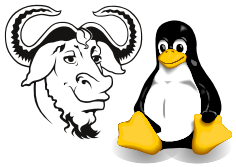



# The Structure of Management Information (SMI)

## 1 Background

The SNMP protocol is called "simple" because the protocol itself is quite simple. However, the difficulty is in applying it to actually managing systems and networks.

There are many terms and standards involved; it is necessary to understand enough of them to make sense of the MIBs that define the objects that you want to monitor and manage. If you can make sense of the MIB files, you can identify the objects that you want to monitor.

### 1.1 Management Information Base (MIB)

The MIBs define the objects that you can manage.

When you installed the Net SNMP software package, you installed some MIB files into the directory `/usr/share/snmp/mibs/`. You can list them all with:

```
$ rpm -ql ucd-snmp | grep snmp/mibs/.*\.txt
```

There are many other MIBs that are not included here; you can download others from somewhere such as `http://www.simpleweb.org/ietf/` and include them into your Net SNMP clients as explained at `http://net-snmp.sourceforge.net/FAQ.html#How_do_`

I_add_a_MIB_ and at http://net-snmp.sourceforge.net/tutorial/commands/mib-options.html.

### 1.2 Structure of Management Information

SMI is a definition of the structure of the MIBs, how they are connected together into a tree. See the RFCs below in section 2.1 on page 10. It specifies which part of ASN.1 will be used to define MIBs.

### 1.3 Abstract Syntax Notation One (ASN.1)

ASN.1 is widely used for many things other than SNMP. See http://asn1.elibel.tm.fr/en/uses/ for a list of some of the applications of ASN.1. There is a web site dedicated to providing information about it at http://asn1.elibel.tm.fr/.

### 1.4 Basic Encoding Rules (BER)

The *basic encoding rules* is an ISO standard. It describes a method for encoding values of each ASN.1 type as a string of octets.

### 1.5 What we are doing today

We will examine the specification for mib-2, on your machine at /usr/share/snmp/mibs/RFC1213-MIB.txt and understand the structure of it.

### 1.6 Syntax of a Managed Object Definition

Every object definition in SMI has the format:

```
name OBJECT-TYPE
    SYNTAX datatype
    ACCESS either read-only, read-write, write-on
    DESCRIPTION
        "Some text that describes this managed obj
    ::= { unique object ID that defines this obje
```

We will refer to this later in our activities.

## 2 The MIB-II Definition

Here I will refer to my edited version of `RFC1213-MIB.txt`, available at `http://nicku.org/snm/lectures/smi/RFC1213-MIB.txt`.

```
RFC1213-MIB DEFINITIONS ::= BEGIN

IMPORTS
        mgmt, NetworkAddress, IpAddress, Counter,
            TimeTicks
        FROM RFC1155-SMI
        OBJECT-TYPE
            FROM RFC-1212;
```

The first line defines the name of the MIB, here `RFC1213-MI`. The format of this definition is always the same.

The `IMPORTS` section of the MIB is sometimes called the *linkage* section. It lets you import definitions of

datatypes and OIDs from other MIBs. Here we get the definition of:

- `mgmt`
- `NetworkAddress`
- `IpAddress`
- `Counter`
- `Gauge`
- `TimeTicks`

from `RFC1155-SMI`, the MIB from the RFC that defines SMIv1.

It also imports `OBJECT-TYPE` from `RFC-1212`, the *Concise MIB Definition*, which defines how MIB files are written.

