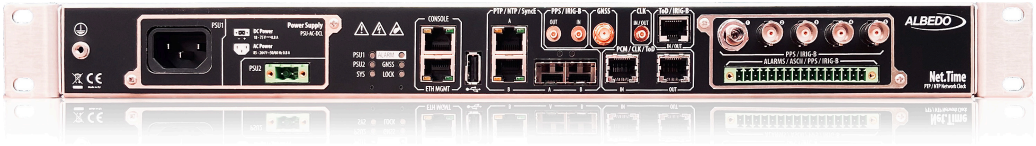

Net.Time: SNMP Management

The Simple Network Management Protocol (SNMP) constitutes a powerful yet simple solution to remote control and automation. Entities controlled through the SNMP interface become SNMP agents responding to queries from remote managers. It is possible for managers to communicate with several agents through a single application to enable collaborative operation across them. SNMP queries can be scripted to implement complex interactions with the agents.

Any user with an Ethernet / IP connection to the agent could use the SNMP interface to generate queries. Agents may change its configuration or operation depending on the received queries. SNMP agents can also reply to controllers with information about the current configuration.

Without an agreement between SNMP managers and agents about the information to be exchanged and the format of this information it would not be possible any communication. Definition and detailed specification of this information is done by one or various *Management Information Bases* (MIBs). MIB modules are text files containing descriptions of the management information available in the agent. ALBEDO Telecom publishes the MIBs required for interaction with Net.Time and documentation about how to use them. Some MIBs, such as the PTP management MIB (ATSL-PTP-MIB), are ALBEDO telecom proprietary. Some others (IF-MIB,...) are defined and maintained by the IETF, IEC or other authority.

1.1.SNMP Management Model

SNMP operates in the Application Layer of the IP protocol stack. SNMP agents receive requests on UDP port 161 and they generate unsolicited notifications to UDP port 162. SNMP messages consist of a header and a payload with a format that depends on the particular SNMP version in use.The payload encodes different messages with different purposes. Five of them (*GetRequest*, *SetRequest*, *GetNextRequest*, *Response* and *Trap*) were defined in the first version of the protocol and two more (*GetBulkRequest* and *InformRequest*) were added in version 2.

SNMP works on the basis that network management systems send out a request and the managed devices reply with a response. This mechanism is implemented with the message types mentioned in the previous paragraph. For example, GET commands (*GetRequest*, *GetNextRequest*, *GetBulkRequest*) retrieve information from the agent, and SET commands (*SetRequest*) can be used to modify the current configuration. Agents can also generate information autonomously (without any query from the manager) through the so called *traps* and *informs*.

The traditional client-server model is useful to configure or retrieve information from managers but it is of limited use if there is the need to report events that could potentially occur at any moment. Asynchronous event reporting is achieved in SNMP with the help of *traps* and *informs*. The main differences between them is that traps are unacknowledged but informs require an acknowledgment message to be successful. In trap reports, the authoritative entity is the agent but informs work in the opposite way. In practical terms, the difference means that when a trap is generated, it is signed and encrypted with the keys from a local user and the target must be aware of the key or community name from the local system but in an inform, authentication and encryption is done with the key / community from a foreign user.

There are currently two popular versions of SNMP, the main difference between them is the security model. Version 2c offers low security level through “communities”. An special name is given to two dif-

