Intel® Virtual RAID on CPU (Intel® VROC), Intel® Rapid Storage Technology enterprise (Intel® RSTe)

Software User Guide

April 2017
Revision 1.3

Intel Corporation | NSG

Revision History

Revision	Description	Date
001	Initial release.	March 2016
005	Added VROC- RSTe 5.0 features	August 2016
5.0-BETA-VC3	Add VROC VC Patch to be applied to RHEL 7.3 VC	November 2016
5.0-PC	Added LED Management Details	February 2017
1.2	Verification of CLI output for updated build	March 2017
1.3	Minor Update	April 2017

Contents

1 Introduction

The purpose of this document is to help enable a user to properly set up, configure, and manage Intel® Virtual RAID on CPU (VROC) RAID volumes on NVMe drives managed by the Intel Volume Management Device (VMD) controller as well as Intel RSTe RAID volumes on SATA drives attached to the SATA and/or sSATA controllers for Linux* Operating System. Within the Linux* OS, the primary configuration software to manage Intel RSTe RAID is the **mdadm** application, a native Linux* tool that is used exclusively with Intel RSTe on Linux.

This document describes how to use this application and many of the options offered to setup, manage and monitor Intel VROC RAID. This document also provides a brief explanation of the RAID levels and options to best suit your applications performance and budget.

Note: The information in this document is only relevant on systems with a supported Intel chipset and, with a supported operating system. Please ensure that your system supports Intel NVMe SSDs before continuing.

Note: The majority of the information in this document is related to either software configuration or hardware integration. Intel is not responsible for the software written by third party vendors or the implementation of Intel components in the products of third party manufacturers.

Customers should always contact the place of purchase or system/software manufacturer with support questions about their specific hardware or software configuration.

1.1 Terminology

Term	Description
API	Application Programming Interface
BIOS	Basic Input/Output System
Continuous Update Policy	When a recovery volume is using this policy, data on the master drive is copied to the recovery drive automatically as long as both drives are connected to the system.
Array	This term is representative of an mdadm container required for Intel metadata based volumes using the IMSM option during Volume creation.
Container	A container is a type of array used with Intel® metadata.
Fio	Flexible I/O Tester
GB	Gigabyte
Hot- Plug	The unannounced removal and insertion of a drive while the system is powered on.

Term	Description
I/O	Input/Output
Initramfs	Initial Ram File system
IMSM	Intel® Matrix Software Manager
IOPS	I/O's Per Second
Intel® Matrix Storage Manager Option ROM	A code module built into the system BIOS that provides boot support for RAID volumes as well as a user interface for configuring and managing RAID volumes.
KB	Kilobyte
Legacy Mode	Refers to the system setting in the BIOS
Matrix RAID	Two independent RAID volumes within a single RAID array.
MB	Megabyte
Member	An NVMe drive used within a RAID array.
Mdadm	mdadm is a Linux utility developed to manage software RAID devices on Linux. It is available under the GPL license version 2 or later and supports Intel NVMe SSDs.
NVMe	Non-volatile Memory Express
On Request Update Policy	When a recovery volume is using this policy, data on the master drive is copied to the recovery drive when you request it. Only changes since the last update process are copied.
OS	Operating System
POST	Power-On Self-Test
Pre-OS	A BIOS option to configure Intel RSTe RAID.
RAID	Redundant Array of Independent Drives: allows data to be distributed across multiple drives to provide data redundancy or to enhance data storage performance.

Term	Description
RAID 0 (striping)	The data in the RAID volume is striped across the array's members. Striping divides data into units and distributes those units across the members without creating data redundancy, but improving read/write performance.
RAID 1 (mirroring)	The data in the RAID volume is mirrored across the RAID array's members. Mirroring is the term used to describe the key feature of RAID 1, which writes duplicate data from one drive to another; therefore, creating data redundancy and increasing fault tolerance.
RAID 5 (striping with parity)	The data in the RAID volume and parity are striped across the array's members. Parity information is written with the data in a rotating sequence across the members of the array. This RAID level is a preferred configuration for efficiency, fault-tolerance, and performance.
RAID 10 (striping and mirroring)	The RAID level where information is striped across a two drive arrays for system performance. Each of the drive in the array has a mirror for fault tolerance. RAID 10 provides the performance benefits of RAID 0 and the redundancy of RAID 1. However, it requires four hard drives so it's the least cost effective.
RAID Array	A logical grouping of physical drives.
RAID Volume	A fixed amount of space across a RAID array that appears as a single physical drive to the operating system. Each RAID volume is created with a specific RAID level to provide data redundancy or to enhance data storage performance.
Recovery Drive	The drive that is the designated target drive in a recovery volume.
Recovery Volume	A volume utilizing Intel® Rapid Recover Technology.
Intel® RSTe	Intel® Rapid Storage Technology enterprise.
RWH	RAID Write Hole
TB	Terabyte
UEFI Mode	*Unified Extensible Firmware Interface*. Refers to the system setting in the BIOS
Volume	This term is representative of mdadm RAID within an Intel metadata based container.
Intel® VROC	Intel® Virtual RAID on CPU

1.2 Reference Documents

Document	Document No./Location
Intel® NVMe SSDs and Intel RSTe for Linux (Quick Start Guide).pdf	333745_001US
NVM_Express_1_2_Gold_20141209.pdf	Nvmexpress.org
Intel® RSTe for Linux* Software User's Manual.pdf (SATA)	327602-001US
Intel® RSTe and Intel VROC Technical Product Specification (TPS)	XXXXX-XXXXX
mdadm man pages	Linux man pages
mdadm.conf man pages	Linux man pages
mdmon man pages	Linux man pages

2 Intel RSTe Features

2.1 New to Intel® RSTe 5.1

Several new features have been added to Intel RSTe which will be up streamed to the Linux open source community for future **mdadm.** These features will include additional command line options to **mdadm,** enhanced support of Intel NVMe SSDs and new features enabled as a combination of Intel based chipsets and processor capabilities. For NVMe enabled features, please review Intel® VROC Technical Product requirements to verify hardware requirements.

1. **Boot from Intel VROC RAID volumes**
2. **NVMe Hot Plug and Surprise Removal on CPU PCIe lanes**
3. **LED Management for Intel VMD enabled storage and PCH SATA storage**
4. **Close the RAID write hole failure – a combination of SW and HW that could allow recovery from a double fault**
5. **RAID/Storage management – using RESTful APIs**
6. **GUI for Linux**
7. **4K native NVMe SSD support**

2.2 Intel Matrix Storage Manager

The Intel® Matrix Storage Manager software package provides high-performance NVMe RAID capabilities and allows you to create RAID arrays and spanned volumes. Matrix arrays indicate there are two RAID volumes in a single RAID container.

As an example, a system with an Intel® C600-A chipset, IMSM allows you to create both a RAID 0 volume as well as a RAID 5 volume across four drives. An important requirement for Matrix RAID is that the volumes within the container span the same set of member drives. When creating a container with the IMSM metadata the mdadm “-e imsm” option is used.

Another feature of the IMSM metadata is the ability to roam an array between Linux and Microsoft Windows* host systems.

Intel RSTe supports the following RAID levels:

- RAID 0
- RAID 1
- RAID 5
- RAID 10
- Intel® Matrix RAID

2.3 Supported Operating Systems

Intel VROC and Intel RSTe are available in the following Linux distributions:

- Red Hat Enterprise Linux (RHEL) 7.3 GA (Intel VROC)
- SUSE Linux Enterprise Server (SLES) 12 SP3 (Intel VROC)
- Red Hat Enterprise Linux (RHEL) 6.7, 6.8, 7.2 and 7.3 GA (Intel RSTe)

- SUSE Linux Enterprise Server (SLES) 11 & 12 (Intel RSTe)
- **NOTE**: Intel VMD has been upstreamed and is included in the Linux 4.10 kernel

2.4 RAID 0 (Striping)

RAID 0 uses the read/write capabilities of two or more drives working in parallel to maximize the storage performance of a computer system.

The following table provides an overview of the advantages, the level of fault-tolerance provided, and the typical usage of RAID 0.

Table 1: RAID 0 Overview

Drives Supported	2-Max supported by host system
Advantage	High transfer rates
Fault-tolerance	None – If one drive fails all data will be lost
Application	Typically used in desktops and workstations for maximum performance for temporary data and high I/O rate. 2-drive RAID 0 available in specific mobile configurations. It also should be noted that although RAID 0 can be scaled to many drives there is a performance sweet spot specific to your implementation.

