SNMP Monitoring

Cumulus Linux utilizes the open source Net-SNMP agent snmpd, v5.7.3, which provides support for most of the common industry-wide MIBs, including interface counters and TCP/UDP IP stack data.

Contents

- Introduction to SNMP (Simple Network Management Protocol)
- Configuring Ports for SNMP to Listen for Requests
- Quick Start Guide
- Starting the SNMP Daemon
- Configuring SNMP
  - Configuring SNMP with NCLU
  - Configuring SNMP Manually
  - Configuring SNMP with Management VRF
  - Configuring the agentAddress
  - Setting up the Custom Cumulus Networks MIBs
  - Setting the Community String
  - Enabling SNMP Support for FRRouting
  - Enabling the 1.3.6.1.2.1 Range
  - Configuring SNMPv3
- snmpwalk a Switch from Another Linux Device
- Troubleshooting Tips Table for snmpwalks
- SNMP Traps
-Generating Event Notification Traps
- snmptrapd.conf
- Supported MIBs
- About Pass Persist Scripts
- Troubleshooting

Introduction to SNMP (Simple Network Management Protocol)

SNMP is an IETF standards-based network management architecture and protocol that traces its roots back to Carnegie-Mellon University in 1982. Since then, it's been modified by programmers at the University of California. In 1995, this code was also made publicly available as the UCD project. After that, ucd-snmp was extended by work done at the University of Liverpool as well as later in Denmark. In late 2000, the project name changed to net-snmp and became a fully-fledged collaborative open source project. The version used by Cumulus Networks is base on the latest net-snmp 5.7.3 branch with added custom MIBs and pass through and pass persist scripts (see below for more information on pass persist scripts).

Configuring Ports for SNMP to Listen for Requests

For security reasons, the default port binding for snmpd is the loopback local address; consequently by default, the SNMP service does not listen for SNMP requests from outside the switch. In order to listen to requests from outside the switch, you need to change this binding to a specific IP address (or all interfaces) after configuring security access (community strings, users, and so forth). This is a change from older versions of Cumulus Linux (before version 3.0), which listened to incoming requests on all interfaces by default. The snmpd configuration file is /etc/snmp/snmpd.conf and should be modified before enabling and starting snmpd. The default configuration has no access community strings defined so snmpd will not respond to any SNMP requests until this is added.

Quick Start Guide

The SNMP daemon, snmpd, uses the configuration file /etc/snmp/snmpd.conf for almost all of its configuration. The syntax of the most important keywords are defined in the following table.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Meaning</th>
</tr>
</thead>
</table>


agentAddress

Required. This command sets the protocol, IP address, and the port for `snmpd` to listen on for incoming requests. The IP address must exist on an interface that has link UP on the switch where `snmpd` is being used. By default, this is set to `udp:127.0.0.1:161`, which means `snmpd` listens on the loopback interface and only responds to requests (`snmpwalk, snmpget, snmpgetnext`) originating from the switch. A wildcard setting of `udp:161,udp6:161` forces `snmpd` to listen on all IPv4 and IPv6 interfaces for incoming SNMP requests. Multiple IP address can be configured as comma-separated values, as in `udp:66.66.66.66:161,udp:77.77.77.77:161,udp6:[2001::]:161`.

rocommunity

Required. This command defines the protocol, IP address, and the port for `snmpd` to listen on for incoming requests. The IP address must exist on an interface that has link UP on the switch where `snmpd` is being used. By default, this is set to `udp:127.0.0.1:161`, which means `snmpd` listens on the loopback interface and only responds to requests (`snmpwalk, snmpget, snmpgetnext`) originating from the switch. A wildcard setting of `udp:161,udp6:161` forces `snmpd` to listen on all IPv4 and IPv6 interfaces for incoming SNMP requests. Multiple IP address can be configured as comma-separated values, as in `udp:66.66.66.66:161,udp:77.77.77.77:161,udp6:[2001::]:161`.

view

This command defines a view name that specifies a subset of the overall OID tree. This restricted view can then be referenced by name in the `rocommunity` command to link the view to a password that is used to see this restricted OID subset. By default, the `snmpd.conf` file contains numerous views with the `systemonly` view name:

```
view systemonly included .1.3.6.1.2.1.1
view systemonly included .1.3.6.1.2.1.2
view systemonly included .1.3.6.1.2.1.3
```

The `systemonly` view is used by `rocommunity` to create a password for access to only these branches of the OID tree.

trapsink

This command defines the IP address of the notification (or trap) receiver for either SNMPv1 traps or SNMPv2 traps. If several sink directives are specified, multiple copies of each notification (in the appropriate formats) are generated. Note that a trap server must be configured to receive and decode these trap messages (for example, `snmptrapd`). The address of the trap receiver can be configured with a different protocol and port but this is most often left out. The defaults are to use the well-known UDP packets and port 162.

createUser

These three commands define an internal SNMPv3 username that is required in order for `snmpd` to send traps. This username is required to authorize the DisMan service even though SNMPv3 is not being configured for use. The example `snmpd.conf` configuration shown below creates `snmptrapusernameX` as the username (this is just an example username) using the `createUser` command. `iquerySecName` defines the default SNMPv3 username to be used when making internal queries to retrieve monitored expressions. `rouser` specifies which username should be used for these SNMPv3 queries. All three are required for `snmpd` to retrieve information and send built-in traps or for those configured with the `monitor` command shown below in the examples.

```
createUser snmptrapusernameX
iquerySecName snmptrapusernameX
rouser snmptrapusernameX
```
**linkUpDownNotifications**  
**yes**  
This command enables link up and link down trap notifications, assuming the other trap configurations settings are set. This command configures the Event MIB tables to monitor the ifTable for network interfaces being taken up or down, and triggering a *linkUp* or *linkDown* notification as appropriate. This is exactly equivalent to the following configuration:

```
notificationEvent  linkUpTrap  linkUp  ifIndex
ifAdminStatus  ifOperStatus

notificationEvent  linkDownTrap  linkDown  ifIndex
ifAdminStatus  ifOperStatus

monitor  -r 60  -e linkUpTrap  "Generate linkUp"
ifOperStatus != 2

monitor  -r 60  -e linkDownTrap  "Generate linkDown"
ifOperStatus == 2
```

**defaultMonitors**  
**yes**  
This command configures the Event MIB tables to monitor the various UCD-SNMP-MIB tables for problems (as indicated by the appropriate xxErrFlag column objects) and send a trap. This assumes the user has downloaded the `snmp-mibs-downloader` Debian package and comments out "mibs" from `/etc/snmp/snmp.conf` (as in: "#mibs"). This command is exactly equivalent to the following configuration:

```
monitor  -o prNames  -o prErrMessage  "process table"
prErrorFlag != 0

monitor  -o memErrorName  -o memSwapErrorMsg  "memory"
memSwapError != 0

monitor  -o extNames  -o extOutput  "extTable"  extResult != 0

monitor  -o dskPath  -o dskErrorMsg  "dskTable"
dskErrorFlag != 0

monitor  -o laNames  -o laErrMessage  "laTable"
laErrorFlag != 0

monitor  -o fileName  -o fileErrorMsg  "fileTable"
fileErrorFlag != 0
```

**Starting the SNMP Daemon**

The following procedure is the recommended process to start `snmpd` and monitor it using `systemctl`.

To start the SNMP daemon:

1. Start the `snmpd` daemon:

```
cumulus@switch:~$ sudo systemctl start snmpd.service
```
2. Configure the `snmpd` daemon to start automatically after reboot:

   ```
cumulus@switch:~$ sudo systemctl enable snmpd.service
   ```

3. To enable `snmpd` to restart automatically after failure:
   a. Create a file called `/etc/systemd/system/snmpd.service.d/restart.conf`
   b. Add the following lines:

   ```
   [Service]
   Restart=always
   RestartSec=60
   ```

   c. Run `sudo systemctl daemon-reload`

Once the service is started, SNMP can be used to manage various components on the Cumulus Linux switch.

