NVIDIA System Management User Guide

Release 25.03

NVIDIA Corporation

Jul 01, 2025

Contents

Chapter 1. Introduction

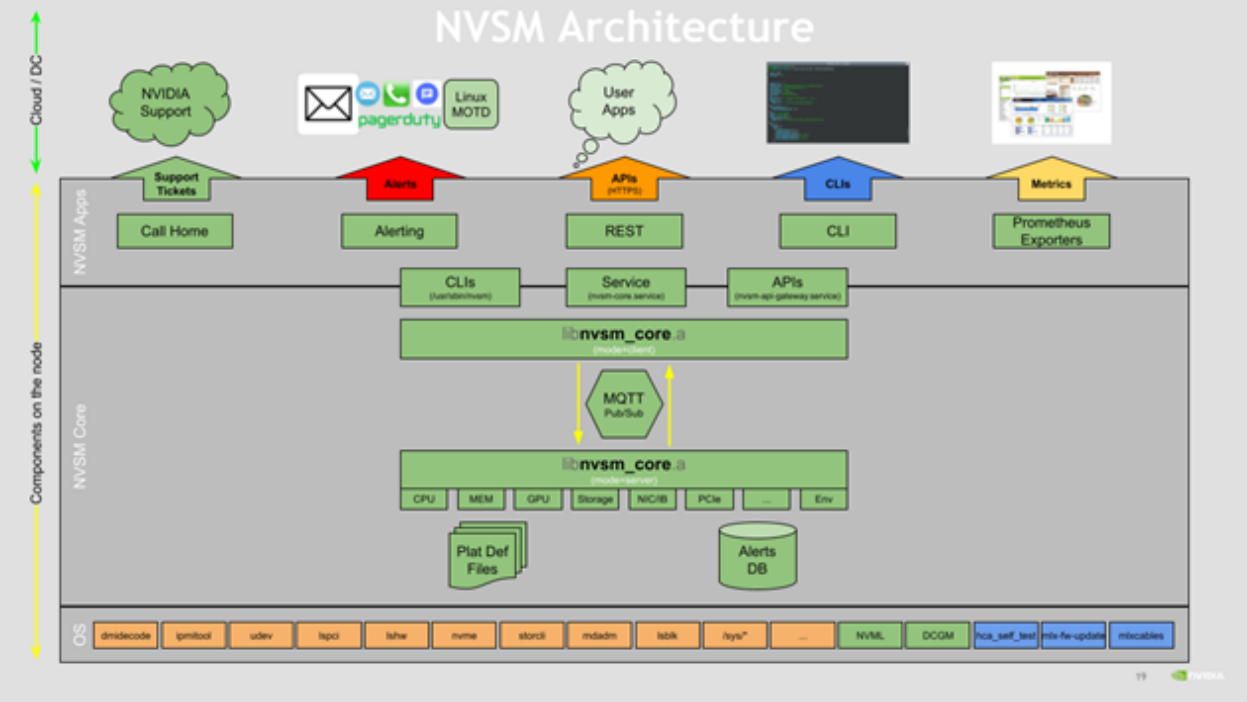
NVIDIA® System Management (NVSM) is an NVIDIA software stack for managing and monitoring NVIDIA-designed servers such as NVIDIA DGX™, CGX, and RTX servers.

- NVSM is an “always-on” health monitoring engine which catches issues proactively as opposed to other tools which need to be run post facto. By virtue of having deep knowledge of the underlying platform, NVSM has the optimal list of health checks to make as well as how frequently each check needs to happen.
- NVSM CLIs and APIs alleviate the need for users to
  - Have deep knowledge of tools such as ipmitool, dmidecode, lspci, storcli, mdadm, and lsblk.
  - Have deep knowledge of platform details such as the intended PCIe hierarchy, storage hierarchy, or error thresholds.
  - Manually correlate information from several tools; in many cases, the output of one tool needs to be manually parsed to know how to use the next tool. For example, BDF in SEL record vs BDF in lspci just to determine which device is faulty.
- NVSM catches issues which some customers might never notice. For example, some PCIe links might be running at lower link width/speed causing jobs to run slow. Without NVSM, customers might suspect something wrong with their jobs OR worse assume that DGX is simply that slow.
- NVSM provides
  - An on-demand health check suite which runs a battery of tests and reports deviations from expected results.
  - The ability to create a bundle of all relevant system logs required by NVIDIA support when reporting an issue.
  - A secure REST API interface removing the need for users/scripts to log-into the system. So it is easy to develop remote management SW applications using these APIs.
  - A Prometheus metrics exporter which can be enabled so an external Prometheus server can pull critical system metrics from the target DGX nodes.
- NVSM’s call-home feature, if enabled, creates a support ticket on behalf of the customer automatically in case of platform issues, even before the customer notices it.
- In addition, NVSM provides other notification mechanisms like email and PagerDuty.

Currently, NVSM supports the following DGX systems:

- DGX servers: Complete NVSM functionality described in this document.
- DGX Station: Functionality is limited to using the CLI to check the health of the system and obtain diagnostic information.

The following is a high level diagram of the NVSM architecture:



Note: “Always on” functionality is not supported on DGX Station.

1.1. Configurable “Always On” Features

NVSM contains the following features that you can configure using the NVSM CLI:

- Health Monitor Alerts
- Health Monitor Policies

1.1.1. Health Monitor Alerts

Alerts are events of significance that require attention. When a health monitor detects such an event in the subsystem that it monitors, it generates an alert to inform the user. The default behavior is to log the alerts in persistent storage as well as to send an E-mail notification to registered users. Refer to the section *Using the NVSM CLI* for details about configuring users for receiving alert E-mail notifications.

Each alert has a ‘state’. An active alert can be in a ‘critical’ or ‘warning’ state. Here, ‘critical’ implies an event that needs immediate action, and ‘warning’ implies an event that needs user attention. When the alerting condition is removed, the alert state changes to ‘cleared’. Details of how to view the generated alerts recorded in the database are available in the section *Using the NVSM CLI*.

1.1.2. Health Monitor Alert List

The following table describes each alert ID:

Message and details	Alert ID	Component ID	Severity	Recommended Action
Unsupported drive configuration. Affected component URI: {{ index .Params "Uri" }}	NV-DRIVE-01	Drive Slot <>	Warning	See Recommended Action A below.
System entered degraded mode, drive in {{ index .Params "DriveSlot" }} is not supported.	NV-DRIVE-07	Drive Slot <>	Warning	See Recommended Action A below.
Unsupported SED drive configuration.	NV-DRIVE-09	Volume label	Warning	See Recommended Action A below.
Unsupported volume encryption configuration.	NV-DRIVE-10	Volume label	Critical	See Recommended Action A below.
M.2 drive firmware version mismatch.	NV-DRIVE-11	Drive Slot <>	Warning	See Recommended Action A below.
System entered degraded mode, volume {{ index .Params "ComponentName" }} is under rebuild.	NV-VOL-01	Volume name	Warning	See Recommended Action A below.
System entered degraded mode, volume {{ index .Params "ComponentName" }} rebuild failed.	NV-VOL-02	Volume name	Critical	See Recommended Action A below.
System entered degraded mode, volume {{ index .Params "ComponentName" }} is in a degraded state.	NV-VOL-03	Volume name	Critical	See Recommended Action A below.
System entered degraded mode, volume {{ index .Params "ComponentName" }} is inactive or in a failed state.	NV-VOL-04	Volume name	Critical	See Recommended Action A below.
Raid-0 Volume {{ index .Params "ComponentName" }} is misconfigured.	NV-VOL-05	Volume name	Warning	See Recommended Action A below.
Raid-0 data volume for caching is not present.	NV-VOL-06		Informational	See Recommended Action A below.

continues on next page

Table 1 – continued from previous page

Message and details	Alert ID	Component ID	Severity	Recommended Action
EFI partition missing on boot volume. Run 'blkid' to check the partition type.	NV-VOL-09	Volume name	Critical	See Recommended Action A below.
Storage Volume {{ index .Params "ComponentName" }} utilization is nearing 90% of {{ index .Params "Capacity" }} bytes.	NV-VOL-10	Volume name	Critical	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (Power supply module has failed.)	NV-PSU-01	<PSU#> where # is the PSU number.	Critical	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (Operating temperature exceeds the thermal specifications of the component.)	NV-PSU-02	<PSU#> where # is the PSU number.	Warning	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (Input to the PSU is missing)	NV-PSU-03	<PSU#> where # is the PSU number.	Critical	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (Input voltage is out of range for the Power Supply Module) (Input voltage is out of range for the Power Supply Module)	NV-PSU-04	<PSU#> where # is the PSU number.	Critical	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (PSU is missing)	NV-PSU-05	<PSU#> where # is the PSU number.	Warning	See Recommended Action A below.
Failures in Power supply modules have been detected. (System is operating in degraded performance mode.)	NV-PSU-06		Warning	Rectify the issues observed on the PSUs. Then see Recommended Action A below.
Failures in Power supply modules have been detected. (System is in power failed state)	NV-PSU-07		Critical	Rectify the issues observed on the PSUs. Then see Recommended Action A below.

continues on next page

Table 1 – continued from previous page

Message and details	Alert ID	Component ID	Severity	Recommended Action
System entered degraded mode, {} is reporting an error. (Operating temperature exceeds the thermal specifications of the component.)	NV-PDB-01	<PDB#> where # is the PDB number	Critical	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (Fan speed reading has fallen below the expected speed setting.)	NV-FAN-01	<FAN#_F> or <FAN#_R> where # is the fan module number. F is for front fan. R is for rear fan.	Critical	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (Fan readings are inaccessible.)	NV-FAN-02	<FAN#_F> or <FAN#_R> where # is the fan module number. F is for front fan. R is for rear fan.	Critical	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (An unrecoverable CPU Internal error has occurred.)	NV-CPU-01	<CPU#> where # is the CPU socket number (CPU0 or CPU1)	Critical	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (CPU Thermtrip has occurred, processor socket temperature exceeded the thermal specifications of the component.)	NV-CPU-02	<CPU#> where # is the CPU socket number (CPU0 or CPU1)	Critical	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (Processor socket temperature exceeded the thermal specifications of the component.)	NV-CPU-03		Critical	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (Processor socket temperature exceeded the thermal specifications of the component.)	NV-CPU-04		Critical	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (Uncorrectable error is reported).	NV-DIMM-01	<CPU#_DIMM_@$> where # = (1, 2) @ = (A, B, C, D, E, F) $ = (1, 2)	Critical	See Recommended Action A below.

continues on next page

Table 1 – continued from previous page

Message and details	Alert ID	Component ID	Severity	Recommended Action
System entered degraded mode, {} is reporting an error. (Correctable errors reported exceeds the configured threshold.)	NV-DIMM-02	<CPU#_DIMM_@$> where # = (1, 2) @ = (A, B, C, D, E, F) $ = (1, 2)	Warning	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (Unrecoverable error is observed on the DIMM, specific details of the error are unavailable.)	NV-DIMM-03	<CPU#_DIMM_@$> where # = (1, 2) @ = (A, B, C, D, E, F) $ = (1, 2)	Critical	See Recommended Action A below.
System entered degraded mode, {} is reporting an error. (DIMM presence is not expected in this slot, please verify the DIMM details.)	NV-DIMM-04			See Recommended Action A below.
System entered degraded mode, GPU is reporting an error (Critical error has been reported by the GPU.)	NV-GPU-01		Critical	See Recommended Action A below.
GPU{} power Limits are not configured correctly (Expected limits (Power: 200000W, Clock: 1597MHz), Actual limits (Power: 200000W, Clock: 1163MHz).)	NV-GPU-02		Critical	See Recommended Action A below.
System entered degraded mode, {ID} is reporting an error. (Link speed degradation observed between { BDF1, BDF2}, expected link speed is {} actual link speed is {})	NV-PCI-01		Warning	See Recommended Action A below.
System entered degraded mode, {ID} is reporting an error. (Link width degradation observed between {BDF1, BDF2},, expected link width is {} actual link width is {})	NV-PCI-02		Warning	See Recommended Action A below.
System entered degraded mode, {ID} is reporting an error. (Correctable errors reported on {BDF}.)	NV-PCI-03		Warning	See Recommended Action A below.

continues on next page

Table 1 – continued from previous page

Message and details	Alert ID	Component ID	Severity	Recommended Action
System entered degraded mode, {ID} is reporting an error. (UnCorrectable errors reported on {BDF})	NV-PCI-04		Critical	See Recommended Action A below.
System entered degraded mode, {ID} is reporting an error. (Device is missing on {BDF})	NV-PCI-05		Critical	See Recommended Action A below.
System entered degraded mode, {ID} is reporting an error. (Device Error Reporting is disabled on {BDF} for {})	NV-PCI-06		Critical	See Recommended Action A below.
System entered degraded mode, {ID} is reporting an error. (Device is disabled on {BDF})	NV-PCI-07		Critical	See Recommended Action A below.
System entered degraded mode, controller {{ index .Params "ComponentName" }} is reporting an error.	NV-CONTROLLER-01	Controller name	Warning	See Recommended Action A below.
System entered degraded mode, Storage controller {{ index .Params "ComponentName" }} is reporting a PHY error.	NV-CONTROLLER-02	Controller name	Warning	See Recommended Action A below.
System entered degraded mode, controller {{ index .Params "ComponentName" }} is set at lower than expected speed.	NV-CONTROLLER-03	Controller name	Warning	See Recommended Action A below.
System entered degraded mode, controller {{ index .Params "ComponentName" }} is reporting an error.	NV-CONTROLLER-04	Controller name	Warning	See Recommended Action A below.
System entered degraded mode, controller {{ index .Params "ComponentName" }} is reporting an error.	NV-CONTROLLER-05	Controller name	Critical	See Recommended Action A below.
System entered degraded mode, controller {{ index .Params "ComponentName" }} is reporting an error.	NV-CONTROLLER-06	Controller name	Critical	See Recommended Action A below.
LEDStatus for controller {{ index .Params "ComponentName" }} needs to be cleared.	NV-CONTROLLER-07	Controller name	Critical	See Recommended Action A below.

continues on next page

Table 1 – continued from previous page

Message and details	Alert ID	Component ID	Severity	Recommended Action
Link error on {}. (Network Link is down)	NV-NET-01		Warning	See Recommended Action B below.
Network traffic errors observed on {}. (Rx collision rate of {}, has crossed threshold value of {} on {}network port.)	NV-NET-02		Warning	See Recommended Action B below.
Network traffic errors observed on {}. (Tx collision rate of {}, has crossed threshold value of {} on {}network port.)	NV-NET-03		Warning	See Recommended Action B below.
Network traffic errors observed on {}. (CRC error rate of {}, has crossed threshold value of {} on {}network port.)	NV-NET-04		Critical	See Recommended Action B below.
{} is reporting an error. ({}Network port is disabled.)	NV-NET-05		Critical	See Recommended Action B below.
Ethernet interface error on port {}. ({}Ethernet health check failing with Online NVRAM test failure.)	NV-ETH-01		Critical	See Recommended Action B below.
Ethernet interface configuration error on {}. (MAC address is missing on the Ethernet interface of {}.)	NV-ETH-02		Critical	See Recommended Action B below.
IB driver error. (HCA self test reports IB driver initialization failure.)	NV-IB-01		Critical	See Recommended Action C below.
Counter errors on IB port {} ({}HCA self test on IB port reports counter error.)	NV-IB-02		Critical	See Recommended Action B below.
Configuration error on IB port {}. (GUID is missing on {}HCA.)	NV-IB-03		Critical	See Recommended Action D below.
System entered degraded mode, {} is reporting a fatal error (Critical error has been reported by the NVSwitch Id {} with SXID error {})	NV-NVSWITCH-01		Critical	See Recommended Action A below.

continues on next page

Table 1 – continued from previous page

Message and details	Alert ID	Component ID	Severity	Recommended Action
System entered degraded mode, {} is reporting a non fatal error (Critical error has been reported by the NVSwitch Id {} with SXID error {})	NV-NVSWITCH-02		Warning	See Recommended Action A below.

1.1.2.1 Recommended Actions

(A)

1. Run 'sudo nvsm dump health'
2. Open a case with NVIDIA Enterprise Support at this address https://nvid.nvidia.com/dashboard/
3. Attach this notification and the nvsysinfo log file from /tmp/nvsm-health- <hostname>-<timestamp>.tar.xz

(B)

1. Check the physical link connection
2. Open a case with NVIDIA Enterprise Support at https://nvid.nvidia.com/dashboard/

(C)

1. Check OFED installation troubleshooting
2. Open a case with NVIDIA Enterprise Support at this address https://nvid.nvidia.com/dashboard/

(D)

1. Check the status of the Subnet Manager
2. Open a case with NVIDIA Enterprise Support at this address https://nvid.nvidia.com/dashboard/

1.1.3. Health Monitor Policies

Users can tune certain aspects of health monitor behavior using health monitor policies. This includes details such as email related configuration for alert notification, selectively disabling devices to be monitored, etc. Details of the supported policies and how to configure them using the CLI are provided in the section *Using the NVSM CLI*.

1.2. Verifying the Installation

Before using NVSM, you can verify the installation to make sure all the services are present.

1.2.1. Verifying NVSM Services with systemctl

NVSM is part of the DGX OS image and is launched by systemd when DGX boots. The following are the services running under NVSM:

- `nvsm-api-gateway.service`
- `nvsm-core.service`
- `nvsm-mqtt.service`
- `nvsm-notifier.service`
- `nvsm.service`

You can verify if each NVSM service is up and running using the `systemctl` command. For example, the following command verifies the core service:

```
$ sudo systemctl status nvsm-core
```

You can view all the NVSM services and their status with the following command:

```
$ sudo systemctl status -all nvsm*
```

1.2.2. Verifying NVSM Services with nvsm status

The `nvsm status` command displays the NVSM services and their status, example output:

```
$ sudo nvsm status
SERVICE                   ENABLED  ACTIVE  SUB       DESCRIPTION
================================================================================
nvsm-mqtt.service         enabled    active  running         MQTT broker for NVSM API
↪for signaling within NVSM API components
nvsm-core.service         enabled    active  running         NVSM Core Service for
↪System Management
nvsm-api-gateway.service  enabled    active  running         NVSM API Server to
↪provide DGX System Management APIs
nvsm-notifier.service     enabled    active  running         NVSM Notifier service.

Overall
================================================================================
Overall Health: Healthy
Overall Status: Active
```

If you run `nvsm status` while NVSM is starting, the output resembles the following example:

```
$ sudo nvsm status
SERVICE                   ENABLED  ACTIVE     SUB         DESCRIPTION
================================================================================
```

(continues on next page)

(continued from previous page)

```
nvsm-mqtt.service         enabled  active     running     MQTT broker for NVSM API
↪for signaling within NVSM API components
nvsm-core.service         enabled  activating start-post  NVSM Core Service for
↪System Management
nvsm-api-gateway.service  enabled  inactive   dead        NVSM API Server to provide
↪DGX System Management APIs
nvsm-notifier.service     enabled  inactive   dead        NVSM Notifier service.

Overall
================================================================================
Overall Health: Transient
Overall Status: Starting

Recommendations:
================================================================================
0. NVSM is starting, this state should be transient, please try again later
1. nvsm-core.service is activating. If it stay in this state, please run "journalctl -
↪fu nvsm-core.service" for more details
```

Note: The `nvsm` CLI command works only if all NVSM services are up and running.

