

NVIDIA Base Command Manager 11

NVIDIA Mission Control Manual

Revision: 5e8253781

Date: Thu Apr 2 2026

Table Of Contents

Table Of Contents	3
0.1 About This Manual	7
0.2 About The Manuals In General	7
0.3 Getting Administrator-Level Support	8
0.4 Getting Professional Services	8
1 Introduction	9
1.1 NVIDIA Mission Control: Overview And Licensing	9
1.2 NVIDIA Mission Control Features	9
1.3 Checking That The NVIDIA Mission Control Edition Is Running	10
1.4 NVIDIA Mission Control Installation Wizards	11
2 NetQ 5.0 Settings For NVLink Monitoring	13
2.1 NMX Settings For NVLink Monitoring	13
2.2 Installing NetQ 5.0 Using The Mission Control Wizard	13
2.2.1 Permanent License Generation And Application Guide	13
2.2.2 Downloading The NetQ Installation Package	14
2.3 NMX Server Authentication Configuration In BCM	18
2.3.1 NMX Manager Metrics Forwarding Via BCM And prometheusmetricforwarders	19
3 Rack Management And Management Policies	21
3.1 Introduction	21
3.2 Basic Rack Management	21
3.3 Introduction To Advanced Features In Rack Management	24
3.3.1 How BMS And BCM Work Together	24
3.3.2 Commands For Monitoring Advanced Features In Rack Management	26
3.3.3 Managing Leaks With rules In leakactionpolicies	28
4 BCM Power Shelf Integration	33
4.1 Power Shelf Listing And Overview	33
4.2 Power Shelves Networking Configuration	34
4.3 Access Configuration For The Power Management Controller	34
4.4 Power Shelf Settings Overview	35
4.5 Power Shelf Metrics Overview	35
4.6 Power Shelf Firmware	37
4.7 Power Shelf Circuit Monitoring	37
4.8 Power Shelves And Rack Mode Commands	37
4.8.1 The list Command	37
4.8.2 The rackoverview Command	38

4.8.3	The <code>display</code> Command	39
4.8.4	Power Management Of Racks	40
5	NVIDIA Autonomous Job Recovery	43
5.1	Introduction	43
5.2	Prerequisites To Install Autonomous Job Recovery: Kubernetes Installation	43
5.2.1	Kubernetes Components Needed For Installation Of Autonomous Job Recovery	43
5.2.2	Settings For Kubernetes Components Used In Autonomous Job Recovery Installation	44
5.3	Prerequisites To Install Autonomous Job Recovery: Slurm Installation	45
5.4	Autonomous Job Recovery Installation With <code>cm-mission-control-setup</code>	45
5.4.1	Credentials Required To Install Autonomous Job Recovery When Running <code>cm-mission-control-setup</code>	45
5.4.2	Running <code>cm-mission-control-setup</code>	45
5.5	Post-installation Checks	46
5.5.1	Grafana Access	47
5.5.2	Autonomous Job Recovery Failure When The Slurm Daemon Spool Directory Is Not Cleaned Up	47
6	NVIDIA Autonomous Hardware Recovery	49
6.1	Introduction	49
6.2	Prerequisites To Install Autonomous Hardware Recovery	49
6.3	Installation Of NVIDIA Autonomous Hardware Recovery Using <code>cm-mission-control-setup</code>	50
6.4	Verifying Autonomous Hardware Recovery Is Up And Ready	54
6.4.1	Checking The Pods	54
6.4.2	Testing The Web Interface	54
7	NVLink Technology	59
7.1	NVLink Overview	59
7.1.1	NVLink Generations	59
7.1.2	NVLink Vs PCIe	59
7.2	NVLink Architecture In GB200/GB300	59
7.2.1	Compute Tray Configuration	60
7.2.2	NVLink Switch Trays	60
7.2.3	NVL72 Topology	60
7.3	NVLink Commands	60
7.3.1	Listing NVLink Switches: <code>list</code>	60
7.3.2	NVLink Switch Status	61
7.3.3	NVLink Fabric Information: <code>nvfabricinfo</code>	61
7.3.4	NVLink Platform Information: <code>nvplatforminfo</code>	62
7.3.5	NVLink Domain Information: <code>nvdomaininfo</code>	63
7.3.6	NVLink SDN Partition Information: <code>nvsdnpartitioninfo</code>	64
7.3.7	NVLink Link Status: <code>nvlinkinfo</code>	64
7.3.8	NVLink Monitoring With NMX	65
8	NVLink Switch Integration	67
8.1	NVLink Switch Listing And Overview	67

8.2	NVLink Switch Configuration	68
8.3	NVLink Switch Monitoring	70
8.4	NVLink Switch Redfish Events	70
9	Power Reservation Steering	71
9.1	Introduction	71
9.2	Deployment Of PRS	71
9.3	Changes Made On Deployment, And Managing Changes Post-Deployment With cmsh	73
9.3.1	PRS Values	73
9.3.2	Slurm Values	74
9.3.3	PRS Removal	75
9.4	Changes Made On Deployment, And Managing Changes Post-Deployment With Base View	75
9.4.1	Base View Power Reservation Steering Wizard	75
9.4.2	Base View Power Reservation Steering Client And Server Configuration Overlays	76
10	Domain Power Service	77
10.1	Introduction	77
10.2	Deployment Of DPS	77
10.3	Known Issues For BCM 11.31.0	78
10.3.1	Known Issue When Undeploying DPS	78
10.3.2	Verify Deployment Failure Due To Mismatching LDAP User Password	79
11	NVIDIA Mission Control DGX GB200 Measurables	81
11.1	Circuit-Related DGX GB200 Metrics	81
11.2	Leak Detection DGX GB200 Measurables	82
11.3	DGX GB200 NVLink Measurables	83
11.4	DGX GB200 Power Shelf Measurables	84
11.5	Cooling Distribution Unit Metrics On The DGX GB200	85
11.6	GPU Measurables For The DGX GB200	86
11.7	Prometheus Metrics For The DGX GB200	88
11.8	Redfish Metrics For The DGX GB200	90
11.9	Redfish Enums For The DGX GB200	106
11.10	Health Checks For The DGX GB200	111

Preface

Welcome to the *NVIDIA Mission Control Manual* for NVIDIA Base Command Manager 11.

0.1 About This Manual

The NVIDIA Mission Control features of NVIDIA Base Command Manager (BCM) are available on clusters that are running the NVIDIA Mission Control edition of BCM. The features are listed in chapter 1 of this manual.

This manual is aimed at helping cluster administrators install, understand, configure, and manage the NVIDIA Mission Control features and capabilities of NVIDIA Base Command Manager. The administrator is expected to be reasonably familiar with the *BCM Administrator Manual*. The *NVIDIA Mission Control Manual* is thus a supplementary manual to the *BCM Administrator Manual* for the NVIDIA Mission Control features.

0.2 About The Manuals In General

Regularly updated versions of the manuals are available on updated clusters by default at `/cm/shared/docs/cm`. The latest updates are always online at <https://docs.nvidia.com/base-command-manager>.

- The *Installation Manual* describes installation procedures for the basic cluster.
- The *Administrator Manual* describes the general management of the cluster.
- The *User Manual* describes the user environment and how to submit jobs for the end user.
- The *Cloudbursting Manual* describes how to deploy the cloud capabilities of the cluster.
- The *Developer Manual* has useful information for developers who would like to program with BCM.
- The *Edge Manual* explains how BCM can be used with edge sites.
- The *Containerization Manual* describes how to manage containers with BCM.

If the manuals are downloaded and kept in one local directory, then in most pdf viewers, clicking on a cross-reference in one manual that refers to a section in another manual opens and displays that section in the second manual. Navigating back and forth between documents is usually possible with keystrokes or mouse clicks.

For example: `<Alt>-<Back arrow>` in Acrobat Reader, or clicking on the bottom leftmost navigation button of xpdf, both navigate back to the previous document.

The manuals constantly evolve to keep up with the development of the BCM environment and the addition of new hardware and/or applications. The manuals also regularly incorporate feedback from administrators and users, who can submit comments, suggestions or corrections via the website

<https://enterprise-support.nvidia.com/s/create-case>

Section 14.2 of the *Administrator Manual* has more details on submitting an issue.

0.3 Getting Administrator-Level Support

If the reseller from whom BCM was bought offers direct support, then the reseller should be contacted.

Otherwise the primary means of support is via the website

<https://enterprise-support.nvidia.com/s/create-case>

Section 14.2 of the *Administrator Manual* has more details on working with support.

0.4 Getting Professional Services

The BCM support team normally differentiates between

- regular support (customer has a question or problem that requires an answer or resolution), and
- professional services (customer asks for the team to do something or asks the team to provide some service).

Professional services can be provided via the NVIDIA Enterprise Services page at:

<https://www.nvidia.com/en-us/support/enterprise/services/>

1 Introduction

1.1 NVIDIA Mission Control: Overview And Licensing

The NVIDIA Mission Control features are extended features that are only available in the NVIDIA Mission Control edition of NVIDIA Base Command Manager (BCM). NVIDIA Mission Control features are targeted at NVIDIA B200, GB200, and other OEM platforms. The NVIDIA sales team must be contacted in case an NVIDIA Mission Control edition is required.

The NVIDIA Mission Control features of the NVIDIA Mission Control edition of BCM can be enabled on a BCM cluster with an active BCM license. Activation of the BCM license is described in Chapter 4 of the *Installation Manual* of BCM

1.2 NVIDIA Mission Control Features

The NVIDIA Mission Control edition is capable of running the following:

- NVIDIA-conforming BMS: a *Building Management System* software that aids in cluster management in the building.
- Leak detection: monitoring for leak detection in the fluid-cooled processors. Associated with this are the possible automated actions taken.
- Autonomous hardware recovery: Automates some hardware management operations in order to keep the cluster up.
- Autonomous job recovery: integrates a suite of autonomous software services to enhance job submission, restart procedures, carry out telemetry collection, and carry out anomaly resolution processes.
- NVLink Management Software (NetQ): Integrated with NMX, to manage Ethernet and NVLink
- IMEX (Internode Memory Exchange Service): a secure service that facilitates the mapping of GPU memory over NVLink between the GPUs in an NVLink domain. IMEX configuration is described in section 7.5.1 of the *Administrator Manual* as part of workload management configuration.
- GB200 rack management: Automated management of rack power up and power down sequence.
- GB200 firmware management: GB200 firmware and BIOS management.
- Power reservation steering: Predicting power consumption, and capping the power to active components based on that prediction, to reduce data center energy costs.
- Run:ai: Kubernetes-based workload management and orchestration via Run:ai.

1.3 Checking That The NVIDIA Mission Control Edition Is Running

Whether the NVIDIA Mission Control edition is running on the cluster can be checked:

- in cmlsh:
 - using the `licenseinfo` command within cmlsh

Example

```
root@basecm11:~# cmlsh -c 'main licenseinfo | egrep Version\|Edition'
Version                10 and above
Edition                NVIDIA Mission Control
```

or

- using the `verify-license` utility from a command line:

Example

```
root@basecm11:~# verify-license info
===== Certificate Information =====
Version:                10
Edition:                NVIDIA Mission Control
...
```

or

- using the `editiondisabledfeatures` command.
 - * If the NVIDIA Mission Control edition is running, then the features are all available, as indicated by:

Example

```
root@basecm11:~# cmlsh -c 'main editiondisabledfeatures'
No disabled features
```

- * If the NVIDIA Mission Control edition is not running, then the features are not available, as indicated by:

```
root@basecm11:~# cmlsh -c 'main editiondisabledfeatures'
LeakDetection
DisplayRack
RackPower
AutonomousHardwareRecovery
AutonomousJobRecovery
Firmware.flash:GB200
BMS
Run:ai
IMEX
NMX
PRS
```

- in Base View the status of the NVIDIA Mission Control features can be seen in the navigation paths:

Cluster > License Information > Mission Control

or

Cluster > Overview > License Information

1.4 NVIDIA Mission Control Installation Wizards

Installation of NVIDIA Mission Control features can be carried out with:

- a `cm-mission-control-setup` TUI wizard, which comes with the `cm-setup` package
- a Base View GUI wizard, if available.

Starting with BCM release version 11.32, the `cm-mission-control-setup` wizard has some of its components decoupled, so that they are now plugins. This allows minor BCM releases to come out more frequently, since developing NVIDIA Mission Control plugin components can be carried out more independently from the BCM release schedule, compared with developing a monolithic `cm-mission-control-setup`.

The decoupled version automatically comes with a new system package `cm-config-apt-nmc`, which lets the cluster administrator select updated plugins for installation on all BCM nodes from the NMC package repository (`edge.urm.nvidia.com/artifactory/sw-dgxc-nmc-cm-setup-plugins-debian-local`). By default, updating is skipped, unless explicitly chosen, when the wizard is run.

The plugins are implemented as TUI plugins, and there are additionally some GUI (Base View) plugins which run the TUI version in their backend.

Decoupled plugin components at the time of writing (February 2026) are:

- NetQ (Chapter 2) (TUI only)
- AJR (Chapter 5) (TUI and Base View)
- AHR (Chapter 6) (TUI and Base View)
- DPS (Chapter 10) (TUI only)

The plugins are packages containing Python scripts:

- The TUI plugin package name format is:
`cm-setup-<component>-bcm<major version>` (for example: `cm-setup-netq-bcm11`)
- The Base View plugin package names format is:
`cm-base-view-<component>-bcm<major version>` (for example: `cm-base-view-netq-bcm11`)

2 NetQ 5.0 Settings For NVLink Monitoring

NetQ is a scalable network monitoring and management tool for NVLink, Ethernet, or both. It can work with NVLink fabrics and Cumulus Linux switches. NetQ version 5.0 is unified with NMX-Manager, and uses NMX-Manager capabilities if used in NVLink mode.

2.1 NMX Settings For NVLink Monitoring

NMX is a product suite for HPC cluster network monitoring and management. The system is made up of:

- NMX Telemetry
- NMX Manager
- NMX Controller
- NMX Oasis

The NVIDIA Mission Control edition of BCM can monitor NMX telemetry services that run on:

- NVLinks (direct GPU interconnection within a server)
- NVLink switches (direct GPU interconnection across servers)

2.2 Installing NetQ 5.0 Using The Mission Control Wizard

2.2.1 Permanent License Generation And Application Guide

When installing NetQ, the system receives an evaluation license valid for 60 days. When the evaluation license expires, REST API access is blocked until a new license is applied.

Generating a License File

Before generating the license file, the following must be prepared:

- A list of servers with the MAC address of each server on which the NetQ software is to be installed
- Access to the NVIDIA Licensing Portal (NLP) with valid credentials

The following steps generate the license file:

- the NLP is accessed, and logged into with the prepared credentials.
- The **Network Entitlements** tab is clicked. This displays a list of the serial licenses of all software products. For each product software product some license information details are shown, including the license status.
- The license can be selected from the list, and activated by clicking on the **Actions** button.
- The MAC addresses are configured.
 - In the MAC Address field, the MAC address of the delegated license-registered host is added.
 - If applicable, in the HA MAC Address field, the High Availability (HA) server MAC address is added.
 - **Note:** If there is more than one NIC installed on a UFM Server, then any of the MAC addresses can be used.
- The license is then generated and downloaded:
 - Clicking on **Generate License File** creates the license key file for the software.
 - Clicking on **Download License File** allows the license file to be saved on to a local computer.

License Regeneration Notes

If the cluster administrator regenerates a license, then the following must be considered:

- If the NIC or server is replaced, then the steps to generate license file must be repeated for the new MAC address.
- Regeneration of the license can only be carried out two times. For further regeneration, support should be contacted at enterprisesupport@nvidia.com.

2.2.2 Downloading The NetQ Installation Package

The NetQ installation package can be downloaded from the NVIDIA Licensing Portal (NLP).

Downloading NetQ

The following steps allow the package to be downloaded:

1. The NLP is accessed, and logged into with the prepared credentials.
2. **Software Downloads** is clicked, the product family is filtered to NetQ, the relevant version is found, and the “Appliance” platform package is downloaded.
3. **Download** is clicked.
4. The file is saved to the local drive.
5. **Close** is clicked

The .tar.gz file is copied over to the BCM head node using scp or rsync.

The Debian packages can be downloaded from:

<https://download.nvidia.com/cumulus/apps3.cumulusnetworks.com/repos/deb/pool/netq-5.0/>

The relevant apps and agents package for ubuntu24 and the relevant CPU architecture (arm or amd) should be searched for. These three files are required:

1. The NetQ tarball
2. The netq-agent .deb package
3. The netq-apps .deb package

Installation

Running `cm-mission-control-setup` brings up the TUI installation wizard for NetQ. Deployment of NetQ is done by the cluster administrator with the following steps:

1. The NetQ installation is selected (figure 2.1):

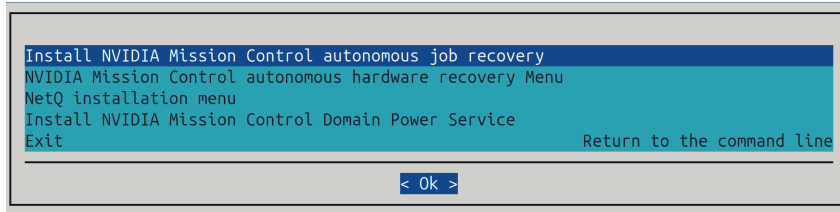


Figure 2.1: NetQ deployment: NetQ installation selection

2. The Kubernetes cluster where NetQ is to be installed is chosen (figure 2.2):

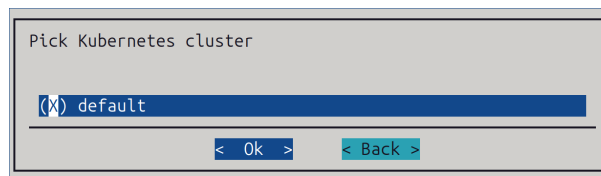


Figure 2.2: NetQ deployment: choosing a Kubernetes cluster

3. The node category for the nodes where NetQ is to be installed is chosen (figure 2.3):

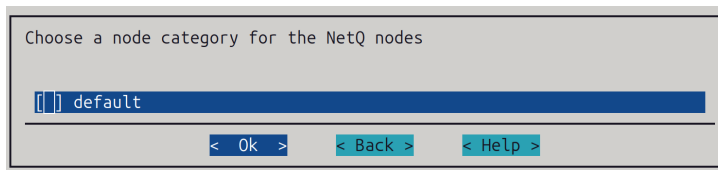


Figure 2.3: NetQ deployment: choosing a node category

4. If no category is set, then 3 nodes can be chosen (figure 2.4):

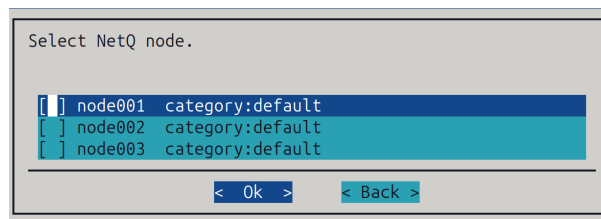


Figure 2.4: NetQ deployment: choosing 3 nodes

5. The NetQ overlay name and priority are set. In most cases, there is no need to change the suggested values (figure 2.5):

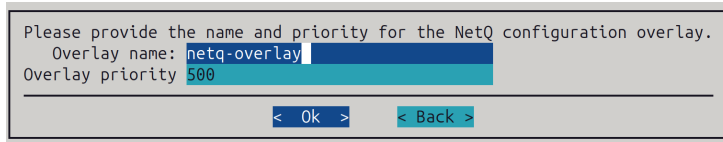


Figure 2.5: NetQ deployment: choosing an overlay

6. A virtual IP address is set for NetQ access (figure 2.6):

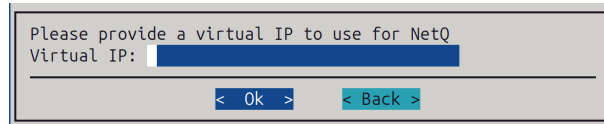


Figure 2.6: NetQ deployment: setting a virtual IP address

7. The paths to the NetQ tarball and Debian packages are set (figure 2.7):

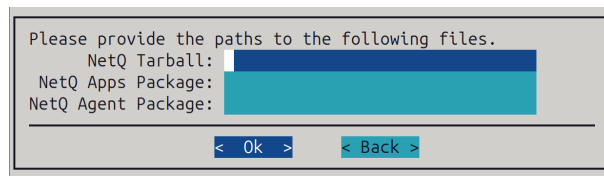


Figure 2.7: NetQ deployment: setting paths to the installation packages

8. The NetQ deployment mode is set (figure 2.8):

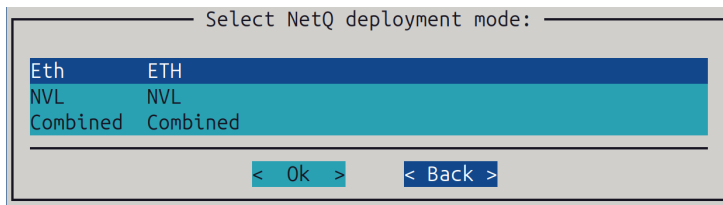


Figure 2.8: NetQ deployment: setting the NetQ deployment mode

9. The Kong (NMX API) read/write password is set, if NVL Mode has been chosen (figure 2.9):

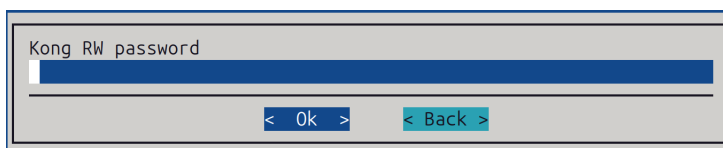


Figure 2.9: NetQ deployment: setting the read/write password

10. The Kong (NMX API) read-only password is set, if NVL Mode has been chosen (figure 2.10):

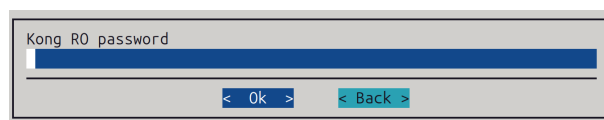


Figure 2.10: NetQ deployment: setting the read-only password

11. The storage path for Longhorn (a replicated block storage system for Kubernetes) is set (figure 2.11):

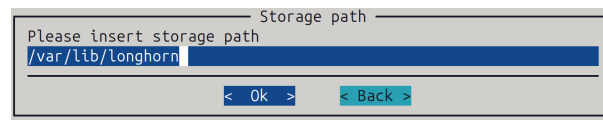


Figure 2.11: NetQ deployment: setting the storage path for Longhorn

12. The configuration is saved and deployed (figure 2.12):

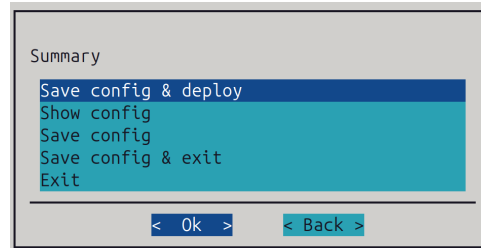


Figure 2.12: NetQ deployment: saving and deploying the configuration

Notes:

- Three nodes must be selected for NetQ deployment.
- If the selected nodes do not meet the necessary hardware requirement, then a warning is displayed.
- Kong passwords must be at least 8 characters long.

Post Installation Validation

The pods can be examined to validate NetQ:

```
kubectl get pods -A
```

All pods should be in a Running or Complete state.

A connection can be made using the virtual IP address and a URL in the form:

```
https://<VirtualIP>/nmx/swag/index.html
```

To log in, the user name is `rw-user` for read/write access, or `ro-user` for read-only access. The password is the password that was set during the installation.

Uninstall NetQ

A NetQ installation can be uninstalled by running `cm-mission-control-setup` and selecting `NVIDIA Mission Control NetQ uninstallation` (figure 2.13):

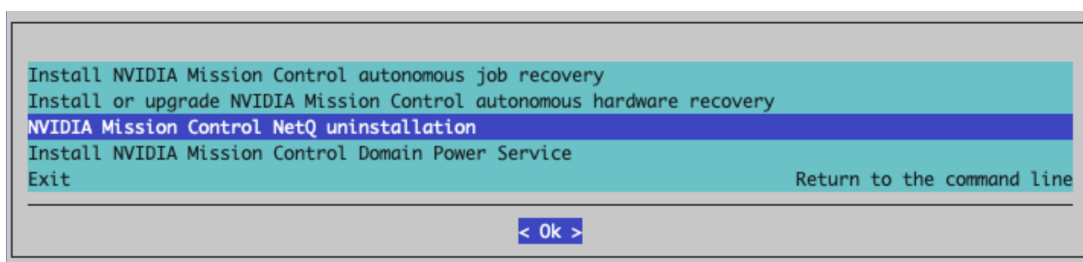


Figure 2.13: NetQ uninstallation: selecting the uninstallation

2.3.1 NMX Manager Metrics Forwarding Via BCM And prometheusmetricforwarders



Prometheus servers can be used to pick up NMX Manager metrics via Prometheus exporters. However in a typical BCM configuration, such as in a BCM type 1 network (section 3.3.9 of the *Installation Manual*), the Prometheus exporter is behind the BCM head node firewall. This means that by default an external Prometheus server cannot reach the exporter endpoint.

In such a case the `prometheusmetricforwarders` submode can be used to define the endpoint URL and HTTP method that the exporter uses. This forwards the exporter endpoint to the external network interface of the head node. The external Prometheus server can then reach the forwarded exporter endpoint.

Example

```
[basecm11->partition[base]->nmxmsettings->prometheusmetricforwarders[all]]% show
Parameter                               Value
-----
Name                                     all
Revision
Urls                                     http://10.140.0.41:9091/metrics
Method                                   GET
```

The corresponding Base View navigation path for configuring Prometheus metrics forwarding is using:

Cluster > Settings > BMC Settings > NMX settings >  > Prometheus metric forwarders
> 

3 Rack Management And Management Policies

3.1 Introduction

Rack management is the process of using tools and software to organize and manage the infrastructure components and racks in a data center.

A DGX GB200 system can be thought of as a “disaggregated supercomputer in a rack”. The cluster administrator can manage the individual nodes in the traditional way, but BCM rack management features introduced in BCM version 11 allow the GB200 nodes to be managed more conveniently as unified NVL domains.

3.2 Basic Rack Management

Basic rack management is covered in section 3.16 of the *Administrator Manual*, The commands in rack mode that are discussed there include:

- The `list` command, which can give useful information on how the devices in racks are organized, if the attributes of racks in rack mode have been set up in a meaningful way, An example is given in section 4.8.1.
- The `display` command, which is useful for seeing the position of where devices are located in the rack, if the cluster administrator has recorded the device positions in the rack, An example is given in section 4.8.3.
- The `addrackposition` command (page 187 of the *Administrator Manual*) was introduced in BCM version 11. It allows rack positions in a cluster to be filled more easily than with a looping script.
- The `rackoverview` command (page 183 of the *Administrator Manual*) was also introduced in BCM version 11. It allows rack positions in a cluster to be overviewed conveniently. It shows information about the types of entities in a rack. It also then lists some more detailed information about some of the entity types.

