User Manual

Table of contents

Introduction 3

Licensing 6

Installation Procedure 9

HTTPS/Authentication Mechanism 14

NMX Services Registration 20

Bring-Up 24

REST Interface 37

Key Performance Indicators (KPI) 39

Prometheus Endpoint 53

Alerts Management 55

Troubleshooting 61

Tech-Support Files Retrieval 62

Partitions Management 68

- Partition Health 72

- Introduction
- Licensing
- Installation Procedure
- HTTPS/Authentication Mechanism
- NMX Services Registration
- Bring-Up
- REST Interface
- Key Performance Indicators (KPI)
- Prometheus Endpoint
- Alerts Management
- Troubleshooting
- Tech-Support Files Retrieval
- Partitions Management

Introduction

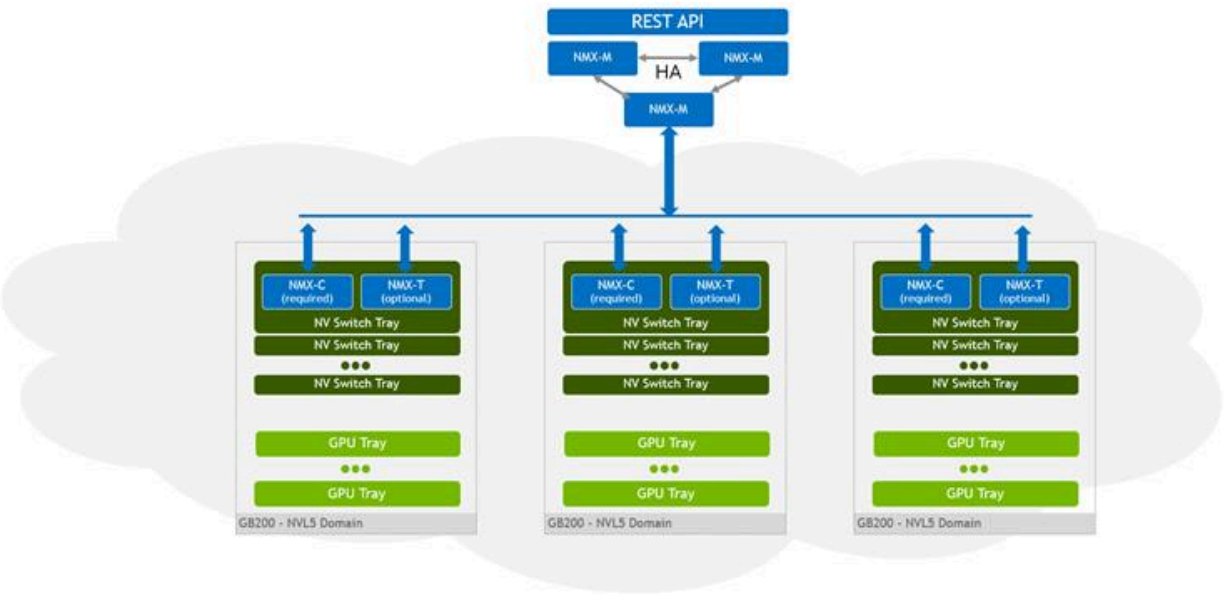
NMX Manager is the top-layer solution responsible for interacting with and controlling all instances of NMX-T and NMX-C within the network. It is a key component of the NMX solution, designed to collect and process data center telemetry, monitor system performance, and provide insights and predictive analytics regarding the operability and health of the systems.

The primary functions of NMX Manager are:

1. **Telemetry Aggregation:** To aggregate telemetry data streamed from the NMX Telemetry subsystem and make it accessible to external entities via a Prometheus endpoint.

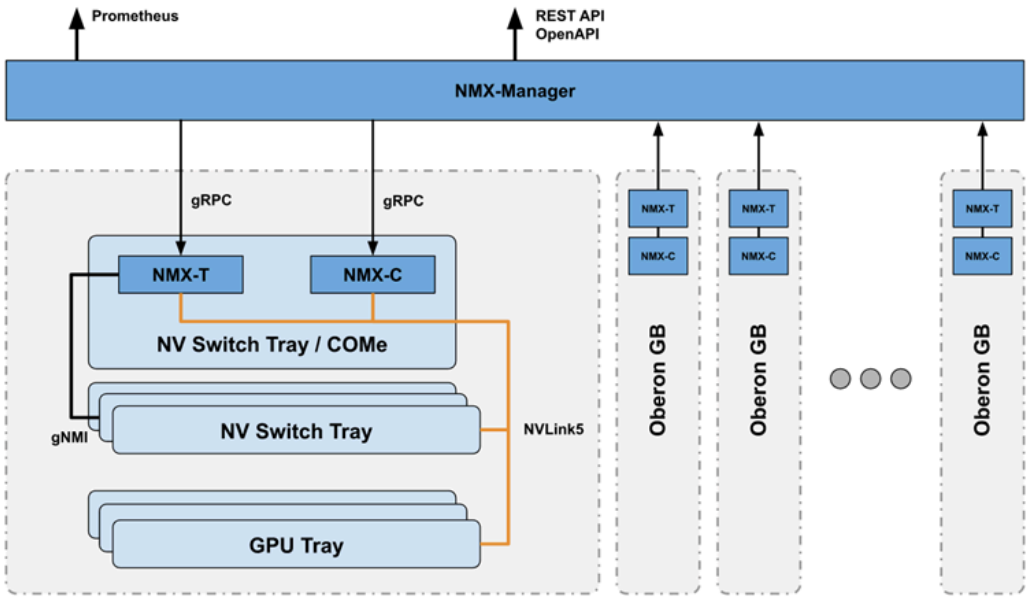
2. **Configuration and Resource Management:** To access all NMX-C instances for configuring the NNVL network and allocating/reserving GPUs for AI jobs/workloads.

NMX Manager is built on an event-driven microservice architecture, enabling seamless communication between microservices through Apache Kafka, which serves as the event bus.



Its components are:

- Southbound Gateway - a service that provides an interface for NMX-T and NMX-C
- Northbound Gateway - provides a REST API for query and provisioning



NMX Solution Topology

NMX-Manager can be installed as a cluster (minimum 3 node) on a customer's site. The NMX-Manager software will be provided as a tarball package along with the VM image.

> **Note**
>
> The majority of nodes must be operational for NMX-M to function. A three-node cluster can tolerate a one-node failure, but not a two-node failure.

Hardware and Software Requirements

Below are the hardware and software requirements for deploying the NMX-M VM:

Resource	Requirements
Processor	48 Cores

Resource	Requirements
Memory	512 GB RAM
Local disk storage	3TB NVMe storage Enterprise grade
Hypervisor	QCOW2 (QEMU Copy On Write) image for Ubuntu servers on QEMU/KVM
OS Support/Version	Ubuntu 24.04 LTS
Software	Ansible

Licensing

Evaluation Period

Upon installation, the system receives an evaluation license **valid for 60 days**.

When the evaluation license expires, REST API access is blocked until a new license is applied.

Applying a New License

To obtain a new license file, visit the NVIDIA Licensing Portal: https://ui.licensing.nvidia.com.

To replace the existing license, perform the following steps:

1. Upload the received license file to the directory:

   ```
   /opt/nvidia/nmx/licenses
   ```

   on any server.

2. Run the following script:

   ```
   /opt/nvidia/nmx/scripts/license-config.sh
   ```

3. Select option 1 `Apply new license (replace existing)` .

4. Choose license file from the list

5. Confirm that the displayed new license details are correct.

6. Wait a few moments for the license to be applied.

The script applies the new license, replacing the previous license.

Viewing Active License Details

1. Run the following script:

```
/opt/nvidia/nmx/scripts/license-config.sh
```

2. Select option **2** `Get active license information` .

3. Wait a few moments for the active license details to be displayed.

The script outputs the license **type** (Evaluation, Permanent, Subscription), **issue date**, and **expiration date**.

Receiving Notifications

The system sends notifications when the license is about to expire or has already expired. Notifications repeat every 24 hours until the license status is updated.

To receive these notifications, configure a webhook receiver during or after the installation (see Alerts Management section).

Types of Notifications

Users receive the following alerts regarding license status:

- **Expires soon:** Sent when the license expires in less than 60 days.

Job	**licensing**
Alert Name	LicenseValidation
Severity	warning

- **Expired:** Sent when the license has expired.

Job	**licensing**
Alert Name	LicenseValidation

Job	licensing
Severity	critical

License Expiration

Upon license expiration:

- The system sends an expiration notification and logs this event.
- REST API access is blocked until a valid new license is applied.

Installation Procedure

> **Note**
>
> In this NMX-M version (85.1.3000), the VM image is named using the `netq` prefix (e.g., `netq-4.15.0-...`).
>
> This naming reflects internal packaging alignment with NetQ, but it does **not** affect product functionality. The image installs NMX-M as described in this documentation.

> **Note**
>
> Make sure that NMX-M is being deployed in a secure environment.

Downloading the Product Image

1. Log in to your NVIDIA Application Hub account.
2. Select **NVIDIA Licensing Portal**.
3. Select **Software Downloads** from the menu.
4. In the search field above the table, enter **NetQ**.
5. For deployments using KVM, download the **NetQ SW 4.15 KVM** image (which includes the NMX-M image). For deployments using VMware, download the **NetQ SW 4.15 VMware** image.
6. If prompted, read the license agreement and proceed with the download.

Installing the VM Image

To install the VM image, follow the steps below:

1. Mount the VM image (`.qcow2` file) using the appropriate KVM commands.

   Example: Copy the Base image to `- /var/lib/libvirt/images/`

   ```
   virt-install --connect qemu:///system --ram 524288 -n ubuntu1 --os-type linux --os-variant generic --vcpus=48 --disk path=/var/lib/libvirt/images/netq-VERSION-ubuntu-24.04-ts-qemu-scale.qcow2,format=qcow2,device=disk,bus=virtio --vnc --noautoconsole --import --network=bridge=br0,model=virtio,mac=<MAC> --check all=off
   ```

2. Log in to the `nvidia` user account on each of the three servers using the default password `nvidia`

3. Upon first login, you will be prompted to set a new password.

   Use the same new password on all three servers.