If any sub service fails or stuck in starting, run the following command to get additional information:

```
sudo systemctl status <service-name>
```

For example:

```
sudo systemctl status nvsm-core.service
```

Chapter 2. Release Notes

2.1. NVSM 24.03.03 Release

NVSM Version 24.03.03 was released in April 2024.

2.1.1. Changes and New Features

The following are the changes in 24.03.03.

- Expanded software health service (`nvsm show health -swh`) to include Kubernete and Slurm stack deployment verification.
- Deprecated `nvsm-health` command in favor of `nvsm show health`.
- Improved NVSM parsing of IPMI System Event Log(SEL) records, to avoid generating false alerts.
- Updated DIMM consistency validation and support for additional DIMM vendors for DGX H100/H800 platforms.

2.1.2. Known Issues

- The `nvsm.service` shows as inactive with GPU driver R550; the issue does not impact any NVSM functionality.
- When more than 56 Virtual Functions (VFs) are created on Infiniband NICs, nvsm show health reports as unhealthy in GPUDirect Topology consistency check. The issue will be fixed in future releases.

2.2. NVSM 23.12.01 Release

NVSM Version 23.12.01 was released in December 2023.

2.2.1. Changes and New Features

The following are the changes in 23.12.01.

- Introduced the software health service (`nvsm show health -swh`) for DGX OS and container stack deployment verification.
- Enhanced functionality to collect MLX cable information in `nvsm dump health`.
- Improved accuracy of NVSM alert generation based on System Event Log (SEL) records.

2.2.2. Bug Fixes

- Fixed an issue with raid volume rebuilding on encrypted root filesystem.

Chapter 3. Using the NVSM CLI

NVIDIA DGX-2 servers running DGX OS version 4.0.1 or later should come with NVSM pre-installed.

NVSM CLI communicates with the privileged NVSM API server, so NVSM CLI requires superuser privileges to run. All examples given in this guide are prefixed with the `sudo` command.

3.1. Using the NVSM CLI Interactively

Starting an interactive session

The command “sudo nvsm” will start an NVSM CLI interactive session.

```
user@dgx-2:~$ sudo nvsm
[sudo] password for user:
nvsm->
```

Once at the “`nvsm->` ” prompt, the user can enter NVSM CLI commands to view and manage the DGX system.

Example command

One such command is “show fans”, which prints the state of all fans known to NVSM.

```
nvsm-> show fans
/chassis/localhost/thermal/fans/FAN10_F
Properties:
    Status_State = Enabled
    Status_Health = OK
    Name = FAN10_F
    MemberId = 19
    ReadingUnits = RPM
    LowerThresholdNonCritical = 5046.000
    Reading = 9802 RPM
    LowerThresholdCritical = 3596.000
    ...
    /chassis/localhost/thermal/fans/PDB_FAN4
Properties:
    Status_State = Enabled
    Status_Health = OK
    Name = PDB_FAN4
    MemberId = 23
    ReadingUnits = RPM
    LowerThresholdNonCritical = 11900.000
```

(continues on next page)

(continued from previous page)

```
    Reading = 14076 RPM
    LowerThresholdCritical = 10744.000
nvsm->
```

Leaving an interactive session

To leave the NVSM CLI interactive session, use the “exit” command.

```
nvsm-> exit
user@dgx2:~$
```

3.2. Using the NVSM CLI Non-Interactively

Any NVSM CLI command can be invoked from the system shell, without starting an NVSM CLI interactive session. To do this, simply append the desired NVSM CLI command to the “sudo nvsm” command. The “show fans” command given above can be invoked directly from the system shell as follows.

```
user@dgx2:~$ sudo nvsm show fans
/chassis/localhost/thermal/fans/FAN10_F
Properties:
    Status_State = Enabled
    Status_Health = OK
    Name = FAN10_F
    MemberId = 19
    ReadingUnits = RPM
    LowerThresholdNonCritical = 5046.000
    Reading = 9802 RPM
    LowerThresholdCritical = 3596.000
...
/chassis/localhost/thermal/fans/PDB_FAN4
Properties:
    Status_State = Enabled
    Status_Health = OK
    Name = PDB_FAN4
    MemberId = 23
    ReadingUnits = RPM
    LowerThresholdNonCritical = 11900.000
    Reading = 14076 RPM
    LowerThresholdCritical = 10744.000
user@dgx2:~$
```

The output of some NVSM commands can be too large to fit on one screen, it is sometimes useful to pipe this output to a paging utility such as “less”.

```
user@dgx2:~$ sudo nvsm show fans | less
```

Throughout this chapter, examples are given for both interactive and non-interactive NVSM CLI use cases. Note that these interactive and non-interactive examples are interchangeable.

3.3. Getting Help

Apart from the NVSM CLI User Guide (this document), there are many sources for finding additional help for NVSM CLI and the related NVSM tools.

3.3.1. nvsm “man” Page

A man page for NVSM CLI is included on DGX systems with NVSM installed. The user can view this man page by invoking the “man nvsm” command.

```
user@dgx2:~$ man nvsm
```

3.3.2. nvsm –help/-h Flag

By passing the –help or -h flag, the nvsm command will display a help message that is similar to “`man nvsm`”. The help message can also be invoked through “`nvsm --help`”. It shows a description, nvsm command verbs, options and a few examples

Example output:

```
user@dgxa100:~$ sudo nvsm --help
Run 'sudo nvsm [command] -h' for a command-specific help message
NVSM(1)                                NVSM CLI                                NVSM(1)

NAME
  nvsm - NVSM CLI Documentation

  User Guide: https://docs.nvidia.com/datacenter/nvsm/latest/pdf/nvsm-user-guide.pdf

SYNOPSIS
  nvsm [help] [--color WHEN] [-i] [--log-level LEVEL] [--] [<command>]

DESCRIPTION
  nvsm(1), also known as NVSM CLI, is a command-line interface for System Management
↪on
  NVIDIA DGX systems. Internally, NVSM CLI is a client of the NVSM (NVIDIA System
↪Management)
  API server, which is facilitated by the nvsm(1) daemon.

  Invoking the nvsm(1) command without any arguments will start an NVSM CLI
↪interactive session.
  Alternatively, by passing commands as part of the [<command>]  argument, NVSM CLI
↪can be run
  in a non-interactive mode.

  Note: nvsm must be run with root privileges.

NVSM COMMANDS
  nvsm show [-h, --help] [-level LEVEL] [-display CATEGORIES] [-all] [target] [where]
↪:
      Display information about devices and other entities managed by NVSM
```

(continues on next page)

(continued from previous page)

```
  nvsm cd [-h, --help] [target]:
      Change the working target address used by NVSM verbs

  nvsm set [-h, --help] [target] :
      Change the value of NVSM target properties

  nvsm start [-h, --help] [-noblock] [-force] [-quiet] [-timeout TIMEOUT] [target] :
      Start a job managed by NVSM

  nvsm dump health [-h, --help] [-o OUTPUT] [-t, -tags "tag1,tag2"]
      [-tfp, -tar_file_path "/x/y/path"] [-tfn, -tar_file_name "name.tar.xz"] :
      Generates a health report file

  nvsm stress-test [--usage, -h, --help] [-force] [-no-prompt] [<test>] [DURATION] :
      NVIDIA System Management Stress Testing

  nvsm lock [-h, --help] [target] :
      Enable locking of SED

  nvsm create [-h, --help] [target] :
      The create command is used to generate new resources on demand

OPTIONS
  --color WHEN
      Control colorization of output. Possible values for WHEN are "always", "never",
↪or "auto".
      Default value is "auto".

  -i, --interactive
      When this option is given, run in interactive mode. The default is automatic.

  --log-level LEVEL
      Set the output logging level. Possible values for LEVEL are "debug", "info",
↪"warning",
      "error", and "critical". The default value is "warning".

EXAMPLES
   sudo nvsm help
      Display the help message for NVSM CLI

   sudo nvsm show -h
      Display the help message for the NVSM show command

   sudo nvsm show gpus
      Display information for all GPUs in the system.

   sudo nvsm
      Run nvsm in interactive mode

   sudo nvsm show versions
      Display system version properties

   sudo nvsm update firmware
      Run through the steps of selecting a firmware update container on the local DGX
↪system,
```

(continues on next page)

(continued from previous page)

```
      and running it to update the firmware on the system. This requires that you
↪have already
      loaded the container onto the DGX system.

   sudo nvsm dump health
      Produce a health report file suitable for attaching to support tickets.

AUTHOR
       NVIDIA Corporation

COPYRIGHT
       2021, NVIDIA Corporation
```

3.3.3. Help for NVSM CLI Commands

Each NVSM command verb within the NVSM CLI interactive session, such as show, `cd, set, start, dump health, stress-test, lock` and `create` recognizes a "`-h` " or "`--help` " flag that describes the NVSM command and its arguments. These commands also have their own man pages, which can be invoked, for example, using "`man nvsm_show` ".

The help messages show the description, NVSM command nouns (or sub commands), options and examples.

Example output:

```
user@dgxa100:~$ sudo nvsm show -h
NVSM_SHOW(1)               NVSM CLI                                  NVSM_SHOW(1)

NAME
   nvsm_show - NVSM SHOW CLI Documentation

SYNOPSIS
   nvsm show [-h, --help] [-level LEVEL] [-display CATEGORIES] [-all] [target] [where]

DESCRIPTION
   Show is used to display information about system components. It displays
↪information
   about devices and other entities managed by NVSM

OPTIONS
   --help, -h
          show this help message and exit

   -level LEVEL, -l LEVEL
          Specify the target depth level to which the show command will traverse the
          target hierarchy.
          The default value for LEVEL is 1, which means "the current target only".

   -display CATEGORIES, -d CATEGORIES
          Select  the  categories of information displayed about the given target.
          Valid values for CATEGORIES are 'associations', 'targets', 'properties',
↪'verbs',
          and 'all'. The default value for CATEGORY is 'all'. Multiple values can be
          specified by separating those values with colon. Sub-arguments for
↪properties
```

(continues on next page)

(continued from previous page)

```
          are supported which are separated by comma with paranthesis as optional.

   -all, -a
          Show data that are normally hidden. This includes OEM properties and OEM
↪targets
          unique to NVSM.

   target The target address of the Managed Element to show. The target address can
↪be relative
          to the current working target, or it can be absolute. Simple globbing to
↪select multiple
          Managed Elements is also possible.

   where  Using this argument, targets can be filtered based on the value of their
↪properties.
          This can be used to quickly find targets with interesting properties.
↪Currently this
          supports '==' and '!=' operations, which mean 'equal' and 'not equal'
↪respectively.
          UNIX-style wildcards using '*' are also supported.

COMMANDS
   show alerts
          Display warnings and critical alerts for all subsystems

   show drives
          Display the storage drives

   show versions
          Display system version properties

   show fans
          Display information for all the fans in the system.

   show firmware
          Walk through steps of selecting a firmware update container on the local
↪DGX system,
          and run it to show the firmware versions installed on system. This requires
↪that you
          have already loaded the container onto the DGX system.

   update firmware
          Walk  through  steps of selecting a firmware update container on the local
↪DGX system,
          and run it to update the firmware on system. This requires that you have
↪already loaded
          the container onto the DGX system.

   show gpus
          Display information for all GPUs in the system

   show health
          Display overall system health

   show memory
          Display information for all installed DIMMs
```

(continues on next page)

(continued from previous page)

```
   show networkadapters
          Display information for the physical network adapters

   show networkdevicefunctions
          Display information for the PCIe functions for a given network adapter

   show networkinterfaces
          Display information for each logical network adapter on the system.

   show networkports
          Display information for the network ports of a given networkadapter

   show nvswitches
          Display information for all the NVSwitch interconnects in the system.

   show policy
          Display alert policies for subsystems

   show power
          Display information for all power supply units (PSUs) in the system.

   show processors
          Display information for all processors in the system.

   show storage
          Display storage related information

   show temperature
          Display temperature information for all sensors in the system

   show volumes
          Show storage volumes

   show powermode
          Display the current system power mode

   show led
          Lists values for available system LED status. Includes u.2 NVME, Chassis/
↪Blade LED
          status(on applicable platforms) disable exporters
          Disable NVSM metric collection data

   show controllers
          List applicable controllers properties. Applicable for SAS storage
↪controller in dgx1,
          and M.2 and U.2 NVMe controller properties for other platforms.

EXAMPLES
   sudo nvsm show -h
          Display the help message for the NVSM show command

   sudo nvsm show health -h
          Display the help message for the NVSM show health command

   sudo nvsm show gpus
```

(continues on next page)

(continued from previous page)

```
          Display information for all GPUs in the system.

   sudo nvsm show versions
          Display system version properties

   sudo nvsm show storage
          View all storage-related information

   sudo nvsm show processors
          Information for all CPUs installed on the system

AUTHOR
       NVIDIA Corporation

COPYRIGHT
       2021, NVIDIA Corporation

21.07.12-4-g5586f4ba   Aug 04, 2021            NVSM_SHOW(1)
```

When a wrong command is entered, the CLI prompts the user to check the specified help message.

```
:~$ sudo nvsm show wrong_command
ERROR:nvsm:Target address "wrong_command" does not exist
Run: 'sudo nvsm show --help' for more options
```

3.4. Setting DGX H100 BMC Redfish Password

In DGX H100, Redfish services in BMC can be accessed using the BMC Redfish host IP address, which is termed the **Host Interface**. NVSM deployed on the Host OS communicates over Host Interface with the BMC Redfish services for the system data.

The Redfish host interface is a secured communication channel. As a prerequisite, BMC credentials with minimal read type access is set up in the Host OS before making any communication with the BMC Redfish services via NVSM.

The following NVSM commands sets up the BMC credentials for NVSM consumption in Host OS:

```
# nvsm set -bmccred   (or)   # nvsm set --bmccredentials
```

```
$ sudo nvsm set -bmccred
BMC credentials entered will be encrypted and stored.
Enter BMC username: admin
Enter BMC password:
Re Enter BMC password:
Entered credentials stored successfully.
```

Credentials get encrypted and stored on the Host.

3.5. Examining System Health

The most basic functionality of NVSM CLI is examination of system state. NVSM CLI provides a “show” command for this purpose.

Because NVSM CLI is modeled after the SMASH CLP, the output of the NVSM CLI “show” command should be familiar to users of BMC command line interfaces.

3.5.1. List of Basic Commands

The following table lists the basic commands (primarily “show”). Detailed use of these commands are explained in subsequent sections of the document.

Note: On DGX Station, the following are the only commands supported.

- `nvsm show health`
- `nvsm dump health`

Global Commands	Descriptions
$ sudo nvsm show alerts	Displays warnings and critical alerts for all subsystems
$ sudo nvsm show policy	Displays alert policies for subsystems
$ sudo nvsm show versions	Displays system version properties

Health Commands	Descriptions
$ sudo nvsm show health	Displays overall system health
$ sudo nvsm dump health	Generates a health report file

Storage Commands	Descriptions
$ sudo nvsm show storage	Displays all storage-related information
$ sudo nvsm show drives	Displays the storage drives
$ sudo nvsm show controllers	Display the storage controllers
$ sudo nvsm show volumes	Displays the storage volumes

GPU Commands	Descriptions
$ sudo nvsm show gpus	Displays information for all GPUs in the system.

Processor Commands	Descriptions
$ sudo nvsm show processors	Displays information for all CPUs in the system
$ sudo nvsm show cpus	Alias for “show processors”

Memory Commands	Descriptions
$ sudo nvsm show memory	Displays information for all installed DIMMs
$ sudo nvsm show dimms	Alias for “show memory”

Thermal Commands	Descriptions
$ sudo nvsm show fans	Displays information for all the fans in the system.
$ sudo nvsm show temperatures	Displays temperature information for all sensors in the system
$ sudo nvsm show temps	Alias for “show temperatures”

Network Commands	Descriptions
$ sudo nvsm show networkadapters	Displays information for the physical network adapters
$ sudo nvsm show networkinterfaces	Displays information for the logical network interfaces
$ sudo nvsm show networkports	Displays information for the network ports of a given network adapter
$ sudo nvsm show networkdevicefunctions	Displays information for the PCIe functions for a given network adapter

Power Commands	Descriptions
$ sudo nvsm show power	Displays information for all power supply units (PSUs) in the system.
$ sudo nvsm show powermode	Display the current system power mode
$ sudo nvsm show psus	Alias for “show power”

NVSwitch Commands	Descriptions
$ sudo nvsm show nvswitches	Displays information for all the NVSwitch interconnects in the system.

Firmware Commands	Descriptions
$ sudo nvsm show firmware	Guides you through the steps of selecting a firmware update container on your local DGX system, and running it to show the firmware versions installed on the system. This requires that you have already loaded the container onto the DGX system.
$ sudo nvsm update firmware	Guides you through the steps of selecting a firmware update container on your local DGX system, and running it to update the firmware on the system. This requires that you have already loaded the container onto the DGX system.

3.5.2. Show Health

The “show health” command can be used to quickly assess overall system health.

```
user@dgx-2:~$ sudo nvsm show health
```

Example output:

```
...
Checks
------Verify installed DIMM memory sticks.........................
HealthyNumber of logical CPU cores [96]..........................
HealthyGPU link speed [0000:39:00.0][8GT/s]......................
HealthyGPU link width [0000:39:00.0][x16].........................
Healthy
...
Health Summary
--------------
205 out of 205 checks are Healthy
Overall system status is Healthy
```

If any system health problems are found, this will be reflected in the health summary at the bottom of the “show health” output”. Detailed information on health checks performed will appear above.

3.5.3. Dump Health

The “dump health” command produces a health report file suitable for attaching to support tickets.

```
user@dgx-2:~$ sudo nvsm dump health
```

Example output:

```
Writing output to /tmp/nvsm-health-dgx-1-20180907085048.tar.xzDone.
```

The file produced by “dump health” is a familiar compressed tar archive, and its contents can be examined by using the “tar” command as shown in the following example.

```
user@dgx-2:~$ cd /tmp
user@dgx-2:/tmp$ sudo tar xlf nvsm-health-dgx-1-20180907085048.tar.xz
user@dgx-2:/tmp$ sudo ls ./nvsm-health-dgx-1-20180907085048
date            java         nvsysinfo_commands  sos_reports
df              last         nvsysinfo_log.txt   sos_strings
dmidecode       lib          proc                sys
etc             lsb-release  ps                  uname
free            lsmod        pstree              uptime
hostname        lsof         route               usr
initctl         lspci        run                 var
installed-debs  mount        sos_commands        version.txt
ip_addr         netstat      sos_logs            vgdisplay
```

The option `-qkd` or `--quick_dump` can be used to collect the health report more quickly, at the cost of higher CPU/memory consumption.

```
# nvsm dump health -qkd
```

3.5.4. Show Versions

The `nvsm show versions` command displays hardware components on board, along with their firmware versions. It also shows the installed version of NVSM, Datacenter GPU Manager, and OS among others.

```
user@dgxa100:~$ sudo nvsm show versions
```

After installing NVSM, the command `sudo nvsm show versions` will take approximately three minutes to run.

