -h jmw -c public getnext ipRouteDest.130.127.66.11
ipRouteDest.130.127.66.12 : 130.127.66.12
```

Group processing in SNMP – II

```
[L:\] ==> snmpgrp -h jmw3 ip
IP group ------------------------------

          Forwarding:  1
          DefaultTTL:  30
       ipInReceives:  444610
      ipInHdrErrors:  0
     ipInAddrErrors:  0
    ipForwDatagrams:  0
  ipInUnknownProtos:  16
       ipInDiscards:  0
       ipInDelivers:  439710
        OutRequests:  102428
        OutDiscards:  0
        OutNoRoutes:  0
       ReasmTimeout:  0
         ReasmReqds:  6250
           ReasmOKs:  1367
         ReasmFails:  1
            FragOKs:  710
          FragFails:  0
        FragCreates:  3220

End of group ip -----------------
```

Note that OS/2's group retriever doesn't get tables associated with the group

```
[D:\] ==> snmpgrp -h jmw3 tcp

TCP group-------------------------
    RtoAlgorithm:  4
          RtoMin:  2000
          RtoMax:  128000
         MaxConn:  255
     ActiveOpens:  450
    PassiveOpens:  7
    AttemptFails:  0
     EstabResets:  454
       CurrEstab:  3
          InSegs:  126722
         OutSegs:  160938
     RetransSegs:  1827
           InErr:  8
         OutRsts:  0
                 :  26

End of group tcp ----------------
```

Group processing in Linux

```
bash# snmpbulkwalk -v 2 jmw3 noAuth interfaces

interfaces.ifNumber.0 = 3
interfaces.ifTable.ifEntry.ifIndex.1 = 1
interfaces.ifTable.ifEntry.ifIndex.2 = 2
interfaces.ifTable.ifEntry.ifIndex.3 = 3
interfaces.ifTable.ifEntry.ifDescr.1 = "lo0" Hex: 6C 6F 30
interfaces.ifTable.ifEntry.ifDescr.2 = "plip0"
interfaces.ifTable.ifEntry.ifDescr.3 = "eth0" Hex: 65 74 68 30
interfaces.ifTable.ifEntry.ifType.1 = softwareLoopback(24)
interfaces.ifTable.ifEntry.ifType.2 = other(1)
interfaces.ifTable.ifEntry.ifType.3 = ethernet-csmacd(6)
interfaces.ifTable.ifEntry.ifMtu.1 = 3584
interfaces.ifTable.ifEntry.ifMtu.2 = 1500
interfaces.ifTable.ifEntry.ifMtu.3 = 1500
interfaces.ifTable.ifEntry.ifSpeed.1 = Gauge: 20000000
interfaces.ifTable.ifEntry.ifSpeed.2 = Gauge: 0
interfaces.ifTable.ifEntry.ifSpeed.3 = Gauge: 10000000

bash# snmpbulkwalk -v 2 jmw3 noAuth tcp

tcp.tcpRtoAlgorithm.0 = other(1)
tcp.tcpRtoMin.0 = 0
tcp.tcpRtoMax.0 = 0
tcp.tcpMaxConn.0 = 0
tcp.tcpActiveOpens.0 = 6
tcp.tcpPassiveOpens.0 = 0
tcp.tcpAttemptFails.0 = 0
tcp.tcpEstabResets.0 = 0
tcp.tcpCurrEstab.0 = Gauge: 3
tcp.tcpInSegs.0 = 8992
tcp.tcpOutSegs.0 = 9919
tcp.tcpRetransSegs.0 = 0
```

In linux bulkwalk does retreive the tables.

```
tcp.tcpConnTable.tcpConnEntry.tcpConnState.0.0.0.0.7.0.0.0.0.0 = listen(2)
tcp.tcpConnTable.tcpConnEntry.tcpConnState.0.0.0.0.9.0.0.0.0.0 = listen(2)
tcp.tcpConnTable.tcpConnEntry.tcpConnState.0.0.0.0.11.0.0.0.0.0 = listen(2)
       :
tcp.tcpConnTable.tcpConnEntry.tcpConnState.130.127.48.118.1023.130.127.48.24.513
= established(5)
tcp.tcpConnTable.tcpConnEntry.tcpConnState.130.127.48.118.6000.130.127.48.24.3978
2 = established(5)
tcp.tcpConnTable.tcpConnEntry.tcpConnState.130.127.48.118.6000.130.127.48.24.3978
3 = established(5)
       :
tcp.tcpConnTable.tcpConnEntry.tcpConnLocalAddress.0.0.0.0.7.0.0.0.0.0 =
IpAddress: 0.0.0.0
tcp.tcpConnTable.tcpConnEntry.tcpConnLocalAddress.0.0.0.0.9.0.0.0.0.0 =
IpAddress: 0.0.0.0
tcp.tcpConnTable.tcpConnEntry.tcpConnLocalAddress.0.0.0.0.11.0.0.0.0.0 =
IpAddress: 0.0.0.0
       :
tcp.tcpConnTable.tcpConnEntry.tcpConnLocalAddress.130.127.48.118.1023.130.127.48.
24.513 = IpAddress: 130.127.48.118
tcp.tcpConnTable.tcpConnEntry.tcpConnLocalAddress.130.127.48.118.6000.130.127.48.
24.39782 = IpAddress: 130.127.48.118
tcp.tcpConnTable.tcpConnEntry.tcpConnLocalAddress.130.127.48.118.6000.130.127.48.
24.39783 = IpAddress: 130.127.48.118
```

(Here again the answer is part of the question!)

MIB – II Object definitions

The start of the MIB identifies:
The internets place in the hierarchy and
The sub categories of internet

