GRAID SupremeRAID™
SR-1000
User Guide

Version 1.0.6

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 Driver 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. Minimum system requirements:
    - CPU: 2 GHz or faster with at least 8 cores
    - RAM: 16 GB
    - An extra PCIe Gen3 or Gen4 x16 slot
2. Install the GRAID SupremeRAID SR-1000 card into a PCIe x16 slot.
3. Make sure IOMMU function is disabled in the system BIOS.
4. Download the GRAID SupremeRAID software package from the GRAID technology website or contact your GRAID partner.

Installing on Linux via Pre-installer

The GRAID pre-installer is an executable file containing the required dependencies and a setup script which will install the NVIDIA driver. The script is designed to minimize the effort required to prepare the environment before installing the GRAID SupremeRAID driver. You can follow the following steps to prepare the environment and install the GRAID SupremeRAID driver via pre-installer in every Linux distro.

CentOS and RHEL

1. These steps should be performed on a terminal that does not run the GUI console
2. Download the latest version of the pre-installer and make it executable

```
$ wget https://download.graidtech.com/driver/pre-install/graid-sr-pre-installer-1.0.0-31.run
$ chmod +x graid-sr-pre-installer-1.0.0-31.run
```

3. Execute the pre-installer and follow the instructions to finish the pre-installation process as shown below in the output example:

```
[root@graid ~]# sudo ./graid-sr-pre-installer-1.0.0-31.run
Extracting installer files, please wait a few seconds ...
Extracting installer files done.

Starting installer ...
This pre-installer is for CentOS
Running service check
Service check succeeded.

Running system check
System check succeeded.

Running install packages and kernel setting.
Generating grub configuration file ...
Adding boot menu entry for EFI firmware configuration
done
Generating new initramfs...
Generated new initramfs.
Install packages and kernel setting succeeded.

chvt: ioctl VT_ACTIVATE: No such device or address
Disabled Xorg instance.
Nouveau module has been loaded, pre-installer will unload nouveau for NVIDIA driver install.
Unload nouveau module successfully.
Running install NVIDIA Driver. (This step will take a while.)
Mon Oct  4 14:30:35 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.63.01    Driver Version: 470.63.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| 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 T1000        Off  | 00000000:13:00.0 Off |                  N/A |
| 37%   39C    P0    N/A /  50W |      0MiB /  3911MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
Install NVIDIA Driver succeeded.

This pre-installer will reboot the system for apply previous setting!
Do you want to continue? [Y/n]
```

4. After running the pre-installation script the system will need to be rebooted. When prompted, type Y to reboot the system when prompted
5. Install the SupremeRAID driver

```
$ sudo rpm -Uvh <graid-sr-x.x.x-xxx.git_xxxxxxx.000.el8.x86_64.rpm>
```

6. Apply the SupremeRAID license key

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

Ubuntu

1. These steps should be performed on a terminal that does not run the GUI console
2. Download the latest version of the pre-installer and make it executable

```
$ wget https://download.graidtech.com/driver/pre-install/graid-sr-pre-installer-1.0.0-31.run
$ chmod +x graid-sr-pre-installer-1.0.0-31.run
```

3. Execute the pre-installer and follow the instructions to finish the pre-installation process as shown below in the output example:

```
root@graid:~$ sudo ./graid-sr-pre-installer-1.0.0-31.run
Reading package lists... Done
Building dependency tree
Reading state information... Done
tar is already the newest version (1.30+dfsg-7ubuntu0.20.04.1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Extracting installer files, please wait a few seconds ...
Extracting installer files done.

Starting installer ...
This pre-installer is for Ubuntu
Running service check
Service check succeeded.

Running system check
System check succeeded.

Running install packages and kernel setting.
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.4.0-81-generic
Found initrd image: /boot/initrd.img-5.4.0-81-generic
Adding boot menu entry for UEFI Firmware Settings
done
Generating new initramfs...
update-initramfs: Generating /boot/initrd.img-5.4.0-81-generic
Generated new initramfs.
Install packages and kernel setting succeeded.

Nouveau module has been loaded, pre-installer will unload nouveau for NVIDIA driver install.
Unload nouveau module successfully.
Running install NVIDIA Driver. (This step will take a while.)
Sat Oct  2 17:23:26 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.63.01    Driver Version: 470.63.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| 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 T1000        Off  | 00000000:0B:00.0 Off |                  N/A |
| 35%   39C    P0    N/A /  50W |      0MiB /  3911MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
Install NVIDIA Driver succeeded.

This pre-installer will reboot the system for apply previous setting!
Do you want to continue? [Y/n]
```

4. After running the pre-installation script the system will need to be rebooted. When prompted type Y to reboot the system when prompted