Example output:

```
initializing NVSM Core...

/versions
Properties:
    dgx-release = 5.1.0
    nvidia-driver = 470.57.01
    cuda-driver = 11.4
    os-release = Ubuntu 20.04.2 LTS (Focal Fossa)
    kernel = 5.4.0-77-generic
    nvidia-container-runtime-docker = 3.4.0-1
    docker-ce = 20.10.7
    platform = DGXA100
    nvsm = 21.07.12-5-g9775e940-dirty
    mlnx-ofed = MLNX_OFED_LINUX-5.4-1.0.3.0:
    datacenter-gpu-manager = 1:2.2.9
    datacenter-gpu-manager-fabricmanager = 470.57.01-1
    sBIOS = 1.03
    vBIOS-GPU-0 = 92.00.45.00.06
```

(continues on next page)

(continued from previous page)

```
    vBIOS-GPU-1 = 92.00.45.00.06
    vBIOS-GPU-2 = 92.00.45.00.06
    vBIOS-GPU-3 = 92.00.45.00.06
    vBIOS-GPU-4 = 92.00.45.00.06
    vBIOS-GPU-5 = 92.00.45.00.06
    vBIOS-GPU-6 = 92.00.45.00.06
    vBIOS-GPU-7 = 92.00.45.00.06
    BMC = 0.14.17
    CEC-BMC-1 = 03.28
    CEC-Delta-2 = 04.00
    PSU-0 Chassis-1  = 01.05.01.05.01.05
    PSU-1 Chassis-1  = 01.05.01.05.01.05
    PSU-2 Chassis-1  = 01.05.01.05.01.05
    PSU-3 Chassis-1  = 01.05.01.05.01.05
    PSU-4 Chassis-1  = 01.05.01.05.01.05
    PSU-5 Chassis-1  = 01.07.01.05.01.06
    MB-FPGA = 0.01.03
    MID-FPGA = 0.01.03
    NvSwitch-0 = 92.10.18.00.02
    NvSwitch-1 = 92.10.18.00.02
    NvSwitch-2 = 92.10.18.00.02
    NvSwitch-3 = 92.10.18.00.02
    NvSwitch-4 = 92.10.18.00.02
    NvSwitch-5 = 92.10.18.00.02
    SSD-nvme0 (S/N S4YPNE0MB00495) System-1 = EPK9CB5Q
    SSD-nvme1 (S/N S436NA0M510827) System-1 = EDA7602Q
    SSD-nvme2 (S/N S436NA0M510817) System-1 = EDA7602Q
    SSD-nvme3 (S/N S4YPNE0MB01307) System-1 = EPK9CB5Q
    SSD-nvme4 (S/N S4YPNE0MC01447) System-1 = EPK9CB5Q
```

3.5.5. Show Storage

NVSM CLI provides a “show storage” command to view all storage-related information. This command can be invoked from the command line as follows.

```
user@dgx-2:~$ sudo nvsm show storage
```

The following NVSM commands also show storage-related information.

- `user@dgx-2:~$ sudo nvsm show drives`
- `user@dgx-2:~$ sudo nvsm show volumes`
- `user@dgx-2:~$ sudo nvsm show controllers`
- `user@dgx-2:~$ sudo nvsm show led`

Within an NVSM CLI interactive session, the CLI targets related to storage are located under the /systems/localhost/storage/1 target.

```
user@dgx2:~$ sudo nvsm
nvsm-> cd /systems/localhost/storage/
nvsm(/systems/localhost/storage/)-> show
```

Example output:

```
/systems/localhost/storage/
Properties:
    DriveCount = 10
    Volumes = [ md0, md1, nvme0n1p1, nvme1n1p1 ]
Targets:
    alerts
    drives
    policy
    volumes
Verbs:
    cd
    show
```

3.5.5.1 Show Storage Alerts

Storage alerts are generated when the DSHM monitoring daemon detects a storage-related problem and attempts to alert the user (via email or otherwise). Past storage alerts can be viewed within an NVSM CLI interactive session under the /systems/localhost/storage/1/alerts target.

```
user@dgx-2:~$ sudo nvsm
nvsm-> cd /systems/localhost/storage/alerts
nvsm(/systems/localhost/storage/alerts)-> show
```

Example output:

```
/systems/localhost/storage/alerts
Targets:
    alert0
    alert1
Verbs:
    cd
    show
```

In this example listing, there appear to be two storage alerts associated with this system. The contents of these alerts can be viewed with the “show” command.

For example:

```
nvsm(/systems/localhost/storage/alerts)-> show alert1
```

```
/systems/localhost/storage/alerts/alert1
Properties:
    system_name = dgx-2
    message_details = EFI System Partition 1 is corrupted
nvme0n1p1
    component_id = nvme0n1p1
    description = Storage sub-system is reporting an error
    event_time = 2018-07-14 12:51:19
    recommended_action =
         1. Please run nvsysinfo
         2. Please open a case with NVIDIA Enterprise Support at this address https://
↪nvid.nvidia.com/enterpriselogin
         3. Attach this notification and the nvsysinfo log file from /tmp/nvsysinfo-
↪XYZ*
```

(continues on next page)

(continued from previous page)

```
    alert_id = NV-VOL-03
    system_serial = productserial
    message = System entered degraded mode, storage sub-system is reporting an error
    severity = Warning
Verbs:
    cd
    show
```

The message seen in this alert suggests a possible EFI partition corruption, which is an error condition that might adversely affect this system's ability to boot. Note that the text seen here reflects the exact message that the user would have seen when this alert was generated.

Possible categories for storage alerts are given in the table below.

Alert ID	Severity	Details
NV-DRIVE-01	Critical	Drive missing
NV-DRIVE-07	Warning	System has unsupported drive
NV-DRIVE-09	Warning	Unsupported SED drive configuration
NV-DRIVE-10	Critical	Unsupported volume encryption configuration
NV-DRIVE-11	Warning	M.2 firmware version mismatch
NV-VOL-01	Critical	RAID-0 corruption observed
NV-VOL-02	Critical	RAID-1 corruption observed
NV-VOL-03	Warning	EFI System Partition 1 corruption observed
NV-VOL-04	Warning	EFI System Partition 2 corruption observed
NV-CONTROLLER-01	Warning	Controller is reporting an error
NV-CONTROLLER-02	Warning	Storage controller is reporting PHY error
NV-CONTROLLER-03	Warning	Controller set at lower than expected speed
NV-CONTROLLER-04	Critical	Controller is reporting an error
NV-CONTROLLER-05	Critical	Controller is reporting an error
NV-CONTROLLER-06	Critical	Controller is reporting an error
NV-CONTROLLER-07	Critical	LEDStatus for controller needs to be cleared

3.5.5.2 Show Storage Drives

Within an NVSM CLI interactive session, each storage drive on the system is represented by a target under the /systems/localhost/storage/drives target. A listing of drives can be obtained as follows.

```
user@dgx-2:~$ sudo nvsm
nvsm-> cd /systems/localhost/storage/drives
nvsm(/systems/localhost/storage/drives)-> show
```

Example output:

```
/systems/localhost/storage/drives
Targets:
    nvme0n1
    nvme1n1
    nvme2n1
    nvme3n1
    nvme4n1
    nvme5n1
    nvme6n1
    nvme7n1
    nvme8n1
    nvme9n1
Verbs:
    cd
    show
```

Details for any particular drive can be viewed with the “show” command.

For example:

```
nvsm(/systems/localhost/storage/drives)-> show nvme2n1
```

```
/systems/localhost/storage/drives/nvme2n1
Properties:
    Capacity = 3840755982336
    BlockSizeBytes = 7501476528
    SerialNumber = 18141C244707
    PartNumber = N/A
    Model = Micron_9200_MTFDHAL3T8TCT
    Revision = 100007C0
    Manufacturer = Micron Technology Inc
    Status_State = Enabled
    Status_Health = OK
    Name = Non-Volatile Memory Express
    MediaType = SSD
    IndicatorLED = N/A
    EncryptionStatus = N/A
    HotSpareType = N/A
    Protocol = NVMe
    NegotiatedSpeedsGbs = 0
    Id = 2
Verbs:
    cd
    show
```

3.5.5.3 Show Storage Volumes

Within an NVSM CLI interactive session, each storage volume on the system is represented by a target under the /systems/localhost/storage/volumes target. A listing of volumes can be obtained as follows.

```
user@dgx-2:~$ sudo nvsm

nvsmnvsm-> cd /systems/localhost/storage/volumes
nvsm(/systems/localhost/storage/volumes)-> show
```

Example output:

```
/systems/localhost/storage/volumes
Targets:
    md0
    md1
    nvme0n1p1
    nvme1n1p1
Verbs:
    cd
    show
```

Details for any particular volume can be viewed with the “show” command.

For example:

```
nvsm(/systems/localhost/storage/volumes)-> show md0
```

```
/systems/localhost/storage/volumes/md0P
roperties:
    Status_State = Enabled
    Status_Health = OK
    Name = md0
    Encrypted = False
    VolumeType = RAID-1
    Drives = [ nvme0n1, nvme1n1 ]
    CapacityBytes = 893.6G
    Id = md0
Verbs:
    cd
    show
```

3.5.6. Show GPUs

Information for all GPUs installed on the system can be viewed invoking the “show gpus” command as follows.

```
user@dgx-2:~$ sudo nvsm show gpus
```

Within an NVSM CLI interactive session, the same information can be accessed under the /systems/localhost/gpus CLI target.

```
user@dgx-2:~$ sudo nvsm
nvsm-> cd /systems/localhost/gpus
nvsm(/systems/localhost/gpus)-> show
```

Example output:

```
/systems/localhost/gpus
Targets:
    0
    1
    2
    3
    4
    5
```

(continues on next page)

(continued from previous page)

```
    6
    7
    8
    9
    10
    11
    12
    13
    14
    15
Verbs:
    cd
    show
```

Details for any particular GPU can also be viewed with the “show” command.

For example:

```
nvsm(/systems/localhost/gpus)-> show 6
/systems/localhost/gpus/6
Properties:
    Inventory_ModelName = Tesla V100-SXM3-32GB
    Inventory_UUID = GPU-4c653056-0d6e-df7d-19c0-4663d6745b97
    Inventory_SerialNumber = 0332318503073
    Inventory_PCIeDeviceId = 1DB810DE
    Inventory_PCIeSubSystemId = 12AB10DE
    Inventory_BrandName = Tesla
    Inventory_PartNumber = 699-2G504-0200-000
Verbs:
    cd
    show
```

3.5.6.1 Showing Individual GPUs

Details for any particular GPU can also be viewed with the “show” command.

For example:

```
   nvsm(/systems/localhost/gpus)-> show GPU6
/systems/localhost/gpus/GPU6
Properties:
    Inventory_ModelName = Tesla V100-SXM3-32GB
    Inventory_UUID = GPU-4c653056-0d6e-df7d-19c0-4663d6745b97
    Inventory_SerialNumber = 0332318503073
    Inventory_PCIeDeviceId = 1DB810DE
    Inventory_PCIeSubSystemId = 12AB10DE
    Inventory_BrandName = Tesla
    Inventory_PartNumber = 699-2G504-0200-000
    Specifications_MaxPCIeGen = 3
    Specifications_MaxPCIeLinkWidth = 16x
    Specifications_MaxSpeeds_GraphicsClock = 1597 MHz
    Specifications_MaxSpeeds_MemClock = 958 MHz
    Specifications_MaxSpeeds_SMClock = 1597 MHz
    Specifications_MaxSpeeds_VideoClock = 1432 MHz
    Connections_PCIeGen = 3
    Connections_PCIeLinkWidth = 16x
```

(continues on next page)

(continued from previous page)

```
    Connections_PCIeLocation = 00000000:34:00.0
    Power_PowerDraw = 50.95 W
    Stats_ErrorStats_ECCMode = Enabled
    Stats_FrameBufferMemoryUsage_Free = 32510 MiB
    Stats_FrameBufferMemoryUsage_Total = 32510 MiB
    Stats_FrameBufferMemoryUsage_Used = 0 MiB
    Stats_PCIeRxThroughput = 0 KB/s
    Stats_PCIeTxThroughput = 0 KB/s
    Stats_PerformanceState = P0
    Stats_UtilDecoder = 0 %
    Stats_UtilEncoder = 0 %
    Stats_UtilGPU = 0 %
    Stats_UtilMemory = 0 %
    Status_Health = OK
Verbs:
    cd
    show
```

3.5.6.2 Identifying GPU Health Incidents

Explain the benefits of the task, the purpose of the task, who should perform the task, and when to perform the task in 50 words or fewer.

NVSM uses NVIDIA Data Center GPU Manager (DCGM) to continuously monitor GPU health, and reports GPU health issues as "GPU health incidents". Whenever GPU health incidents are present, NVSM indicates this state in the "`Status_HealthRollup`" property of the `/systems/localhost/gpus` CLI target.

"`Status_HealthRollup`" captures the overall health of all GPUs in the system in a single value. Check the "`Status_HealthRollup`" property before checking other properties when checking for GPU health incidents.

To check for GPU health incidents, do the following,

1. Display the "Properties" section of GPU health

   ```
   ~$ sudo nvsm
   nvsm-> cd /systems/localhost/gpus
   nvsm(/systems/localhost/gpus)-> show -display properties
   ```

   A system with a GPU-related issue might report the following.

   ```
   Properties:
       Status_HealthRollup = Critical
       Status_Health = OK
   ```

   The "`Status_Health = OK`" property in this example indicates that NVSM did not find any system-level problems, such as missing drivers or incorrect device file permissions.

   The "`Status_HealthRollup = Critical`" property indicates that at least one GPU in this system is exhibiting a "Critical" health incident.

2. To find this GPU, issue the following command to list the health status for each GPU..

   ```
   ~$ sudo nvsm
   nvsm-> show -display properties=*health /systems/localhost/gpus/*
   ```

The GPU with the health incidents will be reported as in the following example for GPU14.

```
/systems/localhost/gpus/GPU14
Properties:
    Status_Health = Critica
```

3. Issue the following command to show the detailed health information for a particular GPU (GPU14 in this example).

```
nvsm-> cd /systems/localhost/gpus
nvsm(/systems/localhost/gpus)-> show -level all GPU14/health
```

The output shows all the incidents involving that particular GPU.

```
/systems/localhost/gpus/GPU14/health
Properties:
    Health = Critical
Targets:
    incident0
Verbs:
    cd
    show/systems/localhost/gpus/GPU2/health/incident0
Properties:
    Message = GPU 14's NvLink link 2 is currently down.
    Health = Critical
    System = NVLink
Verbs:
    cd
    show
```

The output in this example narrows down the scope to a specific incident (or incidents) on a specific GPU. DCGM will monitor for a variety of GPU conditions, so check “`Status_HealthRollup` ” using NVSM CLI to understand each incident.

3.5.7. Show Processors

Information for all CPUs installed on the system can be viewed using the “show processors” command.

```
user@dgx-2$ sudo nvsm show processors
```

From within an NVSM CLI interactive session, the same information is available under the /systems/localhost/processors target.

```
user@dgx-2:~$ sudo nvsm
nvsm-> cd /systems/localhost/processors
nvsm(/systems/localhost/processors)-> show
```

Example output:

```
/systems/localhost/processors
Targets:
    CPU0
    CPU1
    alerts
    policy
```

(continues on next page)

(continued from previous page)

```
Verbs:
    cd
    show
```

Details for any particular CPU can be viewed using the “show” command.

For example:

```
nvsm(/systems/localhost/processors)-> show CPU0/systems/localhost/processors/CPU0
Properties:
    Id = CPU0
    InstructionSet = x86-64
    Manufacturer = Intel(R) Corporation
    MaxSpeedMHz = 3600
    Model = Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz
    Name = Central Processor
    ProcessorArchitecture = x86
    ProcessorId_EffectiveFamily = 6
    ProcessorId_EffectiveModel = 85
    ProcessorId_IdentificationRegisters = 0xBFEBFBFF00050654
    ProcessorId_Step = 4
    ProcessorId_VendorId = GenuineIntel
    ProcessorType = CPU
    Socket = CPU 0
    Status_Health = OK
    Status_State = Enabled
    TotalCores = 24
    TotalThreads = 48
Verbs:
    cd
    show
```

3.5.7.1 Show Processor Alerts

Processor alerts are generated when the DSHM monitoring daemon detects a CPU Internal Error (IERR) or Thermal Trip and attempts to alert the user (via email or otherwise). Past processor alerts can be viewed within an NVSM CLI interactive session under the /systems/localhost/processors/alerts target.

```
user@dgx-2:~$ sudo nvsm
nvsm-> cd /systems/localhost/processors/alerts
nvsm(/systems/localhost/processors/alerts)-> show
```

Example output:

```
/systems/localhost/processors/alerts
Targets:
    alert0
    alert1
    alert2
Verbs:
    cd
    show
```

This example listing appears to show three processor alerts associated with this system. The contents of these alerts can be viewed with the “show” command.

For example:

```
nvsm(/systems/localhost/processors/alerts)-> show alert2
```

```
/systems/localhost/processors/alerts/alert2
Properties:
      system_name = xpl-bu-06
      component_id = CPU0
      description = CPU is reporting an error.
      event_time = 2018-07-18T16:42:20.580050
      recommended_action =
      1. Please run nvsysinfo
      2. Please open a case with NVIDIA Enterprise Support at this address https://
↪nvid.nvidia.com/enterpriselogin
      3. Attach this notification and the nvsysinfo log file from /tmp/nvsysinfo-XYZ*
      severity = Critical
      alert_id = NV-CPU-02
      system_serial = To be filled by O.E.M.
      message = System entered degraded mode, CPU0 is reporting an error.
      message_details = CPU Thermtrip has occurred, processor socket temperature
↪exceeded the thermal specifications of the component.
Verbs:
    cd
    show
```

Possible categories for processor alerts are given in the table below.

Alert ID	Severity	Details
NV-CPU-01	Critical	An unrecoverable CPU Internal error has occurred.
NV-CPU-02	Critical	CPU Thermtrip has occurred, processor socket temperature exceeded the thermal specifications of the component.

3.5.8. Show Memory

Information for all system memory (i.e. all DIMMs installed near the CPU, not including GPU memory) can be viewed using the “show memory” command.

```
user@dgx-2:~$ sudo nvsm show memory
```

From within an NVSM CLI interactive session, system memory information is accessible under the /systems/localhost/memory target.

```
lab@xpl-dvt-42:~$ sudo nvsm
nvsm-> cd /systems/localhost/memory
nvsm(/systems/localhost/memory)-> show
```

Example output:

```
/systems/localhost/memory
Targets:
    CPU0_DIMM_A1
```

(continues on next page)

(continued from previous page)

```
    CPU0_DIMM_A2
    CPU0_DIMM_B1
    CPU0_DIMM_B2
    CPU0_DIMM_C1
    CPU0_DIMM_C2
    CPU0_DIMM_D1
    CPU0_DIMM_D2
    CPU0_DIMM_E1
    CPU0_DIMM_E2
    CPU0_DIMM_F1
    CPU0_DIMM_F2
    CPU1_DIMM_G1
    CPU1_DIMM_G2
    CPU1_DIMM_H1
    CPU1_DIMM_H2
    CPU1_DIMM_I1
    CPU1_DIMM_I2
    CPU1_DIMM_J1
    CPU1_DIMM_J2
    CPU1_DIMM_K1
    CPU1_DIMM_K2
    CPU1_DIMM_L1
    CPU1_DIMM_L2
    alerts    policy
Verbs:
    cd
    show
```

Details for any particular memory DIMM can be viewed using the “show” command.