```
RFC1155-SMI DEFINITIONS ::= BEGIN;
    nullOID       OBJECT IDENTIFIER ::= { ccitt 0 }
    internet      OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
    directory     OBJECT IDENTIFIER ::= { internet 1 }
    mgmt          OBJECT IDENTIFIER ::= { internet 2 }
    experimental  OBJECT IDENTIFIER ::= { internet 3 }
    private       OBJECT IDENTIFIER ::= { internet 4 }
    enterprises   OBJECT IDENTIFIER ::= { private 1 }
END
```

This is followed by the MIB definition which begins by Importing these defined types

```
RFC1213-MIB DEFINITIONS ::= BEGIN
     IMPORTS
        mgmt, NetworkAddress, IpAddress, Counter, Gauge, TimeTicks
                          FROM RFC1155-SMI
                OBJECT-TYPE
                          FROM RFC-1212;

       --  This MIB module uses the extended OBJECT-TYPE macro as
       --  defined in [14];

       --  MIB-II (same prefix as MIB-I)

         mib-2      OBJECT IDENTIFIER ::= { mgmt 1 }
```

And defines mib–2 (with the same value as Mib–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 }
          icmp         OBJECT IDENTIFIER ::= { mib-2 5 }
          tcp          OBJECT IDENTIFIER ::= { mib-2 6 }
          udp          OBJECT IDENTIFIER ::= { mib-2 7 }
          egp          OBJECT IDENTIFIER ::= { mib-2 8 }
          -- historical (some say hysterical)
          -- cmot      OBJECT IDENTIFIER ::= { mib-2 9 }
```

The remainder of the MIB contains object definitions:
The general form is

```
obj_name OBJECT-TYPE
SYNTAX     data-type
ACCESS     read-only or read-write
STATUS     mandatory, deprecated, current, etc.
DESCRIPTION descriptive statement
::= { parent index}
```

```
      -- the System group
      -- Implementation of the System group is mandatory for all
      -- systems.  If an agent is not configured to have a value
      -- for any of these variables, a string of length 0 is
      -- returned.

          sysDescr OBJECT-TYPE
              SYNTAX  DisplayString (SIZE (0..255))
              ACCESS  read-only
              STATUS  mandatory
              DESCRIPTION
               "A textual description of the entity.  This value
                should include the full name and version
                identification of the system's hardware type,
                software operating-system, and networking
                software.  It is mandatory that this only contain
                printable ASCII characters."
           ::= { system 1 }
```

Table definitions are uglier still..

```
     -- the IP address table
     -- The IP address table contains this entity's IP addressing
     -- information.

     ipAddrTable OBJECT-TYPE
         SYNTAX  SEQUENCE OF IpAddrEntry
         ACCESS  not-accessible
         STATUS  mandatory
         DESCRIPTION
               "The table of addressing information relevant to
                this entity's IP addresses."
         ::= { ip 20 }
```

```
ipAddrEntry OBJECT-TYPE
    SYNTAX  IpAddrEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "The addressing information for one of this
                 entity's IP addresses."
    INDEX  {ipAdEntAddr}
    ::= { ipAddrTable 1 }
```