2.5 RAID 1 (Mirroring)

RAID 1 arrays contain two drives where the data between the two is mirrored in real time to provide good data reliability in the case of a single disk failure; when one disk drive fails, all data is immediately available on the other without any impact to the integrity of the data.

The following table provides an overview of the advantages, the level of fault-tolerance provided, and the typical usage of RAID 1.

Table 2: RAID 1 Overview

Drives Supported	2
Advantage	Redundancy of data. One drive may fail, but data will continue to be accessible. A rebuild to a new drive is recommended to maintain data redundancy.
Fault-tolerance	Excellent – Drive mirroring means that all data on one drive is duplicated on another drive.
Application	Typically used for smaller systems where capacity of one disk is sufficient and for any application(s) requiring very high availability. Available in specific mobile configurations.

2.6 RAID 5 (Striping with Parity)

RAID 5 arrays contain a minimum of three or more drives where the data and parity are striped across all the drives in the array. Parity is a mathematical method for recreating data that was lost from a single drive, which increases fault-tolerance. If there are N drives in the RAID 5 array, the capacity for data would be N - 1 drives. For example, if the RAID 5 array has 5 drives, the data capacity for this RAID array consists of 4 drives.

The following table provides an overview of the advantages, the level of fault-tolerance provided, and the typical usage of RAID 5.

Table 3: RAID 5 Overview

Drives Supported	3-Max supported by host system
Advantage	High percentage of usable capacity and high read performance as well as fault-tolerance.
Fault-tolerance	Excellent - Parity information allows data to be rebuilt after replacing a failed drive with a new drive.
Application	Storage of large amounts of critical data. Not available in mobile configurations. As with RAID 0 Striping, although RAID 5 can be scaled to many drives there is a performance sweet spot specific to your implementation.

2.7 RAID 10

A RAID 10 array uses four drives to create a combination of RAID levels 0 and 1. It is a striped set whose members are each a mirrored set. It provides a great balance between performance and excellent fault tolerance as it allows 2 drives to fail while still maintaining access to data but, has a low cost effectiveness.

The following table provides an overview of the advantages, the level of fault-tolerance provided, and the typical usage of RAID 10.

Table 4: RAID 10 Overview

Drives Supported	4
Advantage	Combines the read performance of RAID 0 with the fault-tolerance of RAID 1.
Fault-tolerance	Excellent – Drive mirroring means that all data on one drive is duplicated on another drive.
Application	High-performance applications requiring data protection, such as video editing.

2.8 Intel® VROC RAID Write Hole Closure

The Intel RSTe 5.X will support the ability to close the RAID Write Hole scenario in RAID 5 configurations. This applies to Intel VROC Enabled platforms.

RAID Write Hole (RWH) is a fault scenario, related to parity based RAID. It occurs when a power-failure/crash and a drive-failure (e.g., strip write or complete drive crash) occur at the same time or very close to each other. Unfortunately, these system crashes and disk failures are correlated events. This can lead to silent data corruption or irrecoverable data due to lack of atomicity of write operations across member disks in parity based RAID. Due to the lack of atomicity, the parity of an active stripe during a power-fail may be incorrect and inconsistent with the rest of the strip data; data on such inconsistent stripes does not have the desired protection, and worse, can lead to incorrect corrections (silent data errors).

The previous RSTe mechanisms implemented to address the RAID Write Hole condition encompassed a combination of Dirty Stripe Journaling and Partial Parity Logging. This implementation only partially closed the RAID Write Hole. With Intel RSTe 5.x, the RWH solution included will completely close this condition (when RWH is enabled). When RWH is disabled, the old implementation (using Dirty Stripe Journaling and Partial Parity Logging) is used.

3 Installation of Intel® RSTe 5.X for Linux

For the Production Copy release package for Intel RSTe 5.X and Intel VROC on Purley, the following instructions will provide a user an opportunity to install an archive with custom components with Intel RSTe 5.X functionality that incorporates VMD that is part of the Purley (Sky Lake) server class processors.

3.1 Installation Requirements

The installation files support the following OS base distributions for Intel RSTe and Intel VROC:

- RHEL 7.3 GA
- SLES 12 SP3

Note: Intel VROC 5.X will not provide support for SLES 12 SP2. Intel RSTe for PCH (SATA/sSATA) support for SLES 12 will continue to be supported.

3.2 Installation Files

The following files are included in the Intel RSTe 5.x installation kit:

rste-5.1_PV_rhel7.3.iso Package:

UPDATES.IMG

SRC directory

RPM file containing the latest ledmon source

RPM file containing the latest libpciac source

RPM file containing the latest mdadm source

RPM file containing the latest md-rste source

RPM file containing the latest util-linus source

RPM file containing the latest vmd source

RPMS directory

REPODATA directory

~~(**GZ names may vary with each release)~~The latest set of GZ files

Formatted: Indent: Left: 0.5"

The latest supporting filelists.xml.gz files

The latest supporting primary.xml.gz files

The latest supporting other.xml.gz files

The latest ~~145BADF0.GZ~~

~~243D5E59.GZ~~

~~3741F59D.GZ~~

~~REPOMD.XML~~repomd.xml file

The latest kmod-md-rste RPM file(s)

The latest kmod-vmd RPM file(s)

The latest ledmon RPM file(s)

The latest libblkid RPM file(s)

The latest libmoung RPM file(s)

The latest libpciaccess RPM file(s)

The latest libuuid RPM file(s)

The latest mdadm RPM file(s)

The latest util-linux RPM file(s)

This is a custom modified RPM files that contain the latest Intel VMD-enabled NVMe Driver, and Intel RSTe 5.X packaged and applied to RHEL 7.3 GA release.

3.3 RHEL 7.3 Installation Process

This process will allow users to install RHEL 7.3 GA on:

1. A Purley platform with VMD disabled and install to a SATA/NVMe device, apply/install the RPM files to upgrade to Intel RSTe 5.X Production Copy package and then enable VMD on the system.
2. A Purley platform with VMD enabled and install to an VMD managed NVMe device or Intel VROC managed RAID volume

3.3.1 Installation with VMD disabled

This process will focus on installing RHEL 7.3 GA on a platform that has VMD disabled for the installation and will enable VMD one the patches are applied.

1. Use a tool to write the rste-5.1_PV_rhel7.3.iso file to a USB drive
2. Install stock RHEL7.3 GA to a SATA drive
3. Once RHEL 7.3 GA has been installed, mount the USB drive
4. Go down to the RPMS directory and perform
5. Yum install ./*
6. Reboot
7. Go into the BIOS setup and enable VMD

3.3.2 Installation with VMD enabled

This process will focus on installing RHEL 7.3 GA on a platform that has VMD. These steps support installing RHEL 7.3 to a SATA drive, NVMe drive or Intel VROC managed RAID volume (creating an Intel VROC managed RAID volume in the UEFI HII is not covered in this document).

1. Use a tool to write the rste-5.1_PV_rhel7.3.iso file to a USB drive.
    a. If you are creating this USB using Linux, you may use dd to convert the file.
        i. Alternately, you may place this file on an HTTP/HTTPS/NFS server and use the add option instruction from step 5.
    b. If using Windows to create the USB, you will need to unzip the iso onto the USB drive.
    c. This will preferably be a clean, formatted USB in FAT32.

i. The name of the USB **must** be RSTE.

ii. The files contained after the unzip should be in the main tree will read as follows:

1. src (folder)
2. rpms (folder)
3. updates.image

2. Insert the USB drive into the platform along with the installation media.
3. Begin installing RHEL 7.3 and when prompted please go to the "Install Red Hat Enterprise Linux 7.3" and press "e" to enter into edit mode
4. You will see the following lines of text:

   setparams 'Install Red Hat Enterprise Linux 7.3'

   ```
   linuxefi /images/pxeboot/vmlinuz inst.stage2=hd:LABEL=RHEL-7.3\x20Serv\er.x86_64 quiet

   initrdefi /images/pxeboot/initrd.img


   Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to discard edits and return to the
   menu. Pressing Tab lists possible completions.
   ```

5. Append the following text the first "linuxefi…" line so that you have the following string upon the screen:

   ```
   linuxefi /images/pxeboot/vmlinuz inst.stage2=hd:LABEL=RHEL-7.3\x20Serv\er.x86_64 quiet
   inst.updates=LABEL=RSTE modprobe.blacklist=qat_c62x

   initrdefi /images/pxeboot/initrd.img
   ```

6. Press "Ctrl-x" to continue the installation process.

Note: There is only 1 space following quiet inst.updates=LABEL=RSTE modprobe.blacklist=quat_c62x

If you use the BKC auto installer to perform the installation, you will not be required to add the modprobe.blacklist=qat_c62x parameter.

modprobe.blacklist=quat_c62x is specifically applicable to systems that are using Lewisburg platform hardware. Not all systems will be required to utilize this additional appended string. It will not harm non-Lewisburg platforms.