   Note: Changing the password will automatically log you out.

4. Re-login as the `nvidia` user using the new password on only one of the servers.

   The remaining installation steps will be performed only on this server.

> **Note**
>
> The vmimage .qcow2 image already contains NVL tarball at `/mnt/installables/NetQ-4.15.0.tgz` and the CLI to install it.

Installing NetQ Manager (NVL)

Pre-requisites

- Three VMs running Ubuntu 24.04
- Minimum hardware per VM:
    - 48 vCPUs
    - 512 GB RAM
    - 3 TB disk

The install process performs the following pre-checks:

- Validates minimum hardware requirements
- Ensures all server IPs are unique
- Verifies master/worker initialization
- Confirms reachability of worker IPs

How To Run the Installer

1. **Initialize the master node.**

   On the VM where you want to run the installation, execute:

   ```
   netq install cluster master-init
   ```

   This creates the `netq-admin` user, generates an SSH key pair, and outputs a command to initialize the worker nodes using the encoded public key.

2. **Initialize the worker nodes.**

   On each of the other two VMs, run the command generated in Step 1:

```
netq install cluster worker-init <ssh-key>
```

3. **Install NVL using the CLI method** (optional alternative in steps 4-6):

```
netq install nvl servers <server-01> <server-02> <server-03> \
bundle </mnt/installables/<netq-image-name>, http URL> \
kong-rw-password <kong-rw-password> \ kong-ro-password <kong-
ro-password> \ alertmanager-webhook-url <webhook-url>
```

Required arguments:

- IPs of all three target servers
- Path to the NetQ tarball bundle
- Kong read/write and read-only passwords

Optional argument (will default to empty string if omitted):

- Alertmanager webhook URL

OR:

a. Generate a config JSON template (for config-based installation):

* To generate the config file at the default location: {

```
netq install nvl config generate
```

This creates a file at `/tmp/nvl-cluster-config.json`.

* To specify a custom output location:

```
netq install nvl config generate <full-path-to-config-json>
```

This creates the file at the specified path, e.g., `/home/nvidia/nvl-config.json`

b. Edit the config JSON file.

Replace `<INPUT>` placeholders with:

-
  -
    - Server IP addresses
    - Kong user passwords
    - Alertmanager webhook URL

c. Install NVL using the config file:

```
netq install nvl bundle </mnt/installables/<netq-image-name> \
kong-rw-password <kong-rw-password> \ kong-ro-password <kong-ro-
password> \ <full-path-to-config-json>
```

Post Installation

4. **Register the services.**

-
  - For **manual** mTLS setup and registration, see the NMX Services Registration page.
  - For **automated** configuration, see the Bring-Up page.

HTTPS/Authentication Mechanism

In NMX-M, security and user authentication are critical components achieved through the use of HTTPS and a Kong Basic Authentication mechanism. HTTPS (Hypertext Transfer Protocol Secure) is employed to protect data transmitted between clients (such as web browsers) and the server by encrypting this data using SSL/TLS protocols. HTTPS ensures that sensitive information, such as login credentials and personal details is safeguarded from interception or tampering by unauthorized parties. When a client connects to a server over HTTPS, the server presents a digital certificate verified by a trusted Certificate Authority (CA). This certificate authenticates the server's identity and establishes a secure connection, ensuring data integrity, confidentiality, and authentication.

In addition to HTTPS, our system utilizes a basic authentication approach with pre-defined users that are set upon installation. This provides a flexible and secure way to authenticate and authorize users interacting with our REST API. To implement this, we use the Kong API gateway as a reverse proxy, configured with Basic Authentication and ACL plugins. This setup allows Kong to authenticate users attempting to access specific resources using the existing user accounts on the system.

By combining HTTPS and a robust authentication mechanism, our system provides a secure and reliable environment for users, protecting their data and ensuring proper access controls.

Authentication

In NMX-M, there are two users configured for interacting with the API:

- `ro-user`
- `rw-user`

The password for each user is set during the NMX-M cluster installation process. Those users are configured in Kong’s basic authentication settings and applied to all routes. The login information is encrypted and kept in a dedicated PostgreSQL DB.

Authorization

Each user has its own ACL group configuration.

Ready Only ACL Group

This group includes the "ro-user" and grants access to read-only API endpoints. Any NMX API endpoint that uses the GET HTTP method can be accessed here.

Relevant API Endpoints

Group	Operation	Method	URL
KPI	Get KPI	GET	/nmx/v1/kpis
Metric	Get Metrics	GET	/nmx/v1/metrics
Services	List NMX Services	GET	/nmx/v1/services
	Get NMX Service	GET	/nmx/v1/services/{id}
Compute Nodes	List Compute Nodes	GET	/nmx/v1/compute-nodes
	Get Compute Nodes Count	GET	/nmx/v1/compute-nodes/count
	Get Compute Node	GET	/nmx/v1/compute-nodes/{id}
Switch Nodes	List Switch Nodes	GET	/nmx/v1/switch-nodes
	Get Switch Nodes Count	GET	/nmx/v1/switch-nodes/count
	Get Switch Node	GET	/nmx/v1/switch-node/{id}
Switches	List Switches	GET	/nmx/v1/switches
	Get Switches Count	GET	/nmx/v1/switches/count
	Get Switch	GET	/nmx/v1/switches/{id}
Chassis	List Chassis	GET	/nmx/v1/chassis
	Get Chassis Count	GET	/nmx/v1/chassis/count
	Get Chassis	GET	/nmx/v1/chassis/{id}
Ports	List Ports	GET	/nmx/v1/ports
	Get Ports Count	GET	/nmx/v1/ports/count
	Get Port	GET	/nmx/v1/ports/{id}
GPU	List GPUs	GET	/nmx/v1/gpus

Group	Operation	Method	URL
	Get GPU Count	GET	/nmx/v1/gpus/count
	Get GPU	GET	/nmx/v1/gpus/{id}
Operations	List Operations	GET	/nmx/v1/operations
	Get Operation	GET	/nmx/v1/operations/{id}

Ready Write ACL Group

Includes the "rw-user" and allows access to all API endpoints. Any NMX API endpoint, regardless of its HTTP method, can be accessed here.

Relevant API Endpoints

In addition to the above GET endpoints.

Group	Operation	Method	URL
Services	Add NMX Service	POST	/nmx/v1/services
	Delete NMX Service	DELETE	/nmx/v1/services/{id}
Compute Nodes	Update Compute Node	PUT	/nmx/v1/compute-nodes/{id}
Switch Nodes	Update Switch Node	PUT	/nmx/v1/switch-nodes/{id}
Switches	Update Switch	PUT	/nmx/v1/switches/{id}
Chassis	Update Chassis	PUT	/nmx/v1/chassis/{id}
GPU	Update GPU	PUT	/nmx/v1/gpus/{id}
Operations	Cancel Operation	DELETE	/nmx/v1/operations/{id}

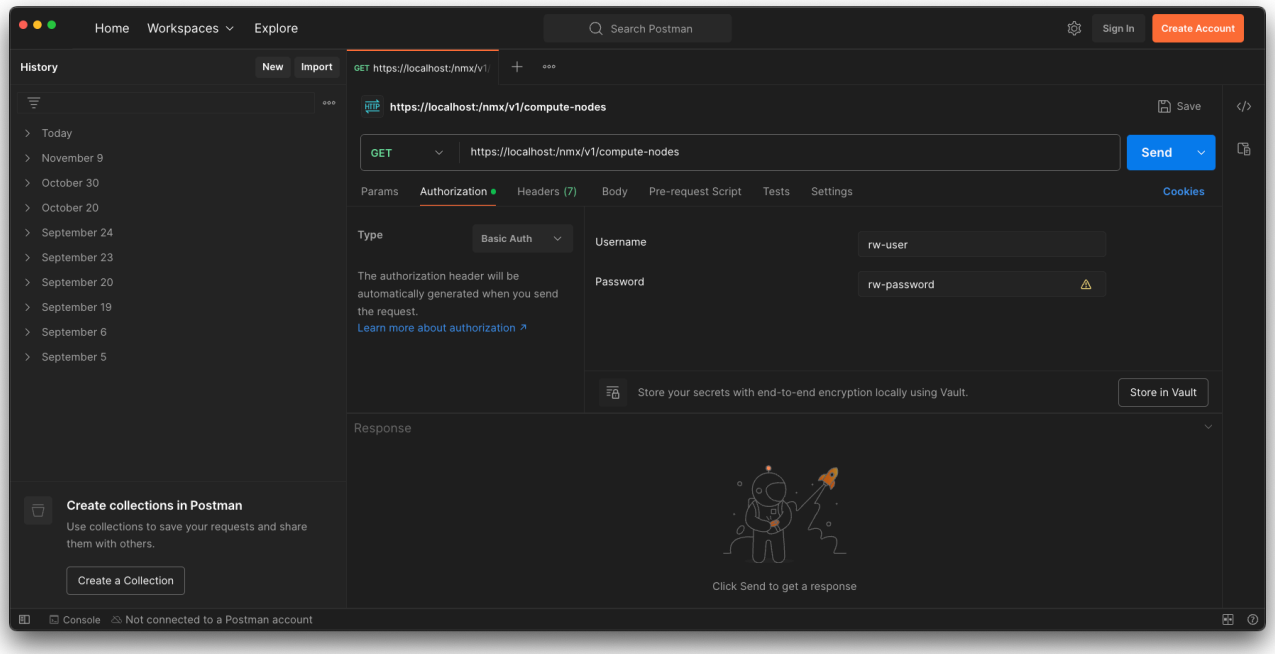
Examples

Read Only Endpoint

Postman

Authorization settings tab:

1. Select type: Basic Auth
2. Username: rw-user ; Password: <password defined during cluster installation>



Terminal

1. In a terminal window, use "bash plus curl" to execute requests.
2. Run the following curl command, enter values for the various parameters.