### Configuring SNMP

Cumulus Linux ships with a production usable default `snmpd.conf` file included. This section covers a few basic configuration options in `snmpd.conf`. For more information regarding further configuring this file, refer to the `snmpd.conf` man page.

Cumulus Linux 3.4 and later releases support configuring SNMP with NCLU.

The default `snmpd.conf` file does not include all supported MIBs or OIDs that can be exposed.

Customers must at least change the default community string for v1 or v2c environments or the `snmpd` daemon will not respond to any requests.

### Configuring SNMP with NCLU

The table below highlights the structure of NCLU commands available for configuring SNMP. An example command set can be found below the table.

<table>
<thead>
<tr>
<th>Command</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>net del all</code> or <code>net del snmp-server all</code></td>
<td>Removes all entries in the <code>/etc/snmp/snmpd.conf</code> file and replaces them with defaults. The defaults remove all SNMPv3 usernames, readonly-communities, and a listening-address of localhost will be configured.</td>
</tr>
<tr>
<td><code>net add snmp-server listening-address localhost</code></td>
<td>For security reasons, the localhost is set to a listening address 127.0.0.1 by default. This means that the SNMP agent will only respond to requests originating on the switch itself. One or more IP addresses can be deleted.</td>
</tr>
<tr>
<td><code>net add snmp-server listening-address all</code></td>
<td>Configures the <code>snmpd</code> agent to listen on all interfaces for UDP port 161 SNMP requests.</td>
</tr>
</tbody>
</table>
net add snmp-server listening-address IP_ADDRESS IP_ADDRESS ...

Sets the SNMP agent snmpd to listen to a specific IPv4 or IPv6 address, or a group of addresses with space separated values, for incoming SNMP queries.

```plaintext
net add snmp-server listening-address 10.10.10.10
net add snmp-server listening-address 10.10.10.10 44.44.44.44
```

net add snmp-server viewname [view name] (included | excluded) [OID or name]

Creates a view to restrict MIB tree exposure. By itself, this view definition has no effect; however, when linked to an SNMPv3 username or community password, and a host from a restricted subnet, any SNMP request with that username/password must have a source IP address within the configured subnet.

Note that OID can be either a string of period separated decimal numbers or a unique text string that identifies an SNMP MIB object. Some MIBs are not installed by default and must be installed either by hand or with the latest Debian package called snmp-mibs-downloader. Specific viewname entries can be removed with the delete command or with just a view name to remove all entries matching that view name. A specific view name can be defined multiple times and fine tuned to provide or restrict access using included or excluded command to specify branches of certain MIB trees.

```plaintext
net add snmp-server viewname cumulusOnly included .1.3.6.1.4.1.40310
```

net add snmp-server (readonly-community | readonly-community-v6) [password] access (any | localhost | [network]) [(view [view name]) or [oid [oid or name]]

Defines the community password, and which parts of the OID tree to apply the password to for incoming SNMP requests.

```plaintext
net add snmp-server trap-destination (localhost | [ipaddress]) community-password [password] [version [1 | 2c]]
```

Sets the SNMP Trap destination IP address. Multiple destinations can exist, but at least one must be set to enable SNMP Traps to be sent. Removing all settings will disable SNMP traps.

The default version is 2c, unless otherwise configured.

```plaintext
net add snmp-server trap-link-up [check-frequency [seconds]]
```

Enables notifications for interface link-up to be sent to SNMP Trap destinations.

```plaintext
net add snmp-server trap-link-down [check-frequency [seconds]]
```

Enables notifications for interface link-down to be sent to SNMP Trap destinations.

```plaintext
net add snmp-server trap-snmp-auth-failures
```

Enables SNMP Trap notifications to be sent for every SNMP authentication failure.

```plaintext
```

Enables a trap when the cpu-load-average exceeds the configured threshold. Only integers or floating point numbers can be used.

This table covers system setting configuration commands for SNMPv2-MIB:

<table>
<thead>
<tr>
<th>Command</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>net add snmp-server system-location [string]</td>
<td>Sets the system physical location for the node in the SNMPv2-MIB system table.</td>
</tr>
</tbody>
</table>
The example commands below enable an SNMP agent to listen on all IP addresses with a community string password, set the trap destination host IP address, and create four types of SNMP traps:

```bash
cumulus@switch:~$ net add snmp-server listening-address all
cumulus@switch:~$ net add snmp-server readonly-community tempPassword access any
cumulus@switch:~$ net add snmp-server trap-destination 1.1.1.1 community-password mypass version 2c
cumulus@switch:~$ net add snmp-server trap-link-up check-frequency 15
cumulus@switch:~$ net add snmp-server trap-link-down check-frequency 10
cumulus@switch:~$ net add snmp-server trap-cpu-load-average one-minute 7.45 five-minute 5.14
cumulus@switch:~$ net add snmp-server trap-snmp-auth-failures
```

### Configuring SNMP Manually

There are times when you need to manually edit the SNMP configuration; for example, there may not be the necessary option in NCLU. In cases like this, you need to edit the configuration directly, which is stored in the `/etc/snmp/snmpd.conf` file.

Use caution when making direct edits to the file, however, because the next time you use NCLU to update your SNMP configuration, some of the options you specified may get overwritten by NCLU as it may not be able to correctly parse the syntax. However, if NCLU is not aware of a given configuration option, it will not overwrite such an option.

Make sure you do not delete the `snmpd.conf` file as this can cause issues with the package manager the next time you update Cumulus Linux.

### Configuring SNMP with Management VRF

When you configure Management VRF, you need to be aware of the interface IP addresses on which SNMP is listening. If you set listening-address to all, the `snmpd` daemon responds to incoming requests on all interfaces in the default VRF. If you prefer to listen on a limited number of IP addresses, Cumulus Networks recommends that you run only one instance of the `snmpd` daemon and that all of the configured interfaces are in the same VRF.

SNMP configuration in NCLU is not VRF aware so the `snmpd` daemon is always started in the default VRF. Because interfaces in a particular VRF (routing table) are not aware of interfaces in a different VRF, the `snmpd` daemon only responds to polling requests and sends traps on the interfaces of the VRF on which it is running.

When management VRF is configured, most users will want to start the `snmpd` daemon in the management VRF to receive and respond to SNMP polling requests on eth0. Follow these guidelines:

1. Configure all the required SNMP settings with NCLU. Pay particular attention to the listening-address configuration setting, which should contain one or more IP addresses that belong to interfaces within a single VRF (if management VRF is configured, this is typically the IP address of eth0). You can use IP addresses other than eth0, but the interfaces for these IP addresses must be in the same VRF (typically the management VRF).
2. Commit the changes to start the `snmpd` daemon in the default VRF.
3. Manually stop the `snmpd` daemon from running in the default VRF.
4. Manually restart the `snmpd` daemon in the management VRF.

