NMX Manager (NMX-M) Documentation v85.1.0009: PS Level

Table of contents

Introduction

NMX Manager is part of the NMX solution aimed at collecting and processing data center telemetry, monitoring and providing insights and predictions on the operability and health of its systems. The NMX Manager part is to aggregate the streamed telemetry from NMX Telemetry subsystem and make it available to any external entity via Prometheus endpoint.

Further information on this product can be found in the following sections:

- Release Notes
- User Manual

Software Download

Log into the NVIDIA Licensing Portal —> Entitlements —> Networking —> Product Family —> NMX

Document Revision History

For the list of changes made to the User Manual, refer to User Manual Revision History.

For the list of changes made to the Release Notes, refer to Release Notes History.

Release Notes

Release Notes History

Revision	Date	Description
85.1.0009-PS	January 14, 2025	Initial release of this Release N

Revision	Date	Description
		otesversion.

Release Notes contain the following sections:

- Changes and new Features
- General Support
- Bug Fixes in this Version
- Known Issues

Changes and new Features

Feature/ Change	Description
Rev. 85.01.0009	
Security Hardening	This release contains important reliability improvements and security hardening enhancements. NVIDIA recommends upgrading your system to this release to improve the devices' security and reliability.

Feature/ Change	Description
Enhance ments	

General Support

The following are the tested and supported NMX services:

- NMX-C / NMX-T - v0.9

Bug Fixes in this Version

This version does not contain any bug fixes.

Known Issues

RM #	Issue
4225157	**Description:** When partition IDs are duplicate across domain UUIDs, GPU assignment is done incorrectly.
	Workaround: Make sure all partition IDs are unique across all domain UUIDs to ensure a single correct GPU-to-partition match.
	Keywords: Partitions Manager
	Discovered in Version: 85.1.0004
3867360	**Description:** Prometheus endpoint does not clean the metrics when the telemetry traffic stops.
	Workaround: N/A
	Keywords: Prometheus endpoint
	Discovered in Version: 85.1.0004

User Manual

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.

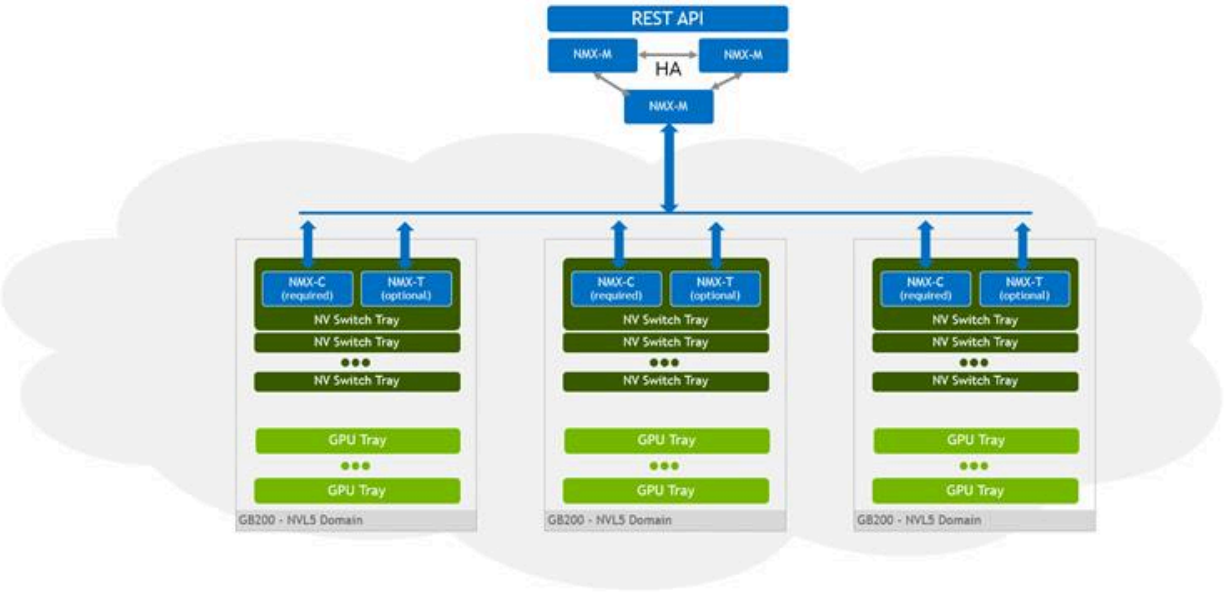
The primary functions of NMX Manager are:

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.