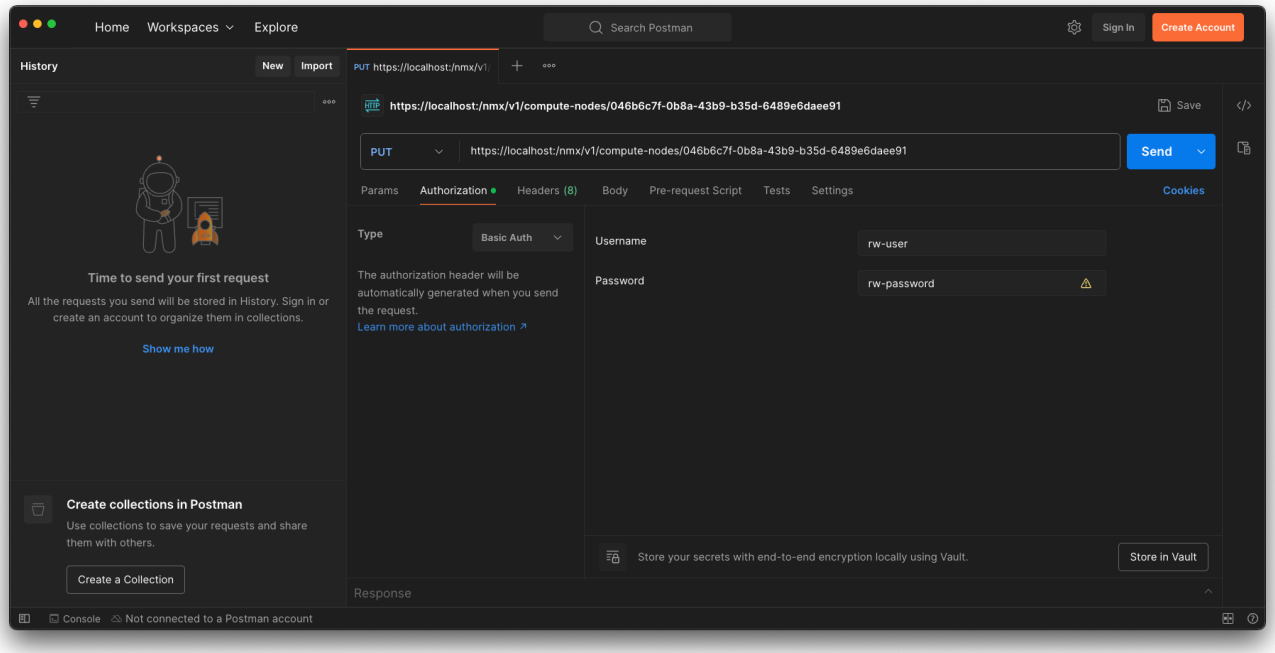
```
curl -X 'GET' \
  'https://<ip_address>/nmx/v1/compute-nodes' \
  -u ro-user:ro-password
```

Read Write Endpoint

Postman

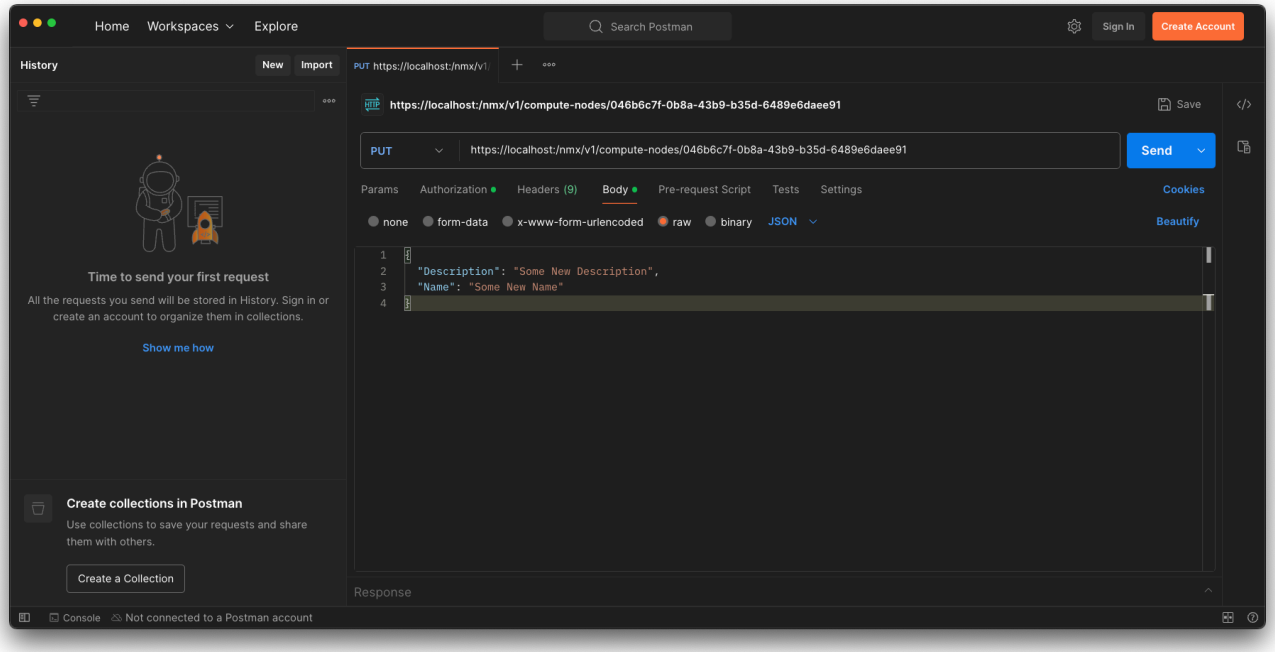
Authorization settings tab:

1. Select type: Basic Auth
2. Username: rw-user ; Password: <password defined during cluster installation>



Example Request Body

1. Select Method: PUT
2. Select type: JSON
3. Fill in the request body details as seen below.



Terminal

1. In a terminal window, use "bash plus curl" to execute requests.
2. Run the following curl command, and enter values for the various parameters.

```
curl -X 'PUT' \
  'https://<ip_address>/nmx/v1/compute-nodes/<id>' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -u rw-user:rw-password \
  -d '{
  "Description": "Some New Description",
  "Name": "Some New Name"
}
```

NMX Services Registration

In order for NMX-M to receive network telemetry and control information, NMX-M needs to register with NMX-T and NMX-C.

NMX Manager communicates with those services using mTLS encryption, which requires certificates to be configured.

During installation, mTLS certificates are automatically created and configured.

To have certificates configured for switch-trays that host NMX-C and NMX-T, use `create-certificate.sh` that can be found at this path: `/opt/nvidia/nmx/Installation/Ansible/tools`

Generate Certificates via NMX Manager

1. Log in via SSH to the NMX Manager node that was used to run the NMX Manager installation process.

```
ssh nvidia@<IP_Address>
```

2. Use the `create-certificate.sh` script as root. Provide a certificate name, typically the switch's hostname or IP address.

```
nvidia@ubuntu:~$ su
root@ubuntu:/home/nvidia# /opt/nvidia/nmx/Installation/Ansible/tools/create-certificate.sh example
certificate.cert-manager.io/example-certificate created
Certificate is ready after 10 seconds.
Extracting secret data to local files...
Files created:
-rw-r--r-- 1 root root 1094 May 29 11:57 example-ca.crt
-rw-r--r-- 1 root root 1424 May 29 11:57 example-tls.crt
```

```
-rw-r--r-- 1 root root 3243 May 29 11:57 example-tls.key
-rw------- 1 root root 3907 May 29 11:57 example-tls.p12
Done.
```

3. Copy `example-ca.crt` and `example-tls.p12` to the NVOS switch-tray and configure NMX services to use it.

4. Follow NVIDIA NVOS User Manual for NVLink Switches → NVLink Switching → Cluster Management → Cluster Manager → Cluster Manager Commands to apply the certificates to both NMX-C and NMX-T.

Registration

This can be accomplished by sending a POST request to the `nmx/v1/services` API using the following format:

Example POST Request:

```
{
  "Name": "Some-Name",
  "Description": "This is a description.",
  "ServiceType": "TELEMETRY",
  "ServiceConnectionInformation": {
    "Address": "10.188.47.166",
    "PortNumber": 9351
  }
}
```

Equivalent cURL Command:

```
curl --request POST \
  --url https://NMX_M-IP/nmx/v1/services \
  --header 'Authorization: Basic cnctdXNlcjpOdmlkaWExMg==' \
  --header 'Content-Type: application/json' \
```

```
  --data '{
  "Name": "Some Name",
  "Description": "Sample Description",
  "ServiceType": "CONTROLLER",
  "ServiceConnectionInformation": {
    "Address": "10.188.47.166",
    "PortNumber": 9370
  }
}'
```

Note: The **Authorization** header will use either `rw-user` or `ro-user` along with the corresponding password.

Checking the Status of Registered Services

To retrieve the status of all registered services, perform a **GET** request to the same API.

Example Response:

```
[
  {
    "Status": "UP",
    "Description": "Description",
    "Address": "10.188.47.166",
    "UpSince": "2000-01-23T04:56:07.000+00:00",
    "Name": "Name",
    "ApplicationUUID": "046b6c7f-0b8a-43b9-b35d-6489e6daee91",
    "Version": "Version",
    "StatusInfo": "StatusInfo",
    "ServiceType": "TELEMETRY",
    "ID": "551137c2f9e1fac808a5f572",
    "ClusterDomainUUID": "046b6c7f-0b8a-43b9-b35d-6489e6daee91",
    "PortNumber": 9351,
    "RegisteredAt": "2000-01-23T04:56:07.000+00:00"
  }
```

```
]
```

Equivalent cURL Command:

```
curl --request GET \
  --url https://NMX-M-IP/nmx/v1/services \
  --header 'Authorization: Basic cnctdXNlcjpOdmlkaWExMg=='
```

Note: The **Authorization** header will use either `rw-user` or `ro-user` along with the corresponding password.

