SNMPv3 a practical introduction

T D Evans

Updated 2022 August 24

Table of Contents

Introduction

The purpose of this document is to provide a practical introduction to using SNMP. The emphasis of this document is SNMP Version 3. The Simple Network Management Protocol (SNMP) is described on Wikipedia as “an Internet Standard protocol for collecting and organizing information about managed devices on IP networks and for modifying that information to change device behaviour”. The term “SNMP” is often used broadly in describing entities and configurations that make use of the SNMP protocols.

This document does not describe the network protocols in detail; it does not describe packet structures or sequences of packet exchanges. The focus of this document is providing information to allow systems to be configured to make use of SNMPv3.

SNMP, and particularly Version 3, is often perceived as problematic and difficult to use. It is hoped that this document will change some of that perception and encourage the use of SNMP Version 3. Version 3 offers significant security improvements over previous versions. With the ever growing importance of cyber security, it is important that the most secure technologies are used where ever possible. Like many technologies that have been around for a long time, a number of myths have grown and it worth dispelling some of them here.

SNMP is sometimes perceived as difficult or obscure. It is certainly the case that it has evolved over many years and that different vendors have interpreted and implemented aspects in different ways. However this is generally the case with well established technologies; DBAs will be familiar with the evolution of SQL and different vendor implementations, programmers will be used to the development of languages and different levels of standards compliance and “features” found in different compilers and interpretors from different vendors. Proposed standards for Version 3 of SNMP were introduced around 1998 and formalised by the end of 2002, and should now be used where ever possible. The idiosyncrasies of different vendors must be dealt with by consulting the vendor documentation as with all technologies. SNMP is designed to work in heterogeneous environments, with a vast range of technologies and thus has its own characteristics that are not "like" other specific technologies.

SNMP does not flood the network. SNMP has been around since the late 1980s. When the technology was first introduced, networks using shared 10Mb Ethernet (or 4Mb / 16Mb TokenRing) were common. Unfortunately the new SNMP technology, at that time, was sometimes misconfigured, sometimes resulting in systems constantly re-sending alerts when problems occurred (SNMP Traps). Networks using 1980s technology could sometimes be overwhelmed with SNMP alert (Trap) messages. In the decades that have passed, the network technology in use has changed considerably and the understanding and best practice for SNMP has also changed.

Sometimes SNMP is dismissed as being insecure, again for historical reasons. The world was a different place when SNMP was introduced in the late 1980s. Around the late 1980s most networks were isolated at a time that pre-dates the World Wide Web. Network analysers were rare, expensive, proprietary items of equipment or software. The earlier versions of SNMP simply relied on what was effectively a password sent in plain text (a community string). Version 3 introduced significant security improvements at the beginning of the century. Implementations of the technology are available to tunnel SNMP over security protocols like TLS, DTLS and SSH.

SNMP is not obsolete. There is no other technology that can replace SNMP in terms of its ability to manage the vast range of equipment and software in todays heterogeneous environments.

The technology is not irrelevant. The fortunes and popularity of vendor specific solutions and vendor independent, standards driven solutions, wax and wane over the years. When people and vendors become frustrated by the slow pace of standards organisations, vendor's solutions become more popular. As people become disillusioned by vendor lock-in, they look to independent solutions. If an organisation obtains its entire IT infrastructure, (all applications, software, computers, printers, network switches etc) from a single vendor, then it would be appropriate to simply use that vendor's management solution. There are few vendors that offer the full range of IT infrastructure, as described above; in practice most organisations have heterogeneous IT infrastructures, in which there will always be a place for SNMP. Some vendors will promote their management solutions as being a superior solution to SNMP (which may well be the case for their own products), but it is worth noting that such solutions often manage other vendor's products via SNMP.

SNMP has not been killed by the cloud. At the time of writing, the popularity of cloud solutions has grown vastly and this is certainly a technology that is here to stay and will continue to grow. As organisations move to the cloud there will remain a need to connect the devices used by staff to the cloud which will remain some kind of infrastructure of routers and / or switches. The routers and / or switches connecting staff devices to the cloud is likely to be best managed via SNMP. The migration to the cloud can be seen as another form of out-sourcing (the popularity of out-sourcing also waxes and wanes). Use of the cloud is attractive because of the potential cost savings and flexibility. Security concerns and vendor lock-in, can be inhibiting factors. Some organisations adopt hybrid solutions to meet their needs. The need for SNMP to manage infrastructure will remain for the foreseeable future.

The fact that Microsoft has deprecated SNMP agent support in its product and plans to completely remove SNMP agent support from all its products does not make SNMP irrelevant. Microsoft has cited security concerns in making this move. As indicated above, there will always remain a range of items, such as switches and routers that are best managed via SNMP. It is worth noting that the Microsoft management product, System Center Operations Manager 2019 can be configured to manage network devices via SNMP. This is described in the Microsoft on-line document Monitoring networks by using Operations Manager, available at the time of writing, at the following URL:

https://docs.microsoft.com/en-us/system-center/scom/manage-monitor-networkdevice-overview?view=sc-om-2019 [https://docs.microsoft.com/en-us/system-center/scom/manage-monitor-networkdevice-overview?view=sc-om-2019]

The third paragraph begins:

“Operations Manager can discover and monitor network devices that use the Simple Network Management Protocol (SNMP v1, v2c, and v3.)”

It is worth noting that the “System Center Operations Manager 2019” product can be configured to work with SNMP v1 which is regarded as insecure, and it is recommended that the product is configured to work with version 3.

Information on configuring Windows to support SNMP through PowerShell is given in Appendix A, *Windows PowerShell SNMP enabling*.

Who should read this document and why another document

This document is aimed at systems administrators who need to manage systems in heterogeneous environments. The environment will probably include printers and network equipment.

In many organisations a network management system is used (which may be proprietary from a vendor). The system can manage various items via SNMP (often equipment such as printers and switches) once the management system and items to be managed have been configured to talk to each other via SNMP. Once communications have been established, management systems can provide friendly GUI interfaces for further configuration, hiding the SNMP details. But first communications need to be configured. This document aims to provide information to facilitate such configuration.

Many systems administrators do not use SNMP interactively from a command line. They focus on configuring systems to communicate via SNMP and use the interfaces of various management systems. This document does not focus on SNMP command line tools.

This document is Operating System agnostic. The document does not focus on any particular vendor or technology, however examples are included from various vendors as described in the section called “About Examples”.

The readier is assumed to have knowledge of IP, and some knowledge of SNMP (perhaps v1 and 2c).

Although many excellent books about SNMP are now out of date, fortunately there are a number of good books and other resources available for SNMP. Some books focus on the network protocols, and are concerned with packet structures, for example. Other books are aimed at programmers. Many books focus on a single operating system (often Linux) or a particular technology or implementation (e.g. net-snmp). Some books focus on a selection of SNMP solutions, perhaps devoting a chapter to each one. This document aims to provide a practical introduction. While examples are used to illustrate various configurations, the aim is not to provide complete descriptions of any particular solution.

Terminology Targets and Managers

The term SNMP is used for all versions of SNMP; the term SNMPv3 refers specifically to versions 3.

In SNMP, typically requests for information or requests to make changes are send from one system, referred to here as a manager, to a second system, referred to here as a target. There are also facilities for the system, referred to as a target, to send unprompted messages (traditionally known as traps) back to the systems referred to here as managers.

About Examples

Examples are provided from a few popular commercial and Open Source products. This document does not promote or endorse any of the products included in the examples. The examples are for illustration only.

Cisco Switches

Cisco managed switches support SNMP and are often configured through a command line interface. A description of the Cisco IOS and command line interface is outside the scope of this document. But for readers familiar with the interface, examples of configuring these switches for SNMPv3 via the command line are included.

Linux

A system running Linux can be configured to be managed via SNMP. This is usually achieved via a package that includes an SNMP daemon, snmpd. Generally the software used will be the net-snmp collection. See:

http://www.net-snmp.org

Sometimes it is possible just to install the daemon, for example:

```
# sudo apt-get install snmpd
```

MIB Browser

MIB Browser is a product from iReasoning Inc which the company describes as:

“iReasoning MIB browser is an indispensable tool for engineers to manage SNMP enabled network devices and applications. It allows users to load MIBs, issue SNMP requests, and receive traps.”

The product runs on any OS which has JVM support, including Linux, macOS and windows. There are three versions of the product, a free Personal Edition, Professional and Enterprise editions. Unfortunately the Personal Edition does not support SNMPv3, only older versions of the protocol.

Monitoring software Icinga, Naemon, Nagios, Shinken, Sensu, etc

There are several monitoring packages, generally derived from Nagios, that can monitor a range of systems. All of these packages can use a common core set of plug in modules to perform specific tasks (there are many additional plug in modules). A bundle of more than fifty standard plugins is available from “The Monitoring Plugins Project”. See: https://www.monitoring-plugins.org

The module used for checking via SNMP is check_snmp. See:

https://www.monitoring-plugins.org/doc/man/check_snmp.html

Some examples of using "check_snmp" are included in this document and are relevant to monitoring systems such as Icinga, Naemon, Nagios, Shinken, Sensu, etc.

Multi Router Traffic Grapher (MRTG)

Wikipedia describes MRTG -

The Multi Router Traffic Grapher (MRTG) is free software for monitoring and measuring the traffic load on network links. It allows the user to see traffic load on a network over time in graphical form.

It was originally developed by Tobias Oetiker and Dave Rand to monitor router traffic, but has developed into a tool that can create graphs and statistics for almost anything.

MRTG is written in Perl and can run on many Operating Systems, including Windows, Linux, versions of Unix, and Mac OS.

See: https://en.wikipedia.org/wiki/Multi_Router_Traffic_Grapher

The home page for MRTG is https://oss.oetiker.ch/mrtg

Although MRTG can gather statistics via a variety of methods, such as scripts, its focus is on gathering information via SNMP.

The product has not changed much for a number of years, possibly because it already does what it needs to do. Part of the design of MRTG is to keep things relatively simple; it can be seen broadly as a wrap around for RRDTool (https://oss.oetiker.ch/rrdtool). MRTG is a popular tool for creating performance graphs relatively easily.

TKMIB

TKMIB is a Perl GUI MIB browser included within the Net-SNMP package. The tool presents a tree view of a target and allows SNMP requests to be made to the target.

For information on Net-SNMP see:

www.net-snmp.org

Security - human communications

For security reasons it is important that permission is obtained prior to managing, or interactively accessing a system via SNMPv3. Under some circumstances written approval may be required.

People responsible for systems to be managed by SNMPv3 and those responsible for management systems, and people using interactive tools should talk to each other. As well as clearly establishing permission and the “rules of engagement”, the opportunity can be taken to agree the protocols and security parameters that are to be used.

SNMP has no auto-discovery capability and there are generally no mechanisms for systems to "negotiate" security configurations. SNMP is designed to operate in heterogeneous environments for management purposes. It is appropriate and essential for the people responsible for systems that will communicate via SNMP to agree and configure the security parameters required.

SNMP Versions

There are three versions of SNMP in use, the original Version 1, Version 2c and Version 3. Use of version 3 is recommended because of the significant security and other improvements that were introduced.

For two systems to communicate via SNMP, they must use the same version. Some software may try to establish communications with another system by trying different versions, but once communications are established both systems will be using the same version. In general it is necessary to specify the version to be used.

Specifying version 3

Both systems communicating via SNMP must use the same version and version 3 is recommend. The way in which the version is specified will depend upon the system in use. As explained below, systems using SNMPv3 use a “user” name (specific to SNMPv3) to communicate. Because the use of a user is specific to version 3, often the process of setting up an SNMPv3 user, specifies the use of version 3.

Example check_snmp

When using the "check_snmp" plug-in, used by several monitoring systems, several parameters are specified, including the SNMP version with the -P parameter, e.g.

```
check_snmp -P 3 <additional parameters>
```

Example MRTG

MRTG has SNMPv3 disabled by default. To enable SNMPv3, the configuration file should include the directive:

```
EnableSnmpV3: yes
```

Note that the the perl Net::SNMP library also needs to be installed to support SNMPv3, which typically does not get installed as a dependency, because MRTG will install and run with earlier SNMP versions.

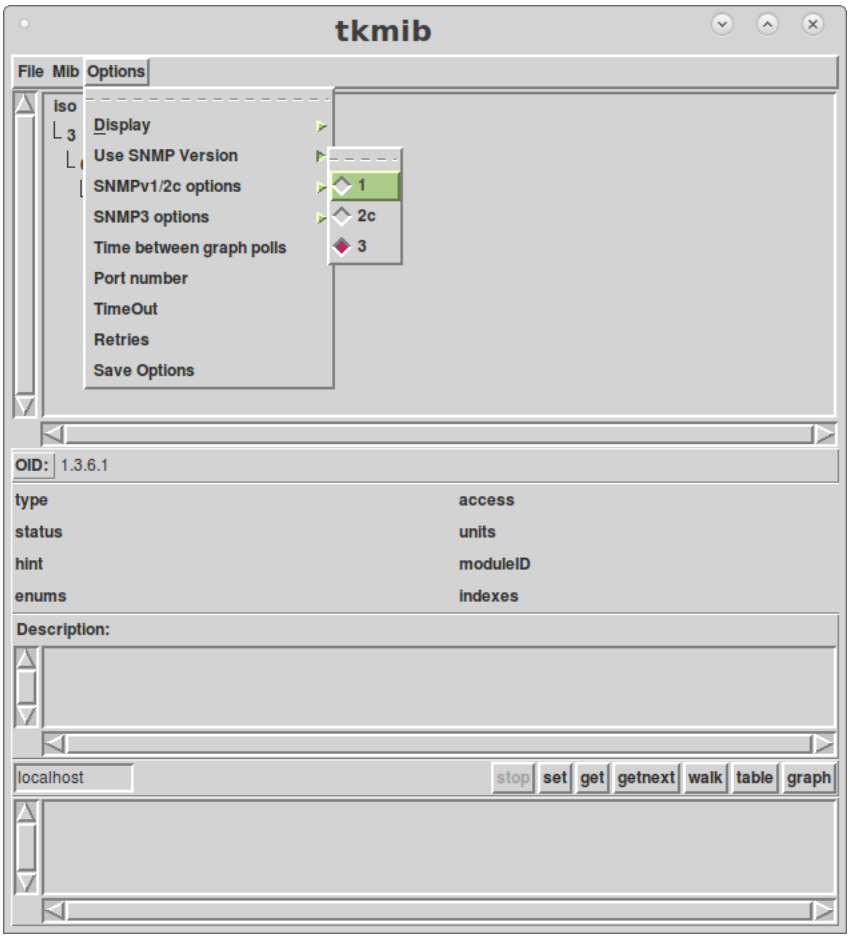
When defining targets for MRTG to query via SNMP, it is also necessary to indicate that version 3 should be used by appending 3 as an additional parameter. E.g.

```
Target[myswitch]: 2:switcha:::::3
```

Example TKMIB

From the "Options" drop down menu, the "Use SNMP Version" option is selected and then the third item 3 is chosen to specify SNMPv3.