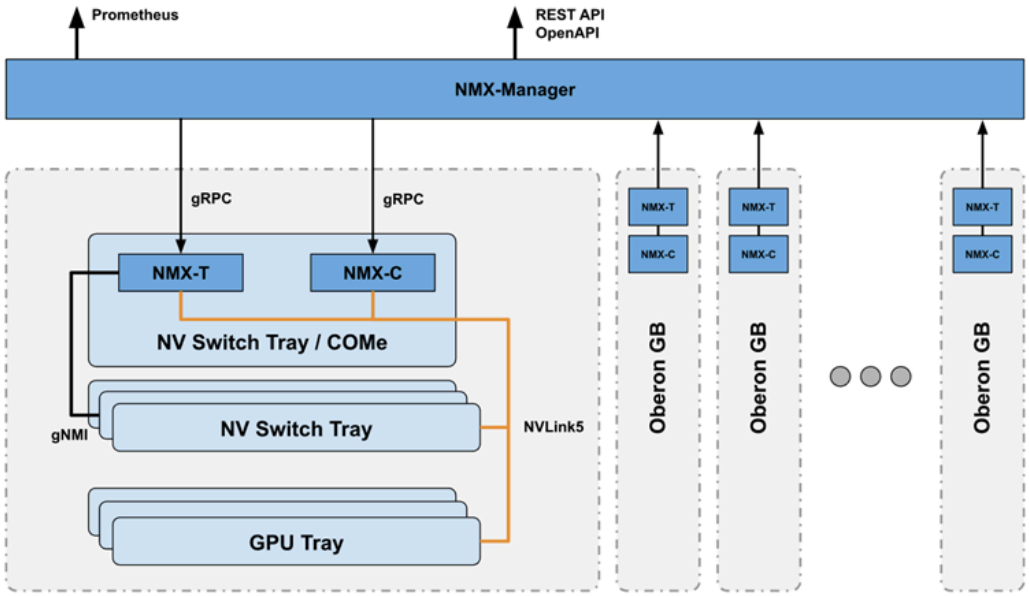
NMX Manager is built on an event-driven microservice architecture, enabling seamless communication between microservices through Apache Kafka, which serves as the event

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.

Hardware and Software Requirements

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

Resource	Requirements
Processor	48 Cores
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

Installation Procedure

NMX-M installation is now simplified using a one-click shell script that ensures full air-gapped support. The below are detailed instructions of how to seamlessly install NMX-M on your dedicated machines.

The NMX-M package is distributed as a single tarball, `NMX-MGR-<VERSION>-VM.tar.gz`. Within this tarball, there are several additional tarballs organized into various categories:

- Installation
- Infra
- Services
- RKE2

Installing the VM Image

To install the VM image, follow the steps below:

1. Extract NMX-MGR-<VERSION>-VM.tar.gz
2. 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 512000-n ubuntu1 --os-type linux --os-variant generic --vcpus=48 --disk path=/var/lib/libvirt/images/ubuntu-24.04-nmx-base.qcow2,format=qcow2,device=disk,bus=virtio --vnc --noautoconsole --import --network=bridge=br0,model=virtio,mac=<MAC> --check all=off
   ```

3. Login as root on the machine.

Installing the NMX Manager (NMX-M)

To install NMX Manager (NMX-M) software package, follow the steps below:

1. Copy the `NMX-MGR-M-<VERSION>.tar.gz ` file to one of the VMs. Once the tarball is in place, execute the following command to extract its contents.

   ```
   tar -xzf NMX-MGR-M-*.tar.gz
   ```

2. Navigate to the created `NMX-M` directory. Run the installation script using the following command.

   ```
   cd NMX-M
   ./install.sh
   ```

3. Install the required tools. The installation script will begin by installing all the necessary tools to support the installation. This includes:

- Ansible
- kubectl
- zarf
- helm

4. Configure Kubernetes Cluster. You will be required to specify the number of server Virtual Machines (VMs) for the Kubernetes cluster (NODES) and provide their respective IP addresses. Please note that a minimum of three VMs is needed.

   Additionally, during the setup process, you will be required to enter the SSH password twice:

   - Once for the local user

   - Once for the root user (same as local user)

5. Install RKE2. The script will proceed by running the RKE2 (Rancher Kubernetes Engine 2) installation.

6. Provide Client Certificates for mTLS. When prompted, provide the client certificates or create temporary certificates to support southbound mTLS (mutual Transport Layer Security).

   1. **Use existing certificates**: Place the required certificate files (client.cert, client.key, and rootCA.crt)

      1. Log in with an additional shell to install the machine and place the certificates in the folder – NMX-M/Installation/Ansible with the required names.

      2. If you do not yet have the certificates, proceed with the installation using option 2. Once you have obtained the certificates, rerun the installation.

   2. **Create temporary certificates**: If the client has not provided the necessary certificates yet or for testing reasons, you can generate temporary self-signed certificates as a stopgap measure.

7. Set the API user passwords. You will be asked twice to set passwords for the API users `` `ro_user ` `` and `` `rw_user` ``. Each password must be at least 8 characters long and include a mix of letters and numbers.

8. The script installs the infrastructure components.

9. The script installs the NMX-M Microservices.

> **Note**
>
> The installation script is designed to continue on failure, allowing you to resume the installation at any part if necessary.

Mutual TLS (mTLS) Configuration

Mutual TLS (mTLS) , is a security protocol that ensures both the client and server in a network communication authenticate each other using certificates before establishing a connection. This is an enhancement over standard Transport Layer Security (TLS), where only the server is authenticated by the client.

> **Note**
>
> In this release, mTLS is enabled by default in NMX-M. NMX-M communicates with both NMX-T and NMX-C using mTLS over gRPC, therefore, please make sure that mTLS is also configured on both NMX-T and NMX-C.

The procedures below must be performed after the NMX-M deployment.

Installing Certificates

> **Note**
>
> Make sure to generate and provide the location of the certificates when scripts prompt for it before running the following script.

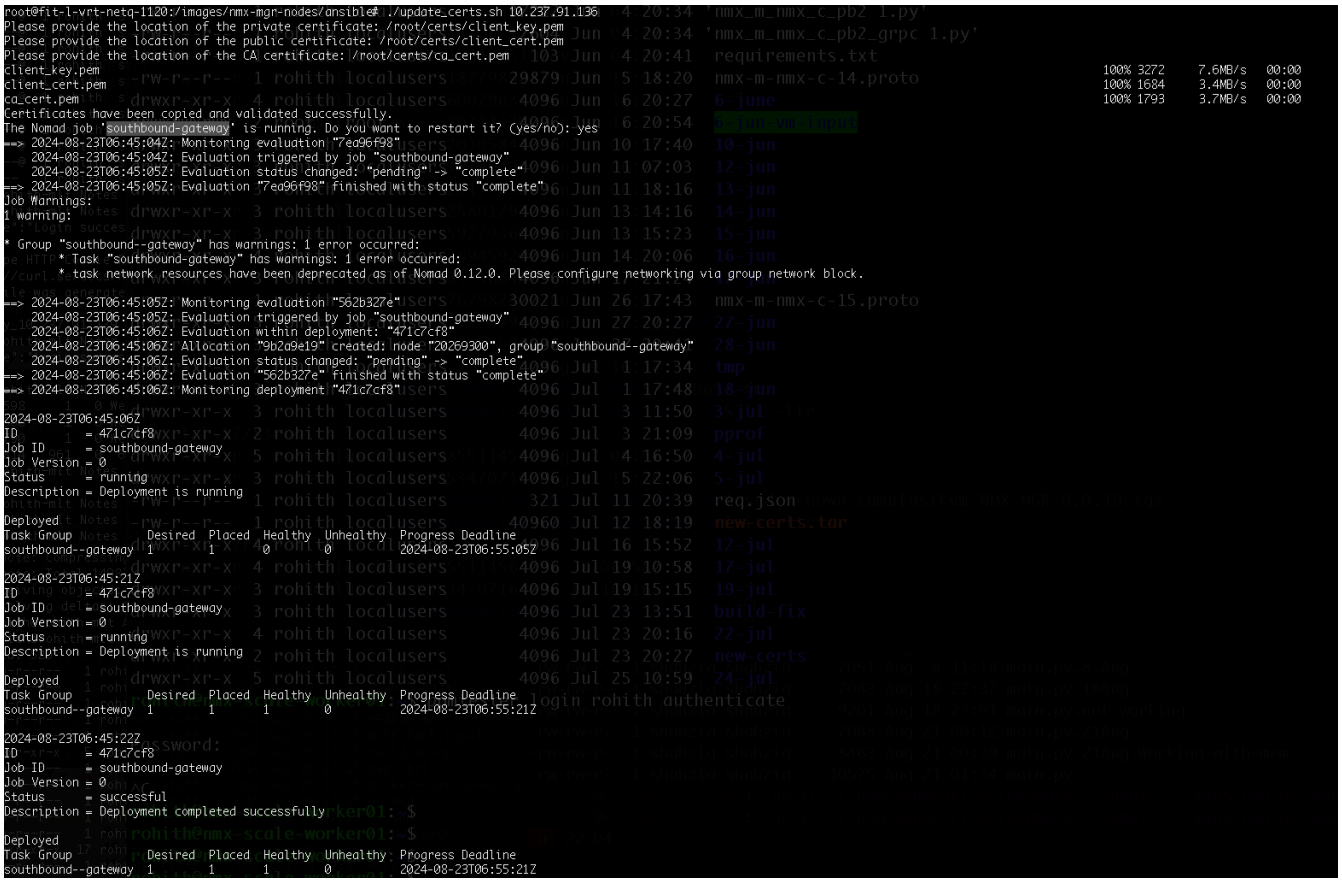
> **Note**
>
> Ensure that the certificates used by NMX-M, NMX-T, and NMX-C are either issued by the same CA or signed by a CA that is trusted and approved.

1. From the Ansible folder run the following command.

```
#update_certs.sh <ip of the VM>
```

2. Select 'yes' when prompted to reboot the southbound-gateway.

The below is an output sample. Please ensure to provide the certificate path specific to your environment.



Skipping SAN Validation

> **Note**
>
> This step is required only if authentication is failing due to a SAN header check.

1. To bypass the SAN header check, use the following script to update the mTLS configuration and skip SAN validation.

   ```
   #update_mtls_config.sh <ip of the VM>
   ```

2. Select 'enable' when prompted to enable mTLS.
3. Select 'yes' when prompted to skip SAN validation in mTLS configuration.
4. Select 'yes' when prompted to reboot the southbound-gateway.

The below is an output sample.