Filtering Services

You can filter the response to retrieve only **Telemetry** or **Controller** services. For example, the following command retrieves only Telemetry services:

```
curl --request GET \
  --url https://NMX-M-IP/nmx/v1/services \TELEMETRY&offset=0&limit=1' \
  --header 'Authorization: Basic cnctdXNlcjpOdmlkaWExMg=='
```

Bring-Up

Overview

The Bring-Up process is a fully automated method for configuring and registering a designated switch tray with the NMX Manager. This process enables telemetry collection and NVLink domain management.

Each switch tray is configured to ensure both `nmx-controller` and `nmx-telemetry` services are active, and a mutual LTS (mTLS)-secured gRPC connection is established and maintained.

Switch Profile API

The NMX Manager provides endpoints for securely managing switch credentials via switch profiles.

Default Profile: username: `admin`, password: `admin`, (NVIDIA recommends to change or override for enhanced security)

Auth Required	Action	Endpoint	Description
ro-user, rw-user	Retrieve	`GET /nmx/v1/switch-profiles`	Retrieve a list of switch profiles
rw-user	Create	`POST /nmx/v1/switch-profiles`	Create a new switch profile
ro-user, rw-user	Retrieve	`GET /nmx/v1/switch-profiles/{id}`	Retrieve a specific switch profile
rw-user	Delete	`DELETE /nmx/v1/switch-profiles/{id}`	Delete a switch profile (except default)
rw-user	Update	`PATCH /nmx/v1/switch-profiles/{id}`	Update an existing switch profile `curl -X 'PATCH' \`

Auth Required	Action	Endpoint	Description
			'https://<NMX-Manager-API>/nmx/v1/switch-profiles/<switch-profile-id>' \ -H 'accept: */*' \ -H 'Content-Type: application/json' \ -d '{ "Password": "admin" }'

Bring-Up API

Bring-up is an asynchronous operation that tracks the bring-up process for one or more switches.

Auth Required	Action	Endpoint	Description
ro-user, rw-user	Retrieve	`GET /nmx/v1/bring-up`	Retrieve bring-up operations with optional filters (`pending`, `in-progress`, `failed`, `completed`)
rw-user	Create	`POST /nmx/v1/bring-up`	Initiate a new bring-up process for one or more switches
ro-user, rw-user	Retrieve	`GET /nmx/v1/bring-up/{id}`	Get bring-up status for a specific operation

Usage Examples

The following example demonstrates how to start a new bring-up operation using a POST request:

```
curl -X 'POST' \
  'https://<NMX-Manager-API>/nmx/v1/bring-up' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'FmConfig=@<fm-config-file>' \
  -F 'ProfileID=<switch-profile-id>' \
  -F 'Switches={
  "Address": ""
}'

curl -X 'POST' \
  'https://<NMX-Manager-API>/nmx/v1/bring-up' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'FmConfig=@<fm-config-file>' \
  -F 'ProfileID=' \
  -F 'Switches={
  "Address": "<switch-A-IP-Address-or-hostname>"
}' \
  -F 'Switches={
  "Address": "<switch-B-IP-Address-or-hostname>"
}'

curl -X 'POST' \
  'https://<NMX-Manager-API>/nmx/v1/bring-up' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'FmConfig=@<fm-config-file>' \
  -F 'ProfileID=' \
  -F 'Switches={
  "Address": "<switch-A-IP-Address-or-hostname>"
}' \
  -F 'Switches={
  "Address": "<switch-B-IP-Address-or-hostname>",
  "ProfileID": "<custom-switch-profile-id>"
```

```
}'

curl -X 'POST' \
  'https://<NMX-Manager-API>/nmx/v1/bring-up' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'FmConfig=@fm_config_72x1_C9_S9 2.cfg' \
  -F 'ProfileID=' \
  -F 'Switches={
  "Address": "<switch-A-IP-Address-or-hostname>",
  "ProfileID": "<custom-switch-profile-A-id>"
}' \
  -F 'Switches={
  "Address": "<switch-B-IP-Address-or-hostname>",
  "ProfileID": "<custom-switch-profile-B-id>"
}'
```

fm_config / SDN Config

- Provide an `fm_config` / SDN config file that matches all switches included in this POST request.
- For multiple topologies, use separate POST requests for each topology.

Find more information and examples:

- NVIDIA NVOS User Manual for NVLink Switches → NVLink Switching → SDN → SDN Configuration
- NVIDIA NVOS User Manual for NVLink Switches → NVLink Switching → Cluster Management → Cluster Control → Cluster Control Commands

Switch Profile

- Ensure you have a switch profile with the required switch credentials. If not, create one in advance via `POST /nmx/v1/switch-profiles`.

- A single bring-up request may specify a global switch profile for all switches, or a separate switch profile for each listed switch.
- If a switch profile is specified for a particular switch, it takes precedence over the global profile specified in the request.

If all initial validations pass, A `HTTP 202 Accepted` response is returned with a JSON body containing a bring-up operation ID to track the process:

```
{
        "operationId": "682880baaf653727786b618f"
}
```

To track operation progress:

```
curl -X 'GET' \
  'https://<NMX-Manager-API>/nmx/v1/bring-up/682880baaf653727786b618f' \
  -H 'accept: application/json'
```

Step-by-Step Bring-Up Workflow

Initial Response

When the bring-up process has not yet started, the switch status is marked as "pending." It changes as soon as the NMX Manager begins the bring-up process.

```
{
    "CreatedAt": "2025-05-15T07:07:31.428Z",
    "ID": "682880baaf653727786b618f",
    "Status": "in-progress",
    "Switches": [
        {
```

```
                "Address": "<switch-IP-Address-or-hostname>",
                "CurrentStep": "Initial bring-up task",
                "StartedAt": "2025-05-15T07:07:31.485Z",
                "Status": "pending",
                "StatusDetails": "",
                "UpdatedAt": "2025-05-15T07:07:31.485Z"
            }
        ],
        "UpdatedAt": "2025-05-15T07:07:31.485Z"
}
```


Bring-Up Execution Steps

Step 1: Pre Bring-up Validation

The NMX manager sends an API request to the switch tray to check whether the `nmx-controller` and `nmx-telemetry` services are already active.

If detected, the bring-up process is skipped to avoid overwriting the existing configuration. Bring-up can only be performed once per switch. Even if services are later shut down, the NMX Manager remembers and blocks repeated bring-up attempts, unless the services are explicitly deregistered from the NMX Manager by the user.


```
{
        "CreatedAt": "2025-05-15T07:07:31.428Z",
        "ID": "682880baaf653727786b618f",
        "Status": "in-progress",
        "Switches": [
            {
                "Address": "<switch-IP-Address-or-hostname>",
                "CurrentStep": "Step 1: Is switch configured request",
                "StartedAt": "2025-05-15T07:07:31.485Z",
                "Status": "in-progress",
                "StatusDetails": "Step 1: Is switch configured request: sent to switch-gateway.",
                "UpdatedAt": "2025-05-15T07:07:31.515Z"
```

```
            }
        ],
        "UpdatedAt": "2025-05-15T07:07:31.515Z"
}
```


Step 2: Enable Cluster

The NMX Manager instructs the switch tray to start the `nmx-controller` and `nmx-telemetry` services required for cluster operations.


```
{
        "CreatedAt": "2025-05-15T07:07:31.428Z",
        "ID": "682880baaf653727786b618f",
        "Status": "in-progress",
        "Switches": [
            {
                "Address": "<switch-IP-Address-or-hostname>",
                "CurrentStep": "Step 2: Enable cluster request",
                "StartedAt": "2025-05-15T07:07:31.485Z",
                "Status": "in-progress",
                "StatusDetails": "Step 2: Enable cluster request: sent to switch-gateway.",
                "UpdatedAt": "2025-05-15T07:07:37.278Z"
            }
        ],
        "UpdatedAt": "2025-05-15T07:07:37.278Z"
}
```


Step 3: Import And Configure Certificates

The NMX Manager generates the required certificates and sends them to the switch. Certificates are stored locally and configured for both `nmx-controller` and `nmx-telemetry`.