### Running Multiple Instances of `snmpd`

More complex configurations are possible; for example, you can run more than one `snmpd` daemon (one in each VRF designed to receive SNMP polling requests). Cumulus Networks does not recommend this for memory and cpu resource reasons. However, if this is required, you must use a separate configuration file with each instance of the `snmpd` daemon. You can use a copy of the `/etc/snmp/snmpd.conf` file. When you use this file, start an `snmpd` daemon with the following command:
To use management VRF, you need to configure the IP address of eth0 as the listening-address. In the example below, eth0 IP address is 10.10.10.10. You can also add other snmp-server configurations, then commit the changes.

```bash
cumulus@switch:~$ net add snmp-server listening-address 10.10.10.10
cumulus@switch:~$ net add snmp-server readonly-community tempPassword access any
cumulus@switch:~$ net pending
```

This restarts the snmpd daemon in the default VRF. Then, to run snmpd in the correct VRF, stop the daemon in the default VRF (or stop any other snmpd daemons that happen to be running), then restart snmpd in the management VRF so that it can respond to requests on interfaces only in that VRF. Make sure that only one instance of the snmpd daemon is running and that it is running in the desired VRF. Assuming the Management VRF has been enabled, the following example shows how to stop snmpd and restart it in the management VRF.

```bash
cumulus@switch:mgmt-vrf:~$ systemctl stop snmpd.service
root@switch:mgmt-vrf:/home/cumulus# systemctl status snmpd@mgmt.service
Dec 07 20:05:41 cel-redxp-01 systemd[1]: Started Simple Network Management Protocol (SNMP) Daemon..
```

Configuring the agentAddress

As mentioned earlier, you need to configure the transport protocol, IP address and port where SNMP listens. In Cumulus Linux, the transport defaults to UDP, the IP address defaults to the localhost (127.0.0.1) and the port defaults to 161. If you want to change any of these settings, do the following:

1. Open the /etc/snmp/snmpd.conf file in a text editor, and edit the following line:
Setting up the Custom Cumulus Networks MIBs

No changes are required in the `/etc/snmp/snmpd.conf` file on the switch, in order to support the custom Cumulus Networks MIBs. The following lines are already included by default:

```
view systemonly included .1.3.6.1.4.1.40310.1
view systemonly included .1.3.6.1.4.1.40310.2
sysObjectID 1.3.6.1.4.1.40310
pass_persist .1.3.6.1.4.1.40310.1 /usr/share/snmp/resq_pp.py
pass_persist .1.3.6.1.4.1.40310.2 /usr/share/snmp/cl_drop_cntrs_pp.py
```

However, several files need to be copied to the server, in order for the custom Cumulus MIB to be recognized on the destination NMS server.

- `/usr/share/snmp/mibs/Cumulus-Snmp-MIB.txt`
- `/usr/share/snmp/mibs/Cumulus-Counters-MIB.txt`
- `/usr/share/snmp/mibs/Cumulus-Resource-Query-MIB.txt`

Setting the Community String

The `snmpd` authentication for versions 1 and 2 is disabled by default in Cumulus Linux. This password (called a community string) can be enabled by setting `rocommunity` (for read-only access) or `rwcommunity` (for read-write access). Setting a community string is required.

To enable read-only querying by a client:

1. Open `/etc/snmp/snmpd.conf` in a text editor.
2. To allow read-only access, uncomment the following line, then save the file:

```
rocommunity public default -V systemonly
```

The line can be broken down as follows:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>rocommunity</td>
<td>Read-only community; <code>rwcommunity</code> is for read-write access.</td>
</tr>
</tbody>
</table>
public | Plain text password/community string.

Cumulus Networks strongly recommends you change this password to something else.

default | The *default* keyword allows connections from any system. The *localhost* keyword allows requests only from the local host. A restricted source can either be a specific hostname (or address), or a subnet, represented as IP/MASK (like 10.10.0.0/255.255.255.0), or IP/BITS (like 10.10.0.0/24), or the IPv6 equivalents.

systemonly | The name of this particular SNMP view. This is a user-defined value.

3. **Restart** `snmpd`:

```bash
cumulus@switch:~$ sudo systemctl restart snmpd.service
```

### Enabling SNMP Support for FRRouting

As of Cumulus Linux 3.3.1, SNMP is now supported for FRRouting. To enable SNMP support for FRRouting, you need to:

- Configure AgentX (ASX) access in FRRouting
- Create an SNMP-specific `frr.conf` file
- Restart the SNMP and FRRouting services

Enabling FRRouting includes support for BGP. However, if you plan on using the BGP4 MIB, you need to expose `.1.3.6.1.2.1.15` in the `/etc/snmp/snmpd.conf` file.

Similarly, if you plan on using the OSPFv2 MIB, you need to expose `.1.3.6.1.2.1.14` in the `/etc/snmp/snmpd.conf` file, and expose `.1.3.6.1.2.1.102` for the OSPv3 MIB.

To enable SNMP support for FRRouting, do the following:

1. **Configure AgentX access in FRRouting:**

   ```bash
cumulus@switch:~$ net add routing agentx
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

2. **Update the SNMP configuration to enable FRRouting to respond to SNMP requests.** Open the `/etc/snmp/snmpd.conf` file in a text editor, and make sure `agentx` is enabled and the daemon has some listening addresses and some type of authentication configured:

   ```
   agentAddress udp:161
   rocommunity public default

   # make sure snmpd agentx service is configured to allow FRRouting to respond to snmp requests. These are default and should already be configured
   agentxperms 777 777 snmp snmp
   agentxsocket /var/agentx/master
   master  agentx
   ```

   The rocommunity password is defined above.

3. **Optionally, you need to uncomment parts of snmpd.conf if you intend to use SNMP with the following MIBs:**
   - For the BGP4 MIB, uncomment the `view systemonly included .1.3.6.1.2.1.15` line below.
   - For the OSPF MIB, uncomment the `view systemonly included .1.3.6.1.2.1.14` line below.
For the OSPFV3 MIB, uncomment the view systemonly included .1.3.6.1.2.1.102 line below.

```plaintext
# Note: FRRouting snmpd support has been reenabled.
# Please see FRRouting documentation for instructions
# on enabling AgentX functionality in FRRouting and
# also set agentxsocket and agentxperms at the bottom
# of this config file.
#
# Uncomment the following to enable OSPF, OSPFV3 and BGP4 MIBs
# The following line exposes OSPF-MIB in this view
# view systemonly included .1.3.6.1.2.1.14
# frrouting bgp
# The following line exposes BGP4-MIB in this view
# view systemonly included .1.3.6.1.2.1.15
# frrouting ospf6
# The following line exposes OSPFV3-MIB in this view
# view systemonly included .1.3.6.1.3.102
#
# This pass persist script bgp4_pp.py is deprecated. Please enable
# AgentX support in FRRouting for BGP4-MIB support.
```

4. After you save the snmpd.conf file, create a file called /etc/snmp/frr.conf that contains the following line:

```plaintext
agentXSocket /run/agentx/master
```

5. After you save this file, restart the snmpd and FRRouting services for these changes to take effect and to reload the FRRouting daemons with AgentX access:

```plaintext
cumulus@switch:~$ sudo systemctl restart snmpd.service
cumulus@switch:~$ sudo systemctl restart frr.service
```

To verify the configuration, run snmpwalk. For example, if you have a running OSPF configuration with routes, you can check this OSPF-MIB first from the switch itself with:

```plaintext
cumulus@switch:~$ sudo snmpwalk -v2c -cpublic localhost .1.3.6.1.2.1.14
```

---

**Enabling the .1.3.6.1.2.1 Range**

Some MIBs, including storage information, are not included by default in snmpd.conf in Cumulus Linux. This results in some default views on common network tools (like librenms) to return less than optimal data. More MIBs can be included, by enabling all the .1.3.6.1.2.1 range. This simplifies the configuration file, removing concern that any required MIBs will be missed by the monitoring system. Various MIBs were added to version 3.0 and include the following: ENTITY and ENTITY-SENSOR MIB and parts of the BRIDGE-MIB and Q-BRIDGE-MIBs. These are included in the default configuration.

The view of the BRIDGE-MIB and Q-BRIDGE-MIB are commented out.

This configuration grants access to a large number of MIBs, including all MIB2 MIBs, which could reveal more data than expected. In addition to being a security vulnerability, it could consume more CPU resources.
To enable the .1.3.6.1.2.1 range:

1. Open `/etc/snmp/snmpd.conf` in a text editor.
2. Make sure the following lines are included in the configuration:

```plaintext
# ACCESS CONTROL
#
# system
view systemonly included .1.3.6.1.2.1
# frrouting ospf6
view systemonly included .1.3.6.1.3.102
# lldpd (Note: lldpd must be restarted with the -x option
# configured in order to send info to snmpd via Agent X
view systemonly included .1.0.8802.1.1.2
# Cumulus specific
view systemonly included .1.3.6.1.4.1.40310.1
view systemonly included .1.3.6.1.4.1.40310.2
```

3. Restart `snmpd`:

```
cumulus@switch:$ sudo systemctl restart snmpd.service
```

### Configuring SNMPv3

SNMPv3 is often used to enable authentication and encryption, as community strings in versions 1 and 2c are sent in plaintext. SNMPv3 usernames are added to the `/etc/snmp/snmpd.conf` file, along with plaintext authentication and encryption pass phrases.

The NCLU command structures for configuring SNMP user passwords are:

```
cumulus@switch:$ net add snmp-server username <username> [auth-none] |
[ (auth-md5 | auth-sha) <auth-password> ]
cumulus@switch:$ net add snmp-server username <username> auth-(none | sha |
| md5) (oid <OID> | view <view>)
```

An example is shown below, defining five users, each with a different combination of authentication and encryption:
After configuring user passwords and restarting the `snmpd` daemon, the user access can be checked with a client.

The `snmp` Debian package contains `snmpget`, `snmpwalk`, and other programs that are useful for checking daemon functionality from the switch itself or from another workstation.

The following commands check the access for each user defined above from the localhost:

```bash
cumulus@switch:~$ net add snmp-server username user1 auth-none
cumulus@switch:~$ net add snmp-server username user2 auth-md5 user2password
cumulus@switch:~$ net add snmp-server username user3 auth-md5 user3password encrypt-des user3encryption
cumulus@switch:~$ net add snmp-server username user666 auth-sha user666password encrypt-aes user666encryption
cumulus@switch:~$ net add snmp-server username user999 auth-md5 user999password encrypt-des user999encryption
cumulus@switch:~$ net add snmp-server username user1 auth-none oid 1.3.6.1.2.1
cumulus@switch:~$ net add snmp-server username user1 auth-none oid system
cumulus@switch:~$ net add snmp-server username user2 auth-md5 test1234 view testview oid 1.3.6.1.2.1
cumulus@switch:~$ net add snmp-server username user3 auth-sha testshax encrypt-aes testaesx oid 1.3.6.1.2.1
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit

# simple no auth user
#createUser user1

# user with MD5 authentication
#createUser user2 MD5 user2password

# user with MD5 for auth and DES for encryption
#createUser user3 MD5 user3password DES user3encryption

# user666 with SHA for authentication and AES for encryption
createUser user666 SHA user666password AES user666encryption

# user999 with MD5 for authentication and DES for encryption
createUser user999 MD5 user999password DES user999encryption

# restrict users to certain OIDs
# (Note: creating rouser or rwuser will give
# access regardless of the createUser command above. However,
# createUser without rouser or rwuser will not provide any access).
rouser user1 noauth 1.3.6.1.2.1
rouser user2 auth 1.3.6.1.2.1
rwuser user3 priv 1.3.6.1.2.1
rwuser user666
rwuser user999

```
A slightly more secure method of configuring SNMPv3 users without creating cleartext passwords is the following:

1. Install the `net-snmp-config` script that is in `libsnmp-dev` package:

```
cumulus@switch:~$ sudo -E apt-get update
cumulus@switch:~$ sudo -E apt-get install libsnmp-dev
```

2. Stop the daemon:

```
cumulus@switch:~$ sudo systemctl stop snmpd.service
```

3. Use the `net-snmp-config` command to create two users, one with MD5 and DES, and the next with SHA and AES.

```
cumulus@switch:~$ sudo net-snmp-config --create-snmpv3-user -a md5authpass -x desprivpass -A MD5 -X DES userMD5withDES
cumulus@switch:~$ sudo net-snmp-config --create-snmpv3-user -a shaauthpass -x aesprivpass -A SHA -X AES userSHAwithAES
cumulus@switch:~$ sudo systemctl start snmpd.service
```

This adds a `createUser` command in `/var/lib/snmp/snmpd.conf`. Do not edit this file by hand, unless you are removing usernames. It also adds the `rwuser` in `/usr/share/snmp/snmpd.conf`. You may want to edit this file and restrict access to certain parts of the MIB by adding `noauth`, `auth` or `priv` to allow unauthenticated access, require authentication or to enforce use of encryption, respectively.

The `snmpd` daemon reads the information from the `/var/lib/snmp/snmpd.conf` file and then the line is removed (eliminating the storage of the master password for that user) and replaced with the key that is derived from it (using the EngineID). This key is a localized key, so that if it is...
stolen it cannot be used to access other agents. To remove the two users userMD5withDES and userSHAwithAES, you need simply stop the snmpd daemon and edit the files /var/lib/snmp/snmpd.conf and /usr/share/snmp/snmpd.conf. Simply remove the lines containing the username. Then restart the snmpd daemon as in step 3 above.

From a client, you would access the MIB with the correct credentials. (Again, note that the roles of -x, -a and -X and -A are reversed on the client side as compared with the net-snmp-config command used above.)

```
snmpwalk -v 3 -u userMD5withDES -l authPriv -a MD5 -x DES -A md5authpass -X desprivpass localhost 1.3.6.1.2.1.1.1
snmpwalk -v 3 -u userSHAwithAES -l authPriv -a SHA -x AES -A shaauthpass -X aesprivpass localhost 1.3.6.1.2.1.1.1
```

**snmpwalk a Switch from Another Linux Device**

One of the most important ways to troubleshoot is to snmpwalk the switch from another Linux device that can reach the Cumulus Linux switch.

snmpwalk does not show enterprise MIBs by default (from the 1.3.6.1.4.1 tree). These need to be explicitly named.

For this demonstration, another switch running Cumulus Linux within the network is used.

1. Open /etc/apt/sources.list in an editor.
2. Add the following line, and save the file:

   ```
   deb http://ftp.us.debian.org/debian/ jessie main non-free
   ```

3. Update the switch:

   ```
   cumulus@switch:~$ sudo -E apt-get update
   ```

4. Many SNMP clients (snmpwalk, snmpget and snmpgetnext) as well as the SNMP agent (snmpd) can benefit from having MIBs installed.

   Enabling monitoring for traps with defaultMonitors and monitor (when referring to OIDs by name) require MIBs to be installed on the switch.

   Install the snmp and snmp-mibs-downloader packages:

   ```
   cumulus@switch:~$ sudo -E apt-get install snmp snmp-mibs-downloader
   ```

5. Verify that the "mibs :" line is commented out in /etc/snmp/snmp.conf:

   ```
   #
   # As the snmp packages come without MIB files due to license reasons,
   # loading
   # of MIBs is disabled by default. If you added the MIBs you can
   # reenable
   # loading them by commenting out the following line.
   #mibs :
   ```
6. Perform an `snmpwalk` on the switch. The switch running `snmpd` in the demonstration is using IP address 192.168.0.111. It is possible to `snmpwalk` the switch from itself. Run the following command, which rules out an SNMP problem against a networking problem.