```
root@fit-l-vrt-netq-1120:/images/nmx-mgr-nodes/ansible# ./update_mtls_config.sh 10.237.91.136
Do you want to enable mTLS? (enable/disable): enable

 System information as of Fri Aug 23 06:57:46 UTC 2024

  System load:           0.25
  Usage of /:            0.3% of 2.91TB
  Memory usage:          0%
  Swap usage:            0%
  Processes:             440
  Users logged in:       0
  IPv4 address for eth0: 10.237.91.136
  IPv6 address for eth0: fdfd:fdfd:10:237:250:56ff:fe30:5b88
Set 'enabled: true' under 'tls:' in /home/nvidia/endpoint-gateway/prometheus/production/southbound.yaml
Do you want to skip SAN validation in mTLS config? (yes/no): yes

 System information as of Fri Aug 23 06:57:46 UTC 2024

  System load:           0.25
  Usage of /:            0.3% of 2.91TB
  Memory usage:          0%
  Swap usage:            0%
  Processes:             440
  Users logged in:       0
  IPv4 address for eth0: 10.237.91.136
  IPv6 address for eth0: fdfd:fdfd:10:237:250:56ff:fe30:5b88
SAN Validation already Disabled
The Nomad job 'southbound-gateway' is running. Do you want to restart it? (yes/no): yes
==> 2024-08-23T06:58:02Z: Monitoring evaluation "21e4776f"
    2024-08-23T06:58:02Z: Evaluation triggered by job "southbound-gateway"
    2024-08-23T06:58:02Z: Evaluation status changed: "pending" -> "complete"
==> 2024-08-23T06:58:02Z: Evaluation "21e4776f" finished with status "complete"
Job Warnings:
1 warning:

* Group "southbound--gateway" has warnings: 1 error occurred:
        * Task "southbound-gateway" has warnings: 1 error occurred:
        * task network resources have been deprecated as of Nomad 0.12.0. Please configure networking via group network block.

==> 2024-08-23T06:58:02Z: Monitoring evaluation "5041332b"
    2024-08-23T06:58:02Z: Evaluation triggered by job "southbound-gateway"
    2024-08-23T06:58:03Z: Evaluation within deployment: "16e498c8"
    2024-08-23T06:58:03Z: Allocation "12f303e6" created: node "20269300", group "southbound--gateway"
    2024-08-23T06:58:03Z: Evaluation status changed: "pending" -> "complete"
==> 2024-08-23T06:58:03Z: Evaluation "5041332b" finished with status "complete"
==> 2024-08-23T06:58:03Z: Monitoring deployment "16e498c8"

2024-08-23T06:58:03Z
ID          = 16e498c8
Job ID      = southbound-gateway
Job Version = 0
Status      = running
Description = Deployment is running

Deployed
Task Group           Desired  Placed  Healthy  Unhealthy  Progress Deadline
southbound--gateway  1        1       0        0          2024-08-23T07:08:02Z

2024-08-23T06:58:18Z
ID          = 16e498c8
Job ID      = southbound-gateway
Job Version = 0
Status      = running
Description = Deployment is running

Deployed
Task Group           Desired  Placed  Healthy  Unhealthy  Progress Deadline
southbound--gateway  1        1       1        0          2024-08-23T07:08:17Z

2024-08-23T06:58:20Z
ID          = 16e498c8
Job ID      = southbound-gateway
Job Version = 0
Status      = successful
Description = Deployment completed successfully

