nShield Security World

nShield SNMP Monitor v13.9.0 Install and User guide

28 October 2025

Table of Contents

1. Introduction

Entrust provides an SNMP that can monitor network interfaces. You can add the nShield SNMP monitoring agent to your SNMP manager application. It is based on the open-source Net-SNMP project, version 5.7.3.

2. Install and activate the nShield SNMP monitoring agent

On Linux, the SNMP agent is installed with the installation of the Security World Software and starts automatically.

On Windows, the SNMP agent can be installed and activated separately. After installing the SNMP components, an activation command can be issued.

2.1. Default installation settings

When installing Security World Software, you may be prompted to select Security World Software components from a list. If you select `all` components, then the SNMP agent is installed as part of a full Security World Software installation. The default installation directory for the nShield Management Information Base (MIB) and the SNMP configuration files (`snmp.conf` and `snmpd.conf`) is `/opt/nfast/etc/snmp/` (**Linux**) or `%NFAST_HOME%\etc\snmp\` (**Windows**).

2.2. Do you already have an SNMP agent running?

If you already have another SNMP agent running, you must configure the ports used by the agents in order to avoid conflicts before enabling the SNMP agent. A port is assigned by editing the `agentaddress` entry in the `snmpd.conf` file or by editing the `defaultPort` entry in `snmp.conf` file. If both files have been edited, the `agentaddress` entry is `snmpd.conf` file takes priority for snmpd, and the `defaultPort` entry in `snmp.conf` is ignored.

If no existing SNMP agent is found, the SNMP agent runs on the default port 161. If an existing SNMP agent is detected, and no SNMP agent configuration files are found (implying a fresh installation), the installer automatically configures the SNMP agent to use the first unused port above 161 by creating a new `snmpd.conf` configuration file with the appropriate directive. It then displays a message indicating the number of the port that is has selected.

If an existing SNMP agent is found and an existing SNMP agent installation exists, the installer checks the existing configuration files for an appropriate directive and warns you if one does not exist. If you need to edit these configuration files yourself, a port is assigned by editing the `agentaddress` entry in `snmpd.conf` file or editing the `defaultPort` entry in `snmp.conf` file. If both files have been edited, the `agentaddress` entry in `snmpd.conf` file takes priority for snmpd, and the `defaultPort` entry in `snmp.conf` is ignored.

2.3. Start the SNMP agent

Linux

The SNMP agent is started automatically however it can be stopped and started manually. To stop, start, or restart (stop and immediately start again) the SNMP daemon:

```
/opt/nfast/scripts/init.d/ncsnmpd stop|start|restart
```

See Basic configuration for more information on additional parameters accepted by snmpd.

Windows

To register the SNMP agent as a Windows service, enter the following command with administrative privileges:

```
%NFAST_HOME\bin\snmpd -register [params]
```

See Basic configuration for more information on additional parameters accepted by snmpd.

This installs the agent as a Windows Service but does not start it automatically.

By default, the SNMP agent logs start-up and shut-down to the Event Viewer. More detailed logging can be configured by providing additional parameters when running the SNMP agent either from the command line or when registering as a service.

To unregister the SNMP agent as a Windows service, enter the following command:

```
snmpd -unregister
```

The SNMP agent can be started and stopped from the services control panel or from the command prompt using:

```
net start "nCipher SNMP Agent"
net stop "nCipher SNMP Agent"
```

3. Basic configuration

3.1. Protect the SNMP installation

The SNMP agent allows other computers on the network to connect to it and make requests for information. The SNMP agent is based on the NET-SNMP code base, which has been tested but not fully reviewed by Entrust. We strongly recommend that you deploy the SNMP agent only on a private network or a network protected from the global Internet by appropriate network protection systems, for example a firewall or a network Intrusion Detection/Prevention System.

The default nShield SNMP installation allows read-only access to the Management Information Base (MIB). There is no default write access to any part of the MIB.

Every effort has been taken to ensure the confidentiality of cryptographic keys even when the SNMP agent is enabled. In particular, the nShield module is designed to prevent the theft of keys even if the security of the host system is compromised, provided that the Administrator Cards are used only with trusted hosts. Care must be used when changing the configuration of the SNMP agent.

We strongly advise that you use the SNMP User-based Security Model (USM) with Authentication and Privacy protocols selected, to ensure only authorised users can obtain information from the SNMP agent and the confidentiality and data integrity of the transferred information is protected.

Care has also been taken to ensure that malicious attackers are unable to inundate your module with requests by flooding your SNMP agent. Command results from administration or statistics commands are cached, and thus the maximum rate at which the SNMP agent sends commands to the module is throttled. For more information on setting the cache time-outs, see The SNMP configuration file: snmp.conf.

3.2. Configure the SNMP agent

The Security World Software package uses various configuration files to configure its applications. This section describes the overall nature of the configuration files for the SNMP agent.

If you are installing the SNMP agent to a host that has an existing SNMP agent installation, you may need to edit the SNMP configuration files (`snmpd.conf` and `snmp.conf`) associated with the SNMP agent to change the port on which the agent listens for SNMP requests. For

more information, see Do you already have an SNMP agent running?.

Make sure you protect access to the configuration files, since these contain information that defines the security parameters of the SNMP system. The default location for the nShield SNMP configuration files is `/opt/nfast/etc/snmp/` (**Linux**) or `%NFAST_HOME%\etc\snmp\` (**Windows**).

3.3. Create the configuration files (Windows)

On Windows, the `snmp.conf` and `snmpd.conf` files are not created automatically by the installation. Instead, example files (`example.snmp.conf` and `example.snmpd.conf`) are created in that location, which you can copy, rename (to `snmp.conf` and `snmpd.conf`), and edit with your desired configuration settings.

The sample `snmpd.conf` file includes `agentuser` and `agentgroup` directives, however these are inoperative in Windows.

You can override the default search path by setting the environment variable `SNMPCONFPATH` to a colon-separated (":") list of directories for which to search.

3.3.1. Re-read SNMP configuration files

The SNMP agent reads its configuration files on startup, and any changes made after this point will have no effect. If new directives are added and need to be applied, the SNMP agent can be forced to re-read its configuration files with:

- An snmp `set` of integer(1) to `enterprises.nCipher.reloadConfig.0(.1.3.6.1.4.1.7682.999.0)`
- kill `-HUP` signal sent to the `snmpd` agent process
- stop then restart the SNMP agent.

3.3.2. The SNMP configuration file: snmp.conf

The `snmp.conf` configuration file contains directives that apply to all SNMP applications. These directives can be configured to apply to specific applications. The `snmp.conf` configuration file is not required for the agent to operate and report MIB entries.

3.3.3. The SNMP agent configuration file: snmpd.conf

The `snmpd.conf` configuration file defines how the SNMP agent operates. It is required only if an agent is running.

The `snmpd.conf` file can contain any of the directives available for use in the `snmp.conf` file and may also contain the following Security World Software-specific directives:

Directive	Description
`statstimeout`	This directive specifies the length of time for which statistics commands are cached. The default is 5 seconds.
`admintimeout`	This directive specifies the length of time for which administrative commands are cached. The default is 60 seconds.
`keytable`	This directive sets the initial state of the key table to `none`, `all`, or `query`. See `listKeys` in Administration sub-tree overview.
`enable_trap_zero_suffix`	This directive appends the '.0' suffix to object identifiers (OIDs) for backward compatibility. The default is `0` (disabled): the directive can be set to `1` to restore the suffix. Valid values are 0 and 1.
`memoryUsageOkThreshold`	This directive specifies the threshold (as a percentage) below which HSM memory usage is considered to be ok. The default is 0. See Memory usage monitoring for more details.
`memoryUsageHighThreshold`	This directive specifies the threshold (as a percentage) at which HSM memory usage is considered to be too high. The default is 0. See Memory usage monitoring for more details.

