# Cumulus NetQ Integration Guide

# Table of Contents

# Cumulus NetQ Integration Guide

After you have completed the installation of Cumulus NetQ, you may want to configure some of the additional capabilities that NetQ offers or integrate it with third-party software or hardware.

This topic describes how to:

- integrate and configure NetQ event notifications with popular notification applications
- integrate with customer applications using the Cumulus NetQ API
- integrate with a hardware chassis

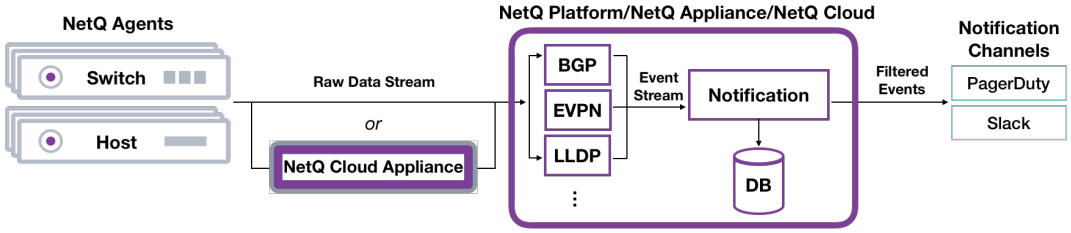
# Integrate NetQ with Notification Applications

After you have installed NetQ applications package and the NetQ Agents, you may want to configure some of the additional capabilities that NetQ offers. This topic describes how to install, setup, and configure these capabilities.

## Integrate NetQ with an Event Notification Application

To take advantage of the numerous event messages generated and processed by NetQ, you must integrate with third-party event notification applications. You can integrate NetQ with the PagerDuty and Slack tools. You may integrate with one or both of these applications.

Each network protocol and service in the NetQ Platform receives the raw data stream from the NetQ Agents, processes the data and delivers events to the Notification function. Notification then stores, filters and sends messages to any configured notification applications. Filters are based on rules you create. You must have at least one rule per filter.



> ⓘ NOTE
>
> You may choose to implement a proxy server (that sits between the NetQ Platform and the integration channels) that receives, processes and distributes the notifications rather than having them sent directly to the integration channel. If you use such a proxy, you must configure NetQ with the proxy information.

In either case, notifications are generated for the following types of events:

- Network Protocols
  - BGP status and session state
  - CLAG (MLAG) status and session state
  - EVPN status and session state
  - LLDP status
  - LNV status and session state *
  - OSFP status and session state
  - VLAN status and session state *
  - VXLAN status and session state *
- Interfaces
  - Link status
  - Ports and cables status
- Services status
  - NetQ Agent status
  - PTM
  - SSH *
  - NTP status *
- Trace status

- Sensors
  - Fan status
  - PSU (power supply unit) status
  - Temperature status
- System
  - Configuration File changes
  - Cumulus Linux License status
  - Cumulus Linux Support status

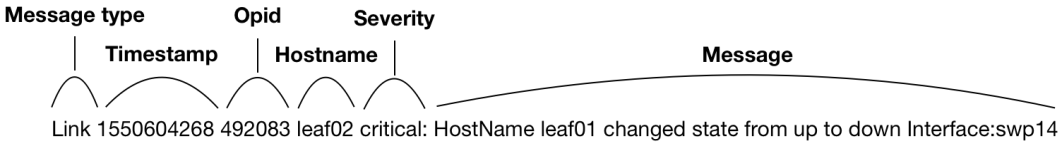
** This type of event can only be viewed in the CLI with this release.*

### Event Message Format

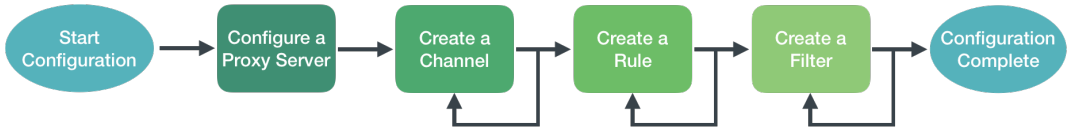
Messages have the following structure: `<message-type><timestamp><opid><hostname><severity><message>`

| Element | Description |
|---|---|
| message type | Category of event; *bgp*, *clag*, *configdiff*, *evpn*, *link*, *lldp*, *lnv*, *node*, *sensor*, *services*, *trace*, *vlan* or *vxlan* |
| timestamp | Date and time event occurred |
| opid | Identifier of the service or process that generated the event |
| hostname | Hostname of network device where event occurred |
| severity | Severity level in which the given event is classified; *debug*, *error*, *i* *critical* |
| message | Text description of event |

For example:



To set up the integrations, you must configure NetQ with at least one channel. Optionally, you can define rules and filters to refine what messages you want to view and where to send them. You can also configure a proxy server to receive, process, and forward the messages. This is accomplished using the NetQ CLI in the following order:




## Notification Commands Overview

The NetQ Command Line Interface (CLI) is used to filter and send notifications to third-party tools based on severity, service, event-type, and device. You can use TAB completion or the `help` option to assist when needed. The command syntax is:

```
##Proxy
netq add notification proxy <text-proxy-hostname> [port <text-proxy-port>]
netq show notification proxy
netq del notification proxy

##Channels
netq add notification channel slack <text-channel-name> webhook <text-webhook-url> [severity info|severity warning|severity error|severity debug] [tag <text-slack-tag>]
netq add notification channel pagerduty <text-channel-name> integration-key <text-integration-key> [severity info|severity warning|severity error|severity debug]

##Rules and Filters
netq add notification rule <text-rule-name> key <text-rule-key> value <text-rule-value>
```

```
netq add notification filter <text-filter-name> [severity info|severity warning|
severity error|severity debug] [rule <text-rule-name-anchor>] [channel <text-
channel-name-anchor>] [before <text-filter-name-anchor>|after <text-filter-name-
anchor>]

##Management
netq del notification channel <text-channel-name-anchor>
netq del notification filter <text-filter-name-anchor>
netq del notification rule <text-rule-name-anchor>
netq show notification [channel|filter|rule] [json]
```

The options are described in the following sections where they are used.

### Configure a Proxy Server

To send notification messages through a proxy server instead of directly to a notification channel, you configure NetQ with the hostname and optionally a port of a proxy server. If no port is specified, NetQ defaults to port 80. Only one proxy server is currently supported. To simplify deployment, configure your proxy server before configuring channels, rules, or filters.To configure the proxy server:

```
cumulus@switch:~$ netq add notification proxy <text-proxy-hostname> [port
<text-proxy-port]
cumulus@switch:~$ netq add notification proxy proxy4
Successfully configured notifier proxy proxy4:80
```

You can view the proxy server settings by running the `netq show notification proxy` command.

```
cumulus@switch:~$ netq show notification proxy
Matching config_notify records:
Proxy URL        Slack Enabled            PagerDuty Enabled
---------------- ------------------------ ---------------------------------
proxy4:80        yes                      yes
```

You can remove the proxy server by running the `netq del notification proxy` command. This changes the NetQ behavior to send events directly to the notification channels.

```
cumulus@switch:~$ netq del notification proxy
Successfully overwrote notifier proxy to null
```

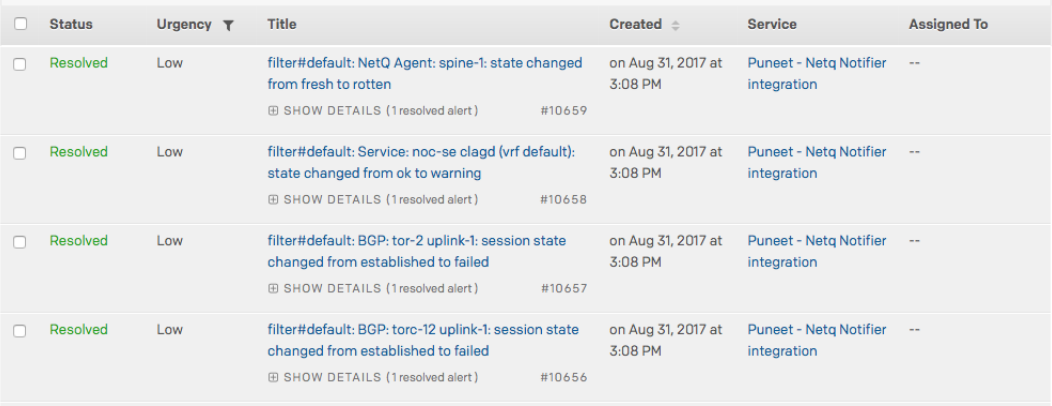
### Create Channels

Create one or more PagerDuty and Slack channels to present the notifications.

#### CONFIGURE A PAGERDUTY CHANNEL

NetQ sends notifications to PagerDuty as PagerDuty events.

For example:



| Status | Urgency | Title | Created | Service | Assigned To |
|---|---|---|---|---|---|
| Resolved | Low | filter#default: NetQ Agent: spine-1: state changed from fresh to rotten<br>SHOW DETAILS (1 resolved alert) #10659 | on Aug 31, 2017 at 3:08 PM | Puneet - Netq Notifier integration | -- |
| Resolved | Low | filter#default: Service: noc-se clagd (vrf default): state changed from ok to warning<br>SHOW DETAILS (1 resolved alert) #10658 | on Aug 31, 2017 at 3:08 PM | Puneet - Netq Notifier integration | -- |
| Resolved | Low | filter#default: BGP: tor-2 uplink-1: session state changed from established to failed<br>SHOW DETAILS (1 resolved alert) #10657 | on Aug 31, 2017 at 3:08 PM | Puneet - Netq Notifier integration | -- |
| Resolved | Low | filter#default: BGP: torc-12 uplink-1: session state changed from established to failed<br>SHOW DETAILS (1 resolved alert) #10656 | on Aug 31, 2017 at 3:08 PM | Puneet - Netq Notifier integration | -- |

To configure the NetQ notifier to send notifications to PagerDuty:

1. Configure the following options using the `netq add notification channel` command:

| Option | Description |
|---|---|
| CHANNEL_TYPE <text-channel-name> | The third-party notification channel and name; use *pagerduty* in this case. |
| integration-key <text-integration-key> | The integration key is also called the service_key or routing_key. The default is an empty string (“”). |
| severity | (Optional) The log level to set, which can be one of *info*, *warning*, *error*, *critical* or *debug*. The severity defaults to *info*. |

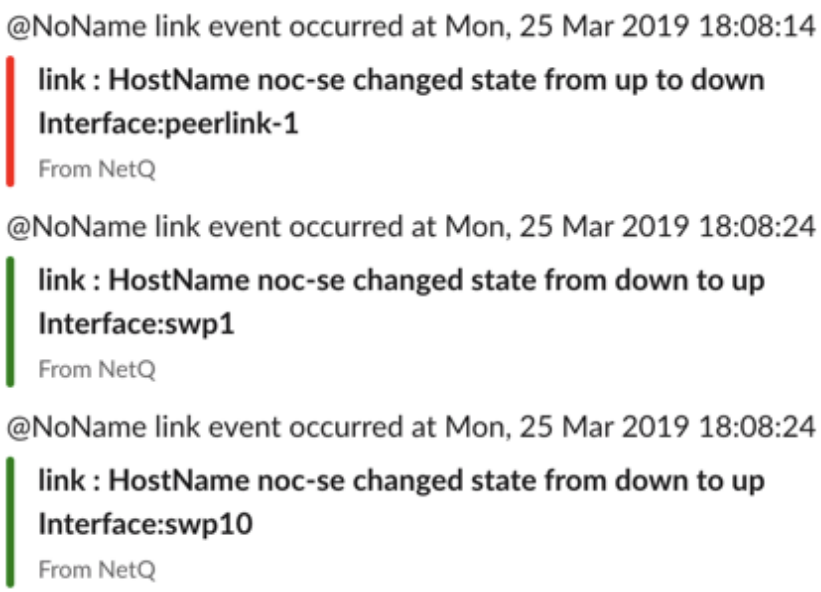
```
cumulus@switch:~$ netq add notification channel pagerduty pd-netq-events
integration-key c6d666e210a8425298ef7abde0d1998
Successfully added/updated channel pd-netq-events
```

2. Verify that the channel is configured properly.

```
cumulus@switch:~$ netq show notification channel
Matching config_notify records:
Name            Type            Severity        Channel Info
--------------- --------------- --------------- ------------------------
pd-netq-events  pagerduty       info            integration-key: c6d666e
                                                210a8425298ef7abde0d1998
```

### CONFIGURE A SLACK CHANNEL

NetQ Notifier sends notifications to Slack as incoming webhooks for a Slack channel you configure. For example:




To configure NetQ to send notifications to Slack:

1. If needed, create one or more Slack channels on which to receive the notifications.
    a. Click **+** next to **Channels**.
    b. Enter a name for the channel, and click **Create Channel**.
    c. Navigate to the new channel.

d. Click **+ Add an app** link below the channel name to open the application directory.
e. In the search box, start typing *incoming* and select ** **Incoming WebHooks** when it appears.
f. Click **Add Configuration** and enter the name of the channel you created (where you want to post notifications).
g. Click **Add Incoming WebHooks integration**.
h. Save WebHook URL in a text file for use in next step.

2. Configure the following options in the `netq config add notification channel` command:

| Option | Description |
|---|---|
| CHANNEL_TYPE <text-channel-name> | The third-party notification channel name; use *slack* in this case. |
| WEBHOOK | Copy the WebHook URL from the text file OR in the desired channel, locate the initial message indicating the addition of the webhook, click **incoming-webhook** link, click **Settings**.<br><br>Example URL: `https://hooks.slack.com/services/text/moretext/evenmoretext` |
| severity | The log level to set, which can be one of *error, warning, info,* or *debug*. The severity defaults to *info*. |
| tag | Optional tag appended to the Slack notification to highlight particular channels or people. The tag value must be preceded by the @ sign. For example, *@netq-info*. |

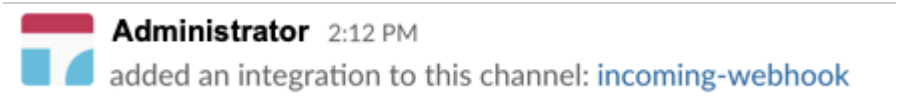
```
cumulus@switch:~$ netq add notification channel slack slk-netq-events
webhook https://hooks.slack.com/services/text/moretext/evenmoretext
Successfully added/updated channel netq-events
```

3. Verify the channel is configured correctly.

   From the CLI:

```
cumulus@switch:~$ netq show notification channel
Matching config_notify records:
Name            Type            Severity Channel Info
--------------- --------------- -------- ----------------------
slk-netq-events slack           info     webhook:https://hooks.s
                                         lack.com/services/text/
                                         moretext/evenmoretext
```

   From the Slack Channel:




### Create Rules

Each rule is comprised of a single key-value pair. The key-value pair indicates what messages to include or drop from event information sent to a notification channel. You can create more than one rule for a single filter. Creating multiple rules for a given filter can provide a very defined filter. For example, you can specify rules around hostnames or interface names, enabling you to filter messages specific to those hosts or interfaces. You should have already defined the PagerDuty or Slack channels (as described earlier).

There is a fixed set of valid rule keys. Values are entered as regular expressions and *vary according to your deployment*.

| Service | Rule Key | Description | Example Rule Values |
|---|---|---|---|
| BGP | message_type | Network protocol or service identifier | bgp |
| | hostname | User-defined, text-based name for a switch or host | server02, leaf11, exit01, spine-4 |
| | peer | User-defined, text-based name for a peer switch or host | server4, leaf-3, exit02, spine06 |
| | desc | Text description | |
| | vrf | Name of VRF interface | mgmt, default |
| | old_state | Previous state of the BGP service | Established, Failed |
| | new_state | Current state of the BGP service | Established, Failed |

| Service | Rule Key | Description | Example Rule Values |
|---|---|---|---|
| | old_last_reset_time | Previous time that BGP service was reset | Apr3, 2019, 4:17 pm |
| | new_last_reset_time | Most recent time that BGP service was reset | Apr8, 2019, 11:38 am |
| MLAG (CLAG) | message_type | Network protocol or service identifier | clag |
| | hostname | User-defined, text-based name for a switch or host | server02, leaf-9, exit01, spine04 |
| | old_conflicted_bonds | Previous pair of interfaces in a conflicted bond | swp7 swp8, swp3 swp4 |
| | new_conflicted_bonds | Current pair of interfaces in a conflicted bond | swp11 swp12, swp23 swp24 |
| | old_state_protodownbond | Previous state of the bond | protodown, up |

| Service | Rule Key | Description | Example Rule Values |
|---|---|---|---|
| | new_state_protodownbond | Current state of the bond | protodown, up |
| ConfigDiff | message_type | Network protocol or service identifier | configdiff |
| | hostname | User-defined, text-based name for a switch or host | server02, leaf11, exit01, spine-4 |
| | vni | Virtual Network Instance identifier | 12, 23 |
| | old_state | Previous state of the configuration file | created, modified |
| | new_state | Current state of the configuration file | created, modified |
| EVPN | message_type | Network protocol or service identifier | evpn |

| Service | Rule Key | Description | Example Rule Values |
|---|---|---|---|
| | hostname | User-defined, text-based name for a switch or host | server02, leaf-9, exit01, spine04 |
| | vni | Virtual Network Instance identifier | 12, 23 |
| | old_in_kernel_state | Previous VNI state, in kernel or not | true, false |
| | new_in_kernel_state | Current VNI state, in kernel or not | true, false |
| | old_adv_all_vni_state | Previous VNI advertising state, advertising all or not | true, false |
| | new_adv_all_vni_state | Current VNI advertising state, advertising all or not | true, false |
| Link | message_type | Network protocol or service identifier | link |

| Service | Rule Key | Description | Example Rule Values |
|---|---|---|---|
| | hostname | User-defined, text-based name for a switch or host | server02, leaf-6, exit01, spine7 |
| | ifname | Software interface name | eth0, swp53 |
| LLDP | message_type | Network protocol or service identifier | lldp |
| | hostname | User-defined, text-based name for a switch or host | server02, leaf41, exit01, spine-5, tor-36 |
| | ifname | Software interface name | eth1, swp12 |
| | old_peer_ifname | Previous software interface name | eth1, swp12, swp27 |
| | new_peer_ifname | Curent software interface name | eth1, swp12, swp27 |

| Service | Rule Key | Description | Example Rule Values |
|---|---|---|---|
| | old_peer_hostname | Previous user-defined, text-based name for a peer switch or host | server02, leaf41, exit01, spine-5, tor-36 |
| | new_peer_hostname | Current user-defined, text-based name for a peer switch or host | server02, leaf41, exit01, spine-5, tor-36 |
| Node | message_type | Network protocol or service identifier | node |
| | hostname | User-defined, text-based name for a switch or host | server02, leaf41, exit01, spine-5, tor-36 |
| | ntp_state | Current state of NTP service | in sync, not sync |
| | db_state | Current state of DB | Add, Update, Del, Dead |
| NTP | message_type | Network protocol or service identifier | ntp |

| Service | Rule Key | Description | Example Rule Values |
|---|---|---|---|
| | hostname | User-defined, text-based name for a switch or host | server02, leaf-9, exit01, spine04 |
| | old_state | Previous state of service | in sync, not sync |
| | new_state | Current state of service | in sync, not sync |
| Port | message_type | Network protocol or service identifier | port |
| | hostname | User-defined, text-based name for a switch or host | server02, leaf13, exit01, spine-8, tor-36 |
| | ifname | Interface name | eth0, swp14 |
| | old_speed | Previous speed rating of port | 10 G, 25 G, 40 G unknown |
| | old_transreceiver | Previous transceiver | 40G Base-CR4, 25G Base-CR |
| | old_vendor_name | Previous vendor name of installed port module | Amphenol, OEM Mellanox, Fiberstore, Finisar |

| Service | Rule Key | Description | Example Rule Values |
|---|---|---|---|
| | old_serial_number | Previous serial number of installed port module | MT1507VS05177 AVE1823402U, PTN1VH2 |
| | old_supported_fec | Previous forward error correction (FEC) support status | none, Base R, RS |
| | old_advertised_fec | Previous FEC advertising state | true, false, not reported |
| | old_fec | Previous FEC capability | none |
| | old_autoneg | Previous activation state of auto-negotiation | on, off |
| | new_speed | Current speed rating of port | 10 G, 25 G, 40 G |
| | new_transreceiver | Current transceiver | 40G Base-CR4, 25G Base-CR |

| Service | Rule Key | Description | Example Rule Values |
|---|---|---|---|
| | new_vendor_name | Current vendor name of installed port module | Amphenol, OEM Mellanox, Fiberstore, Finisar |
| | new_part_number | Current part number of installed port module | SFP-H10GB-CU1M, MC3309130-001 603020003 |
| | new_serial_number | Current serial number of installed port module | MT1507VS05177 AVE1823402U, PTN1VH2 |
| | new_supported_fec | Current FEC support status | none, Base R, RS |
| | new_advertised_fec | Current FEC advertising state | true, false |
| | new_fec | Current FEC capability | none |
| | new_autoneg | Current activation state of auto-negotiation | on, off |

| Service | Rule Key | Description | Example Rule Values |
|---|---|---|---|
| Sensors | sensor | Network protocol or service identifier | Fan: fan1, fan-2<br>Power Supply Unit: psu1, psu2<br>Temperature: psu1temp1, temp2 |
| | hostname | User-defined, text-based name for a switch or host | server02, leaf-26, exit01, spine2-4 |
| | old_state | Previous state of a fan, power supply unit, or thermal sensor | Fan: ok, absent, bad<br>PSU: ok, absent, bad<br>Temp: ok, busted, bad, critical |
| | new_state | Current state of a fan, power supply unit, or thermal sensor | Fan: ok, absent, bad<br>PSU: ok, absent, bad<br>Temp: ok, busted, bad, critical |
| | old_s_state | Previous state of a fan or power supply unit. | Fan: up, down<br>PSU: up, down |
| | new_s_state | Current state of a fan or power supply unit. | Fan: up, down<br>PSU: up, down |

| Service | Rule Key | Description | Example Rule Values |
|---|---|---|---|
| | new_s_max | Current maximum temperature threshold value | Temp: 110 |
| | new_s_crit | Current critical high temperature threshold value | Temp: 85 |
| | new_s_lcrit | Current critical low temperature threshold value | Temp: -25 |
| | new_s_min | Current minimum temperature threshold value | Temp: -50 |
| Services | message_type | Network protocol or service identifier | services |
| | hostname | User-defined, text-based name for a switch or host | server02, leaf03 exit01, spine-8 |

| Service | Rule Key | Description | Example Rule Values |
|---|---|---|---|
| | name | Name of service | clagd, lldpd, ssh ntp, netqd, net-agent |
| | old_pid | Previous process or service identifier | 12323, 52941 |
| | new_pid | Current process or service identifier | 12323, 52941 |
| | old_status | Previous status of service | up, down |
| | new_status | Current status of service | up, down |

NOTE

Rule names are case sensitive, and no wildcards are permitted. Rule names may contain spaces, but must be enclosed with single quotes in commands. It is easier to use dashes in place of spaces or mixed case for better readability. For example, use bgpSessionChanges or BGP-session-changes or BGPsessions, instead of 'BGP Session Changes'.

