GRAID SupremeRAID™
SR-1000
User Guide

Version 1.0.0

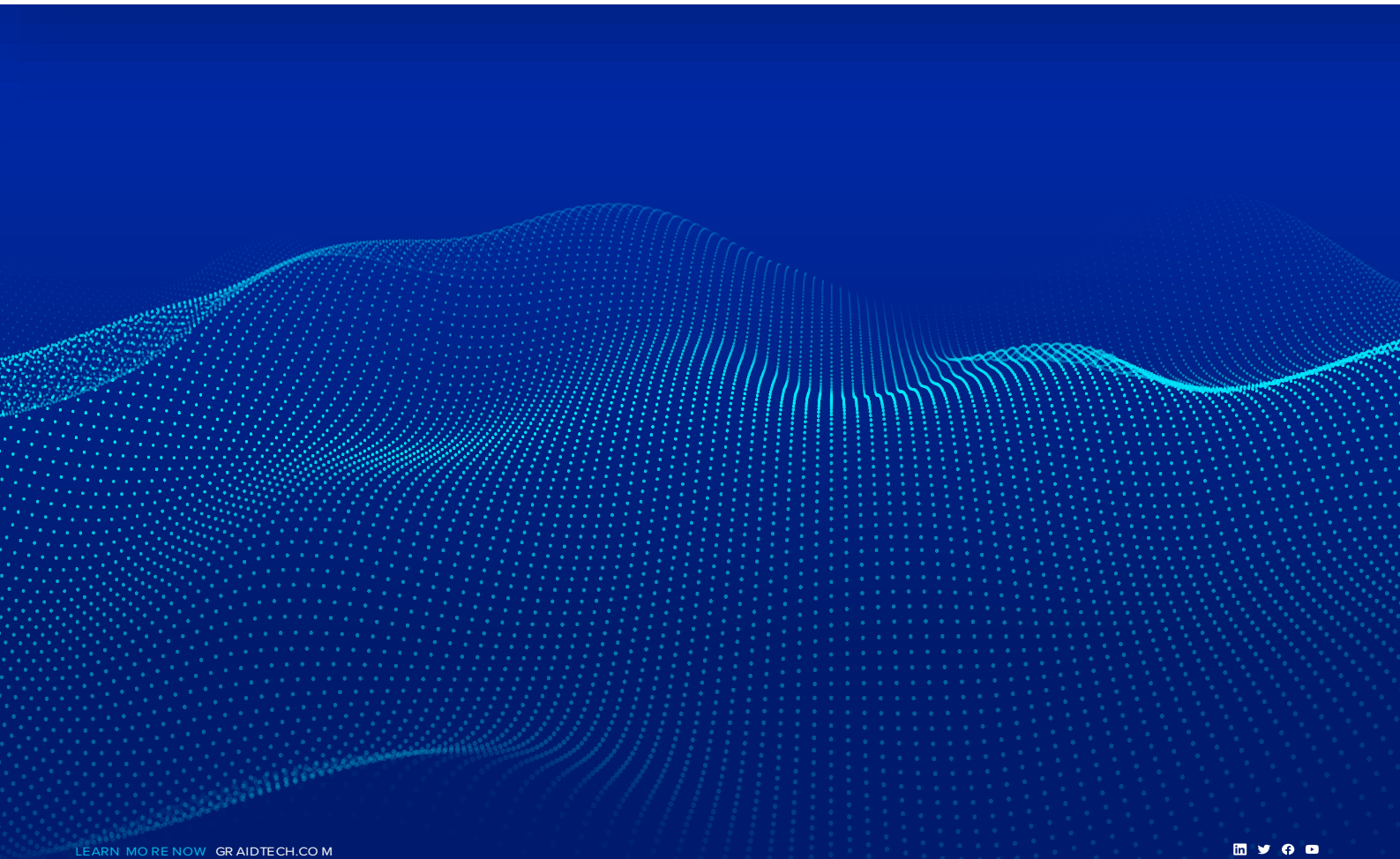


Table of Contents

Introduction

About this document

This guide is intended for administrators and users of the GRAID SupremeRAID™ SR-1000. The guide contains instructions on how to install the SupremeRAID™ software package and how to manage the RAID components by command-line tool.

What is GRAID SupremeRAID™

GRAID SupremeRAID™ is the most powerful, high-speed data protection solution specially designed for NVMe SSDs. GRAID SupremeRAID™ works by installing a virtual NVMe controller onto the operating system and integrating a PCIe RAID card into the system equipped with a high-performance AI processor to handle all RAID operations of the virtual NVMe controller.

GRAID SupremeRAID™ SR-1000 Specifications

GRAID SupremeRAID™ Software Module Specifications	
Supported RAID levels	RAID 0, 1, 10, 5, 6
Maximum number of physical drives	32
Maximum number of drive groups	4
Maximum number of virtual drives per drive group	8
Maximum size of the drive group	Defined by physical drive sizes
OS Support	Linux: CentOS 8.3, RHEL 8.4, Ubuntu 20.04, openSUSE Leap 15.2, SLES 15 SP2

GRAID SupremeRAID™ Card Specifications	
Host Interface	x16 PCIe Gen 3.0
Max Power Consumption	50 W
Form Factor	2.713" H x 6.137" L Single Slot
Product Weight	132.6 g

Installation

This chapter will describe how to install the GRAID SupremeRAID™ software package.