There may be a tolerance gap between the `memoryUsageOkThreshold` and the `memoryUsageHighThreshold` values.

The timeouts should be set to values that achieve a balance between receiving up to date information whilst preventing excessive load.

3.4. The SNMP agent persistent configuration file

On running the SNMP agent for the first time, the `persist` directory will be created. This contains configuration files that are maintained by the SNMP agent. This directory will be created in `/opt/nfast/etc/snmp/persist` (**Linux**) or `%NFAST_HOME%\etc\snmp\persist` (**Windows**).

Modifications should only be made to the persist folder's `snmp.conf` file in order to create users. The files within this directory should otherwise only be managed by the SNMP agent itself.

User creation can be performed with the `createUser` directive. See USM users. On initializa-

tion of the agent the information is read from the file and the lines are removed (eliminating the storage of the master password for that user) and replaced with the key that is derived from it. This key is a localised key, so that unlike the password, if it is stolen it can not be used to access other agents.

Do not modify the persistent `snmpd.conf` file while the agent is running. The file is only read on initialization of the agent and it is overwritten when the SNMP agent terminates. Any changes made to this file while the SNMP agent directives is running will be lost. The SNMP agent should be stopped prior to adding `createUser` directories to the configu ration file.

3.5. Agent Behaviour

There are a small number of directives that control the behaviour of the SNMP Agent when considering it as a daemon providing a network service.

3.6. agentaddress directive

The listening address(es) that the SNMP Agent will use are defined by the `agentaddress` directive. It takes a comma separated list of address specifiers where an address specifier consists of one or more of:

- a transport specifier udp: or tcp
- a hostname or IPv4 address
- a port number (for example, :161 or :1161).

The default behaviour is to listen on UDP port 161 on all IPv4 interfaces (the equivalent to udp:161).

```
agentaddress localhost : 161,tcp:1161
```

`agentaddress` will listen on UDP port 161, but only on the loopback interface (the port specifi cation ":161" is not strictly necessary as this is the default port). It will also listen on TCP port 1161 on all IPv4 interfaces.

3.7. agentgroup and agentuser directives (Linux)

The user and group that the SNMP Agent changes to after opening the listening port(s) are defined using the `agentgroup` and `agentuser` directives. The following must be used:

```
agentgroup ncsnmpd
agentuser ncsnmpd
```

3.8. System information (Linux)

Most of the scalar objects in the .iso.org.dod.internet.mgmt.mib-2.system sub-tree can be configured.

```
sysLocation STRING
sysContact STRING
sysName STRING
```

The three directives above set the system location, contact or name for the SNMP Agent respectively. Ordinarily these objects are writable via a suitably authorised SNMP SET request, however, specifying one of these directives in the configuration file makes the cor responding object read-only.

```
sysServices INTEGER
```

Sets the value of the sysService.0 object. RFC1213 defines how the integer value is calcu-lated.

```
sysDescr STRING
sysObjectID OID
```

The two directives above set the system description and object ID for the agent. These objects are not SNMP-writable, but these directives can be used by a network administra-tor to configure suitable values for them.

4. USM users

The SNMPv3 protocol supports a User based Security Model as defined in RFC-3414. USM provides authentication and privacy (encryption) functions and operates at the message level allowing for the following security level to be used with SNMPv3:

- Communication without authentication and privacy (`noauth`)
- Communication with authentication and without privacy (`auth`)
- Communication with authentication and privacy (`priv`).

Within this document the three possible security levels are referred to as `noauth`, `auth` and `priv`. However, other forms are sometimes used within the NET-SNMP and the equivalents are:

Security level	Equivalents
`noauth`	`noauthnopriv`
`auth`	`authnopriv`
`priv`	`authpriv`

Users can be added to the SNMP configuration with the `createUser` directive, defining the security mechanisms to be used. Both the SHA and AES passhrases have to be defined.

```
createUser [-e ENGINEID] {username [SHA authpassphrase] [AES privpassphrase]}
```

It would not normally be necessary to specify the engine ID, but if it is specified, `ENGINEID` is defined as a hexadecimal string of octets starting with the 0x prefix. The encoding of the engine ID is defined in the description of `SnmpEngineID` from RFC3411.

The following recommendations should be followed when defining the security parameters for SNMPv3:

- Select a 'Security Level' of Priv, (`authpriv`) or auth (`authNoPriv`).
  - `Priv` is the preferred 'Security Level', since this will provide both data source authentication and confidentially protection for the SNMP messages.
  - `auth` is the minimum 'Security Level' that should be selected, since this will ensure that SNMP data sent/received has not been tampered with and has been sent from an authorised entity.
- Define separate `authpassphrase` and `privpassphrase`.
  - It is good security practice to have key separation.
- Use randomly generated passphrases which contain upper and lower case characters,

numbers and symbols (for example, ASCII characters 0x20 - 0x7E).
    - This should give an entropy per character of 6.57bits,
- Use either 15 char for 96 bits of security strength keys and 20 char for 128 bits security strength keys.
    - The minimum length of both `Auth` and `Priv` passphrases is eight characters.
    - If a random passphrase is not used, consult NIST SP800-63-2 - Appendix A to determine the security strength of the password and the resultant keys. See https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf.

MD5 and DES are not supported or enabled in the nShield distribution of SNMP. Only SHA may be used for authentication, and only AES may be used for privacy (encryption).

It is strongly recommended that `createUser` directives be added to the `persist/snmpd.conf` file, so that the passphrases are not available after the SNMP agent is installed. The user can then be referenced in access control directives(s) after which it can be used.

You can use `snmpwalk` to test that the user configuration is correct:

```
/opt/nfast/bin/snmpwalk -v3 -l authPriv -u {username} -x AES -X {privpassphrase} -a SHA -A {authpassphrase}
localhost .1.3.6.1.4.1.7682
```

5. Traditional access control

Most simple access control requirements can be specified using the directives `rouser` /`rwuser` (for SNMPv3) or `rocommunity`/`rwcommunity` (for SNMPv1 or SNMPv2c).

```
rouser [-s usm] USERNAME [noauth | auth | priv [OID | -V VIEW [CONTEXT]]
rwuser [-s usm] USERNAME [noath | auth | priv [OID | -V VIEW [CONTEXT]]
```

These directives specify that an SNMPv3 user (USERNAME) will be allowed read-only or read-write access respectively. The default (unspecified) security level is `auth`, which is the recommended minimum security level (see above). It is not recommended to use the usm security level `noauth`, where all SNMP messages are unauthenticated and any tampering of the message cannot be detected. Using `noauth` will reduce the security of the SNMP messages to the level of SNMPv1 or SNMPv2c.

OID restricts access for that user to the subtree rooted at the given OID.

`VIEW` restricts access for that user to the specified View-based Access Control Model (VACM) view name. An optional context can also be specified, or `context` to denote a context prefix. If no context field is specified (or the token * is used), the directive will match all possible contexts. (Contexts are a mechanism within SNMPv3 whereby an agent can support parallel versions of the same MIB objects, referring to different underlying data sets.)

A security model can be specified with `-s SECMODEL` however the default security model `usm` is the only security model which is supported in the nShield distribution of SNMP.