For example:

```
nvsm(/systems/localhost/memory)-> show CPU2_DIMM_B1
```

```
/systems/localhost/memory/CPU2_DIMM_B1
Properties:
    CapacityMiB = 65536
    DataWidthBits = 64
    Description = DIMM DDR4 Synchronous
    Id = CPU2_DIMM_B1
    Name = Memory Instance
    OperatingSpeedMhz = 2666
    PartNumber = 72ASS8G72LZ-2G6B2
    SerialNumber = 1CD83000
    Status_Health = OK
    Status_State = Enabled
    VendorId = Micron
Verbs:
    cd
    show
```

3.5.8.1 Show Memory Alerts

On DGX systems with a Baseboard Management Controller (BMC), the BMC will monitor DIMMs for correctable and uncorrectable errors. Whenever memory error counts cross a certain threshold (as determined by SBIOS), a memory alert is generated by the DSHM daemon in an attempt to notify the user (via email or otherwise).

Past memory alerts are accessible from an NVSM CLI interactive session under the /systems/localhost/memory/alerts target.

```
user@dgx-2:~$ sudo nvsm
nvsm-> cd /systems/localhost/memory/alerts
nvsm(/systems/localhost/memory/alerts)-> show
```

Example output:

```
/systems/localhost/memory/alerts
Targets:
    alert0
Verbs:
    cd
    show
```

This example listing appears to show one memory alert associated with this system. The contents of this alert can be viewed with the “show” command.

For example:

```
nvsm(/systems/localhost/memory/alerts)-> show alert0
```

```
/systems/localhost/memory/alerts/alert0
Properties:
   system_name = xpl-bu-06
   component_id = CPU1_DIMM_A2
   description = DIMM is reporting an error.
   event_time = 2018-07-18T16:48:09.906572
   recommended_action =
       1. Please run nvsysinfo
       2. Please open a case with NVIDIA Enterprise Support at this address https://
↪nvid.nvidia.com/enterpriselogin
       3. Attach this notification and the nvsysinfo log file from /tmp/nvsysinfo-XYZ*
   severity = Critical
   alert_id = NV-DIMM-01
   system_serial = To be filled by O.E.M.
   message = System entered degraded mode, CPU1_DIMM_A2 is reporting an error.
   message_details = Uncorrectable error is reported.
Verbs:
    cd
    show
```

Possible categories for memory alerts are given in the table below.

Alert Type	Severity	Details
NV-DIMM-01	Critical	Uncorrectable error is reported.

3.5.9. Show Fans and Temperature

NVSM CLI provides a “show fans” command to display information for each fan on the system.

```
~$ sudo nvsm show fans
```

Likewise, NVSM CLI provides a “show temperatures” command to display temperature information for each temperature sensor known to NVSM.

```
~$ sudo nvsm show temperatures
```

Within an NVSM CLI interactive session, targets related to fans and temperature are located under the /chassis/localhost/thermal target.

```
~$ sudo nvsm
nvsm-> cd /chassis/localhost/thermal
nvsm(/chassis/localhost/thermal)-> show
```

Example output:

```
/chassis/localhost/thermal
Targets:
    alerts
    fans
    policy
    temperatures
Verbs:
    cd
    show
```

3.5.9.1 Show Thermal Alerts

The DSHM daemon monitors fan speed and temperature sensors. When the values of these sensors violate certain threshold criteria, DSHM generates a thermal alert in an attempt to notify the user (via email or otherwise).

Past thermal alerts can be viewed in an NVSM CLI interactive session under the /chassis/localhost/thermal/alerts target.

```
user@dgx-2:~$ sudo nvsm
nvsm-> cd /chassis/localhost/thermal/alerts
nvsm(/chassis/localhost/thermal/alerts)-> show
```

Example output:

```
/chassis/localhost/thermal/alerts
Targets:
    alert0
Verbs:
    cd
    show
```

This example listing appears to show one thermal alert associated with this system. The contents of this alert can be viewed with the “show” command.

For example:

```
nvsm(/chassis/localhost/thermal/alerts)-> show alert0
/chassis/localhost/thermal/alerts/alert0
Properties:
   system_name = system-name
    component_id = FAN1_R
    description = Fan Module is reporting an error.
    event_time = 2018-07-12T15:12:22.076814
    recommended_action =
        1. Please run nvsysinfo
        2. Please open a case with NVIDIA Enterprise Support at this address https://
↪nvid.nvidia.com/enterpriselogin        3. Attach this notification and the
↪nvsysinfo log file from /tmp/nvsysinfo-XYZ*
    severity = Critical
    alert_id = NV-FAN-01
    system_serial = To be filled by O.E.M.
    message = System entered degraded mode, FAN1_R is reporting an error.
    message_details = Fan speed reading has fallen below the expected speed setting.
Verbs:    cd    show
```

From the message in this alert, it appears that one of the rear fans is broken in this system. This is the exact message that the user would have received at the time this alert was generated, assuming alert notifications were enabled.

Possible categories for thermal-related (fan and temperature) alerts are given in the table below.

Alert ID	Severity	Details
NV-FAN-01	Critical	Fan speed reading has fallen below the expected speed setting.
NV-FAN-02	Critical	Fan readings are inaccessible.
NV-PDB-01	Critical	Operating temperature exceeds the thermal specifications of the component.

3.5.9.2 Show Fans

Within an NVSM CLI interactive session, each fan on the system is represented by a target under the /chassis/localhost/thermal/fans target. The “show” command can be used to obtain a listing of fans on the system.

```
user@dgx-2:~$ sudo nvsm
```

```
nvsm-> cd /chassis/localhost/thermal/fans
```

```
nvsm(/chassis/localhost/thermal/fans)-> show
```

Example output:

```
/chassis/localhost/thermal/fans
Targets:
    FAN10_F
    FAN10_R
    FAN1_F
    FAN1_R
```

(continues on next page)

(continued from previous page)

```
    FAN2_F
    FAN2_R
    FAN3_F
    FAN3_R
    FAN4_F
    FAN4_R
    FAN5_F
    FAN5_R
    FAN6_F
    FAN6_R
    FAN7_F
    FAN7_R
    FAN8_F
    FAN8_R
    FAN9_F
    FAN9_R
    PDB_FAN1
    PDB_FAN2
    PDB_FAN3
    PDB_FAN4
Verbs:
    cd
    show
```

Again using the “show” command, the details for any given fan can be obtained as follows.

For example:

```
nvsm(/chassis/localhost/thermal/fans)-> show PDB_FAN2
/chassis/localhost/thermal/fans/PDB_FAN2
Properties:
    Status_State = Enabled
    Status_Health = OK
    Name = PDB_FAN2
    MemberId = 21
    ReadingUnits = RPM
    LowerThresholdNonCritical = 11900.000
    Reading = 13804 RPM
    LowerThresholdCritical = 10744.000
Verbs:
    cd
    show
```

3.5.9.3 Show Temperatures

Each temperature sensor known to NVSM is represented as a target under the /chassis/localhost/thermal/temperatures target. A listing of temperature sensors on the system can be obtained using the following commands.

```
nvsm(/chassis/localhost/thermal/temperatures)-> show
```

Example output:

```
/chassis/localhost/thermal/temperatures
Targets:
```

(continues on next page)

(continued from previous page)

```
    PDB1
    PDB2
Verbs:
    cd
    show
```

As with fans, the details for any temperature sensor can be viewed with the “show” command.

For example:

```
nvsm(/chassis/localhost/thermal/temperatures)-> show PDB2
/chassis/localhost/thermal/temperatures/PDB2
Properties:
    Status_State = Enabled
    Status_Health = OK
    Name = PDB2
    PhysicalContext = PDB
    MemberId = 1
    ReadingCelsius = 20 degrees C
    UpperThresholdNonCritical = 127.000
    SensorNumber = 66h
    UpperThresholdCritical = 127.000
Verbs:
    cd
    show
```

3.5.10. Show Power Supplies

NVSM CLI provides a “show power” command to display information for all power supplies present on the system.

```
user@dgx-2:~$ sudo nvsm show power
```

From an NVSM CLI interactive session, power supply information can be found under the /chassis/localhost/power target.

```
user@dgx-2:~$ sudo nvsm
nvsm-> cd /chassis/localhost/power
nvsm(/chassis/localhost/power)-> show
```

Example output:

```
/chassis/localhost/power
Targets:
    PSU1
    PSU2
    PSU3
    PSU4
    PSU5
    PSU6
    alerts    policyVerbs:    cd    show
```

Details for any particular power supply can be viewed using the “show” command as follows.

For example:

```
nvsm(/chassis/localhost/power)-> show PSU4
```

```
/chassis/localhost/power/PSU4
Properties:
    Status_State = Present
    Status_Health = OK
    LastPowerOutputWatts = 442
    Name = PSU4
    SerialNumber = DTHTCD18240
    MemberId = 3
    PowerSupplyType = AC
    Model = ECD16010081
    Manufacturer = Delta
Verbs:
    cd
    show
```

3.5.10.1 Show Power Alerts

The DSHM daemon monitors PSU status. When the PSU status is not Ok, DSHM generates a power alert in an attempt to notify the user (via email or otherwise).

Prior power alerts can be viewed under the /chassis/localhost/power/alerts target of an NVSM CLI interactive session.

```
user@dgx-2:~$ sudo nvsm
nvsm-> cd /chassis/localhost/power/alerts
nvsm(/chassis/localhost/power/alerts)-> show
```

Example output:

```
/chassis/localhost/power/alerts
Targets:
    alert0
    alert1
    alert2
    alert3
    alert4
Verbs:
    cd
    show
```

This example listing shows a system with five prior power alerts. The details for any one of these alerts can be viewed using the “show” command.

For example:

```
nvsm(/chassis/localhost/power/alerts)-> show alert4
/chassis/localhost/power/alerts/alert4
Properties:
   system_name = system-name
   component_id = PSU4
   description = PSU is reporting an error.
   event_time = 2018-07-18T16:01:27.462005
   recommended_action =
       1. Please run nvsysinfo
```

(continues on next page)

(continued from previous page)

```
      2. Please open a case with NVIDIA Enterprise Support at this address https://
↪nvid.nvidia.com/enterpriselogin
      3. Attach this notification and the nvsysinfo log file from /tmp/nvsysinfo-XYZ*
   severity = Warning
   alert_id = NV-PSU-05
   system_serial = To be filled by O.E.M.
   message = System entered degraded mode, PSU4 is reporting an error.
   message_details = PSU is missing
Verbs:
    cd
    show
```

Possible categories for power alerts are given in the table below.

Alert ID	Severity	Details
NV-PSU-01	Critical	Power supply module has failed.
NV-PSU-02	Warning	Detected predictive failure of the Power supply module.
NV-PSU-03	Critical	Input to the Power supply module is missing.
NV-PSU-04	Critical	Input voltage is out of range for the Power Supply Module.
NV-PSU-05	Warning	PSU is missing

3.5.11. Show Network Adapters

NVSM CLI provides a `show networkadapters` command to display information for each physical network adapter in the chassis.

```
~$ sudo nvsm show networkadapters
```

Within an NVSM CLI interactive session, targets related to network adapters are located under the `/chassis/localhost/NetworkAdapters` target.

```
~$ sudo nvsm
nvsm-> cd /chassis/localhost/NetworkAdapters
nvsm(/chassis/localhost/NetworkAdapters)-> show
```

3.5.11.1 Display a List of Muted Adapters

To display a list of the muted adapters, run the following command:

```
$ sudo nvsm show /chassis/localhost/NetworkAdapters/policy
/chassis/localhost/NetworkAdapters/policy
Properties:
mute_monitoring = <NOT_SET>
mute_notification = <NOT_SET>
```

3.5.11.2 Show Network Ports

NVSM CLI provides a `show networkports` command to display information for each physical network port in the chassis.

```
~$ sudo nvsm show networkports
```

Within an NVSM CLI interactive session, targets related to network adapters are located under the /chassis/localhost/NetworkAdapter/<id>/NetworkPort target, where <id> is one of the network adapter IDs displayed from the `nvsm show networkadapters` command.

```
~$ sudo nvsm
nvsm-> cd /chassis/localhost/NetworkAdapters/<id>/NetworkPorts
nvsm(/chassis/localhost/NetworkAdapters/<id>/NetworkPorts)-> show
```

3.5.11.3 Show Network Device Functions

NVSM CLI provides a `show networkdevicefunctions` command to display information for each network adapter-centric PCIe function in the chassis.

```
~$ sudo nvsm show networkdevicefunctions
```

Within an NVSM CLI interactive session, targets related to network adevice functions are located under the `/chassis/localhost/NetworkAdapter/<id>/NetworkDeviceFunctions` target, where<id> is one of the network adapter IDs displayed from the `nvsm show networkadapters` command.

```
~$ sudo nvsm
nvsm-> cd /chassis/localhost/NetworkAdapters/<id>/NetworkDeviceFunctions
nvsm(/chassis/localhost/NetworkAdapters/<id>/NetworkDeviceFunctions)-> show
```

3.5.11.4 Display a List of Interfaces

Run the following command:

```
$ sudo nvsm show /chassis/localhost/NetworkAdapters
/chassis/localhost/NetworkAdapters
Targets:
PCI0000_0c_00
PCI0000_12_00
PCI0000_4b_00
PCI0000_54_00
PCI0000_8d_00
PCI0000_94_00
PCI0000_ba_00
PCI0000_cc_00
PCI0000_e1_00
PCI0000_e2_00
```

3.5.11.5 Show Network Interfaces

NVSM CLI provides a `show networkinterfaces` command to display information for each logical network adapter on the system.

```
~$ sudo nvsm show networkinterfaces
```

In an NVSM CLI interactive session, targets related to network adapters are located under the `/system/localhost/networkinterfaces` target.

```
~$ sudo nvsm
nvsm-> cd /system/localhost/NetworkInterfaces
nvsm(/system/localhost/NetworkInterfaces)-> show
```

3.5.11.6 Add an Interface to the Mute Notifications

Here is an example of a command you can run to add an interface to the mute notifications:

```
$ sudo nvsm set chassis/localhost/NetworkAdapters/policy mute_notification=PCI0000_0c_
↪00,PCI0000_12_00
```

3.6. Examining Software Health

NVSM monitor software health services helps to identify and troubleshoot the system issues which exist at various levels in the software layer. Software layer refers to the installed packages, services and configurations part of the operating system deployed on DGX servers.

Software health service can be displayed using the following command:

```
sudo nvsm show health --software_health
```

Or

```
sudo nvsm show health -swh
```

Example output:

```
Info
----
TimeStamp:           Mon Jan 29 03:30:03 UTC 2024
Nvsm Version:        23.12.01
Product Name:        DGXA100
Serial Number:       <serial number>
Host Name:           <hostname>

Checks
---------

Checking DGX OS packages/services
---------------------------------
Version Compatibility:
  Check nvidia-driver, nvidia-utils, libnvidia-compute..... Healthy
```

(continues on next page)

(continued from previous page)

```
     nvidia-driver:535.129.3  nvidia-utils:535.129.3  libnvidia-compute:535.129.3
  Check nvidia-driver & nvidia-fabricmanager.............................. Healthy
     nvidia-driver:535.129.3  nvidia-fabricmanager:535.129.3
  Check nvidia-driver & libnvidia-nscq.................................... Healthy
     nvidia-driver:535.129.3  libnvidia-nscq:535.129.3
Service check:
  Check nvsm(nvsm.service)................................................ Healthy
  Check persistenced manager(nvidia-persistenced.service)................. Healthy
  Check fabric manager(nvidia-fabricmanager.service)...................... Healthy
  Check mig manager(nvidia-mig-manager.service)........................... Healthy
  Check nvidia acs disable(nvidia-acs-disable.service).................... Healthy
  Check nvidia Mellanox Config(nvidia-mlnx-config.service)................ Healthy
  Check dcgm(nvidia-dcgm.service)......................................... Healthy
Packages check:
  Check dgx-release....................................................... Healthy
  Check base packages..................................................... Healthy
  Check upgrade related packages DGX.....................................
↪Informational
     Package nvidia-peer-memory not installed.
Platform specific checks:
  Check Nvidia built kernel being used.................................... Healthy
     linux-nvidia:5.15.0
  Check packages in hold state...........................................
↪Informational
     Package dgx-a100-system-configurations is in hold state.
     Package dgx-a100-system-tools-extra is in hold state.
     Package dgx-a100-system-tools is in hold state.
        dgx-a100-system-configurations:23.3.-1  dgx-a100-system-tools-extra:22.12.-1
↪dgx-a100-system-tools:22.12.-1
  Check ubuntu upgrade readiness.......................................... Healthy
     ubuntu-release-upgrader-core:22.4.17
  Check Kernel Params..................................................... Healthy
  Check libnvidia-ml.so.1 linked to the installed driver.................. Healthy
     nvidia-driver:535.129.3
  Check nvidia driver installed via .run file............................. Healthy
  Check if nvidia-driver is DKMS installed................................ Healthy
  Check package version consistency....................................... Healthy
  Check dgx-release and dgx-os version.................................... Healthy
     dgx-release:6.1.0
  Check nvidia-driver version installed is loaded......................... Healthy
     nvidia-driver:535.129.3
  Check for any partial upgrade in the system............................. Healthy
  Check MAX_ACC_OUT_READ value set right.................................. Healthy
  Check for key ring validity............................................. Healthy
Version support matrix check:
  Check DGX AX00 matrix................................................... Healthy
Proxy configuration check:
  Check apt proxy configuration........................................... Healthy
     No proxy configuration found.
Package repository configuration check:
  Check dgx repository.................................................... Healthy
  Check nvidia hpc sdk repository......................................... Healthy
     Configuration /etc/apt/preferences.d/hpc-sdk-repo not present.
  Check cuda compute repository..........................................
↪Informational
     Conflicting configuration
```

(continues on next page)

(continued from previous page)

```
     deb [signed-by=/usr/share/keyrings/cuda-archive-keyring.gpg
     https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64/ /
     found in the file /etc/apt/sources.list.d/cuda-ubuntu2004-x86_64.list .
  Check apt update..................................................... Healthy
  Check jammy-updates/dgx priority set to highest...................... Healthy
  Check jammy/dgx priority set to highest.............................. Healthy
  Check jammy/common priority set to highest........................... Healthy
  Check jammy-updates/common priority set to highest................... Healthy

Checking Container infrastructure packages/services
---------------------------------------------------
Version Compatibility:
  Check libnvidia-container-tools & nvidia-container-toolkit.......... Healthy
     libnvidia-container-tools:1.14.3  nvidia-container-toolkit:1.14.3
  Check nvidia-container-toolkit-base & libnvidia-container-tools...... Healthy
     nvidia-container-toolkit-base:1.14.3  libnvidia-container-tools:1.14.3
  Check libnvidia-container1 & libnvidia-container-tools............... Healthy
     libnvidia-container1:1.14.3  libnvidia-container-tools:1.14.3
Service check:
  Check Docker services(docker.service)................................ Healthy
  Check Containerd services(containerd.service)........................ Healthy
Packages check:
  Check base Packages.................................................. Healthy
File configuration checks:
  Check docker configuration.....................................................
↪Informational
     Config default-runtime:nvidia not found in file /etc/docker/daemon.json
     gpus will not get enabled on containers.
  Check container configuration..................................................
↪Informational
     Config default_runtime_name = "nvidia" not found in file /etc/containerd/config.
↪toml
     gpus will not get enabled on containers.

Health Summary
----------------
39 out of 44 checks are healthy
0 out of 44 checks are unhealthy
0 out of 44 checks are unknown
5 out of 44 checks are informational

100.0% [========================================]
Status: Healthy
```

Software health services formats the output as explained below.