Figure 1. tkmib SNMP version choice



User-based Security Model (USM)

Overview

SNMPv3 is designed to work in a heterogeneous environment. SNMPv3 security was not designed to use any third party security facility or directory services (and is thus not dependent on any other security facility). SNMPv3 does not use Kerberos or directory services such as Microsoft Active Directory or LDAP; it is designed to function in environments where these, or other facilities, may not be available. SNMPv3 uses its own security arrangements that are independent of the systems it is deployed on.

In many organisations, a single Network Management System will be used to manage multiple targets. However SNMPv3 is designed to be scalable and a number of entities may communicate with multiple targets. For example monitoring and alerting may be implemented using one particular software package, while performance monitoring may be implemented with another, both packages possibly running on the same platform. SNMPv3 is designed to be capable of supporting communications in a many to many arrangement.

To support potentially many to many communications, SNMPv3 requires communicating systems to use a set of matching security arrangements. Different pairs of systems can use different sets of arrangements within a common environment. SNMPv3 utilises user ids to identify communicating entities for security purposes.

User

It is important to understand that an SNMPv3 user has nothing to do with users that may exist on a target or within a directory system. SNMPv3 users are completely separate from the users that may exist in an etc/passwd file on a Unix system, for example, or within a directory service such as Active Directory or LDAP.

An SNMPv3 user is typically stored in an SNMPv3 configuration file on a system that is using SNMPv3. Often within an organisation a single SNMPv3 user id is sufficient for one or two management systems and all targets. However for scalability, multiple user ids can be defined on both targets and management systems. It is worth noting that it is not unusual for some items of equipment to only support a single user.

Templates

The standards talk about using "templates" to create users. For scalability, the recommendation is that a "template" user is created from which further users can be set up as required. The template user is a standard SNMPv3 user and can be used in the same way as any other. In practice, often a single user is set up (which could be regarded as a template), that user is used, and no further users are created. As an example Cisco switches often support SNMPv3, but the documentation often makes no reference to templates.

Group on target

To provide additional flexibility and scalability, systems to be accessed via SNMPv3 implement groups containing one or more SNMPv3 users. It should be noted that groups are not part of the SNMPv3 communications protocol. The implementation of groups allows greater control of security access for different sets of SNMPv3 users as explained later in the section called “Groups”. Groups in SNMPv3 implementations are also typically stored within SNMPv3 configuration files (often the same that contain the user ids) and have nothing to do with any other kind of groups on systems or within directory services. Groups are used with view based access control as explained later in the section called “Views”.

Often a single SNMPv3 group will be used on a target system, often containing a single user. It is typical for systems to be accessed via SNMPv3, to require a group to be specified when users are created; thus often at least one group should be defined before SNMPv3 users are created.

On items of equipment that only use a single SNMPv3 user there will probably be no groups defined.

Authentication and Encryption

Overview

Security within SNMPv3 is specific to SNMPv3 and independent of any other security arrangements that may exist on a system where SNMPv3 has been deployed. It is designed to operate in a heterogeneous environment where different security technologies may exist. Because SNMPv3 is designed to function with almost any hardware or software, the security is different to almost any other security system.

Unlike other security systems, SNMPv3 separates the tasks of authentication and encrypting communications to ensure privacy. There are a number of reasons for this separation.

The RFC talks about the availability and distribution of encryption technology.

For most other security systems, for example Kerberos, communicating systems use the same security technology and thus once authentication has been established, two parties will know how to communicate with each other. The situation is different for SNMPv3.

Levels

SNMPv3 can support various combinations of authentication and encryption (privacy) to accommodate legacy systems. Unless there are exceptional circumstances, both authentication and encryption (privacy) should be used and this is specified by setting the security parameter, referred to as the security level to be Authentication with Privacy, often specified as a value such as "AuthPriv". The use of other modes is not recommended, but they are discussed in the section called “Legacy modes”.

For two SNMPv3 entities to communicate, they must both be configured with a common set of security settings; this involves two parts. Common authentication parameters must be configured and common encryption (privacy) parameters must be configured.

Authentication

For authentication, three elements are needed: a username, a password and an authentication protocol. The original RFC mandated that at least two protocols be supported; one using MD5 and one using SHA (a.k.a. SHA-1). The two protocols are HMAC-MD5-96 and HMAC-SHA-96 respectively (see below), but will be referred to here as "MD5" and "SHA" for convenience. Any number of different protocols could be used in the future. At the time of writing the two original authentication protocols are in use and there are some systems that support the use of SHA-2, also known as SHA-256.

It is essential that two communicating SNMPv3 entities use the same authentication protocol.

The use of MD5 is not recommended because MD5 is no longer considered secure, however it is preferable to transmitting clear text as was the case with previous versions of SNMP. Sadly SHA version 1 is no longer viewed as secure as it was when SNMPv3 was first specified, however SHA version 1 is preferable to MD5; and there are now some systems that support the use of SHA-2 a.k.a. SHA-256.

Both the protocols above, using MD5 or SHA involve additional mechanisms.

The MD5 hash function is used in HMAC mode and the output is truncated to 96 bits. The HMAC (hash-based message authentication code) is described in RFC 2104. This is the HMAC-MD5-96 authentication protocol.

The SHA hash function is used in HMAC mode and the output is truncated to 96 bits. This is the HMAC-SHA-96 authentication protocol.

Technical details of the authentication protocols can be found in RFC 3414.

The username should be assumed to be case sensitive and should probably begin with a letter. The constraints will be implementation dependent, but it will be good practice to limit usernames to a short string of alpha-numeric characters.

Section 11.2 Defining Users of RFC 3414 specifies a minimum length of eight characters for passwords which are case sensitive.

The three authentication parameters are listed below with examples.

Authentication parameter	**Example**	**Comments**
username	mercury	Not related to system or directory services users.
password	Nuef0dapfps#	Never use examples from online documents as passwords for production systems
protocol	SHA	One of: • MD5 • SHA a.k.a SHA-1 • SHA-2 (a.k.a. SHA 256)

Encryption

For encryption (privacy) two parameters are needed, a password and an encryption protocol. The original specification described the use of DES with AES added later. Additional protocols could become available in the future.

It is worth noting that there are variations of the above protocols and alternative terms are sometimes used for the same protocol. The original DES protocol is sometimes referred to as DES56. Some systems support the use of Triple DES otherwise known as 3Des.

There are a variety of versions of AES implemented by some systems. The original version of AES is sometimes referred to as AES128; some systems implement encryption based on AES192 or AES256.

It is essential that two communicating SNMPv3 entities use the same privacy protocol. It is recommended that the most secure common protocol is used.

Because DES is no longer considered secure, the use of the strongest version of AES is recommended if it is supported by both communicating entities.

The two encryption parameters are listed below with examples.

Authentication parameter	**Example**	**Comments**
password	Pf1bs#n#buuac	Passwords found in books should not be used under any circumstances
protocol	AES	One of: • DES • AES a.ka. AES128 • AES192 • AES256

Legacy modes

As explained above authentication and privacy (encryption) are separate in SNMPv3. It is recommended that both authentication and privacy are used whenever possible, but there are other modes that may need to be used, described below.

Authnopriv

Because authentication and privacy (encryption) are separate in SNMPv3, it is possible for two systems to communicate via SNMPv3 using authentication, but without using privacy (encryption). The reason sometimes given for the separation of authentication and privacy is because of export restrictions on encryption technology at the time the standards were being defined. Such restrictions have long since ceased to be relevant and there is no reason not to use the privacy modes available today.

Unfortunately some vendors took advantage of the separation of authentication and privacy and sold encryption technology as optional extra features (sometimes at significant cost). Because encryption technology was often not needed in the past for business reasons, the encryption technology was often not purchased, a consequence of which was that it was not available for SNMPv3. Regrettably there are still some vendors effectively selling encryption technology (that is freely available).

Some legacy systems may only support SNMPv3 with authentication, but not privacy. Legacy systems that do no support privacy will need to be configured with the Authnopriv setting. SNMPv3 users defined on these systems will not have a privacy technology specified (DES or AES) and will not have a privacy password. Systems communicating with legacy systems in authentication mode only will also specify Authnopriv and not supply a privacy technology or password.

noauthnopriv

It is possible to configure SNMPv3 to operate without authentication or privacy. Such configurations should only be used in a lab environment for testing purposes or developing SNMPv3 software. In this mode noauthnopriv is specified and no parameters for authentication or privacy technology are given and no passwords are provided.

unused mode

For the sake of completeness, an unused mode is described here. Observant readers familiar with truth tables will note that there is a theoretical mode of operation that would use privacy (encryption), but with no authentication. Such a mode is not defined in any standards as far as the author is aware, and the author is not aware of any implementation of this strange configuration.

Examples

In this section some examples are given illustrating the User-based Security Model. First, examples of creating groups are given as this is sometimes necessary before users are created. This is followed by examples for various targets, focusing on user creation.

Groups on targets

On target systems, groups can be created to hold users.

Groups on Cisco devices

On many Cisco devices, a group is specified with:

```
snmp-server group <Groupname> <Version> <Level>
```

Where <Groupname> is the name of the new group

<Version> is one of v1, v2c, or v3

<Level> is one of "noauth", "auth" or "priv" with priv indicating both authentication and privacy (encryption) and is the recommend level.

E.g.

```
snmp-server icinga1 v3 priv
```

Groups on Linux systems

On Linux, the listening process is usually snmpd. This is configured with a configuration file such as snmpd.conf typically in /etc/snmp Groups are defined by entering one or more group entries in to the configuration file (one for each user in a group) with:

```
group <group name> usm <username>
```

Where <group name> is the name of the group, and <username> is the name of the user, e.g:

```
#         sec.model sec.name
group netmanager usm    naemon1
group netmanager usm    rrdgrapher1
```

Note that the third entry is "usm" (for User-based Security Model) and not v3; (alternatives shown in examples in the man pages are v1 and v2c for older SNMP versions).

Targets

Some examples are given below for configuring systems to be managed (or interactively accessed) via SNMPv3. For convenience these systems are referred to here as targets.

Linux

On Linux, the "net-snmp" packages are typically used. Information about SNMP users is typically held in files such as snmpd.conf. Often there will be a general configuration file and a separate file containing the credentials in encrypted form.

Distributions often include the "snmpusm" utility to create SNMPv3 users; the utility can also be used to clone, delete and manage SNMPv3 users.

To create a user:

```
snmpusm -v3 -l priv -u <TEMPLATE USER> -A <AUTH_PASSWORD> -a <AUTH_PROTOCOL>
 -X <PRIV_PASSWORD> -x <PRIV_PROTOCOL> localhost create <SNMP-user>
```

Where the variables are defined as follows:

<TEMPLATE USER>	an existing template user. See the section called “Templates”.
<AUTH_PASSWORD>	the authentication password
<AUTH_PROTOCOL>	the authentication protocol
<PRIV_PASSWORD>	the privacy (encryption) password
<PRIV_PROTOCOL>	the privacy protocol
<SNMP-user>	the SNMP user

Another utility (often found on RPM based distributions) to create SNMPv3 users is called "net-create-v3-user":

```
net-snmp-create-v3-user -ro -A <AUTH_PASSWORD> -a <AUTH_PROTOCOL>
 -X <PRIV_PASSWORD> -x <PRIV_PROTOCOL> <SNMP-user>
```

Where the variables are defined as follows:

<AUTH_PASSWORD>	the authentication password
<AUTH_PROTOCOL>	the authentication protocol
<PRIV_PASSWORD>	the privacy (encryption) password
<PRIV_PROTOCOL>	the privacy protocol
<SNMP-user>	the SNMP user

E.g:

```
# net-snmp-create-v3-user -ro -A Nuef0dapfps# -a sha -X Pf1bs#n#buuac
 -x AES mrtguser1
```

Network Switches and Routers

Many network devices have Web based or other GUI interfaces which makes configuring SNMPv3 users easier. Authentication and privacy protocols are selected via tick boxes or pull-down menus and user names and passwords are entered in the appropriate fields. Some devices are often configured via command line interfaces, such as Cisco devices.

Cisco devices

On Cisco switches, SNMPv3 users can be created with the snmp-server user command:

```
snmp-server user <SNMP-user> <GROUP> v3 auth <AUTH-PROTOCOL>
  <AUTH_PASSWORD> priv <PRIV-PROTOCOL> <PRIV_PASSWORD>
```

<SNMP-user>	the SNMPv3 user
<GROUP>	an SNMPv3 group
<AUTH_PASSWORD>	the authentication password
<AUTH_PROTOCOL>	the authentication protocol

<PRIV_PASSWORD> the privacy (encryption) password
<PRIV_PROTOCOL> the privacy protocol

E.g.

```
snmp-server user mrtguser1 icinga1 v3 auth sha Nuef0dapfps#
 priv aes Pf1bs#n#buuac
```

Interactive

There are numerous interactive tools available that can be used to interrogate and manage systems via SNMPv3. Three examples are given below, a command line example and two MIB Browsers.

Linux command line

There is a collection of command line tools for Linux, that includes "snmpget", "snmpwalk" etc. Information on these commands can be found with:

```
# man snmpcmd
```

For use with SNMPv3, a set of security parameters must be supplied that matches those expected by the target.

```
-v3 -l <LEVEL> -a <AUTH-PROTO> -A <AUTH_PASSWORD> -x <PRIV-PROTO>
 -X <PRIV_PASSWORD> -u <SNMP-user>
```

Where the variables are defined as follows:

<LEVEL> The security level and should be Authentication with Privacy (encryption), indicated with authPriv. (legacy levels might be used in special circumstances.
<AUTH_PASSWORD> the authentication password
<AUTH_PROTOCOL> the authentication protocol
<PRIV_PASSWORD> the privacy (encryption) password
<PRIV_PROTOCOL> the privacy protocol
<SNMP-user> the SNMPv3 user

E.g.

```
<command to use> -v3 -l authPriv -A Nuef0dapfps# -a sha -X Pf1bs#n#buuac
 -x AES -u mrtguser1 <additional parameters>
```

MIB Browser

When using MIB Browser to access a target via SNMPv3 an SNMPv3 user needs to be specified with the appropriate parameters. From the MIB Browser interface, clicking on "Advanced..." brings up the

"Advanced Properties of SNMP Agent" dialogue window. In the "Advanced Properties of SNMP Agent" dialogue window, selecting 3 from the "SNMP Version" drop down list, will display the necessary fields for the version 3 parameters.