Use Tab completion to view the command options syntax.

### EXAMPLE RULES

Create a BGP Rule Based on Hostname:

```
cumulus@switch:~$ netq add notification rule bgpHostname key hostname value spine-01
Successfully added/updated rule bgpHostname
```

Create a Rule Based on a Configuration File State Change:

```
cumulus@switch:~$ netq add notification rule sysconf key configdiff value updated
Successfully added/updated rule sysconf
```

Create an EVPN Rule Based on a VNI:

```
cumulus@switch:~$ netq add notification rule evpnVni key vni value 42
Successfully added/updated rule evpnVni
```

Create an Interface Rule Based on FEC Support:

```
cumulus@switch:~$ netq add notification rule fecSupport key new_supported_fec value supported
Successfully added/updated rule fecSupport
```

Create a Service Rule Based on a Status Change:

```
cumulus@switch:~$ netq add notification rule svcStatus key new_status value
down
Successfully added/updated rule svcStatus
```

Create a Sensor Rule Based on a Threshold:

```
cumulus@switch:~$ netq add notification rule overTemp key new_s_crit value 24
Successfully added/updated rule overTemp
```

Create an Interface Rule Based on Port:

```
cumulus@switch:~$ netq add notification rule swp52 key port value swp52
Successfully added/updated rule swp52
```

### VIEW THE RULE CONFIGURATIONS

Use the `netq show notification` command to view the rules on your platform.

```
cumulus@switch:~$ netq show notification rule

Matching config_notify records:
Name            Rule Key        Rule Value
--------------- --------------- --------------------
```

```
bgpHostname     hostname        spine-01
evpnVni         vni             42
fecSupport      new_supported_fe supported
                c
overTemp        new_s_crit      24
svcStatus       new_status      down
swp52           port            swp52
sysconf         configdiff      updated
```

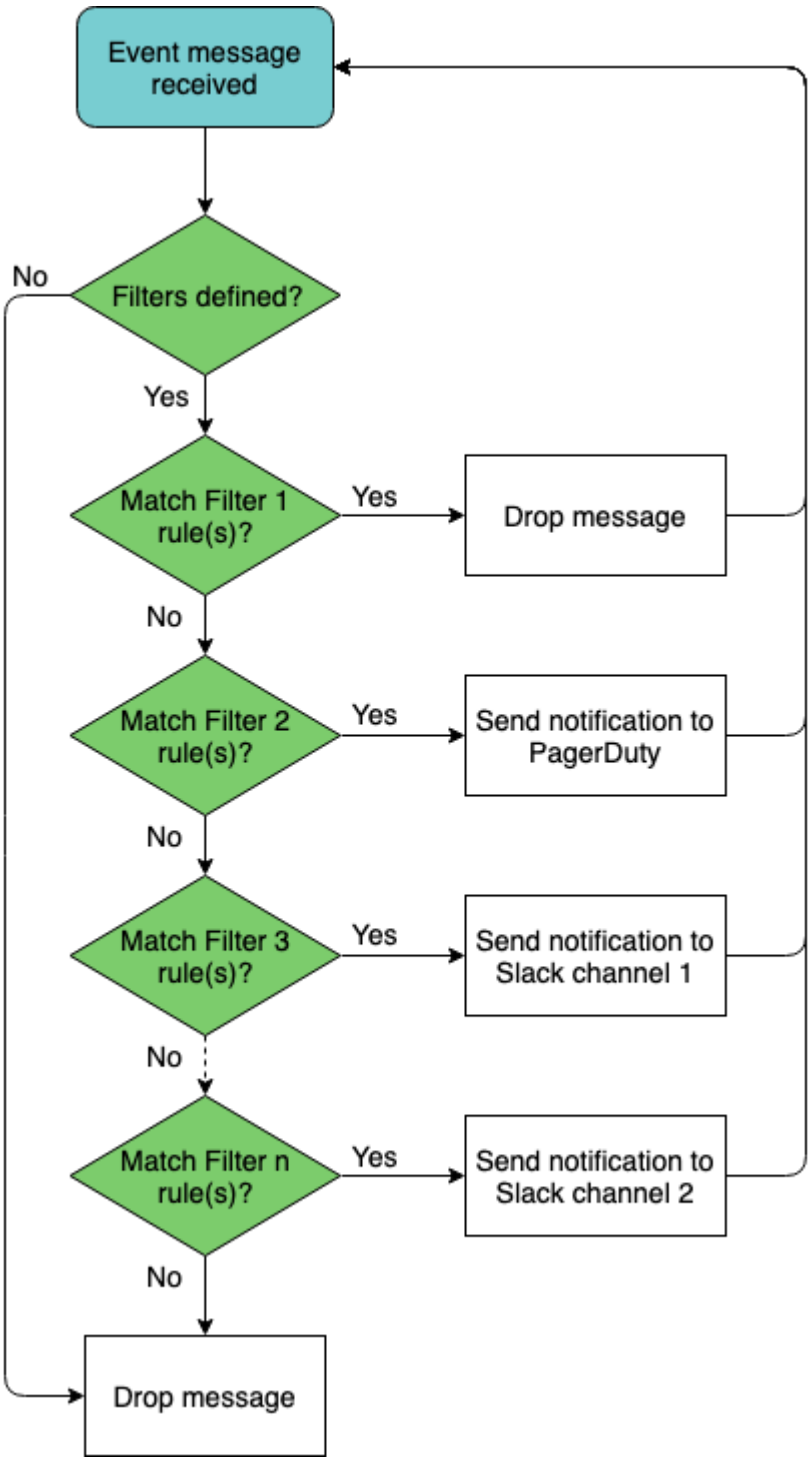
### Create Filters

You can limit or direct event messages using filters. Filters are created based on rules you define; like those in the previous section. Each filter contains one or more rules. When a message matches the rule, it is sent to the indicated destination. Before you can create filters, you need to have already defined the rules and configured PagerDuty and/or Slack channels (as described earlier).

As filters are created, they are added to the bottom of a filter list. By default, filters are processed in the order they appear in this list (from top to bottom) until a match is found. This means that each event message is first evaluated by the first filter listed, and if it matches then it is processed, ignoring all other filters, and the system moves on to the next event message received. If the event does not match the first filter, it is tested against the second filter, and if it matches then it is processed and the system moves on to the next event received. And so forth. Events that do not match any filter are ignored.

You may need to change the order of filters in the list to ensure you capture the events you want and drop the events you do not want. This is possible using the *before* or *after* keywords to ensure one rule is processed before or after another.

This diagram shows an example with four defined filters with sample output results.


Event message received
Filters defined?
No
Yes
Match Filter 1 rule(s)?
Yes
Drop message
No
Match Filter 2 rule(s)?
Yes
Send notification to PagerDuty
No
Match Filter 3 rule(s)?
Yes
Send notification to Slack channel 1
No
Match Filter n rule(s)?
Yes
Send notification to Slack channel 2
No
Drop message

> ⓘ NOTE
>
> Filter names may contain spaces, but *must* be enclosed with single quotes in commands. It is easier to use dashes in place of spaces or mixed case for better readability. For example, use bgpSessionChanges or BGP-session-changes or BGPsessions, instead of 'BGP Session Changes'. Filter names are also case sensitive.

### EXAMPLE FILTERS

Create a filter for BGP Events on a Particular Device:

```
cumulus@switch:~$ netq add notification filter bgpSpine rule bgpHostname
channel pd-netq-events
Successfully added/updated filter bgpSpine
```

Create a Filter for a Given VNI in Your EVPN Overlay:

```
cumulus@switch:~$ netq add notification filter vni42 severity warning rule
evpnVni channel pd-netq-events
Successfully added/updated filter vni42
```

Create a Filter for when a Configuration File has been Updated:

```
cumulus@switch:~$ netq add notification filter configChange severity info rule
sysconf channel slk-netq-events
Successfully added/updated filter configChange
```

Create a Filter to Monitor Ports with FEC Support:

```
cumulus@switch:~$ netq add notification filter newFEC rule fecSupport channel
slk-netq-events
Successfully added/updated filter newFEC
```

Create a Filter to Monitor for Services that Change to a Down State:

```
cumulus@switch:~$ netq add notification filter svcDown severity error rule
svcStatus channel slk-netq-events
Successfully added/updated filter svcDown
```

Create a Filter to Monitor Overheating Platforms:

```
cumulus@switch:~$ netq add notification filter critTemp severity error rule
overTemp channel pd-netq-events
Successfully added/updated filter critTemp
```

Create a Filter to Drop Messages from a Given Interface, and match against this filter before any other filters. To create a drop style filter, do not specify a channel. To put the filter first, use the *before* option.

```
cumulus@switch:~$ netq add notification filter swp52Drop severity error rule
swp52 before bgpSpine
Successfully added/updated filter swp52Drop
```

### VIEW THE FILTER CONFIGURATIONS

Use the `netq show notification` command to view the filters on your platform.

```
cumulus@switch:~$ netq show notification filter
Matching config_notify records:
Name            Order     Severity        Channels        Rules
--------------- ---------- ---------------- ---------------- ----------
swp52Drop       1         error           NetqDefaultChann swp52
                                            el
bgpSpine        2         info            pd-netq-events   bgpHostnam
                                                           e
vni42           3         warning         pd-netq-events   evpnVni
configChange    4         info            slk-netq-events  sysconf
newFEC          5         info            slk-netq-events  fecSupport
svcDown         6         critical        slk-netq-events  svcStatus
critTemp        7         critical        pd-netq-events   overTemp
```

### REORDER FILTERS

When you look at the results of the `netq show notification filter` command above, you might notice that although you have the drop-based filter first (no point in looking at something you are going to drop anyway, so that is good), but the critical severity events are processed last, per the current definitions. If you wanted to process those before lesser severity events, you can reorder the list using the *before* and *after* options.

For example, to put the two critical severity event filters just below the drop filter:

```
cumulus@switch:~$ netq add notification filter critTemp after swp52Drop
Successfully added/updated filter critTemp
cumulus@switch:~$ netq add notification filter svcDown before bgpSpine
Successfully added/updated filter svcDown
```

> TIP
> You do not need to reenter all the severity, channel, and rule information for existing rules if you only want to change their processing order.

Run the `netq show notification` command again to verify the changes:

```
cumulus@switch:~$ netq show notification filter
Matching config_notify records:
Name            Order     Severity        Channels        Rules
--------------- ---------- ---------------- ---------------- ----------
swp52Drop       1         error           NetqDefaultChann swp52
```

```
                              el
critTemp        2         critical        pd-netq-events   overTemp
svcDown         3         critical        slk-netq-events  svcStatus
bgpSpine        4         info            pd-netq-events   bgpHostnam
                                                           e
vni42           5         warning         pd-netq-events   evpnVni
configChange    6         info            slk-netq-events  sysconf
newFEC          7         info            slk-netq-events  fecSupport
```

## Example Notification Configurations

Putting all of these channel, rule, and filter definitions together you create a complete notification configuration. The following are example notification configurations are created using the three-step process outlined above. Refer to Integrate NetQ with an Event Notification Application for details and instructions for creating channels, rules, and filters.

### Create a Notification for BGP Events from a Selected Switch

In this example, we created a notification integration with a PagerDuty channel called *pd-netq-events*. We then created a rule *bgpHostname* and a filter called *4bgpSpine* for any notifications from *spine-01*. The result is that any info severity event messages from Spine-01 are filtered to the *pd-netq-events* ** channel.

```
cumulus@switch:~$ netq add notification channel pagerduty pd-netq-events
integration-key 1234567890
Successfully added/updated channel pd-netq-events
cumulus@switch:~$ netq add notification rule bgpHostname key node value
spine-01
```

```
Successfully added/updated rule bgpHostname

cumulus@switch:~$ netq add notification filter bgpSpine rule bgpHostname
channel pd-netq-events
Successfully added/updated filter bgpSpine
cumulus@switch:~$ netq show notification channel
Matching config_notify records:
Name            Type            Severity        Channel Info
--------------- --------------- --------------- ----------------------
pd-netq-events  pagerduty       info            integration-key: 1234567
                                                890

cumulus@switch:~$ netq show notification rule
Matching config_notify records:
Name            Rule Key        Rule Value
--------------- --------------- --------------------
bgpHostname     hostname        spine-01

cumulus@switch:~$ netq show notification filter
Matching config_notify records:
Name            Order     Severity        Channels        Rules
--------------- ---------- --------------- --------------- ----------
bgpSpine        1         info            pd-netq-events   bgpHostnam
                                                           e
```

### Create a Notification for Warnings on a Given EVPN VNI

In this example, we created a notification integration with a PagerDuty channel called *pd-netq-events*. We then created a rule *evpnVni* and a filter called *3vni42* for any warnings

messages from VNI 42 on the EVPN overlay network. The result is that any warning severity event messages from VNI 42 are filtered to the *pd-netq-events* channel.

```
cumulus@switch:~$ netq add notification channel pagerduty pd-netq-events
integration-key 1234567890
Successfully added/updated channel pd-netq-events

cumulus@switch:~$ netq add notification rule evpnVni key vni value 42
Successfully added/updated rule evpnVni

cumulus@switch:~$ netq add notification filter vni42 rule evpnVni channel pd-
netq-events
Successfully added/updated filter vni42

cumulus@switch:~$ netq show notification channel
Matching config_notify records:
Name            Type            Severity        Channel Info
--------------- --------------- --------------- ------------------------
pd-netq-events  pagerduty       info            integration-key: 1234567
                                                890

cumulus@switch:~$ netq show notification rule
Matching config_notify records:
Name            Rule Key        Rule Value
--------------- --------------- --------------------
bgpHostname     hostname        spine-01
evpnVni         vni             42

cumulus@switch:~$ netq show notification filter
```

```
Matching config_notify records:
Name            Order     Severity        Channels        Rules
--------------- ---------- --------------- --------------- ----------
bgpSpine        1         info            pd-netq-events  bgpHostnam
                                                          e
vni42           2         warning         pd-netq-events  evpnVni
```

### Create a Notification for Configuration File Changes

In this example, we created a notification integration with a Slack channel called *slk-netq-events*. We then created a rule *sysconf* and a filter called *configChange* for any configuration file update messages. The result is that any configuration update messages are filtered to the *slk-netq-events* channel.

```
cumulus@switch:~$ netq add notification channel slack slk-netq-events webhook
https://hooks.slack.com/services/text/moretext/evenmoretext
Successfully added/updated channel slk-netq-events

cumulus@switch:~$ netq add notification rule sysconf key configdiff value updated
Successfully added/updated rule sysconf

cumulus@switch:~$ netq add notification filter configChange severity info rule
sysconf channel slk-netq-events
Successfully added/updated filter configChange

cumulus@switch:~$ netq show notification channel
Matching config_notify records:
Name            Type            Severity Channel Info
--------------- ---------------- -------- ----------------------
```

```
slk-netq-events slack          info     webhook:https://hooks.s
                                        lack.com/services/text/
                                        moretext/evenmoretext

cumulus@switch:~$ netq show notification rule
Matching config_notify records:
Name           Rule Key        Rule Value
-------------- --------------- -------------------
bgpHostname    hostname        spine-01
evpnVni        vni             42
sysconf        configdiff      updated

cumulus@switch:~$ netq show notification filter
Matching config_notify records:
Name           Order     Severity        Channels        Rules
-------------- ---------- --------------- --------------- ----------
bgpSpine       1         info            pd-netq-events   bgpHostnam
                                                          e
vni42          2         warning         pd-netq-events   evpnVni
configChange   3         info            slk-netq-events  sysconf
```

### Create a Notification for When a Service Goes Down

In this example, we created a notification integration with a Slack channel called *slk-netq-events*. We then created a rule *svcStatus* and a filter called *svcDown* for any services state messages indicating a service is no longer operational. The result is that any service down messages are filtered to the *slk-netq-events* channel.

```
cumulus@switch:~$ netq add notification channel slack slk-netq-events webhook
https://hooks.slack.com/services/text/moretext/evenmoretext
Successfully added/updated channel slk-netq-events

cumulus@switch:~$ netq add notification rule svcStatus key new_status value
down
Successfully added/updated rule svcStatus

cumulus@switch:~$ netq add notification filter svcDown severity error rule
svcStatus channel slk-netq-events
Successfully added/updated filter svcDown

cumulus@switch:~$ netq show notification channel
Matching config_notify records:
Name           Type            Severity Channel Info
-------------- --------------- -------- ----------------------
slk-netq-events slack          info     webhook:https://hooks.s
                                        lack.com/services/text/
                                        moretext/evenmoretext

cumulus@switch:~$ netq show notification rule
Matching config_notify records:
Name           Rule Key        Rule Value
-------------- --------------- --------------------
bgpHostname    hostname        spine-01
evpnVni        vni             42
svcStatus      new_status      down
sysconf        configdiff      updated
```

```
cumulus@switch:~$ netq show notification filter
Matching config_notify records:
Name            Order     Severity        Channels        Rules
--------------- ---------- ---------------- ---------------- ----------
bgpSpine        1         info            pd-netq-events   bgpHostnam
                                                           e
vni42           2         warning         pd-netq-events   evpnVni
configChange    3         info            slk-netq-events  sysconf
svcDown         4         critical        slk-netq-events  svcStatus
```