Each configuration action is processed asynchronously via the NVOS API, and job success is verified by polling.

```
{
    "CreatedAt": "2025-05-15T07:07:31.428Z",
    "ID": "682880baaf653727786b618f",
    "Status": "in-progress",
    "Switches": [
        {
            "Address": "<switch-IP-Address-or-hostname>",
            "CurrentStep": "Step 3: Import certificates request",
            "StartedAt": "2025-05-15T07:07:31.485Z",
            "Status": "in-progress",
            "StatusDetails": "Step 3: Import certificates request: sent to switch-gateway.",
            "UpdatedAt": "2025-05-15T07:07:59.62Z"
        }
    ],
    "UpdatedAt": "2025-05-15T07:07:59.62Z"
}
```


Step 4: Enable mTLS For NMX Services

The NMX Manager sends a request to configure both services for mTLS encryption, ensuring secure gRPC communication. Each operation is tracked and validated.


```
{
    "CreatedAt": "2025-05-15T07:07:31.428Z",
    "ID": "682880baaf653727786b618f",
    "Status": "in-progress",
    "Switches": [
        {
            "Address": "<switch-IP-Address-or-hostname>",
            "CurrentStep": "Step 4: Enable encryption request",
            "StartedAt": "2025-05-15T07:07:31.485Z",
            "Status": "in-progress",
            "StatusDetails": "Step 4: Enable encryption request: sent to switch-gateway.",
            "UpdatedAt": "2025-05-15T07:08:39.559Z"
```

```
            }
        ],
        "UpdatedAt": "2025-05-15T07:08:39.559Z"
}
```


Step 5: Import and Configure SDN Config

The NMX Manager uploads the SDN config (`fm_config`) provided in the bring-up request to the switch file system. `nmx-controller` is configured accordingly. Jobs are tracked and confirmed.


```
{
        "CreatedAt": "2025-05-15T07:07:31.428Z",
        "ID": "682880baaf653727786b618f",
        "Status": "in-progress",
        "Switches": [
            {
                "Address": "<switch-IP-Address-or-hostname>",
                "CurrentStep": "Step 5: Install SDN request",
                "StartedAt": "2025-05-15T07:07:31.485Z",
                "Status": "in-progress",
                "StatusDetails": "Step 5: Install SDN request: sent to switch-gateway.",
                "UpdatedAt": "2025-05-15T07:09:02.492Z"
            }
        ],
        "UpdatedAt": "2025-05-15T07:09:02.492Z"
}
```


Step 6: Wait for NMX Controller Status And Register

The NMX Manager polls the controller until its `addition-info` field reports `CONTROL_PLANE_STATE_CONFIGURED`. Once confirmed, registration begins using a secure gRPC connection.

```
{
    "CreatedAt": "2025-05-15T07:07:31.428Z",
    "ID": "682880baaf653727786b618f",
    "Status": "in-progress",
    "Switches": [
        {
            "Address": "<switch-IP-Address-or-hostname>",
            "CurrentStep": "Step 6: Wait configured request",
            "StartedAt": "2025-05-15T07:07:31.485Z",
            "Status": "in-progress",
            "StatusDetails": "Step 6: Wait configured request: sent to switch-gateway.",
            "UpdatedAt": "2025-05-15T07:09:17.31Z"
        }
    ],
    "UpdatedAt": "2025-05-15T07:09:17.31Z"
}
```


Step 7: Register NMX Telemetry

NMX Manager initiates registration of `nmx-telemetry` over a secure gRPC connection. This is also tracked and processed by backend components.


```
{
    "CreatedAt": "2025-05-15T07:07:31.428Z",
    "ID": "682880baaf653727786b618f",
    "Status": "in-progress",
    "Switches": [
        {
            "Address": "<switch-IP-Address-or-hostname>",
            "CurrentStep": "Registration request",
            "StartedAt": "2025-05-15T07:07:31.485Z",
            "Status": "in-progress",
            "StatusDetails": "Step 7: NMX Telemetry Registration request: sent to inventory.",
```

```
                "UpdatedAt": "2025-05-15T07:09:55.687Z"
            }
        ],
        "UpdatedAt": "2025-05-15T07:09:55.687Z"
}
```


Final Result: Registration Completed

If successful, both `nmx-controller` and `nmx-telemetry` are registered, and their ObjectIDs are returned.


```
{
        "CreatedAt": "2025-05-15T07:18:06.216Z",
        "ID": "682880baaf653727786b618f",
        "Status": "completed",
        "Switches": [
            {
                "Address": "<switch-IP-Address-or-hostname>",
                "CurrentStep": "Registration response",
                "NMX-Controller-ID": "682595b7799bc550eec18a77",
                "NMX-Telemetry-ID": "682595b8799bc550eec18a78",
                "StartedAt": "2025-05-15T07:18:06.298Z",
                "Status": "completed",
                "StatusDetails": "bring-up completed successfully",
                "UpdatedAt": "2025-05-15T07:20:25.754Z"
            }
        ],
        "UpdatedAt": "2025-05-15T07:20:25.761Z"
}
```


Operational Considerations

Asynchronous Processing

- All POST requests return HTTP `202 Accepted` with an `operationId` and a `Location` header.
- Operations continue until each sub-task (per switch) reaches a terminal state: `failed` or `completed`.
- Sub-tasks progress independently and in parallel.

Cancellation

- Bring-up operations cannot be canceled once started.

Timeouts

-
    - If no progress is made within 10 minutes (configurable), the operation is marked as failed.
    - As long as one sub-task is progressing, the operation continues. Progress is indicated by the `UpdatedAt` field.

Troubleshooting

If bring-up fails, perform a cleanup:

1. SSH into the switch.
2. Retrieve existing certificates:

```
nv show system security ca-certificate
nv show system security certificate
```

3. Delete NMX-added certificates:

```
nv action delete system security certificate
<certificate_name>
nv action delete system security ca-certificate
<ca_certificate_name>
```

4. Remove SDN config:

```
nv action delete sdn config apps nmx-controller type
fm_config files fm_config.cfg
```

5. Reset SDN configuration:

```
nv action reset sdn factory-default
```

6. Disable cluster state:

```
nv set cluster state disabled
nv config apply
```

7. Clean up certificate and config files:

```
rm /tmp/cert.p12 /tmp/ca-cert.crt /tmp/fm_config.cfg
```

REST Interface

The API follows the OpenAPI 3.0 specification, providing a solid foundation for the REST API documentation. It offers detailed information about the API's capabilities, usage, and integration, enabling developers to build custom applications that interact with the NMX platform.

With a standard, language-agnostic interface, the API ensures easy integration and a smooth development experience.

The API provides access to resources and functionalities through a set of defined endpoints. Each endpoint has its own schema, which includes the payload, parameters, and the expected request/response formats.

For more information on using the NMX-M REST API, refer to the documentation through the Swagger interface here. The full object model can also be downloaded here.

Known Limitations

Support of the uint64 Format

NMX-M REST API includes fields in the numeric uint64 format. Tools such as Swagger UI have limited JSON parsing capabilities and cannot parse these values. In particular, Swagger UI uses native Javascript JSON.parse, which cannot properly interpret such large numbers.

It is recommended that, when developing browser tools that need to parse JSON containing uint64 format, you preprocess JSON first, as shown below:

```
const preprocessed = jsonAsString.replace(/:\s*(\d{16,})/g, ': "$1"');
const result = JSON.parse(preprocessed, (key, value) => typeof value === 'string' && /:\s*(\d{16,})/g.test(value) ? BigInt(value) : value)
```

Rendering and Performance

- Large size of some endpoints' responses may cause slow loading or unresponsiveness, or even freeze.

This is a browser and JavaScript limitation. Use other tools like Insomnia, Postman or Terminal to get this scale of data.

- Complex schemas with deep nesting can lead to rendering errors.

Key Performance Indicators (KPI)

The KPI REST endpoint is designed to provide fast and actionable insights into the network cluster's status. This API will offer key performance indicators (KPIs) that give a clear overview of the cluster's health, with the option to dive deeper into specific events. By using this endpoint, NOC operators can quickly identify network issues, assess their severity, and take immediate action, such as opening tickets for further investigation. The KPIs will be crucial for monitoring network performance and ensuring prompt response times.

This KPI is an aggregate across all domains. For troubleshooting, users should access per-domain data using compute-node or switch-node REST API calls.

Accessing API via Curl / Wget

```
curl --request GET \
  --url https://xxxx.xxxx.xxxx.xxxx/nmx/v1/kpis \
  --user "<username>:<password>" --insecure
```

```
wget --quiet \
  --method GET \
  --user '<username>' \
  --password '<password>' \
  --output-document \
  - https://xxxx.xxxx.xxxx.xxxx/nmx/v1/kpis --no-check-certificate
```

Note

Replace `<username>` with either `rw-user` (read-write access) or `ro-user` (read-only access).

Replace `<password>` with the actual password associated with the user.

API Filter Parameters

REST API supports filtering based on different values of the *filter* parameter. The allowed values are:

- Filtering based on Health:
  - HEALTH
  - SWITCH_HEALTH
  - GPU_HEALTH
  - DOMAIN_HEALTH
  - COMPUTE_HEALTH
- Filtering based on Inventory:
  - INVENTORY
  - COMPUTE_ALLOCATION
  - CONNECTION_COUNT
  - CABLE_TYPE
  - CABLE_PN
  - CABLE_FW_VERSION
  - PORT_COUNT
  - LINK_UP_COUNT

- LINKDOWN_FREQUENCY
- LINKDOWN_RATE
- CHIP_TEMPERATURE
- EFF_BER
- SYMBOL_BER
- RAW_BER

The API mentioned above supports filtering by appending the filter parameter to the endpoint, such as: `kpis?filter=HEALTH`

Sample Output

- **With filtering:**

```
curl --request GET --url
https://xxxx.xxxx.xxxx.xxxx/nmx/v1/kpis?filter=SWITCH_HEALTH --
user "<username>:<password>" --insecure
```


```
{
    "Data": [
        {
            "HEALTHY": 18
        },
        {
            "MISSING_NVLINK": 0
        },
        {
            "UNHEALTHY": 0
        },
        {
            "UNKNOWN": 0
        }
    ],
```

```
        "Description" :  "Number of switch instances per health state",
        "Title" :  "Switch Health Count",
        "Type" :  "histogram"
}
```