For a DGX GB200 platform that has been configured, the output of the `rackoverview` command looks similar to the following for a specified rack (here it is a rack named `b05` that has 18 nodes, 9 switches, and 8 power shelves):

Example

```
[a03-p1-head-01->rack]% rackoverview b05
```

Type	Up	Down	Closed	Total
Nodes	18	0	0	18

DPU nodes	0	0	0	0
Managed switches	0	0	0	0
NVLink switches	9	0	0	9
Power shelves	8	0	0	8
Devices	0	0	0	0
Cores	2,592	-	-	2,592
GPUs	72	-	-	72

Name	Value
-----	-----
User CPU	0.03%
System CPU	0.07%
Idle CPU	99.9%
Other CPU	0.0%
Memory used	1.09 TiB (3.75%)
Memory unused	28.4 TiB (97.9%)
Memory total	29.0 TiB
Total GPU utilization	0 W
Total GPU power usage	11.6 KW
Total GPU NVlink bandwidth	0 W
Average GPU temperature	30.9028 C
Total CPU power usage	1.60 KW
Average CPU temperature	48.6361 C

Node	GPU	Utilization	Temperature	Power usage	Memory used...
-----	-----	-----	-----	-----	-----
b05-p1-dgx-05-c01	gpu0	0.0%	30 C	167.184 W	0 B ...
b05-p1-dgx-05-c01	gpu1	0.0%	31 C	154.726 W	0 B ...
b05-p1-dgx-05-c01	gpu2	0.0%	32 C	154.894 W	0 B ...
b05-p1-dgx-05-c01	gpu3	0.0%	31 C	170.649 W	0 B ...
b05-p1-dgx-05-c02	gpu0	0.0%	31 C	162.994 W	0 B ...
b05-p1-dgx-05-c02	gpu1	0.0%	31 C	151.328 W	0 B ...
b05-p1-dgx-05-c02	gpu2	0.0%	31 C	151.149 W	0 B ...
b05-p1-dgx-05-c02	gpu3	0.0%	31 C	170.537 W	0 B ...
b05-p1-dgx-05-c03	gpu0	0.0%	31 C	166.664 W	0 B ...
b05-p1-dgx-05-c03	gpu1	0.0%	31 C	154.725 W	0 B ...
b05-p1-dgx-05-c03	gpu2	0.0%	31 C	151.047 W	0 B ...
b05-p1-dgx-05-c03	gpu3	0.0%	31 C	170.125 W	0 B ...
b05-p1-dgx-05-c04	gpu0	0.0%	31 C	170.642 W	0 B ...
b05-p1-dgx-05-c04	gpu1	0.0%	31 C	155.034 W	0 B ...
b05-p1-dgx-05-c04	gpu2	0.0%	31 C	152.078 W	0 B ...
b05-p1-dgx-05-c04	gpu3	0.0%	31 C	166.551 W	0 B ...
b05-p1-dgx-05-c05	gpu0	0.0%	31 C	170.516 W	0 B ...
b05-p1-dgx-05-c05	gpu1	0.0%	31 C	158.7 W	0 B ...
b05-p1-dgx-05-c05	gpu2	0.0%	31 C	158.7 W	0 B ...
b05-p1-dgx-05-c05	gpu3	0.0%	31 C	174.413 W	0 B ...
b05-p1-dgx-05-c06	gpu0	0.0%	30 C	166.44 W	0 B ...
b05-p1-dgx-05-c06	gpu1	0.0%	30 C	154.628 W	0 B ...
b05-p1-dgx-05-c06	gpu2	0.0%	31 C	154.865 W	0 B ...
b05-p1-dgx-05-c06	gpu3	0.0%	31 C	170.423 W	0 B ...
b05-p1-dgx-05-c07	gpu0	0.0%	31 C	164.078 W	0 B ...
b05-p1-dgx-05-c07	gpu1	0.0%	31 C	158.303 W	0 B ...
b05-p1-dgx-05-c07	gpu2	0.0%	31 C	150.958 W	0 B ...
b05-p1-dgx-05-c07	gpu3	0.0%	31 C	170.537 W	0 B ...
b05-p1-dgx-05-c08	gpu0	0.0%	31 C	166.661 W	0 B ...
b05-p1-dgx-05-c08	gpu1	0.0%	31 C	155.875 W	0 B ...
b05-p1-dgx-05-c08	gpu2	0.0%	31 C	154.726 W	0 B ...
b05-p1-dgx-05-c08	gpu3	0.0%	32 C	170.311 W	0 B ...
b05-p1-dgx-05-c09	gpu0	0.0%	31 C	166.331 W	0 B ...

b05-p1-dgx-05-c09	gpu1	0.0%	31 C	155.469 W	0 B	...
b05-p1-dgx-05-c09	gpu2	0.0%	30 C	154.751 W	0 B	...
b05-p1-dgx-05-c09	gpu3	0.0%	32 C	175.429 W	0 B	...
b05-p1-dgx-05-c10	gpu0	0.0%	30 C	174.181 W	0 B	...
b05-p1-dgx-05-c10	gpu1	0.0%	31 C	158.523 W	0 B	...
b05-p1-dgx-05-c10	gpu2	0.0%	31 C	152.808 W	0 B	...
b05-p1-dgx-05-c10	gpu3	0.0%	31 C	166.551 W	0 B	...
b05-p1-dgx-05-c11	gpu0	0.0%	30 C	166.704 W	0 B	...
b05-p1-dgx-05-c11	gpu1	0.0%	31 C	154.726 W	0 B	...
b05-p1-dgx-05-c11	gpu2	0.0%	31 C	155.034 W	0 B	...
b05-p1-dgx-05-c11	gpu3	0.0%	30 C	170.679 W	0 B	...
b05-p1-dgx-05-c12	gpu0	0.0%	31 C	170.424 W	0 B	...
b05-p1-dgx-05-c12	gpu1	0.0%	31 C	155.601 W	0 B	...
b05-p1-dgx-05-c12	gpu2	0.0%	31 C	150.892 W	0 B	...
b05-p1-dgx-05-c12	gpu3	0.0%	31 C	166.806 W	0 B	...
b05-p1-dgx-05-c13	gpu0	0.0%	31 C	166.766 W	0 B	...
b05-p1-dgx-05-c13	gpu1	0.0%	30 C	151.894 W	0 B	...
b05-p1-dgx-05-c13	gpu2	0.0%	31 C	151.22 W	0 B	...
b05-p1-dgx-05-c13	gpu3	0.0%	31 C	170.205 W	0 B	...
b05-p1-dgx-05-c14	gpu0	0.0%	31 C	172.778 W	0 B	...
b05-p1-dgx-05-c14	gpu1	0.0%	31 C	162.152 W	0 B	...
b05-p1-dgx-05-c14	gpu2	0.0%	31 C	150.857 W	0 B	...
b05-p1-dgx-05-c14	gpu3	0.0%	31 C	166.33 W	0 B	...
b05-p1-dgx-05-c15	gpu0	0.0%	31 C	166.518 W	0 B	...
b05-p1-dgx-05-c15	gpu1	0.0%	31 C	154.729 W	0 B	...
b05-p1-dgx-05-c15	gpu2	0.0%	31 C	154.1 W	0 B	...
b05-p1-dgx-05-c15	gpu3	0.0%	31 C	170.537 W	0 B	...
b05-p1-dgx-05-c16	gpu0	0.0%	30 C	174.067 W	0 B	...
b05-p1-dgx-05-c16	gpu1	0.0%	30 C	158.385 W	0 B	...
b05-p1-dgx-05-c16	gpu2	0.0%	31 C	151.157 W	0 B	...
b05-p1-dgx-05-c16	gpu3	0.0%	31 C	166.77 W	0 B	...
b05-p1-dgx-05-c17	gpu0	0.0%	31 C	166.161 W	0 B	...
b05-p1-dgx-05-c17	gpu1	0.0%	31 C	158.62 W	0 B	...
b05-p1-dgx-05-c17	gpu2	0.0%	31 C	151.058 W	0 B	...
b05-p1-dgx-05-c17	gpu3	0.0%	31 C	169.704 W	0 B	...
b05-p1-dgx-05-c18	gpu0	0.0%	31 C	166.441 W	0 B	...
b05-p1-dgx-05-c18	gpu1	0.0%	31 C	155.164 W	0 B	...
b05-p1-dgx-05-c18	gpu2	0.0%	31 C	150.758 W	0 B	...
b05-p1-dgx-05-c18	gpu3	0.0%	31 C	169.752 W	0 B	...

Switch	Utilization	Temperature	Power usage	Fan speed	Links active	L...
B05-P1-NVSW-01	30.2%	0 C	0 W	0 RPM	0	0
B05-P1-NVSW-02	27.5%	0 C	0 W	0 RPM	0	0
B05-P1-NVSW-03	0.0%	0 C	0 W	0 RPM	0	0
B05-P1-NVSW-04	27.6%	0 C	0 W	0 RPM	0	0
B05-P1-NVSW-05	27.6%	0 C	0 W	0 RPM	0	0
B05-P1-NVSW-06	27.5%	0 C	0 W	0 RPM	0	0
B05-P1-NVSW-07	27.5%	0 C	0 W	0 RPM	0	0
B05-P1-NVSW-08	27.5%	0 C	0 W	0 RPM	0	0
B05-P1-NVSW-09	27.6%	0 C	0 W	0 RPM	0	0

Power shelf	Input power	Output power	Temperature	Fan speed	Active PSU	To...
B05-P1-PWR-01	3.7 KW	3.5 KW	43.3333 C	6.7 KRPM	6	6
B05-P1-PWR-02	3.11 KW	2.95 KW	42.4 C	6.7 KRPM	6	6
B05-P1-PWR-03	3.7 KW	3.5 KW	44.1667 C	6.7 KRPM	6	6
B05-P1-PWR-04	3.8 KW	3.6 KW	44 C	6.7 KRPM	6	6
B05-P1-PWR-05	3.6 KW	3.4 KW	45.5 C	6.7 KRPM	6	6

B05-P1-PWR-06	3.8 KW	3.6 KW	47.1667 C	6.7 KRPM	6	6
B05-P1-PWR-07	3.5 KW	3.4 KW	46 C	6.7 KRPM	6	6
B05-P1-PWR-08	3.6 KW	3.5 KW	47 C	6.7 KRPM	6	6
[a03-p1-head-01->rack]%						

3.3 Introduction To Advanced Features In Rack Management

Over time, clusters have become more powerful and their cooling requirements have become more demanding. Liquid cooling technology has therefore become more common over time.

The monitoring associated with liquid cooling has also developed. Monitoring of the electrical supply, fans, component temperatures, and coolant values such as flow and and pressure has become popular. Coolant monitoring in BCM includes an ability to detect leaks and then take actions. An NVIDIA-conforming BMS (section 3.3.1) provides some of these monitoring and managing features, including features associated with liquid cooling and electrical isolation.

The cluster administrator can monitor these advanced features using mostly rack mode commands (section 3.3.2).

3.3.1 How BMS And BCM Work Together

A Building Management System (BMS) is a software used to manage IT assets in a building. A BMS software that can integrate with BCM, is known as a *conforming BMS*. The conformance is to a reference design for data center planning (NVIDIA Controls and Monitoring Reference Design), available to some NVIDIA product users.

Since BCM version 11, BCM integrates with a conforming BMS for leak detection and control in the data center (DC). The *NVIDIA Mission Control Software Installation Guide* describes the implementation of the integration between BCM and a conforming BMS at:

<https://docs.nvidia.com/mission-control/docs/nmc-software-installation-guide/2.0.0/integration-of-bms-with-bcm.html>

The guide is also meant to serve as a guide to NVIDIA customers who may use different BMS solutions.

BCM communicates with a conforming BMS. Together they monitor the DC. The integration works with DGX GB200 systems that are liquid-cooled, as well as other systems.

In a DC, a conforming BMS is able to detect most of the detected leak-related events. The remaining detected leak-related events, from the in-tray BMCs (compute tray and switch tray), are dealt with by BCM.

BCM uses the MQTT protocol for two-way communication with the conforming BMS, using a BCM service `cm-mqtt`.

The BMS sends BCM information about the inventory of the DC, and about the leak-related events it knows about (non-tray events). MQTT topics are created, and structured in a way that BCM can parse.

BCM in turn sends the BMS instructions on liquid management or power management, according to defined policies on events such as leaks or power outages (section 4.7).

Building Management System Configuration

BMS integration can be configured in BCM in partition mode in `cmsh` and Base View.

The type of BMS can be configured in partition mode:

Example

```
root@basecm11:~# cmsh
[basecm11]% partition
[basecm11->partition[base]]% set bms nvidia-conforming\ bms; commit
```

In the preceding example, the BMS is set to an NVIDIA-conforming BMS. The possible types of BMS configuration can be:

- NVIDIA-conforming BMS
- pipe
- file
- URL

If `bms` is set to the URL of a BMS server, for example `http://bmsserver/path`, then credentials for it can then be set as follows:

Example

```
root@basecm11:~# cmsh
[basecm11]% partition
[basecm11->partition[base]]% set bms url
[basecm11->partition*[base*]]% set bmscertificate <BMS certificate>
[basecm11->partition*[base*]]% set bmspath http://bmsserver/path
[basecm11->partition*[base*]]% set bmsprivatekey <BMS private key>
[basecm11->partition*[base*]]% commit
```

In the following example, an `mqtt` role is assigned in the `mqtt` configuration overlay. The overlay specifies where the `cm-mqtt` service should run. In this case it runs on all the head nodes:

Example

```
root@basecm11:~# cmsh
[basecm11->partition[base]]% configurationoverlay
[basecm11->configurationoverlay]% add mqtt
[basecm11->configurationoverlay*[mqtt*]]% set allheadnodes yes
```

Within the role assignment, the administrator specifies the hostname or IP address of where the BMS server runs, and where the `cm-mqtt` service pulls data from:

```
[basecm11->configurationoverlay*[mqtt*]]% roles
[basecm11->configurationoverlay*[mqtt*]->roles]% assign mqtt
[basecm11->configurationoverlay*[mqtt*]->roles*[mqtt*]]% show
Parameter                               Value
-----
Name                                     mqtt
Revision
Type                                     MQTTRole
Add services                             yes
Servers                                  <0 in submode>
CA certificate path                      /cm/local/apps/cmd/pythoncm/lib/python3.12/site-packages/\
pythoncm/etc/cacert.pem
Private key path                          /cm/local/apps/cmd/cm-mqtt/etc/mqtt.key
Certificate path                          /cm/local/apps/cmd/cm-mqtt/etc/mqtt.pem
Write named pipe path                    /var/spool/cmd/mqtt.pipe
[basecm11->configurationoverlay*[mqtt*]->roles*[mqtt*]]% servers
[basecm11->configurationoverlay*[mqtt*]->roles*[mqtt*]->servers]% list
Server (key)    Port    Disabled
-----
[basecm11->configurationoverlay*[mqtt*]->roles*[mqtt*]->servers]% add mqhost
[basecm11->configurationoverlay*[mqtt*]->roles*[mqtt*]->servers*[mqhost*]]% show
Parameter                               Value
-----
```

```

Revision
Server                mqhost
Port                  1883
Topic                 BCM/#
Disabled              no
Username
Password              < not set >
Transport             tcp
Protocol              v3.1.1
Certificate required   yes
CA certificate
Certificate
Private key
[basecm11->configurationoverlay[mqtt*]->roles[mqtt*]->servers[mqhost*]]% set certificate
...

```

A username, password, and service certificate can be set. These allow BCM to communicate with the BMS securely:

```

[basecm11->configurationoverlay[mqtt]->roles[mqtt]->servers[mqhost]]% show
Parameter                Value
-----
Revision
Server                mqhost
Port                  1883
Topic                 BCM/#
Disabled              no
Username              cronus
Password              *****
Transport             tcp
Protocol              v3.1.1
Certificate required   yes
CA certificate
Certificate
Private key
[basecm11->configurationoverlay[mqtt]->roles[mqtt]->servers[mqhost]]%

```

If the cluster manager has correctly established communication with the BMS, then the status of the BMS (metrics for power circuit, CDUs, electrical, and liquid cooling) can be seen from within `cmsh` (section 4.7).

Example

```

[basecm11]% powercircuit overview
Power circuit  Power      Current    Current limit  Current phase 1  Current phase 2 ...
-----
RPP-B12-3     14.0 KW   26.1118 A  0 A             17.7822 A       26.1118 A       ...
RPP-B14-3     15.3 KW   29.6178 A  0 A             17.7419 A       29.6178 A       ...
RPP-B21-5     7.5 KW    11.2255 A  0 A             10.4872 A       10.5412 A       ...
...

```

3.3.2 Commands For Monitoring Advanced Features In Rack Management

The `electricaloverview` Command

The `electricaloverview` command within rack mode gives a convenient overview of the electrical supply to the cluster.

The power used and the power budget of a specified rack (B05 in the following example) can be seen:

Example

```
[basecm11->rack[B05]]% electricaloverview
Rack          Power used      Power budget    Isolation status
-----
B05           28.5 kW         135 kW          0
```

If no rack is specified, then the command displays the information for all racks.

The liquidcoolingoverview Command

The `liquidcoolingoverview` command can be run from within rack mode, and conveniently displays coolant metrics:

```
[basecm11->rack]% liquidcoolingoverview a05
      Supply      Return      Flow      Differential      Isolation
Rack temperature temperature rate      pressure      status
-----
A05  26.03 C      30.78 C      70.938 LPM  31 KPa          0
```

If no rack is specified, then the command displays the information for all racks.

The leakdetectionoverview Command

The `leakdetectionoverview` command within rack mode gives a convenient overview about sensor readings related to leak detection in BCM and the electrical isolation status.

The output from the command applied to a specific rack looks similar to the following:

```
[basecm11->rack]% leakdetectionoverview a05
      Leak      Leak      Leak      Liquid      Liquid      Liquid      Electrical      Electrical      Electrical
Rack detected tray detected sensor request request isolation request request isolation
-----
A05  0          0          0          0          0          0          0          0          0
[basecm11->rack]%
```

If no rack is specified, then the command displays the information for all racks.

The leakinfo Command

The `leakinfo` command, is related to leakage detection, but it is not a rack mode command. It is a device mode command. It is noted in this section for completeness. It can be run at device mode level to list all leaks detected in the cluster.

Typically, and ideally, running the command shows something similar to:

```
[basecm11->device]% leakinfo
Hostname          Timestamp Severity  Info
-----
[basecm11->device]%
```

If there are issues on the devices, then the output may look similar to:

```
[basecm11->device]% leakinfo
Hostname          Timestamp Severity  Info
-----
a05-dgx-01-c09   15:09:23   large      The state of resource `Chassis_0_LeakDete...
a07-dgx-03-c16   15:38:48   large      The state of resource `Chassis_0_LeakDete...
b05-dgx-05-c08   09:11:06   large      The state of resource `Chassis_0_LeakDete...
b05-dgx-05-c10   09:11:05   large      The state of resource `Chassis_0_LeakDete...
```

```
b06-dgx-06-c17 11:49:30 large The state of resource `Chassis_0_LeakDete...
[basecm11->device]%
```

Running `help leakinfo` lists further command options.

The `isolationrequestinfo` Command

If a leak is detected, then BCM receives the report. BCM then:

- powers off the node that is leaking
- powers off the rack if multiple nodes are leaking
- isolates the rack to prevent other nodes suffering the same fate

Isolation means that power no longer runs through the leaking racks.

The isolation status of a rack can be viewed with the `isolationrequestinfo` command, which normally shows something similar to:

Example

```
[basecm11->rack]% isolationrequestinfo
Rack      Timestamp  Kind      Value      Info
-----
[basecm11->rack]%
```

During a leakage event, the output may show something similar to:

Example

```
[basecm11->rack]% isolationrequestinfo
Rack  Timestamp Kind Value Info
-----
B05   03:40:01 4      Default building policy/Rack
[basecm11->rack]%
```

3.3.3 Managing Leaks With rules In `leakactionpolicies`

The cluster administrator has the freedom to construct actions based on triggers based on leak detection measurables. While that is powerful, that can be quite labor-intensive for cluster administrators.

BCM provides a simpler way to deal with leaks, based on leak action policies. Leak action policies are a collection of rules that carry out actions to be followed with leaks are detected. Racks play an important part in these policies and rules.

The policies use predefined scopes, or groupings. The names of these groupings are rules. The groupings are listed next, in order from largest to smallest, along with their associated descriptions:

- **Building:** a building is made up of rooms
- **Room:** A room has rows of racks in it.
- **Row:** A row of racks has individual racks in it.
- **Rack:** An individual rack has devices in it.
- **Device:** The smallest possible grouping

The policies can be listed in the `leakactionpolicies` submode, under `partition` mode.

Example

```
[basecm11->partition[base]->leakactionpolicies]% list
Name (key)          Rules
-----
Default building policy  Building, Device, Rack, Room, Row
Default device policy    Device
Default rack policy      Device, Rack
Default room policy      Device, Rack, Room, Row
Default row policy       Device, Rack, Row
```

The rules used by each policy are also shown, under the Rules column in the preceding example.

Each rule in a policy is a copy of a template. By default the rules per policy are identical.

Just the rules used by a specific policy can also be viewed:

Example

```
[basecm11->partition[base]->leakactionpolicies]% use default building policy
[basecm11->partition[base]->leakactionpolicies[Default building policy]]% show
Parameter          Value
-----
Name                Default building policy
Revision
Rules               Building, Device, Rack, Room, Row
```

Within a policy, the value of the settings for these rules can be listed:

Example

```
[basecm11->...leakactionpolicies[Default building policy]->rules]% list
Name (key)  Scope      Power Power  Electrical Liquid  Minimal Maximal
           off      off rack isolation isolation devices devices
-----
Building   building  yes  yes   no      no      11     100000
Device     device    yes  no    no      no
Rack       rack      yes  yes   yes     yes     3      100
Room       room      yes  yes   no      no      7      100000
Row        row       yes  yes   no      no      5      100
```

The rules for each policy can be modified independently for each policy, so that, for example the Rack rule values may differ between Default row policy and Default building policy. However, a sensible cluster administrator is not expected to carry out such a modification.

Another way of phrasing this is: The values of the rules can be changed from the defaults. In that case, a sensible administrator is expected to keep the values synchronized across the rules in the leak action policies.

In the preceding example:

If the minimal devices criterion is met for leaking for each rule in a policy, then the values for the column header are applied to the rules of that policy.

So, for the policy rules in the preceding example, when leaking is detected:

- If only one or two devices are detected as leaking, then only the devices involved are powered off.
- If, in one rack, more than 3 devices, but less than 5 devices, are detected as leaking, then in addition to the device rules being applied, the entire rack is powered off. Isolation is also carried out for the electrical power and liquid coolant.

- If more devices meet the conditions, so a larger scope is affected by leaking, then the rules for the larger scope are also applied.

The rules for a specific policy can be removed. The row rule for Default building policy can be removed with, for example:

```
[basecm11->partition[base]->leakactionpolicies[Default building policy]]% rules
[basecm11->...leakactionpolicies[Default building policy]->rules]% remove <TAB><TAB>
building device rack room row
[basecm11->...leakactionpolicies[Default building policy]->rules]% remove row; commit
```

Custom Rules For leakactionpolicies

The names for the rules (Device, Rack, Row ...) are really just default labels, and a sensible cluster administrator is expected to have the description associated with the name match the physical reality. Creating a custom rule may help with this.

Custom rules can be added by the cluster administrator, by first adding another policy:

```
[basecm11->partition[base]->leakactionpolicies[Default building policy]]% add mypolicy
[basecm11->partition*[base*]->leakactionpolicies*[mypolicy*]]% list
Name (key)          Rules
-----
Default building policy  Building, Device, Rack, Room, Row
Default device policy    Device
Default rack policy      Device, Rack
Default room policy      Device, Rack, Room, Row
Default row policy       Device, Rack, Row
mypolicy               none
[basecm11->partition*[base*]->leakactionpolicies*[mypolicy*]]% show
Parameter          Value
-----
Name                mypolicy
Revision
Rules               none
```

By default, there are no rules in the new policy:

```
[basecm11->partition*[base*]->leakactionpolicies*[mypolicy*]]% rules
[basecm11->partition*[base*]->leakactionpolicies*[mypolicy*]->rules]% list

Name (key)  Scope      Power off  Power off rack  Electrical  Liquid  Minimal  Maximal
-----  -----  -
isolation  isolation  devices  devices
```

The cluster administrator may have a testing room in which to keep a rack. In that case, the following might be added:

```
[basecm11->partition*[base*]->leakactionpolicies*[mypolicy*]->rules]% add testroom
[basecm11->partition*[base*]->leakactionpolicies*[mypolicy*]->rules*[testroom*]]% show
Parameter          Value
-----
Name                testroom
Revision
Scope              rack
Disabled           no
Minimal devices    1
Maximal devices    100
Grace period       0s
Correlation window 5m
Power off          yes
```

```
Power off rack          no
Electrical isolation    no
Liquid isolation        no
Minimal severity        1
Maximal severity        100
```

Adding a rule adds a default with the preceding rack-based rule values.

The rule values can be modified using the `set` command.

Running `set` without any options provides a help text explaining what can be set:

```
[basecm11->partition*[base*]->leakactionpolicies*[mypolicy*]->rules]% set
```


4 BCM Power Shelf Integration

The NVIDIA GB200 server platforms use power shelves to distribute power in racks. Several power shelves are used per rack. Each power shelf takes up one rack unit. Each power shelf comprises several PSUs (Power Supply Units).

The reference design for an NVIDIA GB200 rack has 6 power shelves, and uses 6 PSUs in each power shelf. Each PSU is rated up to 5.5kW, so that each power shelf is rated up to 33kW.

This chapter considers an NVIDIA GB200 server platform that has been configured to work with BCM.

4.1 Power Shelf Listing And Overview

The `list -t powershelf` command in device mode can be used to list the power shelves of a configured cluster

Power shelves are configured as devices on an IPMI network, as indicated by the following example (some rows ellipsized):

Example

```
[basecm11->device]% list -t powershelf
Type          Hostname (key)  MAC                IP           Network  Status
-----
PowerShelf    B05-P1-PWR-01  00:18:23:0C:1E:90  7.241.1.47  rfnet    [ UP ]
...
PowerShelf    B05-P1-PWR-08  00:18:23:0C:1E:85  7.241.1.54  rfnet    [ UP ]
PowerShelf    a05-p1-pwr-01  00:18:23:0C:1E:84  7.241.1.8   rfnet    [ UP ]
...
PowerShelf    a05-p1-pwr-08  00:18:23:0C:1E:8F  7.241.1.80  rfnet    [ UP ]
PowerShelf    b06-p1-pwr-01  00:18:23:0C:40:2A  7.241.1.55  rfnet    [ UP ]
...
PowerShelf    b06-p1-pwr-08  00:18:23:0C:40:99  7.241.1.62  rfnet    [ UP ]
```

The `powershelfoverview` command displays an overview of a particular power shelf, showing the measurements from its PSUs (6 in the following example):

Example

```
[basecm11->device*[B05-P1-PWR-01]]% powershelfoverview
Input  Input  Input  Rail  Rail  Rail  Fan
Supply power current voltage power current voltage Temperature speed Healthy
-----
1      679 W  2.98 A  240.75 V  652 W  13.06 A  49.85 V  44 C  6.7 KRPM  PASS
```

2	632 W	2.8 A	240.75 V	607 W	12.18 A	49.87 V	44 C	6.6 KRPM	PASS
3	599 W	2.64 A	240.75 V	575 W	11.48 A	49.87 V	43 C	6.8 KRPM	PASS
4	599 W	2.63 A	240.75 V	575 W	11.51 A	49.89 V	43 C	6.8 KRPM	PASS
5	617 W	2.76 A	240.5 V	574 W	11.5 A	49.87 V	43 C	6.9 KRPM	PASS
6	609 W	2.64 A	240.5 V	584 W	11.7 A	49.87 V	43 C	6.6 KRPM	PASS

4.2 Power Shelves Networking Configuration

The network interface settings for a specific power shelf can be managed with its Redfish interface (section 3.7.1 of the *Administrator Manual*). For example, a power shelf, here named B05-P1-PWR-01, can have the following values set with the `set` command:

Example

```
[basecm11->device[B05-P1-PWR-01]->interfaces[rf0]]% show
Parameter                               Value
-----
Revision
Type                                     bmc
Network device name                       rf0
Network                                   rfnet
IP                                         7.241.1.47
DHCP                                       no
Alternative Hostname
Additional Hostnames
Switch ports
Start if                                  always
BringUpDuringInstall                      no
On network priority                       60
Bootable                                  no
MAC                                        00:18:23:0C:1E:90
Speed
Card Type
```

Redfish metrics, which are the ones prefixed by RF_ (as seen in section 11.8), are sampled only when the BMC interface starts with rf0, rf1, ...