5. Install the SupremeRAID driver

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

6. Apply the SupremeRAID license key

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

openSUSE and SLES

1. These steps should be performed on a terminal that does not run the GUI console
2. Download the latest version of the pre-installer and make it executable

```
$ wget https://download.graidtech.com/driver/pre-install/graid-sr-pre-installer-1.0.0-31.run
$ chmod +x graid-sr-pre-installer-1.0.0-31.run
```

3. Execute the pre-installer and follow the instructions to finish the pre-installation process, the output example as shown below in the output example:

```
root@graid:~> sudo ./graid-sr-pre-installer-1.0.0-31.run
Loading repository data...
Reading installed packages...
'tar' is already installed.
No update candidate for 'tar-1.30-3.3.2.x86_64'. The highest available version is already installed.
Resolving package dependencies...

Nothing to do.
Extracting installer files, please wait a few seconds ...
Extracting installer files done.

Starting installer ...
This pre-installer is for SUSE
Running service check
Service check succeeded.

Running system check
System check succeeded.

Running install packages and kernel setting.
Generating grub configuration file ...
Found theme: /boot/grub2/themes/SLE/theme.txt
Found linux image: /boot/vmlinuz-5.3.18-22-default
Found initrd image: /boot/initrd-5.3.18-22-default
done
Generating new initramfs...
Generated new initramfs.
Install packages and kernel setting succeeded.

chvt: ioctl VT_ACTIVATE: No such device or address
Disabled Xorg instance.
Nouveau module has been loaded, pre-installer will unload nouveau for NVIDIA driver install.
Failed to stop gdm.service: Unit gdm.service not loaded.
Unload nouveau module successfully.
Running install NVIDIA Driver. (This step will take a while.)
Sat Oct  2 15:34:08 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.63.01    Driver Version: 470.63.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| 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 T1000        Off  | 00000000:1B:00.0 Off |                  N/A |
| 37%   40C    P0    N/A /  50W |      0MiB /  3911MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
Install NVIDIA Driver succeeded.

This pre-installer will reboot the system for apply previous setting!
Do you want to continue? [Y/n]
```

4. After running the pre-installation script the system will need to be rebooted. When prompted type Y to reboot the system when prompted
5. Install the SupremeRAID driver

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

6. Apply the SupremeRAID license key

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

Installing on Linux manually

In addition to preparing the environment through pre-installer, you can also manually install the dependencies required by the driver.

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-$(uname -r) dkms ipmitooltar
```

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

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

Append ***"iommu=pt"*** to **GRUB_CMDLINE_LINUX** then make grub2 config, if you are using 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. Append **"blacklist nouveau"** and **"options nouveau modeset=0"** to the end of **/etc/modprobe.d/graid-blacklist.conf** file to disable the nouveau driver and update the initramfs.

```
$ dracut -f
```

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

```
[root@graid ~]# cat /proc/cmdline
BOOT_IMAGE=(hd3,gpt8)/vmlinuz-4.18.0-305.17.1.el8_4.x86_64 root=UUID=ba33c54d-74c9-409d-ae05-db27cacd68b3 ro
crashkernel=auto resume=UUID=ae5b3808-b657-4593-a598-c5cbc5a87105 rhgb quiet iommu=pt rd.driver.blacklist=nouveau
```

5. Install NVIDIA driver.

```
$ wget https://us.download.nvidia.com/XFree86/Linux-x86_64/470.63.01/NVIDIA-Linux-x86_64-470.63.01.run
$ chmod +x NVIDIA-Linux-x86_64-470.63.01.run
$ sudo ./NVIDIA-Linux-x86_64-470.63.01.run -s --no-systemd --no-opengl-files --no-nvidia-modprobe --dkms
```

If you have the nouveau driver, step 2 disables that or you can let the NVIDIA driver installer attempt to disable it (follow the bellow instructions), once complete you will have to reboot and install the NVIDIA driver again.

```
$ sudo ./NVIDIA-Linux-x86_64-470.63.01.run -s --disable-nouveau
$ reboot
```

6. Confirm the NVIDIA GPU is working via the command ***nvidia-smi***. Below is an example of a successful installation output:

```
[root@graid ~]# nvidia-smi
Tue Sep 21 21:27:37 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.63.01    Driver Version: 470.63.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| 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 T1000        Off  | 00000000:81:00.0 Off |                  N/A |
| 34%   38C    P0    N/A /  50W |      0MiB /  3911MiB |      0%   E. Process |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

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

7. Install the graid software rpm package.

```
$ sudo rpm -Uvh <graid-sr-x.x.x-xxx.git_xxxxxxx.000.el8.x86_64.rpm>
```

8. 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) dkms ipmitool initramfs-tools tar
```

2. Add kernel option. In this step we will block the nouveau driver from loading if installed and disable iommu if you have 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/graid-blacklist.conf** file to disable the nouveau driver and update the initramfs.

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

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