Figure 2. MIB Browser

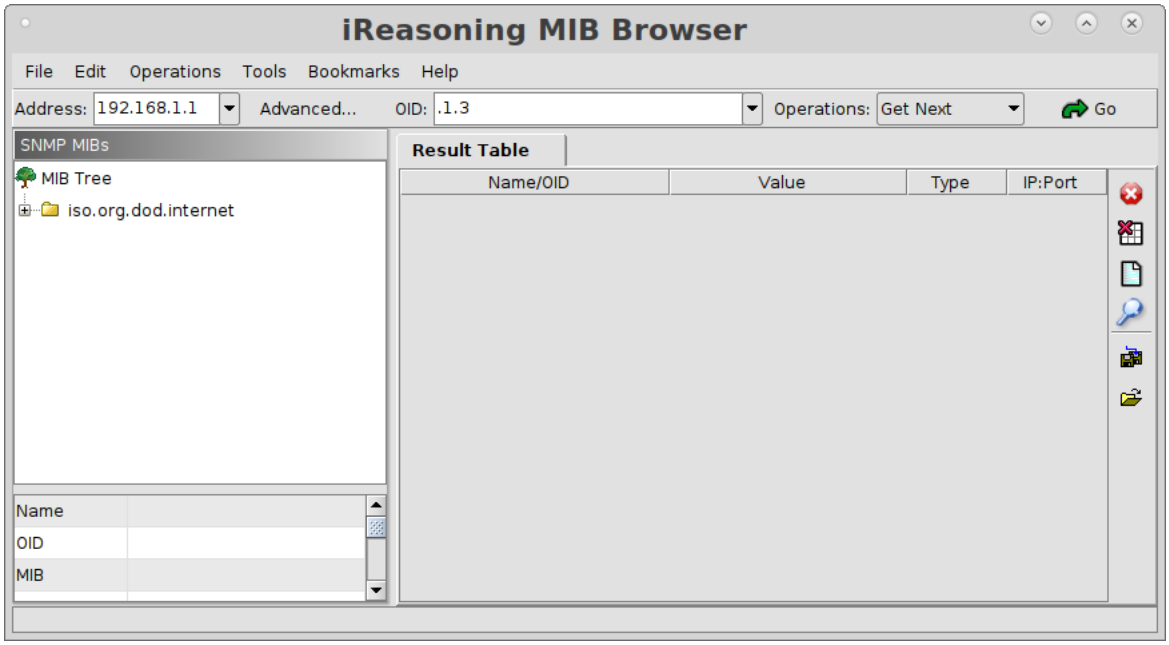
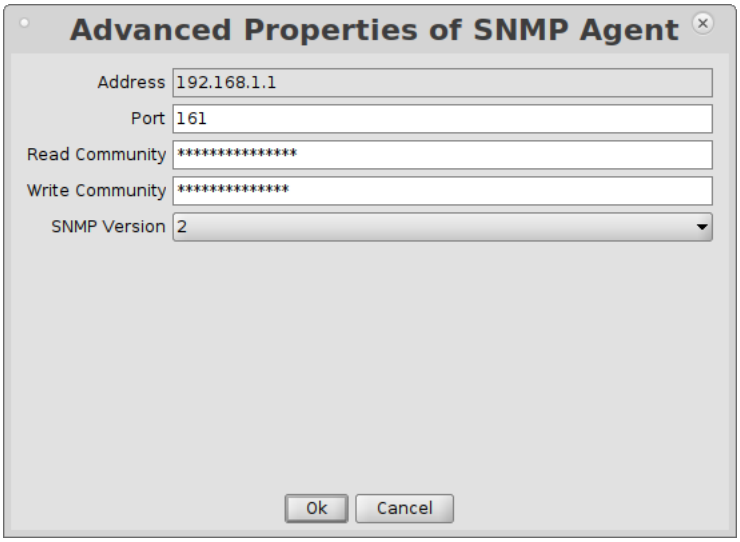


Figure 3. MIB Browser Advanced Properties



TKMIB

When using TKMIB to access a target via SNMPv3, an SNMPv3 user needs to be specified with the appropriate parameters. From the "Options" drop-down menu, select the "SNMP3 Options" item, which presents a drop-down menu for each of the required parameters (see Figure 4, “TKMIB SNMPv3 Options”). Specify the SNMPv3 username from the "Security Name" option (see Figure 5, “TKMIB Security Name”).

Figure 4. TKMIB SNMPv3 Options

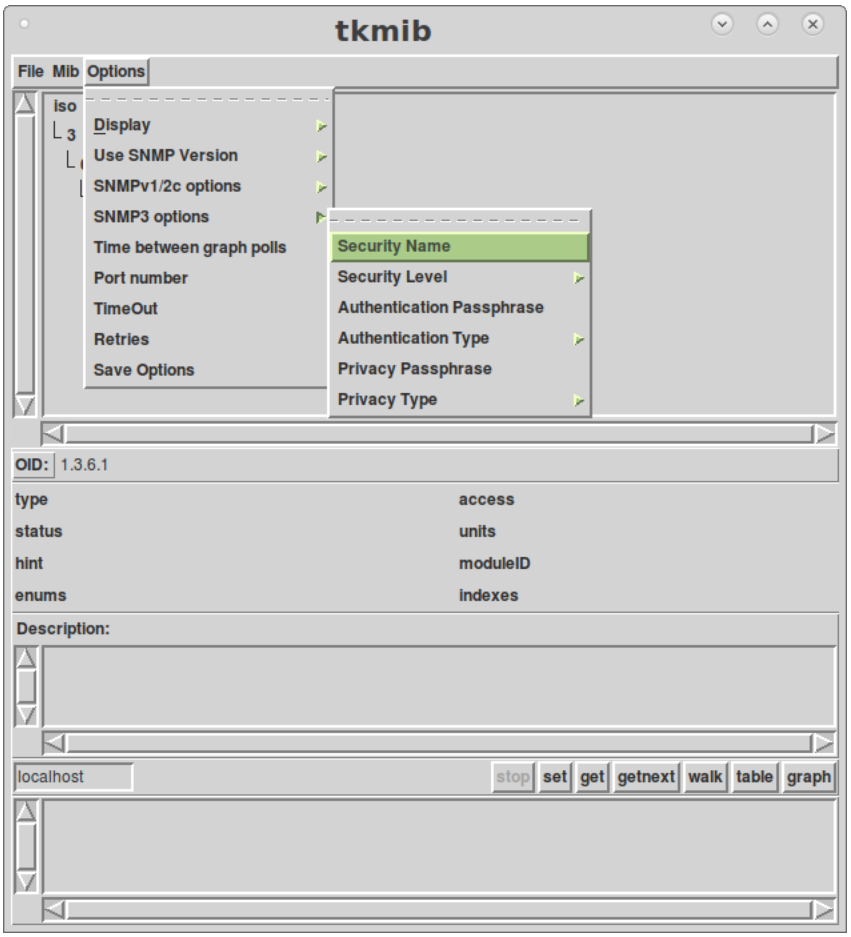
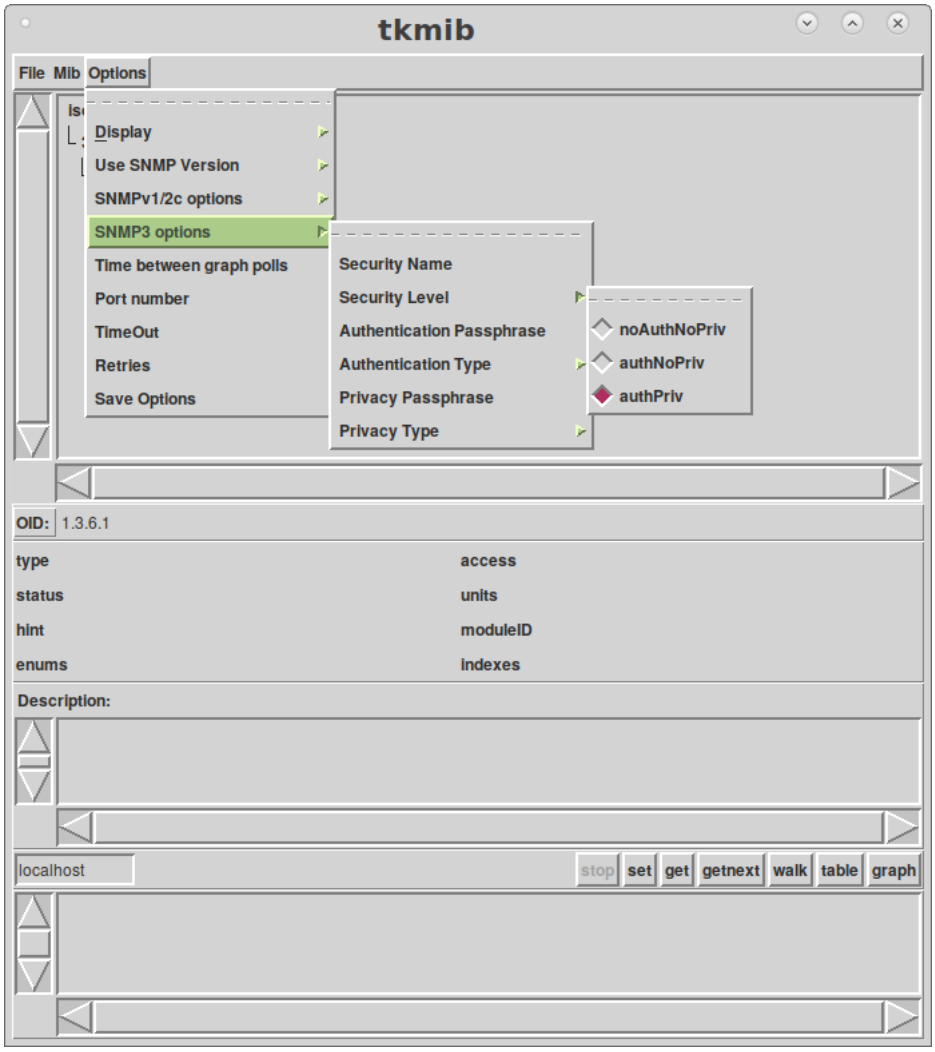


Figure 5. TKMIB Security Name

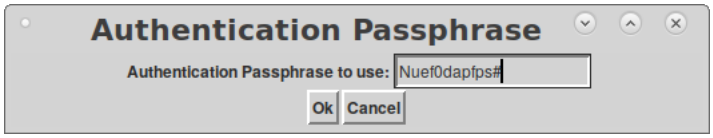
Set the SNMPv3 level from the "Security Level" option (see Figure 6, “TKMIB Security Level”). For authentication and privacy, the "authPriv" level should be chosen whenever possible. Some legacy systems may not support privacy (encryption), in which case "authNoPriv" will have to be used.

Figure 6. TKMIB Security Level



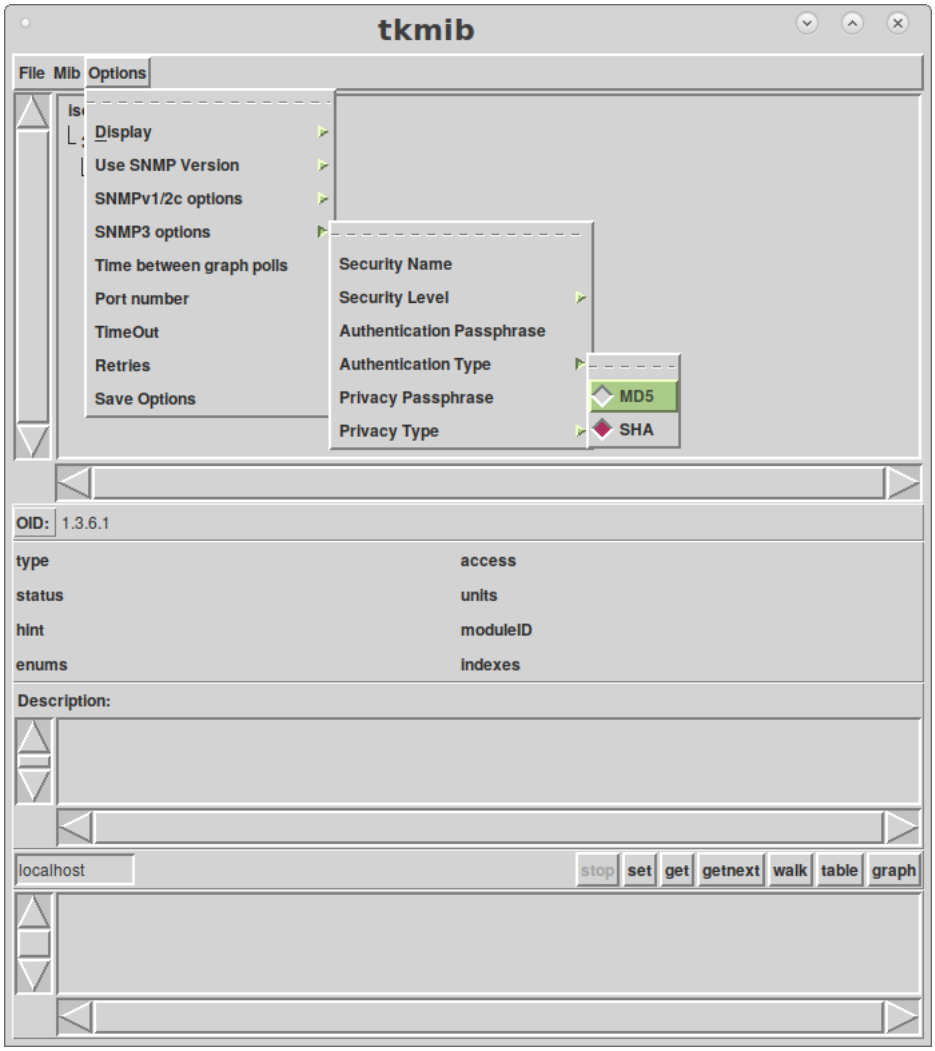
The "Authentication Passphrase" option should be chosen; enter the passphrase in the resulting dialogue (see Figure 7, “TKMIB Authentication Passphrase”).

Figure 7. TKMIB Authentication Passphrase



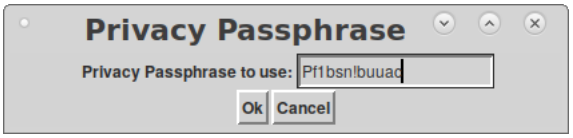
Choose the authentication protocol from the "Authentication Type" (see Figure 8, “TKMIB Authentication Protocol”) which will be one of "MD5" or "SHA" (the latter being recommend).