3.3.3 Loading the Intel Accelerated Storage Manager (Intel ASM)

These instructions will outline out to install the Intel ASM tool on a new install of RHEL 7.3.

1. Once RHEL 7. 3 has been properly booted copy the IntelASM-1.1.0.2-RHEL7.zip to the platform
2. Unzip the file.
3. Go into the IntelASM-1.1.0.2-RHEL7 directory
4. Go to the ISM directory
5. Run "yum install ./*"

6. Go back up one directory level
7. Go to the RSTe directory
8. Run “yum install ./*”
9. Go back up one directory level
10. Go to www directory
11. Run “yum install ./*”
12. Intel ASM has now been installed. Please reference the Intel ASM documentation for instructions how to configure and run the tool.

4 Volume Creation in Linux with mdadm

RAID arrays can be created within the BIOS after enabling VMD and using the Intel VROC integrated features, or using the mdadm command line utility, which supports the Intel® Matrix Storage Manager (IMSM) metadata format when specified with the -e imsm metadata option.

4.1 Erasing RAID Metadata

Having incorrect or bad metadata can cause RAID volumes to be assembled incorrectly. The metadata can be erased on each potential member drive with the following command to ensure the drive is clean. This operation does not attempt to wipe existing user data but, will delete an array if it's a current member drive.

```
# mdadm --zero-superblock /dev/<nvmeXn1>
```

Multiple drives can be specified to clear the superblock at the same time.

4.2 RAID Volume Creation

WARNING: *Creating a RAID array will permanently delete any existing data on the selected drives. Back up all important data before beginning these steps.*

The following shows an example of how to create a RAID5 array with 4 Intel NVMe SSDs:

1. First, a container of Intel IMSM metadata **must** be created.

```
# mdadm -C /dev/md0 /dev/nvme[0-3]n1 -n 4 -e imsm
Continue creating array? Y
mdadm: container /dev/md0 prepared.
```

   The command creates a RAID container with Intel® Matrix Storage Manager metadata format. The device node for the container will be `/dev/md0`. In this example drives nvme0n1, nvme1n1 nvme2n1 and nvme3n1 are used for this RAID container, and the total number of drives is 4. The wildcard expression `/dev/nvme[0-3]n1` can be used to specify the range of drives. Individual drives can be also used.

2. Next, a RAID 5 volume is created.

```
# mdadm -C /dev/md/Volume0 /dev/md0 -n 4 -l 5
```

   The command creates a RAID 5 volume /dev/md/Volume0 within the `/dev/md0` container.

   The following command parameters may also be used in conjunction to give finer control for the creation of the RAID volume.

`-n`	Number of active RAID devices to be used in the volume.
`-x`	Specifies the number of spare devices in the initial array.
`-c`	Specifies the chunk (strip) size in Kilobytes. The default is 128KiB. This value must be a multiple of 4KiB. Optimal chunk size should be considered depending on expected workload profiles.
`-l`	specifies the RAID level. The options are 0, 1, 5, 10.
`-z`	Specifies the size (in Kilobytes) of space dedicated on each disk to the RAID volume. This must be a multiple of the chunk size. For example:

```
# mdadm -C /dev/md/Volume0 /dev/md0 -n 4 -l 5 -z $((100*1024*1024))
```

   The command above creates a RAID volume inside the /dev/md0 container with 100GiB of disk space used on each drive member.

A suffix of ‘M’ or ‘G’ can be given to indicate Megabytes or Gigabytes respectively. This applies also to the –c parameter. So the above command is equivalent to this:

```
# mdadm –C /dev/md/Volume0 /dev/md0 –n 4 –l 5 –z 100G
```

4.3 File System Creation on a RAID Volume

After the RAID volume has been created, a file system can be created in order to allow the mounting of the RAID volume.

```
# mkfs.ext4 /dev/md/Volume0
```

After the file system has been created, it can be mounted to the location of choice:

```
# mount /dev/md/Volume0 /mnt/<mountpoint>
```

4.4 Additional RAID Container and Volume Creation Examples

Create a 2 drive container and a RAID 0 volume

```
# mdadm -C /dev/md/imsm0 /dev/nvme[0-1]n1 -n 2 -e imsm
# mdadm -C /dev/md0 /dev/md/imsm0 -n 2 -l 0
```

Create a 2 drive container and a RAID 1 volume

```
# mdadm -C /dev/md/imsm0 /dev/nvme[0-1]n1 -n 2 -e imsm
# mdadm -C /dev/md0 /dev/md/imsm0 -n 2 -l 1
```

Create a 3 drive container and a RAID 5 volume

```
# mdadm -C /dev/md/imsm0 /dev/nvme[0-2]n1 -n 3 -e imsm
# mdadm -C /dev/md0 /dev/md/imsm0 -n 3 -l 5
```

Create a 4 drive container and a RAID 10 volume

```
# mdadm -C /dev/md/imsm0 /dev/nvme[0-3]n1 -n 4 -e imsm
# mdadm -C /dev/md0 /dev/md/imsm0 -n 4 -l 10
```

Note: To create multiple RAID volumes in the same container, they MUST span equal number of drives. For example, in order to have a RAID 0 volume and a RAID 5 volume in the same container, four drives must be used for both volumes.

Adding a spare drive allows for immediate reconstruction of the RAID volume when a device failure is detected. mdadm will mark the failed device as “bad” and start reconstruction with the first available spare drive. The spare drive can also be used to grow the RAID volume (refer to section 6 for Online Capacity Expansion information). The spare drives sit idle during normal operations. When using mdadm with imsm metadata, the spare drive added to a container is dedicated to that specific container. The following command adds a spare drive to the designated container */dev/md0*.

```
# mdadm -a /dev/md0 /dev/<mvmeXn1>
```

4.5 Creating RAID Configuration File

A configuration file can be created to record the existing RAID volumes. The information can be extracted from the existing RAID setup. The configuration file is typically stored at the default location of /etc/mdadm.conf. This allows a consistent assembling of the appropriate RAID volumes.

```
# mdadm -E –s > /etc/mdadm.conf
```

4.6 RAID Volume Initialization/Resync

Immediately after a RAID volume has been created, initialization (or resync) commences if the RAID level is 1, 10, or 5. During this time, any data stored on RAID level 5 volumes are not guaranteed to be safe if failure occurs. If a drive failure happens during the initialization time, recovery will not be possible. This scenario is also true during RAID volume rebuilds.

5 Additional Volume Operations with mdadm

The mdadm application provides various options to assemble, monitor, examine, or stop RAID volumes.

5.1 Volume Assembly

RAID volumes can be created on a supported system and one of the supported operating systems with the mdadm application. All inactive RAID volumes can be activated using the assemble option with mdadm.

The following command scans for the mdadm configuration file at `/etc/mdadm.conf` in order to assemble the RAID volumes. If the configuration file is not found, it scans all available drives for RAID members and assembles all the RAID volumes:

```
# mdadm -A -s
```

To manually assemble and activate RAID volumes without the configuration file, the following example can be used:

```
# mdadm -A /dev/md0 -e imsm /dev/<member drives>
```

The first command assembles the container with the name `/dev/md0` using the provided list of drives. The second command assembles and activates appropriate volumes with the device nodes.

For additional information on mdadm.conf, consult the Linux man pages.