3.6.1. Software Health Domains

Domains represent a collection of checks which belong to the same system category. Software health services checks the following domains:

- DGX OS packages/services
- Container infrastructure packages/services
- Kubernetes packages/services, if installed
- Slurm packages/services, if installed

3.6.2. Software Health Checks

Checks, which are constituents of a Domain are categorized as given below:

Index	Checks	Description
1	Version Compatibility	Checks in this category verifies the version compatibility between different software packages.
2	Service check	Checks in this category verifies the state and status of different essential software services.
3	Packages check	Checks in this category verifies the deployment state of essential software packages expected for the platform.
4	Platform specific checks	Checks in this category are specific to a platform or domain. These checks verify various system parameters of the system.
5	Version support matrix check	Checks in this category verifies the deployment of a package and the corresponding version of the package.
6	Proxy configuration check	Checks whether the proxy configuration settings made on the system are in the right state.
7	Package repository configuration check	Checks in this category checks the repository settings and the required settings to perform a software update.
8	File configuration checks	Checks the given configuration file and its related contents are set as expected.

3.7. System Monitoring Configuration

NVSM provides a DSHM service that monitors the state of the DGX system.

NVSM CLI can be used to interact with the DSHM system monitoring service via the NVSM API server.

3.7.1. Configuring Email Alerts

In order to receive the Alerts generated by DSHM through email, configure the Email settings in the global policy using NVSM CLI. User shall receive email whenever a new alert gets generated. The sender address, recipient address(es), SMTP server IP address and SMTP server Port number must be configured according to the SMTP server settings hosted by the user.

Email configuration properties

Property	Description
email_sender	Sender email address Must be a valid email address, otherwise no emails will be sent. [sender@domain.com]
email_recipients	List of recipients to which the email shall be sent [user1@domain.com,user2@domain.com]
email_smtp_server_name	SMTP server name that the user wants to use for relaying email [smtp.domain.com]
email_smtp_server_port	Port Number used by the SMTP server for providing SMTP relay service. Numeric value

The following examples illustrate how to configure email settings in global policy using NVSM CLI.

```
user@dgx-2:~$sudo nvsm set /policy email_sender=dgx-admin@nvidia.com
```

```
user@dgx-2:~$sudo nvsm set /policy email_smtp_server_name=smtpserver.nvidia.com
```

```
user@dgx-2:~$sudo nvsm set /policy email_recipients=jdoe@nvidia.com,jdeer@nvidia.com
```

```
user@dgx-2:~$sudo nvsm set /policy email_smtp_server_port=465
```

3.7.2. Generating a Test Alert for Email

From within an NVSM CLI interactive session, a user may generate a test alert in order to trigger an SMTP instance and receive an email notification.

3.7.2.1 Creating a Test Alert

NVSM CLI provides a "`create testalert`" command to generate a dummy alert that will trigger any SMTP or Call Home defined notification. Within an NVSM CLI interactive session, this basic command generates a dummy alert with default component_``id = Test0`` and `severity = Warning`.

```
~$ sudo nvsm create testalert
```

To configure the Severity and Component of a test alert, issue the following:

```
~$ sudo nvsm create testalert <component_id> <severity>
```

Example of generating a dummy alert with `component_id = Email1` and `severity = Critical`:

```
~$ sudo nvsm create testalert Email1 Critical
```

3.7.2.2 Clearing a Test Alert

NVSM CLI also provides a "`clear testalert`" command to dismiss a generated dummy alert. Within an NVSM CLI interactive session, this basic command will clear any test alert with `component_id=Test0`, even if there are multiple such alerts.

```
~$ sudo nvsm clear testalert
```

To specify which test alert to dismiss, issue the following:

```
~$ sudo nvsm clear testalert <component_id>
```

3.7.2.3 Showing a Test Alert

To display all generated test alerts, the NVSM CLI provides a "`show testalerts`" command

```
~$ sudo nvsm show testalerts
```

Example output:

```
/systems/localhost/testalerts/alert0
Properties:
    system_name = system-name5
    message_details = Dummy Test
    component_id = Test0
    description = No component is reporting an error. This is a test.
    event_time = 2021-08-04T15:55:46.926710484-07:00
    recommended_action = Please run 'sudo nvsm clear testalert' to dismiss this alert.
    alert_id = NV-TEST-01
    system_serial = To be filled by O.E.M.
```

(continues on next page)

(continued from previous page)

```
    message = Test Alert.
    severity = Warning
    clear_time = -
    hidden = false
    type = TestAlerts
```

3.7.3. Understanding System Monitoring Policies

From within an NVSM CLI interactive session, system monitor policy settings are accessible under the following targets.

CLI Target	Description
/policy	Global NVSM monitoring policy, such as email settings for alert notifications.
/systems/localhost/gpus/policy	
/systems/localhost/memory/policy	NVSM policy for monitoring DIMM correctable and uncorrectable errors.
/systems/localhost/processors/policy	NVSM policy for monitoring CPU machine-check exceptions (MCE)
/systems/localhost/storage/policy	NVSM policy for monitoring storage drives and volumes
/chassis/policy	
/chassis/localhost/thermal/policy	NVSM policy for monitoring fan speed and temperature as reported by the baseboard management controller (BMC)
/chassis/localhost/power/policy	NVSM policy for monitoring power supply voltages as reported by the BMC
/chassis/localhost/NetworkAdapters/policy	NVSM policy for monitoring the physical network adapters
/chassis/localhost/NetworkAdapters/<ETH *x*>/NetworkPorts/policy	NVSM policy for monitoring the network ports for the specified Ethernet network adapter
/chassis/localhost/NetworkAdapters/<IB *y*>/NetworkPorts/policy	NVSM policy for monitoring the network ports for the specified InfiniBand network adapter
/chassis/localhost/NetworkAdapters/<ETH *x*>/NetworkDeviceFunctions/policy	NVSM policy for monitoring the PCIe functions for the specified Ethernet network adapter
/chassis/localhost/NetworkAdapters/<IB *y*>/NetworkDeviceFunctions/policy	NVSM policy for monitoring the PCIe functions for the specified InfiniBand network adapter

3.7.3.1 Global Monitoring Policy

Global monitoring policy is represented by the /policy target of NVSM CLI.

```
user@dgx-2:~$ sudo nvsm show /policy
```

Example output:

```
/policy
Properties:
    email_sender = NVIDIA DSHM Service
    email_smtp_server_name = smtp.example.com
    email_recipients = jdoe@nvidia.com,jdeer@nvidia.com
    email_smtp_server_port = 465
Verbs:
    cd
    set
    show
```

The properties for global monitoring policy are described in the table below.

Property	Description
email_sender	Sender email address [sender@domain.com]
email_recipients	List of recipients to which the email shall be sent [user1@domain.com,user2@domain.com]
email_smtp_server_name	SMTP server name that the user wants to use for re-laying email [smtp.domain.com]
email_smtp_server_port	Port Number used by the SMTP server for providing SMTP relay service. Numeric value

3.7.3.2 Memory Monitoring Policy

Memory monitoring policy is represented by the /systems/localhost/memory/policy target of NVSM CLI.

```
user@dgx-2:~$ sudo nvsm show /systems/localhost/memory/policy
```

Example output:

```
/systems/localhost/memory/policy
Properties:
    mute_notification = <NOT_SET>
    mute_monitoring = <NOT_SET>

Verbs:
    cd
    set
    show
```

The properties for memory monitoring policy are described in the table below.

Property	Syntax	Description
mute_notification	List of comma separated DIMM IDs Example: CPU1_DIMM_A1,CPU2_DIMM_F2	Email alert notification is suppressed for devices in the list.
mute_monitoring	List of comma separated DIMM IDs Example: CPU1_DIMM_A1,CPU2_DIMM_F2	Health monitoring is suppressed for devices in the list.

3.7.3.3 Processor Monitoring Policy

Processor monitoring policy is represented by the /systems/localhost/processors/policy target of NVSM CLI.

```
user@dgx-2:~$ sudo nvsm show /systems/localhost/processors/policy
```

Example output:

```
/systems/localhost/processors/policy
Properties:
    mute_notification = <NOT_SET>
    mute_monitoring = <NOT_SET>

Verbs:
    cd
    set
    show
```

The properties for processor monitoring policy are described in the table below.

Property	Syntax	Description
mute_notification	List of comma separated CPU IDs. Example: CPU0,CPU1	Email alert notification is suppressed for devices in the list.
mute_monitoring	List of comma separated CPU IDs Example: CPU0,CPU1	Health monitoring is suppressed for devices in the list.

3.7.3.4 Storage Monitoring Policy

Storage monitoring policy is represented by the /systems/localhost/storage/1/policy target of NVSM CLI.

```
user@dgx-2:~$ sudo nvsm show /systems/localhost/storage/policy
```

Example output:

```
/systems/localhost/storage/policy
Properties:
    volume_mute_monitoring = <NOT_SET>
```

(continues on next page)

(continued from previous page)

```
    volume_poll_interval = 10
    drive_mute_monitoring = <NOT_SET>
    drive_mute_notification = <NOT_SET>
    drive_poll_interval = 10
    volume_mute_notification = <NOT_SET>
Verbs:
    cd
    set
    show
```

The properties for storage monitoring policy are described in the table below.

Property	Syntax	Description
drive_mute_notification	List of comma separated drive slots Example: 0, 1 etc	Email alert notification is suppressed for drives in the list.
drive_mute_monitoring	List of comma separated drive slots Example: 0, 1 etc	Health monitoring is suppressed for drives in the list.
drive_poll_interval	Positive integer	DSHM checks the health of the drives periodically. By default, this polling occurs every 10 seconds. The poll interval can be configured through this property.
volume_mute_notification	List of comma separated volume identifier Example: md0, md1 etc	Email alert notification is suppressed for volumes in the list
volume_mute_monitoring	List of comma separated volume identifier Example: md0, md1 etc	Health monitoring is suppressed for volumes in the list
volume_poll_interval	Positive integer	DSHM checks the health of the volumes periodically. By default, this polling occurs every 10 seconds. The poll interval can be configured through this property.

Storage volumes are identified by NVSM uniquely by their associated UUID. The mute monitoring for volume resources will hence use UUID instead of volume name. This is required for NVSM versions greater than 21.09.

Steps to identify the UUID of a volume to be set in mute monitoring and notification are listed below.

1. To get the list of volumes in the server run the below command:

```
# nvsm show volumes
```

```
# nvsm show volumes

/systems/localhost/storage/volumes/md0
Properties:
```

(continues on next page)

(continued from previous page)

```
    CapacityBytes = 1918641373184
    Encrypted = False
    Id = md0
    Name = md0
    Status_Health = OK
    Status_State = Enabled
    VolumeType = Mirrored
```

2. To find the UUID of a particular volume, run the below command. The command lists properties which contain the UUID for the volume with the name md0:

```
# mdadm --detail /dev/{volume name}
```

```
# mdadm --detail /dev/md0
/dev/md0:
           Version : 1.2
     Creation Time : Tue Feb 23 18:04:37 2021
        Raid Level : raid1
        Array Size : 1873673216 (1786.87 GiB 1918.64 GB)
     Used Dev Size : 1873673216 (1786.87 GiB 1918.64 GB)
      Raid Devices : 2
     Total Devices : 2
       Persistence : Superblock is persistent

     Intent Bitmap : Internal

       Update Time : Tue Apr 11 08:13:48 2023
             State : active
    Active Devices : 2
   Working Devices : 2
    Failed Devices : 0
     Spare Devices : 0

Consistency Policy : bitmap

              Name : dgx-20-04:0
              UUID : 3568aa82:dc3da8ac:5c17ea13:b04cf894
            Events : 78460

    Number   Major   Minor   RaidDevice State
       0     259        5        0      active sync   /dev/nvme2n1p2
       1     259       15        1      active sync   /dev/nvme3n1p2
```

3. Run the below command to set the UUID for mute monitoring:

```
# nvsm set /systems/localhost/storage/policy volume_mute_monitoring=<UUID>
```

```
# nvsm set /systems/localhost/storage/policy
volume_mute_monitoring=3568aa82:dc3da8ac:5c17ea13:b04cf894
```

4. Run the below command to set the UUID for mute notification:

```
# nvsm set /systems/localhost/storage/policy volume_mute_notification=<UUID>
```

```
# nvsm set /systems/localhost/storage/policy
volume_mute_notification=3568aa82:dc3da8ac:5c17ea13:b04cf894
```

5. Run the below command to verify that the policies were correctly set:

```
# nvsm show /systems/localhost/storage/policy
```

```
# nvsm show /systems/localhost/storage/policy
/systems/localhost/storage/policy
Properties:
    controller_mute_monitoring = <NOT_SET>
    controller_mute_notification = <NOT_SET>
    controller_poll_interval = 60
    drive_mute_monitoring = <NOT_SET>
    drive_mute_notification = <NOT_SET>
    drive_poll_interval = 60
    volume_mute_monitoring = 3568aa82:dc3da8ac:5c17ea13:b04cf894
    volume_mute_notification = 3568aa82:dc3da8ac:5c17ea13:b04cf894
    volume_poll_interval = 60
Targets:
Verbs:
    cd
    set
    show
```

3.7.3.5 Thermal Monitoring Policy

Thermal monitoring policy (for fan speed and temperature) is represented by the /chassis/localhost/thermal/policy target of NVSM CLI.

```
user@dgx-2:~$ sudo nvsm show /chassis/localhost/thermal/policy
```

Example output:

```
/chassis/localhost/thermal/policy
Properties:
    fan_mute_notification = <NOT_SET>
    pdb_mute_monitoring = <NOT_SET>
    fan_mute_monitoring = <NOT_SET>
    pdb_mute_notification = <NOT_SET>
Verbs:
    cd
    set
    show
```

The properties for thermal monitoring policy are described in the table below.

Property	Syntax	Description
fan_mute_notification	List of comma separated FAN IDs. Example: FAN2_R,FAN1_L,PDB_FAN2	Email alert notification is suppressed for devices in the list.
fan_mute_monitoring	List of comma separated FAN IDs Example: FAN6_F,PDB_FAN1	Health monitoring is suppressed for devices in the list.
pdb_mute_notification	List of comma separated PDB IDs. Example: PDB1,PDB2	Email alert notification is suppressed for devices in the list.
pdb_mute_monitoring	List of comma separated PDB IDs Example: PDB1	Health monitoring is suppressed for devices in the list.

3.7.3.6 Power Monitoring Policy

Power monitoring policy is represented by the /chassis/localhost/power/policy target of NVSM CLI.

```
user@dgx-2:~$ sudo nvsm show /chassis/localhost/power/policy
```

Example output:

```
/chassis/localhost/power/policy
Properties:
    mute_notification = <NOT_SET>
    mute_monitoring = <NOT_SET>

Verbs:
    cd
    set
    show
```

The properties for power monitoring policy are described in the table below.

Property	Syntax	Description
mute_notification	List of comma separated PSU IDs. Example: PSU4,PSU2	Email alert notification is suppressed for devices in the list.
mute_monitoring	List of comma separated FAN IDs Example: PSU1,PSU4	Health monitoring is suppressed for devices in the list.

3.7.3.7 PCIe Monitoring Policy

Memory monitoring policy is represented by the /systems/localhost/pcie/policy target of NVSM CLI.

```
:~$ sudo nvsm show /systems/localhost/pcie/policy
```

Example output:

```
/systems/localhost/pcie/policy
Properties:
```

(continues on next page)

(continued from previous page)

```
    mute_notification = <NOT_SET>
    mute_monitoring = <NOT_SET>

Verbs:
    cd
    set
    show
```

The properties for memory monitoring policy are described in the table below.

Property	Syntax	Description
mute_notification	List of comma separated PCIe IDs	Email alert notification is suppressed for devices in the list.
mute_monitoring	List of comma separated PCIe IDs	Health monitoring is suppressed for devices in the list.

3.7.3.8 GPU Monitoring Policy

Memory monitoring policy is represented by the /systems/localhost/gpus/policy target of NVSM CLI.

```
:~$ sudo nvsm show /systems/localhost/gpus/policy
```

Example output:

```
/systems/localhost/gpus/policy
Properties:
    mute_notification = <NOT_SET>
    mute_monitoring = <NOT_SET>

Verbs:
    cd
    set
    show
```

The properties for memory monitoring policy are described in the table below.

Property	Syntax	Description
mute_notification	List of comma separated GPU IDs	Email alert notification is suppressed for devices in the list.
mute_monitoring	List of comma separated GPU IDs	Health monitoring is suppressed for devices in the list.

3.7.3.9 Network Adapter Monitoring Policies

3.7.3.9.1 Network Adapter Policy

The physical network adapter monitoring policy is represented by the `/chassis/localhost/NetworkAdapters/policy` target of the NVSM CLI.

```
:~$ sudo nvsm show /chassis/localhost/NetworkAdapters/policy
```

Example output:

```
/chassis/localhost/NetworkAdapters/policy
Properties:
    mute_notification = <NOT_SET>
    mute_monitoring = <NOT_SET>
Verbs:
    cd
    set
    show
```

The properties are described in the following table.

Property	Syntax	Description
mute_notification	List of comma separated physical network adapter IDs.	Email alert notification is suppressed for devices in the list.
mute_monitoring	List of comma separated physical network adapter IDs.	Health monitoring is suppressed for devices in the list.

The mute monitoring is assigned by using the Physical Adapter name and not the logical name. To get the physical adapter name use the command:

```
$ sudo nvsm show /chassis/localhost/NetworkAdapters
```

This command will display a list of target adapter names as shown below:

```
:~$:/etc/nvsm/platforms# sudo nvsm show /chassis/localhost/NetworkAdapters
/chassis/localhost/NetworkAdapters
Targets:
PCI0000_0c_00
PCI0000_12_00
PCI0000_4b_00
PCI0000_54_00
PCI0000_8d_00
PCI0000_94_00
PCI0000_ba_00
PCI0000_cc_00
PCI0000_e1_00
PCI0000_e2_00
```

Note: Use these adapter names to assign monitoring policies.

Here is an example that uses the `PCI0000_0c_00` network interface:

```
:~$ sudo nvsm show /chassis/localhost/NetworkAdapters/PCI0000_0c_00/NetworkPorts/
↪policy
```

Example output:

```
/chassis/localhost/NetworkAdapters/PCI0000_0c_00/NetworkPorts/policy
Properties:
    mute_notification = <NOT_SET>
    mute_monitoring = <NOT_SET>
Verbs:
    cd
    set
    show
```

The properties are described in the following table.

Property	Syntax	Description
mute_notification	List of comma separated physical network port IDs.	Email alert notification is suppressed for devices in the list.
mute_monitoring	List of comma separated physical network port IDs	Health monitoring is suppressed for devices in the list.

3.7.3.9.2 Network Devices Functions Policy

The network devices functions monitoring policy is represented by the `/chassis/localhost/NetworkAdapters/<network-id>/NetworkDeviceFunctions/policy` target of NVSM CLI.

The following command uses the `PCI0000_0c_00` network port to demonstrate this command.

```
:~$ sudo nvsm show /chassis/localhost/NetworkAdapters/PCI0000_0c_00/
↪NetworkDeviceFunctions/policy
```

Example output:

```
/chassis/localhost/NetworkAdapters/PCI0000_0c_00/NetworkDeviceFunctions/policy
Properties:
    mute_monitoring = <NOT_SET>
    mute_notification = <NOT_SET>
    rx_collision_threshold = 5
    rx_crc_threshold = 5
    tx_collision_threshold = 5
Verbs:
    cd
    set
    show
```

The properties are described in the following table.