Figure 8. TKMIB Authentication Protocol



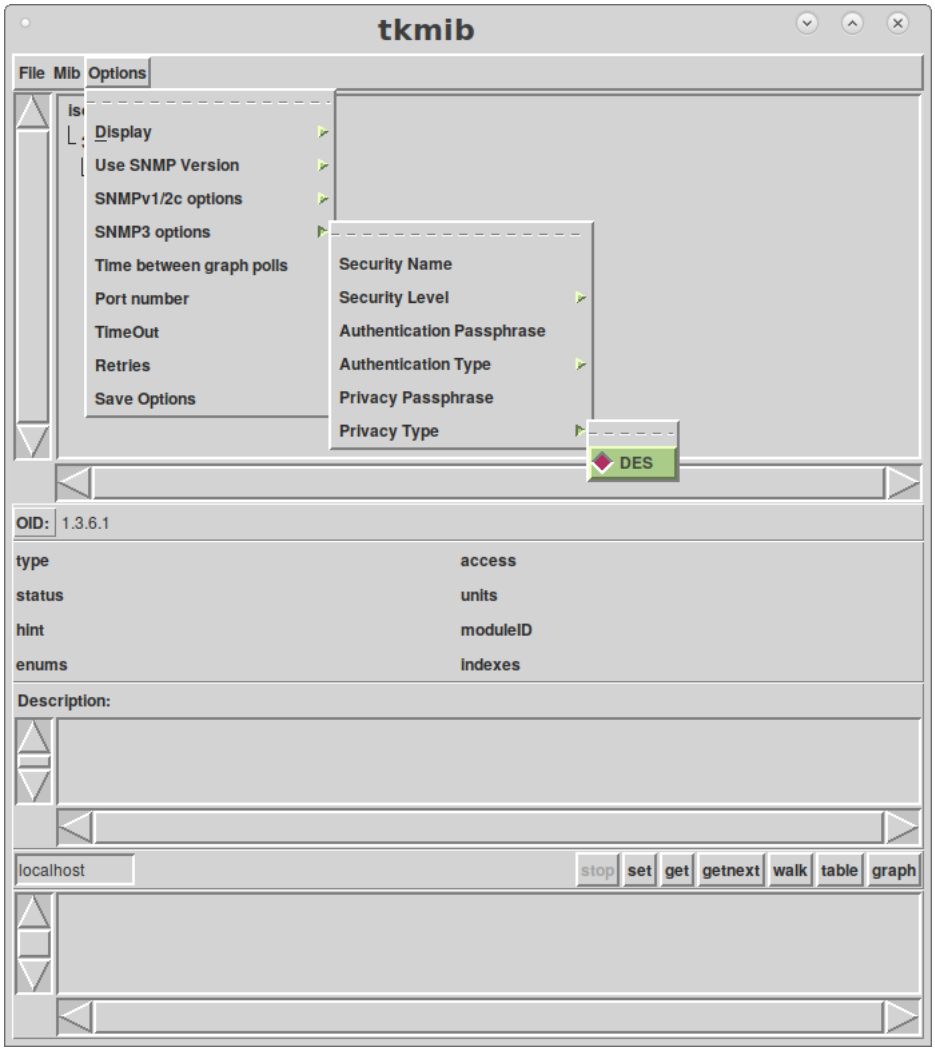
Selecting "Privacy Passphrase" presents a dialogue for the privacy passphrase to be entered (see Figure 9, “TKMIB Privacy Passphrase”).

Figure 9. TKMIB Privacy Passphrase



From "Privacy Type", select the privacy protocol (see Figure 10, “TKMIB Privacy Protocol”). The version of tkmib that ships with net-snmp version 5.9.1 only offers "DES".

Figure 10. TKMIB Privacy Protocol



Managers

Some examples are given here of various systems that obtain information from "targets" for monitoring or measuring purposes.

Example check_snmp

Several management systems derived from Nagios and Nagios itself use "check_snmp"

When using the "check_snmp" plug-in, used by several monitoring systems such as Nagios or one of its forks, several parameters are specified, including the following for SNMP version 3 as show below:

SNMP version: -P or --protocol=3

Security level: -L, or --seclevel=<Level>

where <Level> is the SNMPv3 security level and is one of noAuthNoPriv, authNoPriv, authPriv

Authentication protocol: -a or authproto=<Authentication protocol>

where <Authentication protocol> is one of MD5 or SHA

Authentication password: -A or authpassword=<Authentication password>

where <Authentication password> is the SNMPv3 password to be used

Privacy (encryption) protocol: -x or privproto=<Privacy protocol>

where <Privacy protocol> is one of DES or AES (AES is recommend)

Privacy (encryption) password: -X, --privpasswd=<Privacy password>

where <Privacy password> is the SNMPv3 privacy (encryption) password to be used

User name: -U, --secname=<User name>

where <User name> is the name of the SNMPv3 user to be used.

E.g.

```
check_snmp -P 3 -L authPriv -a SHA -A passw0rd123  x AES -X secret456
 -U mgruser1 <additional parameters>
```

MRTG

When defining a target for MRTG to access, to collect data for graphing, the SNMPv3 parameters need to be specified.

It should be noted that the syntax of the MRTG target definition still requires a legacy community value to be included as used with older versions of SNMP. The syntax requires "<legacy community name>@<target>", but when using SNMPv3, the <legacy community name> can be set to an arbitrary value, such as "null".

After the target entry, a number of colon separated values can be provided the last of which is the SNMP version which should be set to 3 (unused values can be omitted).

```
Target[mytarget_uplink]: 123:null@mytarget.local:::::3
```

The SNMPv3 parameters are specified with the "SnmpOptions" entry:

E.g.

```
SnmpOptions[mytarget_uplink]: username=> mrtgviewer, authprotocol=> sha,
 privprotocol=> aes256, authpassword=> Badword, privpassword=> weakword
```

The colon separated list of values referred to above is specified with:

:<port>:<timeout>:<retries>:<backoff>:<version>

Where:

<port>	the UDP port (default: 161)
<timeoutL>	the initial timeout, in seconds, for SNMP queries (default: 2.0)
<retries>	number of times a timed-out request will be retried (default: 5)
<backoff>	factor by which the timeout is multiplied on every retry (default: 1.0)
<version>	SNMP version (recommended to be 3)

Summary of steps to establish USM based communications

The first and most important step in arranging for a "managing system" to connect to a "target" is the human communications. The person responsible for the "managing system" should talk to whoever is responsible for the "target" to establish clear permissions and agree the "rules of engagement". The opportunity can be taken to agree upon the common parameters.

Both systems should be configured to use SNMPv3 and the relevant vendor documentation will need to be consulted for each system to determine the mechanics of the necessary configuration.

The following parameters will need to be agreed and configured on each system according to the vendor documentation:

Level	Both systems must use the same "level" and authentication with privacy (often indicated with authPriv) is recommended. If either system does not support privacy (encryption), then it will be necessary to use authentication without privacy (often indicated with authNoPriv).
User	A common SNMPv3 user to be used by both systems should be set up. A management system may have one user that is used for all "targets". In some situations a "target" may have a SNMPv3 user to be used by all "management" systems. Both the management and target systems must use the same user.
Authentication method	Both the management and target systems must use the same authentication method, which should be the latest version of SHA supported by both systems. If either does not support SHA then MD5 will have to be used.
Authentication passphrase	A common authentication passphrase must be used by both systems.
Privacy protocol	Both the "management" and "target" systems must use the same privacy protocol, which should be the latest version of AES supported by both systems. If either does not support AES then DES will have to be used.
Privacy passphrase	A common privacy passphrase must be used by both systems.

In addition, on the "target" system a Group may need to be established, if one does not already exist, to contain the user.

Object ID - OID

SNMP can be used to manage an almost limitless number of objects, including software, equipment, and measured values and much more. For example SNMP could manage the size of a database table, the state of a network interface, report a measured temperature value or set a pressure threshold. Each object to be managed by SNMP must be uniquely identified and for this a hierarchical naming space is used. The name space that has been adopted for use by SNMP is the Object Identification (OID) mechanism.

The Object Identification (OID) mechanism is not unique to SNMP. The Object Identification (OID) naming mechanism was jointly developed by ISP/IEC and ITU-T for naming any kind of object, entity or concept with a globally unique name. The scheme is intended to provide permanent names and it not intended for transient entities. The Object Identification (OID) namespace is hierarchical. The upper layers of the tree are used to identify several organisations, each of which is responsible for allocating objects below that part of the tree, including other organisations to which control can be delegated. Each part of the tree, including the root, is managed by a Registration Authority.

The OID namespace is conceptually no more difficult to understand than any other hierarchical naming space, such as filesystems; what is different about the OID space is its vast scale both in terms of depth and breadth. This may sometimes make navigating the namespace seem tedious and laborious.

An appreciation of the OID namespace is required to understand SNMPv3 Views (see the section called “Views”).

OID names and tree

Each object is given interchangeable names; one or more character strings (human readable) and an integer (e.g. "itu-t" / "ccitt" and 0). Both kinds of identifiers are unique within the layer that they occur under a particular part of the tree. It is probably better not to make comparisons with DNS names and IP addresses; in the DNS namespace there can be a many to many relationship between names and addresses. Each OID string and integer are best thought of as aliases for each other. In theory the strings and integers (names and numbers) could be arbitrarily mixed to identify an object, in practice either the names or the numbers are used depending upon the context.

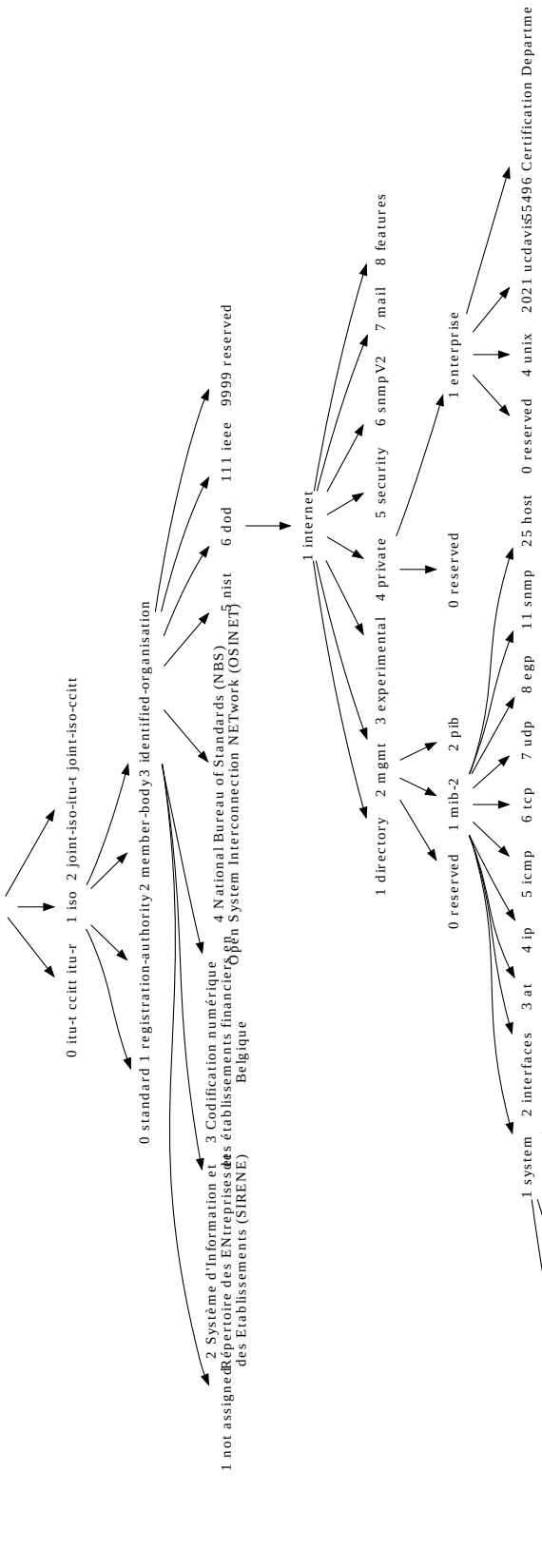
As with other hierarchical namespaces, each node in the tree is identified with a concatenation of names and separators from the root to the node / object to be identified. Like filesystems, the naming proceeds from left to right moving down the tree (opposite to the DNS for example). The separator used is the full stop / period character ("."). The root of the tree is effectively null, indicated by the first "." character.

Top layer, directly under the root is shown in the table below:

OID integer	**OID names**	**Description**
0	itu-t ccitt itu-r	International Telecommunication Union - Telecommunication standardization sector (ITU-T)
1	iso	International Organization for Standardization (ISO)
2	joint-iso-itu-t joint-iso-ccitt	Areas of joint work between ISO/IEC and ITU-T

A small portion of the top of the OID tree is shown below.

Figure 11. Partial OID tree



OID Leaf nodes

As with other hierarchical structures the OID nodes at the end of the branches, below which there are no further nodes, are know as leaf nodes. However in the context of SNMP there is one further detail. The OID name space is used to uniquely identify an object to be managed by SNMP, however there could be multiple instances of an object, examples include multiple cooling fans, multiple extents in a database, multiple pressure sensors and almost limitless others. In SNMP a leaf node is appended with an instance number, which can be an integer from zero to the number of instances (the count can start with 1). There are some objects defined in SNMP, of which there can only be a single instance; in these cases the convention is that the instance number zero is appended to the instance. Examples are given below:

SNMP defines a "sysName" that is an administratively defined unique name reported by SNMP for a system (it may have multiple names in directory systems and defined on the system itself). There is only one instance of sysName hence it ends with ".0".

.iso.org.dod.internet.mgmt.mib-2.system.sysName.0

.1.3.6.1.2.1.1.5.0

An item of networking equipment can have several network interfaces, for which there are several objects, each defined with a unique instance number, corresponding to the interface number:

For interface number 3, (counting from 1) the description (ifDescr) object is:

.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifDescr.3

.1.3.6.1.2.1.2.2.1.2.3

For interface number 2 (counting from 1) the object reporting the number of octets received, (ifInOctets) is:

.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifInOctets.2

1.3.6.1.2.1.2.2.1.10.2

Structure below leaf nodes

Tree structures provide a scalable method of uniquely identifying objects; however there are some situations in which tree structures are not necessarily efficient. Generally leaf nodes represent specific items of data to be retrieved or manipulated. In some circumstances, leaf objects represent a sequence of bytes of data. Further processing of the sequence of bytes is required to extract further structure and the data within it. See Appendix C, *VLANs – a bitmap leaf example*. In this situation vendor documentation may need to be consulted and MIB files may need to be examined. MIBs are described in the section called “Management Information Base - MIB”.

Further information

Further information can be found at "OID Repository". See: http://oid-info.com

Views

For many organisations, the use of one or two SNMPv3 users will be sufficient. Often establishing a very few SNMPv3 users on target systems, sometime set to "read only", will meet the management and monitoring needs of an organization and the SNMPv3 users can have full access to all the information (perhaps "read only") exposed by SNMP. This is often the default configuration. There may be occasions when a

greater degree of granularity is required for controlling access to information via SNMPv3. SNMPv3 is designed to be scalable and can provide more granular access control through a mechanism called View-based Access Control Model (VACM). It is worth mentioning that VACM is unlikely to be implemented on equipment that supports only a single user for SNMPv3 access.