```
root@graid:~/# 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
```

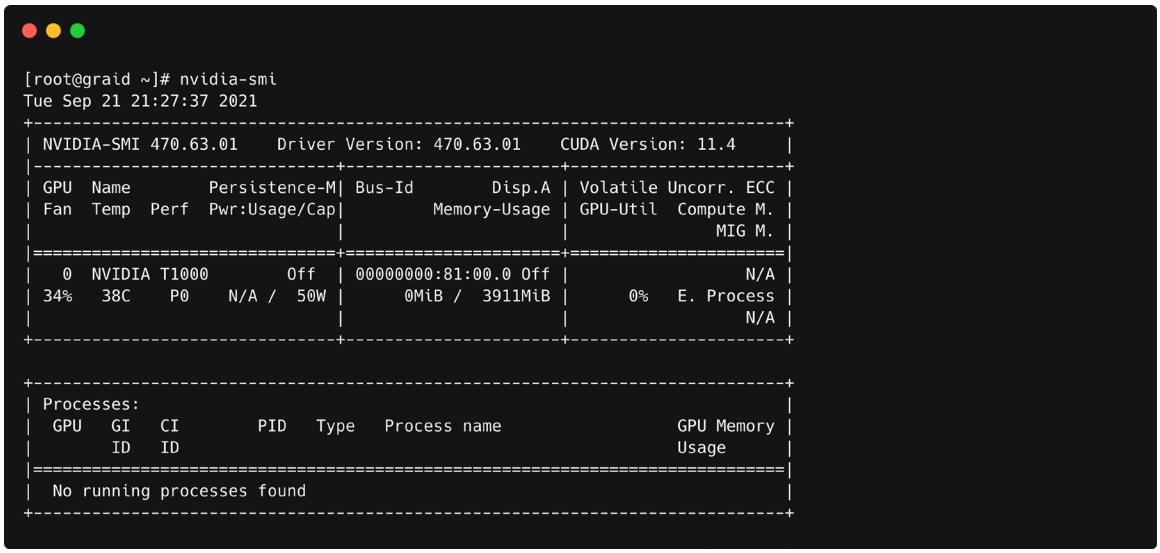
5. Install NVIDIA driver.

```
$ wget https://us.download.nvidia.com/XFree86/Linux-x86_64/470.63.01/NVIDIA-Linux-x86_64-470.63.01.run
$ chmod +x NVIDIA-Linux-x86_64-470.63.01.run
$ sudo ./NVIDIA-Linux-x86_64-470.63.01.run -s --no-systemd --no-opengl-files --no-nvidia-modprobe --dkms
```

If you have the nouveau driver, step 2 disables that or you can let the NVIDIA driver installer attempt to disable it (follow the bellow instructions), once complete you will have to reboot and install the NVIDIA driver again.

```
$ sudo ./NVIDIA-Linux-x86_64-470.63.01.run -s --disable-nouveau
$ reboot
```

6. Confirm the NVIDIA GPU is working via the command ***nvidia-smi***. Below is an example of a successful installation output:



```
[root@graid ~]# nvidia-smi
Tue Sep 21 21:27:37 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.63.01    Driver Version: 470.63.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| 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 T1000        Off  | 00000000:81:00.0 Off |                  N/A |
| 34%   38C    P0    N/A /  50W |      0MiB /  3911MiB |      0%   E. Process |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

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

7. Install the graid software debian package.

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

8. 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 tar
```

3. Add kernel option. In this step we will block the nouveau driver from loading if installed and disable iommu if you have 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/graid-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 and update the initrd.

```
$ sudo mkinitrd
```

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

```
root@graid:~ # cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.3.18-59.5-default root=UUID=7560fe42-0275-4618-b8a0-0785765610c9
modprobe.blacklist=nouveau iommu=pt splash=silent quiet mitigations=auto
```

7. Install NVIDIA driver.

```
$ wget https://us.download.nvidia.com/XFree86/Linux-x86_64/470.63.01/NVIDIA-Linux-x86_64-470.63.01.run
$ chmod +x NVIDIA-Linux-x86_64-470.63.01.run
$ sudo ./NVIDIA-Linux-x86_64-470.63.01.run -s --no-systemd --no-opengl-files --no-nvidia-modprobe --dkms
```

If you have the nouveau driver, step 2 disables that or you can let the NVIDIA driver installer attempt to disable it (follow the bellow instructions), once complete you will have to reboot and install the NVIDIA driver again.

