Cumulus NetQ 2.2

API User Guide

Table of Contents

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 CLI User Guide is available in PDF for offline viewing.

Contents

This topic describes...

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.
eyJ1c2VyIjoiYWRtaW4iLCJvcGlkIjowLCJyb2xlIjoiYWRtaW4iLCJleHBpcmVzQ
XQiOjE1NTYxMjUzNzgyODB9.
_D2Ibhmo_BWSfAMnF2FzddjndTn8LP8CAFFGIj5tn0A","customer_id":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\"}"

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
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 (see page).

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. Click netq-220.json to open the file.