5.2 Stopping the Volumes

To stop all active RAID volumes, the following command can be used. mdadm will scan for and stop all running RAID volumes and containers.

```
# mdadm -S -s
```

However, RAID volume names can be specified to stop the volume directly.

```
# mdadm -S /dev/md/Volume0
```

And to stop a container, the following command can be used.

```
# mdadm -S /dev/md0
```

5.3 Reporting RAID Information

Use the following command, to print out details about a RAID container or volume:

```
# mdadm -D /dev/md0
/dev/md0:
        Version : imsm
     Raid Level : container
  Total Devices : 4

Working Devices : 4
          UUID : b559b502:b199f86f:ee9fbd40:cd10e91d
  Member Arrays :

    Number   Major   Minor   Raid Device
       0       8       32        -        /dev/nvme0n1
       1       8       48        -        /dev/nvme1n1
       2       8       80        -        /dev/nvme2n1
       3       8       96        -        /dev/nvme3n1
```

To display details about a RAID volume:

```
# mdadm -D /dev/md/Volume0
/dev/md/Volume0:
      Container : /dev/md0, member 0
     Raid Level : raid1
     Array Size : 781408256 (745.21 GiB 800.16 GB)
  Used Dev Size : 781409024 (745.21 GiB 800.16 GB)
   Raid Devices : 2
  Total Devices : 2


          State : clean, resyncing
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0


  Resync Status : 38% complete

           UUID : 084d2b20:09897744:36757c5b:77e0e945
    Number   Major   Minor   Raid Device State
       0     259       2        0      active sync   /dev/nvme1n1
       1     259       1        1      active sync   /dev/nvme0n1
```

To print out RAID details about a member drive:

```
# mdadm -E /dev/nvme0n1
/dev/nvme0n1:
          Magic : Intel Raid ISM Cfg Sig.
        Version : 1.1.00
    Orig Family : e39f3751
         Family : e39f3751
     Generation : 0000000d
     Attributes : All supported
           UUID : 1798e8a4:7a6d775d:15edb950:14fe36e
       Checksum : cadac84f correct
    MPB Sectors : 2
          Disks : 2
   RAID Devices : 1

  Disk01 Serial : FT42920017800HGN
          State : active
             Id : 00000000
    Usable Size : 1562818062 (745.21 GiB 800.16 GB)

[Volume0]:
           UUID : a951b88c:400337582ad56c75:323c49c3
     RAID Level : 1 <-- 1
        Members : 2 <-- 2
          Slots : [UU] <-- [UU]
    Failed Disk : none
      This Slot : 1
    Sector Size : 512
     Array Size : 1562816512 (745.21 GiB 800.16 GB)
   Per Dev Size : 1562818048 (745.21 GiB 800.16 GB)
  Sector Offset : 0
    Num Stripes : 6104758
     Chunk Size : 64 KiB <-- 64 KiB
       Reserved : 0
  Migrate State : initialize
      Map State : normal
    Dirty State : clean
     RWH Policy : off


  Disk00 Serial : FT546200U1P6JGN
          State : active
             Id : 00000000
    Usable Size : 3125621262 (1490.41 GiB 1600.32 GB)
```

To get the most current status on all RAID volumes, the file `/proc/mdstat` can be examined. This file is a special file that is updated continuously to show the status of all the containers, and RAID volumes. In the example below, the status shows that currently available supported RAID levels are 0, 1, 5 and 10. md126 is the active RAID volume with RAID level 5 and 128k strip size. The RAID volume contains 4 drives that are all in normal (UP) status. “md127” is the IMSM container for the RAID volume.

```
# cat /proc/mdstat

Personalities : [raid6] [raid5] [raid4] [raid1]
md127 : active raid1 nvme2n1[2] nvme0n1[0]
      104853504 blocks super external:/md1/0 [2/2] [UU]

md0 : inactive nvme2n1[2](S) nvme1n1[1](S) nvme0n1[0](S)
      3315 blocks super external:imsm

unused devices: <none>
```

Note: When creating containers and volumes, you will notice that in `/proc/mdstat` the device names will not match up exactly. For example, when /dev/md/Volume0 is created, md127 will be shown in `/proc/mdstat` and other detail output as well. The `/dev/md/Volume0` is created as an alias of `/dev/md127` device node. Looking in the `/dev/md` directory, one will notice that `/dev/md/Volume0` is a soft link to `/dev/md127`.

5.4 Fail an Active Drive

To mark an active drive as, “failed” (or set as faulty) manually, the following command can to be issued:

```
# mdadm –f /dev/md/Volume0 /dev/<nvmeXn1>
```

5.5 Remove a Failed Drive

To remove a failed drive, the following command needs to be executed. This only works on a container based RAID volume.

```
# mdadm –r /dev/md0 /dev/<nvmeXn1>
```

Note: Once the logical drive is removed from the volume with the command above the system needs to be powered off before removing the physical NVMe SSD from the system.

5.6 Report RAID Details from the System BIOS

To see what Intel® RSTe RAID support is provided by the BIOS issue the command:

```
# mdadm --detail-platform
        Platform : Intel(R) Rapid Storage Technology enterprise
       Version : 5.0.0.1156
    RAID Levels : raid0 raid1 raid10 raid5
    Chunk Sizes : 4k 8k 16k 32k 64k 128k

    2TB volumes : supported

      2TB disks : supported
      Max Disks : 8
    Max Volumes : 2 per array, 8 per controller
 I/O Controller : /sys/devices/pci0000:00/0000:00:11.5 (SATA)
          Port0 : - no device attached -
          Port1 : - no device attached -
          Port2 : - no device attached -
          Port3 : - no device attached -
          Port4 : - no device attached -
          Port5 : - no device attached -

       Platform : Intel(R) Rapid Storage Technology enterprise
        Version : 5.0.0.1156
    RAID Levels : raid0 raid1 raid10 raid5
    Chunk Sizes : 4k 8k 16k 32k 64k 128k
    2TB volumes : supported
      2TB disks : supported
      Max Disks : 8
    Max Volumes : 2 per array, 8 per controller
 I/O Controller : /sys/devices/pci0000:00/0000:17.0 (SATA)
          Port4 : /dev/sda (W629KMP2)
          Port0 : - no device attached -
          Port1 : - no device attached -
          Port2 : - no device attached -
          Port3 : - no device attached -
          Port5 : - no device attached -
          Port6 : - no device attached -
          Port7 : - no device attached -

       Platform : Intel(R) Rapid Storage Technology enterprise
        Version : 5.0.0.1217
    RAID Levels : raid0 raid1 raid10 raid5
    Chunk Sizes : 4k 8k 16k 32k 64k 128k
    2TB volumes : supported
      2TB disks : supported
      Max Disks : 24
    Max Volumes : 2 per array, 24 per controller
 3rd Party NVMe : supported
 I/O Controller : /sys/devices/pci0000:85/0000:85:05.5 (VMD)
 I/O Controller : /sys/devices/pci0000:17/0000:17:05.5 (VMD)
 I/O Controller : /sys/devices/pci0000:5d/0000:5d:05.5 (VMD)
 NVMe under VMD :
/sys/devices/pci0000:5d/0000:5d.05.5/pci10001:00/10001:00:02.0/10001:01:00.0
 NVMe under VMD : /sys/devices/pci0000:5d/0000:5d:05.5
/pci10001:5d.05.5/pci10001:00:03.0/10001:02:00.0
 I/O Controller : /sys/devices/pci0000:d7/0000:d7:05.5 (VMD)
 NVMe under VMD :
/sys/devices/pci0000:d7/0000:d7:05.5/pci10004:00/10004:00:00.0/10004:01:00.0
```

```
NVMe under VMD : /sys/devices/pci0000:d7/0000:d7:05.5/pci10004:00:01.0/10004:02:00.0
I/O Controller : sys/devices/pci0000:ae:05.5 (VMD)
```

5.7 Logging

Various messages coming from MDRAID subsystem in the kernel are logged. Typically the messages are stored in the log file `/var/log/messages` in popular Linux distributions with other kernel status, warning, and error outputs.

Below is an example snippet of what the log may look like:

```
Jun 17 06:20:04 testbox kernel: raid5: allocated 5334kB for md126
Jun 17 06:20:04 testbox kernel: 0: w=1 pa=0 pr=5 m=1 a=0 r=5 op1=0 op2=0
Jun 17 06:20:04 testbox kernel: 1: w=2 pa=0 pr=5 m=1 a=0 r=5 op1=0 op2=0
Jun 17 06:20:04 testbox kernel: 2: w=3 pa=0 pr=5 m=1 a=0 r=5 op1=0 op2=0
Jun 17 06:20:04 testbox kernel: raid5: raid level 5 set md126 active with 5 out of 5
devices, algorithm 0
Jun 17 06:20:04 testbox kernel: RAID5 conf printout:
Jun 17 06:20:04 testbox kernel: --- rd:5 wd:3
Jun 17 06:20:04 testbox kernel: disk 0, o:1, dev:nvme0n1
Jun 17 06:20:04 testbox kernel: disk 1, o:1, dev:nvme1n1
Jun 17 06:20:04 testbox kernel: disk 2, o:1, dev:nvme2n1
Jun 17 06:20:04 testbox kernel: disk 3, o:1, dev:sdb
Jun 17 06:20:04 testbox kernel: md127: detected capacity change from 0 to 40959475712
Jun 17 06:20:04 testbox kernel: md127: unknown partition table
Jun 17 06:20:04 testbox kernel: md: md127 switched to read-write mode.
```

5.8 Raid Level Migration

The RAID level migration feature allows changing of the RAID volume level without loss of data stored on the volume. It does not require re-installation of the operating system. All applications and data remain intact.