Installation prerequisites

1. Install the GRAID SupremeRAID™ SR-1000 card into a PCIe x16 slot.
2. Make sure IOMMU function is disabled in the system BIOS.
3. Download the GRAID SupremeRAID™ software package from the GRAIDTECH.COM or contact your GRAID partner.

Installing on Linux

CentOS and RHEL

1. Install the package dependencies and build tool for dkms.

```
$ sudo yum install --enablerepo=extras epel-release
$ sudo yum update
$ sudo yum install vim wget make automake gcc gcc-c++ kernel-devel dkms ipmitool
```

2. Add kernel option. In this step we will block the nouveau driver from loading if installed and disable iommu if youhave not already disabled it in the system BIOS.

```
$ sudo vim /etc/default/grub
```

Appen ***"rd.driver.blacklist=nouveau iommu=pt"*** to **GRUB_CMDLINE_LINUX** then make grub2 config, if you areusing UEFI boot:

```
$ sudo grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg
```

If you are using legacy boot:

```
$ sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

3. Reboot and make sure the grub config has applied. You can check **/proc/cmdline** file for applied grub config. Forexample:

```
[root@graid-iu /]# cat /proc/cmdline
BOOT_IMAGE=(hd2,gpt2)/vmlinuz-4.18.0-240.22.1.el8_3.x86_64 root=/dev/mapper/cl-root ro crashkernel=auto resum
e=/dev/mapper/cl-swap rd.lvm.lv=cl/root rd.lvm.lv=cl/swap rhgb quiet rd.driver.blacklist=nouveau iommu=pt
[root@graid-iu /]#
```

4. Install NVIDIA driver.

```
$ wget https://us.download.nvidia.com/XFree86/Linux-x86_64/470.57.02/NVIDIA-Linux-x86_64-470.57.02.run
$ chmod +x NVIDIA-Linux-x86_64-470.57.02.run
$ sudo ./NVIDIA-Linux-x86_64-470.57.02.run
```

- **If you have nouveau driver step 2 disables that or you can let the NVIDIA driver installer disable thenouveau driver, but you will have to reboot and install the driver again.**
- **When prompted select to Install with DKMS.**

5. Confirm the NVIDIA GPU is working via the command ***nvidia-smi*** to see if any GPUs are found:

```
root@graid-third:/# nvidia-smi
Wed Jun 23 15:25:19 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 465.31       Driver Version: 465.31       CUDA Version: 11.3     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA Quadro P...  Off  | 00000000:81:00.0 Off |                  N/A |
| 68%   38C    P0    21W /  75W |      0MiB /  5059MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

6. Install the graid software rpm package.

```
$ sudo rpm -Uvh <graid-sr-x.x.x-x.gxxxxxxx.000.x86_64.rpm>
```

7. Apply the license to activate the GRAID service.

```
$ sudo graidctl apply license <LICENSE_KEY>
```

Ubuntu

1. Install the package dependencies and build tool for dkms.

```
$ sudo apt-get update
$ sudo apt-get install make automake gcc g++ linux-headers-$(uname -r) dkmsipmitool initramfs-tools
```

2. Add kernel option. In this step we will block the nouveau driver from loading if installed and disable iommu if youhave not already disabled it in the system BIOS.

```
$ sudo vim /etc/default/grub
```

Appen ***"iommu=pt"*** to **GRUB_CMDLINE_LINUX** then make grub2 config, if you are using UEFI boot:

```
$ sudo grub2-mkconfig -o /boot/efi/EFI/ubuntu/grub.cfg
```

If you are using legacy boot:

```
$ sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

3. Append **"blacklist nouveau"** and **"options nouveau modeset=0"** to the end of **/etc/modprobe.d/blacklist-nouveau.conf** file to disable the nouveau driver.
4. Update initramfs.

```
$ sudo update-initramfs -u
```

5. Reboot and make sure the grub config has applied. You can check **/proc/cmdline** file for applied grub config. Forexample:

```
root@graid-third:/# cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-5.4.0-74-generic root=/dev/mapper/ubuntu--vg-ubuntu--lv ro
rd.driver.blacklist=nouveau iommu=pt
root@graid-third:/#
```

6. Install NVIDIA driver.