Deployed
Task Group           Desired  Placed  Healthy  Unhealthy  Progress Deadline
southbound--gateway  1        1       1        0          2024-08-23T07:08:17Z
```

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:

1. ro-user
2. 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
Metric	Get Metric	GET	/nmx/v1/metric
Services	List NMX Services	GET	/nmx/v1/services
	Get NMX Service	GET	/nmx/v1/services/{id}

Group	Operation	Method	URL
Compute Nodes	List Compute Nodes	GET	/nmx/v1/compute-nodes
	Get Compute Nodes Count	GET	/nmx/v1/compute-nodes/cou

Group	Operation	Method	URL
			nt
	Get Compute Node	GET	/nmx/v1/compute-node/{id}
Switch Nodes	List Switch Nodes	GET	/nmx/v1/switch-nodes
	Get Switch Nodes Count	GET	/nm

Group	Operation	Method	URL
			x/v1/switch-nodes/count
	Get Switch Node	GET	/nmx/v1/switch-node/{id}
Switches	List Switches	GET	/nmx/

Group	Operation	Method	URL
			v1/switches
	Get Switches Count	GET	/nmx/v1/switches/count
	Get Switch	GET	/nmx/v1/switch/{id}

Group	Operation	Method	URL
Chassis	List Chassis	GET	/nmx/v1/chassis
	Get Chassis Count	GET	/nmx/v1/chassis/count
	Get Chassis	GET	/nmx/v1/chassis/{i

Group	Operation	Method	URL
			d }
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	/nm

Group	Operation	Method	URL
			x/v1/gpu
	Get GPU Count	GET	/nmx/v1/gpu/count
	Get GPU	GET	/nmx/v1/gpu/{id}
Operations	List Operations	GET	/nmx/v1/operati

Group	Operation	Method	URL
			ons
	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

Group	Operation	Method	URL
			/services
	Delete NMX Service	DELETE	/nmx/v1/services/{id}
Compute Nodes	Update Compute Node	PUT	/nmx/v1/compute-node/{i

Group	Operation	Method	URL
			d}
Switch Nodes	Update Switch Node	PUT	/nmx/v1/switch-node/{id}
Switches	Update Switch	PUT	/nmx/v1/switch/{id}
Chassis	Update Chassis	PUT	/nmx/v1/ch

Group	Operation	Method	URL
			assis/{id}
GPU	Update GPU	PUT	/nmx/v1/gpu/{id}
Operations	Cancel Operation	DELETE	/nmx/v1/operations/{id}

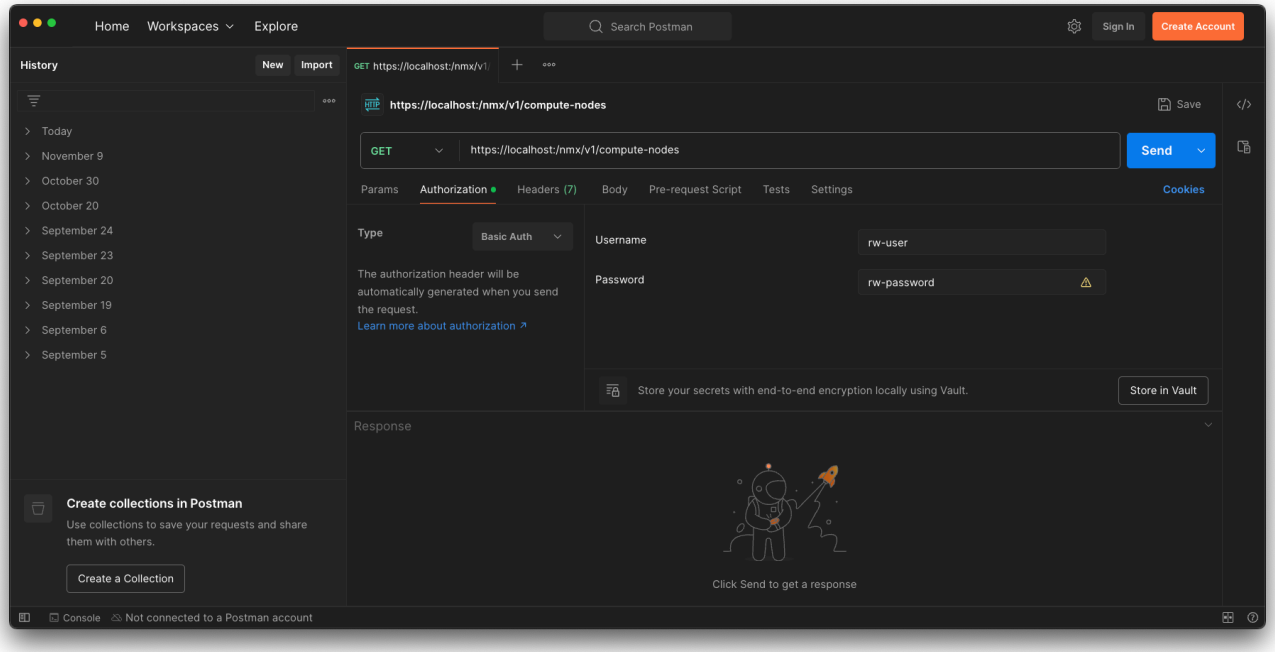
Examples

Read Only Endpoint

Postman

Authorization settings tab:

1. Select type: Basic Auth
2. Username: ro-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.

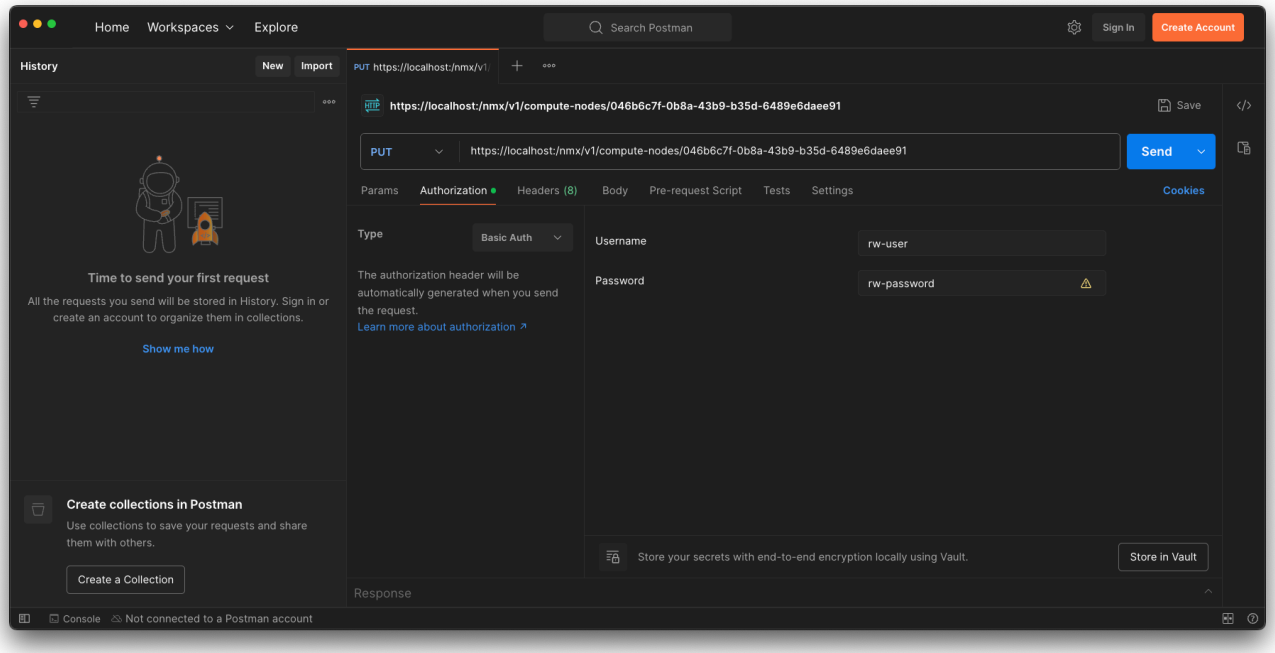
```
curl -X 'GET' \
  'https://<ip_address>/nmx/v1/compute-nodes' \
  -u ro-user:ro-password
```

Read Write Endpoint

Postman

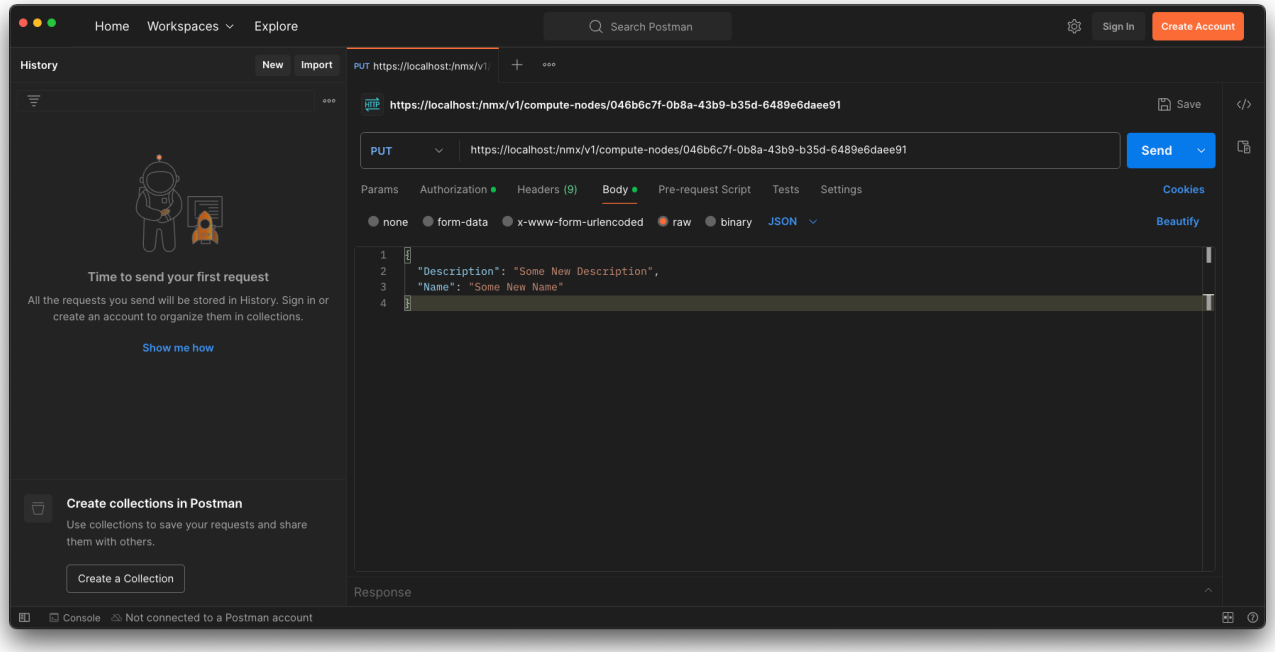
Authorization settings tab:

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

Once the installation is completed, register to the NMX Telemetry Service and NMX Controller Service. See section "Add an NMX Service", for directions and examples on how to register services via the NMX-M API.

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.

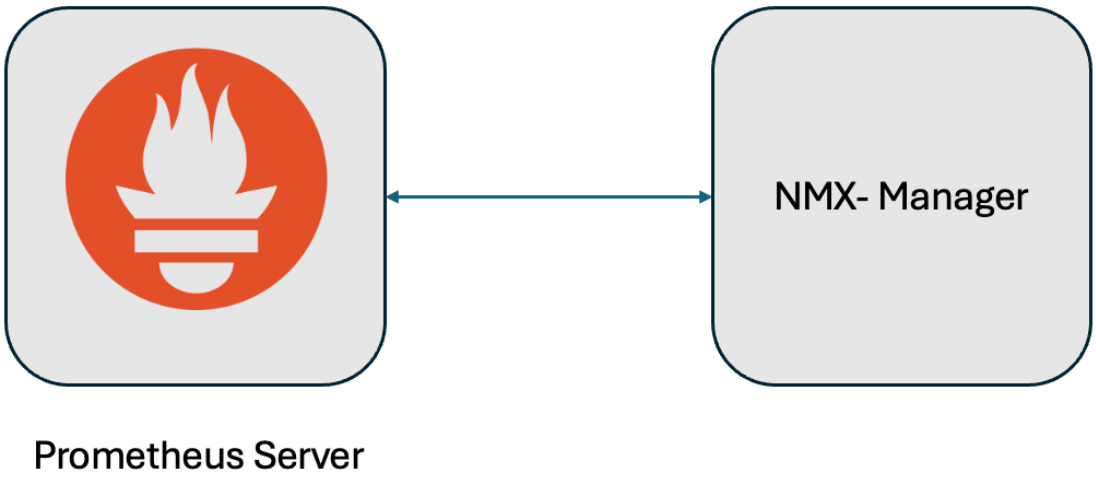
  This is a browser and JavaScript limitation. Use other tools like Insomnia, Postman or Terminal to get this scale of data.

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

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/metric'
```

```
    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.

Troubleshooting

To extract observability Tar process:

1. Connect to one of the machines via SSH.
2. Go to `/home/nvidia/scripts/.`