Warning: *Even though the data remains intact you should always back-up your data before performing migration operations.*

The following table shows the available migration support with Intel® IMSM metadata. You must have the number of drives necessary for the level you're converting to as spare drives.

Table 5: Migration Capabilities with IMSM

Destination → ↓Source level	**RAID 0**	**RAID 1**	**RAID 10**	**RAID 5**
RAID 0	N/A	No	**Yes**	**Yes**
RAID 1	**Yes**	N/A	No	***Yes**
RAID 10	**Yes**	No	N/A	***Yes**
RAID 5	No	No	No	N/A

*Migrations from RAID 1 to RAID 5 or from RAID 10 to RAID 5 must be done in two steps. A conversion to RAID 0 first is necessary before converting to RAID 5. During the second step (migration from RAID 0 to RAID 5) the addition of spare drive(s) may be needed.

Before performing any grow operations (if not already done), one environmental variable needs to be set:

```
# export MDADM_EXPERIMENTAL=1
```

Note: This feature is only available starting from mdadm v3.2.5. This setting is case sensitive

The following is an example of migration from RAID 1 to RAID 5:

```
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4] [raid1]
md127 : active raid1 nvme2n1[2] nvme0n1[0]
      104853504 blocks super external:/md1/0 [2/2] [UU]
md0 : inactive nvme2n1[2](S) nvme1n1[1](S) nvme0n1[0](S)
      3315 blocks super external:imsm
unused devices: <none>
```

First step is to migrate from RAID 1 to RAID 0:

```
# mdadm –G /dev/md127 –l 0
Personalities : [raid1] [raid0]
md127 : active raid0 nvme0n1[1]
      781408256 blocks super external:/md0/0 64k chunks
md0 : inactive nvme1n1[1](S) nvme0n1[0](S)
      2210 blocks super external:imsm
unused devices: <none>
```

Use Online Capacity Expansion to go from 1-disk RAID 0 to 2-disk RAID 0:

```
# mdadm -G /dev/md0 -n 2
# cat /proc/mdstat
Personalities : [raid1] [raid0] [raid6] [raid5] [raid4]
md127 : active raid4 nvme2n1[2] nvme0n1[1]
      781408256 blocks super external:/md0/0 level 4, 64k chunks algorithm 5 [2/1]
[U_]
md0 : inactive nvme2n1[2](S) nvme1n1](S) nvme0n1[0](S)
      3315 blocks super external:imsm
unused devices: <none>
```

Add a spare disk to the container m:

```
# mdadm -a /dev/md0 /dev/nvme2n1
# cat /proc/mdstat
Personalities : [raid0] [raid1]
md127 : active raid0 mvme0n1[1]
      781408256 blocks super external:/md0/0 64k chunks
md0 : inactive – nvme2n1[2](S) nvme1n1[1](S) nvme0n1[0](S)
      3315 blocks super external:imsm
unused devices: <none>
```

Migrate from RAID 0 to RAID 5:

```
# mdadm -G /dev/md127 -l 5 --layout=left-asymmetric
# cat /proc/mdstat
Personalities : [raid0] [raid1] [raid5] [raid10]
md127 : active raid5 nvme2n1[2] nvme1n1[1] nvme0n1[1]
      204800 blocks super external:/md0/0 level 5, 64k chunk, algorithm 0 [3/3] [UUU]
md0 : inactive -
      3315 blocks super external:imsm
unused devices: <none>
```

Warning: *IMSM metadata only supports the left-asymmetric layout of RAID 5. The default layout is left-symmetric, so during migrations the layout for IMSM metadata has to be specified explicitly.*

5.9 Freezing Reshape

If a RAID volume is in the process of reshape, the reshape process should be frozen during the initramfs booting phase and resumed when the system is fully up and running. Starting with mdadm 3.2.5 these features are supported. Distributions from the Operating System Vendors should have taken cared of this in their init script setup utilities, but details are described below for customers that are building their own distribution.

Warning: *Even though the data remains intact you should always back-up your data before performing reshape operations.*

The parameters `--freeze-reshape` is used to pause the reshape operation during system start up initramfs phase. For example:

```
# mdadm -As --freeze-reshape
```

When reshape is frozen, the status provided by /proc/mdstat will denote the state with a hyphen such as “super external:-md127/0” instead of “super external:/md127/0”:

```
Personalities : [raid5]
md127 : active raid5 nvme2n1[2] nvme1n1[1] nvme0n1[0]
      204800 blocks super external:-md0/0 level 5, 128k chunk, algorithm 0 [3/3] [UUU]
      [>....................]  reshape =  2.2% (116736/5242880) finish=501934.9min
speed=0K/sec

md0 : inactive -
      9459 blocks super external:imsm

unused devices: <none>
```

Once the system is up, the following example with the parameter `--continue` can be used to resume the reshape process:

```
# mdadm -G /dev/md0 --continue
```

or with a volume:

```
# mdadm -G /dev/md/Volume0 -continue
```

5.10 RAID write hold closure example and man page

The Intel RSTe 5.X driver will refrain from rebuilding a RAID 5 volume before or during the RWH recovery process.

If the RWH condition occurs during a recovery process and the recovery has not completed, Intel RSTe 5.X will attempt to resume the recover process from where it was interrupted.

If the RWH condition occurs during the process of the OS going into hibernation mode (S4) the RWH recovery process will be able to fix the failure condition for all the data being written during hibernation.

For RWH example:

To create a RAID5 volume with enabled RWH policy:

```
# mdadm -C /dev/md0 /dev/nvme[0-3]n1 -n 4 -e imsm

Continue creating array? Y
mdadm: container /dev/md0 prepared.

# mdadm -C /dev/md/vol0 -l5 -n4 /dev/nvme[0-3]n1 --rwh-policy=ppl
```

To check the current RWH policy:

```
# mdadm -D /dev/md/Volume0

/dev/md/Volume0:

     Container : /dev/md0, member 0

    Raid Level : raid5

    Array Size : 117212672 (1117.83 GiB 1200.26 GB)

 Used Dev Size : 390708224 (372.61 GiB 400.09 GB)

  Raid Devices : 4

 Total Devices : 4


         State : clean, resyncing

 Active Devices : 4

Working Devices : 40

 Failed Devices : 0

  Spare Devices : 0


         Layout : left-asymmetric

      Chunk Size: 128K

     RWH Policy : ppl


           UUID : a0470058:714d8834:f9daaa0:f20a1a1aec

   Number     Major     Minor     RaidDevice    State

   0          259       1         0             active sync   /dev/nvme2n1

   0          259       0         1             active sync   /dev/nvme3n1

   0          259       3         2             active sync   /dev/nvme4n1

   0          259       2         3             active sync   /dev/nvme1n1
```

To disable RWH policy for a running array:

```
# mdadm --rwh-policy=off /dev/md/Volume0
```

To enable RWH policy for a running array:

```
# mdadm --rwh-policy=ppl /dev/md/Volume0
```

6 Online Capacity Expansion

The Online Capacity Expansion (OLCE) feature allows the capacity expansion of the RAID volumes. With the “online” feature, the operation can be performed while a file system is mounted on top of the RAID volume. This allows avoiding having down time from taking the RAID volume offline for service or loss of data.

The size of a RAID volume can be increased by adding additional drives to the RAID container or (only if it is the last volume in the container) by expanding it on existing unused drive space available to the RAID volume. In the first case if two volumes exist in the same container, OLCE is performed automatically on both volumes (one by one) because of the requirement that all volumes must span the same set of disks for IMSM metadata.

The following commands can be issued to grow the RAID volume. The first assumes that it is the last volume in the container and we have additional room to grow, and the second assumes that an additional drive has been added to the IMSM container.