```
$ wget https://us.download.nvidia.com/XFree86/Linux-x86_64/470.57.02/NVIDIA-Linux-x86_64-470.57.02.run
$ chmod +x NVIDIA-Linux-x86_64-470.57.02.run
$ sudo ./NVIDIA-Linux-x86_64-470.57.02.run
```

- **If you have nouveau driver step 3 disables that or you can let the NVIDIA driver installer disable thenouveau driver, but you will have to reboot and install the driver again.**
- **When prompted select to Install with DKMS.**

7. Confirm the NVIDIA GPU is working via the command ***nvidia-smi*** to see if any GPUs are found:

```
root@graid-third:/# nvidia-smi
Wed Jun 23 15:25:19 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 465.31       Driver Version: 465.31       CUDA Version: 11.3     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA Quadro P...  Off  | 00000000:81:00.0 Off |                  N/A |
| 68%   38C    P0    21W /  75W |      0MiB /  5059MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

8. Install the graid software debian package.

```
$ sudo dpkg -i <graid-sr-x.x.x-x.gxxxxxxx.000.x86_64.deb>
```

9. Apply the license to activate the GRAID service.

```
$ sudo graidctl apply license <LICENSE_KEY>
```

OpenSUSE

1. Install openSUSE and select all online repositories.
2. Install the package dependencies and build tool for dkms.

```
$ zypper install sudo vim wget libpci3 dkms kernel-default-devel ipmitool
```

3. Add kernel option. In this step we will block the nouveau driver from loading if installed and disable iommu if youhave not already disabled it in the system BIOS.

```
$ sudo vim /etc/default/grub
```

Appen ***"iommu=pt"*** to **GRUB_CMDLINE_LINUX** then make grub2 config, if you are using UEFI boot:

```
$ sudo grub2-mkconfig -o /boot/efi/EFI/opensuse/grub.cfg
```

If you are using legacy boot:

```
$ sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

4. Append **"blacklist nouveau"** to the end of **/etc/modprobe.d/50-blacklist.conf** file to disable the nouveau driver.
5. Set **allow_unsupported_modules** option to **1** in **/etc/modprobe.d/10-unsupported-modules.conf** file.
6. Update initrd.

```
$ sudo mkinitrd
```

7. Reboot and make sure the grub config has applied, you can check **/proc/cmdline** file for applied grub config. Forexample:

```
graid-third:~ # cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.3.18-59.5-default root=UUID=7560fe42-0275-4618-b8a0-0785
765610c9 modprobe.blacklist=nouveau iommu=pt splash=silent quiet mitigations=auto
graid-third:~ #
```

8. Install NVIDIA driver.

```
$ wget https://us.download.nvidia.com/XFree86/Linux-x86_64/470.57.02/NVIDIA-Linux-x86_64-470.57.02.run
$ chmod +x NVIDIA-Linux-x86_64-470.57.02.run
$ sudo ./NVIDIA-Linux-x86_64-470.57.02.run
```

- **If you have nouveau driver step 3 disables that or you can let the NVIDIA driver installer disable thenouveau driver, but you will have to reboot and install the driver again.**
- **When prompted select to Install with DKMS.**

9. Confirm the NVIDIA GPU is working via the command ***nvidia-smi*** to see if any GPUs are found:

```
root@graid-third:/# nvidia-smi
Wed Jun 23 15:25:19 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 465.31       Driver Version: 465.31       CUDA Version: 11.3     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA Quadro P...  Off  | 00000000:81:00.0 Off |                  N/A |
| 68%   38C    P0    21W /  75W |      0MiB /  5059MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

10. Install the graid software rpm package.

```
$ sudo rpm -Uvh <graid-sr-x.x.x-x.gxxxxxxx.000.x86_64.rpm>
```

11. Apply the license to activate the GRAID service.

```
$ sudo graidctl apply license <LICENSE_KEY>
```

SLES

1. Install SLES with the following extensions and modules:
    1. SUSE Package Hub 15 SP2 x86_64
    2. Desktop Applications Module 15 SP2 x86_64
    3. Development Tools Module 15 SP2 x86_64
2. Install the package dependencies and build tool for dkms.

```
$ zypper install sudo vim wget libpci3 dkms kernel-default-devel ipmitool
```

3. Add kernel option. In this step we will block the nouveau driver from loading if installed and disable iommu if youhave not already disabled it in the system BIOS.

```
$ sudo vim /etc/default/grub
```

Appen ***"iommu=pt"*** to **GRUB_CMDLINE_LINUX** then make grub2 config, if you are using UEFI boot:

```
$ sudo grub2-mkconfig -o /boot/efi/EFI/sles/grub.cfg
```

If you are using legacy boot:

```
$ sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

4. Append **"blacklist nouveau"** to the end of **/etc/modprobe.d/50-blacklist.conf** file to disable the nouveau driver.
5. Set **allow_unsupported_modules** option to **1** in **/etc/modprobe.d/10-unsupported-modules.conf** file.
6. Update initrd.

```
$ sudo mkinitrd
```

7. Reboot and make sure the grub config has applied. You can check **/proc/cmdline** file for applied grub config. Forexample:

```
graid-third:~ # cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.3.18-59.5-default root=UUID=7560fe42-0275-4618-b8a0-0785
765610c9 modprobe.blacklist=nouveau iommu=pt splash=silent quiet mitigations=auto
graid-third:~ #
```

8. Install NVIDIA driver.