4.3 Access Configuration For The Power Management Controller

Access to the Power Management Controller (PMC) is configured in the BMC settings:

Example

```
[basecm11->device[B05-P1-PWR-01]->bmcsettings]% show
Parameter                               Value
-----
Revision
User name                                 root
Password                                  *****
User ID                                    -1
Power reset delay                         0s
Extra arguments
Privilege                                  administrator
```

Authentication allows NVIDIA Mission Control to monitor the power shelves using Redfish sampling.

4.4 Power Shelf Settings Overview

An overview of the settings for a particular power shelf can be viewed with the `show` command:

Example

```
[basecm11->device[B05-P1-PWR-01]]% show
Parameter                               Value
-----
Hostname                                B05-P1-PWR-01
IP                                        7.241.1.47
Network                                  rfnet
Revision
Type                                      PowerShelf
Mac                                       00:18:23:0C:1E:90
Model
Activation                               Tue, 11 Feb 2025 16:45:39 PST
Rack                                       B05:6
Chassis                                   < not set >
Access Settings                           <submode>
Management network                         rfnet
Power control                             none
Custom power script
Custom power script argument
Power distribution units
Default gateway metric                    0
Switch ports                              B04-P1-00B-02:1
Interfaces                                 <1 in submode>
PMC Settings                              <submode>
Userdefined1
Userdefined2
User defined resources
Supports GNSS                             no
Custom ping script
Custom ping script argument
Partition                                  base
Part number
Serial number
Notes                                      <0B>
Prometheus metric forwarders              <0 in submode>
```

4.5 Power Shelf Metrics Overview

A listing of the power shelf metrics can be seen with some grepping (the output for PSUs 2 to 5 have been ellipsized for readability in this manual):

```
[basecm11->monitoring->measurable]% list -f name:40,parameter:9,producer
|head -2; list -f name:40,parameter:9,producer| egrep -i
'(powershel|redfish_power_shelf)'
```

name (key)	parameter	producer
AveragePowerShelfTemperature		AggregatePowerShelf
PowerShelvesClosed		ClusterTotal
PowerShelvesDown		ClusterTotal
PowerShelvesTotal		ClusterTotal
PowerShelvesUp		ClusterTotal
RF_Active_PSU		redfish_power_shelf
RF_PowerShelf_Status	1	redfish_power_shelf
...		

RF_PowerShelf_Status	6	redfish_power_shelf
RF_PowerShelf_Status	total	redfish_power_shelf
RF_Power_Supply_Energy	1	redfish_power_shelf
...		
RF_Power_Supply_Energy	6	redfish_power_shelf
RF_Power_Supply_Energy	total	redfish_power_shelf
RF_Power_Supply_FanSpeed	1	redfish_power_shelf
...		
RF_Power_Supply_FanSpeed	6	redfish_power_shelf
RF_Power_Supply_FanSpeed	average	redfish_power_shelf
RF_Power_Supply_InputCurrent	1	redfish_power_shelf
...		
RF_Power_Supply_InputCurrent	6	redfish_power_shelf
RF_Power_Supply_InputCurrent	total	redfish_power_shelf
RF_Power_Supply_InputPower	1	redfish_power_shelf
...		
RF_Power_Supply_InputPower	6	redfish_power_shelf
RF_Power_Supply_InputPower	total	redfish_power_shelf
RF_Power_Supply_InputVoltage	1	redfish_power_shelf
...		
RF_Power_Supply_InputVoltage	6	redfish_power_shelf
RF_Power_Supply_LifetimeReading_Energy	1	redfish_power_shelf
...		
RF_Power_Supply_LifetimeReading_Energy	6	redfish_power_shelf
RF_Power_Supply_OutputPower	1	redfish_power_shelf
...		
RF_Power_Supply_OutputPower	6	redfish_power_shelf
RF_Power_Supply_OutputPower	total	redfish_power_shelf
RF_Power_Supply_PowerFactor_Input	1	redfish_power_shelf
...		
RF_Power_Supply_PowerFactor_Input	6	redfish_power_shelf
RF_Power_Supply_RailCurrent	1	redfish_power_shelf
...		
RF_Power_Supply_RailCurrent	6	redfish_power_shelf
RF_Power_Supply_RailCurrent	total	redfish_power_shelf
RF_Power_Supply_RailPower	1	redfish_power_shelf
...		
RF_Power_Supply_RailPower	6	redfish_power_shelf
RF_Power_Supply_RailPower	total	redfish_power_shelf
RF_Power_Supply_RailVoltage	1	redfish_power_shelf
...		
RF_Power_Supply_RailVoltage	6	redfish_power_shelf
RF_Power_Supply_Temperature	1	redfish_power_shelf
...		
RF_Power_Supply_Temperature	6	redfish_power_shelf
RF_Power_Supply_Temperature	average	redfish_power_shelf
RF_Power_Supply_code_error	1	redfish_power_shelf
...		
RF_Power_Supply_code_error	6	redfish_power_shelf
RF_Power_Supply_message_error	1	redfish_power_shelf
...		
RF_Power_Supply_message_error	6	redfish_power_shelf
RF_Summary_Metrics_Power		redfish_power_shelf
RF_Summary_Metrics_Temperature		redfish_power_shelf
RF_Total_PSU		redfish_power_shelf
TotalPowerShelfInputCurrent		AggregatePowerShelf
TotalPowerShelfInputPower		AggregatePowerShelf
TotalPowerShelfRailCurrent		AggregatePowerShelf
TotalPowerShelfRailPower		AggregatePowerShelf

The descriptions for these metrics are as indicated in table 11.4.

4.6 Power Shelf Firmware

Power shelf firmware is managed by the `firmware` command options as described in Section 14.5.3 of the *Administrator Manual*.

4.7 Power Shelf Circuit Monitoring

A power circuit typically supplies several power shelves, based on the rated power of the circuit and power shelves. Administration and wiring convenience may also play a role in allocating a circuit to a power shelf.

So, for example, the administrator may configure a rack with 6 power shelves so that it is supplied power from one power circuit.

The power circuit breakers used by each power circuit, typically provided by a remote power panel (RPP), can have their locations noted for administrative convenience.

Example

```
[basecm11->powercircuit]% list
Name (key) Row  Room  Building  Location  Racks
-----
RPP-B12-3  1    G     Watchtower  Trantor
RPP-B14-3
RPP-B21-5
RPP-B21-6
RPP-B22-5
...
```

The cluster may have been set up to monitor the power circuit breaker for the power reaching BCM (section 3.3.1). In that case, the `overview` command typically displays the power consumption and current per

Example

```
phase. [basecm11->powercircuit]% overview RPP-B14-3
Power circuit Power  Current  Current  Current  Current  Current  Racks
-----
RPP-B14-3    15.6 KW 30.137 A 0 A      18.28 A 30.119 A 28.134 A 1
```

A value of 0 A means that the current limit is not reported.

4.8 Power Shelves And Rack Mode Commands

This section discusses some of the commands that can be useful for setting up and viewing devices from within the rack mode of `cmsh` (section 3.16 of the *Administrator Manual*), but with a focus on power shelf management.

4.8.1 The `list` Command

If the attributes of racks in rack mode have been set up in a meaningful way, then the `list` command can give useful information on how the power shelves in racks are organized:

Example

```
[basecm11->rack]% list
Name (key) Room x-Coord- y-Coord Height Devices
-----
A01 T 1 1 48
A02 T 1 2 48
A03 T 1 3 48
A04 T 1 4 48
A05 T 1 5 48 a05-p1-pwr-01..a05-p1-pwr-08,a05...
A06 T 1 6 48
A07 T 1 7 48
A08 T 1 8 48
A09 T 1 9 48
A10 T 1 10 48
A11 T 1 11 48
A12 T 1 12 48
...
```

The `help set` command describes how these values can be set.

In the preceding example, the administrator has noted that a rack, for example A05, is in a room T, and is at an (x,y) coordinate of (1,5) in that room.

The power shelf devices have also been given meaningful names: `a05-p1-pwr-01`, `a05-p1-pwr-02...`, which here has a prefix that is the rack it is in, and uses `pwr` to indicate it is a power shelf.

4.8.2 The `rackoverview` Command

The `rackoverview` command displays information about the types of entities in a rack. It also then lists some more detailed information about some of the entity types.

The power shelf one of the types that has more detailed information listed (some output truncated and ellipsized, because this section is focussing on the power shelves):

Example

```
[basecm11->rack]% rackoverview <TAB><TAB>
a01 a02 a03 a04 a05 a06 a07 a08 a09 a10 a11 a12 b01 b02 b03 b04...
[basecm11->rack]% rackoverview a05
Type Up Down Closed Total
-----
Nodes 18 0 0 18
DPU nodes 0 0 0 0
Managed switches 0 0 0 0
NVLink switches 0 9 0 9
Power shelves 8 0 0 8
Devices 0 0 0 0
Cores 2,592 - - 2,592
GPUs 72 - - 72
...

Power shelf Input power Output power Temperature Fan speed Active PSU Total PSU
-----
a05-p1-pwr-01 0 W 0 W 0 C 0 RPM 0 6
a05-p1-pwr-02 0 W 0 W 0 C 0 RPM 0 6
a05-p1-pwr-03 3.7 KW 3.6 KW 43.3333 C 6.7 KRPM 6 6
a05-p1-pwr-04 2.51 KW 2.38 KW 42.75 C 6.7 KRPM 6 6
```

```

a05-p1-pwr-05  2.83 KW  2.69 KW  49 C           6.8 KRPM  6      6
a05-p1-pwr-06  3.7 KW   3.5 KW   45 C           6.7 KRPM  6      6
a05-p1-pwr-07  3.7 KW   3.5 KW   46.3333 C     6.7 KRPM  6      6
a05-p1-pwr-08  3.8 KW   3.7 KW   47.3333 C     6.7 KRPM  6      6
[basecm11->rack]%

```

In the preceding, the A05 rack is seen to have some active power shelves, which show some power-related values.

4.8.3 The display Command

If the cluster administrator has recorded the device positions in the rack, then the `display` command in rack mode is useful for seeing the position of where power shelves are located in the rack:

Example

```

[basecm11->rack]% display |less -R
...
          A05                      B05

48
47
46
45
44
43
42  a05-p1-pwr-08          42  B05-P1-PWR-08
41  a05-p1-pwr-07          41  B05-P1-PWR-07
40  a05-p1-pwr-06          40  B05-P1-PWR-06
39  a05-p1-pwr-05          39  B05-P1-PWR-05
38
37  a05-p1-dgx-01-c18     37  b05-p1-dgx-05-c18
36  a05-p1-dgx-01-c17     36  b05-p1-dgx-05-c17
35  a05-p1-dgx-01-c16     35  b05-p1-dgx-05-c16
34  a05-p1-dgx-01-c15     34  b05-p1-dgx-05-c15
33  a05-p1-dgx-01-c14     33  b05-p1-dgx-05-c14
32  a05-p1-dgx-01-c13     32  b05-p1-dgx-05-c13
31  a05-p1-dgx-01-c12     31  b05-p1-dgx-05-c12
30  a05-p1-dgx-01-c11     30  b05-p1-dgx-05-c11
29  a05-p1-dgx-01-c10     29  b05-p1-dgx-05-c10
28  a05-p1-dgx-01-c09     28  b05-p1-dgx-05-c09
27  a05-p1-nvsw-09        27  B05-P1-NVSW-09
26  a05-p1-nvsw-08        26  B05-P1-NVSW-08
25  a05-p1-nvsw-07        25  B05-P1-NVSW-07
24  a05-p1-nvsw-06        24  B05-P1-NVSW-06
23  a05-p1-nvsw-05        23  B05-P1-NVSW-05
22  a05-p1-nvsw-04        22  B05-P1-NVSW-04
21  a05-p1-nvsw-03        21  B05-P1-NVSW-03
20  a05-p1-nvsw-02        20  B05-P1-NVSW-02
19  a05-p1-nvsw-01        19  B05-P1-NVSW-01
18  a05-p1-dgx-01-c08     18  b05-p1-dgx-05-c08
17  a05-p1-dgx-01-c07     17  b05-p1-dgx-05-c07
16  a05-p1-dgx-01-c06     16  b05-p1-dgx-05-c06
15  a05-p1-dgx-01-c05     15  b05-p1-dgx-05-c05
14  a05-p1-dgx-01-c04     14  b05-p1-dgx-05-c04
13  a05-p1-dgx-01-c03     13  b05-p1-dgx-05-c03
12  a05-p1-dgx-01-c02     12  b05-p1-dgx-05-c02
11  a05-p1-dgx-01-c01     11  b05-p1-dgx-05-c01
10

```

09	a05-p1-pwr-04	09	B05-P1-PWR-04
08	a05-p1-pwr-03	08	B05-P1-PWR-03
07	a05-p1-pwr-02	07	B05-P1-PWR-02
06	a05-p1-pwr-01	06	B05-P1-PWR-01
05		05	
04		04	
03		03	
02		02	
01		01	

4.8.4 Power Management Of Racks

For devices in general, power operation commands are described in Section 4.2 of the *Administrator Manual*. Power status responses are described on page 222 of the *Administrator Manual*.

Racks can take a while to power up through the configured sequence of components (power shelves, switches, nodes). The BUSY status can therefore be visible for a while. In this status, the power up operation can be cancelled.

A session showing power commands running on the racks of a DGX OS cluster may show output similar to the following:

Example

```
[basecm11->rack]% events off
Private events:  off
Broadcast events: off
[basecm11->rack]% power on -b nv1[001-016]
[basecm11->rack]% power status
post::powershelf ..... [  BUSY  ] nv1001
post::powershelf ..... [  BUSY  ] nv1002
post::powershelf ..... [  BUSY  ] nv1003
...
[basecm11->rack]% power status
post::switch ..... [  BUSY  ] nv1001
post::switch ..... [  BUSY  ] nv1002
post::switch ..... [  BUSY  ] nv1003
...
[basecm11->rack]% power cancel nv1001
Cancelled: 1 operations
[basecm11->rack]% power status
none ..... [CANCELLED] nv1001
post::compute ..... [  BUSY  ] nv1002
post::compute ..... [  BUSY  ] nv1003
...
```

More details on power commands use in the rack mode of cmsh can be seen with:

Example

```
[basecm11->rack]% help power
Name:
    power - Manipulate the power state of rack

Usage:
    power [OPTIONS] on
    power [OPTIONS] off
    power [OPTIONS] cancel
    power [OPTIONS] status
```

Options:

- b, --background
Run in background, output will come as events

- d, --delay <seconds>
Wait <seconds> between executing two sequential power commands. This option is ignored for the status command

- m, --mode <mode>
Mode: full(default), domain(switch and compute tray), compute, emergency(off only)

- w, --overview
Create overview instead of individual rack list

- f, --force
Force power command on devices which have been closed or are reported as leaking

- timeout <seconds>
Maximal time to wait for all result to come in

- message <message>
An optional message saved to DB for on/off/reset operations

- initial-wait-switch <seconds>
Wait the specified time before switch power on after power shelves were found UP

- initial-wait-node <seconds>
Wait the specified time before node power on after swirches were found UP

- maximal-redfish-wait <seconds>
Maximal time to wait between power shelf power on and the redfish of switch/node becoming available

- maximum-wait <seconds=30>
Maximal time to wait between for devices to enter the correct state

Examples:

power on rack01	Power on rack01
power off -m compute rack01	Power off the compute tray in rack01
power cancel	Cancel all active rack power operations
power cancel rack01	Cancel power operation on rack01
power status	Show status of all active power operations

5 NVIDIA Autonomous Job Recovery

5.1 Introduction

NVIDIA Mission Control can provide autonomous job recovery. This capability can be installed on a cluster managed by BCM.

Autonomous job recovery can

- monitor Kubernetes jobs and pods
- monitor Slurm jobs and scheduling

and analyze what is being monitored. If it detects processes going wrong, then it can take action to recover from what is wrong, with the aim of reducing downtime.

Autonomous job recovery can display its monitoring visually in a Grafana dashboard using Loki logging.

Autonomous job recovery can be installed and integrated into BCM if Kubernetes and Slurm are installed first.

The physical hardware requirements for a cluster running autonomous job recovery are:

- the nodes should have at least 16 GB of RAM
- the cluster should have at least 30 CPU cores
- the cluster should have at least 64 GB RAM on the head node
- the cluster should have at least 20GB of persistent storage

5.2 Prerequisites To Install Autonomous Job Recovery: Kubernetes Installation

To install autonomous job recovery, Kubernetes should already have been installed and integrated with BCM with appropriate components. Kubernetes installation and integration is typically done with the `cm-kubernetes-setup` utility (section 4.2 of the *Containerization Manual*).

5.2.1 Kubernetes Components Needed For Installation Of Autonomous Job Recovery

The Kubernetes deployment procedure should be carried out so that it includes at least the following components (figure 5.1):

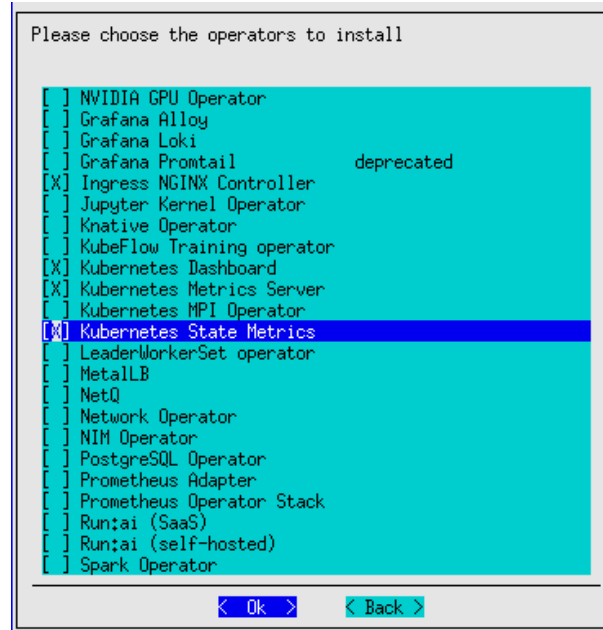


Figure 5.1: Selection of Kubernetes components for installing autonomous job recovery

- Prometheus Operator Stack
- Prometheus Adapter
- Grafana Promtail
- Kubernetes Metrics Server
- Kubernetes State Metrics
- Ingress NGINX Controller

Kyverno should not be installed at the time of writing (April 2025).

5.2.2 Settings For Kubernetes Components Used In Autonomous Job Recovery Installation

In other screens that come up with the preceding components selection, the following settings should be used:

- In the screen asking to configure the Kubernetes StorageClass, a local path storage class should be enabled, with the default NFS storage enabled (figure 5.2):

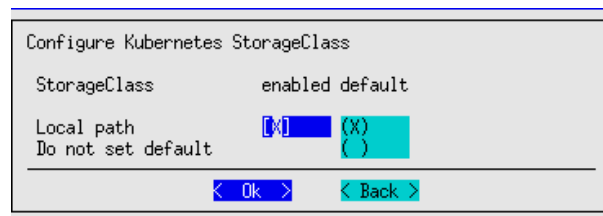


Figure 5.2: Setting the storage class

- In the screen that asks about Loki access (figure 5.3):
 - Loki should be added as the data source for Grafana

- the Loki API should be exposed for Ingress

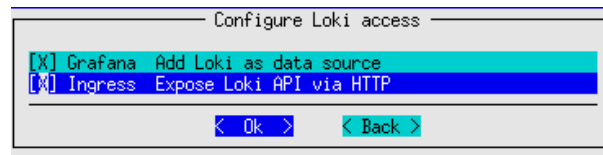


Figure 5.3: Setting up Loki as data source for Grafana, and Loki access via Ingress

- In the screen that configures logging for Grafana (figure 5.4), Grafana Promtail is recommended to have its collection disabled for `/var/log` from the Kubernetes worker nodes.

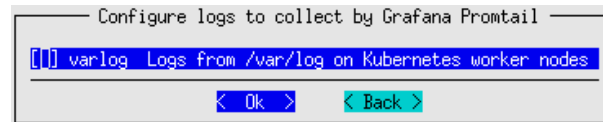


Figure 5.4: Log collection configuration for Grafana Promtail

5.3 Prerequisites To Install Autonomous Job Recovery: Slurm Installation

To install autonomous job recovery, Slurm should already have been installed and integrated with BCM. Slurm installation and integration is typically done with the `cm-wlm-setup` utility (section 7.3.2 of the *Administrator Manual*).

If Slurm is not to be run during regular use, then Slurm should still be installed before autonomous job recovery is installed, despite autonomous job recovery not being applied to Slurm during regular use. This is because autonomous job recovery installation makes use of Slurm configuration.

5.4 Autonomous Job Recovery Installation With `cm-mission-control-setup`

If Kubernetes has been installed (section 5.2), and Slurm has been installed (section 5.3), then autonomous job recovery can be installed with the `cm-mission-control-setup` utility. The `cm-mission-control-setup` utility is part of the `cm-setup` package.

5.4.1 Credentials Required To Install Autonomous Job Recovery When Running `cm-mission-control-setup`

The following credentials are required to install autonomous job recovery:

- Ingress credentials for Loki API access. This would have been set up during the `cm-kubernetes-setup` session.
- MySQL root password for the main CMDaemon database. By default this is the same as the root user password of the operating system.
- NVCR personal token for operator and container images access. The personal token is a string with the prefix: `nvapi-`

5.4.2 Running `cm-mission-control-setup`

The `cm-mission-control-setup` script should be run without options on the active head node. It brings up a TUI dialog (figure 5.5):

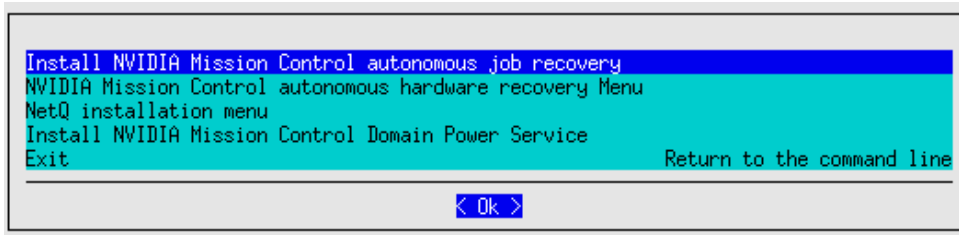


Figure 5.5: Selection from `cm-mission-control-setup` of autonomous job recovery

If there is more than one Kubernetes instance on the cluster, then a screen appears asking for one of them to be selected for the installation of autonomous job recovery.

If there is more than one Slurm WLM instance on the cluster, then a screen appears asking for one of them to be selected for the installation of autonomous job recovery.

Next, a screen appears asking for the MySQL access credentials.

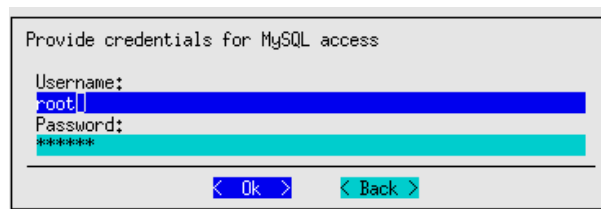


Figure 5.6: Autonomous job recovery deployment configuration: MySQL access

Next, a screen appears asking for the Helm repository credentials.

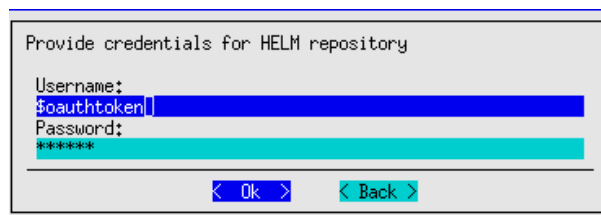


Figure 5.7: Autonomous job recovery deployment configuration: Helm repository access

Next, a screen appears asking for the credentials for Loki API access via Ingress.

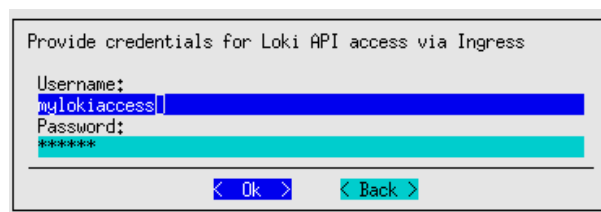


Figure 5.8: Autonomous job recovery deployment configuration: Loki API access via Ingress

The configuration settings can then be saved and deployment of autonomous job recovery is started.

Deployment can take up to 30 minutes.

5.5 Post-installation Checks

5.5.1 Grafana Access

Grafana for the Kubernetes Prometheus stack is exposed by default at:

```
https://<cluster IP address>/grafana/login
```

as set by the Kubernetes NGINX Ingress Controller.

The default administrator username and password are not changed automatically.

5.5.2 Autonomous Job Recovery Failure When The Slurm Daemon Spool Directory Is Not Cleaned Up

The `slurmd` spool directory is normally cleaned up automatically when a compute node reboots. The `slurmd` spool directory path is defined in the Slurm configuration file `slurm.conf` by the parameter `SlurmdSpoolDir`. `SlurmdSpoolDir` has a default value of `/cm/local/apps/slurm/var/spool/`.

A clean `slurmd` spool is needed after rebooting for Slurm to function correctly. If the spool directory contents persist after a reboot, then it means that a job may not resume properly.

The spool directory contents can persist in, for example, the following use case:

- when autonomous job recovery decides to reboot a node where a job is running, and
- if the job is submitted with the `--requeue` command line option, and
- it uses PMIX

In that use case, Slurm does not clean up PMIX temporary files such as `stepd.slurm.pmix.$SLURM_JOBID.0` from the default spool directory. So when the job starts on the rebooted node, `slurmstepd` cannot start the job again because the file exists.