Warning: *Even though the data remains intact you should always back-up your data before performing expansion operations.*

If there is additional room in the last volume of the container, the volume can be grown to the maximum available capacity.

Note: This feature is only available starting with mdadm v3.2.5:

```
# mdadm -G /dev/md/Volume0 --size=max
```

The example below adds a single drive to the RAID container and then grows the volume(s). Because IMSM volumes inside a container must span the same number of drives, all volumes are expanded. A backup file that MDRAID will store the backup superblock is specified. This file must not reside on any of the active RAID volumes that are being worked on.

```
# mdadm -a /dev/md0 /dev/<nvmeXn1>
# mdadm -G /dev/md0 -n 4 --backup-file=/tmp/backup
```

7 RAID Monitoring

There are two components within the mdadm tools to monitor events for the RAID volumes. mdadm can be used to monitor general RAID events, and mdmon provides the ability to monitor "metadata event" occurrences such as drive failures, clean-to-dirty transitions, etc. for external metadata based RAID volumes. The kernel provides the ability to report such actions to the user space via sysfs, and mdmon takes action accordingly with the monitoring capability. The mdmon polls the sysfs looking for changes in the entry value for, *array_state*, *sync_action*, and per drive *state* attribute files.

7.1 mdmon

The mdadm monitor, mdmon, is automatically started when MDRAID volumes are activated by mdadm through creation or assemble. However, if it is not, the daemon can be started manually as shown below:

1. Run the following command:

```
# mdmon /dev/md0
```

The --all parameter can be used in place of the container name to star monitors for all active containers.

> ***Note:*** mdmon must be started in the initramfs in order to support an external metadata RAID array as the root file system. mdmon needs to be restarted in the new namespace once the final root files system has been mounted.

```
# mdmon --takeover --all
```

7.2 Monitoring Using mdadm

mdadm monitoring can be started with the following command line:

```
# mdadm --monitor --scan --daemonise --syslog
```

The command above runs mdadm as a daemon to monitor all md devices. All events will be reported to syslog. The user can monitor the syslog and filter for specific mdadm events generated.

There are additional command line parameters that can be passed to mdmon at startup.

Table 6: mdadm monitor Parameters

Long form	Short form	Description
`--mail`	`-m`	Provide mail address to email alerts or failures to.
`--program or --alert`	`-p`	Provide program to run when an event is detected.
`--delay`	`-d`	Seconds of delay between polling state. Default is 60s.
`--config`	`-c`	Specify a different config file.
`--scan`	`-s`	Find mail-address/program settings in config file.
`--oneshot`	`-1`	Check for degraded arrays and then exit.
`--test`	`-t`	Generate a Test Message event against each array at startup.
`--syslog`	`-y`	Cause all events to be reported through 'syslog'. The messages have facility of 'daemon' and varying priorities.
`--increment`	`-r`	Give a percentage increment. mdadm will generate RebuildNN events with the NN indicating the percentage at which the rebuild event had happened.
`--daemonise`	`-f`	Run as background daemon if monitoring.
`--pid-file`	`-i`	Write the pid of the daemon process to specified file.
`--no-sharing`	`N/A`	This inhibits the functionality for moving spare drives between arrays.

The following table presents all the events that are reported by mdadm monitor:

Table 7: Monitoring Events

Event Name	Description
DeviceDisappeared	An MD array previously configured no longer exists.
RebuildStarted	An MD array started reconstruction.
RebuildNN	NN is a 2 digit number that indicates rebuild has passed that many percent of the total. For example, Rebuild50 will trigger an event when 50% of rebuild has completed.
RebuildFinished	An MD array has completed rebuild.
Fail[1]	An active component of an array has been marked faulty.
FailSpare[1]	A spare drive that was being rebuilt to replace a faulty device has failed.
SpareActive	A spare drive that was being rebuilt to replace a fault device is rebuilt and active.
NewArray	A new MD array has been detected in /proc/mdstat.
DegradedArray[1]	A newly discovered array appears to be degraded.
MoveSpare	A spare drive has been moved from one array in a spare group to another array to replace a failed drive. Both arrays are labeled with the same spare group.
SparesMissing[1]	The spare device(s) does not exist in comparison to the config file when the MD array is first discovered.
TestMessage[1]	Discovered new array while --test flag was used.

1: The events indicated cause email to be sent.

7.3 Configuration File for Monitoring

mdadm will check the mdadm.conf configuration file to extract the appropriate entries for monitoring. The following entries we can set to pass to mdmon:

- MAILADDR: This configuration entry allows an E-mail address to be used for alerts. Only one email address should be used.
- MAILFROM: This configuration entry sets the email address to appear from the alert emails. The default from would be the “root” user with no domain. This entry overrides the default.
- PROGRAM: This configuration entry sets the program to run when mdmon detects potentially interesting events on any of the arrays it is monitoring. There can be only one PROGRAM line in the configuration file.

7.4 Examples of monitored events in syslog

```
Personalities : [raid5]

md127 : active raid5 nvme2n1[2] nvme1n1[1] nvme0n1[0]
      204800 blocks super external:/md0/0 level 5, 128k chunk, algorithm 0 [3/3] [UUU]

md0 : inactive -
      3315 blocks super external:imsm
unused devices: <none>
```

In order to monitor all RAID containers a mdadm daemon can be started using the following command:

```
# mdadm --monitor --scan --daemonise --syslog
```

All events now will be written to syslog. After a mdadm daemon has been started the following messages can be found in /var/log/messages or the corresponding syslog file the distribution has designated:

```
May 15 09:58:40 myhost mdadm[9863]: NewArray event detected on md device /dev/md127
May 15 09:58:40 myhost mdadm[9863]: NewArray event detected on md device /dev/md0
```

When a spare drive has been added:

```
May 15 09:59:07 myhost mdadm[9863]: SpareActive event detected on md device
/dev/md127, component device /dev/<nvmeXn1>
```

When an OLCE command is finished:

```
May 15 09:59:16 myhost mdadm[9863]: RebuildFinished event detected on md device
/dev/md127
```

When a drive fails:

```
May 15 10:01:04 myhost mdadm[9863]: Fail event detected on md device /dev/md127,
component device /dev/<nvmeXn1>
```

When a rebuild finishes:

```
May 15 10:02:22 myhost mdadm[9863]: RebuildFinished event detected on md device
/dev/md127
May 15 10:02:22 myhost mdadm[9863]: SpareActive event detected on md device /dev/md127
```

When all MD devices are stopped:

```
May 15 10:03:27 myhost mdadm[9863]: DeviceDisappeared event detected on md device
/dev/md127
May 15 10:03:27 myhost mdadm[9863]: DeviceDisappeared event detected on md device
/dev/md0
```

8 Recovery of RAID Volumes

Recovery is one of the most important aspects of using RAID. It allows rebuilding of redundant RAID volumes on a system when a drive failure occurs. Recovery is only possible in the case of following RAID levels: 1, 5, and 10. General recovery is possible if no more than one drive fails. However in the case of RAID 10, recovery may be possible even if two out of four drives fail if the two failed drives are members of two different mirrored pairs. If both drives of one mirror fail, recovery is not possible.