### Create a Filter to Drop Notifications from a Given Interface

In this example, we created a notification integration with a Slack channel called *slk-netq-events*. We then created a rule *swp52* and a filter called *swp52Drop* that drops all notifications for events from interface *swp52*.

```
cumulus@switch:~$ netq add notification channel slack slk-netq-events webhook
https://hooks.slack.com/services/text/moretext/evenmoretext
Successfully added/updated channel slk-netq-events

cumulus@switch:~$ netq add notification rule swp52 key port value swp52
Successfully added/updated rule swp52

cumulus@switch:~$ netq add notification filter swp52Drop severity error rule
swp52 before bgpSpine
Successfully added/updated filter swp52Drop

cumulus@switch:~$ netq show notification channel
Matching config_notify records:
```

```
Name           Type           Severity Channel Info
-------------- -------------- -------- ----------------------
slk-netq-events slack          info     webhook:https://hooks.s
                                        lack.com/services/text/
                                        moretext/evenmoretext


cumulus@switch:~$ netq show notification rule
Matching config_notify records:
Name           Rule Key       Rule Value
-------------- -------------- --------------------
bgpHostname    hostname       spine-01
evpnVni        vni            42
svcStatus      new_status     down
swp52          port           swp52
sysconf        configdiff     updated

cumulus@switch:~$ netq show notification filter
Matching config_notify records:
Name           Order     Severity       Channels       Rules
-------------- --------- -------------- -------------- ----------
swp52Drop      1         error          NetqDefaultChann swp52
                                        el
bgpSpine       2         info           pd-netq-events   bgpHostnam
                                                          e
vni42          3         warning        pd-netq-events   evpnVni
configChange   4         info           slk-netq-events  sysconf
svcDown        5         critical       slk-netq-events  svcStatus
```

### Create a Notification for a Given Device that has a Tendency to Overheat (using multiple rules)

In this example, we created a notification when switch *leaf04* has passed over the high temperature threshold. Two rules were needed to create this notification, one to identify the specific device and one to identify the temperature trigger. We sent the message to the *pd-netq-events* channel.

```
cumulus@switch:~$ netq add notification channel pagerduty pd-netq-events
integration-key 1234567890
Successfully added/updated channel pd-netq-events

cumulus@switch:~$ netq add notification rule switchLeaf04 key hostname value
leaf04
Successfully added/updated rule switchLeaf04
cumulus@switch:~$ netq add notification rule overTemp key new_s_crit value 24
Successfully added/updated rule overTemp

cumulus@switch:~$ netq add notification filter critTemp rule switchLeaf04
channel pd-netq-events
Successfully added/updated filter critTemp
cumulus@switch:~$ netq add notification filter critTemp severity critical rule
overTemp channel pd-netq-events
Successfully added/updated filter critTemp

cumulus@switch:~$ netq show notification channel
Matching config_notify records:
Name            Type            Severity        Channel Info
--------------- --------------- --------------- ------------------------
```

```
pd-netq-events  pagerduty       info            integration-key: 1234567
                                                890


cumulus@switch:~$ netq show notification rule
Matching config_notify records:
Name            Rule Key        Rule Value
--------------- --------------- --------------------
bgpHostname     hostname        spine-01
evpnVni         vni             42
overTemp        new_s_crit      24
svcStatus       new_status      down
switchLeaf04    hostname        leaf04
swp52           port            swp52
sysconf         configdiff      updated
cumulus@switch:~$ netq show notification filter
Matching config_notify records:
Name            Order     Severity        Channels        Rules
--------------- ---------- --------------- --------------- ----------
swp52Drop       1         error           NetqDefaultChann swp52
                                            el
bgpSpine        2         info            pd-netq-events   bgpHostnam
                                                           e
vni42           3         warning         pd-netq-events   evpnVni
configChange    4         info            slk-netq-events  sysconf
svcDown         5         critical        slk-netq-events  svcStatus
critTemp        6         critical        pd-netq-events   switchLeaf
                                                           04
                                                           overTemp
```

### View Notification Configurations in JSON Format

You can view configured integrations using the `netq show notification` commands. To view the channels, filters, and rules, run the three flavors of the command. Include the `json` option to display JSON-formatted output.

For example:

```
cumulus@switch:~$ netq show notification channel json
{
   "config_notify":[
      {
         "type":"slack",
         "name":"slk-netq-events",
         "channelInfo":"webhook:https://hooks.slack.com/services/text/moretext/
evenmoretext",
         "severity":"info"
      },
      {
         "type":"pagerduty",
         "name":"pd-netq-events",
         "channelInfo":"integration-key: 1234567890",
         "severity":"info"
   }
   ],
   "truncatedResult":false
}

cumulus@switch:~$ netq show notification rule json
{
```

```
"config_notify":[
  {
    "ruleKey":"hostname",
    "ruleValue":"spine-01",
    "name":"bgpHostname"
  },
  {
    "ruleKey":"vni",
    "ruleValue":42,
    "name":"evpnVni"
  },
  {
    "ruleKey":"new_supported_fec",
    "ruleValue":"supported",
    "name":"fecSupport"
  },
  {
    "ruleKey":"new_s_crit",
    "ruleValue":24,
    "name":"overTemp"
  },
  {
    "ruleKey":"new_status",
    "ruleValue":"down",
    "name":"svcStatus"
  },
  {
    "ruleKey":"configdiff",
    "ruleValue":"updated",
```

```
        "name":"sysconf"
    }
    ],
    "truncatedResult":false
}

cumulus@switch:~$ netq show notification filter json
{
    "config_notify":[
        {
            "channels":"pd-netq-events",
            "rules":"overTemp",
            "name":"1critTemp",
            "severity":"critical"
        },
        {
            "channels":"pd-netq-events",
            "rules":"evpnVni",
            "name":"3vni42",
            "severity":"warning"
        },
        {
            "channels":"pd-netq-events",
            "rules":"bgpHostname",
            "name":"4bgpSpine",
            "severity":"info"
        },
        {
            "channels":"slk-netq-events",
```

```
            "rules":"sysconf",
            "name":"configChange",
            "severity":"info"
        },
        {
            "channels":"slk-netq-events",
            "rules":"fecSupport",
            "name":"newFEC",
            "severity":"info"
        },
        {
            "channels":"slk-netq-events",
            "rules":"svcStatus",
            "name":"svcDown",
            "severity":"critical"
    }
    ],
    "truncatedResult":false
}
```


## Manage Event Notification Integrations

You might need to modify event notification configurations at some point in the lifecycle of your deployment. Optionally, you might want to configure a proxy.

### Remove an Event Notification Channel

You can delete an event notification integration using the `netq config del notification` command. You can verify it has been removed using the related `show` command.

For example, to remove a Slack integration and verify it is no longer in the configuration:

```
cumulus@switch:~$ netq del notification channel slk-netq-events
cumulus@switch:~$ netq show notification channel
Matching config_notify records:
Name            Type            Severity        Channel Info
--------------- --------------- --------------- ------------------------
pd-netq-events  pagerduty       info            integration-key: 1234567
                                                890
```

### Delete an Event Notification Rule

To delete a rule, use the following command, then verify it has been removed:

```
cumulus@switch:~$ netq del notification rule swp52
cumulus@switch:~$ netq show notification rule
Matching config_notify records:
Name            Rule Key        Rule Value
--------------- --------------- --------------------
bgpHostname     hostname        spine-01
evpnVni         vni             42
overTemp        new_s_crit      24
svcStatus       new_status      down
switchLeaf04    hostname        leaf04
sysconf         configdiff      updated
```

### Delete an Event Notification Filter

To delete a filter, use the following command, then verify it has been removed:

```
cumulus@switch:~$ netq del notification filter bgpSpine
cumulus@switch:~$ netq show notification filter
Matching config_notify records:
Name            Order     Severity        Channels        Rules
--------------- ---------- ---------------- ---------------- ----------
swp52Drop       1         error           NetqDefaultChann swp52
                                          el
vni42           2         warning         pd-netq-events   evpnVni
configChange    3         info            slk-netq-events  sysconf
svcDown         4         critical        slk-netq-events  svcStatus
critTemp        5         critical        pd-netq-events   switchLeaf
                                          04
                                          overTemp
```

## Integrate with a Hardware Chassis

NetQ can run within a Facebook Backpack chassis, Cumulus Express CX-10256-S chassis or Edgecore OMP-800 chassis.

Keep the following issues in mind if you intend to use NetQ with a chassis:

- You must assign a unique hostname to every node that runs the NetQ Agent. By default, all the fabric cards in the chassis have the same hostname.
- The NetQ Agent must be installed on every line card.
- No information is returned about the ASIC when you run `netq show inventory asic`. This is a known issue.

- Since the chassis sensor information is shared, every line card and fabric card can report the same sensor data. By default, sensor data is disabled on a chassis to avoid this duplication . To enable sensor data on a line card, edit `/etc/netq/netq.yml` or `/etc/netq/config.d/user.yml` and set the `send_chassis_sensor_data` keyword to *true*, then restart the NetQ Agent with `netq config agent restart`. Configuring NetQ in this way prevents any duplication of data in the NetQ database.

```
cumulus@chassis:~$ sudo nano /etc/netq/netq.yml


...
netq-agent:
  send_chassis_sensor_data: true
...
```

# Integrate NetQ with Your LDAP Server

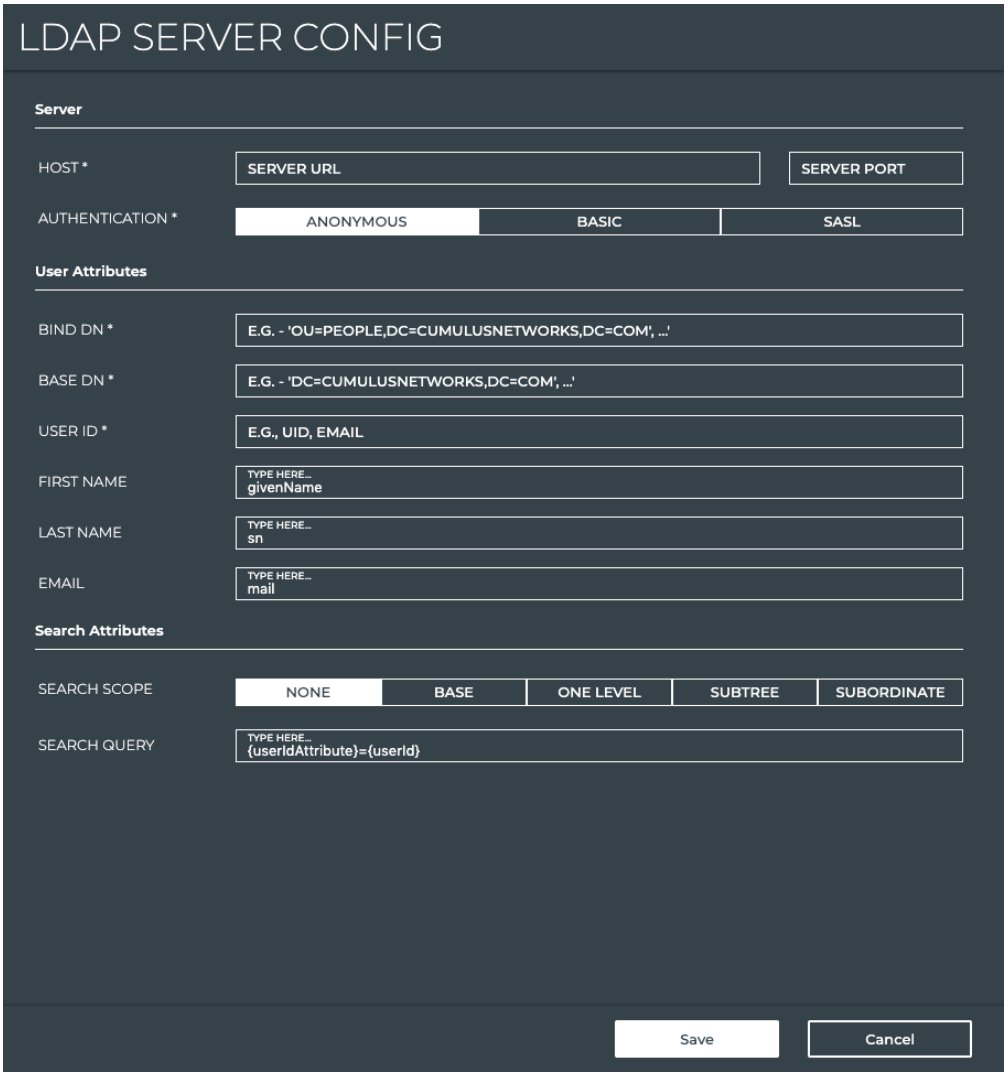
With this release and an administrator role, you are able to integrate the NetQ role-based access control (RBAC) with your lightweight directory access protocol (LDAP) server in on-premises deployments. NetQ maintains control over role-based permissions, but LDAP is used for authentication of the users. A copy of each user from LDAP is stored in the local NetQ database.

Configuring an LDAP server does not prevent you from configuring local users (stored and managed in the NetQ database) as well.

## Create an LDAP Configuration

One LDAP server can be configured per admin user account. Once LDAP is configured, you can validate the connectivity (and configuration) and save the configuration.

1. Click
   ≡
   , then select *Management* under **Admin**.
2. Locate the LDAP Server Info card, and click **Configure LDAP**.




3. Obtain and enter the following information about your LDAP server:

| Parameter | Description |
|---|---|
| Server | • **Host***: URL of the LDAP server<br>• **Server Port***: Name of the port on which to communicate with the LDAP server<br>• **Authentication***: Select from<br>◦ *Anonymous*: LDAP client does not require authentication<br>◦ *Basic*: LDAP client sends the username as an LDAP distinguished name along with a password as clear text to the LDAP server. Also called Simple authentication.<br>◦ *SASL*: LDAP client and server negotiate an authentication mechanism |
| User Attributes | • **Bind DN***: Username (Distinguished Name) used for search queries<br>• **Base DN***: Base of the subtree, where in directory structure search query begins<br>• **User ID***: Type of identifier used to specify an LDAP user<br>• **First Name**: Given name of LDAP user<br>• **Last Name**: Surname of LDAP user<br>• **Email**: Electronic mail address for LDAP user |
| Search Attributes | • **Search Scope**: Specifies the portion of the target subtree used in a search query. Select from<br>◦ *None*: No search allowed for user on this LDAP server<br>◦ *Base*: Search for users at the base level only; no subordinates<br>◦ *One Level*: Search for immediate children of user; not at base or for any descendants<br>◦ *Subtree*: Search for users from base, subordinates at any depth<br>◦ *Subordinate*: Search for subordinates at any depth of user; but not at base<br>• **Search Query**: Actual search query |

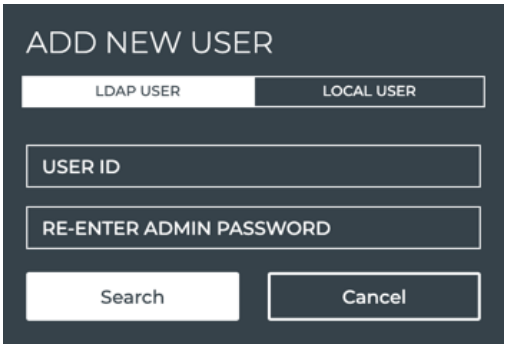
**Note**: Items with an asterisk (*) are required. All others are optional.

4. Click **Save** to complete the configuration, or click **Cancel** to discard the configuration.

> ⚠ IMPORTANT
>
> LDAP config cannot be changed once configured. If you need to change the configuration, you must delete the current LDAP configuration and create a new one. Note that if you change the LDAP server configuration, all users created against that LDAP server remain in the NetQ database and continue to be visible, but are no longer be viable. You must manually delete those users if you do not want to see them.

## Add LDAP Users to NetQ

1. Click
   ≡
   , then select *Management* under **Admin**.
2. Locate the User Accounts card, and click **Manage**.
3. On the User Accounts tab, click **Add User**.

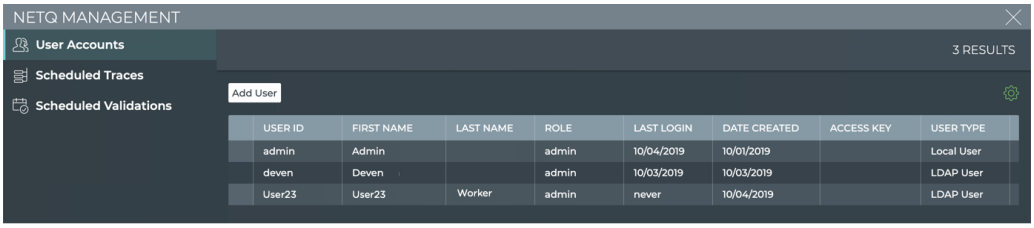



4. Select **LDAP User**.
5. Enter the user's ID.
6. Enter your administrator password.
7. Click **Search**.
8. If the user is found, the email address, first and last name fields are automatically filled in on the Add New User form. If searching is not enabled on the LDAP server, you must enter the information manually.


TIP

If the fields are not automatically filled in, and searching is enabled on the LDAP server, you might require changes to the mapping file.

9. Select the role for this user, *admin* or *user*, in the **User Type** dropdown.
10. Enter your admin password, and click **Save**, or click **Cancel** to discard the user account.



> TIP
>
> LDAP user passwords are not stored in the NetQ database and are always authenticated against LDAP.

11. Repeat these steps to add additional LDAP users.

### Remove LDAP Users from NetQ

You can remove LDAP users in the same manner as local users.

1. Click
   
   , then select *Management* under **Admin**.
2. Locate the User Accounts card, and click **Manage**.
3. Select the user or users you want to remove.
4. Click
   
   in the Edit menu.

> TIP
>
> If an LDAP user is deleted in LDAP it is not automatically deleted from NetQ; however, the login for these LDAP users stop working immediately.

# Integrate NetQ with Grafana

Switches collect statistics about the performance of their interfaces. The NetQ Agent on each switch collects these statistics every 15 seconds and then sends them to your NetQ Server or Appliance.

NetQ only collects statistics for physical interfaces; it does not collect statistics for virtual (non-physical) interfaces, such as bonds, bridges, and VXLANs. Specifically, the NetQ Agent collects the following interface statistics:

- **Transmit**: tx_bytes, tx_carrier, tx_colls, tx_drop, tx_errs, tx_packets
- **Receive**: rx_bytes, rx_drop, rx_errs, rx_frame, rx_multicast, rx_packets

You can use Grafana, an open source analytics and monitoring tool, to view the interface statistics collected by the NetQ Agents. The fastest way to achieve this is by installing Grafana on an application server or locally per user, and then importing the prepared NetQ dashboard.

## Install NetQ Plug-in for Grafana

The first step is to install the NetQ plug-in on your NetQ server or appliance. There are three ways to install the plug-in:

- **Docker File**: Add the following to your existing Dockerfile

```
# grafana docker file
FROM grafana/grafana:6.2.2
```

```
RUN grafana-cli --pluginUrl https://netq-grafana-dsrc.s3-us-
west-2.amazonaws.com/dist.zip plugins install netq-dashboard
```

- **Grafana Docker Image**: Download and run the plug-in in your Grafana Docker container

```
$ docker run -d -p 3000:3000 --name=grafana -e "GF_INSTALL_PLUGINS=https://
netq-grafana-dsrc.s3-us-west-2.amazonaws.com/dist.zip;netq-dashboard"
grafana/grafana
```

- **Grafana CLI**: Download and install the Grafana plug-in using Grafana CLI

```
brew update
brew install grafana
brew services start grafana
grafana-cli --pluginUrl https://netq-grafana-dsrc.s3-us-west-2.amazonaws.com/
dist.zip plugins install netq-dashboard
brew services restart grafana
```

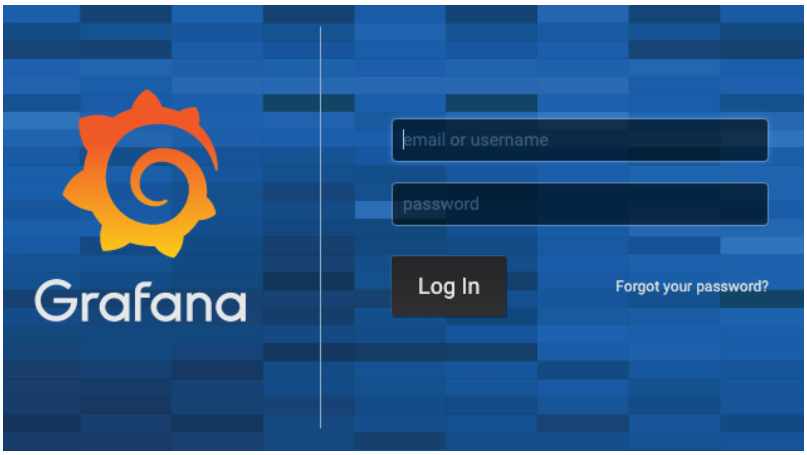
Then restart Grafana.

> ⚠ IMPORTANT
>
> The Grafana GUI is accessed through port 3000 by default. If you are running Grafana on a simulation server, you may need to modify forwarding rules in IPtables to allow access to port 3000.