6 NVIDIA Autonomous Hardware Recovery

6.1 Introduction

NVIDIA Mission Control can provide autonomous hardware recovery. This capability can be installed on a cluster managed by BCM.

It is designed to allow a user to carry out:

- automated baseline tests to validate hardware
- automated health checks to detect hardware failures at the tray, rack, node, and system levels
- automated break/fix workflows (guided recovery on failure)

Autonomous hardware recovery can be integrated with an NVIDIA Base Command Manager (BCM) that is running a Kubernetes instance. The integration is carried out with the BCM script `cm-mission-control-setup` (section 6.3).

The script deploys the autonomous hardware recovery *agents* on all nodes. These agents communicate with the autonomous hardware recovery *backends* that are deployed within Kubernetes worker nodes.

BCM integration allows API access to the agents and provides a GUI to carry out *runbooks*.

More details on autonomous hardware recovery can be found in its dedicated documentation.

The scope of this chapter of the manual is to give guidance on autonomous hardware recovery deployment, and how to access its API and GUI.

6.2 Prerequisites To Install Autonomous Hardware Recovery

Before running the `cm-mission-control-setup` script to install autonomous hardware recovery, the following prerequisites must be met:

- A Kubernetes instance that is integrated with BCM (section 4.2 of the *Containerization Manual*) must be running.
- The Kubernetes instance must have the Kubernetes NGINX Ingress controller add-on for route-based access control installed.
- A user must be added if an extra administrator account is wanted for autonomous hardware recovery. For example:

Example

```

root@basecm11:~# cmsg
[basecm11]% user add extraadmin
[basecm11->user*[extraadmin*]]% set password
enter new password:
retype new password:
[basecm11->user*[extraadmin*]]% commit

```

This extra administrator can then be specified when installing with `cm-mission-control-setup` later on (section 6.3).

- At the time of writing of this section (August 2025), Kyverno must not be installed on the Kubernetes instance, or the installation fails. This is expected to change later on.

As a rough guide to the physical hardware requirements for autonomous hardware recovery, it is expected that a cluster with hundreds of nodes should use a backend node with at least 16 CPU cores per node, at least 32 GB RAM per node, and at least 2 local (non-NFS) hard drives. The hard drives should be at least:

- one unpartitioned drive of 1.5 TB
- one drive with filesystems
 - one filesystem with at least 500 GB available for the autonomous hardware recovery backend
 - one filesystem with at least 20 GB available under `/var` for storing container images

More precise physical hardware requirements are best determined with the help of NVIDIA engineers.

6.3 Installation Of NVIDIA Autonomous Hardware Recovery Using `cm-mission-control-setup`

Running `cm-mission-control-setup` as the root user starts up the TUI installation (figure 6.1):

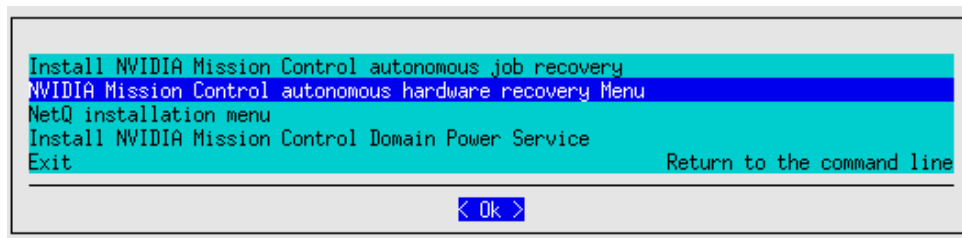


Figure 6.1: Selection from `cm-mission-control-setup` of autonomous hardware recovery

Selecting NVIDIA Mission Control autonomous hardware recovery brings up a user selection screen. The extra administrator account(s) set up earlier for administration of autonomous hardware recovery can then be specified.

After the extra administrator accounts have been specified, the Helm registry credentials screen (figure 6.2) comes up:

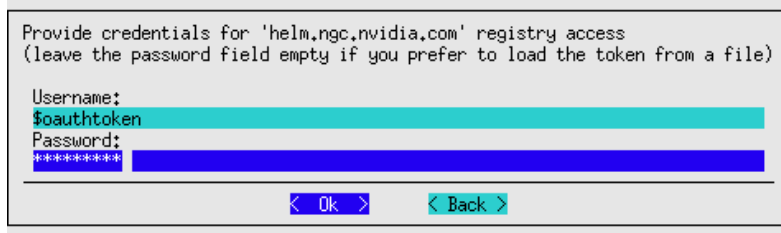


Figure 6.2: Setting Helm credentials for autonomous hardware recovery

If the Helm registry password is accepted, then the NVCR registry credentials screen (figure 6.3) comes up:

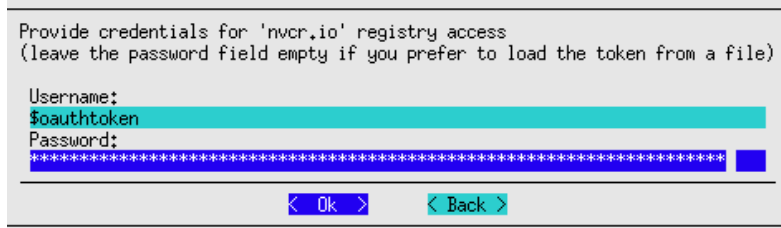


Figure 6.3: Setting NVCR credentials for autonomous hardware recovery

If the NVCR registry password is accepted, then the API registry credentials screen (figure 6.4) comes up:

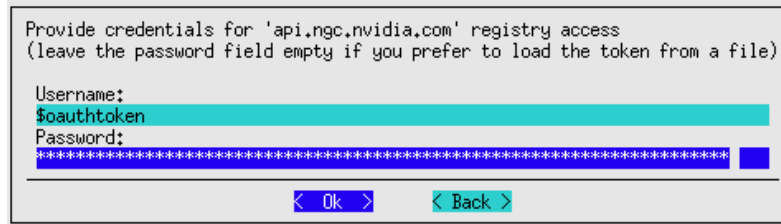


Figure 6.4: Setting API credentials for autonomous hardware recovery

If the password for API registry access is accepted, then a TLS certificates screen (figure 6.5) comes up:

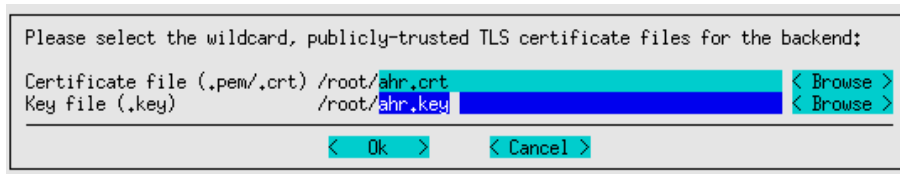


Figure 6.5: Setting TLS certificate files for autonomous hardware recovery

After the path to the TLS certificates are entered, a backend node selection screen (figure 6.6) comes up:

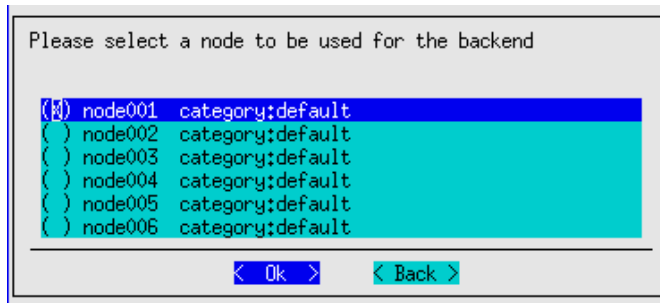


Figure 6.6: Setting a backend node for autonomous hardware recovery

After a backend node is selected, a failover node selection screen (figure 6.7) comes up. Selection is optional:

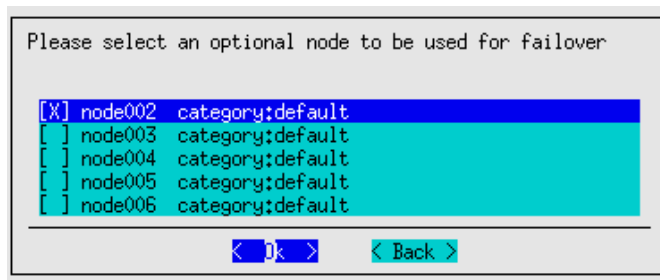


Figure 6.7: Setting an optional failover node for autonomous hardware recovery

After that an agent category selection screen (figure 6.8) comes up:

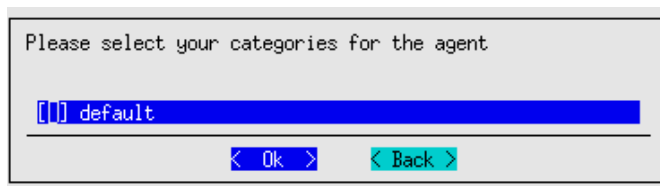


Figure 6.8: Setting an agent category for autonomous hardware recovery

After that a backend customizations screen (figure 6.9) comes up:

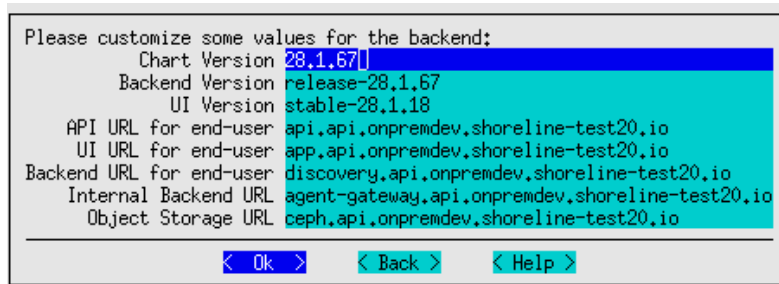


Figure 6.9: Setting backend customizations for autonomous hardware recovery

The endpoints set in the backend should resolve either to the active head node, or to the backend pod node.

A warnings screen appears if the domains cannot be resolved (figure 6.10):

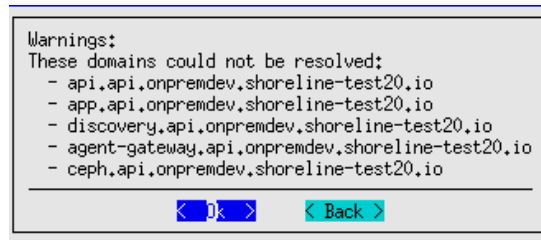


Figure 6.10: Configuration warnings for autonomous hardware recovery

After that a storage customizations screen for the backend appears (figure 6.11) with suggested default values:

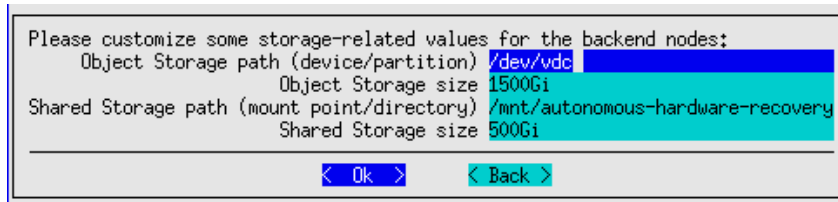


Figure 6.11: Storage customizations for the backend for autonomous hardware recovery

The TUI then asks if monitoring should be set up.

If monitoring is to be configured, then values for the Prometheus and Grafana Tempo endpoints, and for the Prometheus and Grafana Tempo API keys must be entered. The HTTP user ID and password for Loki logging must also be entered (figure 6.12):

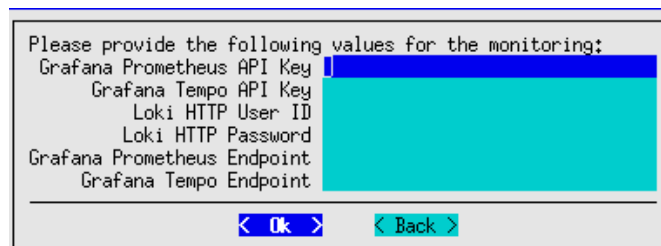


Figure 6.12: Monitoring values configuration for autonomous hardware recovery

The agent version should be set (figure 6.13):

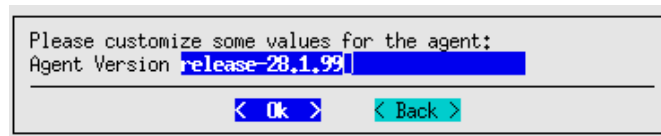


Figure 6.13: Agent version setting for autonomous hardware recovery

The configuration from the wizard can be saved and the configuration can be deployed (figure 6.14):

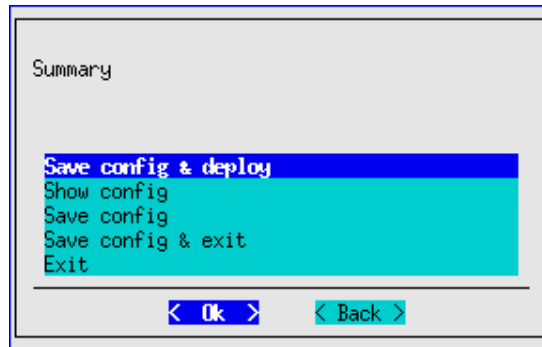


Figure 6.14: Saving the configuration and deploying it for autonomous hardware recovery

6.4 Verifying Autonomous Hardware Recovery Is Up And Ready

6.4.1 Checking The Pods

The pods that run autonomous hardware recovery should be visible in their namespace after the deployment. Their status should be Running, or Completed:

Example

```
root@basecm11:~# kubectl get pods -n autonomous-hardware-recovery
NAME                                READY   STATUS    RESTARTS   AGE
shoreline-backup-29208910-dwrpv     0/1    Completed 0           14m
shoreline-backup-29208915-6fghc     0/1    Completed 0           9m19s
shoreline-backup-29208920-17h25     0/1    Completed 0           4m19s
shoreline-disk-checker-8tstl4       0/1    Completed 0           47m
shorelinebackend-0                  7/7    Running   0           46m
shorelinebackend-failover-0         7/7    Running   0           46m
root@basecm11:~#
```

6.4.2 Testing The Web Interface

The API is described in <https://ssapi.shorelinesoftware.net/>.

Access to the UI website can be tested by running a curl command against the home endpoint page of its domain. The domain was earlier picked up from its certificates, and in the following example is `app.api.onpremdev.shoreline-test20.io`

Example

```
root@basecm11:~# curl https://app.api.onpremdev.shoreline-test20.io/home/
```

This should get output similar to:

Example

```
<!doctype html>
<html lang="en">
  <head>
    <base href="/home/" />

    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <script>
    window.$$version = 'stable-28.1.25';
```

```
</script>
<script type="module" crossorigin src="/home/assets/index-CMnWoHWq.js"></script>
<link rel="stylesheet" crossorigin href="/home/assets/index-DEW7GL4G.css">
</head>
<body>
  <div id="app"></div>
</body>
</html>
```

The web UI can be accessed if the API domain and the application domain point to the external IP address of the head node.

If for some reason DNS resolution is not yet set up for the domain, then, for example, if the head node is at the IP address 10.3.195.201, the following entries can be added to `/etc/hosts` on the head node for the `api` and `app` subdomains:

```
10.3.195.201 api.api.onpremedev.shoreline-test20.io
10.3.195.201 app.api.onpremedev.shoreline-test20.io
```

A web browser that accesses `https://app.api.onpremedev.shoreline-test20.io/` from the head node then displays the autonomous hardware recovery interface landing page (figure 6.15):

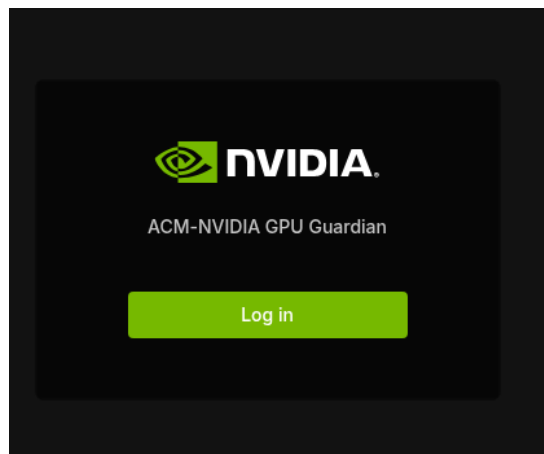


Figure 6.15: Web GUI for autonomous hardware recovery: landing page

A login can be carried out on the GUI with a username/password authentication (figure 6.16):

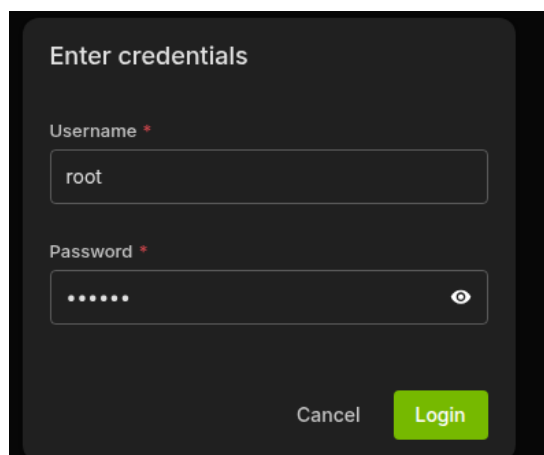


Figure 6.16: Web GUI for autonomous hardware recovery: login page

This opens up the autonomous hardware recovery dashboard.

A `hello world` runbook can be executed after login as follows:

- The runbook window is opened using the navigation panel item Runbooks
- A new runbook is created by clicking on the New Runbook button, and then on the `op statement` option (figure 6.17):

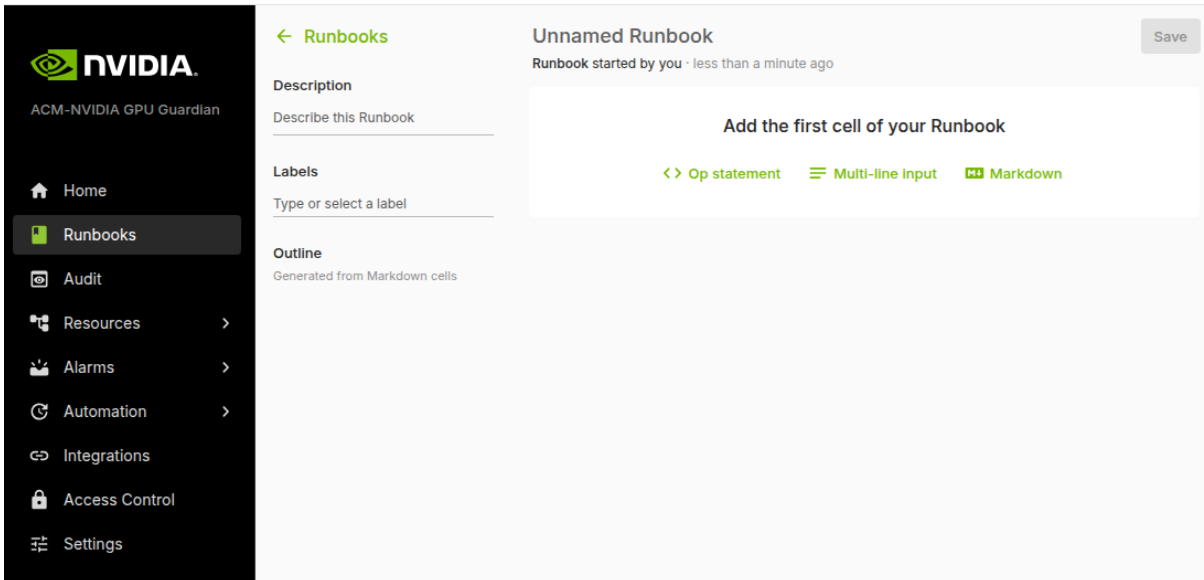


Figure 6.17: Web GUI for autonomous hardware recovery: creating a new runbook

- A cell opens up and prompts the user to enter a statement.

The following statement can be added:

```
host | `echo Hello, World!`
```

The screenshot shows the NVIDIA web GUI for autonomous hardware recovery. The interface is titled 'Create Runbook - Runbooks' and is accessed via the URL 'app.api.onpremdev.shoreline-test20.io/runbooks/new/'. The main content area is titled 'Unnamed Runbook' and shows a runbook started by the user less than a minute ago. The runbook contains a single command: `host | `echo Hello, World!``. The output table shows four rows of 'Hello, World!' for different host IDs. The runbook was started at 07/15/2025, 2:46:24 PM and took 910 ms to execute.

id	type	name	Exit code	Output
1	HOST	pj-b110-u2404-07-14	0	Hello, World!
2	HOST	node002	0	Hello, World!
3	HOST	node003	0	Hello, World!
4	HOST	node001	0	Hello, World!

Figure 6.18: Web GUI for autonomous hardware recovery: adding a “hello world” runbook

- The runbook can be executed with the enter key or by clicking on the Execute button next to the cell. There is hovertext to explain the meanings of the widgets. Outputs of Hello, World! should show up in the GUI (figure 6.18):
- The runbook can be saved for posterity with the Save button.

If the preceding runbook session works, then it indicates that autonomous hardware recovery has been deployed properly.

7 NVLink Technology

This chapter introduces *NVLink technology* as used in NVIDIA GB200 and GB300 server platforms with BCM. NVLink is NVIDIA's high-speed interconnect technology that enables direct GPU-to-GPU communication with significantly higher bandwidth and lower latency than traditional PCIe connections.

Section 7.1 provides an overview of NVLink technology and its role in GB200/GB300 systems.

Section 7.2 describes the NVLink architecture in GB200 and GB300 platforms.

Section 7.3 covers BCM commands for querying NVLink status and information.

7.1 NVLink Overview

NVLink is NVIDIA's proprietary high-bandwidth, energy-efficient interconnect that enables fast communication between GPUs and between GPUs and CPUs. In GB200 and GB300 systems, NVLink provides the backbone for GPU-to-GPU communication within and across compute trays.

7.1.1 NVLink Generations

The GB200 and GB300 platforms use fifth-generation NVLink (NVLink 5), which provides:

- 900 GB/s bidirectional bandwidth per link
- Support for up to 576 GPUs in a single NVLink domain
- NVLink Switch architecture for scaling beyond direct GPU connections
- Integration with NVIDIA Grace CPUs for unified memory architecture

7.1.2 NVLink Vs PCIe

Compared to PCIe Gen5, NVLink 5 provides approximately 14x higher bandwidth. This makes NVLink essential for getting the best out of:

- Large language model (LLM) training requiring frequent gradient synchronization
- Multi-GPU inference workloads
- High-performance computing applications with significant inter-GPU communication
- Memory pooling across multiple GPUs

7.2 NVLink Architecture In GB200/GB300

The NVIDIA GB200 and GB300 platforms implement a sophisticated NVLink topology that connects GPUs within compute trays and across the entire rack through NVLink switches.

7.2.1 Compute Tray Configuration

Each GB200/GB300 compute tray contains:

- 2 NVIDIA Grace CPUs
- 4 NVIDIA Blackwell GPUs (GB200/GB300)
- NVLink connections between all GPUs within the tray
- NVLink connections to external NVLink switch trays

The GPUs within a compute tray are fully connected via NVLink, allowing any GPU to communicate directly with any other GPU in the same tray at full NVLink bandwidth.

7.2.2 NVLink Switch Trays

GB200 NVL72 rack configurations include 9 NVLink switch trays per rack. Each switch tray contains:

- 2 NVSwitch chips per tray
- 72 NVLink ports per switch (36 top, 36 bottom per NVSwitch chip)
- 144 NVLink ports per switch tray total

The NVLink switches enable all-to-all GPU communication across the entire rack, creating a unified GPU fabric where any GPU can communicate with any other GPU at high bandwidth.

7.2.3 NVL72 Topology

The NVL72 reference configuration provides:

- 72 GPUs per rack (18 compute trays × 4 GPUs)
- Full bisection bandwidth between all GPUs
- 130 TB/s total NVLink bandwidth per rack
- Single NVLink domain spanning all 72 GPUs

This topology is optimized for large-scale AI training workloads, where model parallelism requires frequent communication between GPUs.

7.3 NVLink Commands

NVLink status and information can be queried using commands available in `cmsh` device mode.

7.3.1 Listing NVLink Switches: `list`

The `list` command in device mode can filter for NVLink switches using the `--field` option. This displays all NVLink switches configured in the cluster:

Example

```
[basecm11->device]% list --field kind=nvlink
Type      Hostname      MAC              IP              Network      Status
-----
Switch    a07-p01-nvsw-01  E0:9D:73:05:70:6C  10.140.0.45  ipminet0 [  UP  ]
Switch    a07-p01-nvsw-02  B8:E9:24:B9:84:8C  10.140.0.46  ipminet0 [  UP  ]
Switch    a07-p01-nvsw-03  E0:9D:73:05:71:2A  10.140.0.47  ipminet0 [  UP  ]
...
```

The `list` command has many filtering options. Details can be seen by running the `help` command (`help list`).

7.3.2 NVLink Switch Status

NVLink switch status can be monitored through the standard `status` command of `device` mode. The switches report their operational state, link status, and any errors through the BCM monitoring system:

Example

```
[basecm11->device]% use a07-p01-nvsw-01
[basecm11->device[a07-p01-nvsw-01]]% status
a07-p01-nvsw-01 ..... [  UP  ]
```

7.3.3 NVLink Fabric Information: `nvfabricinfo`

The `nvfabricinfo` command is run in `device` mode, and displays information about NVLink fabric domains. It shows the domain identifier, the active switch, and all switches that are part of each NVLink domain:

Example

```
[basecm11->device]% nvfabricinfo
Domain  Active      Switches
-----
A07     a07-p01-nvsw-01  a07-p01-nvsw-01..a07-p01-nvsw-09
```

The output shows:

- **Domain:** The NVLink domain identifier
- **Active:** The currently active/primary switch for the domain
- **Switches:** All NVLink switches that are part of the domain

The `--raw` option displays detailed JSON output showing the status of each switch in the fabric, including NMX controller and telemetry application status:

Example

```
[basecm11->device]% nvfabricinfo --raw
===== A07 =====
{
  "58519ce2-f695-4a4f-976b-99074aa13f65": {
    "apps": {
      "nmx-controller": {
        "addition-info": "CONTROL_PLANE_STATE_CONFIGURED",
        "app-id": "nmx-c-nvos",
        "app-ver": "4.21.30",
        "capabilities": "sm, gfm, fib, gw-api",
        "status": "ok"
      }
    }
  },
}
```

```

    "nmx-telemetry": {
      "app-id": "nmx-telemetry",
      "app-ver": "3.5.1",
      "capabilities": "nvl, gnmi, syslog, bmc",
      "status": "ok"
    }
  },
  "hostname": "a07-p01-nvsw-01",
  "leader": true,
  "success": true
},
"1f79262d-437e-4da4-8197-1522723e0861": {
  "apps": {},
  "hostname": "a07-p01-nvsw-02",
  "leader": false,
  "message": "nmx-controller not started",
  "success": true
},
...
}

```

The raw output for each switch shows:

- **hostname:** The switch hostname
- **leader:** Whether this switch is the active leader for the domain
- **apps:** Running NMX applications (`nmx-controller`, `nmx-telemetry`) with version and status
- **success:** Whether the switch query was successful
- **message:** Status message (for example, if NMX controller is not started)

The `--start` option can be used to switch the active/leader switch for a domain. This is useful for maintenance or failover scenarios:

Example

```
[basecm11->device]% nvfabricinfo --start a07-p01-nvsw-02
```

This command initiates a leader transition, making `a07-p01-nvsw-02` the new active switch for the NVLink domain. The NMX controller starts on the new leader and stops on the previous leader.