Property	Syntax	Description
mute_notification	List of comma separated network-centric PCIe function IDs. Example: PSU4,PSU2	Email alert notification is suppressed for devices in the list.
mute_monitoring	List of comma separated network-centric PCIe function IDs. Example: PSU1,PSU4	Health monitoring is suppressed for devices in the list.
rx_collision_threshold	Positive integer	
rx_crc_threshold	Positive integer	
tx_collision_threshold	Positive integer	

3.7.3.9.3 Muting NetworkPort Link Down Alerts

By default, NVSM creates an NV-NET-01 alert for each network adapter if its link is down. If certain network adapters are not used, their alerts can be muted using the nvsm muteNIC command. When running the command without parameters, it displays the configuration for all network adapters.

Example output:

```
# sudo nvsm muteNIC
NIC        | Interface     | Status
ibp220s0   | PCI0000_dc_00 | muted
ibp41s0f1  | PCI0000_29_00 | muted
ibp94s0    | PCI0000_5e_00 | default
ibp64s0    | PCI0000_40_00 | default
ibp170s0f0| PCI0000_aa_00 | default
ibp79s0    | PCI0000_4f_00 | default
ens6f0     | PCI0000_82_00 | default
ibp24s0    | PCI0000_18_00 | default
ibp154s0   | PCI0000_9a_00 | default
eno3       | PCI0000_0b_00 | default
ibp41s0f0  | PCI0000_29_00 | default
ibp170s0f1| PCI0000_aa_00 | default
ens6f1     | PCI0000_82_00 | default
ibp192s0   | PCI0000_c0_00 | default
ibp206s0   | PCI0000_ce_00 | default
```

To mute the link down alerts on the specified network adapters, use the command:

```
sudo nvsm muteNIC <NIC1>=enable,<NIC2>=enable,...
```

For example:

```
sudo nvsm muteNIC ibp220s0=enable,ibp41s0f1=enable
```

To unmute the link down alerts on the specified network adapters, use the command:

```
sudo nvsm muteNIC <NIC1>=disable,<NIC2>=disable,...
```

For example:

```
sudo nvsm muteNIC ibp220s0=disable,ibp41s0f1=disable
```

Additionally, it is necessary to clear the alert database to see link down alerts again. Use the following command:

```
sudo nvsm_database_rotate.sh -d alert
```

This command stops NVSM, clears the alert database, and then restarts it.

3.8. Performing System Management Tasks

This section describes commands for accomplishing some system management tasks.

3.8.1. Rebuilding a RAID/ESP Array for Current NVSM

On DGX systems, cache drives are configured as a RAID 0 array by default. This volume is mounted to `/raid`. In the example below, it shows as `/dev/md1`, but the name can be different depending on the OS naming schema and configuration.

Additionally for DGX systems with two NVMe OS drives, echo OS drive have two partitions:

- The second partitions are configured as a RAID 1 array with the operating system installed. In the examples below, it shows as `/dev/md0`.
- The first partition is known as the **EFI System Partition (ESP)**. NVSM monitors the content of this partition from both drives. If one of the ESP is corrupted, NVSM can be used to recover that partition from the healthy ESP.

Note: This is not a RAID array, because UEFI does not support booting from software raid volumes.

3.8.1.1 Viewing a Healthy RAID/ESP Volume

On a healthy system, the OS volume appears with `VolumeType = Mirrored` and `Status_Health = OK`. For example:

```
nvsm(/systems/localhost/storage)-> show volumes/md0

/systems/localhost/storage/volumes/md0
Properties:
    CapacityBytes = 1918641373184
    Encrypted = False
    Id = md0
    Name = md0
    Status_Health = OK
    Status_State = Enabled
    VolumeType = Mirrored
Targets:
```

(continues on next page)

(continued from previous page)

```
Verbs:
    cd
    show
```

The cache volume appears with `VolumeType = NonRedundant` and and `Status_Health = OK`. For example:

```
nvsm-> cd /systems/localhost/storage
nvsm(/systems/localhost/storage)-> show volumes/md1

/systems/localhost/storage/volumes/md1
Properties:
    CapacityBytes = 30724962910208
    Encrypted = False
    Id = md1
    Name = md1
    Status_Health = OK
    Status_State = Enabled
    VolumeType = NonRedundant
Targets:
    encryption
Verbs:
    cd
    show
```

The ESP volume appears with `VolumeType = EFI` system partition and `Status_Health = OK`. The name of the ESP volume varies per system; you can use the command **nvsm show volumes** to list all volumes and look for `VolumeType = EFI` system partition. Here's the example from DGX A100:

```
nvsm(/systems/localhost/storage)-> show volumes

...

/systems/localhost/storage/volumes/nvme2n1p1
Properties:
    CapacityBytes = 536870912
    Encrypted = False
    Id = nvme2n1p1
    Name = nvme2n1p1
    Status_Health = OK
    Status_State = Enabled
    VolumeType = EFI system partition

...

/systems/localhost/storage/volumes/nvme3n1p1
Properties:
    CapacityBytes = 536870912
    Encrypted = False
    Id = nvme3n1p1
    Name = nvme3n1p1
    Status_Health = OK
    Status_State = StandbyOffline
    VolumeType = EFI system partition

Targets:
```

(continues on next page)

(continued from previous page)

```
Verbs:
    cd
    show
```

3.8.1.2 Viewing a Degraded RAID/ESP Volume

On a system with degraded OS volume, the md0 volume will appear with only one drive, with the following `Status_Health = Critical` message:

```
nvsm-> cd /systems/localhost/storage
nvsm(/systems/localhost/storage)-> show volumes/md0

/systems/localhost/storage/volumes/md0
Properties:
    CapacityBytes = 1918641373184
    Encrypted = False
    Id = md0
    Name = md0
    Status_Health = Critical
    Status_State = Enabled
    VolumeType = Mirrored
Targets:
Verbs:
    cd
    show
```

On a system with corrupted ESP, the volume will appear with the following `Status_Health = Critical` and `Status_State = UnavailableOffline` messages:

```
nvsm-> cd /systems/localhost/storage
nvsm(/systems/localhost/storage)-> show volumes/nvme2n1p1

/systems/localhost/storage/volumes/nvme2n1p1
Properties:
    CapacityBytes = 536870912
    Encrypted = False
    Id = nvme2n1p1
    Name = nvme2n1p1
    Status_Health = Critical
    Status_State = UnavailableOffline
    VolumeType = EFI system partition
Targets:
Verbs:
    cd
    show
```

3.8.1.3 Rebuilding the RAID/ESP Volume

To rebuild the RAID/ESP volume, make sure that you have replaced failed NVMe drives.

The RAID rebuilding process should begin automatically upon turning on the system. If it does not start automatically, use NVSM CLI to manually rebuild the array as follows.

1. Start an NVSM CLI interactive session and switch to the storage target.

```
~$ sudo nvsm
nvsm-> cd /systems/localhost/storage
```

2. Start the rebuilding process, and select which volumes to rebuild.
   - **raid-1** for OS volume
   - **raid-0** for cache volume
   - **esp** for EFI system partition

   For raid-1 volume, you also need to enter the replaced drive name.

   Note: This is not the partition name. For example, use nvme3 instead of nvme3n1p2.

```
nvsm(/systems/localhost/storage)-> start volumes/rebuild

PROMPT: In order to rebuild volume, volume type is required. Please
        specify the volume type to rebuild from options below.
        raid-0: create raid-0 data volume
        raid-1: rebuild OS boot and root volumes
        esp:    find and replicate an empty EFI system partition

Type of volume rebuild (CTRL-C to cancel): raid-1

PROMPT: In order to rebuild this volume, a spare drive
        is required. Please specify the spare drive to
        use to rebuild RAID-1.

Name of spare drive for RAID-1 rebuild (CTRL-C to cancel): nvme3

WARNING: Once the rebuild process is started, the
         process cannot be stopped.

Start RAID-1 rebuild? [y/n] y
```

3. After entering **y** at the prompt to start the RAID 1 rebuild, the “Initiating rebuild ...” message appears.

```
/systems/localhost/storage/volumes/rebuild started at 2023-04-10 Initiating RAID-
↪1 rebuild on volume md0...
 0.0% [\ ]
```

4. After a few seconds, the “Rebuilding RAID-1 ...” message appears.

```
/systems/localhost/storage/volumes/rebuild started at 2023-04-10 08:22:58.910025
Rebuilding RAID-1...
 31.0% [=============/ ]
```

5. If this message remains at Initiating RAID-1 rebuild for more than 30 seconds, there is a problem with the rebuild process. Verify that the name of the replacement drive is correct and try again.

The RAID 1 rebuild process should take about 1 hour to complete.

For more detailed information on replacing a failed NVMe drive, see the NVIDIA DGX-2 Service Manual or NVIDIA DGX A100 Service Manual.

3.8.2. Rebuilding a RAID 1 Array for Legacy NVSM (< 21.09)

For DGX systems with two NVMe OS drives configure as a RAID 1 array, the operating system is installed on volume `md0`. You can use NVSM CLI to view the health of the RAID volume and then rebuild the RAID array on two healthy drives.

3.8.2.1 Viewing a Healthy RAID Volume

On a healthy system, this volume appears with two drives and `Status_Health = OK`. For example:

```
 nvsm-> cd /systems/localhost/storage
nvsm(/systems/localhost/storage)-> show volumes/md0
/systems/localhost/storage/volumes/md0
Properties:
    Status_State = Enabled
    Status_Health = OK
    Name = md0
    Encrypted = False
    VolumeType = RAID-1
    Drives = [ nvme0n1, nvme1n1 ]
    CapacityBytes = 893.6G
    Id = md0
Targets:
    rebuild
Verbs:
    cd
    show
```

3.8.2.2 Viewing a Degraded RAID Volume

On a system with degraded OS volume, the md0 volume will appear with only one drive, with the following `Status_Health = Warning`, and `Status_State = Degraded` messages:

```
nvsm-> cd /systems/localhost/storage
nvsm(/systems/localhost/storage)-> show volumes/md0
/systems/localhost/storage/volumes/md0
Properties:
    Status_State = Degraded
    Status_Health = Warning
    Name = md0
    Encrypted = False
    VolumeType = RAID-1
```

(continues on next page)

(continued from previous page)

```
    Drives = [ nvme1n1 ]
    CapacityBytes = 893.6G
    Id = md0Targets:
    rebuild
Verbs:
    cd
    show
```

In this situation, the OS volume is missing its parity drive.

3.8.2.3 Rebuilding the RAID 1 Volume

To rebuild the RAID array, make sure that you have installed a known good NVMe drive for the parity drive.

The RAID rebuilding process should begin automatically upon turning on the system. If it does not start automatically, use NVSM CLI to manually rebuild the array as follows.

1. Start an NVSM CLI interactive session and switch to the storage target.

   ```
   $ sudo nvsm
   nvsm-> cd /systems/localhost/storage
   ```

2. Start the rebuilding process and be ready to enter the device name of the replaced drive.

   ```
   nvsm(/systems/localhost/storage)-> start volumes/md0/rebuild
   PROMPT: In order to rebuild this volume, a spare drive
           is required. Please specify the spare drive to use
           to rebuild md0.
   Name of spare drive for md0 rebuild (CTRL-C to cancel): nvmeXn1
   WARNING: Once the volume rebuild process is started, the
            process cannot be stopped.
   Start RAID-1 rebuild on md0? [y/n] y
   ```

3. After entering **y** at the prompt to start the RAID 1 rebuild, the “Initiating rebuild ...” message appears.

   ```
   /systems/localhost/storage/volumes/md0/rebuild started at 2018-10-12 15:27:26.
   ↪525187
    Initiating RAID-1 rebuild on volume md0...
    0.0% [\ ]
   ```

   After about 30 seconds, the `Rebuilding RAID-1 ...` message should appear.

   ```
   /systems/localhost/storage/volumes/md0/rebuild started at 2018-10-12 15:27:26.
   ↪525187
    Rebuilding RAID-1 rebuild on volume md0...
    31.0% [=============/ ]
   ```

   If this message remains at `Initiating RAID-1 rebuild` for more than 30 seconds, there is a problem with the rebuild process. Verify that the name of the replacement drive is correct and try again.

The RAID 1 rebuild process should take about 1 hour to complete.

For more detailed information on replacing a failed NVMe OS drive, see the NVIDIA DGX-2 Service Manual or NVIDIA DGX A100 Service Manual.

3.8.3. Setting MaxQ/MaxP on DGX-2 Systems

Beginning with DGX OS 4.0.5, you can set two GPU performance modes – MaxQ or MaxP.

Note: Support on DGX-2 systems requires BMC firmware version 1.04.03 or later. MaxQ/MaxP is not supported on DGX-2H systems.

3.8.3.1 MaxQ

- Maximum efficiency mode
- Allows two DGX-2 systems to be installed in racks that have a power budget of 18 kW.
- Switch to MaxQ mode as follows.

```
$ sudo nvsm set powermode=maxq
```

The settings are preserved across reboots.

3.8.3.2 MaxP

- Default mode for maximum performance
- GPUs operate unconstrained up to the thermal design power (TDP) level.

  In this setting, the maximum DGX-2 power consumption is 10 kW.
- Provides reduced but better performance than MaxQ when only 3 or 4 PSUs are working.
- If you switch to MaxQ mode, you can switch back to MaxP mode as follows:

```
$ sudo nvsm set powermode=maxp
```

The settings are preserved across reboots.

3.8.4. Performing a Stress Test

NVSM supports functionality to simultaneously stress various components (GPU, PCIe, DIMMs, Storage Drives, CPUs, Network Cards) of the system with large workloads. The stress-test will provide a summary at the end determining whether each stressed component passed the test or failed with some error. NVSM will also monitor various system metrics during the stress-test to provide a clearer picture of the kinds of computational loads imposed. This stress test can be invoked from the CLI.

Syntax:

```
$ sudo nvsm stress-test [--usage] [--force] [--no-prompt] [<test>...] [DURATION]
```

For help on running the test, issue the following.

```
$ sudo nvsm stress-test --usage
```

Example output for `sudo nvsm stress-test 60 --force`:

```
swqa@ubuntu-luna2:~$ sudo nvsm stress-test 60 --force

Initializing NVSM Core...

############################## NVSM STRESS ##############################

WARNING:
These stress tests are potentially disruptive and can interfere with other jobs you might be running.
Make sure to be running as few jobs as possible before proceeding.
(Pass --no-prompt option in command next time to suppress this prompt)
If running GPU Stress Test, expect test to take longer than countdown duration.
The NIC (network interface card) Stress Test is only available on DGX-1, DGX-2, and Luna Systems.
If it is run on non-supported platforms, results for this component will be blank.

Are you sure you want to proceed [y/n]? y

Stressing GPU : Multiplying matrices on GPUs
Stressing CPU : Multiplying matrices on CPUs
Stressing Memory : Mapping pages into memory
Stressing Storage : Performing disk I/O operations
Stressing Network Cards : Performing loopback bandwidth stress


    Fetching stream progress from backend...

    100% Complete...


GPU             PASS
CPU             PASS
Memory          PASS
Storage         PASS
PCIe            PASS
NIC             SKIPPED

Component       PreTest         Min             Max             Avg             Unit

GPU_TEMP        32.5            32.5            32.5            32.5            C
CPU_TEMP        52.5            52.5            52.5            52.5            C
MEM_TEMP        32.75           32.75           32.75           32.75           C
POWERDRAW       4134            4134            4134            4134            W
FANSPEED        6289.55         6289.55         6289.55         6289.55         RPM
GPU_LOAD        0               0               100.00          50.00           %
CPU_LOAD        36.89           36.89           36.89           36.89           %
MEM_LOAD        0.43            0.43            0.43            0.43            %
DISK_LOAD       0.12            0.12            0.12            0.12            %

Stress Test Log:/var/nvsmlog/nvsm/StressTestLog2021-08-03T01:52:22-04:00.nvsmlog
```

Chapter 4. Configuring NVSM Security

This chapter explains shows how to secure the NVSM API installation.

4.1. Overview of NVSM Security

NVSM APIs are served using the HTTPS protocol. HTTPS requires the NVSM API server to possess a public-private key pair as well as a certificate that it presents to connecting clients. The certificate also needs to be signed by a certificate authority (CA) using the private key of that CA.

For proper security, this certificate+key should be provided by users. It cannot be provided by NVIDIA because

- The private key should be known only to the user, and should not be known to NVIDIA, and
- NVIDIA is not a Certificate Authority

To allow the NVSM software stack to work right out of the box, the installation process creates some sample key pairs and certificates. These certificates are created with dummy values for country, organization, organization unit, etc. because the installation does not include these details. Also, the generated CA certificate is self signed. These sample certificates must NOT be used in a production environment.

NVSM allows you to provide your own key-pairs and certificates with correct values that are properly signed by a trusted CA. Details of key generation and certificate chains is beyond the scope of this document. However, an example setup is shown below to show how NVSM can be configured with customer provided/generated keys and certificates.

4.2. What You Need to Configure NVSM Security

To configure NVSM security, you need the following, either copied from a CA provider or generated locally and copied to a location on the system.

- X.509 certificate for the NVSM REST server

  Example path and filename: `/pki/node1.crt`
- Private key file corresponding to the above certificate

  Example path and filename: `/pki/node1.key`

- The certificate of the CA who issued the above certificate

  Example path and filename: `/pki/ca.crt`

An explanation of how to generate or obtain these certificates and keys is beyond the scope of this document since these have to be in compliance with the overall security architecture of the data center. In the most simplest form, users might use commands such as OpensSSL to generate their own certificate chain and keys. You may prefer to use free services such as https://letsencrypt.org/ to acquire them.

4.3. How to Configure NVSM Security

1. Edit the NVSM configuration file to use the paths and filenames of your certificate files and key file.

   Edit the `ca_cert`, `https_cert`, and `https_priv_key` configuration parameters to specify the path and filenames that NVSM shall use. The following use the example path and filenames.

   ```
   "ca_cert":"/pki/ca.crt",
   ```

   ```
   "https_cert":"/pki/node1.crt",
   ```

   ```
   "https_priv_key":"/pki/node1.key",
   ```

2. Restart the NVSM service.

   ```
   $ sudo systemctl restart nvsm
   ```

Chapter 5. NVSM Call Home

The NVIDIA System Manager (NVSM) Call Home, when enabled and with an internet connection, provides additional automation to NVSM health monitoring functionality. Instead of having to contact NVIDIA Enterprise Support to report critical alerts from NVSM, submit system logs, nvsm dump health files, and DGX serial numbers to create a support ticket, NVSM Call Home automates those tasks. This reduces overall turnaround time for resolving issues.