```
$ sudo ./NVIDIA-Linux-x86_64-470.63.01.run -s --disable-nouveau
$ reboot
```

8. Confirm the NVIDIA GPU is working via the command ***nvidia-smi***. Below is an example of a successful installation output:

```
[root@graid ~]# nvidia-smi
Tue Sep 21 21:27:37 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.63.01    Driver Version: 470.63.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| 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 T1000        Off  | 00000000:81:00.0 Off |                  N/A |
| 34%   38C    P0    N/A /  50W |      0MiB /  3911MiB |      0%   E. Process |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

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

9. Install the graid software rpm package.

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

10. 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 tar
```

3. Add kernel option. In this step we will block the nouveau driver from loading if installed and disable iommu if you have 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/graid-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. For example:

```
root@graid:~ # cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.3.18-59.5-default root=UUID=7560fe42-0275-4618-b8a0-0785765610c9
modprobe.blacklist=nouveau iommu=pt splash=silent quiet mitigations=auto
```

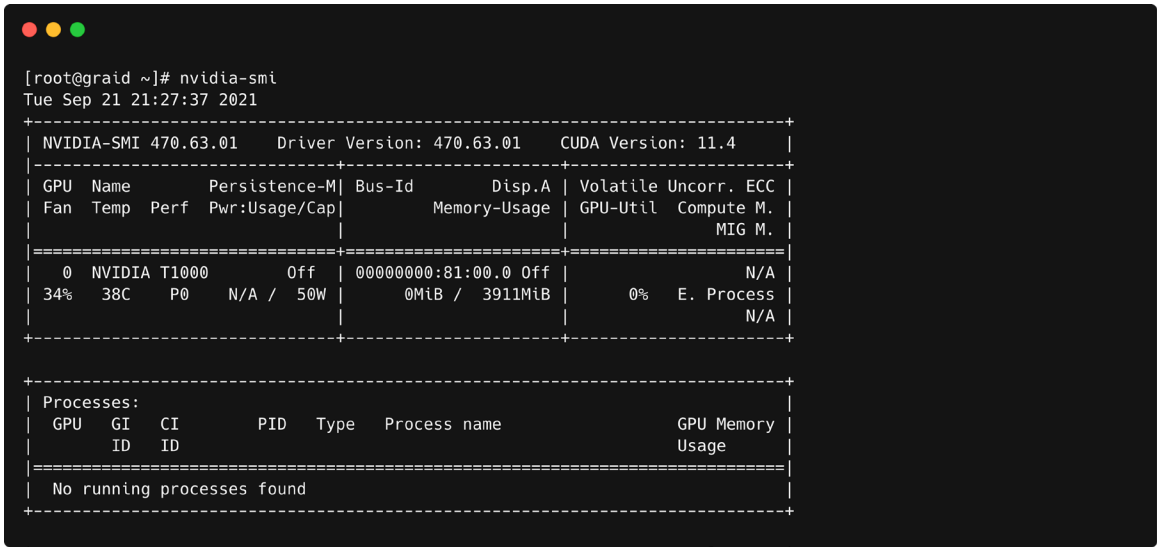
8. Install NVIDIA driver.

```
$ wget https://us.download.nvidia.com/XFree86/Linux-x86_64/470.63.01/NVIDIA-Linux-x86_64-470.63.01.run
$ chmod +x NVIDIA-Linux-x86_64-470.63.01.run
$ sudo ./NVIDIA-Linux-x86_64-470.63.01.run -s --no-systemd --no-opengl-files --no-nvidia-modprobe --dkms
```

If you have the nouveau driver, step 2 disables that or you can let the NVIDIA driver installer attempt to disable it (follow the bellow instructions), once complete you will have to reboot and install the NVIDIA driver again.

