

MIB Samples

RFC 1155 SMI

```
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
```

RFC 1213 MIB II

```
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 }

    -- textual conventions

    DisplayString ::=
        OCTET STRING
    -- This data type is used to model textual information taken
    -- from the NVT ASCII character set.  By convention, objects
    -- with this syntax are declared as having

    --
    --      SIZE (0..255)

    PhysAddress ::=
        OCTET STRING
    -- This data type is used to model media addresses.  For many
    -- types of media, this will be in a binary representation.
    -- For example, an ethernet address would be represented as
    -- a string of 6 octets.

    -- 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 }

transmission OBJECT IDENTIFIER ::= { mib-2 10 }

snmp OBJECT IDENTIFIER ::= { mib-2 11 }

-- 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 }

sysObjectID OBJECT-TYPE
    SYNTAX OBJECT IDENTIFIER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The vendor's authoritative identification of the
        network management subsystem contained in the
        entity. This value is allocated within the SMI
        enterprises subtree (1.3.6.1.4.1) and provides an
        easy and unambiguous means for determining `what
        kind of box' is being managed. For example, if
        vendor `Flintstones, Inc.' was assigned the
        subtree 1.3.6.1.4.1.4242, it could assign the
```

```
        identifier 1.3.6.1.4.1.4242.1.1 to its `Fred
        Router'."
 ::= { system 2 }

sysUpTime OBJECT-TYPE
    SYNTAX  TimeTicks
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The time (in hundredths of a second) since the
        network management portion of the system was last
        re-initialized."
 ::= { system 3 }

sysContact OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "The textual identification of the contact person
        for this managed node, together with information
        on how to contact this person."
 ::= { system 4 }

sysName OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))

    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "An administratively-assigned name for this
        managed node.  By convention, this is the node's
        fully-qualified domain name."
 ::= { system 5 }

sysLocation OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "The physical location of this node (e.g.,
        `telephone closet, 3rd floor')."
 ::= { system 6 }
```

Commands to use on a Cisco router:

```
snmp-server community community-name RO
```

Sets the read-only SNMP v1 community name

```
snmp-server community community-name RW
```

Sets the read-write SNMP v1 community name

```
snmp-server host ip-address community-name
```

Sets the ip address of station capable of receiving SNMP traps (i.e. network management station)

```
snmp-server enable traps
```

Enable transmission of SNMP traps to the network management station.

Commands on a Linux host (from the UCD-SNMP package):

```
snmptrapd -P
```

Starts an SNMP trap receiving daemon, redirecting its output to the terminal. (See **man snmptrapd** for more details).

```
snmpstatus -v 1 -c community-name managed-object-ip-address
```

Gives some introductory information about the managed object. All other SNMP related commands will very similar syntax (version, community).

```
snmpnetstat netstat-options -v 1 -c community-name ip-address
```

Reports data from the managed object in a form resembling output of a UNIX system `netstat` command. Useful **netstat-options** are:

```
snmpnetstat -ni          Report all network interfaces of the system  
snmpnetstat -rn          Show the routing table
```

Marcin Cieślak

```
snmpget [ -m all ] -v 1 -c community-name ip-address  
variable-name
```

Fetches and reports the value of a particular MIB variable, where the name can be given in the dot numerical format or as a name. To use names, a particular MIB file has to be installed on the system in the /usr/share/snmp/mibs directory. If you have installed your own MIB files there, add the `-m all` option to at the beginning of the command. Variables can be referred to as the names relative to the `.iso.org.dod.internet.mgmt (.1.3.6.1.2)` node.

Some examples:

```
snmpget -v 1 -c public 10.0.0.1 sysContact.0  
snmpget -v 1 -c public 10.0.0.1  
.iso.org.dod.internet.mgmt.mib-2.system.sysContact.0  
  
snmpget -v 1 -c public 10.0.0.1 .1.3.6.1.2.1.1.4  
snmpget -v 1 -c public 10.0.0.1 mib-2.1.4
```

All of the above will fetch the value of **sysContact** variable from the standard MIB-II.

```
snmpset [ -m all ] -v 1 -c community-name ip-address  
variable-name variable-type value
```

Modifies the value of the given MIB variable if it is not read-only.

All remarks regarding `snmpget` command still apply.

Please note, that the type has to be specified after the variable name. Please consult the manpage (`man snmpset`) for the list of available types.

Example:

```
snmpset -v 1 -c public 10.0.0.1 sysContact.0 s Sysadmin  
Set the value of the sysContact variable to the octet-string containing ASCII characters "Sysadmin".
```

```
snmpwalk [ -m all ] -v 1 -c community-name ip-address  
variable-name
```

Reports a list of variables starting from the node "variable-name".

This can be used to download almost whole MIB tree or values of different tables.

This command uses SNMP **get-next** request to get the whole list of variables.

Lab Exercises

For all exercises, the group should choose at least one management station. You may configure multiple management stations, to manage every router from a separate PC. You may also configure every router to communicate with multiple management stations.

Exercise 1.

Start **snmptrapd** daemon on the Linux box to receive SNMP messages from routers.

Verify the IP connectivity between your management station and the router.

Configure your cisco router to:

- accept SNMP requests from any machine with a community **ceenet**
- the access to the router should be read-only.
- send SNMP traps to the management station running **snmptrapd**. You may repeat commands for multiple management stations if you want.

Check the basic connectivity with a **snmpstatus** command and see if you can get some information from a Cisco router.

Exercise 2.

Check the routing table and network interface table on a router.

Check the routing table and network interface table on a Linux box (with a plain **netstat -rn** and **netstat -in** commands).

Please note the similarities and differences.

Exercise 3.

Disconnect the Ethernet cable of the router (but **not** the one connecting you to the management station), and see if there are any trap messages coming in the **snmptrapd** window. Reconnect the cable again.

Can you try to interpret the messages?

Exercise 4.

Use the **snmpwalk** command to fetch the subtree fragment starting from variable .1.3.6.1.2.1.14. Save the result to a file.

See if you can easily interpret the results.

Please note that SNMP package was able to recognize the structure of the information.

Exercise 5.

Install OSPF MIB from the /mnt/cisco-v1 directory. Copy the MIB file to the

/usr/share/snmp/mibs directory.

Repeat exercise 4, giving the **-m all** extra option to the **snmpwalk** command.

Please not the difference.

Use the single **snmpget** command to get the OSPF router ID of your router. (Consult the output of **snmpwalk** command and/or MIB file to determine the exact variable name).

Try to issue **snmpwalk** command giving **mib-2.ospf** as a variable name instead of numerical access.

Exercise 6.

Use **snmpget** command to fetch values of **sysContact** and **sysName** MIB variables.

Update your Cisco router configuration to be able to modify your configuration via SNMP with community **writeAccess**.

Use **snmpset** command to set the **sysContact** MIB variable. Use your name and e-mail address as a value.

Verify the result with **snmpget** command again.

Similarly, use **snmpset** command to set the **sysName** MIB variable. Use any fancy string as a value.

Verify the result with **snmpget** command.

Examine what has changed in the console window giving you the access to the router.