```
$ wget https://us.download.nvidia.com/XFree86/Linux-x86_64/470.57.02/NVIDIA-Linux-x86_64-470.57.02.run
$ chmod +x NVIDIA-Linux-x86_64-470.57.02.run
$ sudo ./NVIDIA-Linux-x86_64-470.57.02.run
```

- **If you have nouveau driver step 3 disables that or you can let the NVIDIA driver installer disable thenouveau driver, but you will have to reboot and install the driver again.**
- **When prompted select to Install with DKMS.**

9. Confirm the NVIDIA GPU is working via the command ***nvidia-smi*** to see if any GPUs are found:

```
root@graid-third:/# nvidia-smi
Wed Jun 23 15:25:19 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 465.31       Driver Version: 465.31       CUDA Version: 11.3     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA Quadro P...  Off  | 00000000:81:00.0 Off |                  N/A |
| 68%   38C    P0    21W /  75W |      0MiB /  5059MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

10. Install the graid software rpm package.

```
$ sudo rpm -Uvh <graid-sr-x.x.x-x.gxxxxxxx.000.x86_64.rpm>
```

11. Apply the license to activate the GRAID service.

```
$ sudo graidctl apply license <LICENSE_KEY>
```

Management

Overview of GRAID SupremeRAID™ Software Module

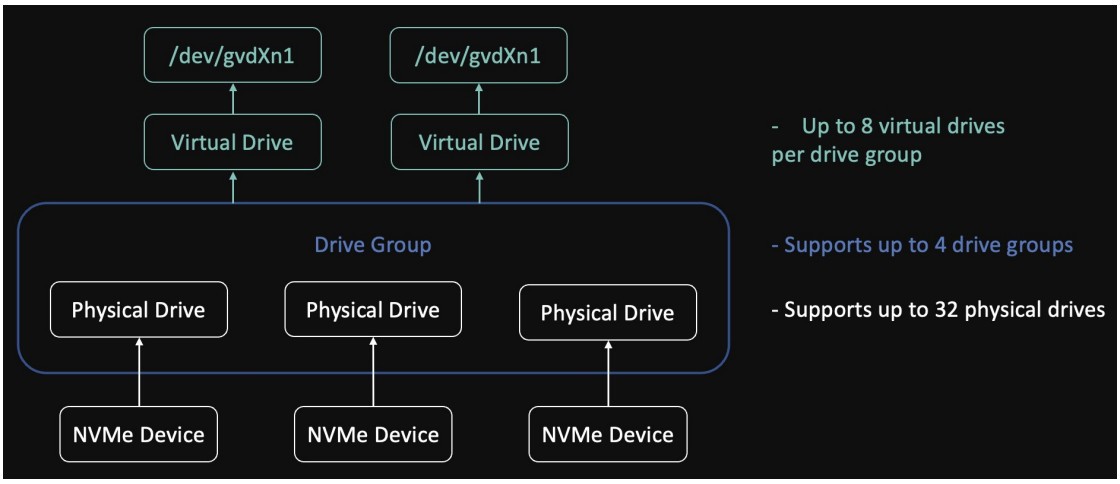
There are three major components of GRAID SupremeRAID™ Software Module:

1. **graidctl** - The command-line management tool.
2. **graid_server** - The management daemon to handle the request from graidctl and control the driver.
3. **graid.ko** - The driver kernel module.

RAID components

There are three major RAID components in SupremeRAID™:

1. **Physical Drive (PD)**: Since NVMe drives are not direct attached to the SupremeRAID™ controller, the user must tellthe controller which SSDs can be managed.
2. **Drive Group (DG)**: The main component of RAID logic, essentially it is a RAID group.
3. **Virtual Drive (VD)**: The user can create multiple virtual drives in the same DG for multiple applications.



Overview of graidctl

Syntax

Use the following syntax to run graidctl commands from your terminal window:

```
graidctl [command] [OBJECT_TYPE] [OBJECT_ID] [flags]
```

where command, OBJECT_TYPE, OBJECT_ID, and flags are:

- **command:** Specifies the operation that you want to perform on one or more resources, for example create, list, describe, delete.
- **OBJECT_TYPE:** Specifies the object type. Object types are case-sensitive, for example license, physical_drive, drive_group.
- **OBJECT_ID**: Specifies the object id. Some commands support operations on multiple objects at the same time, you can simply specify the OBJECT_ID one by one or use dash to describe a range of OBJECT_ID.

For example, to delete physical drive 1, 3, 4, 5 at the same time:

```
$ graidctl delete physical_drive 1 3-5
```

- **flags**: Specifies optional flags.

For example, to force the deletion of a physical drive, append the **--force** flag:

```
$ sudo graidctl delete physical_drive 0 --force
```

If you need help, run ***graidctl help*** from the terminal window.

License Management

To complete the installation, apply the license:

Apply license

To apply the license, run:

```
$ graidctl apply license <LICENSE_KEY>
```

Output example for applying invalid license and valid license:

```
root@graid-third:/home/alven# graidctl apply license 1XD93HXA-QRXGESWG-W8M8A4L7-ADANCSLA
✖ Invalid license: 1XD93HXA-QRXGESWG-W8M8A4L7-ADANCSLA
root@graid-third:/home/alven# graidctl apply license 1XD93HXA-QRXGESWG-W8M8A4L7-ADANCSLB
✔ Apply license 1XD93HXA-QRXGESWG-W8M8A4L7-ADANCSLB successfully.
```

Please note that the license information shown in the figure above is for demonstration only, please use the license key on the license card attached to the product box during the actual installation process.