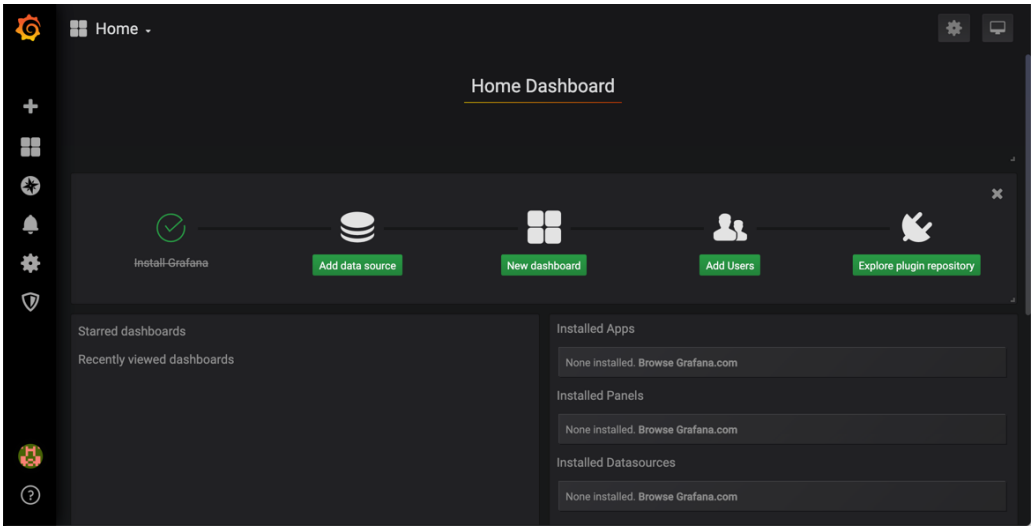
# Set Up a Dashboard

The quickest way to view the interface statistics for your Cumulus Linux network is to make use of the pre-configured dashboard installed with the plug-in. Once you are familiar with that dashboard, you can create new dashboards or add new panels to the NetQ dashboard.

1. Open the Grafana user interface:
    - **Remote access**: Enter *<NetQ-Server-or-Appliance-IPaddr>:3000* in a web browser address field.
    - **Local access**: Enter *localhost:3000* in a web browser address field.
2. Log in using your application credentials.



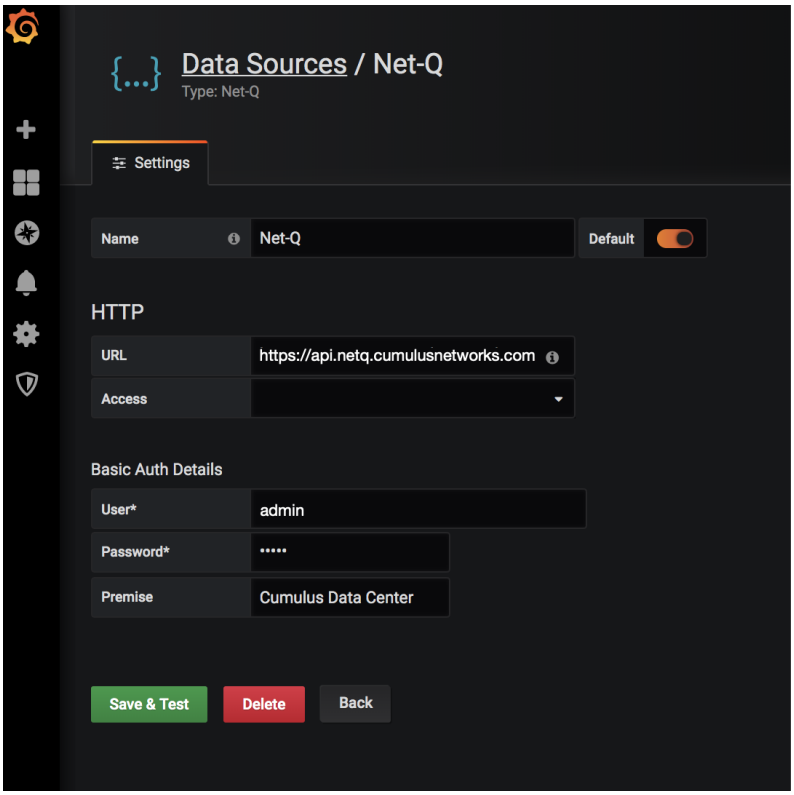

The Home Dashboard appears.




3. Click **Add data source** or

   > *Data Sources*.




4. Enter **Net-Q** in the search box or scroll down to the **Other** category, and select *Net-Q* from there.
5. Enter *Net-Q* into the **Name** field.

6. Enter the URL used to access the NetQ cloud service; for example *api.netq.cumulusnetworks.com*
7. Enter your credentials (the ones used to login)
8. For cloud deployments only, if you have more than one premises configured, you can select the premises you want to view, as follows:

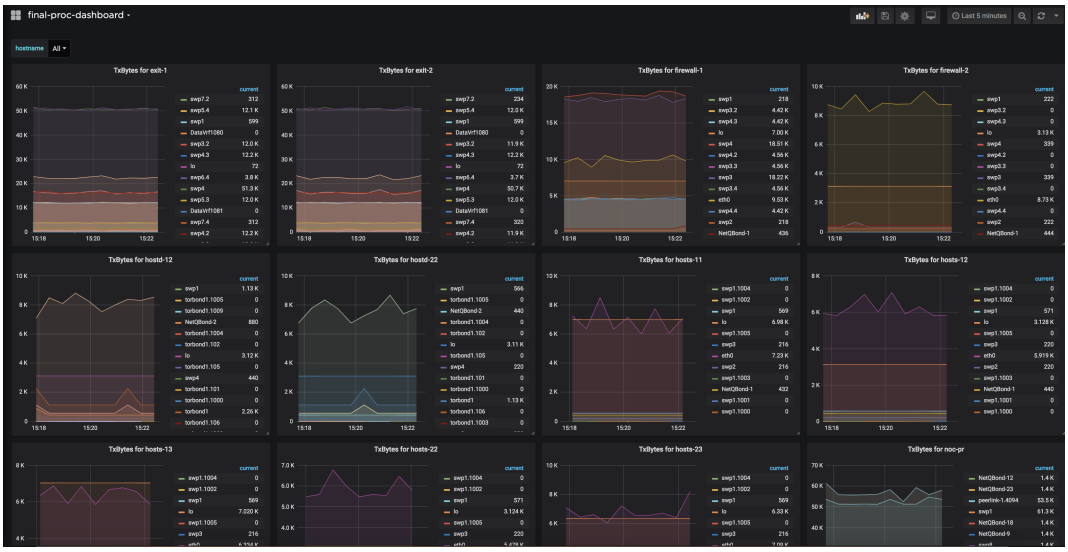
    - If you leave the **Premises** field blank, the first premises name is selected by default
    - If you enter a premises name, that premises is selected for viewing

    *Note*: If multiple premises are configured with the same name, then the first premises of that name is selected for viewing
9. Click **Save & Test**

## Create a Dashboard

You can either use the dashboard provided with the plug-in, NetQ Interface Statistics, or create your own.

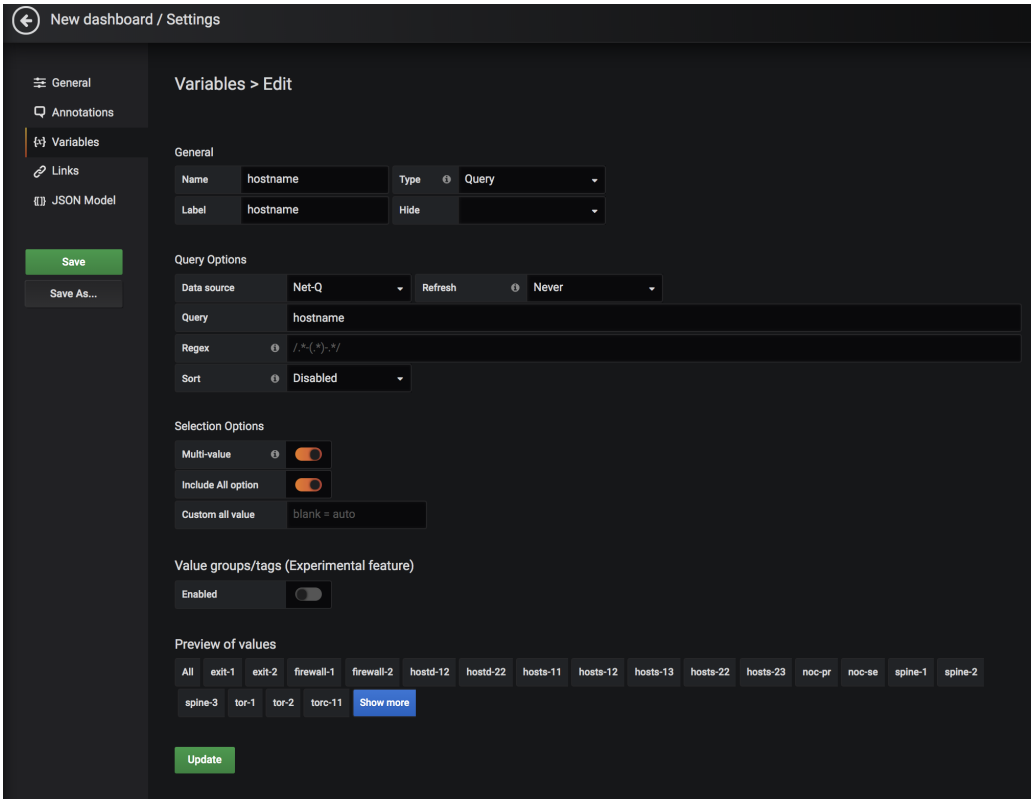
To use the Cumulus-provided dashboard, select the *NetQ Interface Statistics* from the left panel of the Home Page.



If you choose this option, you can skip directly to analyzing your data.

To create your own dashboard:

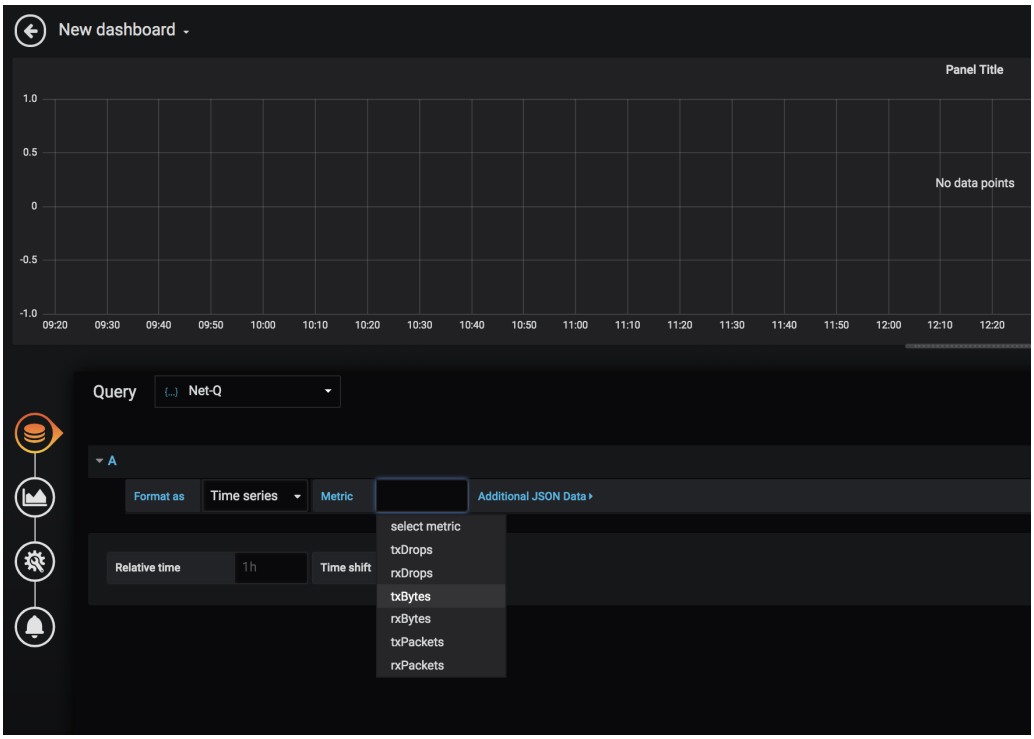
1. Click
   
   to open a blank dashboard.
2. Click
   
   (Dashboard Settings) at the top of the dashboard.
3. Click **Variables**.



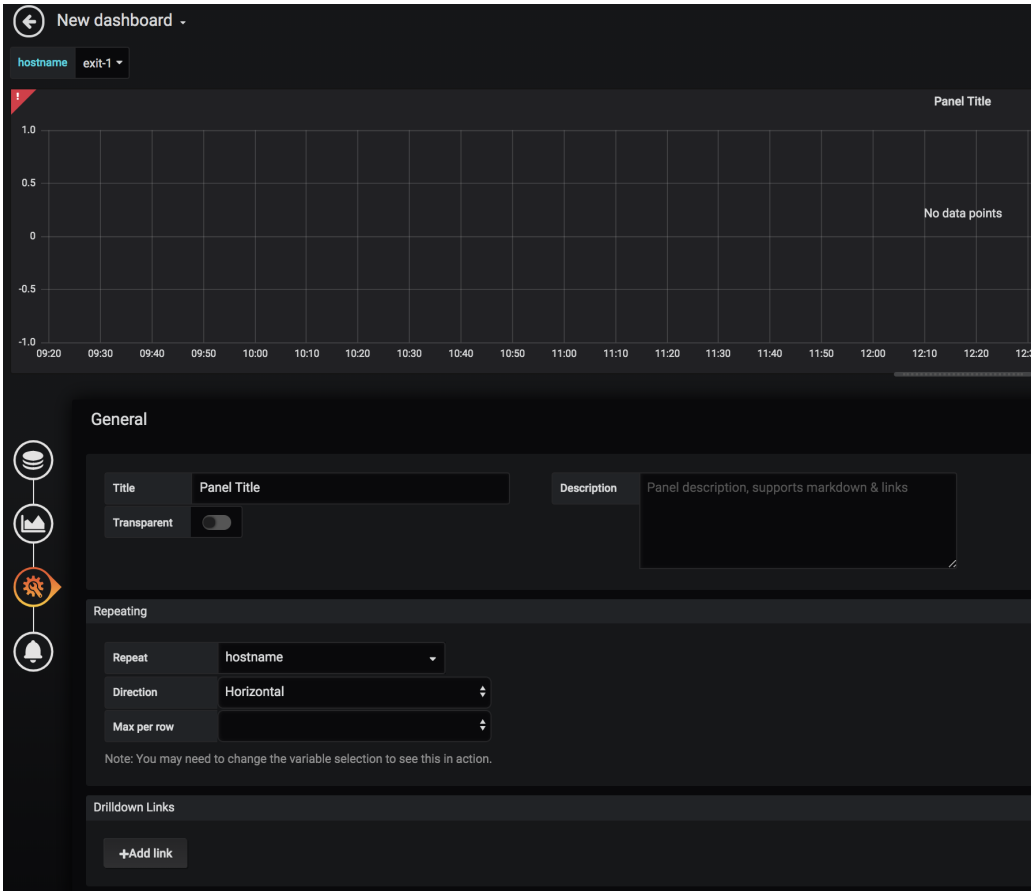

4. Enter *hostname* into the **Name** field.
5. Enter *Hostname* into the **Label** field.
6. Select *Net-Q* from the **Data source** list.
7. Enter *hostname* into the **Query** field.
8. Click **Add**.

You should see a preview at the bottom of the hostname values.

9. Click
   
   to return to the new dashboard.
10. Click **Add Query**.




11. Select *Net-Q* from the **Query** source list.
12. Select the interface statistic you want to view from the **Metric** list.
13. Click the **General** icon.

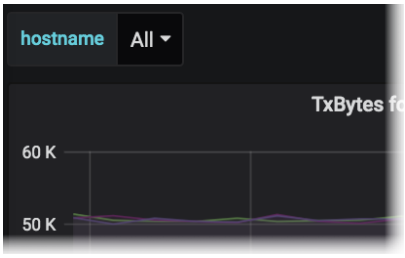



14. Select *hostname* from the **Repeat** list.
15. Set any other parameters around how to display the data.
16. Return to the dashboard.
17. Add additional panels with other metrics to complete your dashboard.

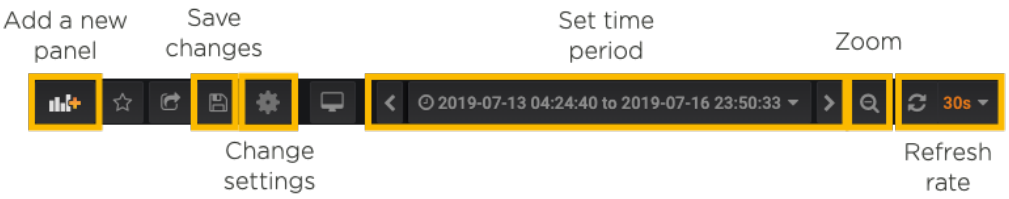
## Analyze the Data

Once you have your dashboard configured, you can start analyzing the data:

1. Select the hostname from the variable list at the top left of the charts to see the statistics for that switch or host.



2. Review the statistics, looking for peaks and valleys, unusual patterns, and so forth.
3. Explore the data more by modifying the data view in one of several ways using the dashboard tool set:




- Select a different time period for the data by clicking the forward or back arrows. The default time range is dependent on the width of your browser window.
- Zoom in on the dashboard by clicking the magnifying glass.
- Manually refresh the dashboard data, or set an automatic refresh rate for the dashboard from the down arrow.
- Add a new variable by clicking the cog wheel, then selecting Variables
- Add additional panels
- Click any chart title to edit or remove it from the dashboard
- Rename the dashboard by clicking the cog wheel and entering the new name

# Integrate with Hardware Chassis

## Integrate with a Hardware Chassis

NetQ can run within a Facebook Backpack chassis, Cumulus Express CX-10256-S chassis or Edgecore OMP-800 chassis.

Keep the following issues in mind if you intend to use NetQ with a chassis:

- You must assign a unique hostname to every node that runs the NetQ Agent. By default, all the fabric cards in the chassis have the same hostname.
- The NetQ Agent must be installed on every line card.
- No information is returned about the ASIC when you run `netq show inventory asic`. This is a known issue.
- Since the chassis sensor information is shared, every line card and fabric card can report the same sensor data. By default, sensor data is disabled on a chassis to avoid this duplication . To enable sensor data on a line card, edit `/etc/netq/netq.yml` or `/etc/netq/config.d/user.yml` and set the `send_chassis_sensor_data` keyword to *true*, then restart the NetQ Agent with `netq config agent restart`. Configuring NetQ in this way prevents any duplication of data in the NetQ database.

```
cumulus@chassis:~$ sudo nano /etc/netq/netq.yml


...
netq-agent:
```

```
  send_chassis_sensor_data: true
...
```

# Cumulus NetQ API User Guide

The NetQ API provides access to key telemetry and system monitoring data gathered about the performance and operation of your data center network and devices so that you can view that data in your internal or third-party analytic tools. The API gives you access to the health of individual switches, network protocols and services, and views of network-wide inventory and events.

This guide provides an overview of the API framework and some examples of how to use the API to extract the data you need. Descriptions of each endpoint and model parameter are contained in the API .json files.

For information regarding new features, improvements, bug fixes, and known issues present in this release, refer to the release notes.

This Cumulus NetQ API User Guide is available in PDF for offline viewing.

## API Organization

The Cumulus NetQ API provides endpoints for:

- **Network routing protocols**: BGP, EVPN, LLDP, CLAG, MSTP, Neighbors, NTP, Routes
- **Virtual networks**: VLAN
- **Services**: Services
- **Interfaces**: Interface, Port
- **Inventory and Devices**: Address, Inventory, MAC Address tables, Node, Sensors
- **Events**: Events

Each endpoint has its own API. You can make requests for all data and all devices or you can filter the request by a given hostname.

Each API returns a predetermined set of data as defined in the API models.

## Get Started

You can access the API gateway and execute requests from a terminal interface against your NetQ Platform or NetQ Appliance through port 32708.

### Log In and Authentication

Use your login credentials that were provided as part of the installation process. For this release, the default is username *admin* and password *admin*.