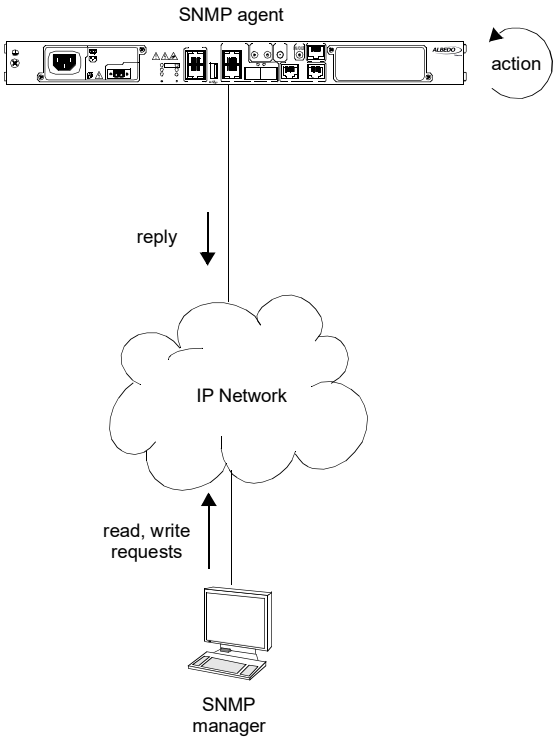

Figure 1 *Client-server SNMP communications model. The manager generates queries. These queries either change the internal configuration of the agent or report status information to the manager*

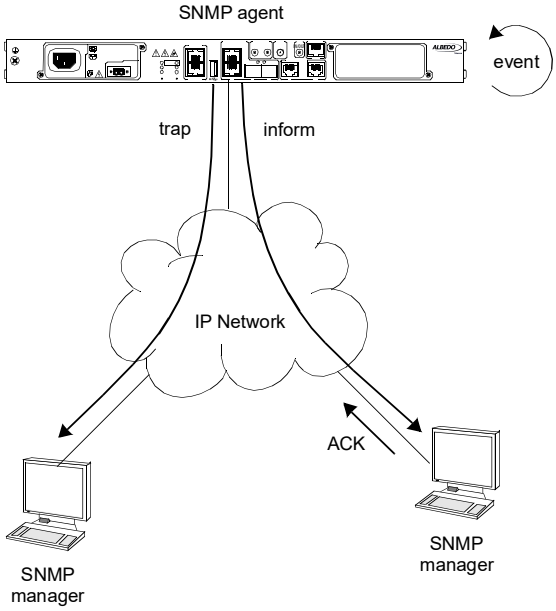

Figure 2 *Traps and informs. They can be used to convey the same kind of information but, unlike traps, informs require acknowledgement from the far end*

ferent communities. One of them enables reading values from the agent (read-only community), and the second allows both reading and writing values in the agent (read-and-write community). SMMP managers not aware of the community names are unable to exchange data with the agent. On the other hand, SNMP version 3 is based on the more sophisticated *User Security Model* (USM) that provides authentication, authorization and privacy to all message transactions. Net.Time supports both protocol versions but network administrators must decide which one to use because it is not allowed to deploy both at the same time.

2. ENABLING SNMP

SNMP is like all other management protocols available in Net.Time such as HTTPS or SSH and it is configured similarly. It is important that all configuration related with SNMP is done from an account with administrator rights because regular users (controllers, viewers) are not allowed to modify the management configuration.

In this note we consider that the Net.Time management interface is already configured and ready to use:

- The device host name is set to “nettime”. The domain name “nettime” is also assigned to the clock.
- SSH management is enabled.

These are the steps to follow to enable SNMP in the unit using the CLI:

1. Log in the system with SSH using administrator credentials

```
ssh admin@nettime
admin@nettime's password:
```

2. Enable SNMP management with the following CLI command:

```
admin@nettime# set netmanager protocol
snmp enable
```

3. SNMP QUERIES, VERSION 2C

SNMP version 2c is a low security implementation of SNMP. It is useful in environments where a simple access to managed entities is important and security is not a critical requirement. Security in SNMP version2c is implemented through "communities". SNMP communities are alphanumeric strings that grant different access levels to the agent management information. For example, the *public* community can be used to grant read access to standard users and the *private* community could be used to grant write access only to privileged users. The community mechanism is simple and easy to use but it offers only limited authorization services and no authenticated or private communications.