```
$ sudo ./NVIDIA-Linux-x86_64-470.63.01.run -s --disable-nouveau
$ reboot
```

9. Confirm the NVIDIA GPU is working via the command ***nvidia-smi***. Below is an example of a successful installation output:



10. Install the graid software rpm package.

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

11. Apply the license to activate the GRAID service.

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

Management

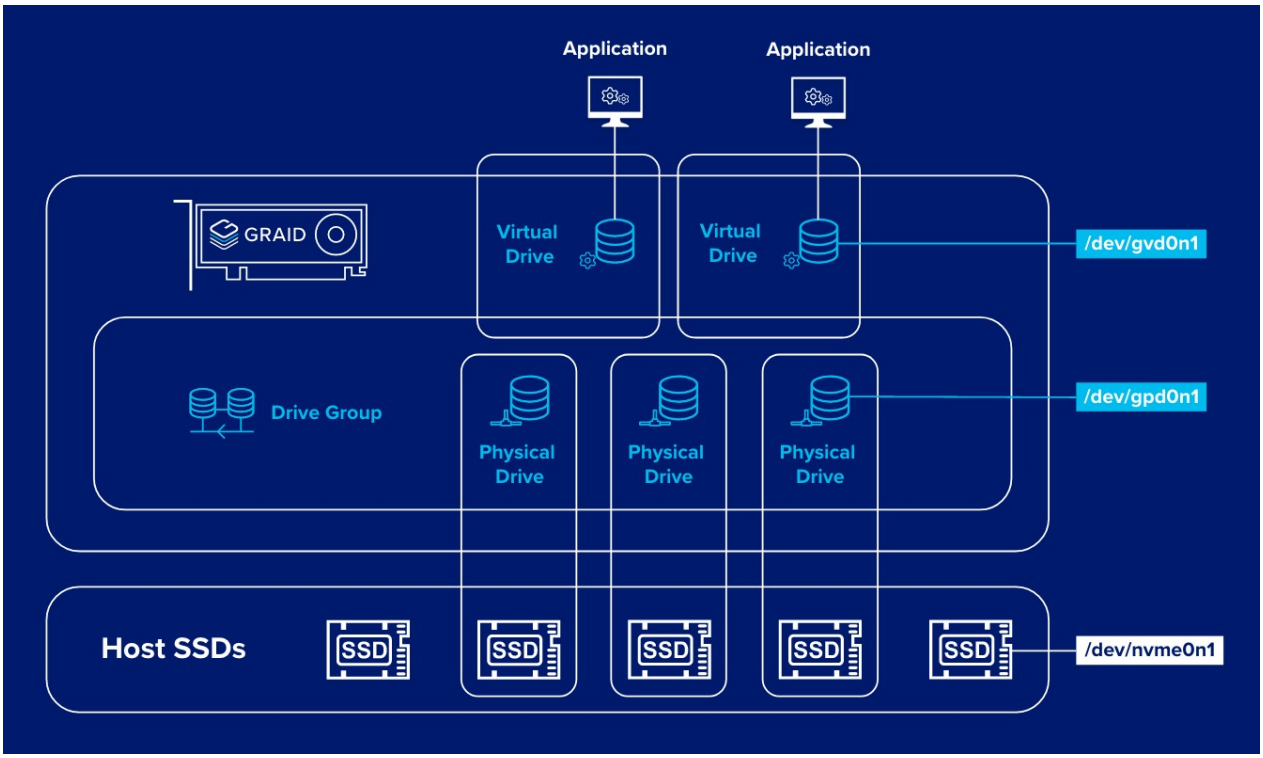
Overview of GRAID SupremeRAID Software Module

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 to control the driver.
3. **graid.ko** - The driver kernel module.

RAID components

There are three major RAID logical components in SupremeRAID, **Physical Drive(PD)**, **Drive Group(DG)**, and **Virtual Drive(VD)**.



Physical Drive (PD)

Since NVMe drives are not directly attached to the SupremeRAID controller, the user must tell the controller which SSDs can be managed. Once an SSD has been created as a physical drive, the SupremeRAID driver will unbond the SSD from the operating system, which means the device node (/dev/nvmeX) disappear and no longer be accessible. At the same time, a corresponding device node will be created (/dev/gpdX) by the SupremeRAID driver. You can check the SSD information such as SSD model or S.M.A.R.T. logs via this device node. If you want to control and access the SSD by /dev/nvmeXn1, you must delete the corresponding physical drive first.

Currently, SupremeRAID SR-1000 supports up to a total of **32** physical drives, regardless of whether the physical drives are created from a native NVMe SSD, a drive connected through NVMe-oF, or a SAS/SATA Disk.

Drive Group (DG)

The main component of RAID logic, essentially it is a RAID group. Once the drive group has been created, the SupremeRAID driver will start to initialize the physical drives with the corresponding RAID mode to make sure the data and the parity are synchronized. There are 2 types of the initialization process, if all physical drives in the drive group (DG) support the deallocate dataset management command, the SupremeRAID driver will perform the fast initialization as default, which means the drive group will turn to the optimal state immediately. Otherwise, the SupremeRAID driver will perform the background initialization. You still can create the virtual drive and access the virtual drive during the background initialization, but the performance will be slightly affected by the initialization traffic.

Currently, SupremeRAID SR-1000 supports up to a total of **4** drive groups, and maxima **32** physical drives in one drive group.

Virtual Drive (VD)

The virtual drive is equivalent to the RAID volume, you can create multiple virtual drives in the same drive group for multiple applications. You'll see the corresponding device node (/dev/nvmeXn1) on the operating system once you create a virtual drive, and you can make the file system or running application on this device node directly. Currently, the SupremeRAID driver supports maxima **8** virtual drives in each drive group.

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 ~]# graidctl apply license 34B45F67-5694YXQB-I4LHTCAT-VYW6CXWJ
✖ Invalid license: 34B45F67-5694YXQB-I4LHTCAT-VYW6CXWJ
[root@graid ~]# graidctl apply license 34B45F67-5694YXQB-I4LHTCAT-VYW6CXWI
✔ Apply license 34B45F67-5694YXQB-I4LHTCAT-VYW6CXWI successfully.
```

Check license information

To obtain the license information, run:

```
$ graidctl describe license
```

Output example:

```
[root@graid ~]# graidctl describe license
✔ Describe license successfully.
{
    "LicenseState": "APPLIED",
    "LicenseKey": "34B45F67-5694YXQB-I4LHTCAT-VYW6CXWI",
    "Feature": {
        "PdNum": 32,
        "Raid5": 1,
        "Raid6": 1,
        "Nvmeof": 1
    },
    "ExpDays": "Unlimited"
}
```

Output content:

Field	Description
License State	The license state
License Key	The applied license key
Feature	The feature set of license key
ExpDays	The expiration date of 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

Remote NVMe-oF Target Management

You can create the physical drives from NVMe-oF device, before that, you have to connect NVMe-oF target.