The information that can be managed via SNMP is organised in the hierarchical OID structure described above. This tree structure provides scope for controlling access to information via SNMP. By granting access to only specific points in the OID tree, access can be limited to only the items below those points in the tree. Not surprisingly the points in the OID tree to which access is granted (and everything below those points) are referred to as Views.

A single view to the root of the OID tree grants access to everything and this is the usual default on many targets.

Groups

To support scalability and for historical reasons, the design decision was taken to enable access to Views only to be granted to SNMPv3 groups and not SNMPv3 users. It should be remembered, however that it is not unusual for a target to be configured with a single SNMPv3 user in a single SNMPv3 group.

It is because of this constraint that many target systems require one or more SNMPv3 groups to be established to contain the SNMPv3 users. From a practical perspective, it is often more convenient to configure the groups first and then configure the users, assigning each user to a group, even it this involves a single group and only one user.

Access

Access control lists map Groups to Views. Entries in an Access control list will grant access (read only, or read write) for a particular Group to a particular View. The mechanism of establishing access (as with creating groups and users) will be specific to each implementation. Some systems provide a GUI interface while others use a command line mechanism. But in all cases the principal is the same.

Tables

The leaf nodes of parts of the OID tree are often presented in the form of tables. While the mechanism of controlling access to points in the OID tree works well for most situations, it may not work well for tables. In some circumstances it may be necessary not only to restrict access to parts of the OID tree, but to confine access to only certain rows in a table.

Given that some tables can have a large number of rows, restricting access to a large portion of those rows, but not the entire table, by specifying all the corresponding points in the OID tree would be very cumbersome. This specific problem is addressed by a mechanism to allow the use of bit masks to switch access on and off for particular rows in a table. This is a highly specialised situation and will not be described further here. If this arrangement is required then the documentation for the particular target system should be consulted for further information.

Examples

Two examples of configuring views are given below.

Example Cisco switch

A view is created with:

```
snmp-server view <view-name> <oid-tree> {excluded | included}
```

Where <view-name> is an arbitary administrativly assigned name that can be referred to in group creation commands, and <oid-tree> is a defined point in the OID hierarchy.

E.g.

```
snmp-server view systemonly .1.3.6.1.2.1.1 included
```

A view can be removed with:

```
no snmp-server view <view-name>
```

The creation of groups is briefly described above, see the section called “Groups on Cisco devices”. Access to views is established as part of the group creation process, by providing additional parameters to the "snmp-server group" command as:

```
snmp-server group <group name> v3 priv [read <view name>] [write <view name>]
 <additional parameters>
```

Where <group name> is the name of the group, and <view name> is the name of a defined view, as described above. E.g.

```
snmp-server group netmanager v3 priv read sytemonly
snmp-server group configurators v3 priv read sytemonly write itemstomanage
```

In the example above "itemstomanage" is the name of another view example.

Example Linux snmpd

On Linux systems the SNMP agent is usually implemented with snmpd. The snmpd agent is configured with a configuration file such as snmpd.conf. Here views are configured with four parameters and an optional fifth parameter. The first parameter is the word "view" to identify a view directive in the configuration file. The second parameter is the name given to the view for cross-referencing purposes. The third parameter is either "included" or "excluded" to indicate whether the view should be included or excluded from collection of OIDs to which access is granted. The fourth parameter is the OID to which access is granted or denied. The last, optional, parameter is a mask which can be used for very fine grained access control to tables as discussed above.

E.g.

```
view  systemonly included  .1.3.6.1.2.1.1
view  systemonly included  .1.3.6.1.2.1.25.1
```

The creation of groups is described above, see the section called “Groups on Linux systems”.

Access to views for groups is granted with one or more access lines in the configuration file.

E.g.

```
#       context model level  prefix read  write notify (unused)
access netmanager "" any priv  exact  systemonly none  none
```

SNMP Engine ID

Version 3 of SNMP introduced the concept of an Engine ID which is an identifier used for SNMP entities to communicate with each other. Systems may have a number of names, several interfaces, and may use Ipv4 and / or Ipv6 (and other protocols in the past); The SNMPEngineID is specific to SNMP entities.

“Within an administrative domain, an snmpEngineID is the unique and unambiguous identifier of an SNMP engine. Since there is a one-to-one association between SNMP engines and SNMP entities, it also uniquely and unambiguously identifies the SNMP entity within that administrative domain.”

See RFC 3411.

The format of an EngineID can vary from vendor to vendor. RFC 3411 section 6.3 refers to SnmpEngineID Formats:

“The SnmpEngineID TEXTUAL-CONVENTION's fifth octet contains a format identifier. The values managed by IANA are in the range 6 to 127, inclusive.”

It is not normally necessary to specify an SNMP EngineID; these are usually generated on systems automatically. It is possible to modify the EngineID on most systems and there may be rare occasions when this is necessary. It is also not usually necessary to include Engine ID for a target when accessing a target from a management system or via a command line interface.

EngineID problems

Duplicate Engine Ids can sometimes arise if systems are cloned or configurations are copied from one system to another and this can cause problems. Different systems will have their own mechanism for changing the EngineID and the system documentation will need to be consulted. Sometimes the documentation will explain the format or the mechanism used to generate the EngineID. If the documentation provides enough information, then the "correct" EngineID can be created.

The EngineID is usually used in the encryption process of creating SNMPv3 user credentials. Sometimes the EngineID can be changed unintentionally (for example if a system is re-built) and this can result in the SNMPv3 users no longer working on the system. In this situation either the EngineID needs to be reset to its original value, if possible, or the SNMPv3 user credentials re-created, probably by re-creating the users.

SNMP Context

An SNMP Context is described in RFC 1411. The RFC describes in detail how unique information may be retrieved and defines the terms contextName and contextEngineID. The interested reader is referred to the RFC.

Just as, in practice, it is often not necessary to specify the SnmpEngineID (described in the section called “SNMP Engine ID”), so it is rare that a contextEngineID needs to be provided. Some management software

provides an option to specify a contextEngineID if a target requires it. Usually the contextEngineID can be omitted, but the vendor documentation should be consulted in cases where it is required.

Example check_snmp

When using the "check_snmp" plug-in, used by several monitoring systems, if an SNMP Context needs to be specified this can be done with either "-N CONTEXT", or "--context=CONTEXT".

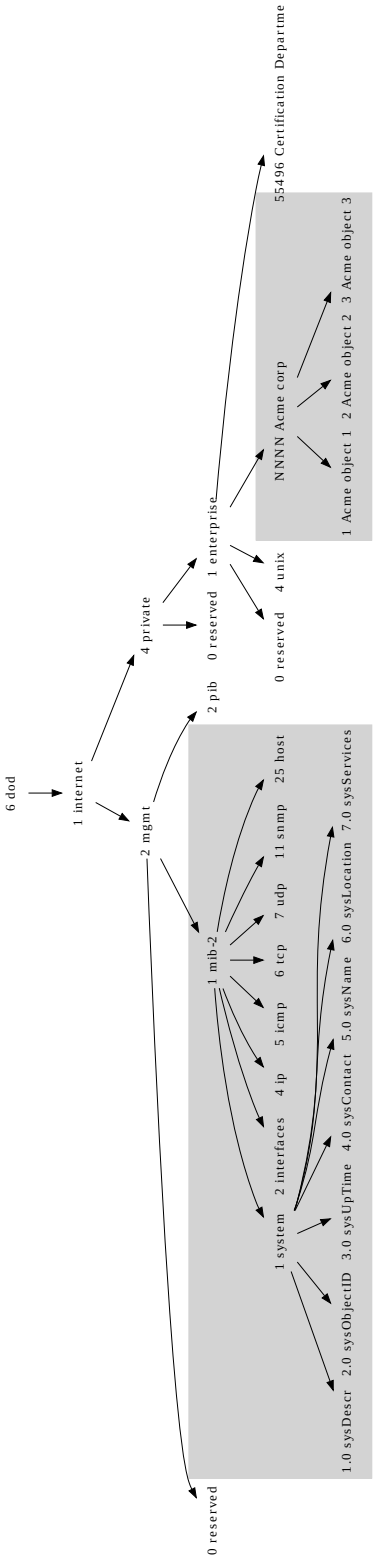
Management Information Base - MIB

Overview

A Management information base (MIB), fully describes a collection of SNMP objects. The OID tree name-space described above, see the section called “Object ID - OID”, provides a unique identifier for each object. SNMP "targets" can be physical items of equipment, items of software or anything else that can be accessed via IP. The total collection of SNMP objects that can be accessed via SNMP for a particular "target" is the Management information base (MIB) for that target.

Often the MIB for a "target" will include a mixture of standards based objects and vendor specific objects. The MIB is a collection of different parts of the OID tree as illustrated below.

Figure 12. Example OIDs in a Vendor MIB



The shaded areas show the parts of the OID included in an example MIB for a vendor.

The identity of objects is defined in the OID tree, but the details of objects are described in MIB files. A MIB file may describe objects from different parts of the OID tree, for example because they are managed by a particular vendor, but sometimes a MIB file will describe all the objects below a given point in the OID tree.

Some of the items in a MIB file refer to the tree structure, broadly "this item is a collection of these items" or "this item is a table of these items". Generally items referring to the structure are read only, but there are some rare exceptions. The main purpose of a MIB file is to describe the "leaf" objects (below which there are no others), which are the useful variables that can be retrieved and sometimes changed.

The variables that can be retrieved, which ones can be changed, and the nature of those variables is determined by the implementation of the SNMP agent or agents within a piece of software or equipment. This is described in MIB files that are used by SNMP Management software.

It is not unusual for the Management information base for a particular "target" (item of equipment or software etc) to be defined by a collection of MIB files that could be quite large. Vendors will often provide a list of the MIB files that are needed to fully describe the MIB for a particular product and the list can be quite extensive.

A Management information base (MIB) file is a structured file, conforming to a defined syntax that can be parsed by SNMP Management software. It is also a plain text format file that can be read by a person with some familiarity with the syntax.

Obtaining MIB files

Many vendors provide MIB files for their products, available from their web sites. Sometimes there will be many files relating to a single product.

There are a number of web sites that provide collections of MIB files, including:

- ByteSphere's MIB Collection (ByteSphere was aquired by EXFO in 2014). The URL is: www.oidview.com/mibs/detail.html
- ICIR MIB index at: www.icir.org/fenner/mibs/mib-index.html (This does not seem to have been updated since 2007)
- mibDepot at www.mibdepot.com/index.shtml

A practical guide to reading MIBs

It can sometimes be useful to read a MIB file, for example to confirm that a particular MIB file is the one required for a task, or to determine what variables can be retrieved and which can be changed. Most descriptions of MIBs begin with a somewhat academic formal description of the syntax; although it may not seem intuitive, perhaps a good place to start is with comments.

Comments

In SNMP MIB files, comments are denoted by a double dash "--" and continue to the end of the line.

```
-- This is a comment.
```

There is no facility for comments to span multiple lines; there is no analogy to the "/* */" syntax in some programming languages. The traditional comment symbols (such as the hash #, or semi-colon ;) can not

be used in MIB files. Multi line comments are often provided, but each line will begin with the double dash "--".

Many MIB files begin with several lines of comments, particularly MIBs from standards bodies.

Although the quality of comments in MIB files varies greatly, often, particularly in those MIBs provided by standards bodies or large reputable vendors, the comments are comprehensive and useful.

Sometimes simply skimming through a MIB file, picking out the comments can be sufficient to gain a good impression of what the MIB file does and the nature of the various variables.

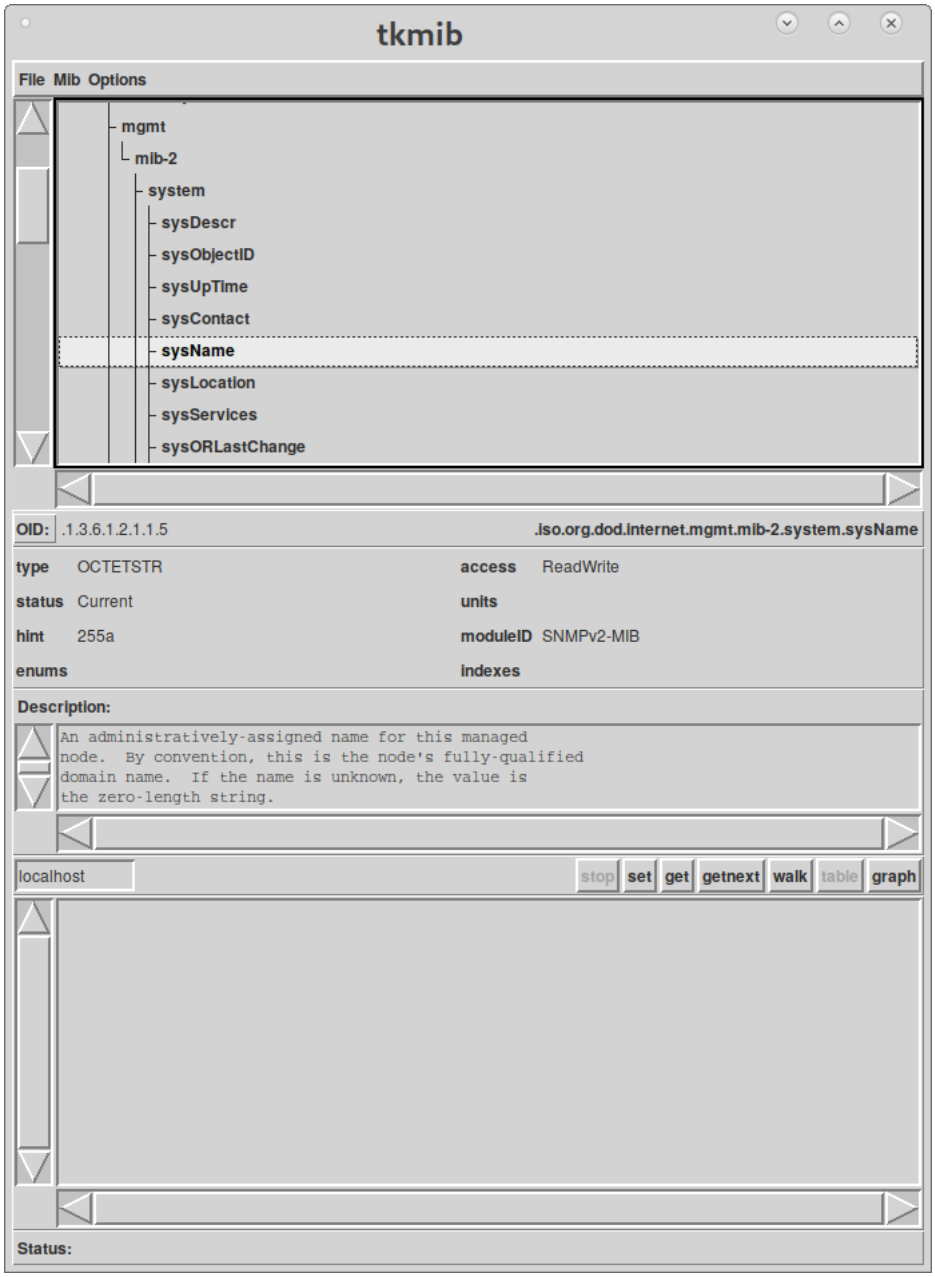
Other useful items are Descriptions, although these can sometimes be confusing.