Assuming that the SNMP protocol is already enabled in the Net.Time unit, the configuration procedure for version 2c is as follows:

1. Log in the system with SSH using administrator credentials

```
ssh admin@nettime
admin@nettime's password:
```

2. Set the SNMP version

```
admin@nettime# set snmp version v2c
```

3. Configure the read-only community name

```
admin@nettime# set snmp ro-community public
```

4. Configure the read-and-write community name

```
admin@nettime# set snmp rw-community public
```

5. Optionally, check that the configuration parameters are correctly entered.

```
admin@nettime# show snmp

SNMP protocol properties
-------------------------------------------------
Engine ID:                  800099F4030021F3072935
Version:                                        2c
Read Community:                             public
Write Community:                            private
```

Starting from this point, remote users are allowed to generate queries to the agent using SNMP version 2. Examples are generated from a manager running NET-SNMP. This is a software available for most Linux and Unix systems. It is also compatible with Microsoft Windows. The package includes several command line tools, libraries, daemons and other components which can be used for various purposes related with agent management. It is of special interest the availability of modules for *python* and *perl* languages because they enable users to write scripts involving SNMP queries and response analysis.

The NET-SNMP suite is freely available from the Internet. This section assumes that NET-SNMP has been previously installed and configured in the machine. The MIBs required for Net.Time management are also assumed to be available to NET.SNMP. Again, it is remarked that complete documentation about how to run NET-SNMP is available in the Internet.

The following command, for example, retrieves an alphanumeric description from one of the interfaces listed in the *ifTable*, which is one of the tables defined in the IF-MIB:

```
[mgr@manager]$ snmpget -v 2c -c public nettime IF-MIB::ifDescr.8
IF-MIB::ifDescr.8 = STRING: GNSS
```

In this command, I*F-MIB::ifDescr.8* works as an Object IDentifier (OID). It identifies the resource to be read or modified by the SNMP query, *nettime* is the host name assigned to the agent; the query destination. The answer to the *snmpget* command is an STRING, which is the data type, of value *GNSS*. The meaning is that the I*F-MIB::ifDescr.8* OID, identifies the Net.Time GNSS port.

The command includes several parameters that are required to generate a correct query. The *-v* parameter sets the SNMP version (*2c*) and *-c* does the same with the community (*public*). For reading operations, this community must be either the ro-community or the rw-community. For writing in the agent it is mandatory to use the rw-community. This is where all the security implemented by the protocol relies. Supposedly, users unaware of community names are not able to access to the resources from the agent. The only authentication available comes by the fact that only "authentic" users are supposed to be aware the correct community names but there is no way to prove that an specific query is generated from an specific user. There is also no privacy and the community names themselves are transmitted unencrypted. Anybody intercepting the traffic could easily retrieve the community and use this information to spoof the legitimate user identity.

There are NET-SNMP commands to write values in the agent but this is possible for OIDs supporting write operations. Writing to an specific resource must be also allowed by the agent. If any of there conditions is not met, the result will be a query failure and the generation of an error message. Other interesting command allows the manager to display the value of several OIDs with a single message. The following example shows the description for all the Net.Time ports:

```
[mgr@manager]$ snmpwalk -v 2c -c public nettime IF-MIB::ifDescr
IF-MIB::ifDescr.1 = STRING: Ethernet Management Port
IF-MIB::ifDescr.2 = STRING: PTP/NTP/SyncE, A
IF-MIB::ifDescr.3 = STRING: PTP/NTP/SyncE, B
IF-MIB::ifDescr.4 = STRING: PCM/CLK/ToD IN
IF-MIB::ifDescr.5 = STRING: PCM/CLK/ToD OUT
IF-MIB::ifDescr.6 = STRING: PPS/IRIG-B OUT
IF-MIB::ifDescr.7 = STRING: PPS/IRIG-B IN
IF-MIB::ifDescr.8 = STRING: GNSS
IF-MIB::ifDescr.9 = STRING: CLK IN/OUT
IF-MIB::ifDescr.10 = STRING: ToD/IRIG-B IN/OUT
IF-MIB::ifDescr.11 = STRING: PTP/NTP/SyncE, A/B
```

4. SNMP QUERIES, VERSION 3

SNMP version 3 is likely to be more useful in realistic deployments due to the more advanced security model it implements based on users with custom profiles. This is known as the *User Security Model* (USM). To deploy SNMP version 3 in Net.Time, at least one USM user must be defined in the system. USM users replace the community names from version 2c.