```
cumulus@switch:~$ snmpwalk -c public -v2c 192.168.0.111 .1
```

Here is some sample output:

```
IF-MIB::ifPhysAddress.2 = STRING: 74:e6:e2:f5:a2:80
IF-MIB::ifPhysAddress.3 = STRING: 0:e0:ec:25:b8:54
IF-MIB::ifPhysAddress.4 = STRING: 74:e6:e2:f5:a2:81
IF-MIB::ifPhysAddress.5 = STRING: 74:e6:e2:f5:a2:82
IF-MIB::ifPhysAddress.6 = STRING: 74:e6:e2:f5:a2:83
IF-MIB::ifPhysAddress.7 = STRING: 74:e6:e2:f5:a2:84
IF-MIB::ifPhysAddress.8 = STRING: 74:e6:e2:f5:a2:85
IF-MIB::ifPhysAddress.9 = STRING: 74:e6:e2:f5:a2:86
IF-MIB::ifPhysAddress.10 = STRING: 74:e6:e2:f5:a2:87
IF-MIB::ifPhysAddress.13 = STRING: 74:e6:e2:f5:a2:8a
IF-MIB::ifPhysAddress.15 = STRING: 74:e6:e2:f5:a2:8c
IF-MIB::ifPhysAddress.16 = STRING: 74:e6:e2:f5:a2:8d
IF-MIB::ifPhysAddress.17 = STRING: 74:e6:e2:f5:a2:8e
IF-MIB::ifPhysAddress.18 = STRING: 74:e6:e2:f5:a2:8f
```

Any information gathered here should verify that `snmpd` is running correctly on the Cumulus Linux side, reducing locations where a problem may reside.

**Troubleshooting Tips Table for snmpwalks**

<table>
<thead>
<tr>
<th>Run snmpwalk from</th>
<th>If it works</th>
<th>If it does not work</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>switch</strong> (switch to monitor)</td>
<td><code>snmpd</code> is serving information correctly.</td>
<td>Is <code>snmpd</code> misconfigured or installed incorrectly?</td>
</tr>
<tr>
<td></td>
<td>The problem resides somewhere else. For example, network connectivity, or Prism misconfiguration.</td>
<td></td>
</tr>
<tr>
<td><strong>switch2</strong> (another Cumulus Linux switch in the network)</td>
<td><code>snmpd</code> is serving information correctly and network reachability works between <code>switch</code> and <code>switch2</code>.</td>
<td>Network connectivity is not able to grab information? Is there an <code>iptables</code> rule blocking? Is the <code>snmpwalk</code> being run correctly?</td>
</tr>
<tr>
<td></td>
<td>The problem resides somewhere else. For example, Prism cannot reach <code>switch</code>, or there is a Prism misconfiguration.</td>
<td></td>
</tr>
<tr>
<td><strong>Nutanix Prism CLI</strong> (SSH to the cluster IP address)</td>
<td><code>snmpd</code> is serving information correctly and network reachability works between <code>switch</code> and the Nutanix Appliance.</td>
<td>Is the right community name being used in the GUI? Is <code>snmp v2c</code> being used?</td>
</tr>
<tr>
<td></td>
<td>The problem resides somewhere else. For example, the GUI might be misconfigured.</td>
<td></td>
</tr>
</tbody>
</table>

**SNMP Traps**

**Generating Event Notification Traps**

The Net-SNMP agent provides a method to generate SNMP trap events via the Distributed Management (DisMan) Event MIB for various system events, including:
- Link up/down.
- Exceeding the temperature sensor threshold, CPU load or memory threshold.
- Other SNMP MIBs.

In order to enable specific types of traps, you need to create the following configurations in `/etc/snmp/snmpd.conf`.

### Defining Access Credentials

An SNMPv3 username is required to authorize the DisMan service even though you are not configuring SNMPv3 here. The example `snmpd.conf` configuration shown below creates `trapusername` as the username using the `createUser` command. `iquerySecName` defines the default SNMPv3 username to be used when making internal queries to retrieve monitored expressions. `rouser` specifies which username should be used for these SNMPv3 queries. All three are required for `snmpd` to retrieve information and send traps (even with the `monitor` command shown below in the examples). Add the following lines to your `/etc/snmp/snmpd.conf` configuration file:

```
createUser trapusername
iquerySecName trapusername
rouser trapusername
```

`iquerySecName` specifies the default SNMPv3 username to be used when making internal queries to retrieve any necessary information — either for evaluating the monitored expression or building a notification payload. These internal queries always use SNMPv3, even if normal querying of the agent is done using SNMPv1 or SNMPv2c. Note that this user must also be explicitly created via `createUser` and given appropriate access rights, for `rouser`, for example. The `iquerySecName` directive is purely concerned with defining which user should be used, not with actually setting this user up.

### Defining Trap Receivers

The following configuration defines the trap receiver IP address where SNMPv2 traps are sent:

```
trap2sink 192.168.1.1 public
    # For SNMPv1 Traps, use
    # trapsink  192.168.1.1  public
```

Although the traps are sent to an SNMPV2 receiver, the SNMPv3 user is still required. Starting with Net-SNMP 5.3, `snmptrapd` no longer accepts all traps by default. `snmptrapd` must be configured with authorized SNMPv1/v2c community strings and/or SNMPv3 users. Non-authorized traps/informs will be dropped. Please refer to the `snmptrapd.conf(5)` manual page for details.

### SNMP Version 3 TRAP and INFORM Messages

You can configure SNMPv3 trap and inform messages with the `trapsess` configuration command. Inform messages are traps that are acknowledged by the receiving trap daemon. You configure inform messages with the `-Ci` parameter. You must specify the EngineID of the receiving trap server with the `-e` field.

```
SNMPv3 TRAP/INFORM
trapsess  -Ci  -e 0x80ccff112233445566778899  -v3  -l authPriv  -u trapuser1  -a MD5  -A  trapuser1password  -x DES  -X  trapuser1encryption 192.168.1.1
```

The SNMP trap receiving daemon must have usernames, authentication passwords, and encryption passwords created with its own EngineID. You must configure this trap server EngineID in the switch `snmpd` daemon sending the trap and inform messages. You specify the level of authentication and encryption for SNMPv3 trap and inform messages with `-l` (NoauthNoPriv, authNoPriv, or authPriv).

You can define multiple trap receivers and use the domain name instead of an IP address in the `trap2sink` directive.
After you complete the configuration, restart the `snmpd` service to apply the changes.

```
cumulus@switch:~$ sudo systemctl restart snmpd.service
```

### Sourcing Traps from a Different Source IP Address

When client SNMP programs (such as `snmpget`, `snmpwalk` or `snmptrap`) are run from the command line, or when `snmpd` is configured to send a trap (based on `snmpd.conf`), you can configure a `clientaddr` in `snmp.conf` that allow the SNMP client programs or `snmpd` (for traps) to source requests from a different source IP address.

```
clientaddr [<transport-specifier>:]<transport-address>
```

specifies the source address to be used by command-line applications when sending SNMP requests. See `snmpcmd(1)` for more information about the format of addresses. This value is also used by `snmpd` when generating notifications.

For more information, read the the `snmp.conf` man page:

```
monitor [OPTIONS] NAME EXPRESSION
```

defines a MIB object to monitor. If the EXPRESSION condition holds then this will trigger the corresponding event, and either send a notification or apply a SET assignment (or both). Note that the event will only be triggered once, when the expression first matches. This monitor entry will not fire again until the monitored condition first becomes false, and then matches again. `NAME` is an administrative name for this expression, and is used for indexing the mteTriggerTable (and related tables). Note also that such monitors use an internal SNMPv3 request to retrieve the values being monitored (even if normal agent queries typically use SNMPv1 or SNMPv2c).

### Monitoring Fans, Power Supplies and Transformers

The usual behavior of an SNMP agent (`snmpd`) is to wait for incoming SNMP requests and respond to them. If no requests are received, an agent will typically not initiate any actions. However, various commands can configure `snmpd` to send traps based on preconfigured settings (load, file, proc, disk or swap commands) or customized monitor commands.

From the `snmpd.conf` man page, the `monitor` command is defined this way:
See the `iquerySecName` token described above.

**EXPRESSION**

There are three types of monitor expression supported by the Event MIB - existence, boolean and threshold tests.

OID | ! OID | != OID

- OID | ! OID | != OID defines an existence(0) monitor test. A bare OID specifies a present(0) test,
- which will fire when (an instance of) the monitored OID is created. An expression of the form ! OID specifies an absent(1) test, which will fire when the monitored OID is deleted. An expression of the form != OID specifies a changed(2) test,
- which will fire whenever the monitored value(s) change. Note that there must be whitespace before the OID token.

OID OP VALUE

- OID OP VALUE defines a boolean(1) monitor test. OP should be one of the defined comparison operators (!=, ==, <, <=, >, >=) and VALUE should be an integer value to compare against. Note that there must be whitespace around the OP token. A comparison such as OID != 0 will not be handled correctly.

OID MIN MAX [DMIN DMAX]

- OID MIN MAX [DMIN DMAX] defines a threshold(2) monitor test. MIN and MAX are integer values, specifying lower and upper thresholds. If the value of the monitored OID falls below the lower threshold (MIN) or rises above the upper threshold (MAX), then the monitor entry will trigger the corresponding event.

- Note that the rising threshold event will only be re-armed when the monitored value falls below the lower threshold (MIN). Similarly, the falling threshold event will be re-armed by the upper threshold (MAX).

- The optional parameters DMIN and DMAX configure a pair of similar threshold tests, but working with the delta differences between successive sample values.

OPTIONS
There are various options to control the behaviour of the monitored expression. These include:

- **-D** indicates that the expression should be evaluated using delta differences between sample values (rather than the values themselves).
- **-d OID** or **-di OID**
  specifies a discontinuity marker for validating delta differences. A -di object instance will be used exactly as given. A -d object will have the instance subidentifiers from the corresponding (wildcarded) expression object appended. If the -I flag is specified, then there is no difference between these two options. This option also implies -D.
- **-e EVENT**
  specifies the event to be invoked when this monitor entry is triggered. If this option is not given, the monitor entry will generate one of the standard notifications defined in the DISMAN-EVENT-MIB.
- **-I** indicates that the monitored expression should be applied to the specified OID as a single instance. By default, the OID will be treated as a wildcarded object, and the monitor expanded to cover all matching instances.
- **-i OID** or **-o OID**
  define additional varbinds to be added to the notification payload when this monitor trigger fires. For a wildcarded expression, the suffix of the matched instance will be added to any OIDs specified using -o, while OIDs specified using -i will be treated as exact instances. If the -I flag is specified, then there is no difference between these two options. See strictDisman for details of the ordering of notification payloads.
- **-r FREQUENCY**
  monitors the given expression every FREQUENCY, where FREQUENCY is in seconds or optionally suffixed by one of s (for seconds), m (for minutes), h (for hours), d (for days), or w (for weeks). By default, the expression will be evaluated every 600s (10 minutes).
- **-S** indicates that the monitor expression should not be evaluated when the agent first starts up. The first evaluation will be done once the first repeat interval has expired.
- **-s** indicates that the monitor expression should be evaluated when the agent first starts up. This is the default behaviour.

Note: Notifications triggered by this initial evaluation will be sent before the coldStart trap.
-u SECNAME
    specifies a security name to use for scanning
local host, instead of the default iquerySecName. Once again, this user must be explicitly created and given suitable access rights.

\texttt{snmpd} can be configured to monitor the operational status of an Entity MIB or Entity-Sensor MIB. The operational status, given as a value of \texttt{ok(1)}, \texttt{unavailable(2)} or \texttt{nonoperational(3)}, can be determined by adding the following example configuration to \texttt{/etc/snmp/snmpd.conf}, and adjusting the values:

- **Using the \texttt{entPhySensorOperStatus} integer:**

\begin{verbatim}
# without installing extra MIBS we can check the check Fan1 status
# if the Fan1 index is 100011001, monitor this specific OID (-I) every
10 seconds (-r), and defines additional information to be included in
the trap (-o).
monitor -I -r 10 -o 1.3.6.1.2.1.47.1.1.1.1.7.100011001 "Fan1 Not OK"
1.3.6.1.2.1.99.1.1.1.5.100011001 > 1
# Any Entity Status non OK (greater than 1)
monitor -r 10 -o 1.3.6.1.2.1.47.1.1.1.1.7 "Sensor Status Failure"
1.3.6.1.2.1.99.1.1.1.5 > 1
\end{verbatim}

- **Using the OID name:**

\begin{verbatim}
# for a specific fan called Fan1 with an index 100011001
monitor -I -r 10 -o entPhysicalName.100011001 "Fan1 Not OK"
entPhySensorOperStatus.100011001 > 1
# for any Entity Status not OK (greater than 1)
monitor -r 10 -o entPhysicalName "Sensor Status Failure"
entPhySensorOperStatus > 1
\end{verbatim}

The OID name can be used if the \texttt{snmp-mibs-download} package is installed.

The \texttt{entPhySensorOperStatus} integer can be found by walking the \texttt{entPhysicalName} table.

- **To get all sensor information, run \texttt{snmpwalk} on the \texttt{entPhysicalName} table. For example:**

\begin{verbatim}
# for a specific fan called Fan1 with an index 100011001
snmpwalk -v 2c -c public 192.168.1.1/16 entPhysicalName
# for any Entity Status not OK (greater than 1)
snmpwalk -v 2c -c public 192.168.1.1/16 entPhysicalName
\end{verbatim}
cumulus@leaf01:~$ snmpwalk -v 2c -cpublic localhost
.1.3.6.1.2.1.47.1.1.1.1.7
iso.3.6.1.2.1.47.1.1.1.1.7.100000001 = STRING: "PSU1Temp1"
iso.3.6.1.2.1.47.1.1.1.1.7.100000002 = STRING: "PSU2Temp1"
iso.3.6.1.2.1.47.1.1.1.1.7.100000003 = STRING: "Temp1"
iso.3.6.1.2.1.47.1.1.1.1.7.100000004 = STRING: "Temp2"
iso.3.6.1.2.1.47.1.1.1.1.7.100000005 = STRING: "Temp3"
iso.3.6.1.2.1.47.1.1.1.1.7.100000006 = STRING: "Temp4"
iso.3.6.1.2.1.47.1.1.1.1.7.100000007 = STRING: "Temp5"
iso.3.6.1.2.1.47.1.1.1.1.7.100011001 = STRING: "Fan1"
iso.3.6.1.2.1.47.1.1.1.1.7.100011002 = STRING: "Fan2"
iso.3.6.1.2.1.47.1.1.1.1.7.100011003 = STRING: "Fan3"
iso.3.6.1.2.1.47.1.1.1.1.7.100011004 = STRING: "Fan4"
iso.3.6.1.2.1.47.1.1.1.1.7.100011005 = STRING: "Fan5"
iso.3.6.1.2.1.47.1.1.1.1.7.100011006 = STRING: "Fan6"
iso.3.6.1.2.1.47.1.1.1.1.7.100011007 = STRING: "PSU1Fan1"
iso.3.6.1.2.1.47.1.1.1.1.7.100011008 = STRING: "PSU2Fan1"
iso.3.6.1.2.1.47.1.1.1.1.7.110000001 = STRING: "PSU1"
iso.3.6.1.2.1.47.1.1.1.1.7.110000002 = STRING: "PSU2"

Enabling MIB to OID Translation

MIB names can be used instead of OIDs, by installing the `snmp-mibs-downloader`, to download SNMP MIBs to the switch prior to enabling traps. This greatly improves the readability of the `snmpd.conf` file.

1. Open `/etc/apt/sources.list` in a text editor.
2. Add the `non-free` repository, and save the file:

```bash
cumulus@switch:~$ sudo deb http://ftp.us.debian.org/debian/ jessie main non-free
```

3. Update the switch:

```bash
cumulus@switch:~$ sudo -E apt-get update
```

4. Install the `snmp-mibs-downloader`:

```bash
cumulus@switch:~$ sudo -E apt-get install snmp-mibs-downloader
```

5. Open the `/etc/snmp/snmp.conf` file to verify that the `mibs` line is commented out:
# As the snmp packages come without MIB files due to license reasons, loading # of MIBs is disabled by default. If you added the MIBs you can reenable # loading them by commenting out the following line.
#mibs :

6. Open the `/etc/default/snmpd` file to verify that the `export MIBS=` line is commented out:

```
# This file controls the activity of snmpd and snmptrapd
# Don't load any MIBs by default.
# You might comment this lines once you have the MIBs Downloaded.
#export MIBS=
```

7. Once the configuration has been confirmed, remove or comment out the `non-free` repository in `/etc/apt/sources.list`.