MIB Description entries

A MIB file will describe several objects. Some objects refer to others (perhaps a table, or sequence etc), others are the leaf items. Broadly, the format consists of an object, sometimes followed by some items of information, including a description, and then the symbols "::=" followed by text defining what the object is (conforming to a defined syntax) in terms of other objects.

The description is free format human readable text enclosed in quotation marks (""). The purpose of descriptions is to allow management software to present a human readable description of an object to someone using the management software (see Figure 13, “tkmib Description”).

Figure 13. tkmib Description



The format of a description is the keyword DESCRIPTION followed by the text enclosed in quotation marks, e.g:

```
DESCRIPTION
    "The value of sysUpTime at the time the interface entered
    its current operational state. If the current state was
    entered prior to the last re-initialization of the local
    network management subsystem, then this object contains a
    zero value."
```

The text of the description begins with a quotation symbol and continues to the closing quotation symbol. The text can span several lines or many pages. Descriptions can be very helpful in explaining what objects are and together with comments can provide a good understanding of the objects in a MIB, however descriptions can sometimes be confusing.

The text of descriptions can be very long. The text is free format and sometimes describes highly specialised and technical details of particular objects. It is not unknown for the text of some descriptions to contain large amounts of highly technical detail; this can make the MIB look complicated. When looking through a MIB file it is important to "match up" pairs of quotation marks enclosing the text of a description and keep in mind that however complex the text may seem, it is simply a description of an object. Note that some descriptions even include samples of code in various languages or pseudo code describing how an object might be processed.

If, while reading through a MIB file, the descriptions and comments are temporarily disregarded, the structure of the MIB file can be seen. There is some structure that relates to the MIB file itself, in terms of "house keeping" that is "wrapped around" the MIB file, but perhaps most often the reader is interested in the "leaf" objects representing the actual variables that can be read and sometimes changed.

Definitions

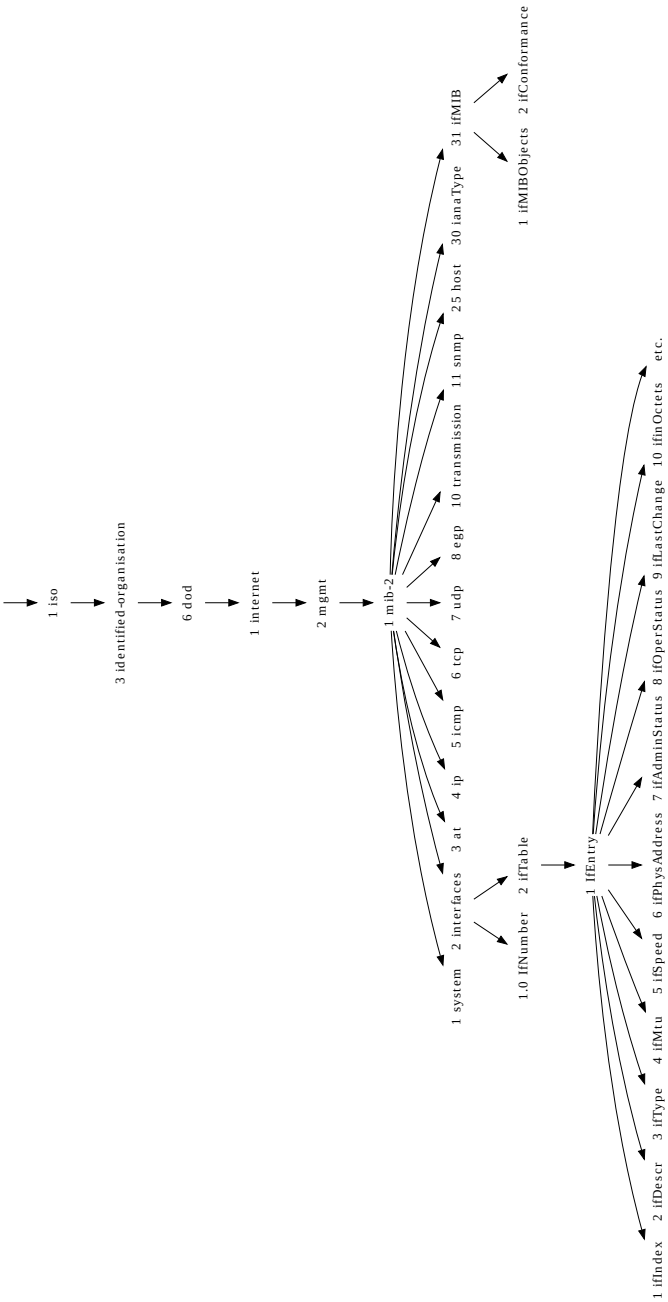
The definitions in MIB files can, perhaps, best be explained with examples; some examples from the Interface MIB (IF-MIB) are used here. It may be useful to refer to the MIB file; a copy of the MIB file can be found at the following URL:

circitor.fr/Mibs/Mib/I/IF-MIB.mib [https://circitor.fr/Mibs/Mib/I/IF-MIB.mib]

The examples used are objects under the "interfaces" object in the OID tree as shown below.

The structure of definitions was very broadly and briefly described above in the section on descriptions. Here definitions are described in more detail starting with "leaf" objects.

Figure 14. OID Interfaces



Leaf objects

The "leaf" objects represent the variables that can be managed and are the "simplest" objects (containing nothing else). An example "leaf" object is shown below to illustrate the structure and syntax (what the object actually represents is not important for considering the structure).

```
ifMtu OBJECT-TYPE
  SYNTAX   Integer32
  MAX-ACCESS read-only
  STATUS   current
```

```
DESCRIPTION
    "The size of the largest packet which can be sent/received
    on the interface, specified in octets. For interfaces that
    are used for transmitting network datagrams, this is the
    size of the largest network datagram that can be sent on the
    interface."
::= { ifEntry 4 }
```

In the above example an object called "ifMtu" is defined to be part of another "containing" object called "ifEntry" and is distinguished from other objects in the "ifEntry" object, by being number four. The object being defined appears at the beginning of the definition and what it is (in terms of other objects) is given at the end of the definition, following the symbols "::=". The use of the "::=" symbols is because of the syntax used in MIB files (rather than simply an equals sign often seen in programming languages and scripts). In this example, following the defining symbols, "::=" the definition is enclosed in curly brackets "{}". The "4" in this example indicates that the object is number 4 within "ifEntry" and is a "leaf" object.

This particular object is .1.3.6.1.2.1.2.2.1.4 or

iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifMtu.4

There are a number of items between the object ("ifMtu" in this case) and the definition, "::= { ifEntry 4 }", including the description explained in the section called “MIB Description entries”. In this example the description is relatively short, however, as mentioned above, descriptions can sometimes be very long, separating an object from the definition part by many lines or even several pages; the reader may need to search for the matching closing quotes to then find the definition of the object.

As well as the description, several other items appear in the example above. Following the object, ("ifMtu" in this case), is the key word OBJECT-TYPE indicating here that this is an object rather than any other kind of definition.

The second line, "SYNTAX Integer32", explains what kind of object this is (in this case, simply a 32 bit Integer that is not expected to change). Possible values for "SYNTAX" include "DisplayString", a text string, and "Gauge32", an integer that typically changes over time. A table of just a few values is shown below.

The third line of the example is "MAX-ACCESS read-only". The "MAX-ACCESS" item defines how a variable can be accessed, in this case, as the name of the value suggests, the variable can be read but not changed. Alternative values, for "leaf" objects include "read-write" for variables that can be read and changed, possibly the value "accessible-for-notify", and the theoretical "write-only". There are other values for "MAX-ACCESS" for "non-leaf" objects.

The fourth line, "STATUS current", shows that this object is a current object (as of the date of the MIB file) and therefore can be used. Alternative values include "deprecated" and "obsolete" indicating that this object was used in the past, but should be avoided in future. Older MIB files may include "mandatory" instead of "current".

SYNTAX values	**Description**
Counter	An increasing 32-bit integer that wraps back to zero when the maximum value is reached
Counter64	An increasing 64-bit integer
IpAddress	A 32-bit IPv4 address
TimeTicks	A 32-bit integer measuring hundredths of seconds.

A second example of a "leaf" object is shown below:

```
ifOperStatus OBJECT-TYPE
  SYNTAX INTEGER {
        up(1),    -- ready to pass packets
        down(2),
        testing(3),  -- in some test mode
        unknown(4),  -- status can not be determined
               -- for some reason.
        dormant(5),
        notPresent(6),  -- some component is missing
        lowerLayerDown(7) -- down due to state of
                 -- lower-layer interface(s)
      }
  MAX-ACCESS read-only
  STATUS   current
  DESCRIPTION
      "The current operational state of the interface. The
      testing(3) state indicates that no operational packets can
      be passed. If ifAdminStatus is down(2) then ifOperStatus
      should be down(2). If ifAdminStatus is changed to up(1)
      then ifOperStatus should change to up(1) if the interface is
      ready to transmit and receive network traffic; it should
      change to dormant(5) if the interface is waiting for
      external actions (such as a serial line waiting for an
      incoming connection); it should remain in the down(2) state
      if and only if there is a fault that prevents it from going
      to the up(1) state; it should remain in the notPresent(6)
      state if the interface has missing (typically, hardware)
      components."
  ::= { ifEntry 8 }
```

This "leaf" object, "ifOperStatus", is the eighth object within "ifEntry" as shown in "::= { ifEntry 8 }". The value of this object is also an Integer, but the possible values are given as a comma separated list annotated by comments (text beginning with "--" to the end of the line).

Looking at the "IF-MIB" file, it can be seen that the second object within "ifEntry" is "ifDescr". The value of this object is a "DisplayString" (free text), but the MIB shows is has a maximum length of 255 bytes:

SYNTAX DisplayString (SIZE (0..255))

A variety of objects are included in the IF-MIB file, a few examples of which are shown below:

The first "leaf" object within "ifEntry" is an index value (in this case an integer for each interface in a target):

```
ifType OBJECT-TYPE
  SYNTAX   IANAifType
```

The above examples provide a broad impression of the structure of "leaf" objects.

The leaf objects are the simplest, containing no other objects, but are found within other "containing" objects.

Non leaf objects

Non leaf objects are perhaps best explained by example and the above "leaf" objects will be used here.

All the above example "leaf" objects are "contained" in the "ifEntry" object shown below:

```
ifEntry OBJECT-TYPE
  SYNTAX   IfEntry
  MAX-ACCESS not-accessible
  STATUS   current
  DESCRIPTION
      "An entry containing management information applicable to a
      particular interface."
   INDEX  { ifIndex }
   ::= { ifTable 1 }
```

This object, "ifEntry" is object number 1 within the "ifTable" object (in this example, there is also an object zero within "ifTable"). The definition shows that it is object number 1 "::= { ifTable 1 }". As with the "leaf" examples above, this establishes the context within the OID tree.

The third line, "MAX-ACCESS not-accessible", shows that this is a "containing" object, and in this case can not be accessed itself. It is the "leaf" objects it contains that can be read (in this case all the leaf objects happen to be read only, but often "leaf" objects can be changed). Other possible values for "MAX-ACCESS" for containing objects, include "read-create" and "accessible-for-modify".

The second line "SYNTAX IfEntry", is similar to the "SYNTAX" lines in the "leaf" examples, except that "IfEntry" is not one of the standard simple values used with leaf objects (such as Integer32). In this case the item is specific to the "ifEntry" object and because of this (and the fact that it is not a standard value) it must be defined elsewhere in the MIB file. Notice the difference in the names, in this case distinguished by the capitalisation of the first letter; "ifEntry" is an object, "IfEntry" is what kind of object "ifEntry" is (defined elsewhere).

Between the "DESCRIPTION" and the definition, "::= { ifTable 1 }", is the line "INDEX { ifIndex }". The INDEX shows that this object contains items that are indexed and the line shows that the value of the index will be held in the "ifIndex" object defined elsewhere.

In the above example, "SYNTAX IfEntry", shows that the object is of the form "IfEntry" (Notice capitalisation) defined elsewhere in the MIB file. In the case of the IF-MIB file, the definition of "IfEntry" immediately follows the object definition and is shown below:

```
IfEntry ::=
  SEQUENCE {
    ifIndex         InterfaceIndex,
    ifDescr         DisplayString,
    ifType         IANAifType,
    ifMtu          Integer32,
    ifSpeed         Gauge32,
    ifPhysAddress      PhysAddress,
    ifAdminStatus      INTEGER,
    ifOperStatus      INTEGER,
    ifLastChange      TimeTicks,
    ifInOctets       Counter32,
```

```
    ifInUcastPkts      Counter32,
    ifInNUcastPkts     Counter32, -- deprecated
    ifInDiscards      Counter32,
    ifInErrors       Counter32,
    ifInUnknownProtos    Counter32,
    ifOutOctets       Counter32,
    ifOutUcastPkts     Counter32,
    ifOutNUcastPkts     Counter32, -- deprecated
    ifOutDiscards      Counter32,
    ifOutErrors       Counter32,
    ifOutQLen        Gauge32,  -- deprecated
    ifSpecific       OBJECT IDENTIFIER -- deprecated
   }
```

In the above example, there are no items between the item, "IfEntry", (Notice capitalisation) and the definition "::= SEQUENCE { }". Within the definition, in particular there is no "OBJECT-TYPE". This does not define an object, rather something referred to in an object. The purpose of this definition is to explain that an "IfEntry", (Notice capitalisation) is a "SEQUENCE"; A "SEQUENCE" is an ordered list of objects of different kinds. Other values include "SEQUENCE OF" which is an ordered list of objects of the same kind.

Between the curly brackets are listed all the objects contained within this sequence, some of which are the leaf examples described above.

Because this is a "SEQUENCE", i.e. an ordered list of objects of different kinds, there must be an index value to define each object's place in the list. Because of this the object, "ifEntry" that uses this definition, and hence the "SEQUENCE", includes "INDEX { ifIndex }". The "ifIndex" is the first object in the "SEQUENCE" above.

Tables

Some objects are tables of other objects, consisting of one or more rows. Each row has the same structure, so given that a "SEQUENCE OF" is an ordered list of objects of the same kind, a table is a "SEQUENCE OF" rows.