3. Run `extract-observability-data.sh.`
4. Send the dump file collected using `scp otel-collector-data.tar` to NVIDIA debug team.

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

Partition Action	Endpoint	Description
		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.
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.

Error Code	Description
	• 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.

Definitions/Abbreviation

Definitions / Abbreviation	Description
NMX	Product suite name for HPC cluster network monitoring and management system comprises NMX Telemetry, NMX Manager, NMX Controller and NMX Oasis.
NMX Telemetry (NMX-T)	NMX subsystem is an integrated solution constructed from multiple services, responsible for the collection, aggregation and transmission of telemetry data collected from various devices, applications and platforms that build the data center.
NMX Manager Connector	Part of the NMX Telemetry, responsible for the authentication, the connection handling to the Southbound Gateway and the telemetry streaming.
NMX Controller (NMX-C)	NMX subsystem is a control plane entity that is responsible for the configuration, monitoring and control of various systems, mainly network devices, that build the Data Center.
NMX Manager (NMX-M)	NMX subsystem that collect telemetry data from NMX Telemetry, can aggregate, analyze, run ML models for inference and pattern detections. NMX Manager can control the behavior of the HPC by changing the configuration of network or compute entities using NMX Controller.
NMX Manager Gateway	Part of NMX Manager, responsible for accepting connections from NMX Manager Connectors from multiple locations, handles the authorization and telemetry streams.
NMX Inference Engine	An AI pipeline receiving streaming telemetry from Kafka topic, runs AI models and returns predictions / actions.
NMX Controller Engine	Part of NMX Manager. Receives abstract action and sends domain specific actions, mainly targeted to NMX Controller to update network configurations.

Definitions / Abbreviation	Description
NMX Oasis Connector	Part of the NMX Manager SaaS. Its role is to connect to NMX Oasis data lake, while complying to data lake communication security, and stream the telemetry and logs data from NMX Manager to the data lake.
NMX Oasis	NMX subsystem is a data lake solution that lives in single/multiple clouds. Its components are API gateways, ETL processes, compute clusters, analysis models and informative dashboards.
Telemetry Agent	An entity that provides the telemetry data to a collector using pull/push methods. An agent is located on a target device, such as a network device, a host or a sensor controller. The agent uses various protocols to provide the telemetry, such as: gNMI, HTTP REST, IB MAD, etc. Agents can also support traps as means of event notifications.
Telemetry Collector	An entity that connects-to or is being-connected-from an agent to collect telemetry data. A collector uses a single protocol to connect to an agent. Collecting telemetry from agents using different protocols requires multiple collectors. Data collected can be metrics, events and logs.