Check license information

To obtain the license information, run:

```
$ graidctl describe license
```

Output example:

```
root@graid-third:/# graidctl describe license
✔ Describe license successfully.
{
    "LicenseState": "APPLIED",
    "LicenseKey": "1XD93HXA-QRXGESWG-W8M8A4L7-ADANCSLB"
}
```

Output content:

Field	Description
License State	The license state
License Key	The applied license key

The license state:

State	Description
UNAPPLIED	The license has not yet been applied
APPLIED	A valid license has been applied
INVALID	A valid license has been applied before, but cannot detect valid RAID card

Host Drive Information

List NVMe drives

To list all NVMe drives that can be created as physical drives, run:

```
$ graidctl list nvme_drive
```

Output example:

```
root@graid-third:/home/alven# graidctl list nvme_drive
✔ List nvme drive successfully.
```

DEVICE PATH (10)	NQN	MODEL	CAPACITY
/dev/nvme0	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A038T1L8	KCM61VUL3T20	3.2 TB
/dev/nvme1	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A06QT1L8	KCM61VUL3T20	3.2 TB
/dev/nvme2	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04WT1L8	KCM61VUL3T20	3.2 TB
/dev/nvme3	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z050A002T1L8	KCM61VUL3T20	3.2 TB
/dev/nvme4	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A003T1L8	KCM61VUL3T20	3.2 TB
/dev/nvme5	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z060A005T1L8	KCM61VUL3T20	3.2 TB
/dev/nvme6	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z0F0A031T1L8	KCM61VUL3T20	3.2 TB
/dev/nvme7	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A002T1L8	KCM61VUL3T20	3.2 TB
/dev/nvme8	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A001T1L8	KCM61VUL3T20	3.2 TB
/dev/nvme9	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04HT1L8	KCM61VUL3T20	3.2 TB

Output content:

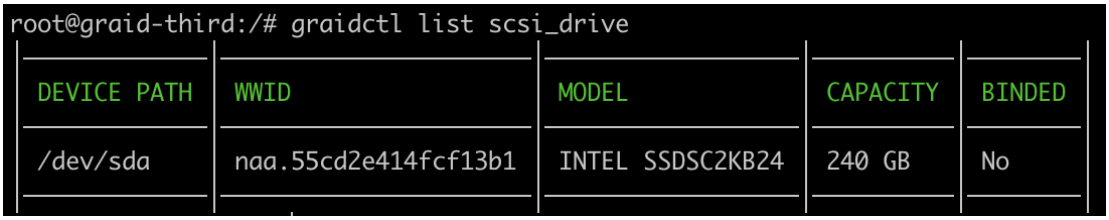
Field	Description
DEVICE PATH	The block device path of the drive
NQN	The NVMe Qualified Name of the drive
MODEL	The model number of the drive
CAPACITY	The capacity of the drive

List SAS/SATA drives

To list all SAS/SATA drives that can be allocated as physical drives, run:

```
$ graidctl list scsi_drive
```

Output example:



Output content:

Field	Description
DEVICE PATH	The block device path of the drive
WWID	Worldwide Identification of the drive
MODEL	The model number the drive model
CAPACITY	The capacity of the drive

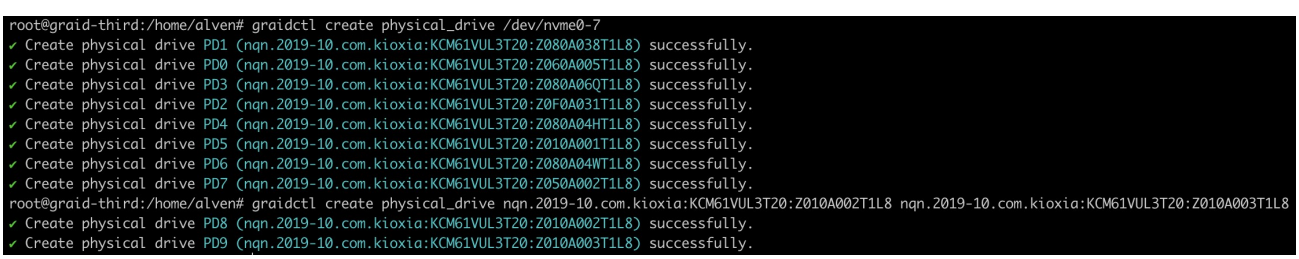
Physical Drive Management

Create physical drive

To create a physical drive, run:

```
$ sudo graidctl create physical_drive <DEVICE_PATH|NQN|WWID>
```

Output example for creating multiple physical drives at the same time with device path and NQN:



List physical drive

To list all of the physical drives, run:

```
$ sudo graidctl list physical_drive
```

Output example:

```
graid-sake:~ # graidctl list physical_drive
✔ List physical drive successfully.
```