Example:

- Read-only user with access to the full OID tree requiring authentication as a minimum:

  ```
  rouser userl
  ```

  Or

  ```
  rouser -s usm user1 auth .1
  ```

- Read-only user with access to the nShield MIB allowing unauthenticated requests:

  ```
  rouser user2 noauth .1.3.6.1.4.1.7682
  ```

- Read-write user with access to the full OID tree requiring authentication as a minimum:

  ```
  rwuser user3
  ```

Or

```
rwuser user3 auth .iso
```

- Read-write user with access to the snmpVacmMIB subtree requiring authentication and encryption:

```
rwuser user4 priv snmpVacmMIB
```

Or

```
rwuser user4 priv .1.3.6.1.6.3.16
```

```
rocommunity COMMUNITY [SOURCE [ OID | -V VIEW [CONTEXT]]
rwcommunity COMMUNITY [SOURCE [ OID | -V VIEW [CONTEXT]]
```

Specifies an SNMPv1 or SNMPv2c community that will be allowed read-only (`GET` and `GET-NEXT`) or read-write (`GET`, `GETNEXT` and `SET`) access respectively. By default, this will provide access to the full OID tree for such requests, regardless of where they were sent from. `SOURCE` allows access either from a particular range of source addresses, or globally (`default`). A restricted source can either be a specific hostname or address (for example, `localhost` or 127.0.0.1), or a subnet - represented as IP/MASK (for example, 10.10.10.0/255.255.255.0), or IP/BITS (for example, 10.10.10.0/24).

OID `VIEW` and `CONTEXT` are as defined for `rouser` and `rwuser`.

Example:

- Setting up a read-only community named `public` that can be accessed by any user with the community name:

```
rocommmunity public
```

- Setting up a read/write community named `private` that can only be accessed from the machine on which the agent is running:

```
rocommmunity private localhost
```

In each case, only one directive should be specified for a given SNMPv3 user, or community string. It is not appropriate to specify both `rouser` and `rwuser` directives referring to the same SNMPv3 user (or equivalent community settings). The `rwuser` directive provides all the access of `rouser` (as well as allowing `SET` support). The same applies to `rwcommunity` and

`rocommunity`.

More complex access requirements (such as access to two or more distinct OID subtrees, or different views for `GET` and `SET` requests) should use VACM configuration directives.

6. VACM configuration

The full flexibility of the VACM, for example allowing access to two or more distinct OID sub trees, or different access requirements for reading and writing, is available using four config uration directives - com2sec, group, view and access. The directives essentially define who has access and what they have access to using four directives. The first two directives (com sec2sec and group) define the who, while the last two (view and access) define the what.

```
Com2sec [-Cn CONTEXT] SECNAME SOURCE COMMUNITY
```

Maps an SNMPv1 or SNMPv2c community string to a security name. As it defines the com- munity and maps it to a security name, rocommunity/rwcommunity directives are not required when using the directive.

SECNAME is the security name to be defined.

SOURCE is as defined for the rocommunity/rwcommunity directives above.

COMMUNITY defines the community name to be mapped to the security name. The same com munity string can be specified in several separate directives with different source tokens, and the first source/community combination that matches the incoming request will be selected. Various source/community combinations can also map to the same security name.

CONTEXT if defined (using -Cn), means that the community string will be mapped to a secu- rity name in the named SNMPv3 context. Otherwise the default context ("") will be used.

Example:

Creating three SNMPv1/v2c community names (private, public and ltd), where private and ltd only allow requests from the machine on which the SNMP Agent is running (note lines beginning with a # in snmpd.conf are treated as comments):

```
#         [-Cn CONTEXT] SECNAME     SOURCE    COMMUNITY
com2sec   ""            sec_private localhost private
com2sec                 sec_public  default   public
com2sec                 sec_limited localhost ltd
```

```
group GROUP v1 | v2c | usm SECNAME
```

Maps a security name (in the specified security model) into a named group. Several group directives can specify the same group name, allowing a single access setting to apply to several users and /or community strings. Note that groups must be set up for the two com-

munity-based models separately - a single com2sec directive will typically be accompanied by two group directives.

- GROUP is the group name being defined/added to.
- v1, v2c, or usm defines the security model to which the definition relates.
- SECNAME is the security (USM) user name or security name defined by com2sec to be added to the group.

Example:

Creating three groups (grp_private, grp_public, grp_limited) for three USM users (user1, user2 and user3) and the three communities shown in the com2sec example above:

```
# GROUP v1|v2c|usm SECNAME
group grp_private v1 sec_private
group grp_private v2c sec_private
group grp_private usm user1

group grp_public v1 sec_public
group grp_public v2c sec_public
group grp_public usm user2

group grp_limited v1 sec_limited
group grp_limited v2c sec_limited
group grp_limited usm user3
```

```
view VNAME included | excluded OID [MASK]
```

Defines a named view - a subset of the overall OID tree. This is most commonly a single sub tree, but several view directives can be given with the same view name (VNAME), to build up a more complex collection of OIDs. An optional mask can also be specified, providing a means of indicating which parts of the OID must be matched.

VNAME is the view being modified.

included | excluded allows you to define whether the view includes or excludes the subtree, allowing the definition of a more complex view (for example, by excluding certain sensitive objects from an otherwise accessible subtree).

MASK is an optional list of hex octets (separated by '.' or ':') whose bits indicate which OID sub-identifiers to match against. So for example if we assume we have on OID with 11 sub-identifiers (.1.3.6.1.x.y.z.table.entry.column.1) where the last four relate to a table, an entry, a column and index 1, specifying a MASK value of " FF.A0 " (1111111110100000) maps to this OID as follows:

```
1.3.6.1.x.y.z.table.entry.column.1
1 1 1 1 1 1 1   1     1     0    1
```

This mask means all parts of the OID except the column must match, therefore defining a view to any column of the first row of the table.

By including and excluding various aspects of the full OID tree, it is possible to define fine grained visibility within a view's definition.

Example:

Creating five views where `vw_sysContact` only has access to the `system.sysContact.0` OID, `vw_nCipher` only has access to the MIB, `vw_global` has access to the full OID tree, `vw_nCipher_stats` only has access to `nCipher.nC-series.statistics` and `vw_nCipher_admin` only has access to `nCipher.nC-series.administration`:

```
#     VNAME             included|excluded   OID                                        [MASK]
view  vw_sysContact     excluded            .1
view  vw_sysContact     included            system.sysContact.0                        FF.80

view  vw_nCipher        excluded            .iso
view  vw_nCipher        included            .1.3.6.1.4.1.7682

view  vw_global         included            .1

view  vw_nCipher_stats  excluded            .1
view  vw_nCipher_stats  included            enterprises.nCipher.nC-series.statistics

view  vw_nCipher_admin  excluded            .1
view  vw_nCipher_admin  included            enterprises.nCipher.nC-series.administration
```

```
access GROUP CONTEXT any | v1 | v2c | usm noauth | auth | priv exact | prefix READ WRITE NOTIFY
```

Maps from a group of users/communities (with a particular security model and minimum security level, and specific context) to one of three views, depending on the request being processed.

`GROUP` is a group name defined by the group directive and specifies the group that access is being defined for.

`CONTEXT` specifies the context for the access (the default context is the empty string ""). The context of incoming requests must match against the context either exactly or by prefix, as specified by the choice of `exact` | `prefix` made in this directive.

`any`, `v1`, `v2c`, or `usm` define the security model to which this definition relates.

`noauth` | `auth` | `priv` define the security level to which this definition relates. For `v1` or `v2c` access, this will need to be `noauth` as these protocols do not support authentication.

`exact` | `prefix` specify how `CONTEXT` should be matched against the context of the incoming request, either an exact match to `CONTEXT`, or prefixed by `CONTEXT`.

READ, WRITE and NOTIFY specifies the view to be used for GET*, SET and TRAP/INFORM requests (although the NOTIFY view is not currently used). The keyword none is used if there is to be no access for that type of request.