Note here the subtle but crucial difference in the use of the names

`ipAddrEntry` and
`IpAddrEntry`

The latter is simply used to define `ipAddrEntry` as a sequence itself and does *not* have a place in the MIB hierarchy

```
IpAddrEntry ::=
    SEQUENCE {
         ipAdEntAddr
            IpAddress,
         ipAdEntIfIndex
            INTEGER,
         ipAdEntNetMask
            IpAddress,
         ipAdEntBcastAddr
            INTEGER,
         ipAdEntReasmMaxSize
            INTEGER (0..65535)
         }

ipAdEntAddr OBJECT-TYPE
    SYNTAX  IpAddress
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
           "The IP address to which this entry's addressing
           information pertains."
    ::= { ipAddrEntry 1 }

ipAdEntIfIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
          "The index value which uniquely identifies the
           interface to which this entry is applicable.  The
           interface identified by a particular value of this
           index is the same interface as identified by the
           same value of ifIndex."
         ::= { ipAddrEntry 2 }
```

```
      ipAdEntNetMask OBJECT-TYPE
         SYNTAX  IpAddress
         ACCESS  read-only
         STATUS  mandatory
         DESCRIPTION
                "The subnet mask associated with the IP address of
                 this entry.  The value of the mask is an IP
                 address with all the network bits set to 1 and all
                 the hosts bits set to 0."
               ::= { ipAddrEntry 3 }

#snmpwalk 130.127.48.1 public ip.ipAddrTable

ip.ipAddrTable.ipAddrEntry.ipAdEntAddr.130.127.48.1 = IpAddress: 130.127.48.1
ip.ipAddrTable.ipAddrEntry.ipAdEntAddr.130.127.56.1 = IpAddress: 130.127.56.1
ip.ipAddrTable.ipAddrEntry.ipAdEntAddr.130.127.87.2 = IpAddress: 130.127.87.2

ip.ipAddrTable.ipAddrEntry.ipAdEntIfIndex.130.127.48.1 = INTEGER: 1
ip.ipAddrTable.ipAddrEntry.ipAdEntIfIndex.130.127.56.1 = INTEGER: 20
ip.ipAddrTable.ipAddrEntry.ipAdEntIfIndex.130.127.87.2 = INTEGER: 3

ip.ipAddrTable.ipAddrEntry.ipAdEntNetMask.130.127.48.1 = IpAddress: 255.255.255.0
ip.ipAddrTable.ipAddrEntry.ipAdEntNetMask.130.127.56.1 = IpAddress: 255.255.255.0
ip.ipAddrTable.ipAddrEntry.ipAdEntNetMask.130.127.87.2 = IpAddress: 255.255.255.0

ip.ipAddrTable.ipAddrEntry.ipAdEntBcastAddr.130.127.48.1 = INTEGER: 1
ip.ipAddrTable.ipAddrEntry.ipAdEntBcastAddr.130.127.56.1 = INTEGER: 1
ip.ipAddrTable.ipAddrEntry.ipAdEntBcastAddr.130.127.87.2 = INTEGER: 1

ip.ipAddrTable.ipAddrEntry.5.130.127.48.1 = INTEGER: 0
ip.ipAddrTable.ipAddrEntry.5.130.127.56.1 = INTEGER: 0
ip.ipAddrTable.ipAddrEntry.5.130.127.87.2 = INTEGER: 0
[root@glint2 bin]
```

Another example is the AT group..

```
          -- the Address Translation group

              atTable OBJECT-TYPE
                      SYNTAX  SEQUENCE OF AtEntry
                      ACCESS  read-write
                      STATUS  mandatory
                      ::= { at 1 }

              atEntry OBJECT-TYPE
                      SYNTAX  AtEntry
                      ACCESS  read-write
                      STATUS  mandatory
                      INDEX   {atIfIndex, atNetAddress}
                      ::= { atTable 1 }

              AtEntry ::= SEQUENCE {
                      atIfIndex
                          INTEGER,
                      atPhysAddress
                          OCTET STRING,
                      atNetAddress
                          NetworkAddress
                      }

              atIfIndex OBJECT-TYPE
                      SYNTAX  INTEGER
                      ACCESS  read-write
                      STATUS  mandatory
                      ::= { atEntry 1 }

              atPhysAddress OBJECT-TYPE
                      SYNTAX  OCTET STRING
                      ACCESS  read-write
                      STATUS  mandatory
                      ::= { atEntry 2 }

              atNetAddress OBJECT-TYPE
                      SYNTAX  NetworkAddress
                      ACCESS  read-write
                      STATUS  mandatory
                      ::= { atEntry 3 }
```

```
home/westall ==> snmpwalk glint2 public at
at.atTable.atEntry.atIfIndex.2.1.130.127.48.118 = 2
at.atTable.atEntry.atIfIndex.2.1.130.127.48.144 = 2
at.atTable.atEntry.atPhysAddress.2.1.130.127.48.118 =  Hex: 00 20 AF 0F 6F 3C
at.atTable.atEntry.atPhysAddress.2.1.130.127.48.144 =  Hex: 08 00 20 88 5E 9E
at.atTable.atEntry.atNetAddress.2.1.130.127.48.118 = IpAddress: 130.127.48.118
at.atTable.atEntry.atNetAddress.2.1.130.127.48.144 = IpAddress: 130.127.48.144
home/westall ==>
```