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

The line above says that the OID of `mib-2` is 1.3.6.1.2.1. `RFC1155-SMI` defines `mgmt` as the OID 1.3.6.1.2.

```
-- 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 }
transmission OBJECT IDENTIFIER ::= { mib-2 10 }
snmp         OBJECT IDENTIFIER ::= { mib-2 11 }
```

So here the `system` group is defines as the OID 1.3.6.1.2.1.1, and so on. A comment is a line starting with `--`.

```
-- the Interfaces table

-- The Interfaces table contains information on th
-- interfaces.  Each interface is thought of as be
-- attached to a 'subnetwork'.  Note that this ter
-- not be confused with 'subnet' which refers to a
-- addressing partitioning scheme used in the Inte
-- of protocols.

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

This is the first managed object shown here. `ifTable` represents a table of network interfaces on a managed device. Notice that object names are defined with mixed case, the first letter is lowercase.

Notice that this follows the format of an `OBJECT-TYPE` in section 1.6 on page 3.

The `SYNTAX` of `ifTable` is `SEQUENCE OF IfEntry`. The object is `not-accessible`, which means that you cannot query the agent for the value of this object. It has a `STATUS` of `mandatory`, which means that if an agent complies wiht the MIBB-II specification, then it must implement this object. The `DESCRIPTION` tells you what this object is. The unique OID is 1.3.6.1.2.1.2.2, or `iso.org.dod.internet.mgmnt.interfaces.2`.

Next, let's look at the `SEQUENCE` definition, which is used with the `SEQUENCE OF` type in the `ifTable` definition.

```
IfEntry ::=
    SEQUENCE {
        ifIndex
            INTEGER,
        ifDescr
            DisplayString,
        ifType
            INTEGER,
        ifMtu
            INTEGER,
        ifSpeed
            Gauge,
        ifPhysAddress
            PhysAddress,
        ifAdminStatus
            INTEGER,
        ifOperStatus
            INTEGER,
```

```
        ifLastChange
            TimeTicks,
        ifInOctets
            Counter,
        ifInUcastPkts
            Counter,
        ifInNUcastPkts
            Counter,
        ifInDiscards
            Counter,
        ifInErrors
            Counter,
        ifInUnknownProtos
            Counter,
        ifOutOctets
            Counter,
        ifOutUcastPkts
            Counter,
        ifOutNUcastPkts
            Counter,
        ifOutDiscards
            Counter,
        ifOutErrors
            Counter,
        ifOutQLen
            Gauge,
        ifSpecific
            OBJECT IDENTIFIER
    }
```

The name of the `SEQUENCE` (`IfEntry`) is mixed-case, but the first letter is capitalised, which is different from the object definition for `ifTable`. A `SEQUENCE` is a list of objects that go into one row of a table. After this, we must have `OBJECT-TYPE` definitions that define each of these variables. A table can have any number of rows. The agent manages the number of rows. An NMS can also add rows to a table using a *set* operation.

`IfEntry` is the data type; rather like a `struct` definition in the C language.

Let's look at `ifEntry`, the definition of what we find in the table, the actual rows of the table themselves. It looks almost the same as the definition for `ifTable`, except that it has a new clause, `INDEX`. The index is a unique value that identifies a single row in the table, like an array index. A table is rather like an array of `struct`s in C. The agent assigns these index values. If a router has eight interfaces, then `ifTable` will contain eight rows.

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

Here we now look at the definition for **ifIndex**, the first item in **IfEntry**. Notice that indexes start from 1.

```
ifIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "A unique value for each interface.  I
            ranges between 1 and the value of ifNu
            value for each interface must remain c
            least from one re-initialization of th
            network management system to the next
            initialization."
    ::= { ifEntry 1 }
```

This object is **read-only**, which means that you can see the value, but not change it.

Here is the last object we look at from this table:

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

```
END
```

`ifDescr` is just a textual description of the interface.

The MIB definition finishes with `END`.

### 2.1 Where can I get the standards documents from?

The standard for SMIv1 can be downloaded from `ftp://ftp.rfc-editor.org/in-notes/rfc1155.txt`, and for —SMIv2 at `ftp://ftp.rfc-editor.org/in-notes/rfc2578.txt`. The standards for ASN.1 and BER can be downloaded from `http://asn1.elibel.tm.fr/en/standards/`.