Telemetry Aggregator	An entity that collects telemetry data from various collectors or other aggregators, runs data filtering, transformation and aggregation and provides means of storing the data (temporarily or long term). As mentioned above, aggregators can have a tree structure where an aggregator can connect to multiple aggregators as a method of collecting and transforming various data from various sources.
Circuit Management	The process of connecting a client to a server, monitoring the operational state of the connection, detection of a failed connection and managing the process of connection re-establishment.
NVOS	NVIDIA Networking OS, formerly known as MLNX-OS. NVOS is used as the Switch OS for L1 NVSwitch Trays and L2 NVSwitches
gRPC	Google's Remote Procedure Call (RPC) framework that can run in any environment

Documentation History

Release Notes History

Changes and New Feature History

Feature/ Change	Description
Rev. 85.01.0008	
NMX-M Deployment (including HA, Orchestration)	NMX-M is now deployed on top of Kubernetes, enabling advanced High Availability (HA) and service orchestration.
Partition Management	Added support for additional partitions: • Get list of partitions • Add new partition • Update partition • Remove partition
Observability	A log aggregation system has been implemented, allowing for the collection, aggregation, querying, and alerting of logs from NMX-M, NMX-T, NMX-C services, and monitored systems.
User Management	API user authentication has been added: 'ro-user' and 'rw-user' are now used for secure interactions with the NMX-M API.

Feature/ Change	Description
Air Gap Support - Cluster Mode	Added support for deployment in environments with strict network isolation, allowing organizations to install and use NMX-M without the need for an internet connection.
Bug Fixes	See Bug Fixes.

Feature/ Change	Description
Rev. 85.01.0006	
REST Interface	Added additional API endpoints for listing inventory such as Compute Notes, Switch Nodes, Ports and additional operations. For further information, see section *REST Interface* in the User Manual.
Mutual TLS (mTLS)	Mutual TLS (mTLS), is a security protocol that ensures both the client and server in a network communication authenticate each other using certificates before establishing a connection. This is an enhancement over standard Transport Layer Security (TLS), where only the server is authenticated by the client. For further information, see section *Mutual TLS (mTLS) Configuration* in the User Manual.
IPv6	**[Alpha]** Added support for IPv6 protocol.
HTTPS/Authentication Mechanism	In NMX-M, security and user authentication are critical components, achieved through the use of HTTPS and a PAM-based 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. For further information, see section *HTTPS/Authentication Mechanism* in the User Manual.

Feature/Change	Description
Rev. 85.01.0004	

Feature/Change	Description
General	NMX-M v85.01.0004 is at Engineering Sample level and subject to changes.
Service Management: REST Interface	The REST API Provides endpoints for Service (NMX-T): List, View, Create, Delete NMX services. Each endpoint has its own API, payload, responses, and schema. For more detailed information about our API, including endpoint descriptions, parameters, and examples, please refer to the Swagger UI. For further information, see section *REST Interface* in the User Manual