Connect Remote NVMe-oF target

To connect a remote NVMe-oF target, run:

```
$ graidctl create nvmeof_target <tcp|rdma|fc> <addr> <address family> <service id>
```

Output example:

```
[root@graid ~]# graidctl create nvmeof_target rdma 192.168.2.10 ipv4 4420
✔ Create nvmeof target NVMe-oF target 0: {rdma,192.168.2.10,ipv4,4420} successfully.
```

List Connected Remote NVMe-oF target

To list all of the NVMe-oF targets, run:

```
$ graidctl list nvmeof_target
```

Output example:

```
[root@graid ~]# graidctl list nvmeof_target
✔ List nvmeof target successfully.
|────|──────|────────────────|──────────────────|─────────────|
| ID | TYPE | ADDRESS        | ADDRESS FAMILY   | SERVICE ID  |
|────|──────|────────────────|──────────────────|─────────────|
|  0 | rdma | 192.168.2.10   | ipv4             | 4420        |
|────|──────|────────────────|──────────────────|─────────────|
```

Disconnect Remote NVMe-oF target

To disconnect a NVMe-oF target, run:

```
$ graidctl delete nvmeof_target <target id>
```

Output example:

```
[root@graid ~]# graidctl delete nvmeof_target 0
✔ Delete nvmeof target Target 0 successfully.
```

You must make sure that there are no physical drives created from the target you want to delete, or you cannot delete the target.

Host Drive Information

List NVMe drives

To list all NVMe drives that are attached directly or from NVMe-oF target and can be created as physical drives, run:

```
$ graidctl list nvme_drive
```

Output example:

```
[root@graid ~]# graidctl list nvme_drive
✔ List nvme drive successfully.
|-----------------|-----------------------------------------------|---------------|----------|
| DEVICE PATH (10)| NQN                                           | MODEL         | CAPACITY |
|-----------------|-----------------------------------------------|---------------|----------|
| /dev/nvme0      | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A064T1L8 | KCM61VUL3T20 | 3.2 TB   |
| /dev/nvme1      | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z050A002T1L8 | KCM61VUL3T20 | 3.2 TB   |
| /dev/nvme2      | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A05KT1L8 | KCM61VUL3T20 | 3.2 TB   |
| /dev/nvme3      | nqn.2019-10.com.kioxia:KCM61VUL3T20:X0N0A015T1L8 | KCM61VUL3T20 | 3.2 TB   |
| /dev/nvme4      | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A06QT1L8 | KCM61VUL3T20 | 3.2 TB   |
| /dev/nvme5      | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z0G0A001T1L8 | KCM61VUL3T20 | 3.2 TB   |
| /dev/nvme6      | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z060A006T1L8 | KCM61VUL3T20 | 3.2 TB   |
| /dev/nvme7      | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04WT1L8 | KCM61VUL3T20 | 3.2 TB   |
| /dev/nvme8      | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A003T1L8 | 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:

```
[root@graid ~]# graidctl list scsi_drive
✔ List scsi drive successfully.
|-------------|-----------------------------------------------|-----------------|----------|
| DEVICE PATH | WWID                                          | MODEL           | CAPACITY |
|-------------|-----------------------------------------------|-----------------|----------|
| /dev/sda    | t10.ATA INTEL SSDSC2KB240G7 BTYS83010GKS240AGN | INTEL SSDSC2KB24 | 240 GB   |
| /dev/sdb    | t10.ATA INTEL SSDSC2KB240G8 BTYF052107VH240AGN | INTEL SSDSC2KB24 | 240 GB   |
|-------------|-----------------------------------------------|-----------------|----------|
```

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:

```
[root@graid ~]# graidctl create physical_drive /dev/nvme0-3
✔ Create physical drive PD0 (/dev/nvme0: nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A064T1L8) successfully.
✔ Create physical drive PD1 (/dev/nvme1: nqn.2019-10.com.kioxia:KCM61VUL3T20:Z050A002T1L8) successfully.
✔ Create physical drive PD2 (/dev/nvme2: nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A05KT1L8) successfully.
✔ Create physical drive PD3 (/dev/nvme3: nqn.2019-10.com.kioxia:KCM61VUL3T20:X0N0A015T1L8) successfully.
[root@graid ~]# graidctl create physical_drive nqn.2019-10.com.kioxia:KCM61VUL3T20:X0X0A01ET1L8 \
> nqn.2019-10.com.kioxia:KCM61VUL3T20:Z0F0A031T1L8
✔ Create physical drive PD8 (nqn.2019-10.com.kioxia:KCM61VUL3T20:X0X0A01ET1L8) successfully.
✔ Create physical drive PD9 (nqn.2019-10.com.kioxia:KCM61VUL3T20:Z0F0A031T1L8) successfully.
```

List physical drive

To list all of the physical drives, run:

```
$ sudo graidctl list physical_drive
```

Output example:

```
[root@graid ~]# 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   │ 0       │ UNCONFIGURED_GOOD │
│ 1           │ N/A    │ nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A06QT1L8 │ KCM61VUL3T20 │ 3.2 TB   │ 1       │ UNCONFIGURED_GOOD │
│ 2           │ N/A    │ nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04WT1L8 │ KCM61VUL3T20 │ 3.2 TB   │ 2       │ UNCONFIGURED_GOOD │
│ 3           │ N/A    │ nqn.2019-10.com.kioxia:KCM61VUL3T20:Z050A002T1L8 │ KCM61VUL3T20 │ 3.2 TB   │ 3       │ UNCONFIGURED_GOOD │
│ 4           │ N/A    │ nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A003T1L8 │ KCM61VUL3T20 │ 3.2 TB   │ 4       │ UNCONFIGURED_GOOD │
│ 5           │ N/A    │ nqn.2019-10.com.kioxia:KCM61VUL3T20:Z060A005T1L8 │ KCM61VUL3T20 │ 3.2 TB   │ 5       │ UNCONFIGURED_GOOD │
│ 6           │ N/A    │ nqn.2019-10.com.kioxia:KCM61VUL3T20:Z0F0A031T1L8 │ KCM61VUL3T20 │ 3.2 TB   │ 6       │ UNCONFIGURED_GOOD │
│ 7           │ N/A    │ nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A002T1L8 │ KCM61VUL3T20 │ 3.2 TB   │ 7       │ UNCONFIGURED_GOOD │
│ 8           │ N/A    │ nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04HT1L8 │ KCM61VUL3T20 │ 3.2 TB   │ 8       │ UNCONFIGURED_GOOD │
│ 9           │ N/A    │ nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A001T1L8 │ KCM61VUL3T20 │ 3.2 TB   │ 9       │ UNCONFIGURED_GOOD │
└─────────────┴────────┴──────────────────────────────────────────────────┴──────────────┴──────────┴─────────┴──────────────────┘
```

Output content:

Field	Description
SLOT ID	The slot ID of corresponding NVMe/SAS/SATA drive, please be noticed that the PD ID is not related to SLOT ID, and you must control the physical drives via PD ID all the time
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

Field	Description
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
OFFLINE	The physical drive is marked as offline
REBUILD	The physical drive is being rebuilt
MISSING	The physical drive cannot be detected
INCONSISTENT	The data in the physical drive is inconsistent, it generally occurrs when the physical drive is in REBUILD state and the system encounters abnormal crash
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 ~]# graidctl delete physical_drive 2
✖ Delete physical drive PD2 failed: rpc error: code = NotFound desc = PD2 is still using by DG0
[root@graid ~]# graidctl delete physical_drive 0 1 5-7
✔ 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 ~]# graidctl describe physical_drive 0
✔ Describe physical drive PD0 successfully.
{
    "SlotId": 0,
    "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
```

Mark physical drive online and offline

To mark a physical drive as offline or offline, run:

```
$ graidctl edit pd <PD_ID> marker <offline|online>
```

Once you marked a physical drive as offline, although you mark it online again immediately, the physical drive will still turn to REBUILD state.

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 ~]# graidctl create drive_group raid1 0-1
✔ Create drive group DG0 successfully.
[root@graid ~]# graidctl create drive_group raid5 2-4
✔ Create drive group DG1 successfully.
[root@graid ~]# 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 ~]# 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