To enable SNMP version 3 and configure a USM user, follow these steps in your Net.Time unit.

1. Log in the system with SSH using administrator credentials

```
ssh admin@nettime
admin@nettime's password:
```

2. Set the SNMP version

```
admin@nettime# set snmp version v3
```

3. Define a new USM user (*usmtest*) in the system

```
admin@nettime# set snmp user add name usmtest local access rw auth-proto sha priv-proto aes
Enter your new authentication password:
Write your authentication password again:
Enter your new privacy password:
Write your privacy password again:
```

The previous command defines a user in the local system with authentication based on the *Secure Hash Algorithm* (SHA) and *Advanced Encryption Standard* (AES) symmetric key privacy. The command requests authentication and privacy passwords whenever required.

4. Optionally, check the configuration of the newly generated USM user

```
admin@nettime# show snmp users

Name              Address               Access    Auth     Priv
------------------------------------------------------------------
usmtest           local                  rw        sha      aes
```

From now on, operation is pretty much the same that with version 2c but the NET-SNMP command syntax changes slightly to include information about USM users rather than community names. A valid query to get data from the agent is:

```
[mgr@manager mibs]$ snmpget -v 3 -u usmtest -a sha -A PW -x aes -X PW -l AuthPriv nettime IF-MIB::ifDescr.8
IF-MIB::ifDescr.8 = STRING: GNSS
```

The *snmpget* command now includes options *-a* and *-x* to define the authentication and encryption algorithms together with their passwords (*-A*, *-X*). There is also an option to enter the USM user name (*-u*) and the security level (*-l*). Despite the similarity of the version 2c and 3 commands, they operate in a completely different way. If we could sniff the SNMP traffic flow from the transmission medium we would be unable to understand the message without the encryption password. The agent now rejects unauthenticated queries or queries with wrong authentication credentials.

All other NET-SNMP commands offer similar functionality. For example, the *snmpwalk* query required to get several OIDs in a single command is:

```
[mgr@manager mibs]$ snmpwalk -v 3 -u usmtest -a sha -A PW -x aes -X PW -l AuthPriv nettime IF-MIB::ifDescr
IF-MIB::ifDescr.1 = STRING: Ethernet Management Port
IF-MIB::ifDescr.2 = STRING: PTP/NTP/SyncE, A
IF-MIB::ifDescr.3 = STRING: PTP/NTP/SyncE, B
IF-MIB::ifDescr.4 = STRING: PCM/CLK/ToD IN
IF-MIB::ifDescr.5 = STRING: PCM/CLK/ToD OUT
IF-MIB::ifDescr.6 = STRING: PPS/IRIG-B OUT
IF-MIB::ifDescr.7 = STRING: PPS/IRIG-B IN
IF-MIB::ifDescr.8 = STRING: GNSS
IF-MIB::ifDescr.9 = STRING: CLK IN/OUT
IF-MIB::ifDescr.10 = STRING: ToD/IRIG-B IN/OUT
IF-MIB::ifDescr.11 = STRING: PTP/NTP/SyncE, A/B
```

5. TRAPS

Traps and informs are SNMP messages generated autonomously from the agent to certain target destinations when certain events occur. The main differences between them is that traps are unacknowledged but informs require an acknowledgment message to be successful. In trap reports, the authoritative entity is the agent but informs work in the opposite way. In practical terms, the difference means that when a trap is gen-

erated, it is signed and encrypted with the keys from a local user and the target must be aware of the keys defined in this local system but in an inform, authentication and encryption is done with the keys from a foreign user.