Prometheus Endpoint	The NMX-M exposes Prometheus endpoint for telemetry scraping purposes. For further information, see section *Prometheus Endpoint* in the User Manual

Unsupported Functionalities/Features

The following are the unsupported features for MNX-M v85.1.0004:

- IPv6

Bug Fixes History

Internal Ref.	Issue
4176808	**Description:** Fixed an issue that resulted in the switch-node entity missing the `SwitchIDList`, a list of switches relevant to that Switch Node.
	Keywords: switch-node, REST-API
	Discovered in Version: 85.1.0006
	Fixed in Release: 85.1.0008
4176806	**Description:** Fixed an issue that resulted in the switch entity missing the `PortIDList`, a list of ports relevant to that switch. The switch entity field `SwitchNodeID` was removed.
	Keywords: switch, REST-API
	Discovered in Version: 85.1.0006

Internal Ref.	Issue
	Fixed in Release: 85.1.0008
4176805	**Description:** Fixed an issue that resulted in the chassis entity missing the `ComputeNodeIDList`, a list of compute-nodes relevant to that chassis and the `SwitchNodeIDList`, a list of switch-nodes relevant to that chassis.
	Keywords: chassis, compute-node, switch-node, REST-API
	Discovered in Version: 85.1.0006
	Fixed in Release: 85.1.0008
4176803	**Description:** Fixed an issue that resulted in the GPU entity missing the `PortIDList`, a list of ports relevant to that GPU. The GPU entity field `HostName` was removed.
	Keywords: GPU, REST-API
	Discovered in Version: 85.1.0006
	Fixed in Release: 85.1.0008
4176540	**Description:** Fixed an issue that prevented the Domain UUID filter from being applied to the services in progress.
	Keywords: services, REST-API
	Discovered in Version: 85.1.0006
	Fixed in Release: 85.1.0008

3963239	**Description:** UpSince remains unchanged when the service transitions from Down to Up.
	Keywords: Service registration
	Discovered in Version: 85.1.0004
	Fixed in Release: 85.1.0006
3936942	**Description:** After deleting a service, the status might not be displayed as DRAINING.
	Keywords: Service registration
	Discovered in Version: 85.1.0004
	Fixed in Release: 85.1.0006

User Manual Revision History

Revision	Section	Description
v85.1.0009 PS	Partition Health	Moved under Partitions Management section.
	REST Interface	Updated section by replacing the REST API information with a reference to the documentation via the Swagger interface.
	Partitions Management	New section
v85.1.0008 PS	Installation Procedure	Updated the section: The installation process uses a one-click shell script that ensures full air-gapped support.
	NMX Services Registration	New section
	Mutual TLS (mTLS) Configuration	Removed section
	REST Interface	Updated section, added additional API endpoints for listing inventory such as Partitions and Metrics.
	Partition Health	New section
v85.1.0006 QS	Mutual TLS (mTLS) Configuration	New section
	HTTPS/Authentication Mechanism	New section
	REST Interface	Updated section, added additional API endpoints for listing inventory such as Compute Notes, Switch Nodes, Ports and additional operations.
85.1.0004 ES	All	Initial version of this document.