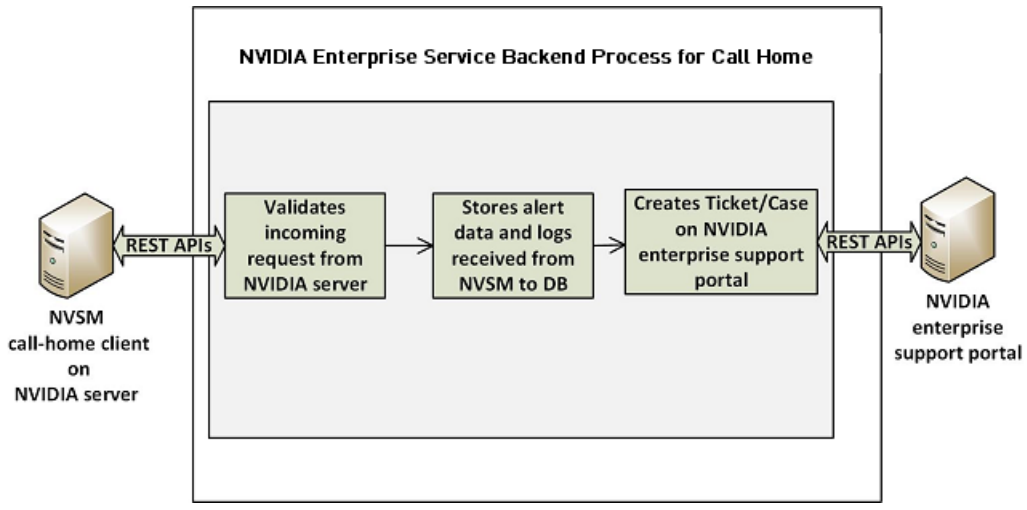
5.1. NVSM Call Home Overview

When NVSM raises a critical alert, NVSM Call Home performs the following actions:

- Proactively pushes **Critical** level alerts to NVIDIA Enterprise Services.
- Collects the system and nvsm dump, and system serial number, and uploads them to NVIDIA Enterprise Services..
- Creates a Support Case on the Enterprise Support portal.

NVSM Call Home also clears resolved alerts and pushes the updated status to NVIDIA Enterprise Services.

The figure below illustrates the end-to-end flow of the NVSM Call Home feature.



When NVSM Call Home is enabled on the DGX system and a critical alert is raised, the NVSM daemon on the DGX system initiates an HTTPS connection to the secured NVIDIA Enterprise Services backend and communicates the alert details and logs using the RESTful interface. The information is validated

and then a new Support Ticket is created on the NVIDIA Enterprise Support Portal. Communication of all alerts, including status changes for the alerts, is through REST calls.

NVSM Call Home operates in three different modes. It is not enabled by default, so to use NVSM Call Home you must enable one of the following modes:

5.1.1. Policy-enabled “automatic” Mode

- This mode batches alert submissions at regular intervals and then pushes them to the NVIDIA Enterprise Support portal.
- An internet connection is required.
- To enable, issue

```
$ sudo nvsm set /policy callhome_enable=true
```

See the section *Using NVSM Call Home in Automatic Mode* for details as well as configuration options for automatic mode.

5.1.2. Policy-enabled Offline Mode

- This mode is useful for air-gapped or highly-secured environments where access to the internet is limited.
- Instead of sending batched submissions to NVIDIA Enterprise Support, the alert and system information are stored on the local system. Users need to manually provide the gathered information to NVIDIA Enterprise Services to create a support case.
- To enable, issue

```
$ sudo nvsm set /policy offline_callhome_enable=true
```

See the section *Using NVSM Offline Call Home* for details as well as configuration options for offline mode.

5.1.3. On-demand Mode

- This mode gathers alert information and creates a submission on-the-fly to the Enterprise Support Portal.
- An internet connection is required.
- To initiate a Call Home submission on-demand, issue

```
$ sudo nvsm set /callhome trigger=true
```

See the section *Using NVSM On-Demand Mode* for details as well as configuration options for on-demand mode.

5.2. Using NVSM Call Home

NVSM Call Home can operate in two modes:

- **Automatic Mode** - NVSM Call Home operates automatically at regular intervals.
- **On-demand Mode** - NVSM Call Home sequence is initiated manually.

You can also set up NVSM Call Home to run offline; for example, on air-gapped systems.

5.2.1. Prerequisites for Using NVSM Call Home in Automatic or On-Demand Mode

5.2.1.1 Enabling Ports

Since NVSM Call Home communicates with the external NVIDIA server, port 443 must be enabled prior to operating NVSM Call Home.

5.2.1.2 Enabling Access

You need to register your system for NVSM Call Home so that the NVIDIA Services Cloud recognizes the system. Contact NVIDIA Enterprise Services to set up NVIDIA Call Home for your DGX system.

5.2.1.3 Validating NVSM Call Home Readiness

Before using NVSM Call Home, make sure the server is ready to support NVSM Call Home by performing a diagnostic test. The test does not create a ticket with NVIDIA Enterprise Services, but does test that the system is able to communicate with the NVIDIA Enterprise Services infrastructure.

To run the diagnostic, issue the following:

```
$ sudo nvsm set /callhome trigger=true diagtest=true
```

Note: This uses the on-demand mode of NVIDIA Call Home, explained in more detail in the section *Using NVSM On-demand Mode*.

To see the result of the last diagnostic test run, issue the following:

```
$ sudo nvsm show /callhome
```

Example output confirming the setup is ready for Call Home operation. Lines of interest are identified in bold.

```
/callhome
Properties:
    Trigger = False
    Op_Description = User initiated call home operation.
    Op_DiagTest = True
```

(continues on next page)

(continued from previous page)

```
    Op_CaseId = none
    Op_State = Succeeded
    Op_StartTime = 2019-06-24T06:10:17Z
    Op_Message = Call Home operation Succeeded
    Op_Email =
```

If the output reports errors or failures, contact NVIDIA technical support for assistance.

5.2.2. Using NVSM Call Home in Automatic Mode

When automatic mode is enabled, NVSM monitors the server continuously and pushes critical or cleared alerts to NVESC and creates a support case on behalf of the registered user.

Automatic Mode Syntax

To enable automatic mode, first configure the email contact.

```
$ sudo nvsm set /policy callhome_email_contact=<email>
```

then enable Call Home.

```
$ sudo nvsm set /policy callhome_enable=true [callhome_batch_interval=<time-in-
↪seconds>]
```

You can also configure the email contact and enable Call Home in the same command.

```
$ sudo nvsm set /policy callhome_email_contact=<email> callhome_enable=true [callhome_
↪batch_interval=<time-in-seconds>]
```

Automatic Mode Configuration Arguments

Configure NVSM Call Home using the following parameters:

- `callhome_email_contact`

  Sets the email-id. This should be a registered user of the NVIDIA Enterprise Support Portal. The email gets embedded in the case/ticket created in NVIDIA Enterprise Support Portal.

- `callhome_batch_interval`

  (Optional) Enabling automatic mode batches alert submissions at regular intervals and then pushes them to the NVIDIA Enterprise Support portal. Any raised alerts within that time frame will be sent (as individual Support Cases). By default, the interval is 600 seconds (10 minutes), but you can use this option to specify other intervals (in seconds).

Automatic Mode Example

The following example illustrates how to use these parameters.

```
$ sudo nvsm set /policy callhome_email_contact=123@example.com callhome_enable=true
↪callhome_batch_interval=610
```

Verifying Automatic Mode Status

To verify the status of the current setup, issue the following.

```
$ sudo nvsm show /policy
```

Example output:

```
/policy
Properties:
   callhome_batch_interval = 610
   callhome_email_contact = 123@example.com
   callhome_enable = True
   email_recipients =
   email_sender =
   email_smtp_server_name =
   email_smtp_server_port = 0
```

`callhome_enable = True` indicates that Call Home automatic mode is enabled.

Disabling Automatic Mode

The Call Home automatic mode will start listening for alerts and raise support cases in the background. If there are any maintenance activities such as reseating or swapping components that would cause NVSM to generate critical alerts, Call Home will raise support cases as well.

To avoid raising support cases during intentional maintenance activities, disable call-home by issuing the following.

```
$ sudo nvsm set /policy callhome_enable=false
```

5.2.3. Using NVSM On-Demand Mode

NVSM Call Home On-Demand mode is a user-triggered call-home action. Triggering Call Home on-demand creates a Support Case with NVIDIA Enterprise Support Portal that includes a captured system dump ('nvsm dump health'). NVSM Call Home On-Demand can be used whether or not automatic mode is enabled.

On-Demand Mode Syntax

To trigger an NVSM Call Home sequence on-demand, issue the following.

```
# sudo nvsm set /callhome trigger=true [description="<description>"] [email=<email>]
```

To cancel an on-demand Call Home in progress, issue the following.

```
# sudo nvsm set /callhome trigger=false
```

See the next section for an explanation of the optional parameters.

On-Demand Mode Configuration Options

You can configure NVSM Call Home triggered on-demand using the following parameters:

- `email`

  This option sets an email-id. The email gets embedded in the case/ticket created in NVIDIA Enterprise Support Portal.

- `description`

  This option lets you describe the purpose or the details for triggering On-Demand Call Home.

  Examples of descriptive strings:

  `"Testing"`

```
    "System running low in performance, takes several minutes to peform an
    nvidia-smi command."
```

On-Demand Example

The following example illustrates how to use these optional parameters.

```
# sudo nvsm set /callhome trigger=true description="testing" email=123@example.com
```

Verifying On-Demand Status

To check the status of a Call Home sequence initiated on-demand, issue the following.

```
# sudo nvsm show /callhome
```

The following example output shows the progress of the Call Home sequence.

```
/callhome
Properties:
   Trigger = True
   Op_Description = testing
   Op_CaseId = none
   Op_State = Running
   Op_StartTime = 2019-06-12T08:28:45Z
   Op_Message = Collecting logs
   Op_Email = 123@example.com
```

The following example output shows that a case ID was created in the NVIDIA Enterprise Support portal.

```
/callhome
Properties:
   Trigger = False
   Op_Description = testing
   Op_CaseId = 0001XXX
   Op_State = Succeeded
   Op_StartTime = 2019-06-12T08:28:45Z
   Op_Message = Call Home operation Succeeded
   Op_Email = 123@example.com
```

`Trigger = False` indicates that the on-demand sequence is not running - in this case because it has completed.

5.2.4. Using NVSM Offline Call Home

To support DGX systems installed in air-gapped or highly-secured environments where access to the internet is limited, NVSM Call Home can be operated in offline mode (Offline Call Home). Like standard Call Home, Offline Call Home software proactively monitors the health of the DGX system and automatically

- Collects system dump and logs, and
- Collects alerts and system information.

However, instead of sending the information to NVIDIA Enterprise Services, NVSM Offline Call Home stores the information in a user-specified directory on the DGX system. Also, unlike standard Call Home, Offline Call Home operates in automatic mode only; there is no on-demand mode in Offline Call Home.

Prerequisites

Offline Call Home and standard Call Home cannot be enabled at the same time. To ensure that standard automatic-mode Call Home is not enabled, issue the following before enabling Offline Call Home.

```
$ sudo nvsm set /policy callhome_enable=false
```

Enabling Offline Call Home

Like standard Call Home, use the `nvsm set /policy` command to enable Offline Call Home.

```
$ sudo nvsm set /policy offline_callhome_enable=true \
offline_callhome_dump_destination_location=<path/to/location> \
offline_callhome_batch_interval=<batch-interval> \
offline_callhome_no_of_dumps_allowed=<number>
```

Offline Call Home Configuration Options

- `offline_callhome_dump_destination_location`

  By default, Offline Call Home stores the system logs at `/var/log/nvsm_offline_callhome`. You can set a different location using this option.

- `offline_callhome_batch_interval`

  Enabling Offline Call Home creates a batch of alerts at regular intervals and then pushes them to local storage. By default, the interval is 600 seconds (10 minutes), but you can use this option to specify other intervals (in seconds).

- `offline_callhome_no_of_dumps_allowed`

  By default, NVSM Offline Call Home will store 9999999 different log files, but you can specify a smaller number for which to allocate space as needed.

Example of Enabling Offline Call Home

The following example illustrates how to use these parameters.

```
$ sudo nvsm set /policy \
offlinecallhome_enable=true \
offline_callhome_dump_destination_location=/tmp/offline_callhome_dump \
callhome_batch_interval=610 \
offline_callhome_no_of_dumps_allowed=10
```

Verifying the Offline Call Home Configurationn

To verify the status of the current setup, issue the following.

```
$ sudo nvsm show /policy
```

Example output showing the offline Call Home policy details.

```
/policy
Properties:
    offline_callhome_batch_interval = 610
    offline_callhome_enable = True
    offline_callhome_dump_destination_location = /tmp/offline_callhome_dump
    offline_callhome_no_of_dumps_allowed = 10
```

Verifying Contents of the Dump File

The contents of each batch is stored in a tar file.

- Tar file naming format:

  `offlinecallhome-nvsm-health_<timestamp>_<serial-number>.tar.xz`

- The tar file contents -
  - System dump
  - JSON metadata file that lists the system and critical alerts.

    JSON file naming format:

    `offlinecallhome_notifications_<timestamp>_<hostname>_<serial-number>.json`

- JSON file format showing type of data included :

```
{
    "system_serial": "<serial number>",
    "system_name": "<hostname>",
    "notifications": [
        {
            "alert_id": "",
            "clear_time": "-",
            "component_id": "",
            "description": "",
            "event_time": "",
            "message": "",
            "message_details": ".",
            "recommended_action": "",
            "severity": "",
            "system_name": "",
            "system_serial": "",
            "type": ""
        }
    ]
}
```

Chapter 6. NVSM Multinode

NVSM Multinode is developed to monitor multiple compute nodes within a cluster. It serves as a single management interface with active monitoring and alerting for a cluster of nodes. It also supports features including cluster wide health with drill down capabilities, and dump collection.

6.1. NVSM Multinode Overview

Aggregator Node - Acts as the central coordinator, running an MQTT server to communicate with compute nodes. It is deployed as a container on an external management server. NVSM instances run for each compute node.

Compute Nodes - NVSM running on DGX systems connect to the aggregator node's MQTT server.

6.2. Prerequisites

System Requirements

- Ubuntu-based management server, can be an external or DGX system.
- Network connectivity between the aggregator node and compute nodes.

Software Requirements

- Ensure Docker (minimum version 20.10.21) and Docker Compose (compatible version) are installed on the system.
- NVSM multinode supported version (24.03.05) onwards should be installed on compute nodes.
- Ensure the aggregator node and all compute nodes time are synced to NTP. Since compute nodes and aggregator nodes connect over MQTT, the timestamp on MQTT message needs to be in sync.

Note: If compute node time and aggregator time is not synchronized then compute node MQTT messages will be assumed stale and NVSM CLI commands will error out: `ServiceUnavailable`. Aggregator and all compute nodes must set up time synchronization using services like ntpdate or chrony.

6.3. Setup Details

Each cluster includes an Aggregator Node (the Container) that runs on x86 Ubuntu servers, and multiple Compute Nodes (DGX Systems). An Aggregator Node system must have access to the host network and management network (BMC). Compute Nodes connect to the Aggregator Node over the MQTT server, hosted by the Aggregator Node. NVSM running on Compute Nodes connect to NVSM also running on the Aggregator Node. The Aggregator Node also includes an NVSM exporters dashboard, serving as a single interface to view all connected nodes, sensor data, health, etc.

6.3.1. Security Warning: Docker Group Access Risks

When deploying the multinode NVSM aggregator, it runs as the root user inside a Docker container. However, due to Docker's security model, any user who is a member of the Docker group on the host system can execute commands inside any running container, including the NVSM aggregator. This means that, unlike the single-node case where only the root user could run NVSM commands, non-root users who belong to the Docker group on the control plane (baremetal) node can also access and execute NVSM commands within the container. Administrators should be aware that this is a fundamental property of Docker membership in the Docker group effectively grants root-level access to all containers. To maintain security, it is essential to restrict Docker group membership to trusted administrators only and to secure the control plane node accordingly, as this node becomes a critical point of control for the entire cluster.

6.3.2. Packages

- **Aggregator** - Aggregator image contains a docker container pre-packaged with NVSM, MQTT server and nvsm-exporter stack.
- **Node Provisioner** - Node provisioner image contains ansible playbook which provisions the aggregator node and compute nodes in the cluster.
- **Prometheus/Grafana (Optional)** - If clients already have their own prometheus/grafana running then only deploy the aggregator container.

6.4. Setup Instructions

Ensure the latest NVSM (Multinode supported version[24.03.05] or above) is installed on all compute nodes. The aggregator container is pre-packaged with the latest NVSM.

6.4.1. Installing Docker

To install docker and docker-compose on Aggregator mode, run the following commands:

```
$ apt-get install docker
$ apt-get install docker-compose
```

6.4.2. Provision Aggregator Node

6.4.2.1 Download Docker Images

Download the required containers from https://catalog.ngc.nvidia.com/containers below:

1. NVSM-Aggregator.
2. NVMS-Provision.
3. NVSM-Grafana.
4. NVSM-Prometheus.

6.4.2.2 Load Docker Images

Run the following command to load the docker images:

```
$ docker load -i nvsm-prometheus_25.03.05.tar.gz
$ docker load -i nvsm-grafana_25.03.05.tar.gz
$ docker load -i nvsm-provision_25.03.05.tar.gz
$ docker load -i nvsm-aggregator_25.03.05.tar.gz
```

Run the following command to ensure the container images are present:

```
$ docker images
$ REPOSITORY
↪TAG      IMAGE ID        CREATED         SIZE
$ nvcr.io/nvstaging/cloud-native/nvsm-grafana         25.03.05       12e4ebaee709
↪3 hours ago     541MB
$ nvcr.io/nvstaging/cloud-native/nvsm-prometheus 25.03.05    68518ef28efc    3 hours
↪ago     376MB
$ nvcr.io/nvstaging/cloud-native/nvsm-provision      25.03.05        3869ef50cbbd
↪3 hours ago     518MB
$ nvcr.io/nvstaging/cloud-native/nvsm-aggregator   25.03.05  eb6068a414be    3 hours
↪ago     1.55GB
```

6.4.3. Create Inventory YAML File

Inventory file stores information of all compute nodes like BMC IP, Host IP and encrypted passwords. Using the inventory file, the node provisioner container invokes ansible playbook to copy certificates from the aggregator node to all compute nodes and restart compute nodes NVSM to connect to aggregator node NVSM.

Since the inventory.yaml file contains username and passwords, it must be secured to the admin user only.