8.1 Removing Failed Drives(s)

Below is the output of /proc/mdstat with an active RAID5 volume with IMSM metadata where md127 is the IMSM container and md126 is the RAID 5 volume. The RAID 5 volume contains drives /dev/nvme0n1, /dev/nvme1n1, /dev/nvme2n1, /dev/nvme3n1

```
Personalities : [raid0] [raid1] [raid5] [raid10]
md127 : active (read-only) raid5 nvme0n1[3] nvme1n1[2] nvme2n1[1] nvme3n1[0]
      39999488 blocks super external:/md0/0 level 5, 512k chunk, algorithm 0 [5/5]
[UUUUU]

md0 : inactive -
      11285 blocks super external:imsm

unused devices: <none>
```

When a drive fails, in this instance /dev/nvme2n1, the following is displayed in /proc/mdstat:

```
Personalities : [raid0] [raid1] [raid5]
md127 : active raid5 nvme0n1[3] nvme3n1[0]
      39999488 blocks super external:/md0/0 level 5, 512k chunk, algorithm 0 [5/4]
[_UUUU]

md0 : inactive - nvme2n1[1](S)
      1045 blocks super external:imsm

unused devices: <none>
```

The failed drive can be removed from the RAID volume by the following command:

```
# mdadm /dev/md/Volume0 --fail detached --remove detached
```

or from the container by the following command:

```
# mdadm --remove /dev/md0 /dev/nvme2n1
```

WARNING: *Once the failed drive has been removed from the array it is required the system be powered off before replacing with a fresh Intel® NVMe SSD.*

8.2 Rebuilding

At this point, this RAID volume is running in degraded mode. However, it is still operational. If there are spare drives available in the container, rebuild of the RAID volume would automatically commence. A spare drive can also be manually added to start the rebuild process:

```
# mdadm -add /dev/md0 /dev/<nvme4n1>

Personalities : [raid0] [raid1] [raid5] [raid10]
md127 : active raid5 nvme0n1[3](S) nvme1n1[2](S) nvme3n1[0](S) /nvme4n1[2](S)
      39999488 blocks super external:/md0/0 level 5, 512k chunk, algorithm 0 [5/4]
```

```
[_UUUU]
      [==>..................]  recovery = 11.5% (1154588/9999872) finish=2.6min
speed=54980K/sec

md0 : inactive - nvme2n1[1](S)
sdb[0](S)
      1254 blocks super external:imsm

unused devices: <none>
```

8.3 Auto Rebuild

Auto-rebuild allows a RAID volume to be automatically rebuilt when a drive fails. There are 3 different scenarios this can happen:

1. There is a rebuild capable RAID volume with no spare drive in the container. If one of the drives in the volume fails it enters degraded mode. When a spare drive is added manually to the container, rebuild starts automatically.
2. There is a rebuild capable RAID volume with at least one spare drive in the container. If one of the drives in the volume fails, the spare drive is automatically pulled in, and the rebuild starts.
3. There are two or more containers. One container has a spare drive and the other one does not. If mdadm is running in monitor mode, and the appropriate policy is configured in the mdadm.conf file, a spare drive will be moved automatically from one container to the other if a RAID volume is degraded and requires a spare drive for rebuild.

For scenario number three, the following example is presented below:

1. Create container ”`md1`” with 3 disks:

```
# mdadm -C /dev/md1 -e imsm -n3 /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1
```

2. Create RAID1 volume “`Volume1`” in container”`md1`”, drive /dev/nvme2n1 remains a spare:

```
# mdadm -C /dev/md/Volume1 -l1 -n2 /dev/nvme0n1 /dev/nvme1n1
```

3. Create container “`md2`” with 2 additional drives:

```
# mdadm -C /dev/md2 -e imsm -n2 /dev/nvme3n1 /dev/nvme4n1
```

4. Create RAID1 volume “`Volume2`” in container ”`md2`”, with no spare drives:

```
# mdadm -C /dev/md/Volume2 -l1 -n2 /dev/nvme0n3 /dev/nvme1n4

Personalities : [raid0] [raid1]
md126: active raid1 nvme4n1[1] nvme3n1[0]
      1048576 blocks super external:/md2/0 [2/2] [UU]

 md2 : -
      2210 blocks super external:imsm

md127: active raid1 nvme0n1[1](S) nvme1n1[0](S)
      1048576 blocks super external:/md1/0 [2/2] [UU]

 md1 : -
      3315 blocks super external:imsm

unused devices: <none>
```

5. Save configuration file:

```
# mdadm -Es > /etc/mdadm.conf
```

6. Add the policy with the same domain and the same action for all drives to the configuration file, which allows the spare to move from one container to another for rebuild:

```
# echo "POLICY domain=DOMAIN path=* metadata=imsm action=spare-same-slot" >> a
```

The configuration file in /etc/mdadm.conf will look similar below:

```
ARRAY metadata=imsm UUID=67563d6a:3d253ad0:6e649d99:01794f88 spares=1


ARRAY /dev/md/Volume2 container=67563d6a:3d253ad0:6e649d99:01794f88 member=0
UUID=76e507f1:fadb9a42:da46d784:2e2166e8


ARRAY metadata=imsm UUID=267445e7:458c89eb:bd5176ce:c37281b7


ARRAY /dev/md/Volume1 container=267445e7:458c89eb:bd5176ce:c37281b7 member=0
UUID=25025077:fba9cfab:e4ad212d:3e5fce11


POLICY domain=DOMAIN path=* metadata=imsm action=spare-same-slot
```

7. Make sure mdadm is in monitor mode:

```
# mdadm --monitor --scan -daemonise
```

8. Fail one of the drives in volume "`Volume2`", the volume without a spare:

```
# mdadm --fail /dev/md/Volume2 /dev/nvme3n1
```

The spare drive /dev/nvme2n1 should be automatically moved from the container "`md1`" to the container "`md2`" and the rebuild of "`Volume2`" will start automatically:

```
 [raid1] [raid0]
: active raid1 nvme2n1[2] nvme4n[1]
md2/0 [2/1] [_U]
 finish=0.0minspeed=144298K/sec

: inactive nvmme3n1[3](S)
      5363 blocks super external:imsm

: active raid1 nvme1n1[1] nvme0n1[0]
      1048576 blocks super external:/md1/0 [2/2] [UU]

: inactive nvmme3n1[3](S)
      2210 blocks super external:imsm

unused devices: <none>
```

When the rebuild has completed:

```
: active raid1 nvme2n1[2] nvme4n[1]
md2/0 [2/2] [UU]

: inactive nvme3n1[3](S)
imsm

: active raid1 nvme1n1[1](S) nvme0n1[0](S)
      1048576 blocks super external:/md1/0 [2/2] [UU]

: inactive nvme3n1[3](S)
      2210 blocks super external:imsm

unused devices: <none>
```

9 MDRAID Sysfs Components

The MDRAID subsystem has sysfs components that provide information or can be used to tweak behavior and performance. All MDRAID devices present in the system are shown in:

```
/sys/block/
```

Example:

```
# ls -l /sys/block/md*
lrwxrwxrwx 1 root root 0 May 17 13:26 /sys/block/md126 ->
../devices/virtual/block/md126
lrwxrwxrwx 1 root root 0 May 17 13:26 /sys/block/md127 ->
../devices/virtual/block/md127
```

Mapping between a device number and its name can be found:

```
# ls -l /dev/md/
total 0
lrwxrwxrwx 1 root root 8 May 17 13:26 imsm0 -> ../md127
lrwxrwxrwx 1 root root 8 May 17 13:26 raid1 -> ../md126
```

md127 is imsm0 and md126 is raid1.

MD devices in /sys/block are symbolic links pointing to the /sys/devices/virtual/block. All MD Devices are in the ‘md’ subdirectory in /sys/devices/virtual/block/mdXYZ directory. In the md directory the following contents can be found:

```
# ls -l /sys/devices/virtual/block/md127/md
total 0
-rw-r--r-- 1 root root 4096 May 18 13:18 array_size
-rw-r--r-- 1 root root 4096 May 17 13:26 array_state
drwxr-xr-x 2 root root    0 May 18 13:18 bitmap
-rw-r--r-- 1 root root 4096 May 18 13:18 chunk_size
-rw-r--r-- 1 root root 4096 May 18 13:18 component_size
drwxr-xr-x 2 root root    0 May 17 13:26 dev-nvme1n1
drwxr-xr-x 2 root root    0 May 17 13:26 dev-nvme2n1
-rw-r--r-- 1 root root 4096 May 18 13:18 layout
-rw-r--r-- 1 root root 4096 May 17 13:26 level
-rw-r--r-- 1 root root 4096 May 18 13:18 max_read_errors
-rw-r--r-- 1 root root 4096 May 17 13:26 metadata_version
--w------- 1 root root 4096 May 17 13:26 new_dev
-rw-r--r-- 1 root root 4096 May 17 13:26 raid_disks
-rw-r--r-- 1 root root 4096 May 18 13:18 reshape_position
-rw-r--r-- 1 root root 4096 May 18 13:18 resync_start
-rw-r--r-- 1 root root 4096 May 18 13:18 safe_mode_delay
```

Since the MD device is a container, the metadata version file will show:

```
# cat /sys/devices/virtual/block/md127/md/metadata_version
external:imsm
```

The directory contains subdirectories dev-nvme1n1 and dev-nvme2n1 specifying the disks that the container is assembled from.