To log in and obtain authorization:

1. Open a terminal window.
2. Enter the following curl command.

```
<computer-name>:~ <username>$ curl --insecure -X POST "https://
<netq.domain>:32708/netq/auth/v1/login" -H "Content-Type: application/json" -
d "{"username":"admin","password":"admin"}"
{"premises":[{"opid":
0,"name":"OPID0"}],"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoiYWRtaW4iL
0,"id":"admin","expires_at":1556125378280,"terms_of_use_accepted":true}
```

3. Copy the access token for use in making data requests.

### API Requests

We will use curl to execute our requests. Each request contains an API method (GET, POST, etc.), the address and API object to query, a variety of headers, and sometimes a body. In the log in step you used above:

- API method = POST
- Address and API object = “https://<netq.domain>:32708/netq/auth/v1/login”
- Headers = -H “Content-Type: application/json”
- Body = -d “{“username”:“admin”,“password”:“admin”}”

> ⓘ NOTE
>
> We have used the *insecure* option to work around any certificate issues with our development configuration. You would likely not use this option.

### API Responses

A NetQ API response is comprised of a status code, any relevant error codes (if unsuccessful), and the collected data (if successful).

The following HTTP status codes might be presented in the API responses:

| Code | Name | Description | Action |
|---|---|---|---|
| 200 | Success | Request was successfully processed. | Review response |
| 400 | Bad Request | Invalid input was detected in request. | Check the syntax of your request and make sure it matches the schema |

| Code | Name | Description | Action |
|---|---|---|---|
| 401 | Unauthorized | Authentication has failed or credentials were not provided. | Provide or verify your credentials, or request access from your administrator |
| 403 | Forbidden | Request was valid, but user may not have needed permissions. | Verify your credentials or request an account from your administrator |
| 404 | Not Found | Requested resource could not be found. | Try the request again after a period of time or verify status of resource |
| 409 | Conflict | Request cannot be processed due to conflict in current state of the resource. | Verify status of resource and remove conflict |
| 500 | Internal Server Error | Unexpected condition has occurred. | Perform general troubleshooting and try the request again |
| 503 | Service Unavailable | The service being requested is currently unavailable. | Verify the status of the NetQ Platform or Appliance, and the associated service |

## Example Requests and Responses

Some command requests and their responses are shown here, but feel free to run your own requests. To run a request, you will need your authorization token. We have piped our responses through a python tool to make the responses more readable. You may chose to do so as well or not.

To view all of the endpoints and their associated requests and responses, refer to View the API.

### Get Network-wide Status of the BGP Service

Make your request to the *bgp* endpoint to obtain status information from all nodes running the BGP service, as follows:

```
curl --insecure -X GET “<https://<netq.domain>:32708/netq/telemetry/v1/object/bgp” -H “Content-Type: application/json” -H “Authorization: <auth-token>” | python -m json.tool

[
 {
   "ipv6_pfx_rcvd": 0,
   "peer_router_id": "0.0.0.0",
   "objid": "",
   "upd8_tx": 0,
   "hostname": "exit-1",
   "timestamp": 1556037420723,
   "peer_asn": 0,
   "state": "NotEstd",
   "vrf": "DataVrf1082",
   "rx_families": [],
   "ipv4_pfx_rcvd": 0,
   "conn_dropped": 0,
   "db_state": "Update",
   "up_time": 0,
   "last_reset_time": 0,
   "tx_families": [],
```

```
    "reason": "N/A",
    "vrfid": 13,
    "asn": 655536,
    "opid": 0,
    "peer_hostname": "",
    "upd8_rx": 0,
    "peer_name": "swp7.4",
    "evpn_pfx_rcvd": 0,
    "conn_estd": 0
  },
  {
    "ipv6_pfx_rcvd": 0,
    "peer_router_id": "0.0.0.0",
    "objid": "",
    "upd8_tx": 0,
    "hostname": "exit-1",
    "timestamp": 1556037420674,
    "peer_asn": 0,
    "state": "NotEstd",
    "vrf": "default",
    "rx_families": [],
    "ipv4_pfx_rcvd": 0,
    "conn_dropped": 0,
    "db_state": "Update",
    "up_time": 0,
    "last_reset_time": 0,
    "tx_families": [],
    "reason": "N/A",
    "vrfid": 0,
```

```
    "asn": 655536,
    "opid": 0,
    "peer_hostname": "",
    "upd8_rx": 0,
    "peer_name": "swp7",
    "evpn_pfx_rcvd": 0,
    "conn_estd": 0
  },
  {
    "ipv6_pfx_rcvd": 24,
    "peer_router_id": "27.0.0.19",
    "objid": "",
    "upd8_tx": 314,
    "hostname": "exit-1",
    "timestamp": 1556037420665,
    "peer_asn": 655435,
    "state": "Established",
    "vrf": "default",
    "rx_families": [
      "ipv4",
      "ipv6",
      "evpn"
    ],
    "ipv4_pfx_rcvd": 26,
    "conn_dropped": 0,
    "db_state": "Update",
    "up_time": 1556036850000,
    "last_reset_time": 0,
    "tx_families": [
```

```
      "ipv4",
      "ipv6",
      "evpn"
    ],
    "reason": "N/A",
    "vrfid": 0,
    "asn": 655536,
    "opid": 0,
    "peer_hostname": "spine-1",
    "upd8_rx": 321,
    "peer_name": "swp3",
    "evpn_pfx_rcvd": 354,
    "conn_estd": 1
  },
...
```

### Get Status of EVPN on a Specific Switch

Make your request to the *evpn/hostname* endpoint to view the status of all EVPN sessions running on that node. This example uses the *server01* node.

```
curl -X GET "https://<netq.domain>:32708/netq/telemetry/v1/object/evpn/
hostname/server01" -H "Content-Type: application/json" -H "Authorization: <auth-
token>" | python -m json.tool

[
  {
    "import_rt": "[\"197:42\"]",
    "vni": 42,
```

```
    "rd": "27.0.0.22:2",
    "hostname": "server01",
    "timestamp": 1556037403853,
    "adv_all_vni": true,
    "export_rt": "[\"197:42\"]",
    "db_state": "Update",
    "in_kernel": true,
    "adv_gw_ip": "Disabled",
    "origin_ip": "27.0.0.22",
    "opid": 0,
    "is_l3": false
  },
  {
    "import_rt": "[\"197:37\"]",
    "vni": 37,
    "rd": "27.0.0.22:8",
    "hostname": "server01",
    "timestamp": 1556037403811,
    "adv_all_vni": true,
    "export_rt": "[\"197:37\"]",
    "db_state": "Update",
    "in_kernel": true,
    "adv_gw_ip": "Disabled",
    "origin_ip": "27.0.0.22",
    "opid": 0,
    "is_l3": false
  },
  {
    "import_rt": "[\"197:4001\"]",
```

```
    "vni": 4001,
    "rd": "6.0.0.194:5",
    "hostname": "server01",
    "timestamp": 1556036360169,
    "adv_all_vni": true,
    "export_rt": "[\"197:4001\"]",
    "db_state": "Refresh",
    "in_kernel": true,
    "adv_gw_ip": "Disabled",
    "origin_ip": "27.0.0.22",
    "opid": 0,
    "is_l3": true
  },
...
```

### Get Status on All Interfaces at a Given Time

Make your request to the *interfaces* endpoint to view the status of all interfaces. By specifying the *eq-timestamp* option and entering a date and time in epoch format, you indicate the data for that time (versus in the last hour by default), as follows:

```
curl -X GET "https://<netq.domain>:32708/netq/telemetry/v1/object/interface?
eq_timestamp=1556046250" -H "Content-Type: application/json" -H
"Authorization: <auth-token>" | python -m json.tool

[
  {
    "hostname": "exit-1",
    "timestamp": 1556046270494,
```

```
    "state": "up",
    "vrf": "DataVrf1082",
    "last_changed": 1556037405259,
    "ifname": "swp3.4",
    "opid": 0,
    "details": "MTU: 9202",
    "type": "vlan"
  },
  {
    "hostname": "exit-1",
    "timestamp": 1556046270496,
    "state": "up",
    "vrf": "DataVrf1081",
    "last_changed": 1556037405320,
    "ifname": "swp7.3",
    "opid": 0,
    "details": "MTU: 9202",
    "type": "vlan"
  },
  {
    "hostname": "exit-1",
    "timestamp": 1556046270497,
    "state": "up",
    "vrf": "DataVrf1080",
    "last_changed": 1556037405310,
    "ifname": "swp7.2",
    "opid": 0,
    "details": "MTU: 9202",
    "type": "vlan"
```

```
  },
  {
    "hostname": "exit-1",
    "timestamp": 1556046270499,
    "state": "up",
    "vrf": "",
    "last_changed": 1556037405315,
    "ifname": "DataVrf1081",
    "opid": 0,
    "details": "table: 1081, MTU: 65536, Members:  swp7.3,  DataVrf1081,  swp4.3,
swp6.3,  swp5.3,  swp3.3, ",
    "type": "vrf"
  },
...
```

### Get a List of All Devices Being Monitored

Make your request to the *inventory* endpoint to get a listing of all monitored nodes and their configuration information, as follows:

```
curl -X GET "https://<netq.domain>:32708/netq/telemetry/v1/object/inventory" -H
"Content-Type: application/json" -H "Authorization: <auth-token>" | python -m
json.tool

[
  {
    "hostname": "exit-1",
    "timestamp": 1556037425658,
    "asic_model": "A-Z",
```

```
    "agent_version": "2.1.1-cl3u16~1556035513.afedb69",
    "os_version": "A.2.0",
    "license_state": "ok",
    "disk_total_size": "10 GB",
    "os_version_id": "A.2.0",
    "platform_model": "A_VX",
    "memory_size": "2048.00 MB",
    "asic_vendor": "AA Inc",
    "cpu_model": "A-SUBLEQ",
    "asic_model_id": "N/A",
    "platform_vendor": "A Systems",
    "asic_ports": "N/A",
    "cpu_arch": "x86_64",
    "cpu_nos": "2",
    "platform_mfg_date": "N/A",
    "platform_label_revision": "N/A",
    "agent_state": "fresh",
    "cpu_max_freq": "N/A",
    "platform_part_number": "3.7.6",
    "asic_core_bw": "N/A",
    "os_vendor": "CL",
    "platform_base_mac": "00:01:00:00:01:00",
    "platform_serial_number": "00:01:00:00:01:00"
  },
  {
    "hostname": "exit-2",
    "timestamp": 1556037432361,
    "asic_model": "C-Z",
    "agent_version": "2.1.1-cl3u16~1556035513.afedb69",
```

```
    "os_version": "C.2.0",
    "license_state": "N/A",
    "disk_total_size": "30 GB",
    "os_version_id": "C.2.0",
    "platform_model": "C_VX",
    "memory_size": "2048.00 MB",
    "asic_vendor": "CC Inc",
    "cpu_model": "C-CRAY",
    "asic_model_id": "N/A",
    "platform_vendor": "C Systems",
    "asic_ports": "N/A",
    "cpu_arch": "x86_64",
    "cpu_nos": "2",
    "platform_mfg_date": "N/A",
    "platform_label_revision": "N/A",
    "agent_state": "fresh",
    "cpu_max_freq": "N/A",
    "platform_part_number": "3.7.6",
    "asic_core_bw": "N/A",
    "os_vendor": "CL",
    "platform_base_mac": "00:01:00:00:02:00",
    "platform_serial_number": "00:01:00:00:02:00"
  },
  {
    "hostname": "firewall-1",
    "timestamp": 1556037438002,
    "asic_model": "N/A",
    "agent_version": "2.1.0-ub16.04u15~1555608012.1d98892",
    "os_version": "16.04.1 LTS (Xenial Xerus)",
```

```
    "license_state": "N/A",
    "disk_total_size": "3.20 GB",
    "os_version_id": "(hydra-poc-01 /tmp/purna/Kleen-Gui1/)\"16.04",
    "platform_model": "N/A",
    "memory_size": "4096.00 MB",
    "asic_vendor": "N/A",
    "cpu_model": "QEMU Virtual  version 2.2.0",
    "asic_model_id": "N/A",
    "platform_vendor": "N/A",
    "asic_ports": "N/A",
    "cpu_arch": "x86_64",
    "cpu_nos": "2",
    "platform_mfg_date": "N/A",
    "platform_label_revision": "N/A",
    "agent_state": "fresh",
    "cpu_max_freq": "N/A",
    "platform_part_number": "N/A",
    "asic_core_bw": "N/A",
    "os_vendor": "Ubuntu",
    "platform_base_mac": "N/A",
    "platform_serial_number": "N/A"
  },
...
```


## View the API

For simplicity, all of the endpoint APIs are combined into a single json-formatted file. There have been no changes to the file in the NetQ 2.3.0 release.

- netq-220.json


```
{
  "swagger": "2.0",
  "info": {
    "description": "This API is used to gain access to data collected by the Cumulus NetQ Platform and Agents for integration with third-party monitoring and analytics  software. Integrators can pull data for daily monitoring of network protocols and services performance, inventory status, and system-wide events.",
    "version": "1.0",
    "title": "Cumulus NetQ 2.2.0 API",
    "termsOfService": "https://cumulusnetworks.com/legal/"
  },
  "host": "<netq-platform-or-appliance-ipaddress>:32708",
  "basePath": "/netq/telemetry/v1",
  "externalDocs": {
    "description": "API Documentation",
    "url": "https://docs.cumulusnetworks.com/display/NETQ/Cumulus+NetQ+API+User+Guide"
  },
  "schemes": [
    "https"
  ],
  "paths": {
    "/object/address": {
      "get": {
        "tags": [
          "address"
        ],
```

```
    "summary": "Get all addresses for all network devices",
    "description": "Retrieves all IPv4, IPv6 and MAC addresses deployed on switches and hosts in your network running NetQ Agents.",
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "eq_timestamp",
        "in": "query",
        "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "count",
        "in": "query",
        "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "offset",
        "in": "query",
        "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would
```

```
display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Address"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
  "/object/address/hostname/{hostname}": {
    "get": {
      "tags": [
        "address"
      ],
      "summary": "Get all addresses for a given network device by hostname",
```

```
    "description": "Retrieves IPv4, IPv6, and MAC addresses of a network device (switch or host) specified by its hostname.",
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "hostname",
        "in": "path",
        "description": "User-specified name for a network switch or host. For example, leaf01, spine04, host-6, engr-1, tor-22.",
        "required": true,
        "type": "string"
      },
      {
        "name": "eq_timestamp",
        "in": "query",
        "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "count",
        "in": "query",
        "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
        "required": false,
```

```
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Address"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
```

```
  },
  "/object/login": {
    "post": {
      "tags": [
        "auth"
      ],
      "summary": "Perform authenticated user login to NetQ",
      "description": "Sends user-provided login credenentials (username and password) to the NetQ Authorization service for validation. Grants access to the NetQ platform and software if user credentials are valid.",
      "operationId": "login",
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "in": "body",
          "name": "body",
          "description": "User credentials provided for login request; username and password.",
          "required": true,
          "schema": {
            "$ref": "#/definitions/LoginRequest"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
```

```
          "schema": {
            "$ref": "#/definitions/LoginResponse"
          }
        },
        "401": {
          "description": "Invalid credentials",
          "schema": {
            "$ref": "#/definitions/ErrorResponse"
          }
        }
      }
    },
    "/object/bgp": {
      "get": {
        "tags": [
          "bgp"
        ],
        "summary": "Get all BGP session information for all network devices",
        "description": "For every Border Gateway Protocol (BGP) session running on the network, retrieves local node hostname, remote peer hostname, interface, router ID, and ASN, timestamp, VRF, connection state, IP and EVPN prefixes, and so forth. Refer to the BGPSession model for all data collected.",
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "eq_timestamp",
```

```
          "in": "query",
          "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
```

```
            "type": "array",
            "items": {
              "$ref": "#/definitions/BgpSession"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
  "/object/bgp/hostname/{hostname}": {
    "get": {
      "tags": [
        "bgp"
      ],
      "summary": "Get all BGP session information for a given network device by hostname",
      "description": "For every BGP session running on the network device, retrieves local node hostname, remote peer hostname, interface, router ID, and ASN, timestamp, VRF, connection state, IP and EVPN prefixes, and so forth. Refer to the BGPSession model for all data collected.",
      "produces": [
        "application/json"
      ],
      "parameters": [
```

```
        {
          "name": "hostname",
          "in": "path",
          "description": "User-specified name for a network switch or host. For
example, leaf01, spine04, host-6, engr-1, tor-22.",
          "required": true,
          "type": "string"
        },
        {
          "name": "eq_timestamp",
          "in": "query",
          "description": "Display results for a given time. Time must be entered in
Epoch format. For example, to display the results for Monday February 13, 2019 at
1:25 pm, use a time converter and enter 1550082300.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value.
For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting
```

location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
"required": false,
"type": "integer"
}
],
"responses": {
"200": {
"description": "successful operation",
"schema": {
"type": "array",
"items": {
"$ref": "#/definitions/BgpSession"
}
}
}
},
"security": [
{
"jwt": []
}
]
}
},
"/object/clag": {
"get": {
"tags": [
"clag"
],

```
    "summary": "Get all CLAG session information for all network devices",
    "description": "For every Cumulus multiple Link Aggregation (CLAG) session running on the network, retrieves local node hostname, CLAG sysmac, remote peer role, state, and interface, backup IP address, bond status, and so forth. Refer to the ClagSessionInfo model for all data collected.",
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "eq_timestamp",
        "in": "query",
        "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "count",
        "in": "query",
        "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "offset",
        "in": "query",
```

```
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/ClagSessionInfo"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
  "/object/clag/hostname/{hostname}": {
    "get": {
      "tags": [
        "clag"
```

```
    ],
    "summary": "Get all CLAG session information for a given network device by hostname",
    "description": "For every CLAG session running on the network device, retrieves local node hostname, CLAG sysmac, remote peer role, state, and interface, backup IP address, bond status, and so forth. Refer to the ClagSessionInfo model for all data collected.",
    "produces": [
     "application/json"
    ],
    "parameters": [
     {
      "name": "hostname",
      "in": "path",
      "description": "User-specified name for a network switch or host. For example, leaf01, spine04, host-6, engr-1, tor-22.",
      "required": true,
      "type": "string"
     },
     {
      "name": "eq_timestamp",
      "in": "query",
      "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
      "required": false,
      "type": "integer"
     },
     {
```

```
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/ClagSessionInfo"
            }
          }
        }
      },
      "security": [
```

```
        {
          "jwt": []
        }
       ]
     }
    },
    "/object/events": {
     "get": {
       "tags": [
         "events"
       ],
       "summary": "Get all events from across the entire network",
       "description": "Retrieves all alarm (critical severity) and informational
(warning, info and debug severity) events from all network devices and services.",
       "produces": [
         "application/json"
       ],
       "parameters": [
        {
          "name": "gt_timestamp",
          "in": "query",
          "description": "Used in combination with lt_timestamp, sets the lower limit
of the timerange to display. Uses Epoch format. Cannot be used with
eq_timestamp. For example, to display events between Monday February 11, 2019
at 1:00am and Tuesday February 12, 2019 at 1:00am, lt_timestamp would be
entered as 1549864800 and gt_timestamp would be entered as 1549951200.",
          "required": false,
          "type": "integer"
        },
```

```
        {
          "name": "lt_timestamp",
          "in": "query",
          "description": "Used in combination with gt_timestamp, sets the upper limit of the timerange to display. Uses Epoch format. Cannot be used with eq_timestamp. For example, to display events between Monday February 11, 2019 at 1:00am and Tuesday February 12, 2019 at 1:00am, lt_timestamp would be entered as 1549864800 and gt_timestamp would be entered as 1549951200.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "eq_timestamp",
          "in": "query",
          "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
```

```
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "type": "string"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
  "/object/evpn": {
    "get": {
```

```
      "tags": [
        "evpn"
      ],
      "summary": "Get all EVPN session information from across the entire network",
      "description": "For every Ethernet Virtual Private Network (EVPN) session running on the network, retrieves hostname, VNI status, origin IP address, timestamp, export and import routes, and so forth. Refer to the Evpn model for all data collected.",
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "eq_timestamp",
          "in": "query",
          "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
```

```
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Evpn"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
```

```
  "/object/evpn/hostname/{hostname}": {
    "get": {
      "tags": [
        "evpn"
      ],
      "summary": "Get all EVPN session information from a given network device by hostname",
      "description": "For every EVPN session running on the network device, retrieves hostname, VNI status, origin IP address, timestamp, export and import routes, and so forth. Refer to the Evpn model for all data collected.",
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "hostname",
          "in": "path",
          "description": "User-specified name for a network switch or host. For example, leaf01, spine04, host-6, engr-1, tor-22.",
          "required": true,
          "type": "string"
        },
        {
          "name": "eq_timestamp",
          "in": "query",
          "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
          "required": false,
```

```
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Evpn"
            }
          }
```

```
        }
       },
       "security": [
        {
          "jwt": []
        }
       ]
      }
    },
    "/object/interface": {
     "get": {
       "tags": [
         "interface"
       ],
       "summary": "Get software interface information for all network devices",
       "description": "Retrieves information about all software interfaces, including type and name of the interfaces, the hostnames of the device where they reside, state, VRF, and so forth. Refer to the Interface model for all data collected.",
       "produces": [
         "application/json"
       ],
       "parameters": [
        {
          "name": "eq_timestamp",
          "in": "query",
          "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
          "required": false,
```

```
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Interface"
            }
          }
```

```
        }
       },
       "security": [
        {
         "jwt": []
        }
       ]
      }
    },
    "/object/interface/hostname/{hostname}": {
     "get": {
      "tags": [
       "interface"
      ],
      "summary": "Get software interface information for a given network device by hostname",
      "description": "Retrieves information about all software interfaces on a network device, including type and name of the interfaces, state, VRF, and so forth. Refer to the Interface model for all data collected.",
      "produces": [
       "application/json"
      ],
      "parameters": [
       {
        "name": "hostname",
        "in": "path",
        "description": "User-specified name for a network switch or host. For example, leaf01, spine04, host-6, engr-1, tor-22.",
        "required": true,
```

```
        "type": "string"
      },
      {
        "name": "eq_timestamp",
        "in": "query",
        "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "count",
        "in": "query",
        "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "offset",
        "in": "query",
        "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
        "required": false,
        "type": "integer"
      }
    ],
```

```
        "responses": {
          "200": {
            "description": "successful operation",
            "schema": {
              "type": "array",
              "items": {
                "$ref": "#/definitions/Interface"
              }
            }
          }
        },
        "security": [
          {
            "jwt": []
          }
        ]
      }
    },
    "/object/inventory": {
      "get": {
        "tags": [
          "inventory"
        ],
        "summary": "Get component inventory information from all network devices",
        "description": "Retrieves the hardware and software component information, such as ASIC, platform, and OS vendor and version information, for all switches and hosts in your network. Refer to the InventoryOutput model for all data collected.",
        "produces": [
```

```
        "application/json"
      ],
      "parameters": [
       {
        "name": "eq_timestamp",
        "in": "query",
        "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
        "required": false,
        "type": "integer"
       },
       {
        "name": "count",
        "in": "query",
        "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
        "required": false,
        "type": "integer"
       },
       {
        "name": "offset",
        "in": "query",
        "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
        "required": false,
        "type": "integer"
       }
```

```
        ],
        "responses": {
          "200": {
            "description": "successful operation",
            "schema": {
              "$ref": "#/definitions/InventoryOutput"
            }
          },
          "400": {
            "description": "Invalid Input"
          }
        },
        "security": [
          {
            "jwt": []
          }
        ]
      }
    },
    "/object/inventory/hostname/{hostname}": {
      "get": {
        "tags": [
          "inventory"
        ],
        "summary": "Get component inventory information from a given network device by hostname",
        "description": "Retrieves the hardware and software component information, such as ASIC, platform, and OS vendor and version information, for the given switchor host in your network. Refer to the InventoryOutput model for all data
```