```
#deb http://ftp.us.debian.org/debian/ jessie main non-free
```

### Configuring Link Up/Down Notifications

The `linkUpDownNotifications` directive is used to configure link up/down notifications when the operational status of the link changes.

```
linkUpDownNotifications yes
```

The default frequency for checking link up/down is 60 seconds. The default frequency can be changed using the `monitor` directive directly instead of the `linkUpDownNotifications` directive. See `man snmpd.conf` for details.

### Configuring Temperature Notifications

Temperature sensor information for each available sensor is maintained in the `ImSensors` MIB. Each platform may contain a different number of temperature sensors. The example below generates a trap event when any temperature sensors exceeds a threshold of 68 degrees (centigrade). It monitors each `lmTempSensorsValue`. When the threshold value is checked and exceeds the `lmTempSensorsValue`, a trap is generated. The `-o lmTempSensorsDevice` option is used to instruct SNMP to also include the `lmTempSensorsDevice MIB` in the generated trap. The default frequency for the `monitor` directive is 600 seconds. The default frequency may be changed using the `-r` option:

```
monitor lmTemSensor -o lmTempSensorsDevice lmTempSensorsValue > 68000
```

Alternatively, temperature sensors may be monitored individually. To monitor the sensors individually, first use the `sensors` command to determine which sensors are available to be monitored on the platform.
cumulus@switch:~$ sudo sensors

CY8C3245-i2c-4-2e
Adapter: i2c-0-mux (chan_id 2)
fan5: 7006 RPM (min = 2500 RPM, max = 23000 RPM)
fan6: 6955 RPM (min = 2500 RPM, max = 23000 RPM)
fan7: 6799 RPM (min = 2500 RPM, max = 23000 RPM)
fan8: 6750 RPM (min = 2500 RPM, max = 23000 RPM)
temp1: +34.0 C (high = +68.0 C)
temp2: +28.0 C (high = +68.0 C)
temp3: +33.0 C (high = +68.0 C)
temp4: +31.0 C (high = +68.0 C)
temp5: +23.0 C (high = +68.0 C)

Configure a `monitor` command for the specific sensor using the `-I` option. The `-I` option indicates that the monitored expression is applied to a single instance. In this example, there are five temperature sensors available. The following monitor directive can be used to monitor only temperature sensor three at five minute intervals.

```
monitor -I -r 300 lmTemSensor3 -o lmTempSensorsDevice.3
lmTempSensorsValue.3 > 68000
```

### Configuring Free Memory Notifications

You can monitor free memory using the following directives. The example below generates a trap when free memory drops below 1,000,000KB. The free memory trap also includes the amount of total real memory:

```
monitor MemFreeTotal -o memTotalReal memTotalFree < 1000000
```

### Configuring Processor Load Notifications

To monitor CPU load for 1, 5 or 15 minute intervals, use the `load` directive in conjunction with the `monitor` directive. The following example will generate a trap when the 1 minute interval reaches 12%, the 5 minute interval reaches 10% or the 15 minute interval reaches 5%.

```
load 12 10 5
```

### Configuring Disk Utilization Notifications

To monitor disk utilization for all disks, use the `includeAllDisks` directive in conjunction with the `monitor` directive. The example code below generates a trap when a disk is 99% full:

```
includeAllDisks 1%
monitor -r 60 -o dskPath -o DiskErrMsg "dskTable" diskErrorFlag !=0
```

### Configuring Authentication Notifications
To generate authentication failure traps, use the `authtrapenable` directive:

```
authtrapenable 1
```

**snmptrapd.conf**

To receive SNMP traps, the Net-SNMP trap daemon can be used on the switch. The configuration file, `/etc/snmp/snmptrapd.conf`, is used to configure how incoming traps should be processed. Starting with release 5.3, it is necessary to explicitly specify who is authorized to send traps and informs to the notification receiver (and what types of processing these are allowed to trigger). There are currently three types of processing that can be specified:

- **log**: Logs the details of the notification, in a specified file, to standard output (or stderr), or via syslog (or similar).
- **execute**: Passes the details of the trap to a specified handler program, including embedded Perl.
- **net**: Forwards the trap to another notification receiver.

Most commonly, this configuration typically is `log,execute,net` to cover any style of processing for a particular category of notification. But it is possible (even desirable) to limit certain notification sources to selected processing only.

```
authCommunity TYPES COMMUNITY [SOURCE [OID | -v VIEW ]] authorizes traps and SNMPv2c INFORM requests with the specified community to trigger the types of processing listed. By default, this allows any notification using this community to be processed. The SOURCE field can be used to specify that the configuration should only apply to notifications received from particular sources. For more information about specific configuration options within the file, look at the `snmpd.conf(5)` man page with the following command:
```

cumulus@switch:~$ man 5 snmptrapd.conf
```
**Supported MIBs**

Below are the MIBs supported by Cumulus Linux, as well as suggested uses for them. The overall Cumulus Linux MIB is defined in `/usr/share/snmp/mibs/Cumulus-Snmp-MIB.txt`.