The MD Volume contents look like below:

```
# ls -l /sys/devices/virtual/block/md126/md/
total 0
-rw-r--r-- 1 root root 4096 May 17 13:26 array_size
-rw-r--r-- 1 root root 4096 May 17 13:26 array_state
drwxr-xr-x 2 root root    0 May 18 13:10 bitmap
--w------- 1 root root 4096 May 18 13:10 bitmap_set_bits
-rw-r--r-- 1 root root 4096 May 17 13:26 chunk_size
-rw-r--r-- 1 root root 4096 May 17 13:26 component_size
-r--r--r-- 1 root root 4096 May 17 13:26 degraded
drwxr-xr-x 2 root root    0 May 17 13:26 dev-nvme1n1
drwxr-xr-x 2 root root    0 May 17 13:26 dev-nvme2n1
-rw-r--r-- 1 root root 4096 May 17 13:26 layout
-rw-r--r-- 1 root root 4096 May 17 13:26 level
-rw-r--r-- 1 root root 4096 May 18 13:10 max_read_errors
-rw-r--r-- 1 root root 4096 May 17 13:26 metadata_version
-r--r--r-- 1 root root 4096 May 18 13:10 mismatch_cnt
--w------- 1 root root 4096 May 17 13:26 new_dev
-rw-r--r-- 1 root root 4096 May 17 13:26 raid_disks
lrwxrwxrwx 1 root root    0 May 17 13:26 rd0 -> dev-nvme1n1
lrwxrwxrwx 1 root root    0 May 17 13:26 rd1 -> dev-nvme2n1
-rw-r--r-- 1 root root 4096 May 18 13:10 reshape_position
-rw-r--r-- 1 root root 4096 May 17 13:26 resync_start
-rw-r--r-- 1 root root 4096 May 17 13:26 safe_mode_delay
-rw-r--r-- 1 root root 4096 May 18 13:10 suspend_hi
-rw-r--r-- 1 root root 4096 May 18 13:10 suspend_lo
-rw-r--r-- 1 root root 4096 May 17 13:26 sync_action
-r--r--r-- 1 root root 4096 May 17 13:26 sync_completed
-rw-r--r-- 1 root root 4096 May 18 13:10 sync_force_parallel
-rw-r--r-- 1 root root 4096 May 18 13:10 sync_max
-rw-r--r-- 1 root root 4096 May 18 13:10 sync_min
-r--r--r-- 1 root root 4096 May 17 13:26 sync_speed
-rw-r--r-- 1 root root 4096 May 18 13:10 sync_speed_max
-rw-r--r-- 1 root root 4096 May 18 13:10 sync_speed_min
```

Several new files are present, and they are related to the RAID Volume properties. Base information can be read from files:

- Array size

```
# cat /sys/devices/virtual/block/md126/md/array_size
1048576
```

- Array state

```
# cat /sys/devices/virtual/block/md126/md/array_state
clean
```

- Raid level

```
# cat /sys/devices/virtual/block/md126/md/level
    raid1
```

- Strip size

```
# cat /sys/devices/virtual/block/md126/md/chunk_size
```

```
65536
```

- Metadata

```
# cat /sys/devices/virtual/block/md126/md/metadata_version
external:/md127/0
```

And this is what is shown in mdstat for the example RAID information:

```
# cat /proc/mdstat
Personalities : [raid1]
md127 : active raid1 nvme0n1[1] nvme1n1[0]
      78148256 blocks super external:/md0/0 [2/2] [UU]

md0 : inactive nvme1n1[1](S) nvme0n1 [0](S)
      2210 blocks super external:imsm

unused devices: <none>
```

10 LED Management

10.1 Intel® RSTe and Intel® VROC LED Management

LED management is achieved using the Linux 'ledmon' & 'ledctl' utilities. Normally drive backplane LEDs are controlled by a hardware RAID controller (PERC), but when using Software RAID on Linux (mdadm) for PCIE SSD, the 'ledmon' daemon will monitor the status of the drive array and update the status of drive LEDs.

'Ledmon' can be run as a daemon to constantly monitor the status of drives and Software RAID and set the drive LEDs appropriately. Only a single instance of the daemon should be running at a time. The 'ledmon' application supports two types of LED systems: A two-LED system (Activity LED and Status LED) and a three-LED system (Activity LED, Locate LED, and Fail LED). This tool has the highest priority when accessing the LEDs.

'ledmon' will be active and running on all modern installations of Linux using 'systemd'. Should there not be 'systemd' active on your installation, you will have to manually add 'ledmon' to the /etc/rc.local file (***see section 10.2.2***).

The 'ledctl' utility can be used to identify an individual drive on a backplane, useful when determining which drive maps to which drive bay slot. The ledctl application uses SGPIO and SES-2 to control LEDs. It implements IBPI patterns of SFF-8489 specification for SGPIO. The service monitors all RAID volumes. There is no method to specify individual volumes to monitor. These utilities have only been verified with Intel® storage controllers.

10.1.1 Installing ledmon Package

The following steps through the process for the user to follow on a standard Intel RSTe supported Linux distribution as a clean installation.

1. First check to see if 'ledmon' is properly installed and started. It should be running on all modern installations on Linux using 'systemd'
   a. If installed, this command will start the 'Ledmon' daemon if not already started.

```
# ledmon
```

   b. If it is running, the following is the expected return text, otherwise continue with #2 to install 'ledmon'.

```
ledmon [306623]: daemon is running…
ledmon [306623]: parent exit status is STATUS_LEDMON_RUNNING
```

2. To install the 'ledmon' package, execute the following command.

```
# yum install ledmon-0.79-3.el7.x86_64.rpm
```

3. If you are simply updating 'ledmon" from an existing older version, use the 'update' command.

```
# yum update ledmon
```

4. Confirm 'Ledmon' is started by repeating #1.

10.1.2 Configuring ledmon to Auto-Start

The following steps through the process for the user to follow on a standard Intel RSTe supported Linux distribution as a clean installation.

1. To ensure that the ledmon daemon starts on each reboot, open file /etc/rc.local using your favorite editor (e.g. VIM or VI).

```
# vi /etc/rc.local
```

2. Insert/add 'ledmon' to the final line of the file as shown below.

```
#!/bin/bash
# THIS FILE IS ADDED FOR COMPATIBILITY PURPOSES
#
# It is highly advisable to create own system services or udev rules
# to run scripts during boot instead of using this file.
#
# In contrast to previous versions due to parallel execution during boot
# this script will be executed during boot.
#
# Please note that you must run 'chmod +x /etc/rc.local' to ensure
# that this script will be executed during boot.
#
touch /var/lock/subsys/local
ledmon
```

Note: *It is important that the addition of ledmon is located on the next line of the /etc/rc.local file. It is to reside by itself within that file on that line. Failure to configure this setting as such can cause the system to not function properly.*

3. Save the file and quit/exit the editor.

Note: For added flexibility, ledmon can be run with the following options listed in the below table.

Option	Usage
-c or --config-path=	Sets the configuration file path. This overrides any other configuration files. (Although the utility currently does not use a config file). The /etc/ledcfg.conf is shared by ledmon and ledctl utilities.
-l or --log-path	Sets the path to a log file. This overrides /var/log/ledmon.log.
-t or --interval=	Sets the time interval in seconds between scans of the sysfs. A minimum of 5 seconds is set.

--quiet, --error, ---warning, --info, --debug, --all	Specifies verbosity level of the log - 'quiet' means no logging at all, and 'all' means to log everything. The levels are given in order. If user specifies more than one verbose option the last option comes into effect.
-h or --help	Prints help text and exits.
-v or --version	Prints version and license information, then exits.

10.1.3 Use of ledmod/ledctl Utilities

1. Running ‘ledctl’ and ‘ledmon’ can be done concurrently, however, ‘ledmon’ application has the highest priority when accessing LEDs then other programs. It means some patterns set by ‘ledctl’ may have no effect (except Locate pattern). If not already started, start the ‘ledmon’ as shown below.

```
 # ledmon
```

2. As an example, use the ‘ledctl’ command below to blink the drive LED on the device node ‘/dev/nvme0n1’.

```
# ledctl locate=/dev/nvme0n1
```

   Note: This command can be used as an on demand tool from the command line to “locate” a drive at will. Each drive attached to the system particularly via surprise hot plug will be re-enumerated and this feature can be useful in identification. The numbers assigned are set as a feature of the Linux kernel, and cannot be reset by the administrator.

3. The following command will turn off the locate LED

```
# ledctl locate_off=/dev/nvme0n1
```

   Note: This feature can also be turned off at will by the system administrator. This is the command by which the administrator may accomplish this.