Example:

Specifying that:

- SNMPv1 requests using the public community only have read access to the enterprises.nCipher.nC-series.statistics subtree,
- SNMPv2c requests using the public community only have read access to the enterprises.nCipher.nC-series.administration.subtree,
- SNMPv3 requests using the user2 USM user, must as a minimum be authenticated, and have read, notify access to the nShield MIB (that is, enterprises nCipher)
- SNMPv3 requests using the user1 USM user, must as a minimum be authenticated and encrypted, and have read, write and notify access to the full OID tree. Note that since requests must be authenticated and encrypted as a minimum, SNMPv1 and v2c requests using the private community cannot be made even though the community is included in grp_private.
- SNMPv1 and SNMPv2 requests using the ltd community and SNMPv3 requests using the user3 USM user, do not require to be authenticated or encrypted, and have read, write access to the system.sysContact.0 OID.

```
#      GROUP       CONTEXT  SECMODEL  LEVEL      PREFIX  READ              WRITE     NOTIFY
access grp_public  ""       v1        noauth     exact   vw_nCipher_stats  none      none
access grp_public  ""       v2c       noauth     exact   vw_nCipher_admin  none      none
access grp_public  ""       usm       auth       exact   vw_nCipher        none      vw_nCipher
access grp_private ""       any       priv       exact   vw_global         vw_global
vw_global
access grp_limited ""       any       noauth     exact   vw_sysContact     vw_sysContact
none
```

7. Trap Configuration

The distribution of SNMP supports SNMPv1, SNMPv2 and SNMPv3 traps. Control over these traps is defined with a number of directives:

7.1. SNMPv1 and SNMPv2 traps

```
trapcommunity COMMUNITY
```

Defines the default community to be used when sending SNMPv1 or SNMPv2 traps. Note that this directive must be used prior to a `trapsink` or `trap2sink` directive that wishes to use this community.

`COMMUNITY` the community name to be used.

Example:

```
trapcommunity traps
```

```
trapsink HOST [COMMUNITY [PORT]]
trap2sink HOST [COMMUNITY [PORT]]
```

Defines a destination for SNMPv1 or SNMPv2 traps generated by the agent.

`HOST` is an address specifier defining the network target that traps will be sent to. It consists of an optional transport specifier (`udp` (default if not specified) or `tcp`), followed by a host-name or IPv4 address, followed by an optional port number. The address components are separated by colons ":". For example, `localhost` or `tcp:192.168.137.2:163`.

`COMMUNITY` if specified will be the community name used for the traps. If it is not specified, the most recently specified `trapcommunity` will be used.

`PORT` allows for port-number to be defined if it is not present as part of the `HOST` specification. If no port is defined, the default port number of 162 will be used.

When a TCP transport specifier is used the SNMP agent establishes the TCP connection with the trap manager at start-up. Therefore the trap manger must be started before the SNMP agent otherwise an error is reported for the line in the `snmpd.conf` file which defines the trap manager.

Likewise when the TCP connection between the SNMP agent and the trap manager is dropped, traps are lost. Therefore it is inadvisable to use TCP instead of UDP for the trans-

port specifier of trap managers.

If TCP is used for the connection between the SNMP agent and the trap manager and the connection is lost, to re-establish the connection the SNMP agent must be restarted, with the trap manager running and able to accept a TCP connection from the SNMP agent.

For issues with the trap manager accepting TCP connections from a SNMP agent refer to trap manager documentation.

Example:

```
trap2sink udp:192.168.137.220:162 traps
```

7.2. SNMPv3 traps

```
trapsess [SNMPCMD_ARGS]  HOST
```

Defines the configuration for a trap. This is the only way to define SNMPv3 traps and it is an alternative method for defining SNMPv1 and SNMPv2 traps.

`SNMPCMD_ARGS` are arguments that would be used for an equivalent `snmptrap` command. So for example to send an SNMPv3 trap as USM user `user1` with authentication and encryption, the value `-v3 -u user1 -l priv` would be used.

`HOST` see host definition for `trap2sink` above. Example:

```
trapsess -v3 -u user1 -l priv udp:192.168.137.220:162
trapsess -v2c -c public 192.168.137.221:162
```

8. Using the SNMP agent with a manager application

The nShield SNMP monitoring agent provides MIB files that can be added to your (third-party) SNMP manager application.

8.1. Manager configuration

The manager application is the interface through which the user is able to perform network management functions. A manager communicates with agents using SNMP primitives (`get`, `set`, `trap`) and is unaware of how data is retrieved from, and sent to, managed devices. This form of encapsulation creates the following:

- The manager is hidden from all platform specific details
- The manager can communicate with agents running on any IP-addressable machine.

As a consequence, manager applications are generic and can be bought off the shelf. You may already be running SNMP managers, and if so, you can use them to query the SNMP agent.

The manager is initially unaware of the MIB tree structure at a particular node. Managed objects can be retrieved or modified, but only if their location in the tree is known.

It is more useful if the manager can *see* the MIB tree present at each managed node. The MIB module descriptions for a particular node must be parsed by a manager-specific MIB compiler and converted to configuration files. These files are read by the manager application at run time.

The SNMP agent is designed to monitor all current nShield modules, working with all supported versions of nShield firmware (contact Support for details of supported firmware).

8.2. MIB module overview

A large proportion of the SNMP system is fully specified by the structure of the MIB; the behavior of the agent depends on relaying information according to the layout of the MIB.

The MIB module resides at a registered location in the MIB tree determined by the Internet Assigned Numbers Authority (IANA). The private enterprise number of 7682 designated by the IANA corresponds to the root of the branch, and by convention this (internal) node is

the company name.

The MIB module groups logically related data together, organizing itself into a classification tree, with managed objects present at leaf nodes. The nC-series node (`enterprises.nCipher.nC-series`) is placed as a sub-tree of the root (`enterprises.nCipher`); this allows future product lines to be added as additional sub-trees. The structure of the tree underneath the registered location is vendor-defined, and this specification defines the structure chosen to represent Security World Software-specific data.

The MIB file is /opt/nfast/etc/snmp/mibs/ncipher-mib.txt (**Linux**) or `%NFAST_HOME%\etc\snmp\mibs\ncipher-mib.txt` (**Windows**).

8.2.1. MIB functionality

The MIB module separates module information into the following categories:

- Retrieval of status and information about installed nC-series modules
- Retrieval of live statistics of performance of installed nC-series modules

These categories form the top-level nodes of the sub-tree; the functionality of the first category is in the administration sub-tree, and the second category is in the statistics sub-tree. The top-level tree also contains three items that it would be useful to check at-a-glance:

Node name	R/W	Type	Remarks
`hardserverFailed`	R	TruthValue	`True` if the remote hardserver is not running. If the hardserver is not running, then most of the rest of the information is unreliable or missing.
`modulesFailed`	R	TruthValue	`True` if any modules have failed.
`load`	R	Unsigned32	Percentage of total available capacity currently utilized.

8.2.2. Traps

The traps sub-tree (`enterprises.nCipher.nC-series.nC-traps`) contains traps that the SNMP agent sends when certain events occur. For details on configuring traps, see USM users.

The following table gives details of the individual traps:

Node name	Description
`hardserverAlert`	This trap is sent when the hardserver fails or is shut down.
`hardserverUnAlert`	This trap is sent when the hardserver restarts.
`moduleAlert`	This trap is sent when a module fails.
`moduleUnAlert`	This trap is sent when a module is restarted after a failure.
`psuAlert`	This trap is sent when a PSU fails.
`psuUnAlert`	This trap is sent when a previously-failed PSU is working again.
`fanfailureAlert`	This trap is sent when a fan fails.
`fanfailureUnAlert`	This trap is sent when a previously-failed fan is working again.
`memoryUsageHighAlert`	This trap is sent when the HSM memory usage high threshold has been reached or exceeded by an HSM. See section on Memory usage monitoring below for more details.
`memoryUsageOkAlert`	This trap is sent when the memory usage for an HSM falls below the HSM memory usage ok threshold. See section on Memory usage monitoring below for more details.