Net.Time can be configured to translate messages logged to the system into SNMP format and, if required, to generate traps and informs based on these messages to one or several destinations. Both SNMP version 2c and 3 are suitable for this purpose but this document focus in the most interesting alternative which is trap generation over SNMP version 3. The steps to follow to configure a version 3 trap target are described below:

1. Log in the system with SSH using administrator credentials

```
ssh admin@nettime
admin@nettime's password:
```

2. Set the SNMP version

```
admin@nettime# set snmp version v3
```

3. If necessary, define a new USM user (*usmtest*) in the system

```
admin@nettime# set snmp user add name usmtest local access rw auth-proto sha priv-proto aes
Enter your new authentication password:
Write your authentication password again:
Enter your new privacy password:
Write your privacy password again:
```

4. Optionally, check the configuration of the newly generated USM user

```
admin@nettime# show snmp users

Name              Address               Access   Auth    Priv
------------------------------------------------------------------
usmtest           local                  rw       sha     aes
```

5. Enable SNMP translation of log messages

```
admin@nettime# set syslog snmp enable
```

From this point, log messages are copied to the *SYSLOG-MSG-MIB::syslogMsgTable* SNMP table and they can be retrieved remotely by managers with the corresponding version 3 query message.

6. To generate traps when log messages are added to the table it is necessary to modify the *SYSLOG-MSG-MIB ::syslogMsgEnableNotifications* OID. This is something it can be done with a SET query to the agent. Here, it is assumed that the manager is running NET-SNMP. The command is then:

```
[mgr@manager]# snmpset -v 3 -u usmtest -a sha -A PW -x aes -X PW -l AuthPriv nettime SYSLOG-MSG-MIB::
syslogMsgEnableNotifications i true
```

7. We still need to define the trap target, which is the IP host where the notifications are to be sent.

```
admin@nettime# set snmp target add trap address 10.10.10.1 version v3 security-name usmtest security-level
priv
```

It can be seen that the trap target is bound to a local USM user (*usmtest*, in this example) whose credentials are used for encryption and / or authentication. It this user is not defined in the system, the

message will not be generated. The security level, *priv* (privacy), in the example must be of the same level or lower than the level defined for the USM user. For example, it is not possible to configure the *priv* level in the trap target for a user which does not have a well defined privacy algorithm and password.

8. Optionally, check the configuration of the newly generated target

```
admin@nettime# show snmp targets
 ID               Address      Ver  SecMod  SecLev               SecName      PDU
-------------------------------------------------------------------------------
  0        10.10.10.1:162       3     usm    priv               usmtest     trap
```

It is important to remark that when translation of log messages into SNMP is enabled, it is not yet possible to generate traps. This is not only because there are no trap targets where to send notifications but also because the system must be instructed to generate Syslog traps by setting *SYSLOG- MSG-MIB ::syslogMsgEnableNotifications* to *true*. Before configuring the target and the MIB to generate notifications, log messages are remotely available with a regular version 3 *snmpwalk* query:

```
[mgr@manager]$ snmpwalk -v 3 -u usmtest -a sha -A PW -x aes -X PW -l AuthPriv nettime SYSLOG-MSG-MIB::
syslogMsgTable
SYSLOG-MSG-MIB::syslogMsgIndex.1 = Gauge32: 1
SYSLOG-MSG-MIB::syslogMsgFacility.1 = INTEGER: kern(0)
SYSLOG-MSG-MIB::syslogMsgSeverity.1 = INTEGER: warning(4)
SYSLOG-MSG-MIB::syslogMsgVersion.1 = Gauge32: 1
SYSLOG-MSG-MIB::syslogMsgTimeStamp.1 = STRING: 2022-12-29,8:56:34.0,+0:0
SYSLOG-MSG-MIB::syslogMsgHostName.1 = STRING: nettime
SYSLOG-MSG-MIB::syslogMsgAppName.1 = STRING:
SYSLOG-MSG-MIB::syslogMsgProcID.1 = STRING:
SYSLOG-MSG-MIB::syslogMsgMsgID.1 = STRING:
SYSLOG-MSG-MIB::syslogMsgSDParams.1 = Gauge32: 0
SYSLOG-MSG-MIB::syslogMsgMsg.1 = STRING: "(751)-ERDY PRP(J): Link ready"
```