<table>
<thead>
<tr>
<th>MIB Name</th>
<th>Suggested Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRIDGE and Q-BRIDGE</td>
<td>The dot1dBasePortEntry and dot1dBasePortIfIndex tables in the BRIDGE-MIB and dot1qBase, dot1qFdbEntry, dot1qTpFdbEntry, dot1qTpFdbStatus, and the dot1qVlanStaticName tables in the Q-BRIDGE-MIB tables. You must uncomment the <code>bridge_pp.py</code> <code>pass_persist</code> script in <code>/etc/snmp/snmpd.conf</code>.</td>
</tr>
<tr>
<td>BGP4, OSPF, OSPFV3, RIPv2</td>
<td>FRRouting SNMP support may be enabled to provide support for OSPF-MIB (RFC-1850), OSPFV3-MIB (RFC-5643), and BGP4-MIB (RFC-4273). To enable this support, see the FRRouting section above.</td>
</tr>
<tr>
<td>CUMULUS-COUNTERS-MIB</td>
<td>Discard counters: Cumulus Linux also includes its own counters MIB, defined in <code>/usr/share/snmp/mibs/Cumulus-Counters-MIB.txt</code>. It has the OID <code>.1.3.6.1.4.1.40310.2</code>.</td>
</tr>
<tr>
<td>CUMULUS-RESOURCE-QUERY-MIB</td>
<td>Cumulus Linux includes its own resource utilization MIB, which is similar to using <code>cl-resource-query</code>. It monitors L3 entries by host, route, nexthops, ECMP groups and L2 MAC/BPU entries. The MIB is defined in <code>/usr/share/snmp/mibs/Cumulus-Resource-Query-MIB.txt</code>, and has the OID <code>.1.3.6.1.4.1.40310.1</code>.</td>
</tr>
<tr>
<td>CUMULUS-POE-MIB</td>
<td>The Cumulus Networks custom Power over Ethernet PoE MIB defined in <code>/usr/share/snmp/mibs/Cumulus-POE-MIB.txt</code>. For devices that provide PoE, this provides users with the system wide power information in <code>poeSystemValues</code> as well as per interface <code>poeObjectsEntry</code> values for the <code>poeObjectsTable</code>. Most of this information comes from the <code>poectl</code> command. This MIB is enabled by uncommenting the following line in <code>/etc/snmp/snmpd.conf</code>:</td>
</tr>
<tr>
<td></td>
<td><code>#pass_persist .1.3.6.1.4.1.40310.3</code></td>
</tr>
<tr>
<td></td>
<td><code>/usr/share/snmp/cl_poe_pp.py</code></td>
</tr>
<tr>
<td>DISMAN-EVENT</td>
<td>Trap monitoring</td>
</tr>
<tr>
<td>ENTITY</td>
<td>From RFC 4133, the temperature sensors, fan sensors, power sensors, and ports are covered.</td>
</tr>
<tr>
<td>ENTITY-SENSOR</td>
<td>Physical sensor information (temperature, fan, and power supply) from RFC 3433.</td>
</tr>
<tr>
<td>HOST-RESOURCES</td>
<td>Users, storage, interfaces, process info, run parameters</td>
</tr>
<tr>
<td>IEEE8023-LAG-MIB</td>
<td>Implementation of the IEEE 8023-LAG-MIB includes the dot3adAggTable and dot3adAggPortListTable tables. To enable this, edit <code>/etc/snmp/snmpd.conf</code> and uncomment or add the following lines:</td>
</tr>
<tr>
<td></td>
<td><code>view systemonly included .1.2.840.10006.300.43</code></td>
</tr>
<tr>
<td></td>
<td><code>pass_persist .1.2.840.10006.300.43</code></td>
</tr>
<tr>
<td></td>
<td><code>/usr/share/snmp/ieee8023_lag_pp.py</code></td>
</tr>
<tr>
<td>IF-MIB</td>
<td>Interface description, type, MTU, speed, MAC, admin, operation status, counters</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>The IF-MIB cache is disabled by default. To enable the counter to reflect traffic statistics, remove the <code>-y</code> option from the <code>SNMPDOPTS</code> line in the <code>/etc/default/snmpd</code> file. The example below first shows the original line, commented out, then the modified line without the <code>-y</code> option:</td>
</tr>
<tr>
<td></td>
<td>cumulus@switch:~$ cat /etc/default/snmpd</td>
</tr>
<tr>
<td></td>
<td># SNMPDOPTS='-y -LS 0-4 d -Lf /dev/null -u snmp -g snmp -I -smux -p /run/snmpd.pid'</td>
</tr>
<tr>
<td></td>
<td>SNMPDOPTS='-LS 0-4 d -Lf /dev/null -u snmp -g snmp -I -smux -p /run/snmpd.pid'</td>
</tr>
<tr>
<td>IP (includes ICMP)</td>
<td>IPv4, IPv4 addresses, counters, netmasks</td>
</tr>
<tr>
<td>IPv6</td>
<td>IPv6 counters</td>
</tr>
<tr>
<td>IP-FORWARD</td>
<td>IP routing table</td>
</tr>
<tr>
<td>LLDP</td>
<td>L2 neighbor info from <code>lldpd</code> (note, you need to enable the SNMP subagent in LLDP). <code>lldpd</code> needs to be started with the <code>-x</code> option to enable connectivity to <code>snmpd</code> (AgentX).</td>
</tr>
<tr>
<td>LM-SENSORS MIB</td>
<td>Fan speed, temperature sensor values, voltages. This is deprecated since the ENTITY-SENSOR MIB has been added.</td>
</tr>
<tr>
<td>NET-SNMP-AGENT</td>
<td>Agent timers, user, group config</td>
</tr>
<tr>
<td>NET-SNMP-EXTEND</td>
<td>Agent timers, user, group config</td>
</tr>
<tr>
<td>NET-SNMP-EXTEND-MIB</td>
<td>See this knowledge base article on extending NET-SNMP in Cumulus Linux to include data from power supplies, fans and temperature sensors.</td>
</tr>
<tr>
<td>NET-SNMP-VACM</td>
<td>Agent timers, user, group config</td>
</tr>
<tr>
<td>NOTIFICATION-LOG</td>
<td>Local logging</td>
</tr>
<tr>
<td>SNMP-FRAMWORK</td>
<td>Users, access</td>
</tr>
<tr>
<td>SNMP-MPD</td>
<td>Users, access</td>
</tr>
<tr>
<td>SNMP-TARGET</td>
<td>Users, access</td>
</tr>
<tr>
<td>SNMP-USER-BASED-SM</td>
<td>Users, access</td>
</tr>
<tr>
<td>SNMP-VIEW-BASED-ACM</td>
<td>Users, access</td>
</tr>
<tr>
<td>SNMPv2</td>
<td>SNMP counters. For information on exposing CPU and memory information via SNMP, see this knowledge base article.</td>
</tr>
<tr>
<td>TCP</td>
<td>TCP related information</td>
</tr>
<tr>
<td>UCD-SNMP</td>
<td>System memory, load, CPU, disk IO</td>
</tr>
<tr>
<td>UDP</td>
<td>UDP related information</td>
</tr>
<tr>
<td></td>
<td>The ENTITY MIB does not currently show the chassis information in Cumulus Linux.</td>
</tr>
</tbody>
</table>

### About Pass Persist Scripts

The pass persist scripts in Cumulus Linux use the `pass_persist extension` to Net-SNMP. The scripts are stored in `/usr/share/snmp` and include:
All the scripts are enabled by default in Cumulus Linux, except for the `bgp4_pp.py` and `cl_poe_pp.py` scripts:

- `bgp4_pp.py` is now handled by FRRouting instead of Quagga, so monitoring has changed accordingly.
- `cl_poe_pp.py` is disabled by default as only some platforms that Cumulus Linux supports are capable of doing Power over Ethernet.

**Troubleshooting**

The following commands can be used to troubleshoot potential SNMP issues:

```bash
cumulus@switch:~$ net show snmp-server status

---------------------------------
Current Status                     failed (failed)
Reload Status                      enabled
Listening IP Addresses             localhost 9.9.9.9
Main snmpd PID                     0
Version 1 and 2c Community String  Configured
Version 3 Usernames                Not Configured
Last Logs (with Errors)            -- Logs begin at Thu 2017-08-03 16:23:05 UTC, end at Fri 2017-08-04 18:17:24 UTC. --
Aug 04 18:17:19 cel-redxp-01
snmpd[8389]: Error opening specified endpoint "9.9.9.9"
Aug 04 18:17:19 cel-redxp-01
snmpd[8389]: Server Exiting with code 1
---------------------------------
```

```bash
cumulus@switch:~$ net show configuration snmp-server

snmp-server
listening-address 127.0.0.1
readonly-community public access default
readonly-community allpass access any
readonly-community temp2 access 1.1.1.1
readonly-community temp2 access 2.2.2.2
trap-destination 1.1.1.1 community-password public version 2c
trap-link-up check-frequency 10
trap-snmp-auth-failures
```
cumulus@switch:~$ net show configuration commands
...
net add snmp-server listening-address all
net add snmp-server readonly-community allpass access any
net add snmp-server readonly-community temp2 access 1.1.1.1
net add snmp-server readonly-community temp2 access 2.2.2.2
net add snmp-server trap-destination 1.1.1.1 community-password public version 2c
net add snmp-server trap-link-up check-frequency 10
net add snmp-server trap-snmp-auth-failures
...