Some traps can take up to five minutes to be received.

Other generic Net-SNMP traps may also be received. These include the two below, see Net-SNMP project website for more details.

Net-SNMP trap name	Description
`SNMPv2-MIB::coldStart`	This trap is sent when the SNMP agent is started
`NET-SNMP-AGENT-MIB::nsNotifyShutdown`	This trap is sent when the SNMP agent is stopped

8.3. Memory usage monitoring

The HSM memory usage thresholds and memory usage traps provide a mechanism to moni tor HSM memory usage for HSMs in which the SNMP agent's client computer are enrolled.

With memory usage monitoring enabled, there will be a `memoryUsageHighAlert` trap sent each time the currently in-use `memoryUsageHighThreshold` is reached or exceeded by an HSM.

With memory usage monitoring enabled, a `memoryUsageHighAlert` trap is also sent:

- If the SNMP agent starts up and recognises that there are HSMs in a high memory

usage state or,

- If HSMs in a high memory usage state are enrolled or,
- If the SNMP agent loses and then re-gains contact with the local hardserver which is connected to HSMs in a high memory usage state or,
- If failed HSMs in a high memory usage state then recover.

For each of the four scenarios above, one `memoryUsageHighAlert` trap will be sent for each HSM in a high memory usage state.

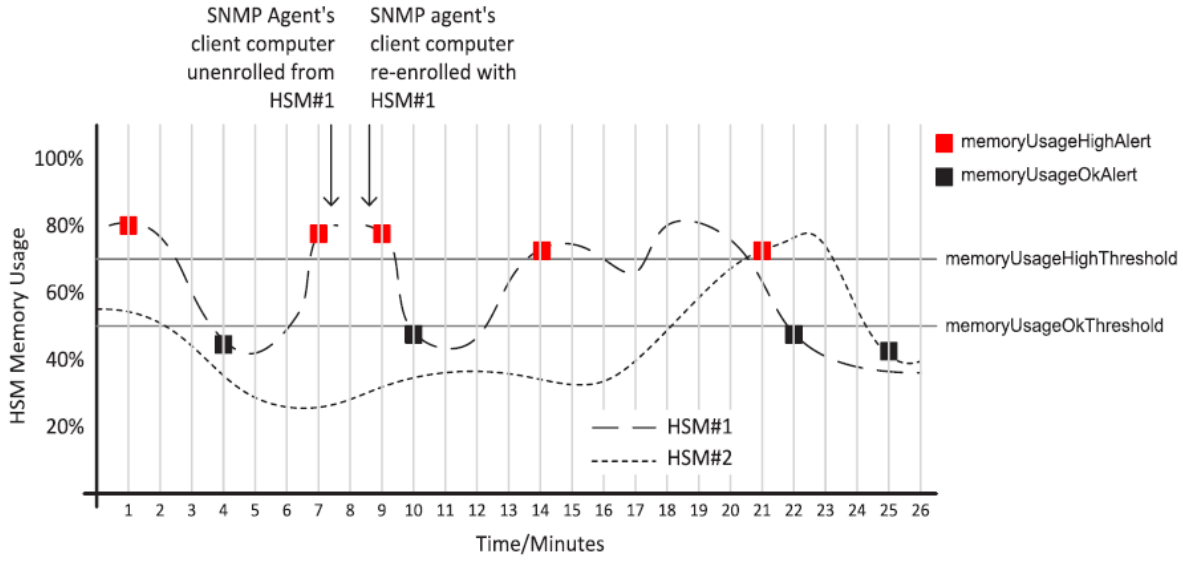
With memory usage monitoring enabled, there will be a `memoryUsageOkAlert` trap sent each time the memory usage for an HSM falls below the currently in-use `memoryUsageOkThreshold`.

The value for `memoryUsageOkThreshold` is read from the snmpd.conf file on starting the SNMP agent and is used provided it contains an integer value in the range 0 to 100 (inclusive); otherwise, the default value of 0 is used. The value for `memoryUsageHighThreshold` is processed in the same way.

Memory usage monitoring is enabled unless the in-use values for `memoryUsageOkThreshold` and `memoryUsageHighThreshold` are both 0 or the in-use values are such that `memoryUsageOkThreshold` > `memoryUsageHighThreshold`.

For example, in `snmpd.conf`, if `memoryUsageOkThreshold` is assigned an invalid value and `memoryUsageHighThreshold` is assigned a valid value of say 75%, then memory usage monitoring will be enabled and the values 0% and 75% will be used respectively.

An example of memory usage monitoring by an SNMP agent on a client computer enrolled with 2 HSMs is given below:



8.4. Administration sub-tree overview

The administration sub-tree (`enterprises.nCipher.nC-series.administration`) contains information about the permanent state of the hardserver and the connected modules. It is likely that most of the information in this branch rarely changes over time, unlike the `statistics` branch. The information given in the administration sub-tree is mostly acquired by the `NewEnquiry` command and is supplied both per-module and (where appropriate) aggregated over all modules.

The following table gives details of the individual nodes in the administration sub-tree:

Node name	R/W	Type	Remarks
`hardserverRunning`	R	Enum 1: Running 2: NotRunning	This variable reflects the current state of the hardserver (`Running` or `NotRunning`).
`noOfModules`	R	Gauge32	Number of nC-series modules.
`hsVersion`	R	DisplayString	Hardserver version string.
`globalSpeedIndex`	R	Gauge32	Number of 1024-bit signatures each second.
`globalMinQ`	R	Gauge32	Minimum recommended queue.
`globalMaxQ`	R	Gauge32	Maximum recommended queue.
`securityWorld`	R	TruthValue	`True` if a Security World is installed and operational.
`swState`	R	DisplayString	Security World display flags, as reported by `nfkminfo`.
`listKeys`	R/W	Integer 1: none 2: all 3: query 4: resetquery	Controls the behavior of the key table (switch off, display all keys, enable individual attribute queries, clear the query fields). Displaying all keys can result in a very long list.
`serverFlags`	R	DisplayString	Supported hardserver facilities (the `NewEnquiry` level 4 flags).
`remoteServerPort`	R	Gauge32	TCP port on which the hardserver is listening.
`swGenTime`	R	DisplayString	Security World's generation time.

Node name	R/W	Type	Remarks
swGeneratingESN	R	DisplayString	ESN of the module that generated the Security World.

listKeys can be preset using the keytable config directive in snmpd.conf file (see The SNMP configuration file: snmp.conf).

8.4.1. Security World hash sub-tree

The following table gives details of the nodes in the Security World hash sub-tree (enterprises.nCipher.nC-series.administration.swHashes):

Node name	R/W	Type	Remarks
hashKNSO	R	MHash	Hash of the Security Officer's key.
hashKM	R	MHash	Hash of the Security World key.
hashKRA	R	MHash	Hash of the recovery authorization key.
hashKRE	R	MHash	Hash of the recovery key pair.
hashKFIPS	R	MHash	Hash of the FIPS authorization key.
hashKMC	R	MHash	Hash of the module certification key.
hashKP	R	MHash	Hash of the passphrase replacement key.
hashKNV	R	MHash	Hash of the nonvolatile memory (NVRAM) authorization key.
hashKRTC	R	MHash	Hash of the Real Time Clock authorization key.
hashKDSEE	R	MHash	Hash of the SEE Debugging authorization key.
hashKFTO	R	MHash	Hash of the Foreign Token Open authorization key.