- **Without Filtering:**

```
curl --request GET --url https://xxxx.xxxx.xxxx.xxxx/nmx/v1/kpis
--user "<username>:<password>" --insecure
```


```
{
    "Health" :  {
        "compute-health" :  {
            "Data" :  [
                {
                    "UNKNOWN" :  0
                },
                {
                    "HEALTHY" :  9
                },
                {
                    "DEGRADED" :  0
                },
                {
                    "UNHEALTHY" :  0
                }
            ],
            "Description" :  "Number of compute nodes per health state",
            "Title" :  "Compute Health Count",
            "Type" :  "histogram"
        },
        "domain-health" :  {
            "Data" :  [
                {
```

```
                    "UNKNOWN" :  0
                },
                {
                    "HEALTHY" :  1
                },
                {
                    "DEGRADED" :  0
                },
                {
                    "UNHEALTHY" :  0
                }
            ],
            "Description" :  "Number of domains per health state",
            "Title" :  "Domain Health Count",
            "Type" :  "histogram"
        },
        "gpu-health" :  {
            "Data" :  [
                {
                    "DEGRADED" :  0
                },
                {
                    "NONVLINK" :  0
                },
                {
                    "UNKNOWN" :  0
                },
                {
                    "DEGRADED_BW" :  0
                },
                {
                    "HEALTHY" :  36
                }
            ],
            "Description" :  "Number of GPU instances per health state",
            "Title" :  "GPU Health Count",
```

```
            "Type": "histogram"
        },
        "switch-health": {
            "Data": [
                {
                    "MISSING_NVLINK": 0
                },
                {
                    "UNHEALTHY": 0
                },
                {
                    "UNKNOWN": 0
                },
                {
                    "HEALTHY": 18
                }
            ],
            "Description": "Number of switch instances per health state",
            "Title": "Switch Health Count",
            "Type": "histogram"
        }
    },
    "Inventory": {
        "cable-fw-version": {
            "Data": [
                {
                    "N/A": 1296
                }
            ],
            "Description": "Count the number of entries for each FWVersion.",
            "Title": "Cable FWVersion Count",
            "Type": "histogram"
        },
        "cable-pn": {
            "Data": [
                {
```

```
                    "N/A": 1296
                }
            ],
            "Description": "Count the number of entries for each PN.",
            "Title": "Cable PN Count",
            "Type": "histogram"
        },
        "cable-type": {
            "Data": [
                {
                    "850 nm VCSEL": 558
                }
            ],
            "Description": "Count the number of entries for each cable type.",
            "Title": "Cable Type Count",
            "Type": "histogram"
        },
        "chip-temperature": {
            "Data": [
                {
                    "30-40": 50
                }
            ],
            "Description": "Number of instances per temperature",
            "Title": "Chip Temperature Count",
            "Type": "histogram"
        },
        "compute-allocation": {
            "Data": [
                {
                    "FULL": 9
                },
                {
                    "PARTIAL": 0
                },
                {
```

```
                        "ALL": 9
                    },
                    {
                        "FREE": 0
                    }
                ],
                "Description": "Number of compute nodes with its GPU instances in allocation state",
                "Title": "Compute GPU Allocation Count",
                "Type": "histogram"
            },
            "connection-count": {
                "Data": [
                    {
                        "DISCOVERED": 0
                    },
                    {
                        "EXPECTED": 0
                    },
                    {
                        "EXPECTED_ACTIVE": 0
                    },
                    {
                        "EXPECTED_INACTIVE": 0
                    },
                    {
                        "UNEXPECTED": 0
                    }
                ],
                "Description": "Number of connection count per connection attribute",
                "Title": "Topology connection Count",
                "Type": "histogram"
            },
            "effective-ber": {
                "Data": [
                    {
```

```
                        "effective_ber" :  0,
                        "node_guid" :  "0xb0cf0e0300db1be0",
                        "port_guid" :  "0x00000002251f681b"
                    },
                    {
                        "effective_ber" :  0,
                        "node_guid" :  "0xb0cf0e0300db19a0",
                        "port_guid" :  "0x00000002251f681b"
                    },
                    {
                        "effective_ber" :  0,
                        "node_guid" :  "0xb0cf0e0300db1be0",
                        "port_guid" :  "0x00000002251f681b"
                    }
                ],
                "Description" :  "List of top 100 port that has the highest EFFECTIVE BER
readings",
                "Title" :  "Top 100 EFFECTIVE BER Ports",
                "Type" :  "histogram"
            },
            "link-down-frequency" :  {
                "Data" :  [
                    {
                        "SWITCH" :  0.00020169558757286254
                    }
                ],
                "Description" :  "Average time between link down events",
                "Title" :  "Link Down Frequency",
                "Type" :  "counter"
            },
            "link-down-rate" :  {
                "Data" :  [
                    {
                        "link_down_rate" :  16.616666666666685,
                        "node_guid" :  "0xb0cf0e0300dafa00",
                        "port_guid" :  "0x00000002251f681b"
```

```
            },
            {
                "link_down_rate": 16.58333333333335,
                "node_guid": "0xdf5abd57894e3a50",
                "port_guid": "0x00000002251f681b"
            },
            {
                "link_down_rate": 16.53333333333335,
                "node_guid": "0xdf5abd57894e3a50",
                "port_guid": "0x00000002251f681b"
            }
        ],
        "Description": "List of top 100 port that has the highest link down events",
        "Title": "Top 100 Link Down Ports",
        "Type": "histogram"
    },
    "link-up-count": {
        "Data": [
            {
                "current": 1296
            },
            {
                "min": 0
            },
            {
                "max": 1296
            }
        ],
        "Description": "Number of links in UP state out of total link number.",
        "Title": "Link UP Count",
        "Type": "histogram"
    },
    "port-count": {
        "Data": [
            {
                "GPU": 648
```

```
            },
            {
                "SWITCH_ACCESS" : 648
            },
            {
                "SWITCH_TRUNK" : 0
            },
            {
                "UNDEFINED" : 0
            }
        ],
        "Description" : "Number of ports per device type.",
        "Title" : "Port Type Count",
        "Type" : "histogram"
    },
    "raw-ber" : {
        "Data" : [
            {
                "node_guid" : "0xb0cf0e0300dafb60",
                "port_guid" : "0x00000002251f681b",
                "raw_ber" : 0
            },
            {
                "node_guid" : "0xb0cf0e0300db1be0",
                "port_guid" : "0x00000002251f681b",
                "raw_ber" : 0
            },
            {
                "node_guid" : "0xfdece0e67e59176f",
                "port_guid" : "0x00000002251f681b",
                "raw_ber" : 0
            }
        ],
        "Description" : "List of top 100 port that has the highest RAW BER readings",
        "Title" : "Top 100 RAW BER Ports",
        "Type" : "histogram"
```

```
        },
        "symbol-ber": {
            "Data": [
                {
                    "node_guid": "0x7e4c3b753098777c",
                    "port_guid": "0x00000002251f681b",
                    "symbol_ber": 0
                },
                {
                    "node_guid": "0xb0cf0e0300db1940",
                    "port_guid": "0x00000002251f681b",
                    "symbol_ber": 0
                },
                {
                    "node_guid": "0xb0cf0e0300dafa40",
                    "port_guid": "0x00000002251f681b",
                    "symbol_ber": 0
                }
            ],
            "Description": "List of top 100 port that has the highest SYMBOL BER readings",
            "Title": "Top 100 SYMBOL BER Ports",
            "Type": "histogram"
        }
    }
}
```


Note: This is a sample response.

Creation of Grafana Dashboard

- **Download Grafana**

https://grafana.com/grafana/download

- **Configure JSON API Data Source**

    1. Click **Connections,** then select **JSON API** from the available options. If it is not installed, install it first.

    2. Click **Add a new data source**.

    3. Provide the details below:

        - **Name:** Add the prefix **kpi** (case sensitive).

          Note: If you do not prefix the name with kpi, update the attached JSON file before importing it into the dashboard.

```
  "templating": {
    "list": [
      {
        "current": {},
        "hide": 1,
        "label": "Kpi Data Source",
        "name": "DS_KPI",
        "options": [],
        "query": "marcusolsson-json-datasource",
        "refresh": 1,
        "regex": "/^kpi/",
        "type": "datasource"
      }
```

        - **URL**. Ex- https://10.xxx.xx.xxx/nmx/v1/kpis

        - **Authentication:** Select **Basic Authentication**

          Example:

          - User: rw-user

          - Password: Nmx12345

        - Skip TLS Certificate Validation

    4. Save and Test.

- **Configuring Dashboard**

    1. Click DashBoards in home screen.

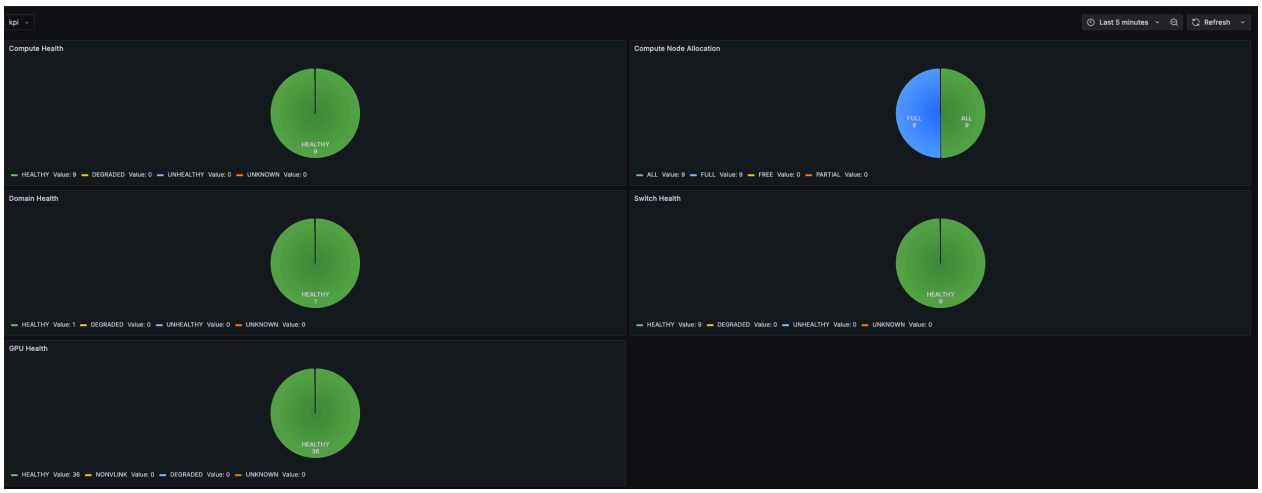
2. Click on New → import.