State	Description
RESYNC	The drive group is re-synchronizing, this usually happens if the system encounters an abnormal crash, you should not replace the physical drive in this state until the re-synchronization process is complete
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 ~]# graidctl delete drive_group 1
✖ Delete drive group DG1 failed: rpc error: code = FailedPrecondition desc = DG1 still has 1VD(s)
[root@graid ~]# 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 ~]# graidctl describe drive_group 1
✔ Describe drive group DG1 successfully.
{
    "DgId": 1,
    "Mode": "RAID5",
    "Capacity": "6.4 TB",
    "Free": "0 B",
    "Used": "6.4 TB",
    "State": "OPTIMAL",
    "PdIds": [
        2,
        3,
        4
    ],
    "VdNum": 1,
    "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 ~]# graidctl create virtual_drive 0
✔ Create virtual drive VD0 in DG0 successfully.
[root@graid ~]# graidctl create virtual_drive 1 100G
✔ Create virtual drive VD0 in DG1 successfully.
[root@graid ~]# 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 ~]# graidctl list virtual_drive
✔ List virtual drive successfully.
|─────────────|───────|────────|──────────────|──────────|
| VD ID (4)   | DG ID | SIZE   | DEVICE PATH  | STATE    |
|─────────────|───────|────────|──────────────|──────────|
|           0 |     0 | 3.2 TB | /dev/gvd0n1  | OPTIMAL  |
|           0 |     1 | 100 GB | /dev/gvd1n1  | OPTIMAL  |
|           0 |     2 | 1.0 TB | /dev/gvd2n1  | OPTIMAL  |
|           1 |     2 | 1.0 TB | /dev/gvd3n1  | 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
NQN	Virtual drive NQN
STATE	Virtual drive state, would be the same with belonging drive group
EXPORTED	Virtual drive is exported via NVMe-oF/iSCSI or not

Delete virtual drive

To delete virtual drives, run:

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

Output example:

```
[root@graid ~]# graidctl delete virtual_drive 0 0
✔ Delete virtual drive VD0 from DG0 successfully.
[root@graid ~]# 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
|————————|————————|————————|————————————|————————————|
| DG ID | VD ID | SIZE           | DEVICE PATH | STATE       |
|————————|————————|————————|————————————|————————————|
|      0 |      0 | 500 GB | /dev/gvd0          | OPTIMAL |
|————————|————————|————————|————————————|————————————|
```

Replace a worn-out or broken SSD

1. First mark the physical drive as bad with the following command (If the physical drive already in **BAD** state, you can skip this step):

```
$ sudo graidctl edit pd <OLD_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>
```

6. Delete the old physical drive

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

Create the physical drive from the NVMe-oF drive

1. Connect remote NVMe-oF target

```
[root@graid ~]# graidctl create nvmeof_target rdma 192.168.2.10 ipv4 4420
✔ Create nvmeof target NVMe-oF target 0: {rdma,192.168.2.10,ipv4,4420} successfully.
```

2. Check NVMe drives from remote NVMe-oF target:

```
[root@graid ~]# graidctl list nvme_drive
✔ List nvme drive successfully.
╭──────────────────┬──────────────────────────────────────────────┬─────────┬──────────╮
│ DEVICE PATH (10) │ NQN                                          │ MODEL   │ CAPACITY │
├──────────────────┼──────────────────────────────────────────────┼─────────┼──────────┤
│ /dev/nvme0       │ nqn.2021-09.com.graidtech:graid0             │ Linux   │ 3.2 TB   │
│ /dev/nvme1       │ nqn.2021-09.com.graidtech:graid1             │ Linux   │ 3.2 TB   │
│ /dev/nvme2       │ nqn.2021-09.com.graidtech:graid2             │ Linux   │ 3.2 TB   │
│ /dev/nvme3       │ nqn.2021-09.com.graidtech:graid3             │ Linux   │ 3.2 TB   │
╰──────────────────┴──────────────────────────────────────────────┴─────────┴──────────╯
```

3. Create physical drives:

```
[root@graid ~]# graidctl create physical_drive nqn.2021-09.com.graidtech:graid0 nqn.2021-09.com.graidtech:graid1
nqn.2021-09.com.graidtech:graid2 nqn.2021-09.com.graidtech:graid3
✔ Create physical drive PD0 (nqn.2021-09.com.graidtech:graid0) successfully.
✔ Create physical drive PD1 (nqn.2021-09.com.graidtech:graid1) successfully.
✔ Create physical drive PD2 (nqn.2021-09.com.graidtech:graid2) successfully.
✔ Create physical drive PD3 (nqn.2021-09.com.graidtech:graid3) successfully.
```

4. Create a RAID5 drive group with 4 physical drives:

```
[root@graid ~]# graidctl create drive_group raid5 0-3
✔ Create drive group DG0 successfully.
```

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
$ sudo rpm -Uvh <graid-x.x.x-PASCAL.RELEASE.git_xxxxxxx.el8.x86_64.rpm>
# In openSUSE, SLES
$ sudo rpm -Uvh <graid-x.x.x-PASCAL.RELEASE.git_xxxxxxx.x86_64.rpm>
# In Ubuntu
$ sudo dpkg -i <graid-x.x.x-PASCAL.RELEASE.git_xxxxxxx.x86_64.deb>
```

5. Start the management service

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

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 and restart 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 must apply the new license of the new GRAID SupremeRAID card on the new host

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

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. 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.