8.4.2. Security World quorums sub-tree

The following table gives details of the nodes in the Security World quorums sub-tree (enterprises.nCipher.nC-series.administration.swQuorums):

Node name	R/W	Type	Remarks
adminQuorumK	R	Gauge32	The default quorum of Administrator cards.
adminQuorumN	R	Gauge32	The total number of cards in the ACS.
adminQuorumM	R	Gauge32	The quorum required for module reprogramming.
adminQuorumR	R	Gauge32	The quorum required to transfer keys for OCS replacement.
adminQuorumP	R	Gauge32	The quorum required to recover the passphrase for an Operator card.
adminQuorumNV	R	Gauge32	The quorum required to access non volatile memory (NVRAM).
adminQuorumRTC	R	Gauge32	The quorum required to update the Real Time Clock.
adminQuorumDSEE	R	Gauge32	The quorum required to view full SEE debug information.
adminQuorumFTO	R	Gauge32	The quorum required to use a Foreign Token Open Delegate Key.

8.4.3. Module administration table

The following table gives details of the nodes in the module administration table (enterprises.nCipher.nC-series.administration.moduleAdminTable):

Node name	R/W	Type	Remarks
moduleAdminIndex	R	Gauge32	Module number of this row in the table.

Node name	R/W	Type	Remarks
mode	R	Integer 1: Operational 2: Pre-init 3: Init 4: Pre-maint 5: Maint 6: AccelOnly 7: Failed 8: Unknown	Current module state.
fwVersion	R	DisplayString	Firmware version string.
speedIndex	R	Gauge32	Speed index (approximate number of 1024-bit modulo exponentiation operations possible per second) of module
minQ	R	Gauge32	Module minimum recommended queue length
maxQ	R	Gauge32	Module maximum recommended queue length
serialNumber	R	DisplayString	Module Electronic Serial Number (ESN).
productName	R	DisplayString	
hwPosInfo	R	DisplayString	Hardware bus/slot info (such as PCI slot number).
moduleSecurityWorld	R	TruthValue	Indicates whether or not the module is in the current SW.
smartcardState	R	DisplayString	Description of smart card in slot (empty, unknown card, admin/operator card from current SW, failed). N/A for acceleration only modules.

Node name	R/W	Type	Remarks
moduleSWState	R	Integer 1: Unknown 2: Usable 3: MaintMode 4: Uninitialized 5: Factory 6: Foreign 7: AccelOnly 8: Failed 9: Unchecked 10: InitMode 11: PreInitMode 12: Unverified 13: UnusedTableEntry	Current module and Security World state.
moduleSWFlags	R	DisplayString	Security World flags for this module.
hashKML	R	MHash	Hash of the module's secret key.
moduleFeatures	R	DisplayString	Features enabled on this module.
moduleFlags	R	DisplayString	Like serverFlags, but for each module.
versionSerial	R	Gauge32	Firmware Version Security Number (VSN);
hashKNETI	R	MHash	K_{NETI} hash, if present.
longQ	R	Gauge32	Max. rec. long queue.
connectionStatus	R	DisplayString	Connection status (for imported modules).
connectionInfo	R	DisplayString	Connection information (for imported modules).
machineTypeSEE	R	DisplayString	SEE machine type.

8.4.4. Slot administration table

The following table gives details of the nodes in the slot administration table (`enterprises.nCipher.nC-series.administration.slotAdminTable`):

Node name	R/W	Type	Remarks
`slotAdminModuleIndex`	R	Integer32	Module number of the module containing the slot.
`slotAdminSlotIndex`	R	Integer32	Slot number (1-based, unlike nCore which is 0-based).
`slotType`	R	Integer 1: Datakey 2: Smart card 3: Emulated 4: Soft token 5: Unconnected 6: Out of range 7: Unknown	Slot type.
`slotFlags`	R	DisplayString	Flags referring to the contents of the slot (from `slotinfo`).
`slotState`	R	Integer 1: Unused 2: Empty 3: Blank 4: Administrator 5: Operator 6: Unidentified 7: Read error 8: Partial 9: Out of range	`Partial` refers to cards in a partially-created card set.
`slotListFlags`	R	DisplayString	Flags referring to attributes of the slot (from `getslotlist`).

Node name	R/W	Type	Remarks
slotShareNumber	R	Gauge32	Share number of card currently in slot, if present.
slotSharesPresent	R	DisplayString	Names of shares present in card currently in slot.

8.4.5. Card set administration table

The following table gives details of the nodes in the card set administration table (enterprises.nCipher.nC-series.administration.cardsetAdminTable):

Node name	R/W	Type	Remarks
hashKLTU	R	MHash	Hash of the token protected by the card set.
cardsetName	R	DisplayString	
cardsetK	R	Gauge32	Required number of cards in the card set.
cardsetN	R	Gauge32	Total number of cards in the card set.
cardsetFlags	R	DisplayString	Other attributes of the card set.
cardsetNames	R	DisplayString	Names of individual cards, if set.
cardsetTimeout	R	Gauge32	Token time-out period, in seconds, or 0 if none.
cardsetGenTime	R	DisplayString	Generation time of card set.

8.4.6. Key administration table

The key administration table is visible as long as the listKeys node in the administration sub-tree is set to a value other than none.

The following table gives details of the nodes in the key administration table (enterprises.nCipher.nC-series.administration.keyAdminTable):

Node name	R/W	Type	Remarks
keyAppname	R	DisplayString	Application name.
keyIdent	R	DisplayString	Name of key, as generated by the application.

Node name	R/W	Type	Remarks
keyHash	R	MHash	
keyRecovery	R	Integer 1: Enabled 2: Disabled 3: No key 4: Unknown 5: Invalid 6: Unset	The value unset is never returned by the key table. If you set the value unset, the keys are not filtered based on any of the attributes.
keyProtection	R	Integer 1: Module 2: Cardset 3: No key 4: Unknown 5: Invalid 6: Unset	The value unset is never returned by the key table. If you set the value unset, the keys are not filtered based on any of the attributes.
keyCardsetHash	R	MHash	Hash of the card set protecting the key, if applicable.
keyFlags	R	DisplayString	Certificate and public key flags.
keyExtraEntries	R	Gauge32	Number of extra key attributes.
keySEEInteg	R	DisplayString	SEE integrity key, if present.
keyGeneratingESN	R	DisplayString	ESN of the module that generated the key, if present.
keyTimeLimit	R	Gauge32	Time limit for the key, if set.
keyPerAuthUseLimit	R	Gauge32	Per-authentication use limit for the key.

8.4.7. Key query sub-tree

The key query sub-tree is used if the listKeys node in the administration sub-tree is set to query.

If these values are set, they are taken as required attributes for filtering the list of available keys; if multiple attributes are set, the filters are combined (AND rather than OR).

The following table gives details of the nodes in the key query sub-tree (`enterprises.nCipher.nC-series.administration.keyQuery`):

Node name	R/W	Type	Remarks
`keyQueryAppname`	R/W	DisplayString	Application name.
`keyQueryIdent`	R/W	DisplayString	Name of key, as generated by the application.
`keyQueryHash`	R/W	DisplayString	
`keyQueryRecovery`	R/W	Integer 1: Enabled 2: Disabled 3: No key 4: Unknown 5: Invalid 6: Unset	The value `unset` is never returned by the key table. If you set the value `unset`, the keys are not filtered based on any of the attributes.
`keyQueryProtection`	R/W	Integer 1: Module 2: Cardset 3: No key 4: Unknown 5: Invalid 6: Unset	The value `unset` is never returned by the key table. If you set the value `unset`, the keys are not filtered based on any of the attributes.
`keyQueryCardsetHash`	R/W	DisplayString	Hash of the card set protecting the key, if applicable.
`keyQueryFlags`	R/W	DisplayString	Certificate and public key flags.
`keyQueryExtraEntries`	R/W	Gauge32	Number of extra key attributes.
`keyQuerySEEInteg`	R/W	DisplayString	SEE integrity key, if present.
`keyQueryGeneratingESN`	R/W	DisplayString	ESN of the module that generated the key, if present.