3. Paste the content of `kpi-dashboard.json` file under the section "Import via dashboard JSON model".

   kpi-dashboard.json

4. Click on Load, followed by Import.

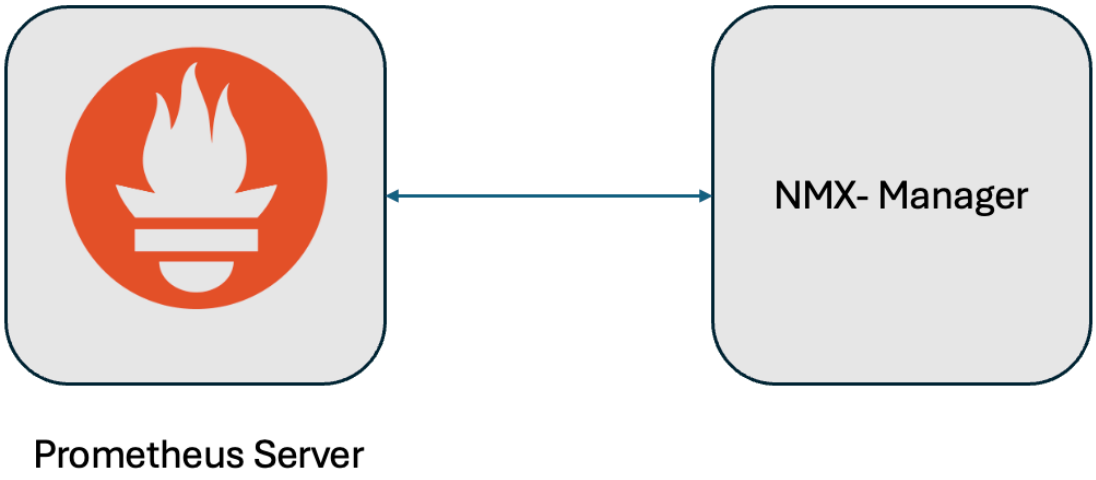
- **Sample Dashboard**



Prometheus Endpoint

The NMX-M will have the capability to expose a Prometheus metrics endpoint for telemetry scraping purposes. Meaning, the NMX-M will present the telemetry data in a format compatible with the Prometheus protocol, allowing for efficient data collection and monitoring. The telemetry data exposed via the Prometheus metrics endpoint will include various telemetry data collected from NMX-T.

The following diagram illustrates how Prometheus metrics can be scraped from the NMX-M.



To enable Prometheus to scrape metrics from the NMX-M, it must be configured as a target endpoint in the Prometheus server configuration. This setup is similar to configuring any other Prometheus endpoint.

To configure the NMX-M as a Prometheus endpoint, you need to add the NMX-M's endpoint details to the Prometheus.yml configuration file.

The below is an example configuration snippet:

```
global:
  scrape_interval: 15s
  evaluation_interval: 15s
scrape_configs:
  - job_name: "prometheus"
    scheme: https
    basic_auth:
      username: 'rw-user'
      password: 'rw-password'
    static_configs:
      - targets: ['10.xxx.xx.xx']
    metrics_path: '/nmx/v1/metrics'
    tls_config:
      insecure_skip_verify: true
    params:
      id: ['e83134ff-89fb-45eb-97ae-920b35f8fde5']
```

where:

job_name	Specifies the name of the scraping job.
id	NVlink5 domain id.

Alerts Management

General Description

NMX-M continuously collects metrics from Telemetry services. Based on predefined alert rules, certain metrics may trigger alerts, which are sent to relevant stakeholders.

Some alerts provide precise event descriptions, such as a port-down alert, while others indicate trends, such as port health degradation.

The triggering conditions are predefined within the system.

Supported Alert Rules

Port Status Warning

The **Port Status** alert is triggered by a group of events affecting port operation. These alerts are based on deviations in a group of metrics—if any metric exceeds two standard deviations (2STD) from the average, an alert is raised. The alert includes the port ID, domain, node ID, and port number, but not the exact metric that caused the alert.

Currently, there are two types of alerts in this group:

1. **Ongoing Port Validation**
    1. **Trigger Condition:** More than 3 errors of a specific metric are detected within a 24-hour window.
    2. **Behavior:** The alert is continuously sent until the condition is resolved. This ensures that external systems receive the alert even if temporarily down.
    3. **Alert Group:** `ongoing_port_validation`
    4. **Alert Name:** `OngoingPortValidation`
    5. **Severity:** `warning`
2. **Anomalies Detection**

On each new metric value, NMX-M analyzes historical data over a predefined time span using a statistical model. Alerts are triggered if values deviate significantly (exceeding 2STD). The following alerts are included:

-
  - **Port Congestion Warning**
    - **Alert Group:** `port_metrics_deviation`
    - **Alert Name:** `PortCongestion`
    - **Severity:** `warning`
  - **Physical Layer Retransmission Warning**
    - **Alert Group:** `port_metrics_deviation`
    - **Alert Name:** `PhysicalLayerRetransmission`
    - **Severity:** `warning`
  - **Port Degradation Warning**
    - **Alert Group:** `port_metrics_deviation`
    - **Alert Names:**
      - `PortDegradationHistogram1`
      - `PortDegradationHistogram2`
      - `PortDegradationHistogram3`
      - `PortDegradationBER`
      - `PortDegradationLinkErrors`
    - **Severity:** `warning`
  - **Packet Discard Warning**

- **Alert Group:** `port_metrics_deviation`
- **Alert Name:** `PacketDiscard`
- **Severity:** `warning`

Webhook Notifications

When an alert is triggered, NMX-M sends a notification to the configured webhook URL.

Setting the Webhook URL

You can set the webhook receiver URL during the installation process.

During installation, you will be prompted to provide one or more webhook URLs. You may enter a single URL or multiple URLs separated by commas. All specified URLs will receive the notifications simultaneously.

Example input format:

```
http://alert1.example.com:9093,http://alert2.example.com:9093/webhook
```

To skip this step and configure the URL later, simply press Enter when prompted.

This URL (or list of URLs) will be used to receive HTTP POST notifications in the Prometheus Alertmanager JSON format:

```
{
   "version": "4",
   "groupKey": <string>,                    // key identifying the group of alerts (e.g. to deduplicate)
   "truncatedAlerts": <int>,                // how many alerts have been truncated due to "max_alerts"
   "status": "<resolved|firing>",
   "receiver": <string>,
```

```
  "groupLabels": <object>,
  "commonLabels": <object>,
  "commonAnnotations": <object>,
  "externalURL": <string>,                  // backlink to the internal alert manager.
  "alerts": [
    {
      "status": "<resolved|firing>",
      "labels": <object>,
      "annotations": <object>,
      "startsAt": "<rfc3339>",
      "endsAt": "<rfc3339>",
      "generatorURL": <string>,             // identifies the entity that caused the alert
      "fingerprint": <string>               // fingerprint to identify the alert
    },
    ...
  ]
}
```

This format makes it compatible with a wide range of existing integrations. For more information, see Prometheus Integrations – Alertmanager Webhook Receiver.

```
{
  "receiver": "webhook-receiver",
  "status": "firing",
  "alerts": [
    {
      "status": "firing",
      "labels": {
        "alertgroup": "ongoing_port_validation",
        "alertname": "OngoingPortValidation",
        "severity": "warning"
      },
      "annotations": {
```

```
            "description": "Error counters have increased\nmultiple times in the last 24 hours for port 123.\n",
            "summary": "Repeated error counter increases for port 123"
          },
          "startsAt": "2025-02-18T13:25:16.860604419Z",
          "endsAt": "0001-01-01T00:00:00Z",
          "generatorURL": "http://c6c0e7e35a8f:8880/vmalert/alert?group_id=10369550901437612 74&alert_id=1074584496268461589",
          "fingerprint": "abcf32454420d0f2"
        }
    ],
    "groupLabels": {
        "alertname": "OngoingPortValidation"
    },
    "commonLabels": {
        "alertgroup": "ongoing_port_validation",
        "alertname": "OngoingPortValidation",
        "severity": "warning"
    },
    "commonAnnotations": {
        "description": "Error counters have increased\nmultiple times in the last 24 hours for port 123.\n",
        "summary": "Repeated error counter increases for port 123"
    },
    "externalURL": "http://224f3bffc652:9093",
    "version": "4",
    "groupKey": "{}:{alertname=\"OngoingPortValidation\"}",
    "truncatedAlerts": 0
}
```


Updating Webhook Receiver URL

You can update the webhook URL at any time after installation.

To do so:

1. Create a `webhook.yaml` file containing the new Alerts webhook receiver URL(s). You can provide a single URL or a comma-separated list. Example:

```
http://alert1.example.com:9093,http://alert2.example.com:9093/webhook
```

2. Run the following script under the **root** user:

```
/opt/nvidia/nmx/scripts/alerts-webhook-url-config.sh
```

3. When prompted, you'll see the following menu:

```
Choose an option:
1) Update webhook receiver URLs
2) Retrieve webhook receiver URLs and their statuses
3) Clear webhook receiver URLs
4) Exit
Enter your selection: ..
```

- To **view** the current configuration, select option 2.
- To **update** the webhook URLs, select option 1.
- To **clear** the URLs, select option 3.

New webhook URLs will begin receiving notifications shortly after the system automatically redeploys the relevant components.

Troubleshooting

> **Note**
>
> All troubleshooting commands must be run as the root user

To extract observability Tar process:

1. Connect to one of the machines via SSH.
2. Go to `/opt/nvidia/nmx/scripts/`
3. Run `extract-observability-data.sh.`
4. A new tar file (`otel-collector-data.tar)` will be created under `/tmp`
5. Download it to your local machine using `scp`
6. Send the dump file `otel-collector-data.tar` to the NVIDIA debug team.

To get The NMX-M version, run the following command from any node:

```
/opt/nvidia/nmx/scripts/version.sh
```

Tech-Support Files Retrieval

For instructions on retrieving technical support files, see the NVOS NVLink User Manual → **Monitoring and Diagnostics** → **Technical Support section**.