The help text (`help nvfabricinfo`) has further details on the `nvfabricinfo` command, including options for filtering and formatting.

7.3.4 NVLink Platform Information: `nvplatforminfo`

The `nvplatforminfo` command is run in `device` mode, and displays detailed NVLink platform information for each GPU in the cluster. It shows the physical location and identification of GPUs within the NVLink fabric, including slot, tray, and module assignments:

Example

```

[basecm11->device]% nvplatforminfo
Node           GPU  IB GUID                Slot Tray Host ID Peer type Module ID
-----
a07-p01-dgx-02-c01  0  0x02e56c0fe280b0f4    1   0    1      1      2
a07-p01-dgx-02-c01  1  0x7b948c1631919719    1   0    1      1      1
a07-p01-dgx-02-c01  2  0x7415054f46b585bb    1   0    1      1      4
a07-p01-dgx-02-c01  3  0x52ef972251ac3248    1   0    1      1      3

```

```

a07-p01-dgx-02-c02 0 0x16d5a2b400bf3441 2 1 1 1 2
a07-p01-dgx-02-c02 1 0x64ba6860fde1f9d4 2 1 1 1 1
a07-p01-dgx-02-c02 2 0x68e21af78702759a 2 1 1 1 4
a07-p01-dgx-02-c02 3 0x89114b471859c432 2 1 1 1 3
...

```

The preceding output shows:

- GPU: The GPU index within the node (0-3 for GB200/GB300 compute trays)
- IB GUID: The InfiniBand Global Unique Identifier for the GPU
- Slot: The physical slot position in the rack
- Tray: The compute tray number
- Host ID: The host identifier within the NVLink domain
- Peer type: The peer connection type
- Module ID: The module identifier within the tray

The help text (`help nvplatforminfo`) has further details on the `nvplatforminfo` command, including options for filtering by node, category, rack, and so on:

7.3.5 NVLink Domain Information: `nvdomaininfo`

The `nvdomaininfo` command is run within `device` mode, and displays the NVLink domain membership status for each GPU. It shows whether GPUs have successfully joined the NVLink cluster, and shows their assigned cluster UUID and clique ID:

Example

```

[basecm11->device]% nvdomaininfo
Node          GPU  Status      Cluster UUID          Clique ID
-----
a07-p01-dgx-02-c01 0  Success    084aea8f-abb0-4209-a016-7ab97d55609c  32766
a07-p01-dgx-02-c01 1  Success    084aea8f-abb0-4209-a016-7ab97d55609c  32766
a07-p01-dgx-02-c01 2  Success    084aea8f-abb0-4209-a016-7ab97d55609c  32766
a07-p01-dgx-02-c01 3  Success    084aea8f-abb0-4209-a016-7ab97d55609c  32766
...
a07-p01-dgx-02-c16 0  Success    3185472a-9c82-4f72-98a7-dc1fddfd188b  32766
a07-p01-dgx-02-c16 1  Success    3185472a-9c82-4f72-98a7-dc1fddfd188b  32766
a07-p01-dgx-02-c16 2  In progress 00000000-0000-0000-0000-000000000000  0
a07-p01-dgx-02-c16 3  In progress 00000000-0000-0000-0000-000000000000  0
...

```

The output shows:

- Status: The domain join status (Success, In progress, or error states)
- Cluster UUID: The unique identifier of the NVLink cluster that the GPU belongs to
- Clique ID: The clique identifier within the NVLink domain

GPUs with an In progress status are still in the process of joining the NVLink domain. They show a zeroed UUID until the join completes. Different cluster UUIDs indicate GPUs belonging to separate NVLink domains.

The help text (`help nvdomaininfo`) has further details on the `nvdomaininfo` command, including options for filtering by node, category, rack, and so on.

7.3.6 NVLink SDN Partition Information: `nvsdnpartitioninfo`

The `nvsdnpartitioninfo` command displays information about NVLink SDN (Software Defined Networking) partitions. It shows how GPUs are grouped into partitions across the NVLink fabric, including the associated switches and health status. The command is run in device mode:

Example

```
[basecm11->device]% nvsdnpartitioninfo
[basecm11->device]% nvsdnpartitioninfo
Rack   Switches           Clique Name           Health   Node GPUs
-----
A07    nvsw-01..nvsw-09  32766  Default Partition   healthy dgx-02-c[01-17]:[0-3]
```

The output shows:

- **Rack:** The rack identifier containing the partition
- **Switches:** The NVLink switches that are part of the partition
- **Clique:** The clique identifier for the partition
- **Name:** The partition name (e.g., `Default Partition`)
- **Health:** The health status of the partition (`healthy`, `degraded`, etc.)
- **Node GPUs:** The nodes and GPU indices included in the partition

The help text (`help nvsdnpartitioninfo`) has further details on the `nvsdnpartitioninfo` command, including options for filtering by node, category, rack, and so on.

7.3.7 NVLink Link Status: `nvlinkinfo`

The `nvlinkinfo` command displays the status of individual NVLink links on each GPU. Each GPU in a GB200/GB300 system has 18 NVLink links connecting it to other GPUs and NVLink switches. The command is run in device mode (output truncated):

Example

```
[basecm11->device]% nvlinkinfo
Node           Type  Entity  Link  Status
-----
a07-p01-dgx-02-c01 GPU   0       0     Up
a07-p01-dgx-02-c01 GPU   0       1     Up
a07-p01-dgx-02-c01 GPU   0       2     Up
...
a07-p01-dgx-02-c01 GPU   0       17    Up
a07-p01-dgx-02-c01 GPU   1       0     Up
...
a07-p01-dgx-02-c01 GPU   3       17    Up
a07-p01-dgx-02-c02 GPU   0       0     Up
...
a07-p01-dgx-02-c18 GPU   3       17    Up
```

The output shows:

- **Type:** The device type (GPU)
- **Entity:** The GPU index within the node (0-3)

- `Link`: The NVLink link number (0-17 for 18 links per GPU)
- `Status`: The link status (Up or Down)

Each compute tray has 4 GPUs with 18 NVLink links each, resulting in 72 link entries per node.

The help text (`help nvlinfo`) has further details on the `nvlinfo` command, including options for filtering by node, category, rack, and so on.

7.3.8 NVLink Monitoring With NMX

The NVIDIA Mission Control features allow NVLink measurables to be monitored (section 11.3).

For comprehensive NVLink monitoring, the NMX (NVIDIA Management Extensions) telemetry system provides detailed metrics on:

- NVLink port utilization and bandwidth
- error counts and link health
- temperature and power consumption
- traffic patterns and congestion

NMX telemetry integration with BCM is covered further in the BCM NMX documentation (section 2.1).

8 NVLink Switch Integration

The NVIDIA GB200 server platforms have 9 NVLink switch trays per rack. An NVL72 reference rack configuration is described in, for example, [the NVIDIA DGX SuperPOD configuration](#).

The switch trays connect GPUs in each rack (36 per rack, or 72 per rack) to each other on a 400 Gb/s InfiniBand NDR network.

This chapter considers an NVIDIA GB200 server platform that has been configured to work with BCM.

8.1 NVLink Switch Listing And Overview

The `list -t switch` command in device mode can be used to list the all switches of a configured cluster. The `list -field kind=nvlink` command in device mode can be used to list only the NVLink switches of a configured cluster.

NVLink switches are configured as devices on a BMC network, as indicated by the following example:

Example

```
[maple->device]% list -t switch
Type      Hostname (key)      MAC                IP                Network           Status
-----
Switch    dgx-gb200-m06-nv1sw1  E0:9D:73:05:70:6C  10.140.0.45       ipminet0          [ UP ]
Switch    dgx-gb200-m07-nv1sw1  B8:E9:24:B9:84:8C  10.140.0.48       ipminet0          [ UP ]
Switch    dgx-gb200-n01-nv1sw1  E0:9D:73:05:70:5C  10.140.0.51       ipminet0          [ UP ]
Switch    dgx-gb200-n07-nv1sw1  E0:9D:73:05:70:AC  10.140.0.54       ipminet0          [ UP ]
...
```

Example

```
[basecm11->device[dgx-gb200-m06-nv1sw1]]% switchoverview
Device      : dgx-gb200-m06-nv1sw1
State       : [ UP ]
Model       : N5400_LD
Serial number : MT1234556789
Part number  : 123-4567-89

Port assignment:

Port  Name   Status Uplink Speed    RX    TX
-----
1     acp1   UP     no    400 Gb/s  64 TiB  64 TiB
2     acp2   UP     no    400 Gb/s  64 TiB  64 TiB
```

```

...
143  acp143  UP    no    400 Gb/s  64 TiB  64 TiB
144  acp144  UP    no    400 Gb/s  64 TiB  64 TiB
145  eth0    UP    no    1.00 Gb/s  1.22 TiB  6.0 GiB
146  eth1    DOWN  no    0 b/s     4.5 KiB  28.7 KiB
147  fnm1    UP    no    100 Gb/s  0 B      0 B
148  fnm2    UP    no    100 Gb/s  0 B      0 B

```

8.2 NVLink Switch Configuration

Each NVLink switch has two redundant ethernet interfaces and a Redfish interface.

Example

```

[basecm11->device[a05-p1-nvsw-01]->interfaces]% list
Type          Network device name  IP          Network
-----
bmc           rf0                  7.241.3.21  ipminet2
physical      eth0                 7.241.3.1   ipminet2
physical      eth1                 7.241.3.11  ipminet2

```

An NVLink runs NVOS and can be configured to boot via ZTP. The JSON template ZTP file that is provided by BCM is used to check the configured image, and to install the `cm-lite-daemon`. If ZTP is configured correctly before the switch first boots, then it is ready for use.

Example

```

[basecm11->device[a05-p1-nvsw-01]->ztpsettings]% show
Parameter          Value
-----
Revision
Script template    nvos-ztp.sh
JSON template      nvlink-nvos.json
Image              nvos-amd64-25.02.2141.bin
Check image on boot no
Run ZTP on each boot yes
Install lite daemon yes
Enable API         yes

```

On an NVLink switch that is already booted, or where ZTP is disabled, `cm-lite-daemon` can be installed manually.

The first step is to download the relevant packages using `cmsh`.

The download needs to be executed once for all switches. If updates are available in the BCM repositories, then the download should be repeated and the subsequent installation process should also be repeated.

```

[basecm11->device]% litedaemon download
Download all/cm-config-cm_1trunk_all.deb for 2004: done
Download all/cm-lite-daemon_1master-100775-cm-752a72d739_all.deb for 2004: done
Download amd64/cm-openssl_3.1.8-100204-cm-a2b95b5141_amd64.deb for 2004: done
Download amd64/cm-python312_3.12.11-100035-cm-f4ae26a924_amd64.deb for 2004: done
Download amd64/cm-python3_1master-100155-cm-eaf713f03f_amd64.deb for 2004: done
...

```

After the packages are downloaded, they can be installed on the NVLink switches using the `litedaemon` package manager with the `install` command. It is recommended to do all 9 switches for a rack at the same time; multiple racks can be done at once.

```
[basecm11->device]% litedaemon install --rack a05 --intersection --field kind=nvlink
success: yes
--- stdout ---
run: /home/admin/cm-lite-daemon-install.py --head-ips 10.141.255.254 --port 8081
      --hostname a05-p1-nvsw-01 --interface eth0
+ ping -c 1 10.141.255.254
Using IP 10.141.255.254 without supplied vrf
* Download: https://10.141.255.254:8081/switch/packages/2404/files *
* Download: https://10.141.255.254:8081/switch/packages/2404/cm-config-cm_1trunk_all.deb *
* Download: https://10.141.255.254:8081/switch/packages/2404/cm-lite-daemon_1master-
100775-cm-752a72d739_all.deb *
* Download: https://10.141.255.254:8081/switch/packages/2404/cm-openssl_3.1.8-100204-
cm-a2b95b5141_amd64.deb *
* Download: https://10.141.255.254:8081/switch/packages/2404/cm-python312_3.12.11-
100035-cm-f4ae26a924_amd64.deb *
* Download: https://10.141.255.254:8081/switch/packages/2404/cm-python3_1master-100155-
cm-eaf713f03f_amd64.deb *
* Install *
* Setup *
* Setup from /home/admin *
* Certificates *
  - cluster.pem
  - bootstrap.pem
  - bootstrap.key
* Register *
* Done *
```

When updated packages have been downloaded they also need to be updated on the NVLink switches. Updates can be carried out using the `--update` option with `litedaemon install` as follows:

```
[basecm11->device]% litedaemon install --update --field kind=nvlink
run: /home/admin/cm-lite-daemon-install.py --head-ips 10.141.255.254 --port 8081
      --hostname a05-p1-nvsw-01 --interface eth0 --update
+ ping -c 1 10.141.255.254
Using IP 10.141.255.254 without supplied vrf
* Download: https://10.141.255.254:8081/switch/packages/2404/files *
* Download: https://10.141.255.254:8081/switch/packages/2404/cm-config-cm_1trunk
_all.deb *
* Download: https://10.141.255.254:8081/switch/packages/2404/cm-lite-daemon_1master-
100775-cm-752a72d739_all.deb *
* Download: https://10.141.255.254:8081/switch/packages/2404/cm-openssl_3.1.8-100204-
cm-a2b95b5141_amd64.deb *
* Download: https://10.141.255.254:8081/switch/packages/2404/cm-python312_3.12.11-
100035-cm-f4ae26a924_amd64.deb *
* Download: https://10.141.255.254:8081/switch/packages/2404/cm-python3_1master-
100155-cm-eaf713f03f_amd64.deb *
* Install *
* Restart *
```

Removing all BCM-related packages can be carried out using the `--update` option with `lightdaemon remove` as follows:

```
[basecm11->device]% litedaemon remove --update --field kind=nvlink
* Systemctl stop *
* Remove *
* Systemctl reload *
```

8.3 NVLink Switch Monitoring

NVLink switches are monitored in two ways.

- OS data using `cm-lite-daemon` running on the switch
- Redfish sampled from the active head node (currently only leak information is reported)

All data can be viewed using standard BCM monitoring

```
[basecm11->device[a05-p1-nvsw-01]]% latestmonitoringdata
...
LoadOne                3.21   33s
RF_LD_LeakDetection    OK     1s    Rollup: OK
RF_LeakDetector        leakage1 OK     1s    Detector: OK, Health: OK, State: Enabled
RF_LeakDetector        leakage2 OK     1s    Detector: OK, Health: OK, State: Enabled
RF_LeakDetector        leakage3 OK     1s    Detector: OK, Health: OK, State: Enabled
RF_LeakDetector        leakage4 OK     1s    Detector: OK, Health: OK, State: Enabled
RF_LeakDetector        leakage5 OK     1s    Detector: OK, Health: OK, State: Enabled
RF_LeakDetector        leakage6 OK     1s    Detector: OK, Health: OK, State: Enabled
RF_LeakDetector        leakage_agg OK     1s    Detector: OK, Health: OK, State: Enabled
```

8.4 NVLink Switch Redfish Events

NVLink switches report leaks to all tools that are subscribed to Redfish events. BCM does this for all NVLink switches if the Redfish IP address and BMC settings are configured correctly.

The `redfishsubscription` command lists the devices that have a Redfish subscription (page 90).

```
[basecm11->device[a05-p1-nvsw-01]]% redfishsubscription
hostname      id      ip          kind      port      subscribed
-----
a05-p1-nvsw-01 1234    7.241.3.21 GB200SW   443       true
```

Testing of the event subscription work can be done from inside `cmsh`.

```
[basecm11->device[a05-p1-nvsw-01]]% redfishevent --message Test
...
```

9 Power Reservation Steering

9.1 Introduction

Power reservation steering (PRS) is about automated speculative planning of power across groups of nodes used by Slurm. PRS is typically done for larger clusters, where minimising the number of idling nodes typically results in significant savings.

The planning is done over a period of time spent measuring existing resource consumption and observing the power consumed. The patterns that are seen in that period of time suggest a direction for the power allocation on the cluster (hence the word *steering*) for the next period of time. Based on the planned steering, a power cap is set for some groups of nodes (power domains) in the cluster. Setting the power cap for those power domains, means that a *reservation* of the remaining power is made for the rest of the cluster during the upcoming period of time.

PRS works using a PRS daemon. This tracks the power availability and manages power planning. A component is the PRS controller that connects to CMDaemon, and which during the time loop period controls the power domains.

DPS (Domain Power Service, Chapter 10) is a datacenter level software intended to optimally manage power allocation across the entire datacenter. By default, if a Slurm topology has PRS enabled, then DPS configures PRS with a power budget for nodes.

9.2 Deployment Of PRS

After a Slurm cluster has been configured and deployed (section 7.3.2 of the *Administrator Manual*), PRS can be deployed.

Deployment of PRS is started in BCM by running the `cm-prs-setup` script. This brings up a TUI wizard (figure 9.1):

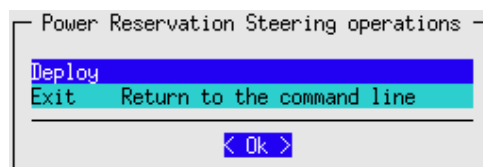


Figure 9.1: Starting deploying with `cm-prs-setup`

Categories can be selected for PRS (figure 9.2):

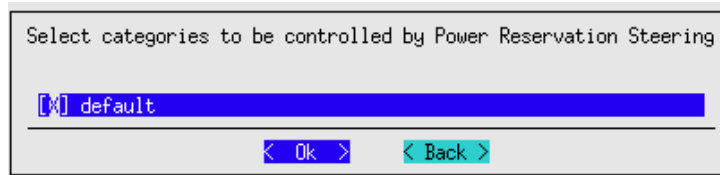


Figure 9.2: Selecting a category with `cm-prs-setup`

If selecting categories is too inclusive, then individual nodes can be selected instead in a following screen.

After categories or individual nodes have been selected, one or more power domains can be added (figure 9.3):

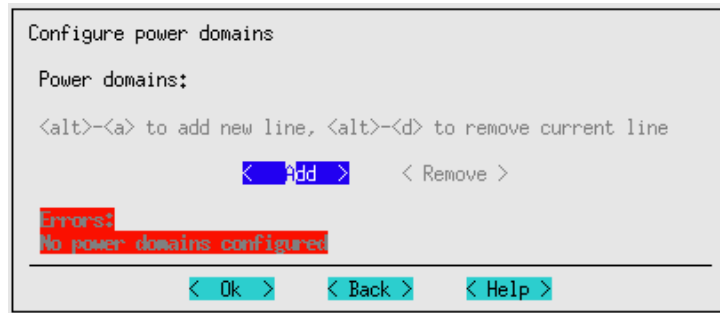


Figure 9.3: Setting a power domain with `cm-prs-setup`

Each power domain can be given an arbitrary name, and a domain grouping (figure 9.4):

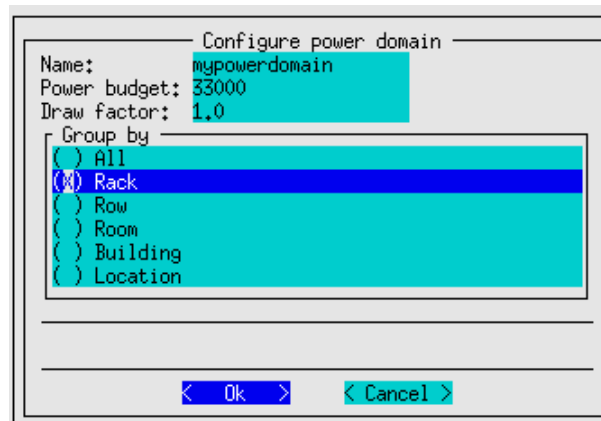


Figure 9.4: Configuring the power domain budget and grouping with `cm-prs-setup`

The default domain group of `all` is simply one big power domain. Groups such as `rack`, `row`, `room`, and so on become new domains, each with the specified power budget.

The power budgets for the domains are set automatically if the nodes have been powered up. Otherwise the budgets must be specified manually.

The maximum power consumption per node is:

- 400W for A100 nodes
- 700W for H100 nodes
- 1000W for B200 nodes

- 2700W for GB200 nodes

The Static power is the power consumption when the node is shut down, and only the BMC is standing by. Typically it is about 20W.

The power draw factor can be set in the range from 0.0-1.0, in steps of 0.1.

The power consumption limits can then be set for the domain.

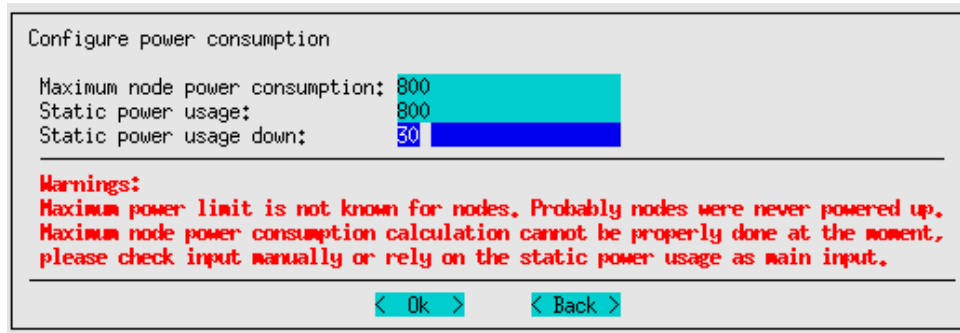


Figure 9.5: Setting power consumption limits with `cm-prs-setup`

After the configuration steps are completed, the configuration can be saved and PRS deployed (figure 9.6):

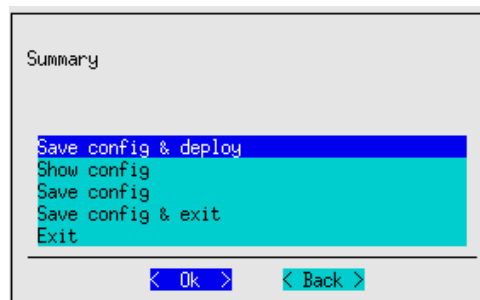


Figure 9.6: Saving the configuration and deploying it with `cm-prs-setup`

9.3 Changes Made On Deployment, And Managing Changes Post-Deployment With cmsh

9.3.1 PRS Values

PRS deployment creates the following configuration overlays:

- `prs-server`: this has the `PRSServerRole` with `PRSDomain` configurations
- `prs-client`: this has the `PRSClietRole` with power usage configuration

The PRS server configuration overlay is always assigned to the head nodes, and the client overlay is assigned to the nodes and categories configured in the wizard.

PRS server configuration values can be managed in `cmsh` with their role under the configuration overlay:

Example

```
[basecm11->configurationoverlay[prs-server]->roles[PRS::Server]]% show
Parameter                               Value
-----
Name                                     PRS::Server
Revision
Type                                     PRSServerRole
Add services                             yes
Config server port                       8880
Job scheduler server port                8881
Timeout                                  10s
Interval                                  30s
Window                                    5
PRS server certificate path               /cm/local/apps/prs/etc/server.pem
PRS server private key path               /cm/local/apps/prs/etc/server.key
PRS server CA certificate path             /cm/local/apps/prs/etc/ca.pem
PRS server CA private key path            /cm/local/apps/prs/etc/ca.key
CMD certificate path                      /cm/local/apps/prs/etc/cmd.pem
CMD private key path                     /cm/local/apps/prs/etc/cmd.key
PRS client certificate path                /cm/local/apps/prs/etc/prs.pem
PRS client private key path                /cm/local/apps/prs/etc/prs.key
PRS client CA certificate path             /cm/local/apps/cmd/pythoncm/lib/python3.12/site-packages/
pythoncm/etc/cacert.pem
Domains                                  <2 in submode>
[basecm11->configurationoverlay[prs-server]->roles[PRS::Server]]% domains
[basecm11->configurationoverlay[prs-server]->roles[PRS::Server]->domains]% list
Name (key)    Power budget Power budget model Power draw model Power draw factor Group by
-----
anotherdom    30.0KW      scalar          linear          1                all
mypowerdomain 10.0KW      scalar          linear          1                all
[basecm11->configurationoverlay[prs-server]->roles[PRS::Server]->domains]%
```

Similarly, the PRS client role values can be managed under their role:

Example

```
[basecm11->configurationoverlay[prs-client]->roles[PRS::Client]]% show
Parameter                               Value
-----
Name                                     PRS::Client
Revision
Type                                     PRSClientRole
Add services                             yes
Static power usage                       800W
Static power usage down                   30W
Min CPU power limit                       0W
Max CPU power limit                       0W
Min GPU power limit                       0W
Max GPU power limit                       0W
```

9.3.2 Slurm Values

On deployment of PRS, the selectType of the Slurm instance is changed from:

```
select/cons_tres
to
select/gnl_cons_tres
```

Example

```
[basecm11->wlm[slurm]]% get selecttype
select/gnl_cons_tres
```

The certificate paths for PRS for the Slurm instances are managed within the PRS settings (section 7.5.1, page 389 of the *Administrator Manual*) of the Slurm instances:

Example

```
[basecm11->wlm[slurm]->prssettings]% show
Parameter          Value
-----
Certificate path    /cm/local/apps/prs/etc/slurm-slurm.pem
Revision
Private key path    /cm/local/apps/prs/etc/slurm-slurm.key
```

9.3.3 PRS Removal

The `cm-prs-setup` script can be used to remove existing PRS domains.

9.4 Changes Made On Deployment, And Managing Changes Post-Deployment With Base View

9.4.1 Base View Power Reservation Steering Wizard

The Base View GUI PRS wizard takes the same inputs as the `cm-prs-setup` TUI PRS wizard of section 9.3.