Node name	R/W	Type	Remarks
keyQueryTimeLimit	R/W	Gauge32	Time limit for the key, if set (0 for no limit).
keyQueryPerAuthUseLimit	R/W	Gauge32	Per-authentication use limit for the key (0 for no limit).

8.5. netHSMNetwork sub-tree

netHSMNetwork sub-tree (enterprises.nCipher.nC-series.netHSMNetwork) contains information about the network interfaces of the network-attached HSMs (nShield Connect, nShield 5c).

To output the network interface information in SNMP, you must enable both NIC exposure settings. For information how to enable them, see NIC Reporting.

8.5.1. netHSMNICTable sub-tree

netHSMNICTable (enterprises.nCipher.nC-series.netHSMNetwork.netHSMNICTable) provides information about the name and link state of the network interface.

Node name	R/W	Type	Remarks
ifModule	R	Integer	The remote module number of the HSM
ifIndex	R	Integer	The number of the network interface on the HSM
ifState	R	Integer	The link state of the network interface. 0 = Down 1 = Up
ifName	R	DisplayString	The name of the interface. The interface names are eth0 and eth1. When interface bonding is enabled, a third interface will show as bond0.

8.5.2. netHSMNICAddressTable sub-tree

`netHSMNICAddressTable` (`enterprises.nCipher.nC-series.netHSMNetwork.netHSMNICAddressTable`) provides the network address of the interface.

The maximum number of addresses that can be reported for an interface is 10.

Node name	R/W	Type	Remarks
`ifAddress`	R	`DisplayString`	The IPv4 or IPv6 address of the interface. When interface bonding is enabled, `eth0` and `eth1` interface addresses are displayed as `0.0.0.0`.

8.6. Statistics sub-tree overview

The statistics sub-tree (`enterprises.nCipher.nC-series.statistics`) contains rapidly changing information about such topics as the state of the nShield modules, the work they are doing, and the commands being submitted.

Do not rely on information returned from the agent to change instantaneously on re-reading the value. To avoid loading the nShield module with multiple time-consuming statistics commands, the agent can choose to cache the values over a specified period. You can configure this period in the agent configuration file see Basic configuration.

8.6.1. Statistics sub-tree

The following table gives details of the nodes in the statistics sub-tree, and the module statistics table (`enterprises.nCipher.nC-series.statistics.moduleStatsTable`):

Node name	R/W	Type	Remarks
`moduleStatsIndex`	R	Integer	Module number of this row (for `moduleStatsTable`).
`hsUptime`	R	Counter32	Uptime of the hardserver.
`cmdCountAll`	R	Counter32	Returned aggregated for all modules and all commands.
`cmdBytesAll`	R	Counter32	

Node name	R/W	Type	Remarks
cmdErrorsAll	R	Counter32	Returned as for cmdCount, returned value is combined module errors added to hardserver marshalling/unmarshalling errors.
replyCountAll	R	Counter32	
replyBytesAll	R	Counter32	
replyErrorsAll	R	Counter32	See notes above for cmdErrors.
clientCount	R	Gauge32	
maxClients	R	Counter32	
deviceFails	R	Counter32	
deviceRestarts	R	Counter32	
outstandingCmds	R	Counter32	Total number of outstanding commands over all modules.
loadAll	R	Counter32	

8.6.2. Module statistics table

The following table gives details of the nodes in the module statistics table (enterprises.nCipher.nC-series.statistics.moduleStatsTable):

Node name	R/W	Type	Remarks
moduleStatsIndex	R	Integer	Module number of this row (for moduleStatsTable).
uptime	R	Counter32	Uptime of the module.
cmdCount	R	Counter32	Returned aggregated for all commands.
cmdBytes	R	Counter32	
cmdErrors	R	Counter32	Returned as for cmdCount all the different error states aggregated into one counter.
replyCount	R	Counter32	
replyBytes	R	Counter32	
replyErrors	R	Counter32	See notes above for cmdErrors.

Node name	R/W	Type	Remarks
loadModule	R	Counter32	
objectCount	R	Gauge32	
clock	R	DisplayString	Depending on the module settings, this can require K_{NSO} permissions to read (and therefore depend on the installation parameters of the agent).
currentTemp	R	DisplayString	Character representation of the cur rent temperature value (SNMP does not provide for a floating-point type). Not available on XC or nShield 5 variants.
maxTemp	R	DisplayString	Maximum temperature the module has ever reached. Not available on XC or nShield 5 variants.
nvRAMInUse	R	Gauge32	
volatileRAMInUse	R	Gauge32	
tempSP	R	DisplayString	Only available on XC variants.
currentCPUTemp1	R	DisplayString	Only available on XC variants.
currentCPUTemp2	R	DisplayString	Only available on XC variants.
currentFanSpeed	R	DisplayString	Only available on XC variants.
currentFanDuty	R	DisplayString	Only available on XC variants.
cpuVoltage1	R	DisplayString	Only available on XC variants.
cpuVoltage2	R	DisplayString	Only available on XC variants.
cpuVoltage3	R	DisplayString	Only available on XC variants.
cpuVoltage4	R	DisplayString	Only available on XC variants.
cpuVoltage5	R	DisplayString	Only available on XC variants.
cpuVoltage6	R	DisplayString	Only available on XC variants.
cpuVoltage7	R	DisplayString	Only available on XC variants.
tempSP	R	DisplayString	Only available on nShield 5 variants.
currentCPUTemp1	R	DisplayString	Only available on nShield 5 variants.
currentCPUTemp2	R	DisplayString	Only available on nShield 5 variants.
currentFanSpeed	R	DisplayString	Only available on nShield 5 variants.

Node name	R/W	Type	Remarks
currentFanDuty	R	DisplayString	Only available on nShield 5 variants.
cpuVoltage1	R	DisplayString	Only available on nShield 5 variants.
cpuVoltage2	R	DisplayString	Only available on nShield 5 variants.
cpuVoltage3	R	DisplayString	Only available on nShield 5 variants.
cpuVoltage4	R	DisplayString	Only available on nShield 5 variants.
cpuVoltage5	R	DisplayString	Only available on nShield 5 variants.
cpuVoltage6	R	DisplayString	Only available on nShield 5 variants.
cpuVoltage7	R	DisplayString	Only available on nShield 5 variants.
cpuVoltage8	R	DisplayString	
cpuVoltage8	R	DisplayString	
cpuVoltage9	R	DisplayString	
cpuVoltage10	R	DisplayString	
cpuVoltage11	R	DisplayString	
nvmFreeSpace	R	Counter32	Free space available on the HSM's NVRAM, in bytes Only available on XC and nShield 5 variants.
nvmWearLevel	R	DisplayString	Wear level of the HSM's NVRAM Only available on XC and nShield 5 variants.
nvmWornBlocks	R	DisplayString	Worn blocks in the HSM's NVRAM Only available on XC and nShield 5 variants.

8.6.3. nShield network-attached HSM statistics table

The following table gives details of the nodes in the nShield network-attached HSM statistics table (enterprises.nCipher.nC-series.statistics.netHSMStatsTable):

Node name	R/W	Type	Remarks
netHSMStatsIndex	R	Integer	Table index (not module number).