Overview

This feature enables the generation and retrieval of technical support packages.

It offers a standardized, secure, and auditable method to collect diagnostic data from network switches using the NMX-M REST API—supporting troubleshooting and collaboration with technical support teams.

The interface provides capabilities to:

- Initiate log collection from one or more switches
- Download collected support packages
- List and delete packages via the NMX-M API

This streamlines engagement with support teams and accelerates root-cause analysis.

User Workflow

1. **Initiate Log Collection**

   Use the API to request log collection from one or more switches. You may specify a `ProfileID` for switch authentication.

2. **Track Operation**

   Receive an `operationId` in response. Use it to monitor collection progress.

3. **List Available Packages**

   Retrieve a list of support packages with metadata (filename, size, creation date, switch address).

4. **Download Package**

   Download a specific support package for offline analysis.

5. **Delete Package (Optional)**

   Remove support packages that are no longer needed.

API Endpoints

Endpoint	Method	Description	Request Parameters / Body	Success Response	Notes
`/nmx/v1/support-packages`	GET	List all support packages	None	`200 OK`, JSON array	Includes metadata f each packa
`/nmx/v1/support-packages`	POST	Initiate log collection from switches	JSON with `Switches` and optional `ProfileID`	`202 Accepted`, `operationId`	Asynchronc response
`/nmx/v1/support-packages/{id}`	GET	Download a support package	Path param: `id`	`200 OK`, binary file	Returns `.tar.gz` archive
`/nmx/v1/support-packages/{id}`	DELETE	Delete a support package	Path param: `id`	`204 No Content`	Irreversibly deletes the file

Request/Response Examples

Initiate Log Collection

```
POST /nmx/v1/support-packages
Content-Type: application/json

{
```

```
  "ProfileID": "551137c2f9e1fac808a5f572", // optional
  "Switches": [
    { "Address": "10.0.0.1" },
    { "Address": "10.0.0.2" }
  ]
}
Response
{
  "operationId": "551137c2f9e1fac808a5f572"
}
```

List Support Packages

```
GET /nmx/v1/support-packages
Response
[
  {
    "fileID": "551137c2f9e1fac808a5f572",
    "filename": "support-logs-2025-05-08.tar.gz",
    "size": 1048576,
    "uploadDate": "2025-05-08T12:34:56Z",
    "switchAddress": "10.0.0.1"
  }
]
```

Download Support Package

```
GET /nmx/v1/support-packages/{id}
Response

Binary file (e.g., application/octet-stream, .tar.gz archive)
```

Delete Support Package

```
DELETE /nmx/v1/support-packages/{id}

Response: HTTP 204 No Content
```

Expected Behavior

- **Asynchronous Collection**: Log collection runs in the background and may take several minutes depending on switch and network conditions.
- **Availability**: Once completed, the package becomes available via the list endpoint and can be downloaded.
- **File Format**: Packages are `.tar.gz` archives containing logs and diagnostics.
- **Metadata**: Each package includes identifying metadata (e.g., switch address, size, timestamp).

Error Handling

Code	Meaning
400	Bad Request – invalid input
404	Not Found – package does not exist
409	Conflict – conflicting or duplicate request
500	Internal Server Error

Example Error Response

```
{
   "httpErrorCode": 400,
   "customErrorCode": "invalid-request",
   "description": "Switch address is missing."
```

```
}
```

Profiles and Operation Tracking

Switch Profiles (ProfileID)

- You may specify a `ProfileID` to authenticate switch access during log collection.
- If omitted, the system uses the configured default profile.

Per-Switch ProfileIDs

You can assign different profiles per switch:

```
{
  "Switches": [
    {
      "Address": "10.0.0.1",
      "ProfileID": "profileid-for-switch-1"
    },
    {
      "Address": "10.0.0.2"
      // This switch will use the default profile if ProfileID is omitted
    }
  ]
}
```

Operation Object and Progress Tracking

Asynchronous Operation

Upon initiating log collection, the API returns an `operationId`. Use this ID to query the status of the request.

Tracking Status

Refer to Operations. When complete, the operation object will include a `result` field referencing the generated support package or packages.

Partitions Management

In NMX-M, network partitions refer to logical groupings of GPUs that exist within the same network domain. Partitions can be created based on one of two member types:

- **GPU-ID-Based:** A list of GPU identifiers.
- **Location-Based:** A set of objects describing the physical location, including attributes such as domain, chassis, slot, and host.

The partition's member type is established at creation and cannot be changed later. All subsequent operations (updates or reads) should match the originally defined type. For example, a partition created using location-based members cannot be updated using GPU IDs; such an attempt will result in a **409 Conflict** error.

API Endpoints

All asynchronous requests (Create, Update, and Delete) return a **202 Accepted** response immediately. This response includes:

- A JSON body:

```
{
   "operationId":  "551137c2f9e1fac808a5f572"
}
```

- A **Location** header pointing to the operations endpoint:

```
/nmx/v1/operations/<op-id>
```

For resource creation, once the operation completes, the new partition can be retrieved by following the **Location** header or using a **GET** request at:

```
/nmx/v1/partitions/<p-id>
```

Both **GET** and **GET-all** endpoints return a **200 OK** response with the partition details.

Partition Management Endpoints

Partition Action	Endpoint	Description
Create	`POST /nmx/v1/partitions`	Initiates the creation of a new network partition. The request body must include a partition name and a members object, which is either GPU-ID-based or location-based (mixing types is not supported).
Update	`PUT /nmx/v1/partitions/<p-id>`	Updates an existing partition. Note that currently the partition name cannot be changed. Instead, it is possible to update the list of members. When using **PUT**, the `members` parameter lists all GPUs that should be part of the partition. The system will determine which members to add or remove to match the desired configuration. **Note** The member type in the update must match the partition's original type. A mismatch will result in a **409 Conflict** error.
Delete	`DELETE /nmx/v1/partitions/<p-id>`	Deletes an existing partition. The deletion process returns an operation and follows the same asynchronous pattern as creation and update.

Partition Action	Endpoint	Description
Retrieve	`GET /nmx/v1/partitions/<p-id>`	Retrieves details of a specific partition, including its health and other metada.
	`GET /nmx/v1/partitions`	Retrieves a list of all partitions.

Partition Lifecycle and Types

Lifecycle Stage	Description
Creation	When a partition is created via a **POST** request, the API determines its type based on the provided members. The assigned type is **permanent**.
Update	Subsequent updates must maintain the same member type. Attempts to use a different member type (e.g., using GPU IDs on a location-based partition) will be rejected with a **409 Conflict** error.
Deletion	Deletion of a partition is performed asynchronously and returns an operation. The partition is removed once the operation completes.
Retrieval	After an operation completes (e.g., partition creation), the partition can be accessed by following the Location header in the operation response or by using the partition ID in a GET request.
Health	Each partition includes a health metric that reflects the status of the underlying managed network.

Operational Considerations

Asynchronous Processing

- All operations (create, update, delete) return a **202 Accepted** response with an `operationId` and a Location header pointing to `/nmx/v1/operations/<op-id>`.
- Operations continue until they reach a terminal state—`failed`, `completed`, or `cancelled`.

Cancellation

- Operations can be canceled **only if** they have not started processing and no critical actions have been taken.
- The cancellation process generates its own non-cancellable operation.

Constraints

Member Exclusivity

- A GPU **cannot** belong to more than one partition.

Domain Consistency

- All GPUs in a partition must belong to the same **network domain**.
- Mixing GPUs from different domains is not possible.

Operation Results

- The asynchronous operations created during partition management are used to track progress.
- The operation should complete before acting on the partition.
- The response includes a **Location** header pointing to the newly created resource.

Error Handling

The following error codes may be returned by the Partitions API:

Error Code	Description
400 Bad Request	The request is malformed or contains an improperly structured ID.
404 Not Found	The specified partition ID does not exist.
409 Conflict	Request contains unsupported arguments, such as: • Updating with an incorrect member type. • Mixing member types during creation. • Assigning a GPU to multiple partitions. • Including GPUs from different domains. • Deleting or canceling a partition in an invalid state.
500 Internal Server Error	Unexpected server error.

Partition Health

The health of a partition can be monitored using the health state maintained within the partition. Depending on the resiliency mode specified for the partition, the partition can get into various states that are listed below:

- Full Bandwidth Mode:
    - HEALTHY: When the partition is marked as healthy, it is expected to be in full-bandwidth and full compute capacity state. This state is considered to be optimal.
    - DEGRADED: In this state, some of the GPUs could be marked as “parked” and their GPU health could be NO_NVLINK. In this state, the rest of the GPUs will be able to communicate with full bandwidth. This state is considered operational.
    - UNHEALTHY: There could be various reasons that cause a partition to go into an unhealthy state. There could be a loss of a switch or other internal failures that result in this state. This state is **not** considered operational.
- Adaptive Bandwidth Mode:
    - HEALTHY: When the partition is marked as healthy, it is expected to be in full-bandwidth and full compute capacity state. This state is considered to be optimal.

- BANDWIDTH: In this state, some of the trunk links might be missing. However, all of the GPUs will be able to communicate with one-another in a degraded bandwidth capacity. This state is considered to be operational.

- UNHEALTHY: There could be various reasons that cause a partition to go into an unhealthy state. There could be a loss of a switch or other internal failures that result in this state. This state is **not** considered operational.

- User Action Required Mode:

  - HEALTHY: When the partition is marked as healthy, it is expected to be in full-bandwidth and full compute capacity state. This state is considered to be optimal.

  - DEGRADED_BANDWIDTH: In this state, some of the trunk links might be missing. However, all of the GPUs will be able to communicate with one-another in a degraded bandwidth capacity. This state is considered to be operational.

  - UNHEALTHY: There could be various reasons that cause a partition to go into an unhealthy (and non-operational) state:

    - There could be a loss of a switch or other internal failures that result in this state.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright 2025. PDF Generated on 09/18/2025