PD ID (10)	DG ID	NQN/WWID	MODEL	CAPACITY	SLOT ID	STATE
0	N/A	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A038T1L8	KCM61VUL3T20	3.2 TB	1	UNCONFIGURED_GOOD
1	N/A	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A06QT1L8	KCM61VUL3T20	3.2 TB	2	UNCONFIGURED_GOOD
2	N/A	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04WT1L8	KCM61VUL3T20	3.2 TB	3	UNCONFIGURED_GOOD
3	N/A	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z050A002T1L8	KCM61VUL3T20	3.2 TB	4	UNCONFIGURED_GOOD
4	N/A	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A003T1L8	KCM61VUL3T20	3.2 TB	5	UNCONFIGURED_GOOD
5	N/A	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z060A005T1L8	KCM61VUL3T20	3.2 TB	6	UNCONFIGURED_GOOD
6	N/A	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z0F0A031T1L8	KCM61VUL3T20	3.2 TB	7	UNCONFIGURED_GOOD
7	N/A	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A002T1L8	KCM61VUL3T20	3.2 TB	8	UNCONFIGURED_GOOD
8	N/A	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04HT1L8	KCM61VUL3T20	3.2 TB	9	UNCONFIGURED_GOOD
9	N/A	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A001T1L8	KCM61VUL3T20	3.2 TB	10	UNCONFIGURED_GOOD

Output content:

Field	Description
SLOT ID	The slot ID of corresponding NVMe/SAS/SATA drive, this information is only for query and will be displayed when the system is provided
DG ID	The drive group ID of physical drive
PD ID	The PD ID is a unique ID provided by SpremeRAID driver when the physical drive is created. It is not related to any SSD information such as slot ID or NQN. The ID will be used for any subsequent operations
NQN/WWID	The NQN or WWID of corresponding NVMe/SAS/SATA drive
MODEL	The model number of corresponding NVMe/SAS/SATA drive
CAPACITY	The capacity of corresponding NVMe/SAS/SATA drive
STATE	The Physical drive state

Physical drive state:

State	Description
ONLINE	The physical drive has been added to a drive group and ready to work
HOTSPARE	The physical drive has been configured as hot spare drive
FAILED	The physical drive can be detected but not functioning normally

State	Description
OFFLINE	The physical drive is marked as offline
REBUILD	The physical drive is being rebuilt
MISSING	The physical drive cannot be detected
UNCONFIGURED_GOOD	The physical drive did not join any drive group
UNCONFIGURED_BAD	The physical drive did not join any drive group and not functioning normally

Delete physical drive

To delete physical drives, run:

```
$ sudo graidctl delete physical_drive <PD_ID>
```

Output example for deleting multiple physical drives at the same time:

```
root@graid-third:/home/alven# graidctl delete physical_drive 2
✖ Delete physical drive PD2 failed: rpc error: code = NotFound desc = PD2 is still using by DG1
root@graid-third:/home/alven# graidctl delete physical_drive 0-1 5 6 7
✔ Delete physical drive PD0 successfully.
✔ Delete physical drive PD1 successfully.
✔ Delete physical drive PD5 successfully.
✔ Delete physical drive PD6 successfully.
✔ Delete physical drive PD7 successfully.
```

From the output above, you can see that you cannot delete a physical drive which are already part of a drive group.

Describe physical drive

To check the detailed information of a physical drive, run:

```
$ sudo graidctl describe physical_drive <PD_ID>
```

Output example:

```
root@graid-third:/home/alven# graidctl describe physical_drive 0
✔ Describe physical drive PD0 successfully.
{
    "SlotId": 1,
    "DgId": -1,
    "PdId": 0,
    "Guid": "nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A038T1L8",
    "Model": "KCM61VUL3T20",
    "Capacity": "3.2 TB",
    "State": "UNCONFIGURED_GOOD",
    "Attrs": {
        "hotspare": "",
        "locating": "false"
    }
}
```

Locate physical drive

To locate a physical drive, run:

```
$ sudo graidctl edit physical_drive <PD_ID> locating start
```

To stop locating a physical drive, run:

```
$ sudo graidctl edit physical_drive <PD_ID> locating stop
```

Assign hot spare drive

To assign a physical drive as global hot spare, run:

```
$ graidctl edit pd <PD_ID> hotspare global
```

To assign a physical drive as hot spare of a specific drive group, run:

```
$ graidctl edit pd <PD_ID> hotspare <DG_ID>
```

You can assign a physical drive as hot spare of multiple drive groups, use comma to separate drive group IDs.

Drive Group Management

Create drive group

To create a drive group, run:

```
$ sudo graidctl create drive_group <RAID_MODE> (PD_IDs) [--background-init]
```

Output example for creating 3 drive groups each by each:

```
root@graid-third:/home/alven# graidctl create drive_group raid1 0-1
✔ Create drive group DG0 successfully.
root@graid-third:/home/alven# graidctl create drive_group raid5 2-4
✔ Create drive group DG1 successfully.
root@graid-third:/home/alven# graidctl create drive_group raid6 5-9
✔ Create drive group DG2 successfully.
```

Required parameters:

Option	Description
RAID_MODE	The RAID mode of the drive group, supports all uppercase or all lowercase(For example, RAID6 or raid6 are both correct)
PD_IDs	The ID of the physical drive that will join the drive group

Optional parameters:

Option	Description
--background-init	Force the use of traditional methods to initialize the drive group

If all physical drives in the same drive group support the deallocate dataset management command, we will use it to synchronize data or parity between physical drives when creating the drive group by default.