collected.",
"produces": [
"application/json"
],
"parameters": [
{
"name": "hostname",
"in": "path",
"description": "User-specified name for a network switch or host. For example, leaf01, spine04, host-6, engr-1, tor-22.",
"required": true,
"type": "string"
},
{
"name": "eq_timestamp",
"in": "query",
"description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
"required": false,
"type": "integer"
},
{
"name": "count",
"in": "query",
"description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
"required": false,
"type": "integer"

```
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "$ref": "#/definitions/InventoryOutput"
          }
        },
        "400": {
          "description": "Invalid Input"
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
```

```
  "/object/lldp": {
    "get": {
      "tags": [
        "lldp"
      ],
      "summary": "Get LLDP information for all network devices",
      "description": "Retrieves Link Layer Discovery Protocol (LLDP) information, such as hostname, interface name, peer hostname, interface name, bridge, router, OS, timestamp, for all switches and hosts in the network. Refer to the LLDP model for all data collected.",
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "eq_timestamp",
          "in": "query",
          "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
```

```
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/LLDP"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
```

```
  },
  "/object/lldp/hostname/{hostname}": {
   "get": {
    "tags": [
     "lldp"
    ],
    "summary": "Get LLDP information for a given network device by hostname",
    "description": "Retrieves Link Layer Discovery Protocol (LLDP) information,
such as hostname, interface name, peer hostname, interface name, bridge,
router, OS, timestamp, for the given switch or host. Refer to the LLDP model for all
data collected.",
    "produces": [
     "application/json"
    ],
    "parameters": [
     {
      "name": "hostname",
      "in": "path",
      "description": "User-specified name for a network switch or host. For
example, leaf01, spine04, host-6, engr-1, tor-22.",
      "required": true,
      "type": "string"
     },
     {
      "name": "eq_timestamp",
      "in": "query",
      "description": "Display results for a given time. Time must be entered in
Epoch format. For example, to display the results for Monday February 13, 2019 at
1:25 pm, use a time converter and enter 1550082300.",
```

```
          "required": false,
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/LLDP"
            }
```

```
          }
         }
        },
        "security": [
         {
          "jwt": []
         }
        ]
       }
      },
      "/object/macfdb": {
       "get": {
        "tags": [
         "macfdb"
        ],
        "summary": "Get all MAC FDB information for all network devices",
        "description": "Retrieves all MAC address forwarding database (MACFDB) information for all switches and hosts in the network, such as MAC address, timestamp, next hop, destination, port, and VLAN. Refer to MacFdb model for all data collected.",
        "produces": [
         "application/json"
        ],
        "parameters": [
         {
          "name": "eq_timestamp",
          "in": "query",
          "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at
```

```
1:25 pm, use a time converter and enter 1550082300.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value.
For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting
location within the set of entries returned. For example, an offset of 100 would
display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/MacFdb"
```

```
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
  "/object/macfdb/hostname/{hostname}": {
    "get": {
      "tags": [
        "macfdb"
      ],
      "summary": "Get all MAC FDB information for a given network device by hostname",
      "description": "Retrieves all MAC address forwarding database (MACFDB) information for a given switch or host in the network, such as MAC address, timestamp, next hop, destination, port, and VLAN. Refer to MacFdb model for all data collected.",
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "hostname",
          "in": "path",
```

```
        "description": "User-specified name for a network switch or host. For example, leaf01, spine04, host-6, engr-1, tor-22.",
        "required": true,
        "type": "string"
      },
      {
        "name": "eq_timestamp",
        "in": "query",
        "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "count",
        "in": "query",
        "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "offset",
        "in": "query",
        "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
        "required": false,
```

```
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/MacFdb"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
  "/object/mstp": {
    "get": {
      "tags": [
        "mstp"
      ],
      "summary": "Get all MSTP information from all network devices",
      "description": "Retrieves all Multiple Spanning Tree Protocol (MSTP)
information, including bridge and port information, changes made to topology,
```

and so forth for all switches and hosts in the network. Refer to MstpInfo model for all data collected.",
"produces": [
"application/json"
],
"parameters": [
{
"name": "eq_timestamp",
"in": "query",
"description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
"required": false,
"type": "integer"
},
{
"name": "count",
"in": "query",
"description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
"required": false,
"type": "integer"
},
{
"name": "offset",
"in": "query",
"description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",

```
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/MstpInfo"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
  "/object/mstp/hostname/{hostname}": {
    "get": {
      "tags": [
        "mstp"
      ],
      "summary": "Get all MSTP information from a given network device by
hostname",
```

```
    "description": "Retrieves all MSTP information, including bridge and port information, changes made to topology, and so forth for a given switch or host in the network.  Refer to MstpInfo model for all data collected.",
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "hostname",
        "in": "path",
        "description": "User-specified name for a network switch or host. For example, leaf01, spine04, host-6, engr-1, tor-22.",
        "required": true,
        "type": "string"
      },
      {
        "name": "eq_timestamp",
        "in": "query",
        "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "count",
        "in": "query",
        "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
```

```
        "required": false,
        "type": "integer"
      },
      {
        "name": "offset",
        "in": "query",
        "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
        "required": false,
        "type": "integer"
      }
    ],
    "responses": {
      "200": {
        "description": "successful operation",
        "schema": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/MstpInfo"
          }
        }
      }
    },
    "security": [
      {
        "jwt": []
      }
    ]
```

```
    }
   },
   "/object/neighbor": {
    "get": {
     "tags": [
      "neighbor"
     ],
     "summary": "Get neighbor information for all network devices",
     "description": "Retrieves neighbor information, such as hostname, addresses, VRF, interface name and index, for all switches and hosts in the network.  Refer to Neighbor model for all data collected.",
     "produces": [
      "application/json"
     ],
     "parameters": [
      {
       "name": "eq_timestamp",
       "in": "query",
       "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
       "required": false,
       "type": "integer"
      },
      {
       "name": "count",
       "in": "query",
       "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
```

```
        "required": false,
        "type": "integer"
      },
      {
        "name": "offset",
        "in": "query",
        "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
        "required": false,
        "type": "integer"
      }
    ],
    "responses": {
      "200": {
        "description": "successful operation",
        "schema": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/Neighbor"
          }
        }
      }
    },
    "security": [
      {
        "jwt": []
      }
    ]
```

```
    }
   },
   "/object/neighbor/hostname/{hostname}": {
    "get": {
     "tags": [
      "neighbor"
     ],
     "summary": "Get neighbor information for a given network device by
hostname",
     "description": "Retrieves neighbor information, such as hostname, addresses,
VRF, interface name and index, for a given switch or host in the network.  Refer to
Neighbor model for all data collected.",
     "produces": [
      "application/json"
     ],
     "parameters": [
      {
       "name": "hostname",
       "in": "path",
       "description": "User-specified name for a network switch or host. For
example, leaf01, spine04, host-6, engr-1, tor-22.",
       "required": true,
       "type": "string"
      },
      {
       "name": "eq_timestamp",
       "in": "query",
       "description": "Display results for a given time. Time must be entered in
Epoch format. For example, to display the results for Monday February 13, 2019 at
```

```
1:25 pm, use a time converter and enter 1550082300.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value.
For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting
location within the set of entries returned. For example, an offset of 100 would
display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Neighbor"
```

```
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
  "/object/node": {
    "get": {
      "tags": [
        "node"
      ],
      "summary": "Get device status for all network devices",
      "description": "Retrieves hostname, uptime, last update, boot and reinitialization time, version, NTP and DB state, timestamp, and its current state (active or not) for all switches and hosts in the network.",
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "eq_timestamp",
          "in": "query",
          "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at
```

```
1:25 pm, use a time converter and enter 1550082300.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value.
For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting
location within the set of entries returned. For example, an offset of 100 would
display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/NODE"
```

```
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
  "/object/node/hostname/{hostname}": {
    "get": {
      "tags": [
        "node"
      ],
      "summary": "Get device status for a given network device by hostname",
      "description": "Retrieves hostname, uptime, last update, boot and reinitialization time, version, NTP and DB state, timestamp, and its current state (active or not) for a given switch or host in the network.",
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "hostname",
          "in": "path",
          "description": "User-specified name for a network switch or host. For example, leaf01, spine04, host-6, engr-1, tor-22.",
```

```
        "required": true,
        "type": "string"
      },
      {
        "name": "eq_timestamp",
        "in": "query",
        "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "count",
        "in": "query",
        "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "offset",
        "in": "query",
        "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
        "required": false,
        "type": "integer"
      }
```

```
        ],
        "responses": {
          "200": {
            "description": "successful operation",
            "schema": {
              "type": "array",
              "items": {
                "$ref": "#/definitions/NODE"
              }
            }
          }
        },
        "security": [
          {
            "jwt": []
          }
        ]
      }
    },
    "/object/ntp": {
      "get": {
        "tags": [
          "ntp"
        ],
        "summary": "Get all NTP information for all network devices",
        "description": "Retrieves all Network Time Protocol (NTP) configuration and status information, such as whether the service is running and if it is in time synchronization, for all switches and hosts in the network. Refer to the NTP model for all data collected.",
```

```
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "eq_timestamp",
        "in": "query",
        "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "count",
        "in": "query",
        "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "offset",
        "in": "query",
        "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
        "required": false,
        "type": "integer"
```

```
        }
       ],
       "responses": {
        "200": {
         "description": "successful operation",
         "schema": {
          "type": "array",
          "items": {
           "$ref": "#/definitions/NTP"
          }
         }
        }
       },
       "security": [
        {
         "jwt": []
        }
       ]
      }
     },
     "/object/ntp/hostname/{hostname}": {
      "get": {
       "tags": [
        "ntp"
       ],
       "summary": "Get all NTP information for a given network device by
hostname",
       "description": "Retrieves all Network Time Protocol (NTP) configuration and
status information, such as whether the service is running and if it is in time
```