Use the below sample file to create an inventory.yaml file:

```
aggregator:
    hosts:
        # Add aggregator host here
        # Example:
        #
        # aggregator.example.com:
    vars:
        # Aggregator
        nvsm_exporter_port: 9123
        nvsm_aggregator_image: "nvcr.io/nvstaging/nvsm/nvsm-aggregator:@PACKAGE_
↪VERSION@"
        nvsm_aggregator_container: "nvsm-aggregator"   # Name of nvsm container
        nvsm_aggregator_network: "nvsm"                # Name of nvsm container
↪network
        # Dashboard stack - Prometheus & Grafana
        nvsm_enable_dashboard: true
        nvsm_prometheus_image: "nvcr.io/nvstaging/nvsm/nvsm-prometheus:@PACKAGE_
↪VERSION@"
        nvsm_grafana_image: "nvcr.io/nvstaging/nvsm/nvsm-grafana:@PACKAGE_VERSION@"
        nvsm_grafana_port: 3000
        nvsm_grafana_admin_user: "nvsm"
        nvsm_grafana_admin_password: "nvsm"
        # Api Gateway
        nvsm_api_gateway_port: 273

compute:
    hosts:
        # Add compute nodes here, echo node should have a nvsm_id and bmc_ip
        # Examples:
        #
        # dgx01.example.com:
        #   nvsm_id: 1
        #   bmc_ip: "192.168.10.1"
        # dgx02.example.com:
        #   nvsm_id: 2
        #   bmc_ip: "192.168.10.2"
        #   # overwrite the group vars if required.
        #   # for example, the host have different user/password
        #   ansible_user: "sshuser02"
        #   ansible_ssh_pass: "sshpwd02"
        #   ansible_sudo_pass: "sshpwd02"
        #   bmc_pass: "bmcpwd02"
        # dgx03.example.com:
        #   nvsm_id: 3
        #   bmc_ip: "192.168.10.3"
```

(continues on next page)

(continued from previous page)

```
        #   ansible_user: "sshuser03"
        #   # Use literal block scalar, if the password contains special characters
↪like double quote(")
        #   # The password here is "specialSSHPass" (including the double quotes)
        #   ansible_ssh_pass: |-
        #     "specialSSHPass"
        #   ansible_sudo_pass: |-
        #     "specialSSHPass"
        #   bmc_pass: |-
        #     "specialBMCPass"
vars:
    # BMC user/password applies to all compute hosts,
    # they can be override by host variables if some hosts have different passwords
    bmc_user: "admin"
    bmc_pass: "admin"
all:
children:
    compute:
    aggregator:
vars:
    # Aggregator variables
    nvsm_mqtt_host: "aggregator.example.com"
    nvsm_mqtt_port: 8883
    # SSH user/password applies to all aggregator and compute hosts,
    # they can be override by host variables if some hosts have different passwords
    # The password here is sshpwd (without the double quotes)
    ansible_user: "sshuser"
    ansible_ssh_pass: "sshpwd"
    ansible_sudo_pass: "sshpwd"
    # Force reprovision
    nvsm_force_reprovision: false
```

Perform the following steps on the Aggregator node:

```
$ mkdir -p /etc/nvsm/aggregator
$ vim /etc/nvsm/aggregator/inventory.yaml
$ sudo chown root:root /etc/nvsm/aggregator/inventory.yaml
$ sudo chmod 0600 /etc/nvsm/aggregator/inventory.yaml
```

Update the inventory file `/etc/nvsm/aggregator/inventory.yaml` with the following information:

- Add host to hosts section under the aggregator , it can be the hostname or host IP address of the aggregator node.
- Change `nvsm_mqtt_host` variable under `vars` of `aggregator` to hostname or host IP address of the aggregator node.
- Add hosts to the hosts section under `compute`, it can be hostname or host IP address of compute nodes.
- Each compute node host must have an unique `nvsm_id`, it specifies compute node id when they connect to the aggregator node. nvsm_id can start from 1 to 1023.
- Each compute node host must have an unique `bmc_ip`, it specifies the BMC ip associated with that node.
- Specify SSH user/password for each compute node by updating `ansible_user`, `ansible_ssh_pass`, `ansible_sudo_pass` under the hosts section. Section `vars` variables under

`all` is applicable to all hosts where variables defined under host section overrides `vars` variables.

6.4.3.1 Inventory File Details

The inventory.yaml are an ansible inventory file, see the details here.

The inventory file have a group called “all”, which contain 2 sub-groups:

- **aggregator** group - contains the host list of aggregator hosts. Currently only one host is supported.
- **compute** group - contains all DGX nodes.

The vars can be defined as the ‘all’ group, it will apply to all hosts (including aggregator and compute). Vars defined in the ‘compute’ group apply to the compute nodes, but do not apply to the aggregator group, but it will override the value from the “all” group if the var is also defined there. Vars defined in a host apply to that host only, and it will override the value defined in a group.

Examples: If all hosts share the same ssh user / passwd, please define ansible_user, ansible_ssh_pass, ansible_sudo_pass under all.vars If a particular hosts have a different user / passwd, please define it under the host, something like

```
$ dgx03.example.com:
    nvsm_id: 3
    bmc_ip: "192.168.10.3"
    # overwrite the group vars if required
    ansible_user: "sshuser"
    ansible_ssh_pass: "sshpwd"
    ansible_sudo_pass: "sshpwd"
```

6.4.4. Run Docker Command to Provision Nodes

6.4.4.1 Start Aggregator Container and Provision Compute Nodes

Run the provision container with configuration as listed in the command, Use the image id for provision container found from docker images command. Provisioner will start the Aggregator, Grafana and Prometheus containers using configurations mentioned in the inventory file.

The inventory file stores information of all compute nodes BMC IP, Host IP and passwords. Using the inventory file, the provision container invokes an ansible playbook to copy certificates from Aggregator node to compute nodes and restart compute nodes NVSM to connect to aggregator node NVSM.

```
$ docker run -it --rm -v /etc/nvsm/aggregator:/etc/nvsm/mn  nvcr.io/nvstaging/nvsm/
↪nvsm-provision:25.03.05
```

Sample Output:

```
Verifying multinode inventory file...
↪
↪                                                Inventory file is not encrypted.
↪Please encrypt it using 'nvsm MultinodeInventory encrypt'
Inventory validation completed successfully!
↪
↪                                                 Starting provisioning...
```

(continues on next page)

(continued from previous page)

```
PLAY [NVSM Aggregator node precheck]
↪*********************************************************************************************

TASK [Ping aggregator node]
↪*********************************************************************************************
ok: [aggregator.example.com]

TASK [Get inventory file stat]
↪*********************************************************************************************
ok: [aggregator.example.com]

TASK [Check inventory file permission]
↪*********************************************************************************************
ok: [aggregator.example.com] => {
    "changed": false,
    "msg": "All assertions passed"
}

TASK [Check if docker command is available]
↪*********************************************************************************************
ok: [aggregator.example.com]

TASK [Check if docker compose plugin is available]
↪*********************************************************************************************
ok: [aggregator.example.com]

TASK [Using 'docker compose']
↪*********************************************************************************************
ok: [aggregator.example.com]

TASK [Check if docker-compose command is available]
↪*********************************************************************************************
skipping: [aggregator.example.com]

TASK [Using 'docker-compose']
↪*********************************************************************************************
skipping: [aggregator.example.com]

TASK [Show docker compose command]
↪*********************************************************************************************
ok: [aggregator.example.com] => {
    "msg": "Using 'docker compose'"
}
PLAY [NVSM Worker node precheck]
↪*********************************************************************************************
TASK [Ping compute node]
↪*********************************************************************************************
ok: [192.168.10.1]
TASK [Print NVSM Node ID]
↪*********************************************************************************************
ok: [192.168.10.1] => {
    "msg": "Precheck node ID 1"
}
TASK [Get Packages Facts]
↪*********************************************************************************************
```

(continues on next page)

(continued from previous page)

```
ok: [192.168.10.1]
TASK [Install NVSM]
↪*********************************************************************************************************
skipping: [192.168.10.1]
TASK [Get Packages Facts]
↪*********************************************************************************************************
skipping: [192.168.10.1]
TASK [Get current NVSM version]
↪*********************************************************************************************************
ok: [192.168.10.1]
TASK [Check if NVSM version meet requirement]
↪*********************************************************************************************************
skipping: [192.168.10.1]
PLAY [NVSM Aggregator node provision]
↪*********************************************************************************************************
TASK [Check if aggregator was provisioned]
↪*********************************************************************************************************
ok: [aggregator.example.com]
TASK [Check if aggregator container exists]
↪*********************************************************************************************************
ok: [aggregator.example.com]
TASK [Remove existing aggregator container]
↪*********************************************************************************************************
changed: [aggregator.example.com]
TASK [Create Aggregator docker-compose.yml]
↪*********************************************************************************************************
ok: [aggregator.example.com]
TASK [Check if aggregator container exists]
↪*********************************************************************************************************
ok: [aggregator.example.com]
TASK [Start aggregator container]
↪*********************************************************************************************************
changed: [aggregator.example.com]
TASK [Wait for NVSM keyfiles to be created]
↪*********************************************************************************************************
ok: [aggregator.example.com] => (item=nvsm-ca.crt)
ok: [aggregator.example.com] => (item=nvsm-client.crt)
ok: [aggregator.example.com] => (item=nvsm-client.key)
TASK [Make sure NVSM MQTT server is ready]
↪*********************************************************************************************************
ok: [aggregator.example.com]
PLAY [NVSM compute node provision - phase2]
↪*********************************************************************************************************
TASK [Copy keyfiles]
↪*********************************************************************************************************
changed: [192.168.10.1] => (item=nvsm-ca.crt)
changed: [192.168.10.1] => (item=nvsm-client.crt)
changed: [192.168.10.1] => (item=nvsm-client.key)
TASK [Copy using inline content]
↪*********************************************************************************************************
changed: [192.168.10.1]
TASK [Restart NVSM]
↪*********************************************************************************************************
changed: [192.168.10.1]
PLAY [NVSM Aggregator node - post config]
↪*********************************************************************************************************
```

(continues on next page)

(continued from previous page)

```
TASK [Restart nvsm-exporter]
↪*********************************************************************************************
changed: [aggregator.example.com]
TASK [Reload nvsm-lifecycled]
↪*********************************************************************************************
changed: [aggregator.example.com]
PLAY [NVSM Aggregator node postcheck]
↪*********************************************************************************************
TASK [Post check]
↪*********************************************************************************************
ok: [aggregator.example.com] => {
    "msg": "NVSM Aggregator node postcheck"
}
PLAY [NVSM Worker node postcheck]
↪*********************************************************************************************
TASK [Post check]
↪*********************************************************************************************
ok: [aggregator.example.com] => {
    "msg": "NVSM Aggregator node postcheck node ID 1"
}
PLAY RECAP
↪*********************************************************************************************
192.168.10.1               : ok=25   changed=7    unreachable=0    failed=0
↪skipped=5    rescued=0    ignored=0
```

Check all containers are running on the Aggregator node:

```
$ sudo docker ps
CONTAINER ID   IMAGE  COMMAND   CREATED   STATUS   NAMES    PORTS
e7f9a56e6ef4   nvcr.io/nvstaging/cloud-native/nvsm-grafana:25.03.05           "/bin/
↪bash -c ./entr_"   About an hour ago   Up About an hour   nvsm-grafana

2fe6fb57ed2a   nvcr.io/nvstaging/cloud-native/nvsm-prometheus:25.03.05    "/bin/bash -
↪c ./entr_"   About an hour ago   Up About an hour   nvsm-prometheus

fff3ed636853   nvcr.io/nvstaging/cloud-native/nvsm-aggregator:25.03.05    "/bin/bash -
↪c /usr/b_"   About an hour ago   Up About an hour   nvsm-aggregator     0.0.0.
↪0:3000->3000/tcp, :::3000->3000/tcp, 0.0.0.0:8883->8883/tcp, :::8883->8883/tcp, 0.0.
↪0.0:9123->9123/tcp, :::9123->9123/tcp
```

6.4.5. Connect to Multinode NVSM

After the provision, within a few minutes (3~5 mins), compute nodes must have connected to aggregator node NVSM. Now it's time to check the NVSM multinode via login to the aggregator container.

6.4.5.1 Enter Aggregator Container

Login to the Aggregator container using the below command:

```
$ docker exec -it nvsm-aggregator /bin/bash
```

6.4.5.2 Examining the Aggregator NVSM Services

Run the following command on the Aggregator container to check running NVSM services:

```
$ nvsm status
SERVICE             ENABLED   ACTIVE   SUB       DESCRIPTION
================================================================================
nvsm-exporter enabled  active  running  NVSM Exporter to provide DGX System
↪Management Metrics
nvsm-lifecycled        enabled  active  running  NVSM aggregator lifecycle manager
nvsm-mqtt                enabled  active  running  MQTT broker for NVSM API for
↪signaling within NVSM API     components
```

Examine the `nvsm_core` running in aggregator:

```
$ # nvsm_lifecycle status
Hostname/IP  Node                                    Command  PID    Stat  StartCount
↪CrashReason  CrashCode
10.33.1.23    23  /usr/sbin/nvsm_core -mode server SERVE 23   99  Running           2
↪        None          0
10.33.1.24    24  /usr/sbin/nvsm_core -mode server SERVE 24  105  Running           2
↪        None          0
```

Examine the status for a given node:

```
$ # nvsm show status --node 23
Node ID: 23
Service Status:
    SERVICE                        STATUS
    Aggregator node nvsm_core      Running
    Compute node nvsm_core         Inactive
```

6.4.5.3 Run NVSM CLI commands for Cluster Nodes

We can run all supported NVSM CLI/show commands targeting any cluster node including show/dump health.

Examples to run the NVSM show commands targeted to a compute node:

```
$ nvsm show power --node 1
$ nvsm show gpus --node 2
$ nvsm show storage --node 3
$ nvsm show network --node 4
$ nvsm show alerts --node 5
$ nvsm show health --node 6
$ nvsm show health --node 7
```

Example to run the NVSM health command on all compute nodes:

```
$ nvsm show health
$ nvsm show health --node all
```

Example to collect nvsm dump from a compute node and store on aggregator node:

Dump collection gets executed on the compute node and copied back to the aggregator node at the output dir path.

```
$ nvsm dump health --node 1
```

Example to run nvsm cli command on any compute node.

```
$ nvsm
$ show
$ cd 1
$ show
$ ...iterate over any resource
```

6.4.5.4 Accessing Aggregator Services

From the aggregator node we can access the exporter having all cluster nodes information.

- **nvsm-exporter** - http://{[}aggregator Hostname/IP]:9123/metrics
- **grafana** - http://{[}aggregator Hostname/IP]:3000

6.5. Admin Tasks

6.5.1. Securing Multinode Inventory File

The multinode inventory file contains the credentials of all systems and BMCs. NVSM provides a set of commands to encrypt and manage the file.

An user can encrypt or decrypt the multinode inventory file at any time without impacting running NVSM services.

6.5.1.1 Encrypting Multinode Inventory File

The following command encrypts the multinode inventory file:

```
$ nvsm MultinodeInventory encrypt

Enter to set keystore password:
Confirm keystore password:
Initializing keystore...
Keystore initialized successfully.
Encrypting inventory file...
Inventory file encrypted successfully.
```

When running the command for the first time, you are prompted to set a keystore password. NVSM then generates a random encryption key, using it to encrypt the multinode inventory file.

Note: Ensure the Admin remembers this password.

Enter this same keystore password for all subsequent NVSM MultinodeInventory commands.

6.5.1.2 Viewing Multinode Inventory File

The following command decrypts the multinode inventory file, and prints it to the console:

```
nvsm MultinodeInventory view

Enter keystore password:
aggregator:
hosts:
....
```

6.5.1.3 Editing Multinode Inventory File

The following command edits the multinode inventory file:

```
nvsm MultinodeInventory edit

Enter the keystore password:
-> Edit your inventory file
The inventory file was updated successfully.
```

This decrypts the multinode inventory into a temp file, and opens it with VI editor. After saving this temp file, NVSM performs the validation, then writes it back with encryption.

6.5.1.4 Decrypting Multinode Inventory File

The following command decrypts the multinode inventory file, although this is highly not recommended:

```
nvsm MultinodeInventory decrypt

Enter keystore password:
Decryption successful
Inventory file decrypted successfully
```

6.5.1.5 Using a New Encryption Key

NVSM does not support setting the encryption key. In the case where the encryption key is leaked however, a new encryption key can be generated by decrypting and encrypting the multinode inventory:

```
nvsm MultinodeInventory decrypt

Enter keystore password:
Decryption successful
Inventory file decrypted successfully
```

(continues on next page)

(continued from previous page)

```
nvsm MultinodeInventory encrypt

Enter to set keystore password:
Confirm keystore password:
Initializing keystore...
Keystore initialized successfully.
Encrypting inventory file...
Inventory file encrypted successfully.
```

Ensure you change your keystore password in this scenario.

6.5.1.6 Changing the Keystore Password

The following command changes the keystore password:

```
nvsm MultinodeInventory passwd

Enter current keystore password:
Enter new password:
Confirm new password:
Keystore password changed successfully.
```

6.5.2. Add New Nodes

On the Aggregator node, edit the multinode inventory file. Add new compute nodes to the compute hosts section, and rerun the multinode provisioning.

```
nvsm MultinodeInventory edit
docker run -it --rm -v /etc/nvsm/aggregator:/etc/nvsm/mn  nvcr.io/nvstaging/nvsm/nvsm-
↪provision:25.03.05
```

6.5.3. Removing Nodes

On the aggregator node, remove compute nodes with the `MultinodeInventory` command:

```
nvsm MultinodeInventory remove [comma separate hostname]
```

This removes the multinode configs on the compute nodes, and restarts the NVSM service on the compute node as a standalone node.

If the compute node is not accessible but returns after, the Admin is responsible for removing the dir path */etc/nvsm/mn*, and restarting NVSM service manually on the compute node:

```
rm -rf /etc/nvsm/mn
systemctl restart nvsm
```

The compute node is then removed from the inventory file, regardless if the above was performed or not:

```
$ # removing 1 node
nvsm MultinodeInventory remove dgx01.example.com

$ # removing multiple nodes with comma separated list
nvsm MultinodeInventory remove dgx01.example.com,dgx02.example.com

$ # It's recommended to backup the inventory file with "-b" option
nvsm MultinodeInventory remove -b dgx01.example.com,dgx02.example.com
```

6.6. NVSM OOB Telemetry Collection

6.6.1. Overview

The NVSM Telemetry feature collects system metrics from BMC and exports them to InfluxDB for analysis and visualization. OOB Telemetry feature is supported for Hopper and Blackwell DGX systems onwards.

Note: Telemetry collection is based on BMC redfish URIs. This feature is only functional from the multinode Aggregator node.

6.6.2. Enabling Telemetry

The following command enables telemetry:

```
nvsm enable telemetry

Enable telemetry collection and export to InfluxDB? [yes/no]: yes
Telemetry collection enabled.
```

After enabling: - NVSM starts metrics collection based on platform configuration. - Initiates InfluxDB export if configured. - Enables periodic collection of Redfish URIs metrics at specified intervals.

6.6.3. Disabling Telemetry

The following command disables telemetry:

```
nvsm disable telemetry

Disable telemetry collection and export to InfluxDB? [yes/no]: yes
Telemetry collection disabled.
```

After disabling: - Stops metric collection and InfluxDB export. - Halts ongoing telemetry operations.

6.6.4. Exporting Data from InfluxDB

InfluxDB is a time-series database used by NVSM to store telemetry metrics. It is optimized for high write and query loads, making it ideal for storing telemetry metrics data.

In the NVSM telemetry implementation: - Data is stored in the “nvidia” organization, the default authentication token is pre-configured. - Bucket for each compute node corresponds to *nvsm-<node_id>*. - Data is stored with timestamps, allowing historical analysis. - Aggregator container has required packages including InfluxDB and Influx cli.

```
$ # Export all data
influx query 'from(bucket:"nvsm-1") |> range(start: 0)' --org nvidia --token <token-
↪id> --raw > all_data.csv

$ # Export last 24 hours data:
influx query 'from(bucket:"nvsm-1") |> range(start: -24h)' --org nvidia --token
↪<token-id> --raw > 24hours_data.csv

$ # Export filtered metrics:
influx query 'from(bucket:"nvsm-1") |> range(start: 0) |> filter(fn: (r) => r._
↪measurement == "thermal")' --org nvidia --token <token-id> --raw > thermal_data.csv
```

Chapter 7. Third-Party Licenses

This NVIDIA product contains third party software that is being made available to you under their respective open source software licenses. Some of those licenses also require specific legal information to be included in the product. This section provides such information.

7.1. mattn/go-sqlite3

The matt/go-sqlite3 software (https://github.com/mattn/go-sqlite3) is provided under the following terms:The MIT License (MIT)

Copyright (c) 2014 Yasuhiro Matsumoto

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Copyright