Node name	R/W	Type	Remarks
netHSMUptime	R	Counter32	Host system uptime.
netHSMCPUUsage	R	Gauge32	CPU usage of unit host processor.
netHSMUserMem	R	Gauge32	Total user memory of unit.
netHSMKernelMem	R	Gauge32	Total kernel memory of unit.
netHSMCurrentTemp	R	DisplayString	Internal unit temperature (sensor 1).
netHSMMaxTemp	R	DisplayString	Maximum recorded temperature (sensor 1).
netHSMCurrentTemp2	R	DisplayString	Internal unit temperature (sensor 2).
netHSMMaxTemp2	R	DisplayString	Maximum recorded temperature (sensor 2).
netHSMVoltage5V	R	DisplayString	unit 5V power reading.
netHSMVoltage12V	R	DisplayString	unit 12V power reading.
netHSMFan1Speed	R	Gauge32	Fan 1 speed (RPM).
netHSMFan2Speed	R	Gauge32	Fan 2 speed (RPM).
netHSMFan3Speed	R	Gauge32	Fan 3 speed (RPM).
netHSMIPAddress	R	IpAddress	IP address of unit.
netHSMDescription	R	DisplayString	Textual description of module (for example, `Local module 3`).
netHSMFan4Speed	R	Gauge32	Fan 4 speed (RPM).
netHSMVoltage3p3V	R	DisplayString	3.3V Supply Rail Voltage
netHSMCurrent3p3V	R	DisplayString	3.3V Supply Rail Current
netHSMCurrent5V	R	DisplayString	5V Supply Rail Current
netHSMCurrent12V	R	DisplayString	12V Supply Rail Current
netHSMVoltage5VSB	R	DisplayString	5V Supply Rail Voltage (Standby)
netHSMCurrent5VSB	R	DisplayString	5V Supply Rail Current (Standby)
netHSMTamperBattery1	R	DisplayString	Voltage of Tamper Battery 1
netHSMTamperBattery2	R	DisplayString	Voltage of Tamper Battery 2
netHSMPSUFailure	R	TruthValue	Power Supply failure status

8.6.4. Per connection statistics table

The following table gives details of the nodes in the per connection statistics table (enterprises.nCipher.nC-series.statistics.connStatsTable):

Node name	R/W	Type	Remarks
connStatsIndex	R	Integer32	Index of this entry.
connNumber	R	Integer32	Hardserver connection number.
connUptime	R	Counter32	Uptime of the connection.
connCmdCount	R	Counter32	Number of commands submitted through this connection.
connCmdBytes	R	Counter32	Number of bytes submitted through this connection.
connCmdErrors	R	Counter32	Number of marshalling errors on commands through this connection.
connReplyCount	R	Counter32	Number of replies received by this connection.
connReplyBytes	R	Counter32	Number of bytes received by this connection.
connReplyErrors	R	Counter32	Number of marshalling errors on replies through this connection.
connDevOutstanding	R	Gauge32	Number of commands outstanding on this connection.
connQOutstanding	R	Gauge32	Number of commands outstanding in the hardserver queue.
connLongOutstanding	R	Gauge32	Number of long jobs outstanding for this connection.
connRemoteIPAddress	R	IpAddress	IP Address of connection client.
connProcessID	R	Integer32	Process identifier reported by connection client.
connProcessName	R	DisplayString	Process name reported by connection client.
connObjectTotal	R	Gauge32	The total object count for a connection

8.6.5. Module/connection statistics table

The following table gives details of the nodes in the per module, per connection statistics table (enterprises.nCipher.nC-series.statistics.connModuleStatsTable).

Node name	R/W	Type	Remarks
connModuleStatsConnId	R	Integer	Identity of this connection
connModuleStatsModuleIndex	R	Integer	Index of the module entry
connModuleStatsObjectCount	R	Gauge32	The object count on this module for this connection

8.6.6. Fan table

The fan table provides the speeds of each fan on the remote module (HSM). The following table gives details of the nodes in the fan table (enterprises.nCipher.softwareVersions.netHSMFanTable):

Node name	R/W	Type	Remarks
netHSMModuleIndex	R	Integer32	Module number
netHSMFanIndex	R	Integer32	Fan number
netHSMFanSpeed	R	Gauge32	Fan speed (RPM)

8.6.7. Software versions table

The following table gives details of the nodes in the software versions table (enterprises.nCipher.softwareVersions.softwareVersionsTable):

Node name	R/W	Type	Remarks
compIndex	R	Integer	Table index.
compName	R	DisplayString	Component name.
compOutput	R	Component output name	Component name.
compMajorVersion	R	Gauge	
compMinorVersion	R	Gauge	
compPatchVersion	R	Gauge	
compRepository	R	DisplayString	Repository name.
compBuildNumber	R	Gauge	

9. SNMP agent command-line

9.1. SNMP agent (snmpd) switches

The SNMP agent that binds to a port and awaits requests from SNMP management software is `snmpd`. Upon receiving a request, snmpd processes the request, collects the requested information and/or performs the requested operation(s) and returns the information to the sender.

The SNMP agent supports a limited subset of command line switches that can be specified when starting the agent.

Usage

```
snmpd [-h] [-v] [-f] [-a] [-d] [-V] [-P PIDFILE):] [-q] [-D] [-p NUM] [-L] [-l LOGFILE] [-r]
```

This command can take the following options:

Option	Description
-h	This option displays a usage message.
-H	This option displays the configuration file directives that the agent understands.
-v	This option displays version information.
-f	This option specifies not forking from the calling shell.
-a	This option specifies logging addresses.
-A	This option specifies that warnings and messages should be appended to the log file rather than truncating it.
-d	This option specifies the dumping of sent and received UDP SNMP packets.
-V	This option specifies verbose display.
-P	PIDFILE This option specifies the use of a file (PIDFILE) to store the process ID.
-q	This option specifies that information be printed in a more easily parsed format (quick print).
-D	This option turns on debugging output.
-p	NUM This option specifies running on port NUM instead of the default: 161.
-c	CONFFILE This option specifies reading CONFFILE as a configuration file.
-C	This option specifies that the default configuration files not be read.
-L	This option prints warnings and messages to stdout and err.

Option	Description
`-s`	This option logs warnings/messages to syslog.
`-r`	This option specifies not exiting if root-only accessible files cannot be opened.
`-I`	[-]INITLIST This option specifies a list of MIB modules to initialize (or not). Run `snmpd` with the `-Dmib_init` option for a list.
`-l`	LOGFILE This option prints warnings/messages to a file LOGFILE (by default, `LOGFILE=log/snmpd.log`).

9.2. Using the SNMP command-line utilities

As an alternative to using an SNMP manager application, we supply several command-line utilities to test your SNMP installation and enable you to obtain information about your nShield module from the SNMP agent. These utilities support the `-h` (display a usage message) as described in the table above.

Utility	Description
`snmptest`	This utility monitors and manages SNMP information.
`snmpget`	This utility runs a single GET request to query for SNMP information on a network entity.
`snmpset`	This utility runs a single SET request to set SNMP information on a network entity.
`snmpgetnext`	This utility runs a single GET NEXT request to query for SNMP information on a network entity.
`snmptable`	This utility obtains and prints an SNMP table.
`snmptranslate`	This utility translates SNMP object specifications into human-readable descriptions.
`snmpwalk`	This utility communicates with a network entity using repeated GET NEXT requests.
`snmpbulkwalk`	This utility communicates with a network entity using BULK requests.

These tools are general purpose SNMP utilities and are configurable for use with other SNMP agents. For more information on configuring and using these tools, refer to the NET-SNMP project Web site: `http://net-snmp.sourceforge.net/`.