The Base View wizard can be accessed using the navigation path:

Mission Control > Power Reservation Steering > Start Wizard

The navigation path is only available if Slurm has previously been installed. Slurm can be installed using the navigation path:

HPC > Workload Management Wizard

The Base View PRS wizard, on starting up, displays an introduction screen as in figure 9.7:

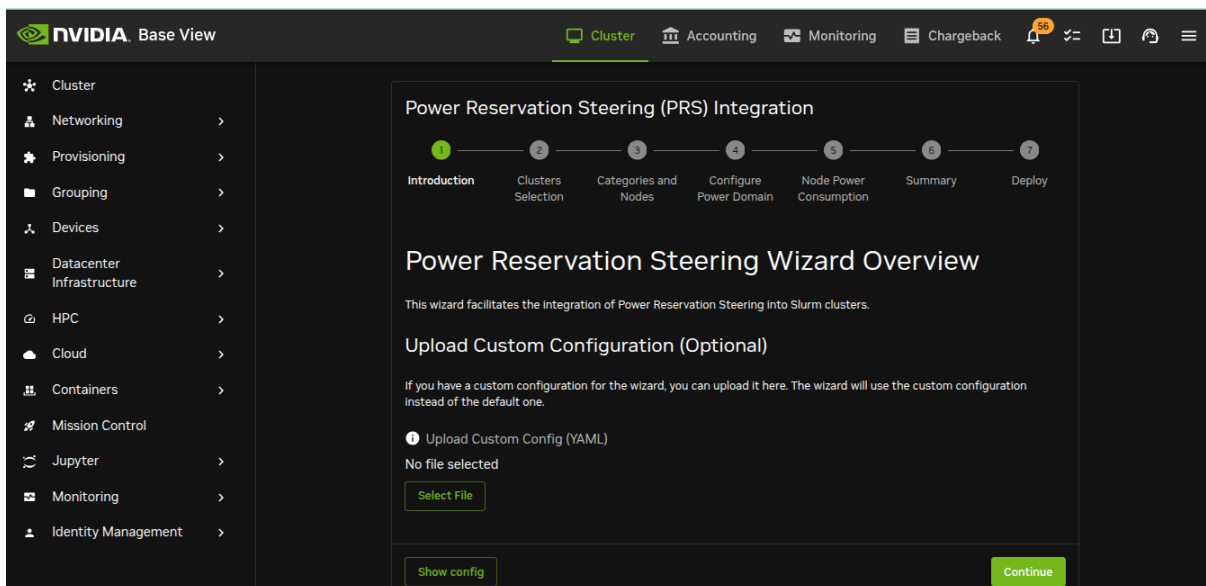


Figure 9.7: Running The PRS Installation Wizard From Base View




The steps that the PRS wizard goes through can be followed in the breadcrumb trail. The steps are:

- Introduction: a custom YAML configuration can be uploaded here, and the existing YAML configuration can be viewed
- Clusters Selection: Slurm instances can be selected here for PRS
- Categories and Nodes: regular and head nodes can be chosen for PRS
- Configure Power Domain: a power domain can be set, with settings as in the TUI version (figure 9.4)
- Node Power Consumption: power consumption for the domain can be set
- Summary: a summary of the settings is displayed
- Deploy: the deployment stages are carried out and shown

9.4.2 Base View Power Reservation Steering Client And Server Configuration Overlays

With Base View, the values set for the PRS client and PRS server can be accessed using their roles at configuration overlay, category, and device level.

For example, for the PRS client as configured in the Base View PRS configuration overlay, the navigation path to the configuration pane (figure 9.8) is:

Configuration Overlay > prs-client  > Settings > Roles  > PRS-client 

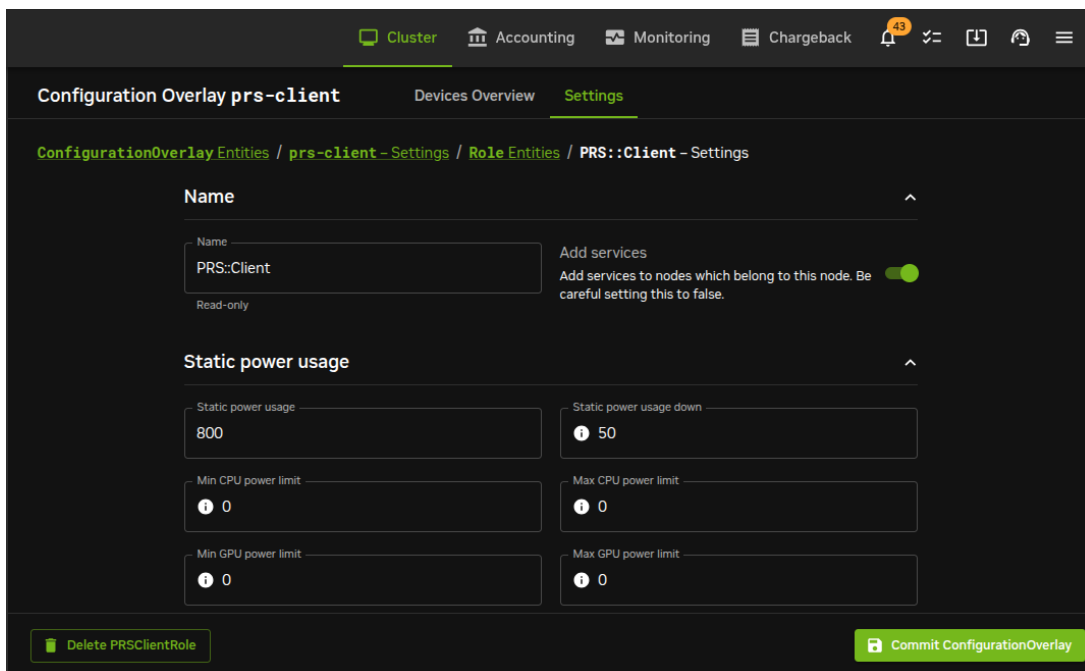


Figure 9.8: PRS Client Values Using The Configuration Role In Base View

10 Domain Power Service

10.1 Introduction

Domain Power Service (DPS) is a datacenter level software (<https://docs.nvidia.com/datacenter/dps/versions/latest/>) intended to optimally manage power allocation across the entire datacenter. By default, if a Slurm topology has PRS (Chapter 9) enabled, then DPS configures PRS with a power budget for nodes.

10.2 Deployment Of DPS

DPS runs within a Kubernetes cluster. A Kubernetes cluster can be set up as described in section 4.2 of the *Containerization Manual*.

DPS works with Slurm power management. After a Slurm cluster has been configured and deployed (section 7.3.2 of the *Administrator Manual*), DPS can be deployed.

Deployment of DPS is started in BCM by running the `cm-mission-control-setup` script. This brings up a TUI wizard (figure 10.1):

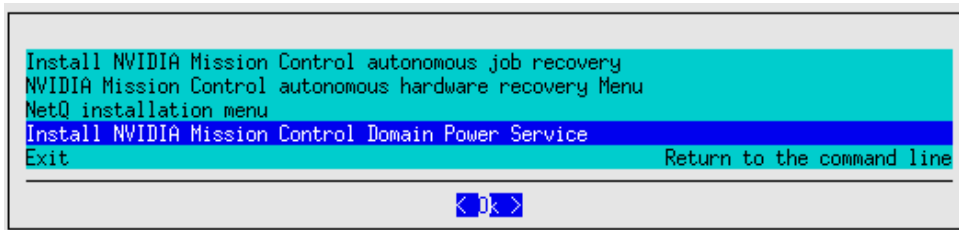


Figure 10.1: Starting DPS deployment with `cm-mission-control-setup`

Selecting DPS deployment brings up a screen that allows a password to be set for the DPS user (figure 10.2):

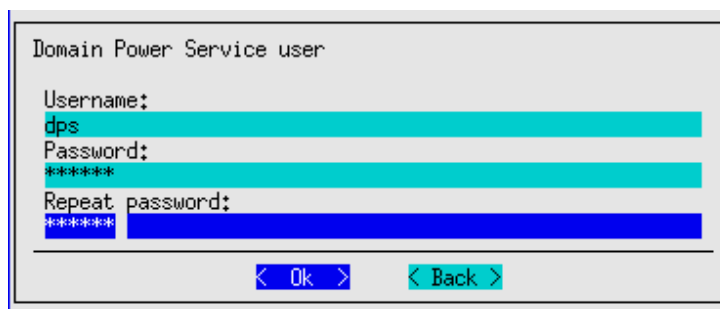


Figure 10.2: Setting a password for DPS with `cm-mission-control-setup`

A DPS version can then be set (figure 10.3):

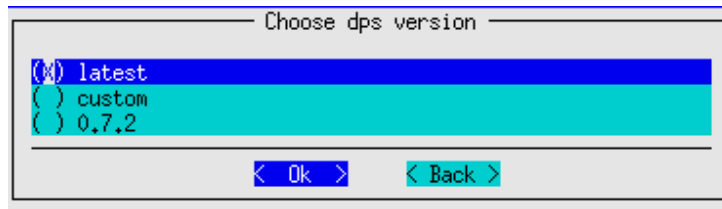


Figure 10.3: Selecting a DPS version with `cm-mission-control-setup`

Deployment can be carried out by the next screen.

DPS can be undeployed using the `cm-mission-control-setup` TUI.

10.3 Known Issues For BCM 11.31.0

10.3.1 Known Issue When Undeploying DPS

If there are DNS zones other than DPS in the file `/etc/bind/named.conf.include` and DPS is undeployed using `cm-mission-control-setup`, then the `named` service enters a broken state.

To avoid this issue, DPS should not be undeployed through `cm-mission-control-setup` in BCM version 11.31.0.

Instead, DPS can be undeployed as follows:

- The `epilogslurmctld` and `prologslurmctld` parameters of the Slurm instance that DPS is running are cleaned up. For example for a Slurm instance `DPSslurm`:

```
root@basecm11:~# cmsh
[basecm11]% wlm use DPSslurm
[basecm11->wlm[DPSslurm]]% clear epilogslurmctld
[basecm11->wlm*[DPSslurm*]]% clear prologslurmctld
[basecm11->wlm*[DPSslurm*]]% commit
[basecm11->wlm[DPSslurm]]%
```

- The Kubernetes module for the instance of Kubernetes where DPS is deployed is loaded.

```
root@basecm11:~# module load kubernetes
```

- DPS is uninstalled with Helm. Also any PVC in the `dps` namespace is deleted, and the `dps` namespace is deleted:

```
root@basecm11:~# helm uninstall -n dps dps
root@basecm11:~# kubectl delete pvc --all -n dps --ignore-not-found=true
root@basecm11:~# kubectl delete namespace dps
```

- if the DPS LDAP user, for example `<dps-ldap-username>`, was created for DPS-usage only, then the user can be removed with:

Example

```
root@basecm11:~# cmsh -c "user; remove <dps-ldap-username>; commit"
```

- It is a good idea to make a backup of `named.conf.include` for reference:

Example

```
root@basecm11:~# cd /etc/bind/
root@basecm11:/etc/bind# cp named.conf.include named.conf.include-$(date +%Y-%m-%d_%H-%M-%S)
```

- The DPS zone, which is a zone defined for DPS in the file `/etc/bind/named.conf.include`, must be removed. The zone in the file looks like this:

```
zone "dps" in {
    type master;
    file "dps.zone";
};
```

The other zones in the file must not be changed.

- It is a good idea to make a backup of `/etc/bind/dps.zone` for reference:

```
root@basecm11:~# cd /etc/bind/
root@basecm11:/etc/bind# cp -v dps.zone dps.zone-$(date +%Y-%m-%d_%H-%M-%S)
```

- A reload of `named` can then be carried out with:

```
root@basecm11:~# systemctl reload named
```

and the DNS server should run properly again.

10.3.2 Verify Deployment Failure Due To Mismatching LDAP User Password

If DPS is deployed using a pre-existing user and if the password in LDAP for that user does not match the password used for DPS setup, then the Verify Deployment stage may fail with an error. The deployment halts, prompting for further input:

Example

```
## Progress: 94
#### stage: dps: Verify Deployment
### ERROR FOUND ###
Failed to login: CalledProcessError('Command \'. /etc/profile.d/modules.sh && module load cm-dpsctl && dpsctl
--host=api.dps --port=443 --insecure-tls-skip-verify --credentials-path /cm/local/apps/dps/etc/creds.yaml
login\' returned exit code EX_ERROR<1(0x01)> while expected (<ExitCodes.EX_OK: 0>,\n\tSTDOUT:\n{\n "status":
{\n "diag_msg": "Command execution failed with an error",\n...\n "ok": false\n }\n}\n\tSTDERR:\n')
Undo/Abort/Skip/Retry/Info/Debug/Remote debug: u/a/s/r/i/d/D:
```

The issue can be worked around by modifying the password (`<dps-user-password>`) used by the LDAP DPS user (`<dps-user>`) to match the password of the DPS configuration:

Example

```
root@basecm11:~# cmlsh -c "user; use <dps-user>; set password <dps-user-password>; commit"
```

A retry (typing `r` at the prompt) of the Deployment Verification stage should then succeed:

Example

```
## Progress: 89
#### stage: dps: Setup Slurm
## Progress: 94
#### stage: dps: Verify Deployment
### ERROR FOUND ###
Failed to login: CalledProcessError('Command \'. /etc/profile.d/modules.sh && module load cm-dpsctl && dpsctl
--host=api.dps --port=443 --insecure-tls-skip-verify --credentials-path /cm/local/apps/dps/etc/creds.yaml
```

```
login\' returned exit code EX_ERROR<1(0x01)> while expected (<ExitCodes.EX_OK: 0>,\n\tSTDOUT:\n{\n  "status":  
\n  "diag_msg": "Command execution failed with an error",\n...\n  "ok": false\n }\n}\n\tSTDERR:\n')  
Undo/Abort/Skip/Retry/Info/Debug/Remote debug: u/a/s/r/i/d/D:r
```

Retrying the stage 'Verify Deployment'.

```
#### stage: dps: Verify Deployment  
## Progress: 100
```

```
Took:      01:18 min.  
Progress: 100/100
```

```
##### Finished execution for 'CM NVIDIA Mission Control Setup', status: completed
```

CM NVIDIA Mission Control Setup finished!

11 NVIDIA Mission Control DGX GB200 Measurables

Measurables are metrics, health checks, and enummetrics (appendix G of the *Administrator Manual*).

This section describes the extra DGX GB200 measurables and their parameters that may not have been described in appendix G of the *Administrator Manual*. The DGX GB200 measurables are grouped in this chapter according to the following themes:

- circuit-related metrics (section 11.1)
- leak detection measurables (section 11.2)
- NVLink measurables (section 11.3)
- power shelf measurables (section 11.4)
- cooling distribution unit metrics (section 11.5)
- GPU measurables (section 11.6)
- Prometheus metrics (section 11.7)
- Redfish metrics (section 11.8)
- Redfish enums (section 11.9)
- health checks (section 11.10)

11.1 Circuit-Related DGX GB200 Metrics

Electrical power in the data center is supplied to the DGX GB200 racks from a circuit that is typically from a remote power panel (RPP) or an overhead busway.

Circuit-related metrics for the DGX GB200 platform are shown in table 11.1.

Table 11.1: Circuit-related DGX GB200 metrics

DGX GB200 Circuit Metric	Parameter	Description
CircuitCurrent		
CircuitPhaseCurrent	phase=1	
CircuitPhaseCurrent	phase=2	
CircuitPhaseCurrent	phase=3	
CircuitPower		
RackCircuitCurrent*	circuit=RPP-B12-3	
RackCircuitCurrentLimit*	circuit=RPP-B12-3	
RackCircuitPhaseCurrent**,**	circuit=RPP-B12-3;phase=3	
RackCircuitPower*	circuit=RPP-B12-3	
TotalCircuitCurrent		Total
TotalCircuitCurrentLimit		Total
TotalCircuitPhaseCurrent		Total
TotalCircuitPhaseCurrent	phase=1	Total
TotalCircuitPhaseCurrent	phase=2	Total
TotalCircuitPhaseCurrent	phase=3	Total
TotalCircuitPower		Total

* The parameter set here depends on the listed power circuit name (section 4.7)

** The phase value can be 1, 2, or 3

11.2 Leak Detection DGX GB200 Measurables

Liquid cooling is used in the DGX GB200. The ability to detect leaking coolant is an important enough concern that there are several measurables associated with it.

Leak detection measurables for the DGX GB200 platform are shown in table 11.2:

Table 11.2: Leak detection DGX GB200 measurables

DGX GB200 leak measurable	Parameter	Description
LeakResponseRackElectrical\IsolationStatus		
LeakResponseRackLiquidIso\IsolationStatus		
LeakSensorFaultRack		
DevicesWithLeaks	Total	Number of devices that have detected a leak
RF_Chassis_0_LeakDetector_0_ColdPlate*		
RF_Chassis_0_LeakDetector_0_Manifold*		
RF_Chassis_0_LeakDetector_0_	state	Enum, not metric

...continues

Table 11.2: Leak detection DGX GB200 measurables...continued

DGX GB200 leak measurable	Parameter	Description
ColdPlate*		
RF_Chassis_0_LeakDetector_0_Manifold*	state	Enum, not metric
RF_LD_LeakDetection		Enum
RF_LeakDetector	Chassis_0_LeakDetector_0_ColdPlate*	Enum
RF_LeakDetector	Chassis_0_LeakDetector_0_Manifold*	Enum

* The substring LeakDetector_0 can also take the value LeakDetector_1

11.3 DGX GB200 NVLink Measurables

NVLink-related measurables for the DGX GB200 platform are shown in table 11.3.

The measurables without the RF_ prefix are provided by the ClusterTotal data producer. The measurables with the RF_ prefix are provided via Redfish.

BCM picks up the Redfish measurables and their descriptions automatically. Many measurables still have no associated descriptions. For now, the names of the measurables can be a guide on what they are about.

Table 11.3: DGX GB200 NVLink measurables

DGX GB200 NVLink measurables	Parameter	Description
NVLinkSwitchesClosed		Number of NVLink switches not marked as UP or DOWN
NVLinkSwitchesDown		Number of NVLink switches marked as DOWN
NVLinkSwitchesTotal		Total number of NVLink switches
NVLinkSwitchesUp		Number of NVLink switches marked as UP
RF_NVLink_Port_Nvidia_BitErrorRate*		
RF_NVLink_Port_Nvidia_LinkDowned*		
RF_NVLink_Port_Nvidia_RXNoProtocol*		
RF_NVLink_Port_Nvidia_TXNoProtocol*		
RF_NVLink_Port_Nvidia_TXWait*		
RF_NVLink_Port_RX*		
RF_NVLink_Port_RXFrames_Networking*		
RF_NVLink_Port_RX_Total		
RF_NVLink_Port_TX*		
RF_NVLink_Port_TXFrames_Networking*		

...continues

Table 11.3: DGX GB200 NVLink measurables...continued

DGX GB200 NVLink measurables	Parameter	Description
RF_NVLink_Port_TX_Total		
RF_NVLink_ResourceHealthRollup_Status		
RF_NVLink_ResourceHealth_Status		
RF_NVLink_ResourceState_Status*		
RF_NVLink_Resource_CurrentSpeed*		
RF_NVLink_Resource_LinkState*		
RF_NVLink_Resource_LinkStatus*		
RF_NVLink_Resource_MaxSpeed*		
RF_NVLink_Resource_Nvidia_RXWidth*		
RF_NVLink_Resource_Nvidia_TXWidth*		

* The parameter here is one of 17 ports from the node to the switch.
It can take a value of acp0 to acp17

11.4 DGX GB200 Power Shelf Measurables

The list of power shelf measurables on a DGX GB200 platform are displayed in table 11.4.

These are produced by the redfish_power_shelf, ClusterTotal, and AggregatePowerShelf data producers.

Table 11.4: Power shelf measurables

Power Shelf Measurable	Parameter	Description
AveragePowerShelfTemperature*		Average power shelf temperature
PowerShelvesClosed**		Number of power shelves not marked as UP or DOWN
PowerShelvesDown**		Number of power shelves marked as DOWN
PowerShelvesTotal**		Total number of power shelves
PowerShelvesUp**		Number of power shelves marked as UP
RF_Active_PSU		Number of PSUs of power shelf active
RF_Health_PSU†		Are all PSUs of power shelf active?
RF_PowerShelf_Status†	1	Status of power shelf PSU 1
RF_PowerShelf_Status	total	Status of power shelf PSU, total
RF_Power_Supply_Energy†	1	Energy consumption of PSU 1 since boot
RF_Power_Supply_Energy	total	Energy consumption of all PSUs of power shelf since boot, total
RF_Power_Supply_FanSpeed†	1	Fan speed of PSU 1
RF_Power_Supply_FanSpeed	average	Fan speed of all PSUs of power shelf, averaged
RF_Power_Supply_Id†	1	
RF_Power_Supply_InputCurrent†	1	Input current for PSU 1
RF_Power_Supply_InputCurrent	total	Input current for all PSUs of power shelf, total

...continues

Table 11.4: Power shelf measurables...continued

Power Shelf Measurable	Parameter	Description
RF_Power_Supply_InputPower [†]	1	Input power for PSU 1
RF_Power_Supply_InputPower	total	Input power for all PSUs of power shelf, total
RF_Power_Supply_InputVoltage [†]	1	Input voltage for PSU 1
RF_Power_Supply_LifetimeReading_Energy [†]	1	Input lifetime reading for energy for PSU 1
RF_Power_Supply_Name [†]	1	
RF_Power_Supply_OutputPower [†]	1	Output power for PSU 1
RF_Power_Supply_OutputPower	total	Output power for all PSUs of power shelf, total
RF_Power_Supply_PowerFactor_Input [†]	1	
RF_Power_Supply_RailCurrent [†]	1	Rail Current for PSU 1
RF_Power_Supply_RailCurrent	total	Rail Current for all PSUs of power shelf, total
RF_Power_Supply_RailPower [†]	1	Rail power for PSU 1
RF_Power_Supply_RailPower	total	Rail power for all PSUs of power shelf, total
RF_Power_Supply_RailVoltage [†]	1	Rail voltage for PSUs 1
RF_Power_Supply_Temperature [†]	1	Temperature for PSU 1
RF_Power_Supply_Temperature	average	Temperature for all PSUs of power shelf, averaged
RF_Power_Supply_code_error [†]	1	Code error for PSU 1
RF_Power_Supply_message_error [†]	1	Message error for PSU 1
RF_Summary_Metrics_Id		
RF_Summary_Metrics_Name		
RF_Summary_Metrics_Power		Power consumption (Watts)
RF_Summary_Metrics_Temperature		Temperature (Celsius)
RF_Total_PSU		
TotalPowerShelfActivePSU*		Total power shelf PSUs active
TotalPowerShelfAvailablePSU*		Total power shelf PSUs available
TotalPowerShelfCriticalPSU [‡]		Are all active PSUs below critical threshold?
TotalPowerShelfDegradedPSU [‡]		Are all active PSUs below degraded threshold?
TotalPowerShelfHealthyPSU [‡]		Are all PSUs active?
TotalPowerShelfInputCurrent*		Total power shelf input current
TotalPowerShelfInputPower*		Total power shelf input power
TotalPowerShelfRailCurrent*		Total power shelf rail current
TotalPowerShelfRailPower*		Total power shelf rail power
TotalPowerShelfPSU*		Total power shelf PSU

* From AggregatePowerShelf data producer

** From ClusterTotal data producer

[†] Takes the PSU number as a parameter. On the DGX GB200 it can be from 1 to 6

[‡] Health check

11.5 Cooling Distribution Unit Metrics On The DGX GB200

Metrics for the cooling distribution units (CDUs) of the DGX GB200 platform are shown in table 11.5. The data producers for these are:

- MonitoringSystem for the metrics in the table beginning with the string CDU
- AggregateCDU for the metrics in the table that begin with the strings Average or Total

Table 11.5: DGX GB200 metrics for CDUs

DGX GB200 CDU Metric	Description
AverageCDULiquidDifferentialPressure	Average CDU liquid cooling differential pressure (Pa)
AverageCDULiquidFlow	Average CDU liquid cooling flow (LPM)
AverageCDULiquidReturnTemperature	Average CDU liquid cooling return temperature (°C)
AverageCDULiquidSupplyTemperature	Average CDU liquid cooling supply temperature (°C)
AverageCDULiquidSystemPressure	Average CDU liquid system pressure (Pa)
CDUAvailable	CDU available
CDULiquidDifferentialPressure	CDU liquid differential pressure (Pa)
CDULiquidFlow	CDU liquid flow (LPM)
CDULiquidReturnTemperature	CDU liquid return temperature (°C)
CDULiquidSupplyTemperature	CDU liquid supply temperature (°C)
CDULiquidSystemPressure	CDU liquid system pressure (Pa)
CDUStatus	CDU status (l/s)
TotalCDUAvailable	Total CDU available
TotalCDUStatus	Total CDU status

11.6 GPU Measurables For The DGX GB200

Some basic GPU measurables are covered in appendix G of the *Administrator Manual*.

Extra GPU measurables available on the DGX GB200 platform are:

Table 11.6:

GPU Measurables For The DGX GB200	Parameter	Description
gpu_board_limit_violation*	gpu0	Board violation limit
gpu_board_limit_violation	total	Throttling duration due to board limits
gpu_c2c_link_bandwidth*	gpu0	GPU C2C link bandwidth
gpu_c2c_link_count*	gpu0	GPU C2C link count
gpu_c2c_link_status*	gpu0	GPU C2C link status
gpu_correctable_remapped_rows*	gpu0	Number of remapped rows for correctable errors
gpu_dec_utilization	average	Average GPU decoder utilization
gpu_dec_utilization	gpu0	GPU decoding usage
gpu_ecc_dbe_agg*	gpu0	Total double bit aggregate ECC errors
gpu_ecc_dbe_vol*	gpu0	Total double bit volatile ECC errors
gpu_ecc_sbe_agg*	gpu0	Total single bit aggregate ECC errors

...continues

Table 11.6: ...continued

GPU Measurables For The DGX GB200	Parameter	Description
gpu_ecc_sbe_vol*	gpu0	Total single bit volatile ECC errors
gpu_enc_utilization	average	Average GPU encoder utilization
gpu_enc_utilization*	gpu0	GPU encoding usage
gpu_enforced_power_limit*	gpu0	GPU enforced power limit
gpu_enforced_power_profile_mask*	gpu0	Enforced workload power profile mask
gpu_fabric_status*	gpu0	Status of the nvlink link domain (Enum)
gpu_fan_speed*	gpu0	GPU fan speed percentage
gpu_hbm_memory_temperature*	gpu0	High bandwidth memory (GPU memory) temperature
gpu_health_driver*,**	gpu0	Driver-related status
gpu_health_hostengine**		Host engine status
gpu_health_inforom*,**	gpu0	Inforom status
gpu_health_mcu*,**	gpu0	Microcontroller unit status
gpu_health_mem*,**	gpu0	Memory status
gpu_health_nvlink*,**	gpu0	NVLINK system status
gpu_health_nvswitch_fatal*,**	gpu0	NV switch fatal errors
gpu_health_nvswitch_non_fatal*,**	gpu0	NV switch non fatal errors
gpu_health_overall**		Overall system status
gpu_health_overall*,**	gpu0	Overall system status for the specified GPU
gpu_health_pcie*,**	gpu0	PCIe system status
gpu_health_pmu*,**	gpu0	Power management unit status
gpu_health_power*,**	gpu0	Power status
gpu_health_sm*,**	gpu0	Streaming multiprocessor status
gpu_health_thermal*,**	gpu0	Temperature status
gpu_low_util_violation*	gpu0	Low utilization violation limit
gpu_low_util_violation	total	Throttling duration due to low utilization limits
gpu_mem_clock*	gpu0	GPU memory clock
gpu_mem_copy_utilization	average	Average GPU memory copu utilization
gpu_mem_copy_utilization*	gpu0	Percentage of the GPU memory copy used
gpu_mem_free*	gpu0	Amount of GPU free memory
gpu_mem_total*	gpu0	GPU framebuffer size
gpu_mem_total	total	combined GPU total memory
gpu_mem_used*	gpu0	Amount of GPU used memory
gpu_mem_used	total	combined GPU memory usage
gpu_mem_utilization	average	Average GPU memory utilization
gpu_mem_utilization*	gpu0	Percentage of GPU memory used
gpu_memory_temp	average	Average GPU memory temperature
gpu_memory_temp*	gpu0	GPU memory temperature
gpu_nvlink_crc_data_errors*	gpu0	Total nvlink data CRC errors over all lanes
gpu_nvlink_crc_flit_errors*	gpu0	Total nvlink flit CRC errors over all lanes
gpu_nvlink_total_bandwidth*	gpu0	Total nvlink bandwidth used
gpu_nvlink_total_bandwidth	total	combined GPU nvlink bandwidth

...continues

Table 11.6: ...continued

GPU Measurables For The DGX GB200	Parameter	Description
gpu_performance_state*	gpu0	GPU performance state (0=highest)
gpu_power_management_limit*	gpu0	GPU power management limit
gpu_power_usage*	gpu0	GPU power usage
gpu_power_usage	total	combined GPU power usage
gpu_power_violation*	gpu0	Throttling duration due to power constraints
gpu_power_violation	total	Throttling duration due to power constraints
gpu_reliability_violation*	gpu0	Reliability violation limit
gpu_reliability_violation	total	Throttling duration due to reliability limits
gpu_requested_power_profile_mask*	gpu0	Requested workload power profile mask
gpu_row_remap_failure*	gpu0	Remapping of rows failures
gpu_shutdown_temp*	gpu0	GPU shutdown temperature
gpu_slowdown_temp*	gpu0	GPU slowdown temperature
gpu_sm_clock*	gpu0	GPU shader multiprocessor clock
gpu_sync_boost_violation*	gpu0	Throttling duration due to sync boost constraints
gpu_sync_boost_violation	total	Throttling duration due to sync boost constraints
gpu_temperature	average	Average GPU temperature
gpu_temperature*	gpu0	GPU temperature
gpu_thermal_violation*	gpu0	Throttling duration due to thermal constraints
gpu_thermal_violation	total	Throttling duration due to thermal constraints
gpu_total_app_clocks_violation*	gpu0	App clock violation limit
gpu_total_app_clocks_violation	total	Throttling duration due to application clocks limits
gpu_total_base_clocks_violation*	gpu0	Base clock violation limit
gpu_total_base_clocks_violation	total	Throttling duration due to base clocks limits
gpu_uncorrectable_remapped_rows*	gpu0	Number of remapped rows for uncorrectable errors
gpu_utilization	average	Average GPU utilization
gpu_utilization*	gpu0	GPU utilization percentage
gpu_xid_error*	gpu0	The value is the specific XID error

* parameter can be gpu0, gpu1...