synchronization, for a given switch or host in the network. Refer to the NTP model for all data collected.",
"produces": [
"application/json"
],
"parameters": [
{
"name": "hostname",
"in": "path",
"description": "User-specified name for a network switch or host. For example, leaf01, spine04, host-6, engr-1, tor-22.",
"required": true,
"type": "string"
},
{
"name": "eq_timestamp",
"in": "query",
"description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
"required": false,
"type": "integer"
},
{
"name": "count",
"in": "query",
"description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
"required": false,

```
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/NTP"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
```

```
  },
  "/object/port": {
   "get": {
    "tags": [
     "port"
    ],
    "summary": "Get all information for all physical ports on all network devices",
    "description": "Retrieves all physical port information, such as speed, connector, vendor, part and serial number, and FEC support, for all network devices. Refer to Port model for all data collected.",
    "produces": [
     "application/json"
    ],
    "parameters": [
     {
      "name": "eq_timestamp",
      "in": "query",
      "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
      "required": false,
      "type": "integer"
     },
     {
      "name": "count",
      "in": "query",
      "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
      "required": false,
```

```
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Port"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
```

```
  },
  "/object/port/hostname/{hostname}": {
   "get": {
    "tags": [
     "port"
    ],
    "summary": "Get all information for all physical ports on a given network device by hostname",
    "description": "Retrieves all physical port information, such as speed, connector, vendor, part and serial number, and FEC support, for a given switch or host in the network. Refer to Port model for all data collected.",
    "produces": [
     "application/json"
    ],
    "parameters": [
     {
      "name": "hostname",
      "in": "path",
      "description": "User-specified name for a network switch or host. For example, leaf01, spine04, host-6, engr-1, tor-22.",
      "required": true,
      "type": "string"
     },
     {
      "name": "eq_timestamp",
      "in": "query",
      "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
```

```
          "required": false,
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Port"
            }
```

```
          }
         }
        },
        "security": [
         {
          "jwt": []
         }
        ]
       }
      },
      "/object/route": {
       "get": {
        "tags": [
         "route"
        ],
        "summary": "Get all route information for all network devices",
        "description": "Retrieves route information, such as VRF, source, next hops, origin, protocol, and prefix, for all switches and hosts in the network. Refer to Route model for all data collected.",
        "produces": [
         "application/json"
        ],
        "parameters": [
         {
          "name": "eq_timestamp",
          "in": "query",
          "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
```

```
          "required": false,
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Route"
            }
```

```
          }
         }
        },
        "security": [
         {
          "jwt": []
         }
        ]
       }
      },
      "/object/route/hostname/{hostname}": {
       "get": {
        "tags": [
         "route"
        ],
        "summary": "Get all route information for a given network device by hostname",
        "description": "Retrieves route information, such as VRF, source, next hops, origin, protocol, and prefix, for a given switch or host in the network. Refer to Route model for all data collected.",
        "produces": [
         "application/json"
        ],
        "parameters": [
         {
          "name": "hostname",
          "in": "path",
          "description": "User-specified name for a network switch or host. For example, leaf01, spine04, host-6, engr-1, tor-22.",
```

```
        "required": true,
        "type": "string"
      },
      {
        "name": "eq_timestamp",
        "in": "query",
        "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "count",
        "in": "query",
        "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "offset",
        "in": "query",
        "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
        "required": false,
        "type": "integer"
      }
```

```
        ],
        "responses": {
          "200": {
            "description": "successful operation",
            "schema": {
              "type": "array",
              "items": {
                "$ref": "#/definitions/Route"
              }
            }
          }
        },
        "security": [
          {
            "jwt": []
          }
        ]
      }
    },
    "/object/sensor": {
      "get": {
        "tags": [
          "sensor"
        ],
        "summary": "Get all sensor information for all network devices",
        "description": "Retrieves data from fan, temperature, and power supply unit sensors, such as their name, state, and threshold status, for all switches and hosts in the network. Refer to Sensor model for all data collected.",
        "produces": [
```

```
        "application/json"
      ],
      "parameters": [
       {
        "name": "eq_timestamp",
        "in": "query",
        "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
        "required": false,
        "type": "integer"
       },
       {
        "name": "count",
        "in": "query",
        "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
        "required": false,
        "type": "integer"
       },
       {
        "name": "offset",
        "in": "query",
        "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
        "required": false,
        "type": "integer"
       }
```

```
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Sensor"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
  "/object/sensor/hostname/{hostname}": {
    "get": {
      "tags": [
        "sensor"
      ],
      "summary": "Get all sensor information for a given network device by
hostname",
      "description": "Retrieves data from fan, temperature, and power supply unit
sensors, such as their name, state, and threshold status, for a given switch or host
in the network. Refer to Sensor model for all data collected.",
```

```
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "hostname",
        "in": "path",
        "description": "User-specified name for a network switch or host. For example, leaf01, spine04, host-6, engr-1, tor-22.",
        "required": true,
        "type": "string"
      },
      {
        "name": "eq_timestamp",
        "in": "query",
        "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
        "required": false,
        "type": "integer"
      },
      {
        "name": "count",
        "in": "query",
        "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
        "required": false,
        "type": "integer"
      },
```

```
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Sensor"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
  "/object/services": {
```

```
    "get": {
      "tags": [
        "services"
      ],
      "summary": "Get all services information for all network devices",
      "description": "Retrieves services information, such as XXX, for all switches and hosts in the network. Refer to Services for all data collected.",
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "eq_timestamp",
          "in": "query",
          "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
```

```
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Services"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
  "/object/services/hostname/{hostname}": {
    "get": {
```

```
      "tags": [
        "services"
      ],
      "summary": "Get all services information for a given network device by hostname",
      "description": "Retrieves services information, such as XXX, for a given switch or host in the network. Refer to Services for all data collected.",
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "hostname",
          "in": "path",
          "description": "User-specified name for a network switch or host. For example, leaf01, spine04, host-6, engr-1, tor-22.",
          "required": true,
          "type": "string"
        },
        {
          "name": "eq_timestamp",
          "in": "query",
          "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
          "required": false,
          "type": "integer"
        },
        {
```

```
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Services"
            }
          }
        }
      },
      "security": [
```

```
        {
          "jwt": []
        }
       ]
     }
   },
   "/object/vlan": {
     "get": {
       "tags": [
         "vlan"
       ],
       "summary": "Get all VLAN information for all network devices",
       "description": "Retrieves VLAN information, such as hostname, interface name, associated VLANs, ports, and time of last change, for all switches and hosts in the network. Refer to Vlan model for all data collected.",
       "produces": [
         "application/json"
       ],
       "parameters": [
         {
           "name": "eq_timestamp",
           "in": "query",
           "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
           "required": false,
           "type": "integer"
         },
         {
```

```
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Vlan"
            }
          }
        }
      },
      "security": [
```

```
        {
          "jwt": []
        }
       ]
      }
    },
    "/object/vlan/hostname/{hostname}": {
      "get": {
        "tags": [
          "vlan"
        ],
        "summary": "Get all VLAN information for a given network device by
hostname",
        "description": "Retrieves VLAN information, such as hostname, interface
name, associated VLANs, ports, and time of last change, for a given switch or  host
in the network. Refer to Vlan model for all data collected.",
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "hostname",
            "in": "path",
            "description": "User-specified name for a network switch or host. For
example, leaf01, spine04, host-6, engr-1, tor-22.",
            "required": true,
            "type": "string"
          },
          {
```

```
          "name": "eq_timestamp",
          "in": "query",
          "description": "Display results for a given time. Time must be entered in Epoch format. For example, to display the results for Monday February 13, 2019 at 1:25 pm, use a time converter and enter 1550082300.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "count",
          "in": "query",
          "description": "Number of entries to display starting from the offset value. For example, a count of 100 displays 100 entries at a time.",
          "required": false,
          "type": "integer"
        },
        {
          "name": "offset",
          "in": "query",
          "description": "Used in combination with count, offset specifies the starting location within the set of entries returned. For example, an offset of 100 would display results beginning with entry 101.",
          "required": false,
          "type": "integer"
        }
      ],
      "responses": {
        "200": {
          "description": "successful operation",
```

```
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Vlan"
            }
          }
        }
      },
      "security": [
        {
          "jwt": []
        }
      ]
    }
  },
  "securityDefinitions": {
    "jwt": {
      "type": "apiKey",
      "name": "Authorization",
      "in": "header"
    }
  },
  "definitions": {
    "Address": {
      "description": "This model contains descriptions of the data collected and
returned by the Address endpoint.",
      "type": "object",
      "required": [
```

```
    "schema"
  ],
  "properties": {
    "schema": {
      "$ref": "#/definitions/Schema"
    },
    "opid": {
      "type": "integer",
      "format": "int32",
      "description": "Internal use only"
    },
    "hostname": {
      "type": "string",
      "description": "User-defined name for a device"
    },
    "ifname": {
      "type": "string",
      "description": "Name of a software (versus physical) interface"
    },
    "timestamp": {
      "type": "integer",
      "format": "int64",
      "description": "Date and time data was collected"
    },
    "prefix": {
      "type": "string",
      "description": "Address prefix for IPv4, IPv6, or EVPN traffic"
    },
    "mask": {
```

```
          "type": "integer",
          "format": "int32",
          "description": "Address mask for IPv4, IPv6, or EVPN traffic"
        },
        "is_ipv6": {
          "type": "boolean",
          "description": "Indicates whether address is an IPv6 address (true) or not
(false)"
        },
        "vrf": {
          "type": "string",
          "description": "Virtual Route Forwarding interface name"
        },
        "db_state": {
          "type": "string",
          "description": "Internal use only"
        }
      }
    },
    "BgpSession": {
      "description": "This model contains descriptions of the data collected and
returned by the BGP endpoint.",
      "type": "object",
      "required": [
        "schema"
      ],
      "properties": {
        "schema": {
          "$ref": "#/definitions/Schema"
```

```
        },
        "opid": {
          "type": "integer",
          "format": "int32",
          "description": "Internal use only"
        },
        "hostname": {
          "type": "string",
          "description": "User-defined name for a device"
        },
        "peer_name": {
          "type": "string",
          "description": "Interface name or hostname for a peer device"
        },
        "timestamp": {
          "type": "integer",
          "format": "int64",
          "description": "Date and time data was collected"
        },
        "db_state": {
          "type": "string",
          "description": "Internal use only"
        },
        "state": {
          "type": "string",
          "description": "Current state of the BGP session. Values include established
and not established."
        },
        "peer_router_id": {
```

```
          "type": "string",
          "description": "If peer is a router, IP address of router"
        },
        "peer_asn": {
          "type": "integer",
          "format": "int64",
          "description": "Peer autonomous system number (ASN), identifier for a
collection of IP networks and routers"
        },
        "peer_hostname": {
          "type": "string",
          "description": "User-defined name for the peer device"
        },
        "asn": {
          "type": "integer",
          "format": "int64",
          "description": "Host autonomous system number (ASN), identifier for a
collection of IP networks and routers"
        },
        "reason": {
          "type": "string",
          "description": "Text describing the cause of, or trigger for, an event"
        },
        "ipv4_pfx_rcvd": {
          "type": "integer",
          "format": "int32",
          "description": "Address prefix received for an IPv4 address"
        },
        "ipv6_pfx_rcvd": {
```

```
          "type": "integer",
          "format": "int32",
          "description": "Address prefix received for an IPv6 address"
        },
        "evpn_pfx_rcvd": {
          "type": "integer",
          "format": "int32",
          "description": "Address prefix received for an EVPN address"
        },
        "last_reset_time": {
          "type": "number",
          "format": "float",
          "description": "Date and time at which the session was last established or
reset"
        },
        "up_time": {
          "type": "number",
          "format": "float",
          "description": "Number of seconds the session has been established, in
EPOCH notation"
        },
        "conn_estd": {
          "type": "integer",
          "format": "int32",
          "description": "Number of connections established for a given session"
        },
        "conn_dropped": {
          "type": "integer",
          "format": "int32",
```

```
        "description": "Number of dropped connections for a given session"
      },
      "upd8_rx": {
        "type": "integer",
        "format": "int32",
        "description": "Count of protocol messages received"
      },
      "upd8_tx": {
        "type": "integer",
        "format": "int32",
        "description": "Count of protocol messages transmitted"
      },
      "vrfid": {
        "type": "integer",
        "format": "int32",
        "description": "Integer identifier of the VRF interface when used"
      },
      "vrf": {
        "type": "string",
        "description": "Name of the Virtual Route Forwarding interface"
      },
      "tx_families": {
        "type": "string",
        "description": "Address families supported for the transmit session channel.
Values include ipv4, ipv6, and evpn."
      },
      "rx_families": {
        "type": "string",
        "description": "Address families supported for the receive session channel.
```

```
Values include ipv4, ipv6, and evpn."
        }
      }
    },
    "ClagSessionInfo": {
      "description": "This model contains descriptions of the data collected and returned by the CLAG endpoint.",
      "type": "object",
      "required": [
        "schema"
      ],
      "properties": {
        "schema": {
          "$ref": "#/definitions/Schema"
        },
        "opid": {
          "type": "integer",
          "format": "int32",
          "description": "Internal use only"
        },
        "hostname": {
          "type": "string",
          "description": "User-defined name for a device"
        },
        "clag_sysmac": {
          "type": "string",
          "description": "Unique MAC address for each bond interface pair. This must be a value between 44:38:39:ff:00:00 and 44:38:39:ff:ff:ff."
        },
```

```
        "timestamp": {
          "type": "integer",
          "format": "int64",
          "description": "Date and time the CLAG session was started, deleted,
updated, or marked dead (device went down)"
        },
        "db_state": {
          "type": "string",
          "description": "Internal use only"
        },
        "peer_role": {
          "type": "string",
          "description": "Role of the peer device. Values include primary and
secondary."
        },
        "peer_state": {
          "type": "boolean",
          "description": "Indicates if peer device is up (true) or down (false)"
        },
        "peer_if": {
          "type": "string",
          "description": "Name of the peer interface used for the session"
        },
        "backup_ip_active": {
          "type": "boolean",
          "description": "Indicates whether the backup IP address has been specified
and is active (true) or not (false)"
        },
        "backup_ip": {
```

```
        "type": "string",
        "description": "IP address of the interface to use if the peerlink (or bond)
goes down"
      },
      "single_bonds": {
        "type": "string",
        "description": "Identifies a set of interfaces connecting to only one of the two
switches in the bond"
      },
      "dual_bonds": {
        "type": "string",
        "description": "Identifies a set of interfaces connecting to both switches in
the bond"
      },
      "conflicted_bonds": {
        "type": "string",
        "description": "Identifies the set of interfaces in a bond that do not match on
each end of the bond"
      },
      "proto_down_bonds": {
        "type": "string",
        "description": "Interface on the switch brought down by the clagd service.
Value is blank if no interfaces are down due to the clagd service."
      },
      "vxlan_anycast": {
        "type": "string",
        "description": "Anycast IP address used for VXLAN termination"
      },
      "role": {
```

```
        "type": "string",
        "description": "Role of the host device. Values include primary and
secondary."
       }
      }
    },
    "ErrorResponse": {
     "description": "Standard error response",
     "type": "object",
     "properties": {
      "message": {
       "type": "string",
       "description": "One or more errors have been encountered during the
processing of the associated request"
       }
      }
    },
    "Evpn": {
     "description": "This model contains descriptions of the data collected and
returned by the EVPN endpoint.",
     "type": "object",
     "required": [
      "schema"
     ],
     "properties": {
      "schema": {
       "$ref": "#/definitions/Schema"
      },
      "hostname": {
```

```
        "type": "string",
        "description": "User-defined name for a device"
      },
      "opid": {
        "type": "integer",
        "format": "int32",
        "description": "Internal use only"
      },
      "vni": {
        "type": "integer",
        "format": "int32",
        "description": "Name of the virtual network instance (VNI) where session is running"
      },
      "origin_ip": {
        "type": "string",
        "description": "Host device's local VXLAN tunnel IP address for the EVPN instance"
      },
      "timestamp": {
        "type": "integer",
        "format": "int64",
        "description": "Date and time the session was started, deleted, updated or marked as dead (device is down)"
      },
      "rd": {
        "type": "string",
        "description": "Route distinguisher used in the filtering mechanism for BGP route exchange"
```

```
        },
        "export_rt": {
          "type": "string",
          "description": "IP address and port of the export route target used in the
filtering mechanism for BGP route exchange"
        },
        "import_rt": {
          "type": "string",
          "description": "IP address and port of the import route target used in the
filtering mechanism for BGP route exchange"
        },
        "in_kernel": {
          "type": "boolean",
          "description": "Indicates whether the associated VNI is in the kernel (in
kernel) or not (not in kernel)"
        },
        "adv_all_vni": {
          "type": "boolean",
          "description": "Indicates whether the VNI state is advertising all VNIs (true)
or not (false)"
        },
        "adv_gw_ip": {
          "type": "string",
          "description": "Indicates whether the host device is advertising the gateway
IP address (true) or not (false)"
        },
        "is_l3": {
          "type": "boolean",
          "description": "Indicates whether the session is part of a layer 3
```

```
configuration (true) or not (false)"
        },
        "db_state": {
          "type": "string",
          "description": "Internal use only"
        }
      }
    },
    "Field": {
      "type": "object",
      "required": [
        "aliases",
        "defaultValue",
        "doc",
        "jsonProps",
        "name",
        "objectProps",
        "order",
        "props",
        "schema"
      ],
      "properties": {
        "props": {
          "type": "object",
          "additionalProperties": {
            "type": "string"
          }
        },
        "name": {
```

```
      "type": "string"
    },
    "schema": {
      "$ref": "#/definitions/Schema"
    },
    "doc": {
      "type": "string"
    },
    "defaultValue": {
      "$ref": "#/definitions/JsonNode"
    },
    "order": {
      "type": "string",
      "enum": [
        "ASCENDING",
        "DESCENDING",
        "IGNORE"
      ]
    },
    "aliases": {
      "type": "array",
      "uniqueItems": true,
      "items": {
        "type": "string"
      }
    },
    "jsonProps": {
      "type": "object",
      "additionalProperties": {
```

```
            "$ref": "#/definitions/JsonNode"
          }
        },
        "objectProps": {
          "type": "object",
          "additionalProperties": {
            "type": "object",
            "properties": {}
          }
        }
      }
    },
    "Interface": {
      "description": "This model contains descriptions of the data collected and
returned by the Interface endpoint.",
      "type": "object",
      "required": [
        "schema"
      ],
      "properties": {
        "schema": {
          "$ref": "#/definitions/Schema"
        },
        "opid": {
          "type": "integer",
          "format": "int32",
          "description": "Internal use only"
        },
        "hostname": {
```

```
        "type": "string",
        "description": "User-defined name for a device"
      },
      "type": {
        "type": "string",
        "description": "Identifier of the kind of interface. Values include bond, bridge, eth, loopback, macvlan, swp, vlan, vrf, and vxlan."
      },
      "timestamp": {
        "type": "integer",
        "format": "int64",
        "description": "Date and time the data was collected"
      },
      "last_changed": {
        "type": "integer",
        "format": "int64",
        "description": "Date and time the interface was started, deleted, updated or marked as dead (device is down)"
      },
      "ifname": {
        "type": "string",
        "description": "Name of the interface"
      },
      "state": {
        "type": "string",
        "description": "Indicates whether the interface is up or down"
      },
      "vrf": {
        "type": "string",
```

```
        "description": "Name of the virtual route forwarding (VRF) interface, if
present"
       },
       "details": {
        "type": "string",
        "description": "???"
       }
      }
    },
    "InventoryModel": {
     "type": "object",
     "required": [
      "label",
      "value"
     ],
     "properties": {
      "label": {
       "type": "string"
      },
      "value": {
       "type": "integer",
       "format": "int32"
      }
     }
    },
    "InventoryOutput": {
     "type": "object",
     "properties": {
      "data": {
```

```
        "$ref": "#/definitions/InventorySampleClass"
      }
    }
  },
  "InventorySampleClass": {
    "type": "object",
    "properties": {
      "total": {
        "type": "integer",
        "format": "int32",
        "example": 100,
        "description": "total number of devices"
      },
      "os_version": {
        "$ref": "#/definitions/InventorySuperModel"
      },
      "os_vendor": {
        "$ref": "#/definitions/InventorySuperModel"
      },
      "asic": {
        "$ref": "#/definitions/InventorySuperModel"
      },
      "asic_vendor": {
        "$ref": "#/definitions/InventorySuperModel"
      },
      "asic_model": {
        "$ref": "#/definitions/InventorySuperModel"
      },
      "cl_license": {
```

```
        "$ref": "#/definitions/InventorySuperModel"
      },
      "agent_version": {
        "$ref": "#/definitions/InventorySuperModel"
      },
      "agent_state": {
        "$ref": "#/definitions/InventorySuperModel"
      },
      "platform": {
        "$ref": "#/definitions/InventorySuperModel"
      },
      "platform_vendor": {
        "$ref": "#/definitions/InventorySuperModel"
      },
      "disk_size": {
        "$ref": "#/definitions/InventorySuperModel"
      },
      "memory_size": {
        "$ref": "#/definitions/InventorySuperModel"
      },
      "platform_model": {
        "$ref": "#/definitions/InventorySuperModel"
      },
      "interface_speeds": {
        "$ref": "#/definitions/InventorySuperModel"
      }
    }
  },
  "InventorySuperModel": {
```

```
    "type": "object",
    "required": [
      "data",
      "label"
    ],
    "properties": {
      "label": {
        "type": "string"
      },
      "data": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/InventoryModel"
        }
      }
    }
  },
  "IteratorEntryStringJsonNode": {
    "type": "object"
  },
  "IteratorJsonNode": {
    "type": "object"
  },
  "IteratorString": {
    "type": "object"
  },
  "JsonNode": {
    "type": "object",
    "required": [
```

```
        "array",
        "bigDecimal",
        "bigInteger",
        "bigIntegerValue",
        "binary",
        "binaryValue",
        "boolean",
        "booleanValue",
        "containerNode",
        "decimalValue",
        "double",
        "doubleValue",
        "elements",
        "fieldNames",
        "fields",
        "floatingPointNumber",
        "int",
        "intValue",
        "integralNumber",
        "long",
        "longValue",
        "missingNode",
        "null",
        "number",
        "numberType",
        "numberValue",
        "object",
        "pojo",
        "textValue",
```

```
      "textual",
      "valueAsBoolean",
      "valueAsDouble",
      "valueAsInt",
      "valueAsLong",
      "valueAsText",
      "valueNode"
    ],
    "properties": {
      "elements": {
        "$ref": "#/definitions/IteratorJsonNode"
      },
      "fieldNames": {
        "$ref": "#/definitions/IteratorString"
      },
      "binary": {
        "type": "boolean"
      },
      "intValue": {
        "type": "integer",
        "format": "int32"
      },
      "object": {
        "type": "boolean"
      },
      "int": {
        "type": "boolean"
      },
      "long": {
```

```
      "type": "boolean"
    },
    "double": {
      "type": "boolean"
    },
    "bigDecimal": {
      "type": "boolean"
    },
    "bigInteger": {
      "type": "boolean"
    },
    "textual": {
      "type": "boolean"
    },
    "boolean": {
      "type": "boolean"
    },
    "valueNode": {
      "type": "boolean"
    },
    "containerNode": {
      "type": "boolean"
    },
    "missingNode": {
      "type": "boolean"
    },
    "pojo": {
      "type": "boolean"
    },
```

```
"number": {
  "type": "boolean"
},
"integralNumber": {
  "type": "boolean"
},
"floatingPointNumber": {
  "type": "boolean"
},
"numberValue": {
  "$ref": "#/definitions/Number"
},
"numberType": {
  "type": "string",
  "enum": [
    "INT",
    "LONG",
    "BIG_INTEGER",
    "FLOAT",
    "DOUBLE",
    "BIG_DECIMAL"
  ]
},
"longValue": {
  "type": "integer",
  "format": "int64"
},
"bigIntegerValue": {
  "type": "integer"
```

```
        },
        "doubleValue": {