List drive group

To list all drive groups, run:

```
$ graidctl list drive_group
```

Output example:

```
root@graid-third:/home/alven# graidctl list drive_group
✔ List drive group successfully.
```

DG ID	MODE	VD NUM	CAPACITY	FREE	USED	STATE
0	RAID1	0	3.2 TB	3.2 TB	0 B	OPTIMAL
1	RAID5	0	6.4 TB	6.4 TB	0 B	OPTIMAL
2	RAID6	0	9.6 TB	9.6 TB	0 B	OPTIMAL

Output content:

Field	Description
DG ID	Drive group ID
MODE	Drive group RAID mode
VD NUM	How many virtual drives in drive group
CAPACITY	Total usable capacity of drive group
FREE	Unused space of drive group
USED	Used space of drive group
STATE	Drive group state

Drive group state:

State	Description
OFFLINE	The drive group cannot serve normally, it is usually caused by the damaged physical drives exceeding the tolerable number
OPTIMAL	The drive group is in optimal state
DEGRADED	The drive group is available and ready for work but the number of missing or failed physical drives has reached the upper limit
PARTIALLY_DEGRADED	The drive group is available and ready for use, but some physical drives are missing or failed
RECOVERY	The drive group is in the process of recovery
FAILED	The drive group cannot serve normally
INIT	The drive group is initializing
RESYNC	The drive group is re-synchronizing
RESCUE	The drive group is in rescue mode

Delete drive group

To delete drive groups, run:

```
$ sudo graidctl delete drive_group <DG_ID>
```

```
root@graid-third:/home/alven# graidctl delete drive_group 1
✖ Delete drive group DG1 failed: rpc error: code = FailedPrecondition desc = DG1 still has 1VD(s)
root@graid-third:/home/alven# graidctl delete drive_group 0 2
✔ Delete drive group DG0 successfully.
✔ Delete drive group DG2 successfully.
```

You cannot delete a drive group that contains a virtual drive. In this example the attempted deletion of drive group 1 failed because it still has one virtual drive on it, however the deletion of drive groups 0 and 2 was successful.

Describe drive group

To check the detailed information of a drive group, run:

```
$ sudo graidctl describe drive_group <DG_ID>
```

Output example:

```
root@graid-third:/home/alven# graidctl describe drive_group 2
✔ Describe drive group DG2 successfully.
{
    "DgId": 2,
    "Mode": "RAID6",
    "Capacity": "9.6 TB",
    "Free": "9.6 TB",
    "Used": "0 B",
    "State": "OPTIMAL",
    "PdIds": [
        5,
        6,
        7,
        8,
        9
    ],
    "VdNum": 0,
    "Attrs": {
        "rebuild_speed": "low"
    }
}
```

Adjust rebuild speed of drive group

To adjust rebuild speed of a drive group, run:

```
$ sudo graidctl edit drive_group <DG_ID> rebuild_speed {low|normal|high}
```

Locate all physical drives in the drive group

To locate all physical drives in drive group, run:

```
$ sudo graidctl edit drive_group <DG_ID> locating start
```

To stop locating all physical drives in drive group, run:

```
$ sudo graidctl edit drive_group <DG_ID> locating stop
```

Degrade and recovery

- If there are multiple drive groups that require recovery at the same time, only one drive group will perform recovery at a time.
- If there are multiple physical drives in the same drive group that need to be rebuilt, these physical drives will be rebuilt at the same time.

Rescue mode

When the drive group that is being initialized has any physical drive damage, or the drive group is recovering but encounters an abnormal system crash, it will affect the integrity of the data in the drive group. In this case, we will force the drive group to go offline to avoid more data damage caused by new data writing. If you need to read the data of the drive group, you can force the drive group to go online through rescue mode. The drive group in rescue mode is read only, and the rescue mode cannot be disabled, to enter the rescue mode, run:

```
$ sudo graidctl edit drive_group <DG_ID> rescue_mode on
```

Virtual Drive Management

Create virtual drive

To create a virtual drive, run:

```
$ sudo graidctl create virtual_drive <DG_ID> [<VD_SIZE>]
```

Output example:

```
root@graid-third:/home/alven# graidctl create virtual_drive 0
✔ Create virtual drive VD0 in DG0 successfully.
root@graid-third:/home/alven# graidctl create virtual_drive 1 100G
✔ Create virtual drive VD0 in DG1 successfully.
root@graid-third:/home/alven# graidctl create virtual_drive 2 1T
✔ Create virtual drive VD0 in DG2 successfully.
```

List virtual drive

To list virtual drives, run:

```
$ graidctl list virtual_drive [--dg-id=<DG_ID>] [--vd-id=<VD_ID>]
```

Output example:

```
root@graid-third:/home/alven# graidctl list virtual_drive
✔ List virtual drive successfully.
```

DG ID (4)	VD ID	SIZE	DEVICE PATH	STATE
0	0	3.2 TB	/dev/gvd0	OPTIMAL
1	0	100 GB	/dev/gvd1	OPTIMAL
2	0	1.0 TB	/dev/gvd2	OPTIMAL
2	1	1.0 TB	/dev/gvd3	OPTIMAL