** health check

11.7 Prometheus Metrics For The DGX GB200

Extra Prometheus metrics available on the DGX GB200 platform are shown in table 11.7.

The data producer for the metrics with a prefix of `job_metadata_` is `JobMetadataSampler`. The remaining metrics are produced by `JobSampler`.

BCM picks up the measurables and their descriptions automatically. Many measurables still have no associated descriptions. For now, the names of the measurables can be a guide on what they are about.

Table 11.7:

Prometheus Metrics For The DGX GB200	Description
job_cpu_power_limit	CPU power limit
job_cpu_power_usage	
job_cpu_temperature	
job_cpuacct_usage_seconds	Total CPU time consumed by processes
job_gpu_board_limit_violation	
job_gpu_c2c_link_count	
job_gpu_c2c_link_status	
job_gpu_correctable_remapped_rows	
job_gpu_dec_utilization	
job_gpu_ecc_dbe_agg	
job_gpu_ecc_dbe_vol	
job_gpu_ecc_sbe_agg	
job_gpu_ecc_sbe_vol	
job_gpu_enc_utilization	
job_gpu_enforced_power_limit	
job_gpu_enforced_power_profile_mask	
job_gpu_fabric_status	
job_gpu_fan_speed	
job_gpu_low_util_violation	
job_gpu_mem_clock	
job_gpu_mem_copy_utilization	
job_gpu_mem_free	
job_gpu_mem_total	
job_gpu_mem_used	
job_gpu_mem_utilization	
job_gpu_memory_temp	
job_gpu_nvlink_crc_data_errors	
job_gpu_nvlink_crc_flit_errors	
job_gpu_nvlink_total_bandwidth	Total nvlink bandwidth used
job_gpu_performance_state	
job_gpu_power_management_limit	
job_gpu_power_usage	
job_gpu_power_violation	
job_gpu_reliability_violation	
job_gpu_requested_power_profile_mask	
job_gpu_row_remap_failure	
job_gpu_shutdown_temp	
job_gpu_slowdown_temp	

...continues

Table 11.7: ...continued

Prometheus Metrics For The DGX GB200	Description
job_gpu_sm_clock	
job_gpu_sync_boost_violation	
job_gpu_temperature	
job_gpu_thermal_violation	
job_gpu_total_app_clocks_violation	
job_gpu_total_base_clocks_violation	
job_gpu_uncorrectable_remapped_rows	
job_gpu_utilization	
job_gpu_wasted	The amount of the allocated GPUs that is not used
job_gpu_workload_power_profile	GPU workload power profile status
job_gpu_xid_error	
job_memory_cache_bytes	Page cache, including tmpfs (shmem)
job_memory_failcnt	Number of times that the memory limit has reached the value set in memory.limit_in_bytes
job_memory_mapped_file_bytes	Size of memory-mapped mapped files, including tmpfs
job_memory_mems_w_failcnt	Number of times that the memory plus swap space limit has reached the value set in memory.mems_w.limit_in_bytes
job_memory_mems_w_usage_bytes	Maximum amount of memory and swap space used by processes
job_memory_rss_bytes	Anonymous and swap cache, not including tmpfs
job_memory_swap_bytes	Swap memory usage
job_memory_usage_bytes	Total memory usage processes
job_metadata_allocated_cpu_cores	
job_metadata_allocated_gpus	
job_metadata_is_running	
job_metadata_is_waiting	
job_metadata_num_cpus	
job_metadata_num_nodes	
job_metadata_pending_jobs	
job_metadata_running_jobs	
job_metadata_running_seconds	
job_metadata_waiting_seconds	

11.8 Redfish Metrics For The DGX GB200

Redfish monitoring can be carried out for devices with a Redfish subscription. The `redfishsubscriptions` command (appendix O.2.4 of the *Administrator Manual*) can be used to check if a device has a subscription.

Example

```
[basecm11->device]% redfishsubscriptions node001
hostname      ...initialized ip      kind      port      ssl      subscribed...
-----
node001      ...true      7.241.2.13 GB200     443      true      true
```

Redfish firmware metrics that may be available on the DGX GB200 platform are shown in table 11.8.

As is usual with hardware, the metrics can change according to hardware and especially with firmware changes. The table should therefore be regarded as an indication of what metrics are available, and not a list of what metrics must exist.

BCM picks up the metrics and their descriptions automatically. Many metrics have no associated descriptions. For now, the names of the metrics can be a guide on what they are about.

The list of Redfish metrics available can be found by running:

Example

```
basecm11->monitoring->measurable]% list metric | grep RF_
```

Table 11.8:

Redfish Metrics For The DGX GB200	Description
RF_AOC_NIC1Temp ^[1]	AOC NIC 1 Temperature
RF_AOC_PORTATemp	AOC PORT A Temperature
RF_AOC_PORTBTemp	AOC PORT B Temperature
RF_AOC_PORTCTemp	AOC PORT C Temperature
RF_Active_PSU ^[2]	
RF_Average_Temp	
RF_BF3_Slot_1_NIC_Temp_0	BF3 Slot 1 NIC Temperature 0
RF_BF3_Slot_2_NIC_Temp_0	BF3 Slot 2 NIC Temperature 0
RF_BF3_Slot_1_Port_0_Temp	BF3 Slot 1 Port 0 Temperature
RF_BF3_Slot_1_Port_1_Temp	BF3 Slot 1 Port 1 Temperature
RF_BF3_Slot_2_Port_0_Temp	BF3 Slot 2 Port 0 Temperature
RF_BF3_Slot_2_Port_1_Temp	BF3 Slot 2 Port 1 Temperature
RF_BF3_Slot_1_Temp	BF3 Slot 1 Temperature
RF_BF3_Slot_2_Temp	BF3 Slot 2 Temperature
RF_BMCTemp	BMC Temperature
RF_BMC_0_DCSCM_Temp_0	BMC 0 DCSCM Temperature 0
RF_C2CPU1Temp	C2 CPU1 Temperature
RF_C2CPU2Temp	C2 CPU2 Temperature
RF_CPU1RearTemp	CPU1 Rear Temperature
RF_CPU1Temp	CPU1 Temperature
RF_CPU1_VCCHV	CPU1_VCCHV

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
RF_CPU1_VCCIN	CPU1_VCCIN
RF_CPU1_VCCON	CPU1_VCCON
RF_CPU1_VRMHVTemp	CPU1_VRMHV Temperature
RF_CPU1_VRMINTemp	CPU1_VRMIN Temperature
RF_CPU1_VRMONTemp	CPU1_VRMON Temperature
RF_CPU2RearTemp	CPU2 Rear Temperature
RF_CPU2Temp	CPU2 Temperature
RF_CPU2_VCCHV	CPU2_VCCHV
RF_CPU2_VCCIN	CPU2_VCCIN
RF_CPU2_VCCON	CPU2_VCCON
RF_CPU2_VRMHVTemp	CPU2_VRMHV Temperature
RF_CPU2_VRMINTemp	CPU2_VRMIN Temperature
RF_CPU2_VRMONTemp	CPU2_VRMON Temperature
RF_CPU_0_CoreUtil_0 ^[3]	CPU_0_CoreUtil_0
<i>In the preceding row, the substring CoreUtil_0 can also take values of CoreUtil_1, CoreUtil_2..., up to CoreUtil_71</i>	
RF_CPU_0_CpuFreq_0 ^[3]	CPU_0_CpuFreq_0
RF_CPU_0_Energy_0 ^[3]	CPU_0_Energy_0
RF_CPU_0_Power_0 ^[3]	CPU_0_Power_0
RF_CPU_0_Processor_code_error ^[3]	
RF_CPU_0_Processor_message_error ^[3]	
RF_CPU_0_Processor_Metrics__Nvidia_AccumulatedGPUContextUtilizationDuration ^[3]	
RF_CPU_0_Processor_Metrics__Nvidia_AccumulatedSMUtilizationDuration ^[3]	
RF_CPU_0_Processor_Metrics__Nvidia_EDPViolationState ^[3]	
RF_CPU_0_Processor_Metrics__Nvidia_GlobalSoftwareViolationThrottleDuration ^[3]	
RF_CPU_0_Processor_Metrics__Nvidia_HardwareViolationThrottleDuration ^[3]	
RF_CPU_0_Processor_Metrics__Nvidia_MemoryPageRetirement ^[3]	
RF_CPU_0_Processor_Metrics__Nvidia_MemorySpareChannelPresence ^[3]	
RF_CPU_0_Processor_Metrics__Nvidia_PCIERXBytes ^[3]	
RF_CPU_0_Processor_Metrics__Nvidia_PCIETXBytes ^[3]	
RF_CPU_0_Processor_Metrics__Nvidia_PerformanceState ^[3]	
RF_CPU_0_Processor_Metrics__Nvidia_	

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
PowerBreakPerformanceState ^[3]	
RF_CPU_0_Processor_Metrics_PowerLimitThrottleDuration ^[3]	
RF_CPU_0_Processor_Metrics_ThermalLimitThrottleDuration ^[3]	
RF_CPU_0_TempAvg_0 ^[3]	CPU_0_TempAvg_0
RF_CPU_0_TempLimit_0 ^[3]	CPU_0_TempLimit_0
RF_Chassis_0_FAN_1_FRONT ^[4]	Chassis 0 FAN 1 FRONT
RF_Chassis_0_FAN_1_PWM ^[4]	Chassis 0 FAN 1 PWM
RF_Chassis_0_FAN_1_REAR ^[4]	Chassis 0 FAN 1 REAR
RF_Chassis_0_Front_IO_Temp_0	Chassis 0 Front IO Temp 0
RF_Chassis_0_LeakDetector_0_ColdPlate ^[5]	Chassis 0 LeakDetector 0 ColdPlate
RF_Chassis_0_LeakDetector_0_Manifold ^[5]	Chassis 0 LeakDetector 0 Manifold
RF_Chassis_0_LeakDetector_1_ColdPlate ^[5]	Chassis 0 LeakDetector 1 ColdPlate
RF_Chassis_0_LeakDetector_1_Manifold ^[5]	Chassis 0 LeakDetector 1 Manifold
RF_Chassis_0_TotalHSC_Power_0	Chassis 0 TotalHSC Power 0
RF_Chassis_Intru_Reading	
RF_Chassis_Intru_ReadingRangeMax	
RF_Chassis_Intru_ReadingRangeMin	
RF_DIMM_SlotPartLocationContext_Location	
RF_DIMM_SlotState_Status	
RF_DIMM_Slot_BusWidthBits	
RF_DIMM_Slot_Capacity	
RF_DIMM_Slotcode_error	
RF_DIMM_Slotmessage_error	
RF_DIMM_Slot_OperatingSpeed	
RF_DIMM_Slot_DataWidthBits	
RF_DIMM_Slot_FirmwareRevision	
RF_DIMM_Slot_Nvidia_RowRemappingFailed	
RF_DIMM_Slot_Nvidia_RowRemappingPending	
RF_DIMM_Slot_OperatingSpeed	
RF_DIMM_Slot_Rank	
RF_Energy_0	Energy_0
RF_FPGA_Temp	FPGA Temp

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
RF_FPGA_ARTTemp	FPGA_ART Temp
RF_GPU_0_DRAM_0_Memory_MetricsCorrectableECCErrorCount_LifeTime ^[6]	
RF_GPU_0_DRAM_0_Memory_MetricsUncorrectableECCErrorCount_LifeTime ^[6]	
RF_GPU_0_DRAM_0_Memory_Metrics_BandwidthUtilization ^[6]	
RF_GPU_0_DRAM_0_Memory_Metrics_CapacityUtilizationPercent ^[6]	
RF_GPU_0_DRAM_0_Memory_Metrics_OperatingSpeed ^[6]	
RF_GPU_0_Processor_MetricsCorrectableErrorCount_PCIEErrors ^[6]	
RF_GPU_0_Processor_MetricsFatalErrorCount_PCIEErrors ^[6]	
RF_GPU_0_Processor_MetricsL0ToRecoveryCount_PCIEErrors ^[6]	
RF_GPU_0_Processor_MetricsNAKReceivedCount_PCIEErrors ^[6]	
RF_GPU_0_Processor_MetricsNAKSentCount_PCIEErrors ^[6]	
RF_GPU_0_Processor_MetricsNonFatalErrorCount_PCIEErrors ^[6]	
RF_GPU_0_Processor_MetricsReplayCount_PCIEErrors ^[6]	
RF_GPU_0_Processor_MetricsReplayRolloverCount_PCIEErrors ^[6]	
RF_GPU_0_Processor_MetricsUnsupportedRequestCount_PCIEErrors ^[6]	
RF_GPU_0_Processor_Metrics_BandwidthUtilization ^[6]	
RF_GPU_0_Processor_Metrics_CoreVoltage ^[6]	
RF_GPU_0_Processor_Metrics_LifeTime_CorrectableECCErrors ^[6]	
RF_GPU_0_Processor_Metrics_LifeTime_UncorrectableECCErrors ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_AccumulatedGPUContextUtilizationDuration ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_AccumulatedSMUtilizationDuration ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_DMMAUtilizationPercent ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_FP16ActivityPercent ^[6]	

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
RF_GPU_0_Processor_Metrics_Nvidia_FP32ActivityPercent ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_FP64ActivityPercent ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_GlobalSoftwareViolationThrottleDuration ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_GraphicsEngineActivityPercent ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_HMMAUtilizationPercent ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_HardwareViolationThrottleDuration ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_IMMAUtilizationPercent ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_IntegerActivityUtilizationPercent ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_NVDecUtilizationPercent ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_NVJpgUtilizationPercent ^[6]	
RF_GPU_0_Processor_Metrics_NVLinkDataRxBandwidth ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_NVLinkDataTxBandwidth ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_NVLinkRawRxBandwidth ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_NVLinkRawTxBandwidth ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_NVOfaUtilizationPercent ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_PCIERXBytes ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_PCIERawRxBandwidth ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_PCIERawTxBandwidth ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_PCIETXBytes ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_SMAActivityPercent ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_SMOccupancyPercent ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_SMUtilizationPercent ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_	

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
SRAMECCErrorThresholdExceeded ^[6]	
RF_GPU_0_Processor_Metrics_Nvidia_TensorCoreActivityPercent ^[6]	
RF_GPU_0_Processor_Metrics_OperatingSpeed ^[6]	
RF_GPU_0_Processor_Metrics_PowerLimitThrottleDuration ^[6]	
RF_GPU_0_Processor_Metrics_ThermalLimitThrottleDuration ^[6]	
RF_GPU_0_Processor_Metricscode_error ^[6]	
RF_GPU_0_Processor_Metricsmessage_error ^[6]	
RF_GPU_0_Processor_Reset_Metrics_ConventionalResetEntry ^[6]	
RF_GPU_0_Processor_Reset_Metrics_ConventionalResetExit ^[6]	
RF_GPU_0_Processor_Reset_Metrics_FundamentalResetEntry ^[6]	
RF_GPU_0_Processor_Reset_Metrics_FundamentalResetExit ^[6]	
RF_GPU_0_Processor_Reset_Metrics_IROtResetExit ^[6]	
RF_GPU_0_Processor_Reset_Metrics_PF_FLR_ResetEntry ^[6]	
RF_GPU_0_Processor_Reset_Metrics_PF_FLR_ResetExit ^[6]	
RF_HGX_BMC_0_Temp_0	HGX BMC 0 Temp 0
RF_HGX_Baseboard_0CoreCount_ProcessorSummary	
RF_HGX_Baseboard_0Count_ProcessorSummary	
RF_HGX_Baseboard_0Model_ProcessorSummary	
RF_HGX_Baseboard_0State_Status	
RF_HGX_Baseboard_0TotalSystemMemoryGiB_MemorySummary	
RF_HGX_Baseboard_0_Description	
RF_HGX_Baseboard_0_LastResetTime	
RF_HGX_Baseboard_0_Nvidia_ISTModeEnabled	
RF_HGX_Baseboard_0_PowerMode	
RF_HGX_Baseboard_0_PowerState	
RF_HGX_CPU_0State_Status ^[3]	
RF_HGX_CPU_0_AssetTag ^[3]	

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
RF_HGX_CPU_0_PartLocation_ServiceLabel ^[3]	
RF_HGX_CPU_0_SKU ^[3]	
RF_HGX_CPU_0code_error ^[3]	
RF_HGX_CPU_0message_error ^[3]	
RF_HGX_Chassis_0_TotalGPU_Power_0	HGX Chassis 0 TotalGPU Power 0
RF_HGX_GPU_0State_Status ^[6]	
RF_HGX_GPU_0_DRAM_0_Power_0 ^[6]	HGX GPU 0 DRAM 0 Power 0
RF_HGX_GPU_0_DRAM_0_Temp_0 ^[6]	HGX GPU 0 DRAM 0 Temp 0
RF_HGX_GPU_0_Energy_0 ^[6]	HGX GPU 0 Energy 0
RF_HGX_GPU_0_MaxPower ^[6]	
RF_HGX_GPU_0_MinPower ^[6]	
RF_HGX_GPU_0_Power_0 ^[6]	HGX GPU 0 Power 0
RF_HGX_GPU_0_SKU ^[6]	
RF_HGX_GPU_0_TEMP_0 ^[6]	HGX GPU 0 TEMP 0
RF_HGX_GPU_0_TEMP_1 ^[6]	HGX GPU 0 TEMP 1
RF_HGX_GPU_0_Voltage_0 ^[6]	HGX GPU 0 Voltage 0
RF_HGX_ProcessorModule_0_Exhaust_Temp_0 ^[12]	HGX ProcessorModule 0 Exhaust Temp 0
RF_HGX_ProcessorModule_0_Inlet_Temp_0 ^[12]	HGX ProcessorModule 0 Inlet Temp 0
RF_HGX_ProcessorModule_0_Inlet_Temp_1 ^[12]	HGX ProcessorModule 0 Inlet Temp 0
RF_HotswapTemp	Hotswap Temp
RF_IO_Board_0_CX7_0_Port_0_Temp	IO Board 0 CX7 0 Port 0 Temp
RF_IO_Board_0_CX7_0_Temp	IO Board 0 CX7 0 Temp
RF_IO_Board_0_CX7_0_Temp_0	IO Board 0 CX7 0 Temp 0
RF_IO_Board_0_CX7_1_Port_0_Temp	IO Board 0 CX7 1 Port 0 Temp
RF_IO_Board_0_CX7_1_Temp	IO Board 0 CX7 1 Temp
RF_IO_Board_0_CX7_1_Temp_0	IO Board 0 CX7 1 Temp 0
RF_IO_Board_1_CX7_0_Port_0_Temp	IO Board 1 CX7 0 Port 0 Temp
RF_IO_Board_1_CX7_0_Temp	IO Board 1 CX7 0 Temp
RF_IO_Board_1_CX7_0_Temp_0	IO Board 1 CX7 0 Temp 0

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
RF_IO_Board_1_CX7_1_Port_0_Temp	IO Board 1 CX7 1 Port 0 Temp
RF_IO_Board_1_CX7_1_Temp	IO Board 1 CX7 1 Temp
RF_IO_Board_1_CX7_1_Temp_0	IO Board 1 CX7 1 Temp 0
RF_InletTemp	Inlet Temp
RF_InterswitchPort_0_Port_MetricsRXFrames_Networking	
RF_InterswitchPort_0_Port_MetricsRXMulticastFrames_Networking	
RF_InterswitchPort_0_Port_MetricsRXUnicastFrames_Networking	
RF_InterswitchPort_0_Port_MetricsTXDiscards_Networking	
RF_InterswitchPort_0_Port_MetricsTXFrames_Networking	
RF_InterswitchPort_0_Port_MetricsTXMulticastFrames_Networking	
RF_InterswitchPort_0_Port_MetricsTXUnicastFrames_Networking	
RF_InterswitchPort_0_Port_Metrics_Nvidia_BitErrorRate	
RF_InterswitchPort_0_Port_Metrics_Nvidia_LinkDowned	
RF_InterswitchPort_0_Port_Metrics_Nvidia_LinkErrorRecovery	
RF_InterswitchPort_0_Port_Metrics_Nvidia_MalformedPackets	
RF_InterswitchPort_0_Port_Metrics_Nvidia_NeighborMTUDiscards	
RF_InterswitchPort_0_Port_Metrics_Nvidia_QP1Dropped	
RF_InterswitchPort_0_Port_Metrics_Nvidia_RXRemotePhysicalErrors	
RF_InterswitchPort_0_Port_Metrics_Nvidia_RXSwitchRelayErrors	
RF_InterswitchPort_0_Port_Metrics_Nvidia_TXWait	
RF_InterswitchPort_0_Port_Metrics_Nvidia_VL15Dropped	
RF_InterswitchPort_0_Port_Metrics_Nvidia_VL15TXBytes	
RF_InterswitchPort_0_Port_Metrics_Nvidia_VL15TXPackets	

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
RF_InterswitchPort_0_Port_Metrics_RXBytes	
RF_InterswitchPort_0_Port_Metrics_RXErrors	
RF_InterswitchPort_0_Port_Metrics_TXBytes	
RF_InterswitchPort_0_ResourceState_Status	
RF_InterswitchPort_0_Resource_CurrentSpeed	
RF_InterswitchPort_0_Resource_LinkState	
RF_InterswitchPort_0_Resource_LinkStatus	
RF_InterswitchPort_0_Resource_MaxSpeed	
RF_LD_Chassis_0_LeakDetector_0_ColdPlate ^[5]	Chassis 0 LeakDetector 0 ColdPlate
RF_LD_Chassis_0_LeakDetector_0_Manifold ^[5]	Chassis 0 LeakDetector 0 Manifold
RF_LD_Chassis_0_LeakDetector_1_ColdPlate ^[5]	Chassis 0 LeakDetector 1 ColdPlate
RF_LD_Chassis_0_LeakDetector_1_Manifold ^[5]	Chassis 0 LeakDetector 1 Manifold
RF_LD_LeakDetection ^[5]	Leak Detection Systems
RF_LeakDetection ^[5]	Leak Detection Systems
RF_M2_SSD1Temp	M2_SSD1 Temp
RF_M2_SSD2Temp	M2_SSD2 Temp
RF_MB1.8VCC	MB 1.8VCC
RF_MB1.8VSB	MB 1.8VSB
RF_MB12V	MB 12V
RF_MB12VSB	MB 12VSB
RF_MB3.3VCC	MB 3.3VCC
RF_MB3.3VSB	MB 3.3VSB
RF_MB5VCC	MB 5VCC
RF_MB5VSB	MB 5VSB
RF_MGX_NVSwitch_0_TEMP_0	MGX NVSwitch 0 TEMP 0
RF_MGX_NVSwitch_1_TEMP_0	MGX NVSwitch 1 TEMP 0
RF_MemCntl_0_Freq_0	MemCntl_0_Freq_0
RF_MemCntl_1_Freq_0	MemCntl_1_Freq_0
RF_NVLinkManagement_0_Port_MetricsRXFrames_Networking	

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
RF_NVLinkManagement_0_Port_MetricsRXMulticastFrames_Networking	
RF_NVLinkManagement_0_Port_MetricsRXUnicastFrames_Networking	
RF_NVLinkManagement_0_Port_MetricsTXDiscards_Networking	
RF_NVLinkManagement_0_Port_MetricsTXFrames_Networking	
RF_NVLinkManagement_0_Port_MetricsTXMulticastFrames_Networking	
RF_NVLinkManagement_0_Port_MetricsTXUnicastFrames_Networking	
RF_NVLinkManagement_0_Port_Metrics_Nvidia_BitErrorRate	
RF_NVLinkManagement_0_Port_Metrics_Nvidia_LinkDowned	
RF_NVLinkManagement_0_Port_Metrics_Nvidia_LinkErrorRecovery	
RF_NVLinkManagement_0_Port_Metrics_Nvidia_MalformedPackets	
RF_NVLinkManagement_0_Port_Metrics_Nvidia_NeighborMTUDiscards	
RF_NVLinkManagement_0_Port_Metrics_Nvidia_QP1Dropped	
RF_NVLinkManagement_0_Port_Metrics_Nvidia_RXRemotePhysicalErrors	
RF_NVLinkManagement_0_Port_Metrics_Nvidia_RXSwitchRelayErrors	
RF_NVLinkManagement_0_Port_Metrics_Nvidia_TXWait	
RF_NVLinkManagement_0_Port_Metrics_Nvidia_VL15Dropped	
RF_NVLinkManagement_0_Port_Metrics_Nvidia_VL15TXBytes	
RF_NVLinkManagement_0_Port_Metrics_Nvidia_VL15TXPackets	
RF_NVLinkManagement_0_Port_Metrics_RXBytes	
RF_NVLinkManagement_0_Port_Metrics_RXErrors	
RF_NVLinkManagement_0_Port_Metrics_TXBytes	
RF_NVLinkManagement_0_ResourceState_Status	
RF_NVLinkManagement_0_Resource_	