          "type": "number",
          "format": "double"
        },
        "decimalValue": {
          "type": "number"
        },
        "booleanValue": {
          "type": "boolean"
        },
        "binaryValue": {
          "type": "array",
          "items": {
            "type": "string",
            "format": "byte",
            "pattern": "^(?:[A-Za-z0-9+/]{4})\*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?$"
          }
        },
        "valueAsInt": {
          "type": "integer",
          "format": "int32"
        },
        "valueAsLong": {
          "type": "integer",
          "format": "int64"
        },
        "valueAsDouble": {
          "type": "number",
```

```
          "format": "double"
        },
        "valueAsBoolean": {
          "type": "boolean"
        },
        "textValue": {
          "type": "string"
        },
        "valueAsText": {
          "type": "string"
        },
        "array": {
          "type": "boolean"
        },
        "fields": {
          "$ref": "#/definitions/IteratorEntryStringJsonNode"
        },
        "null": {
          "type": "boolean"
        }
      }
    },
    "LLDP": {
      "description": "This model contains descriptions of the data collected and
returned by the LLDP endpoint.",
      "type": "object",
      "required": [
        "schema"
      ],
```

```
      "properties": {
        "schema": {
          "$ref": "#/definitions/Schema"
        },
        "opid": {
          "type": "integer",
          "format": "int32",
          "description": "Internal use only"
        },
        "hostname": {
          "type": "string",
          "description": "User-defined name for the host device"
        },
        "ifname": {
          "type": "string",
          "description": "Name of the host interface where the LLDP service is running"
        },
        "timestamp": {
          "type": "integer",
          "format": "int64",
          "description": "Date and time that the session was started, deleted, updated,
or marked dead (device is down)"
        },
        "peer_hostname": {
          "type": "string",
          "description": "User-defined name for the peer device"
        },
        "peer_ifname": {
          "type": "string",
```

```
        "description": "Name of the peer interface where the session is running"
      },
      "lldp_peer_bridge": {
        "type": "boolean",
        "description": "Indicates whether the peer device is a bridge (true) or not (false)"
      },
      "lldp_peer_router": {
        "type": "boolean",
        "description": "Indicates whether the peer device is a router (true) or not (false)"
      },
      "lldp_peer_station": {
        "type": "boolean",
        "description": "Indicates whether the peer device is a station (true) or not (false)"
      },
      "lldp_peer_os": {
        "type": "string",
        "description": "Operating system (OS) used by peer device. Values include Cumulus Linux, RedHat, Ubuntu, and CentOS."
      },
      "lldp_peer_osv": {
        "type": "string",
        "description": "Version of the OS used by peer device. Example values include 3.7.3, 2.5.x, 16.04, 7.1."
      },
      "db_state": {
        "type": "string",
```

```
        "description": "Internal use only"
       }
      }
    },
    "LogicalType": {
      "type": "object",
      "required": [
        "name"
      ],
      "properties": {
        "name": {
          "type": "string"
        }
      }
    },
    "LoginRequest": {
      "description": "User-entered credentials used to validate if user is allowed to
access NetQ",
      "type": "object",
      "required": [
        "password",
        "username"
      ],
      "properties": {
        "username": {
          "type": "string"
        },
        "password": {
          "type": "string"
```

```
      }
     }
    },
    "LoginResponse": {
     "description": "Response to user login request",
     "type": "object",
     "required": [
      "id"
     ],
     "properties": {
      "terms_of_use_accepted": {
       "type": "boolean",
       "description": "Indicates whether user has accepted the terms of use"
      },
      "access_token": {
       "type": "string",
       "description": "Grants jason web token (jwt) access token. The access token also contains the NetQ Platform or Appliance (opid) which the user is permitted to access. By default, it is the primary opid given by the user."
      },
      "expires_at": {
       "type": "integer",
       "format": "int64",
       "description": "Number of hours the access token is valid before it automatically expires, epoch miliseconds. By default, tokens are valid for 24 hours."
      },
      "id": {
       "type": "string"
```

```
        },
        "premises": {
          "type": "array",
          "description": "List of premises that this user is authorized to view",
          "items": {
            "$ref": "#/definitions/Premises"
          }
        },
        "customer_id": {
          "type": "integer",
          "format": "int32",
          "description": "customer id of this user"
        }
      }
    },
    "MacFdb": {
      "description": "This model contains descriptions of the data collected and
returned by the MacFdb endpoint.",
      "type": "object",
      "required": [
        "schema"
      ],
      "properties": {
        "schema": {
          "$ref": "#/definitions/Schema"
        },
        "opid": {
          "type": "integer",
          "format": "int32",
```

```
        "description": "Internal use only"
      },
      "hostname": {
        "type": "string",
        "description": "User-defined name for a device"
      },
      "mac_address": {
        "type": "string",
        "description": "Media access control address for a device reachable via the local bridge member port 'nexthop' or via remote VTEP with IP address of 'dst'"
      },
      "timestamp": {
        "type": "integer",
        "format": "int64",
        "description": "Date and time data was collected"
      },
      "dst": {
        "type": "string",
        "description": "IP address of a remote VTEP from which this MAC address is reachable"
      },
      "nexthop": {
        "type": "string",
        "description": "Interface where the MAC address can be reached"
      },
      "is_remote": {
        "type": "boolean",
        "description": "Indicates if the MAC address is reachable locally on 'nexthop' (false) or remotely via a VTEP with address 'dst' (true)"
```

```
        },
        "port": {
          "type": "string",
          "description": "Currently unused"
        },
        "vlan": {
          "type": "integer",
          "format": "int32",
          "description": "Name of associated VLAN"
        },
        "is_static": {
          "type": "boolean",
          "description": "Indicates if the MAC address is a static address (true) or
dynamic address (false)"
        },
        "origin": {
          "type": "boolean",
          "description": "Indicates whether the MAC address is one of the host's
interface addresses (true) or not (false)"
        },
        "active": {
          "type": "boolean",
          "description": "Currently unused"
        },
        "db_state": {
          "type": "string",
          "description": "Internal use only"
        }
      }
```

```
    },
    "MstpInfo": {
      "description": "This model contains descriptions of the data collected and
returned by the MSTP endpoint.",
      "type": "object",
      "required": [
        "schema"
      ],
      "properties": {
        "schema": {
          "$ref": "#/definitions/Schema"
        },
        "opid": {
          "type": "integer",
          "format": "int32",
          "description": "Internal use only"
        },
        "hostname": {
          "type": "string",
          "description": "User-defined name for a device"
        },
        "bridge_name": {
          "type": "string",
          "description": "User-defined name for a bridge"
        },
        "timestamp": {
          "type": "integer",
          "format": "int64",
          "description": "Date and time data was collected"
```

```
        },
        "db_state": {
          "type": "string",
          "description": "Internal use only"
        },
        "state": {
          "type": "boolean",
          "description": "Indicates whether MSTP is enabled (true) or not (false)"
        },
        "root_port_name": {
          "type": "string",
          "description": "Name of the physical interface (port) that provides the
minimum cost path from the Bridge to the MSTI Regional Root"
        },
        "root_bridge": {
          "type": "string",
          "description": "Name of the CIST root for the bridged LAN"
        },
        "topo_chg_ports": {
          "type": "string",
          "description": "Names of ports that were part of the last topology change
event"
        },
        "time_since_tcn": {
          "type": "integer",
          "format": "int64",
          "description": "Amount of time, in seconds, since the last topology change
notification"
        },
```

```
        "topo_chg_cntr": {
          "type": "integer",
          "format": "int64",
          "description": "Number of times topology change notifications have been
sent"
        },
        "bridge_id": {
          "type": "string",
          "description": "Spanning Tree bridge identifier for current host"
        },
        "edge_ports": {
          "type": "string",
          "description": "List of port names that are Spanning Tree edge ports"
        },
        "network_ports": {
          "type": "string",
          "description": "List of port names that are Spanning Tree network ports"
        },
        "disputed_ports": {
          "type": "string",
          "description": "List of port names that are in Spanning Tree dispute state"
        },
        "bpduguard_ports": {
          "type": "string",
          "description": "List of port names where BPDU Guard is enabled"
        },
        "bpduguard_err_ports": {
          "type": "string",
          "description": "List of port names where BPDU Guard violation occurred"
```

```
        },
        "ba_inconsistent_ports": {
          "type": "string",
          "description": "List of port names where Spanning Tree Bridge Assurance is
failing"
        },
        "bpdufilter_ports": {
          "type": "string",
          "description": "List of port names where Spanning Tree BPDU Filter is
enabled"
        },
        "ports": {
          "type": "string",
          "description": "List of port names in the Spanning Tree instance"
        },
        "is_vlan_filtering": {
          "type": "boolean",
          "description": "Indicates whether the bridge is enabled with VLAN filtering (is
VLAN-aware) (true) or not (false)"
        }
      }
    },
    "Neighbor": {
      "description": "This model contains descriptions of the data collected and
returned by the Neighbor endpoint.",
      "type": "object",
      "required": [
        "schema"
      ],
```

```
      "properties": {
        "schema": {
          "$ref": "#/definitions/Schema"
        },
        "opid": {
          "type": "integer",
          "format": "int32",
          "description": "Internal use only"
        },
        "hostname": {
          "type": "string",
          "description": "User-defined name of a device"
        },
        "ifname": {
          "type": "string",
          "description": "User-defined name of an software interface on a device"
        },
        "timestamp": {
          "type": "integer",
          "format": "int64",
          "description": "Date and time when data was collected"
        },
        "vrf": {
          "type": "string",
          "description": "Name of virtual route forwarding (VRF) interface, when
applicable"
        },
        "is_remote": {
          "type": "boolean",
```

```
        "description": "Indicates if the neighbor is reachable through a local interface (false) or remotely through a ??? (true)"
      },
      "ifindex": {
        "type": "integer",
        "format": "int32",
        "description": "IP address index for the neighbor device"
      },
      "mac_address": {
        "type": "string",
        "description": "MAC address for the neighbor device"
      },
      "is_ipv6": {
        "type": "boolean",
        "description": "Indicates whether the neighbor's IP address is version six (IPv6) (true) or version four (IPv4) (false)"
      },
      "message_type": {
        "type": "string",
        "description": "Network protocol or service identifier used in neighbor-related events. Value is neighbor."
      },
      "ip_address": {
        "type": "string",
        "description": "IPv4 or IPv6 address for the neighbor device"
      },
      "db_state": {
        "type": "string",
        "description": "Internal use only"
```

```
      }
     }
    },
    "NODE": {
     "description": "This model contains descriptions of the data collected and returned by the Node endpoint.",
     "type": "object",
     "required": [
      "schema"
     ],
     "properties": {
      "schema": {
       "$ref": "#/definitions/Schema"
      },
      "opid": {
       "type": "integer",
       "format": "int32",
       "description": "Internal use only"
      },
      "hostname": {
       "type": "string",
       "description": "User-defined name of the device"
      },
      "sys_uptime": {
       "type": "integer",
       "format": "int64",
       "description": "Amount of time, in seconds???, this device has been powered up"
      },
```

```
        "lastboot": {
          "type": "integer",
          "format": "int64",
          "description": "Date and time, in EPOCH format???, this device was last
booted"
        },
        "last_reinit": {
          "type": "integer",
          "format": "int64",
          "description": "Date and time, in xxx????, this device was last initialized"
        },
        "active": {
          "type": "boolean",
          "description": "Indicates whether this device is active(???) (true) or not (false)"
        },
        "version": {
          "type": "string",
          "description": "????"
        },
        "ntp_state": {
          "type": "string",
          "description": "Status of the NTP service running on this device; in sync, not
in sync, or unknown"
        },
        "timestamp": {
          "type": "integer",
          "format": "int64",
          "description": "Date and time data was collected"
        },
```

```
        "last_update_time": {
          "type": "integer",
          "format": "int64",
          "description": "Date and time the device was last updated"
        },
        "db_state": {
          "type": "string",
          "description": "Internal use only"
        }
      }
    },
    "NTP": {
      "description": "This model contains descriptions of the data collected and
returned by the NTP endpoint.",
      "type": "object",
      "required": [
        "schema"
      ],
      "properties": {
        "schema": {
          "$ref": "#/definitions/Schema"
        },
        "hostname": {
          "type": "string",
          "description": "User-defined name of device running NTP service"
        },
        "opid": {
          "type": "integer",
          "format": "int32",
```

```
        "description": "Internal use only"
      },
      "timestamp": {
        "type": "integer",
        "format": "int64",
        "description": "Date and time data was collected"
      },
      "ntp_sync": {
        "type": "string",
        "description": "Status of the NTP service running on this device; in sync, not in sync, or unknown"
      },
      "stratum": {
        "type": "integer",
        "format": "int32",
        "description": "????"
      },
      "ntp_app": {
        "type": "string",
        "description": "Name/version/release? of the NTP service????"
      },
      "message_type": {
        "type": "string",
        "description": "Network protocol or service identifier used in NTP-related events. Value is ntp."
      },
      "current_server": {
        "type": "string",
        "description": "Name or address of server providing time synchronization"
```

```
        },
        "active": {
          "type": "boolean",
          "description": "Indicates whether NTP service is running (true) or not (false)"
        },
        "db_state": {
          "type": "string",
          "description": "Internal use only"
        }
      }
    },
    "Number": {
      "type": "object",
      "description": "????"
    },
    "Port": {
      "description": "This model contains descriptions of the data collected and
returned by the Port endpoint.",
      "type": "object",
      "required": [
        "schema"
      ],
      "properties": {
        "schema": {
          "$ref": "#/definitions/Schema"
        },
        "opid": {
          "type": "integer",
          "format": "int32",
```

```
        "description": "Internal use only"
      },
      "hostname": {
        "type": "string",
        "description": "User-defined name for the device with this port"
      },
      "ifname": {
        "type": "string",
        "description": "User-defined name for the software interface on this port"
      },
      "timestamp": {
        "type": "integer",
        "format": "int64",
        "description": "Date and time data was collected"
      },
      "speed": {
        "type": "string",
        "description": "Maximum rating for port. Examples include 10G, 25G, 40G,
unknown."
      },
      "identifier": {
        "type": "string",
        "description": "Identifies type of port module if installed. Example values
include empty, QSFP+, SFP, RJ45"
      },
      "autoneg": {
        "type": "string",
        "description": "Indicates status of the auto-negotiation feature. Values
include on and off."
```

```
      },
      "db_state": {
        "type": "string",
        "description": "Internal use only"
      },
      "transreceiver": {
        "type": "string",
        "description": "Name of installed transceiver. Example values include 40G Base-CR4, 10Gtek."
      },
      "connector": {
        "type": "string",
        "description": "Name of installed connector. Example values include LC, copper pigtail, RJ-45, n/a."
      },
      "vendor_name": {
        "type": "string",
        "description": "Name of the port vendor. Example values include OEM, Mellanox, Amphenol, Finisar, Fiberstore, n/a."
      },
      "part_number": {
        "type": "string",
        "description": "Manufacturer part number"
      },
      "serial_number": {
        "type": "string",
        "description": "Manufacturer serial number"
      },
      "length": {
```

```
          "type": "string",
          "description": "Length of cable connected (or length the transceiver can transmit or ????). Example values include 1m, 2m, n/a."
        },
        "supported_fec": {
          "type": "string",
          "description": "List of forward error correction (FEC) algorithms supported on this port. Example values include BaseR, RS, Not reported, None."
        },
        "advertised_fec": {
          "type": "string",
          "description": "Type of FEC advertised by this port"
        },
        "fec": {
          "type": "string",
          "description": "????"
        },
        "message_type": {
          "type": "string",
          "description": "Network protocol or service identifier used in port-related events. Value is port."
        },
        "state": {
          "type": "string",
          "description": "Status of the port, either up or down."
        }
      }
    },
    "Premises": {
```

```
      "type": "object",
      "required": [
        "name",
        "opid"
      ],
      "properties": {
        "opid": {
          "type": "integer",
          "format": "int32"
        },
        "name": {
          "type": "string"
        }
      },
      "description": "Premises"
    },
    "Route": {
      "description": "This module contains descirptions of the data collected and returned by the Route endpoint.",
      "type": "object",
      "required": [
        "schema"
      ],
      "properties": {
        "schema": {
          "$ref": "#/definitions/Schema"
        },
        "opid": {
          "type": "integer",
```

```
        "format": "int32",
        "description": "Internal use only"
      },
      "hostname": {
        "type": "string",
        "description": "User-defined name for a device"
      },
      "timestamp": {
        "type": "integer",
        "format": "int64",
        "description": "Date and time data was collected"
      },
      "vrf": {
        "type": "string",
        "description": "Name of associated virtual route forwarding (VRF) interface, if applicable"
      },
      "message_type": {
        "type": "string",
        "description": "Network protocol or service identifier used in route-related events. Value is route."
      },
      "db_state": {
        "type": "string",
        "description": "Internal use only"
      },
      "is_ipv6": {
        "type": "boolean",
        "description": "Indicates whether the IP address for this route is an IPv6
```

```
address (true) or an IPv4 address (false)"
        },
        "rt_table_id": {
          "type": "integer",
          "format": "int32",
          "description": "Routing table identifier for this route"
        },
        "src": {
          "type": "string",
          "description": "Hostname?? of device where this route originated"
        },
        "nexthops": {
          "type": "string",
          "description": "List of hostnames/interfaces/ports remaining to reach
destination????"
        },
        "route_type": {
          "type": "integer",
          "format": "int32",
          "description": "????"
        },
        "origin": {
          "type": "boolean",
          "description": "Indicates whether the source of this route is on the  device
indicated by 'hostname'????"
        },
        "protocol": {
          "type": "string",
          "description": "Protocol used for routing. Example values include BGP, ????"
```

```
    },
    "prefix": {
      "type": "string",
      "description": "Address prefix for this route"
    }
  }
},
"Schema": {
  "type": "object",
  "required": [
    "aliases",
    "doc",
    "elementType",
    "enumSymbols",
    "error",
    "fields",
    "fixedSize",
    "fullName",
    "hashCode",
    "jsonProps",
    "logicalType",
    "name",
    "namespace",
    "objectProps",
    "props",
    "type",
    "types",
    "valueType"
  ],
```

```
      "properties": {
       "props": {
        "type": "object",
        "additionalProperties": {
         "type": "string"
        }
       },
       "type": {
        "type": "string",
        "enum": [
         "RECORD",
         "ENUM",
         "ARRAY",
         "MAP",
         "UNION",
         "FIXED",
         "STRING",
         "BYTES",
         "INT",
         "LONG",
         "FLOAT",
         "DOUBLE",
         "BOOLEAN",
         "NULL"
        ]
       },
       "logicalType": {
        "$ref": "#/definitions/LogicalType"
       },
```

```
"hashCode": {
  "type": "integer",
  "format": "int32"
},
"elementType": {
  "$ref": "#/definitions/Schema"
},
"aliases": {
  "type": "array",
  "uniqueItems": true,
  "items": {
    "type": "string"
  }
},
"namespace": {
  "type": "string"
},
"fields": {
  "type": "array",
  "items": {
    "$ref": "#/definitions/Field"
  }
},
"types": {
  "type": "array",
  "items": {
    "$ref": "#/definitions/Schema"
  }
},
```

```
"fullName": {
  "type": "string"
},
"enumSymbols": {
  "type": "array",
  "items": {
    "type": "string"
  }
},
"doc": {
  "type": "string"
},
"valueType": {
  "$ref": "#/definitions/Schema"
},
"fixedSize": {
  "type": "integer",
  "format": "int32"
},
"name": {
  "type": "string"
},
"error": {
  "type": "boolean"
},
"jsonProps": {
  "type": "object",
  "additionalProperties": {
    "$ref": "#/definitions/JsonNode"
```

```
        }
       },
       "objectProps": {
        "type": "object",
        "additionalProperties": {
         "type": "object",
         "properties": {}
        }
       }
      }
     },
     "Sensor": {
      "description": "This model contains descriptions of the data collected and returned from the Sensor endpoint.",
      "type": "object",
      "required": [
       "schema"
      ],
      "properties": {
       "schema": {
        "$ref": "#/definitions/Schema"
       },
       "opid": {
        "type": "integer",
        "format": "int32",
        "description": "Internal use only"
       },
       "hostname": {
        "type": "string",
```

```
        "description": "User-defined name of the device where the sensor resides"
      },
      "timestamp": {
        "type": "integer",
        "format": "int64",
        "description": "Date and time data was collected"
      },
      "s_prev_state": {
        "type": "string",
        "description": "Previous state of a fan or power supply unit (PSU) sensor. Values include OK, absent, and bad."
      },
      "s_name": {
        "type": "string",
        "description": "Type of sensor. Values include fan, psu, temp.????"
      },
      "s_state": {
        "type": "string",
        "description": "Current state of a fan or power supply unit (PSU) sensor. Values include OK, absent, and bad."
      },
      "s_input": {
        "type": "number",
        "format": "float",
        "description": "????"
      },
      "message_type": {
        "type": "string",
        "description": "Network protocol or service identifier used in sensor-related
```

```
events. Value is sensor."
        },
        "s_msg": {
          "type": "string",
          "description": "Sensor message????"
        },
        "s_desc": {
          "type": "string",
          "description": "User-defined name of sensor. Example values include fan1,
fan-2, psu1, psu02, psu1temp1, temp2. ????"
        },
        "s_max": {
          "type": "integer",
          "format": "int32",
          "description": "Current maximum temperature threshold value"
        },
        "s_min": {
          "type": "integer",
          "format": "int32",
          "description": "Current minimum temperature threshold value"
        },
        "s_crit": {
          "type": "integer",
          "format": "int32",
          "description": "Current critical high temperature threshold value"
        },
        "s_lcrit": {
          "type": "integer",
          "format": "int32",
```

```
        "description": "Current critical low temperature threshold value"
       },
       "db_state": {
        "type": "string",
        "description": "Internal use only"
       },
       "active": {
        "type": "boolean",
        "description": "Indicates whether the identified sensor is operating (true) or not (false)"
       },
       "deleted": {
        "type": "boolean",
        "description": "Indicates whether the sensor ???? has been deleted (true) or not (false)"
       }
      }
    },
    "Services": {
     "description": "This model contains descriptions of the data collected and returned from the Sensor endpoint.",
     "type": "object",
     "required": [
      "schema"
     ],
     "properties": {
      "schema": {
       "$ref": "#/definitions/Schema"
      },
```

```
        "opid": {
          "type": "integer",
          "format": "int32",
          "description": "Internal use only"
        },
        "hostname": {
          "type": "string",
          "description": "User-defined name of the device where the network services
are running."
        },
        "name": {
          "type": "string",
          "description": "Name of the service; for example, BGP, OSPF, LLDP, NTP, and
so forth."
        },
        "vrf": {
          "type": "string",
          "description": "Name of the Virtual Route Forwarding (VRF) interface if
employed."
        },
        "timestamp": {
          "type": "integer",
          "format": "int64",
          "description": "Date and time data was collected"
        },
        "is_enabled": {
          "type": "boolean",
          "description": "Indicates whether the network service is enabled."
        },
```

```
        "is_active": {
          "type": "boolean",
          "description": "Indicates whether the network service is currently active."
        },
        "is_monitored": {
          "type": "boolean",
          "description": "Indicates whether the network service is currently being
monitored."
        },
        "status": {
          "type": "integer",
          "format": "int32",
          "description": "Status of the network service connection; up or down."
        },
        "start_time": {
          "type": "integer",
          "format": "int64",
          "description": "Date and time that the network service was most recently
started."
        },
        "db_state": {
          "type": "string",
          "description": "Internal use only"
        }
      }
    },
    "Vlan": {
      "description": "This model contains descriptions of the data collected and
returned by the VLAN endpoint.",
```

```
"type": "object",
"required": [
 "schema"
],
"properties": {
 "schema": {
  "$ref": "#/definitions/Schema"
 },
 "opid": {
  "type": "integer",
  "format": "int32",
  "description": "Internal use only"
 },
 "hostname": {
  "type": "string",
  "description": "User-defined name for a device"
 },
 "ifname": {
  "type": "string",
  "description": "User-defined name for a software interface"
 },
 "timestamp": {
  "type": "integer",
  "format": "int64",
  "description": "Date and time data was collected"
 },
 "last_changed": {
  "type": "integer",
  "format": "int64",
```

```
        "description": "Date and time the VLAN configuration was changed (updated,
deleted,???)"
       },
       "vlans": {
        "type": "string",
        "description": "List of other VLANs known to this VLAN or on this device????"
       },
       "svi": {
        "type": "string",
        "description": "Switch virtual interface (SVI) associated with this VLAN"
       },
       "db_state": {
        "type": "string",
        "description": "Internal use only"
       },
       "ports": {
        "type": "string",
        "description": "Names of ports on the device associated with this VLAN"
       }
     }
   }
  }
}
```