SNMP traps require an specialized application running in the target host to decode and report notifications. Most MIB browsers offer this option and there is also this possibility in the NET-SNMP software suite. One alternative, which is good for testing purposes is to use the Wireshark sniffer to capture trap notifications. Wireshark includes the resources required authenticate and decrypt SNMP queries. For reference purposes, it follows the result of decoding a trap notification generated by Net.Time.

```
Simple Network Management Protocol
    msgVersion: snmpv3 (3)
    msgGlobalData
        msgID: 16777216
        msgMaxSize: 1458
        msgFlags: 03
        msgSecurityModel: USM (3)
    msgAuthoritativeEngineID: 800099f403XXXXXX072935
        1... .... = Engine ID Conformance: RFC3411 (SNMPv3)
        Engine Enterprise ID: ALBEDO Telecom SL (39412)
        Engine ID Format: MAC address (3)
        Engine ID Data: MAC address: XXXX_07:29:35 (XX:XX:XX:07:29:35)
    msgAuthoritativeEngineBoots: 1
    msgAuthoritativeEngineTime: 172216
    msgUserName: usmtest
    msgAuthenticationParameters: 23ce6d7cd40d24478bcf6f4f
        [Authentication: OK]
    msgPrivacyParameters: 000000011bd62ca8
    msgData: encryptedPDU (1)
        encryptedPDU: ad4ee0db039279a9633b53d3945c066b5efa1126cd395f112dacbd940b7f6a…
            Decrypted ScopedPDU: 30820143040b800099f4030021f30729350400a782013002010002010001…
                contextEngineID: 800099f403XXXXXX072935
                    1... .... = Engine ID Conformance: RFC3411 (SNMPv3)
```

```
    Engine Enterprise ID: ALBEDO Telecom SL (39412)
    Engine ID Format: MAC address (3)
    Engine ID Data: MAC address: XXXX_07:29:35 (XX:XX:XX:07:29:35)
contextName:
data: snmpV2-trap (7)
    snmpV2-trap
        request-id: 0
        error-status: noError (0)
        error-index: 0
        variable-bindings: 12 items
            1.3.6.1.2.1.1.3.0: 17221604
            1.3.6.1.6.3.1.1.4.1.0: 1.3.6.1.2.1.192.1.0.1 (iso.3.6.1.2.1.192.1.0.1)
            1.3.6.1.2.1.192.1.2.1.2: 0
            1.3.6.1.2.1.192.1.2.1.3: 4
            1.3.6.1.2.1.192.1.2.1.4: 1
            1.3.6.1.2.1.192.1.2.1.5: 07e60c1d0a210a0000002b0000
            1.3.6.1.2.1.192.1.2.1.6: "nettime"
            1.3.6.1.2.1.192.1.2.1.7: <MISSING>
            1.3.6.1.2.1.192.1.2.1.8: <MISSING>
            1.3.6.1.2.1.192.1.2.1.9: <MISSING>
            1.3.6.1.2.1.192.1.2.1.10: 0
            1.3.6.1.2.1.192.1.2.1.11: "(751)-ERDY PRP(J): Link ready"
```

This message was encrypted by the agent (*encryptedPDU* field). The information required to allow the receiver to authenticate the message was also generated (*msgAuthenticationParameters* field). The SNMP message version (*snmpv3* for SNMP version 3) and security model (USM) are also displayed in the message. The message payload includes the variable bindings reported by the trap. Variables correspond with OIDs defined in the SYSLOG-SNMP-MIB. The variables are bound to the reported values in a format which is also defined in the MIB file.