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
CurrentSpeed	
RF_NVLinkManagement_0_Resource_LinkState	
RF_NVLinkManagement_0_Resource_LinkStatus	
RF_NVLinkManagement_0_Resource_MaxSpeed	
RF_NVLink_0_Port_MetricsRXFrames_Networking ^[7]	
RF_NVLink_0_Port_MetricsRXMulticastFrames_Networking ^[7]	
RF_NVLink_0_Port_MetricsRXUnicastFrames_Networking ^[7]	
RF_NVLink_0_Port_MetricsTXDiscards_Networking ^[7]	
RF_NVLink_0_Port_MetricsTXFrames_Networking ^[7]	
RF_NVLink_0_Port_MetricsTXMulticastFrames_Networking ^[7]	
RF_NVLink_0_Port_MetricsTXUnicastFrames_Networking ^[7]	
RF_NVLink_0_Port_Metrics_Nvidia_BitErrorRate ^[7]	
RF_NVLink_0_Port_Metrics_Nvidia_EffectiveError ^[7]	
RF_NVLink_0_Port_Metrics_Nvidia_LinkDowned ^[7]	
RF_NVLink_0_Port_Metrics_Nvidia_LinkErrorRecovery ^[7]	
RF_NVLink_0_Port_Metrics_Nvidia_MalformedPackets ^[7]	
RF_NVLink_0_Port_Metrics_Nvidia_NVLinkDataRxBandwidth ^[7]	
RF_NVLink_0_Port_Metrics_Nvidia_NVLinkDataTxBandwidth ^[7]	
RF_NVLink_0_Port_Metrics_Nvidia_NVLinkRawRxBandwidth ^[7]	
RF_NVLink_0_Port_Metrics_Nvidia_NVLinkRawTxBandwidth ^[7]	
RF_NVLink_0_Port_Metrics_Nvidia_NeighborMTUDiscards ^[7]	
RF_NVLink_0_Port_Metrics_Nvidia_QP1Dropped ^[7]	
RF_NVLink_0_Port_Metrics_Nvidia_RXNoProtocolBytes ^[7]	

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
RF_NVLink_0_Port_Metrics_ Nvidia_RXRemotePhysicalErrors ^[7]	
RF_NVLink_0_Port_Metrics_ Nvidia_RXSwitchRelayErrors ^[7]	
RF_NVLink_0_Port_Metrics_ Nvidia_TXNoProtocolBytes ^[7]	
RF_NVLink_0_Port_Metrics_ Nvidia_TXWait ^[7]	
RF_NVLink_0_Port_Metrics_ Nvidia_VL15Dropped ^[7]	
RF_NVLink_0_Port_Metrics_ Nvidia_VL15TXBytes ^[7]	
RF_NVLink_0_Port_Metrics_ Nvidia_VL15TXPackets ^[7]	
RF_NVLink_0_Port_Metrics_ RXBytes ^[7]	
RF_NVLink_0_Port_Metrics_ RXErrors ^[7]	
RF_NVLink_0_Port_Metrics_ TXBytes ^[7]	
RF_NVLink_0_ResourceState_ Status ^[7]	
RF_NVLink_0_Resource_CurrentSpeed ^[7]	
RF_NVLink_0_Resource_LinkState ^[7]	
RF_NVLink_0_Resource_LinkStatus ^[7]	
RF_NVLink_0_Resource_MaxSpeed ^[7]	
RF_NVLink_0_Resource_Nvidia_ RXWidth ^[7]	
RF_NVLink_0_Resource_Nvidia_ TXWidth ^[7]	
RF_NVME_M2_0_Temp_0	NVME M2 0 Temp 0
RF_NVMe_SSDATemp	NVMe_SSDA Temp
RF_NVSwitch_0_ResourceState_ Status	
RF_NVSwitch_0_Resource_#Switch.Reset_ target	
RF_NVSwitch_0_Resource_CurrentBandwidth	
RF_NVSwitch_0_Resource_Enabled	
RF_NVSwitch_0_Resource_FirmwareVersion	
RF_NVSwitch_0_Resource_MaxBandwidth	
RF_NVSwitch_0_Resource_Nvidia_ SwitchIsolationMode	
RF_NVSwitch_1_ResourceState_ Status	

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
Status	
RF_NVSwitch_1_Resource_#Switch.Reset_target	
RF_NVSwitch_1_Resource_CurrentBandwidth	
RF_NVSwitch_1_Resource_Enabled	
RF_NVSwitch_1_Resource_FirmwareVersion	
RF_NVSwitch_1_Resource_MaxBandwidth	
RF_NVSwitch_1_Resource_Nvidia_SwitchIsolationMode	
RF_P1_DIMMA_DTemp	P1_DIMMA D Temp
RF_P1_DIMME_HTemp	P1_DIMME H Temp
RF_P2_DIMMA_DTemp	P2_DIMMA D Temp
RF_P2_DIMME_HTemp	P2_DIMME H Temp
RF_PCB_Inlet_Temp	PCB Inlet Temp
RF_PCH1.05V	PCH 1.05V
RF_PCHPVNN	PCH PVNN
RF_PCHTemp	PCH Temp
RF_PCIE_0_ResourceState_Status	
RF_PCIE_0_Resource_ActiveWidth	
RF_PCIE_0_Resource_CurrentSpeed	
RF_PCIE_0_Resource_MaxSpeed	
RF_PCIE_0_Resource_Width	
RF_PCIE_DeviceLanesInUse_PCIEInterface	
RF_PCIE_DeviceMaxLanes_PCIEInterface	
RF_PCIE_DeviceMaxPCIeType_PCIEInterface	
RF_PCIE_DevicePCIeType_PCIEInterface	
RF_PCIE_DeviceState_Status	
RF_PCIE_Device_Nvidia_AERCorrectableErrorStatus	
RF_PCIE_Device_Nvidia_AERUncorrectableErrorStatus	
RF_PCIE_Device_Nvidia_NVLinkReferenceClockEnabled	
RF_PCIE_Device_Nvidia_PCIEReferenceClockEnabled	
RF_PCIE_Devicecode_error	
RF_PCIE_Devicemessage_error	
RF_PDB_0_HSC_0_Cur_0	PDB 0 HSC 0 Cur 0
RF_PDB_0_HSC_0_Pwr_0	PDB 0 HSC 0 Pwr 0
RF_PDB_0_HSC_0_Temp_0	PDB 0 HSC 0 Temp 0
RF_PDB_0_HSC_0_Volt_In_0	PDB 0 HSC 0 Volt In 0
RF_PDB_0_HSC_0_Volt_Out_0	PDB 0 HSC 0 Volt Out 0

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
RF_PDB_0_HSC_1_Cur_0	PDB 0 HSC 1 Cur 0
RF_PDB_0_HSC_1_Pwr_0	PDB 0 HSC 1 Pwr 0
RF_PDB_0_HSC_1_Temp_0	PDB 0 HSC 1 Temp 0
RF_PDB_0_HSC_1_Volt_In_0	PDB 0 HSC 1 Volt In 0
RF_PDB_0_HSC_1_Volt_Out_0	PDB 0 HSC 1 Volt Out 0
RF_PDB_0_Inlet_Temp_0	PDB 0 Inlet Temp 0
RF_PDB_0_Vreg_0_Temp_0	PDB 0 Vreg 0 Temp 0
RF_PDB_0_Vreg_0_Temp_1	PDB 0 Vreg 0 Temp 1
RF_PDB_0_Vreg_1_Temp_0	PDB 0 Vreg 1 Temp 0
RF_PDB_0_Vreg_1_Temp_1	PDB 0 Vreg 1 Temp 1
RF_PDB_TOTAL_CURRENT	
RF_PDB_TOTAL_PWR	
RF_PDB_TOTAL_VOLT_IN	
RF_PDB_TOTAL_VOLT_OUT	
RF_PS1_Status_Reading	
RF_PS1_Status_ReadingRangeMax	
RF_PS1_Status_ReadingRangeMin	
RF_PWConsumption	PW Consumption
RF_PeripheralTemp	Peripheral Temp
RF_PowerShelf_Status ^{[2],[8]}	Power shelf status of PSU
RF_PowerShelf_Status ^{[2],[8]}	Power shelf status total
RF_Power_0	Power_0
RF_Power_PowerControl	
RF_Power_PowerSupplies	
RF_Power_Redundancy	
RF_Power_Supply_Energy ^{[2],[8]}	Energy consumption since boot of PSU
RF_Power_Supply_Energy ^[9]	Energy consumption since boot of all PSUs
RF_Power_Supply_FanSpeed ^{[2],[8]}	Fan speed of PSU
RF_Power_Supply_FanSpeed ^[10]	Fan speed average of all PSUs
RF_Power_Supply_Id ^{[2],[8]}	ID of PSU
RF_Power_Supply_InputCurrent ^{[2],[8]}	Input current for PSU
RF_Power_Supply_InputCurrent ^[9]	Input current of all PSUs
RF_Power_Supply_InputPower ^{[2],[8]}	Input power for PSU
RF_Power_Supply_InputPower ^[9]	Input power total of all PSUs
RF_Power_Supply_InputVoltage ^{[2],[8]}	Input voltage for PSU
RF_Power_Supply_LifetimeReading_Energy ^{[2],[8]}	Energy consumption of PSU over life-time

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
RF_Power_Supply_Name ^{[2],[8]}	Name of PSU
RF_Power_Supply_OutputPower ^{[2],[8]}	Output power of PSU
RF_Power_Supply_OutputPower ^[9]	Output power total of all PSUs
RF_Power_Supply_PowerFactor_Input ^{[2],[8]}	Input power factor of PSU
RF_Power_Supply_RailCurrent ^{[2],[8]}	Rail current of PSU
RF_Power_Supply_RailCurrent ^[9]	Rail current total of all PSUs
RF_Power_Supply_RailPower ^{[2],[8]}	Rail current of PSU
RF_Power_Supply_RailPower ^[9]	Rail power total of all PSUs
RF_Power_Supply_RailVoltage ^{[2],[8]}	Rail voltage of PSU
RF_Power_Supply_Temperature ^{[2],[8]}	Temperature of PSU
RF_Power_Supply_Temperature ^[10]	Temperature average of all PSUs
RF_Power_Supply_code_error ^{[2],[8]}	Code error of PSU
RF_Power_Supply_message_error ^{[2],[8]}	Message error of PSU
RF_Power_Voltages	
RF_ProcessorModule_0_CPU_0_ CoreUtil_0 ^[11]	ProcessorModule 0 CPU 0 CoreUtil 0. The substring CoreUtil_0 can also take values of CoreUtil_1, CoreUtil_2... up to CoreUtil_71
RF_ProcessorModule_0_CPU_0_ CpuFreq_0 ^[11]	ProcessorModule 0 CPU 0 CpuFreq 0
RF_ProcessorModule_0_CPU_0_ Energy_0 ^[11]	ProcessorModule 0 CPU 0 Energy 0
RF_ProcessorModule_0_CPU_0_ EnforcedEDPc_0 ^[11]	ProcessorModule 0 CPU 0 EnforcedEDPc 0
RF_ProcessorModule_0_CPU_0_ EnforcedEDPp_0 ^[11]	ProcessorModule 0 CPU 0 EnforcedEDPp 0
RF_ProcessorModule_0_CPU_0_ Power_0 ^[11]	ProcessorModule 0 CPU 0 Power 0
RF_ProcessorModule_0_CPU_0_ TempAvg_0 ^[11]	ProcessorModule 0 CPU 0 TempAvg 0
RF_ProcessorModule_0_CPU_0_ TempLimit_0 ^[11]	ProcessorModule 0 CPU 0 TempLimit 0
RF_ProcessorModule_0_MemCntl_ 0_Freq_0 ^[11]	ProcessorModule 0 MemCntl 0 Freq 0
RF_ProcessorModule_0_Vreg_ 0_CpuPower_0 ^[11]	ProcessorModule 0 Vreg 0 CpuPower 0
RF_ProcessorModule_0_Vreg_ 0_CpuVoltage_0 ^[11]	ProcessorModule 0 Vreg 0 CpuVoltage 0
RF_ProcessorModule_0_Vreg_ 0_SocPower_0 ^[11]	ProcessorModule 0 Vreg 0 SocPower 0
RF_ProcessorModule_0_Vreg_ 0_SocVoltage_0 ^[11]	ProcessorModule 0 Vreg 0 SocVoltage 0

...continues

Table 11.8: ...continued

Redfish Metrics For The DGX GB200	Description
RF_SYS_ART1Temp	SYS_ART1 Temp
RF_SYS_ART2Temp	SYS_ART2 Temp
RF_StorageBackplane_0_SSD_0_Temp_0	StorageBackplane 0 SSD 0 Temp 0
RF_StorageBackplane_0_SSD_2_Temp_0	StorageBackplane 0 SSD 2 Temp 0
RF_StorageBackplane_1_SSD_0_Temp_0	StorageBackplane 1 SSD 0 Temp 0
RF_StorageBackplane_1_SSD_2_Temp_0	StorageBackplane 1 SSD 2 Temp 0
RF_Summary_Metrics_Id ^[2]	
RF_Summary_Metrics_Name ^[2]	
RF_Summary_Metrics_Power ^[2]	Power consumption (Watts)
RF_Summary_Metrics_Temperature ^[2]	Temperature (Celsius)
RF_SystemTemp	System Temp
RF_Total_PSU ^[2]	
RF_VBAT_Reading	
RF_VBAT_ReadingRangeMax	
RF_VBAT_ReadingRangeMin	
RF_Vreg_0_CpuPower_0	Vreg_0_CpuPower_0
RF_Vreg_0_CpuVoltage_0	Vreg_0_CpuVoltage_0
RF_Vreg_0_SocPower_0	Vreg_0_SocPower_0
RF_Vreg_0_SocVoltage_0	Vreg_0_SocVoltage_0
RF_Vreg_1_CpuPower_0	Vreg_1_CpuPower_0
RF_Vreg_1_CpuVoltage_0	Vreg_1_CpuVoltage_0
RF_Vreg_1_SocPower_0	Vreg_1_SocPower_0
RF_Vreg_1_SocVoltage_0	Vreg_1_SocVoltage_0

^[1] The substring NIC1Temp can also be NIC2Temp, NIC3Temp...

^[2] from redfish_power_shelf data producer

^[3] The substring CPU_0 can also be CPU_1

^[4] The substring FAN_1 can also be FAN_2, FAN_3...

^[5] From redfish_leak data producer

^[6] The substring GPU_0 can also be GPU_1, GPU_2...

^[7] On the DGX GB200 the value of <number> in ...NVLink_<number>... can be from 0 to 71

^[8] Takes the PSU number as a parameter. On the DGX GB200 it can be from 1 to 6

^[9] Takes total as a parameter. On the DGX GB200 it means the total value of the 6 PSUs

^[10] Takes average as a parameter. On the DGX GB200 it means the average value of the 6 PSUs

^[11] On the DGX GB200 the value of <number> in ...ProcessorModule_<number>... can be 0 or 1

^[12] The substring ProcessorModule_0 can also be ProcessorModule_1

11.9 Redfish Enums For The DGX GB200

Redfish monitoring can be carried out for devices with a Redfish subscription (page 90).

Redfish firmware enums that may be available on the DGX GB200 platform are shown in table 11.9.

As is usual with hardware, the metrics can change according to hardware and especially with firmware

changes. The table should therefore be regarded as an indication of what metrics are available, and not a list of what metrics must exist.

The list of Redfish enums that are there on the DGX GB200 platform can be found by running:

Example

```
basecm11->monitoring->measurable]% list enum | grep RF_
```

Table 11.9:

Redfish Enums For The DGX GB200	Parameter	Description
RF_BF3_Slot_1_NIC_Temp_0	state	
RF_BF3_Slot_2_NIC_Temp_0	state	
RF_BF3_Slot_2_Port_0_Temp	state	
RF_BF3_Slot_2_Port_1_Temp	state	
RF_BF3_Slot_2_Temp	state	
RF_BMC_0_DCSCM_Temp_0	state	
RF_Chassis_0_FAN_1_FRONT ^[1]	state	
RF_Chassis_0_FAN_1_PWM	state	
RF_Chassis_0_FAN_1_REAR	state	
RF_CPU_0_Processor__Nvidia_EDPViolationState		
RF_CPU_0_Processor__Nvidia_PerformanceState		
RF_CPU_0_Processor__Nvidia_PowerBreakPerformanceState		
RF_CPU_1_Processor__Nvidia_EDPViolationState		
RF_CPU_1_Processor__Nvidia_PerformanceState		
RF_CPU_1_Processor__Nvidia_PowerBreakPerformanceState		
RF_DIMM_SlotHealthRollup_Status	state	
RF_DIMM_SlotHealth_Status	state	
RF_ERoT_BMC_0_PowerState		
RF_ERoT_BMC_0HealthRollup_Status	state	
RF_ERoT_BMC_0Health_Status	state	
RF_ERoT_BMC_0_PowerState		
RF_HGX_BMC_0_Temp_0	state	
RF_HGX_Baseboard_0HealthRollup_Status	state	
RF_HGX_Baseboard_0Health_Status	state	
RF_HGX_Baseboard_0_PowerState		
RF_HGX_CPU_0HealthRollup_Status	state	
RF_HGX_CPU_0Health_Status	state	
RF_HGX_CPU_0_PowerState		

...continues

Table 11.9: ...continued

Redfish Enums For The DGX GB200	Parameter	Description
RF_HGX_CPU_1HealthRollup_Status	state	
RF_HGX_CPU_1Health_Status	state	
RF_HGX_CPU_1_PowerState		
RF_HGX_Chassis_0_TotalGPU_Power_0	state	
RF_HGX_ERoT_BMC_0HealthRollup_Status	state	
RF_HGX_ERoT_BMC_0Health_Status	state	
RF_HGX_ERoT_CPU_0HealthRollup_Status	state	
RF_HGX_ERoT_CPU_0Health_Status	state	
RF_HGX_ERoT_CPU_0_PowerState		
RF_HGX_ERoT_CPU_1HealthRollup_Status	state	
RF_HGX_ERoT_CPU_1Health_Status	state	
RF_HGX_ERoT_FPGA_0HealthRollup_Status	state	
RF_HGX_ERoT_FPGA_0Health_Status	state	
RF_HGX_ERoT_FPGA_1HealthRollup_Status	state	
RF_HGX_ERoT_FPGA_1Health_Status	state	
RF_HGX_CPU_0_PowerState		
RF_HGX_CPU_1_PowerState		
RF_HGX_GPU_0HealthRollup_Status ^[2]	state	
RF_HGX_GPU_0Health_Status ^[2]	state	
RF_HGX_GPU_0_DRAM_0_Power_0 ^[2]	state	
RF_HGX_GPU_0_DRAM_0_Temp_0 ^[2]	state	
RF_HGX_GPU_0_Energy_0 ^[2]	state	
RF_HGX_GPU_0_PowerState ^[2]		
RF_HGX_GPU_0_Power_0 ^[2]	state	
RF_HGX_GPU_0_TEMP_0 ^[2]	state	
RF_HGX_GPU_0_TEMP_1 ^[2]	state	
RF_HGX_GPU_0_Voltage_0 ^[2]	state	
RF_HGX_IROt_GPU_0HealthRollup_Status ^[2]	state	
RF_HGX_IROt_GPU_0Health_Status ^[2]	state	
RF_HGX_IROt_GPU_1_PowerState ^[2]		
RF_HGX_ProcessorModule_0_Exhaust_Temp_0	state	
RF_HGX_ProcessorModule_0_Inlet_Temp_0	state	
RF_HGX_ProcessorModule_0_Inlet_Temp_1	state	
RF_HGX_ProcessorModule_1_Exhaust_Temp_0	state	
RF_HGX_ProcessorModule_1_Inlet_Temp_0	state	
RF_HGX_ProcessorModule_1_Inlet_Temp_1	state	
RF_IO_Board_0_CX7_0_Port_0_Temp	state	
RF_IO_Board_0_CX7_0_Temp	state	
RF_IO_Board_0_CX7_0_Temp_0	state	

...continues

Table 11.9: ...continued

Redfish Enums For The DGX GB200	Parameter	Description
RF_IO_Board_0_CX7_1_Port_0_Temp	state	
RF_IO_Board_0_CX7_1_Temp	state	
RF_IO_Board_0_CX7_1_Temp_0	state	
RF_IO_Board_1_CX7_0_Port_0_Temp	state	
RF_IO_Board_1_CX7_0_Temp	state	
RF_IO_Board_1_CX7_0_Temp_0	state	
RF_IO_Board_1_CX7_1_Port_0_Temp	state	
RF_IO_Board_1_CX7_1_Temp	state	
RF_IO_Board_1_CX7_1_Temp_0	state	
RF_IrOT_CX7_0HealthRollup_Status	state	
RF_IrOT_CX7_0Health_Status	state	
RF_IrOT_CX7_1HealthRollup_Status	state	
RF_IrOT_CX7_1Health_Status	state	
RF_IrOT_CX7_2HealthRollup_Status	state	
RF_IrOT_CX7_2Health_Status	state	
RF_IrOT_CX7_3HealthRollup_Status	state	
RF_IrOT_CX7_3Health_Status	state	
RF_LD_LeakDetection ^[3]		
RF_LeakDetector ^[3]	Chassis_0_LeakDetector_0_ColdPlate	
RF_LeakDetector ^[3]	Chassis_0_LeakDetector_0_Manifold	
RF_LeakDetector ^[3]	Chassis_0_LeakDetector_1_ColdPlate	
RF_LeakDetector ^[3]	Chassis_0_LeakDetector_1_Manifold	
RF_NVLinkManagement_0_ResourceHealthRollup_Status	state	
RF_NVLinkManagement_0_ResourceHealth_Status	state	
RF_NVLink_ResourceHealthRollup_Status	state	
RF_NVLink_ResourceHealth_Status	state	
RF_NVLink_ResourceState_Status ^[4]	acpO	
RF_NVLink_Resource_LinkState ^[4]	acpO	
RF_NVLink_0_ResourceHealthRollup_Status	state	
RF_NVLink_0_ResourceHealth_Status	state	
RF_NVME_M2_0_Temp_0	state	
RF_NVSwitch_0_ResourceHealthRollup_Status ^[5]	state	

...continues

Table 11.9: ...continued

Redfish Enums For The DGX GB200	Parameter	Description
RF_NVSwitch_0_ResourceHealth_Status ^[5]	state	
RF_InterswitchPort_0_ResourceHealthRollup_Status	state	
RF_InterswitchPort_0_ResourceHealth_Status	state	
RF_PCIE_0_ResourceHealthRollup_Status	state	
RF_PCIE_0_ResourceHealth_Status	state	
RF_PCIE_DeviceHealthRollup_Status	state	
RF_PCIE_DeviceHealth_Status	state	
RF_PDB_0_HSC_0_Cur_0	state	
RF_PDB_0_HSC_0_Pwr_0	state	
RF_PDB_0_HSC_0_Temp_0	state	
RF_PDB_0_HSC_0_Volt_In_0	state	
RF_PDB_0_HSC_0_Volt_Out_0	state	
RF_PDB_0_HSC_1_Cur_0	state	
RF_PDB_0_HSC_1_Pwr_0	state	
RF_PDB_0_HSC_1_Temp_0	state	
RF_PDB_0_HSC_1_Volt_In_0	state	
RF_PDB_0_HSC_1_Volt_Out_0	state	
RF_PDB_0_Inlet_Temp_0	state	
RF_PDB_0_Vreg_0_Temp_0	state	
RF_PDB_0_Vreg_0_Temp_1	state	
RF_PDB_0_Vreg_1_Temp_0	state	
RF_PDB_0_Vreg_1_Temp_1	state	
RF_ProcessorModule_0_CPU_0_CpuFreq_0 ^[6]	state	
RF_ProcessorModule_0_CPU_0_Energy_0 ^[6]	state	
RF_ProcessorModule_0_CPU_0_EnforcedEDPc_0 ^[6]	state	
RF_ProcessorModule_0_CPU_0_EnforcedEDPp_0 ^[6]	state	
RF_ProcessorModule_0_CPU_0_Power_0 ^[6]	state	
RF_ProcessorModule_0_CPU_0_TempAvg_0 ^[6]	state	
RF_ProcessorModule_0_CPU_0_TempLimit_0 ^[6]	state	
RF_ProcessorModule_0_MemCntl_0_Freq_0 ^[6]	state	
RF_ProcessorModule_0_Vreg_0_CpuPower_0 ^[6]	state	
RF_ProcessorModule_0_Vreg_0_CpuVoltage_0 ^[6]	state	
RF_ProcessorModule_0_Vreg_0_SocPower_0 ^[6]	state	
RF_ProcessorModule_0_Vreg_0_SocVoltage_0 ^[6]	state	
RF_ProcessorModule_0_CPU_0_CoreUtil_0 ^[6]	state	
<i>In the preceding row, the substring CoreUtil_0 can also take values of CoreUtil_1, CoreUtil_2..., up to CoreUtil_71</i>		
RF_StorageBackplane_0_SSD_0_Temp_0 ^[7]	state	

...continues

Table 11.9: ...continued

Redfish Enums For The DGX GB200	Parameter	Description
[1]		The number used in the substring Chassis_0 and the substring FAN_ can vary
[2]		The substring GPU_0 can be GPU_1, GPU_2...
[3]		Produced by redfish_leak data producer
[4]		The parameter substring acp0 can be acp1, acp2...
[5]		The substring NVSwitch_0 can be NVSwitch_1, NVSwitch_2...
[6]		The substring ProcessorModule_0 can be ProcessorModule_1, ProcessorModule_2...
[7]		The substring SSD_0 can be SSD_1, SSD_2..., and the substring StorageBackplane_0 can be StorageBackplane_1, StorageBackplane_2...

11.10 Health Checks For The DGX GB200

Redfish health checks that may be additionally available on the DGX GB200 platform are shown in table 11.10.

As is usual with hardware, the health checks can change according to hardware and especially with firmware changes. The table should therefore be regarded as an indication of the health checks that are available, and not a list of the health checks that must exist.

The list of Redfish health checks that are there can be found by running:

Example

```
[basecm11->monitoring->measurable]% list healthcheck
```

Table 11.10: DGX GB200 Health Checks

DGX GB200 Health Check	Parameter	Description
gpu_health_driver	gpu0	Driver-related status
gpu_health_hostengine		Host engine status
gpu_health_inforom	gpu0	Inforom status
gpu_health_mcu	gpu0	Microcontroller unit status
gpu_health_mem	gpu0	Memory status
gpu_health_nvlink	gpu0	NVLINK system status
gpu_health_nvswitch_fatal	gpu0	NV switch fatal errors
gpu_health_nvswitch_non_fatal	gpu0	NV switch non-fatal errors
gpu_health_overall		Overall system status
gpu_health_overall	gpu0	Overall system status for GPU0
gpu_health_pcie	gpu0	PCIe system status
gpu_health_pmu	gpu0	Power management unit status
gpu_health_power	gpu0	Power status
gpu_health_sm	gpu0	Streaming multiprocessor status
gpu_health_thermal	gpu0	Temperature status
nvsm_pci_health		Checks health of PCI devices in a DGX system

...continues

Table 11.10: DGX GB200 Health Checks...continued

DGX GB200 Health Check	Parameter	Description
nvsm_show_health		Checks health of DGX system
NMXController		Is NMX Controller in the correct state?