In the example above, the "ifEntry" object is held within the "ifTable" object, "::= { ifTable 1 }". The definition of the "ifTable" object is shown below:

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

The "ifTable" object is item two in the interfaces object, ::= { interfaces 2 }. Being a table, all the rows are the same, thus the SYNTAX entry is a SEQUENCE OF rows; in this case each row is an "IfEntry" already defined above. As mentioned above, the "ifTable" object is located within the "interfaces" object. The interfaces object is located within an object called mib-2: "interfaces OBJECT IDENTIFIER ::= { mib-2 2 }".

The entire definition of "interfaces" is shown above, it is a single line and is item two in the "mib-2" object, "::= { mib-2 2 }".

The "mib-2" object is not defined in the example MIB being considered here, IF-MIB, but is a separate MIB file. Each MIB file describes in detail the objects in a portion of the OID tree and has links, like the one shown above to OIDs further up the tree.

MIB file structure

The bulk of the information contained within a MIB file has been described above, but surrounding this is some additional "house keeping" information.

There may be a number of comment lines at the beginning of a MIB file, or none, however the MIB file formally starts with a "DEFINITIONS ::= BEGIN" statement, e.g:

```
IF-MIB DEFINITIONS ::= BEGIN
```

The MIB file finishes with an "END" statement, which is simply the word "END" on a line with nothing else.

Following the "BEGIN" statement, there can be a number of other sections.

A MIB file can make use of items defined in other MIB files by importing them. This is achieved through an "IMPORTS" section that instructs the software parsing the MIB file to import various items. The structure of this section is the word "IMPORTS" followed by a number of import instructions and concluding with a semi-colon. Each import instruction is a comma separated list of items followed by the word "FROM" and the name of the MIB file containing the items, e.g:

```
IMPORTS
  MODULE-IDENTITY, OBJECT-TYPE, Counter32, Gauge32, Counter64,
  Integer32, TimeTicks, mib-2,
  NOTIFICATION-TYPE            FROM SNMPv2-SMI
  TEXTUAL-CONVENTION, DisplayString,
  PhysAddress, TruthValue, RowStatus,
  TimeStamp, AutonomousType, TestAndIncr  FROM SNMPv2-TC
  MODULE-COMPLIANCE, OBJECT-GROUP,
  NOTIFICATION-GROUP            FROM SNMPv2-CONF
  snmpTraps                FROM SNMPv2-MIB
  IANAifType                FROM IANAifType-MIB;
```

The "MODULE-IDENTITY" section contains administration information about the MIB file. This also has a reference to a containing OID. The "MODULE-IDENTITY" section contains a number of items each followed by free text enclosed in quotation marks. Items include "LAST-UPDATED", "ORGANIZATION", "CONTACT-INFO" and "DESCRIPTION" described above. There can also be a number of "REVISION" items, each followed by a DESCRIPTION.

Additional items

It is not the intention to provide a comprehensive description of MIB files here, rather a sufficient overview to allow administrators to make use of MIB files.

An additional item worth mentioning is "DISPLAY-HINT which is an item intended to suggest to parsing software, how an object can be presented. E.g.

```
DISPLAY-HINT "d"
```

The above example is for a decimal number, such as a 32 bit Integer.

Technical detail

This section on MIBs concludes where most descriptions of MIBs begin, with some technical detail. A number of MIB files provide resources used by many other MIB files. The "SNMPv2-SMI" MIB file provides a number of important definitions and links to the top of the OID tree.

As explained above and can be seen from the examples, MIB files conform to a defined syntax. The syntax used by MIB files is a form of ASN.1.

Traps and InformRequests

Systems that are normally the "targets" managed by various management systems or accessed interactively, can also be configured to send unsolicited SNMP messages. These were traditionally known as "traps" and used to send alerts in the event of problems. Traps are sent as single packets (usually via UDP) and are not acknowledged and may be lost. To address this the "InformRequest" notification mechanism was introduced, where the system receiving a notification sends back an acknowledgement. It is worth noting that the system that receives notifications ("traps" and / or "InformRequests") does not need to be the system that normally manages targets.

There are a number of reasons why Traps and "InformRequests" might be used. As an example, a monitoring system may poll a number of targets, interrogating each for a large amount of information. It might be that the polling interval is too long to report some urgent issues in a timely manner. There may also be circumstances in which it is not appropriate for a management system to frequently poll a target for large amounts of information. In this situation, it could be sensible for the management system to infrequently poll, effectively asking "Are you still there? And are you well enough to report problems?". Here the "target", carefully configured, would be relied upon to notify the management system of any issues.

Because of the asynchronous unsolicited nature of Traps and "InformRequests", they should be configured with caution. A particular danger is flapping systems, where something endlessly cycles between an error / broken / failed state and a repaired / working / normal state. In this situation, if the system sends a notification ("Trap" or "InformRequest") each time the state changes, the system receiving the notifications will be bombarded with messages. A system sending notifications should check for flapping conditions and implement appropriate timed-outs. Before implementing a vendors Traps and / or InformRequest options, the documentation should be checked to ensure appropriate timed-outs are implemented and flapping is dealt with. If an administrator is configuring "Traps" and / or "InformRequests", this should probably be done via a script or other mechanism that implements appropriate checks.

A particular hazard is worth highlighting. In some situations events in one system can affect others causing a cascade of events. If a system is flapping and affecting others, the effect can be multiple systems flapping and if each were configured to send a notification is response to each state change, this could result is very large numbers of messages. A classic example of this kind of problem is spanning-tree issues in network switch infrastructures.

Despite the warnings given above, Traps and "InformRequests" can still be useful. To use these facilities, one or more systems need to be configured to receive the Traps and / or "InformRequests", and other systems need to be configured to send them.

Configuring systems to receive Traps and / or "InformRequests" is specific to particular systems. Each management system will have its own configuration arrangements and the documentation will need to be consulted. On Linux systems, for example, "snmptrapd" is often the notification receiver, and tools such as "snmptrapfmt" and "snmptt" provide "trap" handlers.

The way in which systems are configured to send notifications varies from system to system and again the documentation will need to be consulted. There are many systems that are configured through a GUI interface (sometimes WEB based) and to configure the system to send "Traps" and / or "InformRequests", the interface simply prompts for the IP addresses of the system(s) to receive the notifications. Often there is no other configuration. Administrators may be tempted to simply supply appropriate IP addresses, however it is important that the documentation is consulted to understand when and how notifications will be sent and what measures are in place to limit them and deal with flapping.

SNMPv3 over secure transports

The Transport Security Model (TSM) enables security to be implemented at the transport layer. This is described in RFC 5591. Transport Layer Security (TLS), over TCP, and Datagram Transport Layer Security (DTLS), over UDP can be used to tunnel SNMPv3. Transport Layer Security (TLS) is the successor protocol to Secure Sockets Layer (SSL). Some systems also support the use of SSH.

At the time of writing, many vendors still do not support the use of SNMPv3 over secure transports. It will be necessary to check the vendor documentation to check whether any of the mechanisms described above are supported and if so how they are configured.

On Linux systems, snmpd supports the mechanisms described above . See man snmpd

For further information, see:

http://snmp.com/products/techinfo/secmodels.shtml

Further end to end examples

A few case studies are provided here to further illustrate the configuration of SNMPv3 on various systems.

MIB browser to Linux machine

For this example, a test Linux machine has been set up to allow experimenting with SNMPv3. A user with read write access will be set up for testing and a MIB browser will be used to access the Linux machine. In this example the Linux machine will be called "litmus" with a fully qualified domain name of "litmus.lab.local."

The common parameters used in this example are shown below:

Parameter	**Value**
SNMPv3 username	tester1
Security level	AuthPriv
Authentication protocol	SHA
Authentication password	Nuef0dapfps#
Privacy protocol	AES
Privacy password	Pf1bs#n#buuac

On the Linux machine, the "net-snmp" packages have been installed for this example. Information about Views, Groups and Users is held in a one or more configuration files and in this example the configuration file is /etc/snmp/snmpd.conf.

A view is set up to grant access to almost everything for testing purposes. The OID is:

.iso.identified-organization.dod.internet (.1.3.6.1)

The view will be called everything and is set up by including the following in the configuration file:

```
view  everything included  .1.3.6.1
```

A group is set up to contain the user and is granted read and write access to the above view. The group is called testers and is implemented by including the following in the configuration file:

```
group testers usm    tester1
```

The following line is included in the configuration file to grant access to the view for the group:

```
access testers "" any priv  exact  everything everything  none
```

Before creating SNMPv3 users with the net-snmp-create-v3-user command, any running snmp daemons need to be stopped; they can be restarted once the user(s) have been created.

The user can be set up with the following command, entered as a single line on the command prompt:

```
net-snmp-create-v3-user -A Nuef0dapfps# -a SHA -X Pf1bs#n#buuac -x AES tester1
```

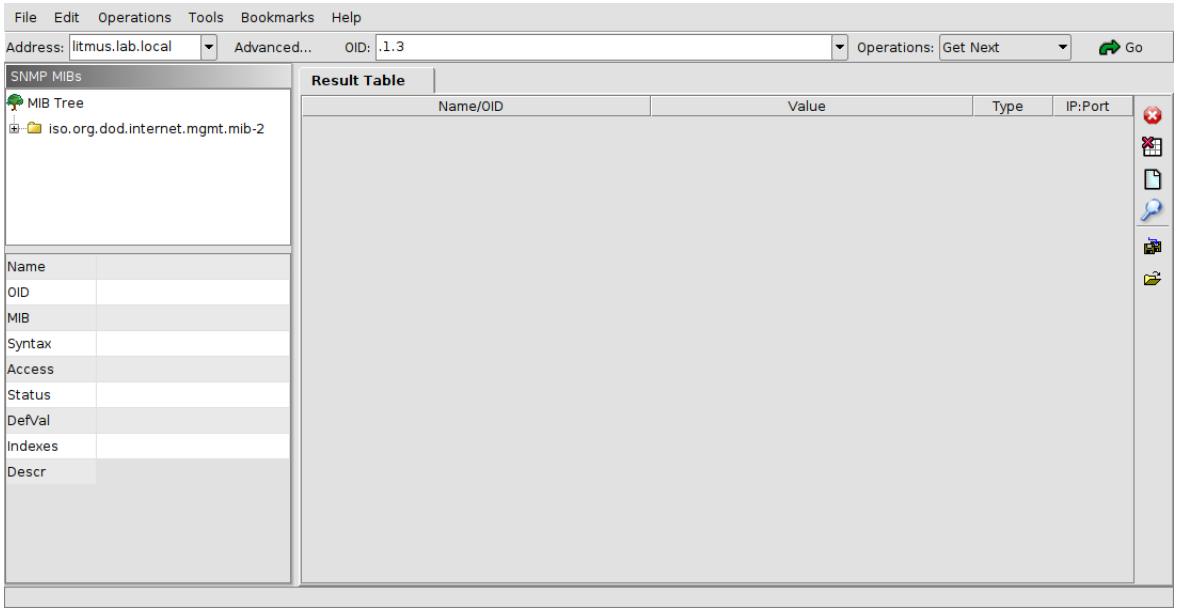
Note that the optional -ro flag is omitted to grant the user read write access.

This will add an entry in a configuration file, where the contents are encrypted once the snmpd daemon is started, this file is typically /var/lib/snmp/snmpd.conf. The general configuration file such as /etc/snmp/snmpd.conf is updated with a simple entry:

```
rwuser tester1
```

For testing the "MIB Browser" product from iReasoning will be used. Once the browser is launched the target can be entered in the "Address:" field.

Figure 15. MIB Browser address



The "Advanced..." button opens up a properties dialogue where the SNMP security parameters can be entered.

From the "SNMP Version" drop down menu, "3" should be selected which will then open up the fields for the SNMPv3 parameters. The commmon parameters described above will be used.

MRTG to Network Switch

For this example, the requirement is to use the performance graphing software, MRTG, to gather data on the traffic from some ports on a Network Switch, to produce graphs. Here the MRTG application is functioning as the manager and communicating via SNMP to a target switch to retrieve data. In this example the switch is a Cisco switch called "sycamore" with a fully qualified domain name of "sycamore.local". Traffic from two ports, 47 and 48 will be graphed. Both systems will use SNMPv3 and Authentication with Privacy (encryption). The administrators of the MRTG system and the switch will agree the common parameters to use. The common parameters used in this example are shown below:

Parameter	**Value**
SNMPv3 username	grapher1
Security level	AuthPriv
Authentication protocol	SHA
Authentication password	Nuef0dapfps#
Privacy protocol	AES256
Privacy password	Pf1bs#n#buuac

On the switch, an SNMPv3 read-only user will be created with the values in the table above and placed in an SNMPv3 group. For additional security, a View will be created to which the group is granted access.

The view will be called "intfaceonly" and will limit access to the following OID:

.iso.identified-organization.dod.internet.mgmt.mib-2.interfaces (.1.3.6.1.2.1.2)

The IOS command to create the view is:

```
snmp-server view intfaceonly .1.3.6.1.2.1.2 included
```

A group, called "graphers" will be created to hold the user and granted access to the view with the IOS command:

```
snmp-server group graphers v3 priv read intfaceonly
```

Once the group has been created, the user can be created with the parameters in the table above and placed in the group. For this example, the IOS command would be as shown below.

```
snmp-server user grapher1 graphers v3 auth SHA Nuef0dapfps#
 priv AES256 Pf1bs#n#buuac
```

MRTG is configured with one or more configuration files. The configuration items relevant to SNMPv3 are illustrated here.

To enable SNMPv3, the configuration file should include the directive:

```
EnableSnmpV3: yes
```

Note that the the perl "Net::SNMP" library also needs to be installed to support SNMPv3. Also the perl "Crypt::Rijndael" library is required to support AES128 AES192 and AES256 encryption.

A full description of the syntax of MRTG configuration files is out of scope of this document, but a few points are worth highlighting. The target definition still requires a legacy community value to be included as used with older versions of SNMP. The syntax requires <legacy community name>@<target>, but when using SNMPv3, the <legacy community name> can be set to an arbitrary value, such as null. As the name suggests, Multi Router Traffic Grapher (MRTG), was originally designed to graph data from network equipment such as routers, and as such it has a couple of short cuts worth noting. MRTG is designed to gather data in pairs (e.g. in and out values), and while any two OIDs can be specified, if the OID identifying an interface number is provided, it only has to be provided once and MRTG will automatically preppend the parts of the OID to retrieve in and out frame counts.

For the example the OIDs required are:

.iso.identified-organization.dod.internet.mgmt..mib-2.interfaces.ifTable.ifEntry.ifInOctets.<if-value> (.1.3.6.1.2.1.2.2.1.10.<if-value>)

And the above, but ending with:

.ifOutOctets.<if-value> (.16.<if-value>)

Where the <if-value> is the specific OID for a given interface. It should be noted that often the interface identifiers do not simply follow a simple sequential numbering scheme starting at one for the first interface and the documentation for a target will need to be consulted.