Output content:

Field	Description
DG ID	ID of belonging drive group
VD ID	Virtual drive ID
SIZE	Virtual drive usable size
DEVICE PATH	Virtual drive device path
STATE	Virtual drive state, would be the same with belonging drive group

Delete virtual drive

To delete virtual drives, run:

```
$ sudo graidctl delete virtual_drive <DG_ID> <VD_ID> [--force]
```

Output example:

```
root@graid-third:/home/alven# graidctl delete virtual_drive 0 0
✔ Delete virtual drive VD0 from DG0 successfully.
root@graid-third:/home/alven# graidctl delete virtual_drive 2 0-1
✔ Delete virtual drive VD1 from DG2 successfully.
✔ Delete virtual drive VD0 from DG2 successfully.
```

As you can see, you cannot delete a virtual drive which is being used by the application unless the force flag is added.

Frequently Used Tasks

Create a RAID-5 Virtual Drive with 5 NVMe SSDs

1. Create a physical drive

```
$ sudo graidctl create physical_drive /dev/nvme0-4
✔ Create physical drive PD0 successfully.
✔ Create physical drive PD1 successfully.
✔ Create physical drive PD2 successfully.
✔ Create physical drive PD3 successfully.
✔ Create physical drive PD4 successfully.
```

2. Create a drive group

```
$ sudo graidctl create drive_group raid5 0-4
✔ Create drive group DG0 successfully.
```

3. Create a virtual drive

```
$ sudo graidctl create virtual_drive 0
✔ Create virtual drive VD0 successfully.
```

4. Check the device path of new virtual drive

```
$ sudo graidctl list virtual_drive --dg_id=0
```

Replace a nearly worn-out SSD

1. First mark the physical drive as bad with the following command:

```
$ sudo graidctl edit pd <PD_ID> marker bad
```

2. Replace the old NVMe SSD with new one, the old physical drive state should now be **MISSING**
3. Check the NQN of the new SSD

```
$ sudo graidctl list nvme_drive
```

4. Create a new physical drive on the new SSD

```
$ sudo graidctl create physical_drive <NEW_SSD_NQN>
```

5. Replace the old physical drive with new physical drive

```
$ sudo graidctl replace physical_drive <OLD_PD_ID> <NEW_PD_ID>
```

Appendix: Software Upgrade

1. Stop applications running on the virtual drive
2. Stop the management service

```
$ sudo systemctl stop graid
```

3. Uninstall the package

```
# In Centos, RHEL, openSUSE, SLES
$ sudo rpm -e graid-sr
# In Ubuntu
$ sudo dpkg -r graid-sr
```

4. Install the new package

```
# In Centos, RHEL, openSUSE, SLES
$ sudo rpm -Uvh <graid-sr-x.x.x-x.gxxxxxxx.000.x86_64.rpm>
# In Ubuntu
$ sudo dpkg -i <graid-sr-x.x.x-x.gxxxxxxx.000.x86_64.deb>
```

5. Start the management service

```
$ sudo systemctl enable graid
$ sudo systemctl start graid
```

Appendix:
Manually migrate RAID configuration between hosts

1. Backup config file **/etc/graid.conf** periodically from original host. For this you can use cp or scp to move it to another system.
2. Setup target host and make sure there is no graid service running. It should only be running in a case where your target host already has a SupremeRAID™ adapter installed and running. In this case we want to stop the service andrestart it with the **graid.conf** file from the originating system.
3. Move all SSDs from the original host to the new host.
4. Copy backed up config to the new host with the same path.
5. If the original card also moved to the new host, you can start the graid service directly

```
$ sudo systemctl start graid
```

Otherwise, you have to apply the new license of the new GRAID SupremeRAID™ card on the new host

```
$ sudo graidctl apply license <LICENSE_KEY>
```

Appendix: Basic Trouble Shooting

Sequential Read Performance is not as expected on new drive group

Unlike SAS/SATA hard drives, many NVMe SSDs support the deallocate dataset management command. If you use this command, you can reset all data in the NVMe SSD immediately, which means you don't have to synchronize data between physical drives when the user is creating a drive group, this is how fast initialization works.

But we also found that for some SSDs, after the deallocate dataset management command issued, the Performance is not as expected when reading unwritten sectors. This behavior will also impact the performance of the new drive group, but it should not affect the real application scenarios. After all, there will be no applications that need to read sectors where data has not been written.

However, if it is purely to test the performance of the product, you can solve this phenomenon by writing the entire virtual drive sequentially with large block size.

Kernel log shows "failed to set APST feature (-19)" when creating physical drives

For some NVMe SSD models, you may find the message "failed to set APST feature (-19)" in the Kernel log when creating the physical drive. This is a normal phenomenon and will not affect the RAID function at all.

The reason that causing this message is when SupremeRAID™ creating the physical drive, in order to obtain complete control of the SSD, the SSD must be unbound from the operating system. During the unbinding process, if the APST feature is turned on, the NVMe driver will still try to set the APST state to SSD, but the APST state cannot be set normally at this moment and cause this error message generated.