For this example, interfaces are identified with sequential numbers and so the MRTG short cut facility can be used and just the interface number provided once, rather than the full OIDs for the "in" and "out" traffic:

```
Target[sycamore_uplink47]: 47:null@sycamore.local:::::3


Target[sycamore_uplink48]: 48:null@sycamore.local:::::3
```

After the target entry, a number of colon separated values can be provided, the last of which is the SNMP version, which should be set to 3 (unused values can be omitted).

The SNMPv3 parameters are specified with the SnmpOptions entry on a single line (shown here on several lines to fit on the page):

```
SnmpOptions[sycamore_uplink47]: username=> grapher1, authprotocol=> sha,
  privprotocol=> aes256, authpassword=> Nuef0dapfps#,
  privpassword=&gt;Pf1bs#n#buuac
```

The correspoding entries for port 48, would be:

```
SnmpOptions[sycamore_uplink48]: username=> grapher1, authprotocol=> sha,
  privprotocol=> aes256, authpassword=> Nuef0dapfps#,
  privpassword=> Pf1bs#n#buuac
```

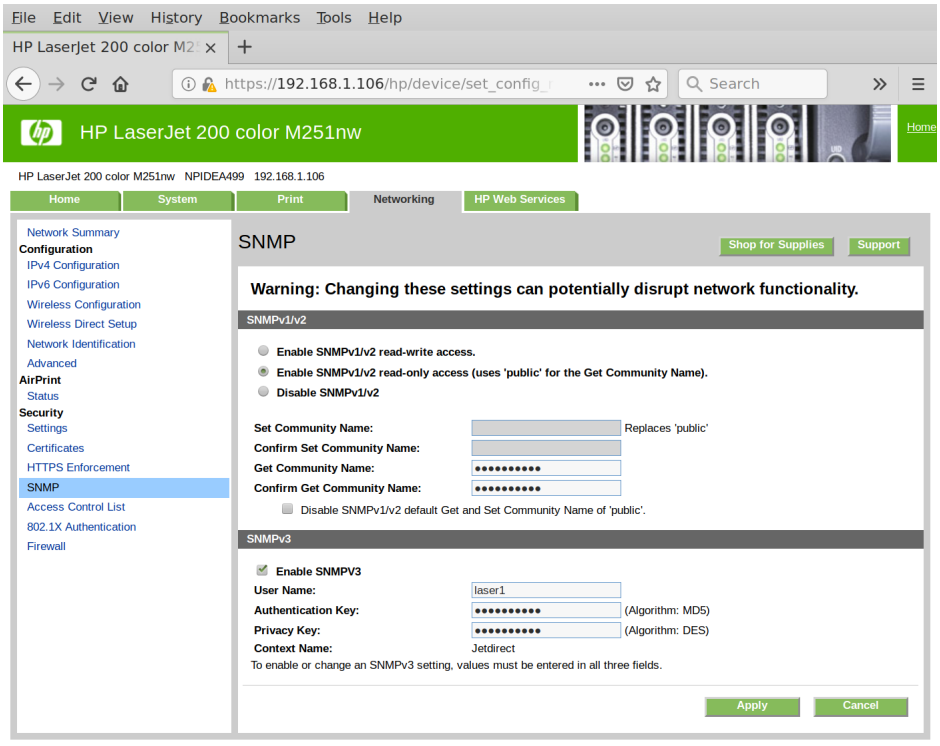
check_snmp to printer

While smaller InkJet printers are unlikely to be managable, lager shared networked printers used in offices are often manageable via SNMP and because problems with office printers affect many people, it is useful to monitor them.

The information exposed via SNMP will depend upon the manufacturer whos documentation should be consulted, together with the MIBs made available.

The way in which printers are configured varies greatly between manufactures and often between models. Sometimes it is possible to access a printer via SSH or telnet, but typically access for management and configuration is via a web interface. An example of a configuration web interface for a LaserJet printer is shown below.

Figure 16. Printer



For a device such as a printer, sometimes only a single SNMPv3 user will be supported and the authentication and privacy protocols may also be limited. For systems that only have a single user, there may be no configuration for groups or views. The administrator of the printer and the administrator of the monitoring

system will need to agree on the parameters to be used which may be limited by the capabilities of the printer. For this example the following parameters will be used:

Parameter	**Value**
SNMPv3 username	laser1
Security level	AuthPriv
Authentication protocol	MD5
Authentication password	Nuef0dapfps#
Privacy protocol	DES
Privacy password	Pf1bs#n#buuac

In this example the printers fully qualified domain name is laserjet1.local

The printer will be configured with the appropriate SNMPv3 values, typically via a Web interface as shown above.

Many monitoring systems such as Icinga, Naemon, Nagios, Shinken, Sensu, etc use a common set of plug-in modules, including the check_snmp module for monitoring via SNMP.

The way in which the check_snmp module is configured will depend on the monitoring system in use and the possible use of third party configuration tools (often web based). Icinga 2 has its own configuration language which introduces a level of indirection to the actual configuration of the module. The configuration of check_snmp itself is described below. The plugin can be used directly from the command line interactively and the command line parameters specified are those that are included in a configuration file.

In this example the SNMP uptime is monitored

.iso.identified-organization.dod.internet.mgmt.mib-2.system.sysUpTime.0 (.1.3.6.1.2.1.1.3.0)

```
check_snmp -H laserjet1.local -P 3 -L authPriv -a MD5 -A Nuef0dapfps#
  -x DES -X Pf1bs#n#buuac -U laser1 -o 1.3.6.1.2.1.1.3.0
```

A. Windows PowerShell SNMP enabling

Some years ago, Microsoft began removing support for SNMP as a "target" from the various Windows operating systems. It is important to note that many Management systems still run on Windows platforms and can be used to manage targets, such as network equipment and printers via SNMP.

At the time of writing, the SNMP agent is no longer being listed as an option on the latest versions of Windows. However the facility can still be enabled via PowerShell.

See:

How to Install and Configure SNMP Service on Windows 10?

https://theitbros.com/snmp-service-on-windows-10/

"

> In Windows 10 1803 and later (1809, 1903), the SNMP service is considered deprecated and is not listed in the Windows features in the Control Panel list.
>
> Microsoft plans to completely remove the SNMP service in the next Windows builds because of the security risks associated with this protocol. Instead of SNMP, it is recommended to use the Common Information Model (CIM), which is supported by Windows Remote Management. On the current builds of Windows 10, the SNMP service is hidden.
>
> The SNMP service is now missing from the Windows 10 image and can only be installed as Feature On Demand (FoD).
>
> "

See:

Features Removed or Deprecated in Windows Server 2012:

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/hh831568(v=ws.11)#snmp

B. Mechanics of SNMP

Many network administrators will manage systems via SNMP by using some management software, such as a monitoring tool, statistics / graphing tool or perhaps a MIB browser. Administrators may need to configure systems to talk to each other via SNMP and will need to consult vendor documentation and MIB files to determine what can be managed. Administrators may not often need to be concerned with the mechanics of SNMP communications, however a brief overview is given here.

Systems to be managed by SNMP will make a number of parameters available to be either interrogated or changed. Each parameter is uniquely identified with an Object Identifier (see the section called “Object ID - OID”). The collection of parameters that a target has is known as a Management Information Base (MIB). The nature of the various parameters (what they represent and how they should be handled) is documented in a format that software can parse and humans can read, called MIB files (see above).

Unless encapsulation mechanisms are in use, SNMP communications occur over UDP, typically ports 161 and 162. For SNMP itself there are no sessions established. Broadly, SNMP interactions can be considered in terms of three elements.

GET requests are sent from a managing system to a target system to retrieve the value of a given parameter (identified by its OID). This will occur in a single UDP datagram, typically using port 161. The target will respond with the relevant information (in one datagram), which the managing system will process using information in the relevant MIB file. There are some variations on this theme described below.

SET requests are sent from a managing system to a target system to change the value of a given parameter (identified by its OID). This will occur in a single UDP datagram, typically port 161. The target will respond with confirmation that the value has been changed (in one datagram).

TRAP messages are sent from a system that is usually the target to the managing system to inform the managing system of an alert. These are also sent as single UDP datagrams, typically port 162.

It is the use of "send and forget" single UDP datagrams with the above fundamental actions, that explains the word "Simple" in the protocol name. There are a number of variations on the above three actions and all of these can be used to provide a complex facility.

GET NEXT requests are similar to GET requests, except that the target responds with the next variable in its Management Information Base (MIB) and the variable's OID. Using a succession of GET NEXT requests, a management system can traverse the MIB of a target.

GET BULK was a feature introduced in SNMPv2 and allows several GET requests to effectively be bundled in to one request sent in one UDP datagram. The target will respond with all the requested parameters, but limited by the amount of information that can be accommodated in a datagram, which means that some information could be lost.

The INFORM REQUEST mechanism was also introduced with version 2. This is now effectively used as a replacement for TRAP messages. A system that is normally a target, will sent an INFORM REQUEST message instead of a TRAP message. The managing system should acknowledge the INFORM REQUEST.

Full details of SNMP can be found in RFC 1157, "A Simple Network Management Protocol (SNMP)".

C. VLANs – a bitmap leaf example

It the section called “Structure below leaf nodes” it was explained that sometimes OID leaf nodes return information that requires further processing, such as bitmaps (rather than items such as strings or counters). This section describes a particular example.

Managed network switches are a popular target for management via SNMP and such switches usually support layer-two Virtual Local Area Networks (VLANs). This provides a useful example of OID leaf nodes returning bitmpas. Many management solutions will hide the details and provide VLAN information transparently, however it can be useful to be aware of VLAN bitmaps for some situations (e.g. using command line tools such as snmp get or MIB Browsers that do not fully decode the bitmaps).

The most commonly used standard for VLANs is IEEE 802.1Q (the details are outside the scope of this document). The IEEE 802.1Q standard allows up to 4096 VLANs to be configured. Ports on a switch can be configured as "Edge" / "Access" ports with a single VLAN or trunking ports carrying any number of the 4096 VLANs. Hybrid ports can also be set up which are combinations of the above two configurations.

Managed switches can have several standard ports (e.g. 8, 12, 24, 48 etc) and a number of "uplink" ports (2, 4 or more). A common configuration is for a switch to have 52 ports, any of which may be configured with up to 4096 VLANs. This gives a potential 212,992 combinations. Navigating this via a tree arrangement would be cumbersome which is why bitmaps are often used in this situation.

There are at least two approaches to this situation with various options for each. Once approach is a VLAN centric solution in which there is an OID leaf node for each of the 4096 VLANs that returns a bitmap indicating which switch ports (if any) carry a particular VLAN. An alternative approach is to have an an OID leaf for each switch port, returning a bitmap indicating which VLANs are implemented on that port. This approach will be described further below (but the concepts apply to both approaches).

With a leaf node for a particular switch port returning a bitmap showing the VLANs implemented on that port, interrogating the port via SNMP will return a string of 4096 bits (512 bytes), typically with 1 indicating a VLAN is implemented and 0 indicating that a VLAN is not implemented on the port.

There are several ways in which the above bitmap can be implemented which can be described as "big-endian" and "little-endian" both in terms of bytes and bits. Broadly successive bytes can correspond to increasing values for the VLAN number, with the first byte mapping VLANs 1 to 8 and the last mapping VLANs 4089 to 4096; the opposite arrangement returns the highest VLAN numbers 4089 to 4096 with successive bytes returning decreasing VLAN numbers. There are also two ways to map the bits in a byte to VLAN numbers, with the most significant bit corresponding to either the highest VLAN number or the lowest. An example implementation is shown below:

byte	1								2								...	512							
bit	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	...	1	2	3	4	5	6	7	8
VLAN	8	7	6	5	4	3	2	1	16	15	14	13	12	11	10	9	...	4096	4095	4094	4093	4092	4091	4090	4089

Vendor documentation will need to be consulted to determine how bitmaps are implemented. The author is unaware of any standard and each vendor has its own implementation.

D. RFCs

RFC	Description
1155	Structure and Identification of Management Information for TCP/IP-based Internets
1156	Management Information Base for Network Management of TCP/IP-based internets
1157	A Simple Network Management Protocol (SNMP)
2104	HMAC: Keyed-Hashing for Message Authentication
3411	An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks
3412	Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)
3413	Simple Network Management Protocol (SNMP) Applications
3414	User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)
3415	View-based Access Control Model
3417	Transport Mappings for the Simple Network Management Protocol (SNMP)
3418	Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)
5590	Transport Subsystem for the Simple Network Management Protocol (SNMP)
5591	Transport Security Model for the Simple Network Management Protocol (SNMP)
5953	Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)

Books

SNMP Mastery (IT Mastery). Michael W Lucas. Tilted Windmill Press. Paperback (15 Feb. 2020). 978-1642350371.

Net-SNMP On CentOS 7.x. Steven Cortright. Amazon. Kindle Edition (5 Jun 2017). ASIN: B071G2KLSW. Copyright © 2017.

Design and Architecture of SNMP Monitoring System. Mr. Muhammad Nauman Khan and Dr. Tauseef Jamal. CreateSpace Independent Publishing Platform. paperback 1st edition (30 Oct. 2015). ISBN-13: 978-1523279913.

A Practical Guide to SNMPv3 and Network Management. Mr. David Zeltserman. Prentice Hall. hardback 1st edition (4 May 1999). ISBN-13: 978-0130214539.

Glossary

AES	Advanced Encryption Standard
AES-128	A version of AES with key lenght 128 bits. This is simply referred to as AES in some SNMPv3 implementations.
AES-192	A version of AES with key lenght 192 bits.

AES-256	A version of AES with key lenght 256 bits.
AuthPriv	Indicates authentication with privacy.
authNoPriv	Indicates authentication without privacy.
DES	Data Encryption Standard
HMAC	Hash-based message authentication code (described in RFC 2104).
HMAC-MD5-96	MD5 hash function used in HMAC mode and the output is truncated to 96 bits.
HMAC-SHA-96	SHA hash function used in HMAC mode and the output is truncated to 96 bits.
MD5	The MD5 message-digest algorithm is a hash function producing a 128-bit hash value. MD5 was published in 1992 and is no longer considered secure.
MIB	Management Information Base
MRTG	Multi Router Traffic Grapher software originally developed by Tobias Oetiker and Dave Rand.
NoAuthNoPriv	Indicates no authentication and no privacy (should only be used for testing).
OID	Object ID
SHA-1	Secure Hash Algorithm 1 is a cryptographic hash function which produces a 160-bit hash value. This is simply referred to as SHA in some SNMPv3 implementations.
SHA-2	SHA-2 referrs to two hash functions known as SHA-256 and SHA-512. These are successors to SHA-1. Some SNMPv3 implementations support the use of SHA-256 and SHA-512.
SNMP	Simple Network Management Protocol
SNMPUSM	An application that can be used to do maintenance on the users known to an SNMP agent, by manipulating the User-based Security Module (USM) table.
TLS	Transport Layer Security
TSM	Transport Security Model
USM	User-based Security Model
VCAM	View-based Access Control Model