Intel® Virtual RAID on CPU (Intel® VROC) for Linux*

User Guide

Revision 013

July 2024

Document Number: 333915-013

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis. You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel® products described herein. You agree to grant Intel® a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel® representative to obtain the latest Intel® product specifications and roadmaps.

All product plans and roadmaps are subject to change without notice.

The products described may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel® technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at Intel®.com.

Intel® disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Altering clock frequency, voltage, or memory interface speeds may void any product warranties and reduce stability, security, performance, and life of the processor and other components. Intel has not validated processor running memory above Plan-Of-Record (POR) speed. DRAM/DIMM devices should support desired speed, check with DRAM/DIMM vendors for details. System manufacturers are responsible for all validation and assume the risk of any stability, security, performance, or other functional issues resulting from such alterations.

Contents

Figures

Tables

Revision History

Document Number	Revision	Description	Date
333915	001	Initial release	March 2016
	002	Updated for Intel® VROC/RSTe 5.0 release	March 2017
	003	Updated for Intel® VROC/RSTe 5.1 release	April 2017
	004	Updated for Intel® VROC/RSTe 5.3 release	October 2017
	005	Updated for Intel® VROC/RSTe 5.4 release	March 2018
	006	Updated for Intel® VROC 6.0 release	December 2018
	007	Updated for Intel® VROC 6.3 release	November 2019
	008	Updated for Intel® VROC 7.0 release	June 2020
	009	Updated for Intel® VROC 7.5 release	December 2020
	010	Updated for Intel® VROC 8.0 release	November 2022
	011	Revision Update Only. No Content Change	April 2023
	012	Updated for Intel® VROC 8.2 release	November 2023
	013	Updated for Intel® VROC 9.0 release	July 2024

§§

1 Introduction

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

This document describes how to use this application and many of the options offered to setup, manage, and monitor Intel® VROC RAID.

Note: The information in this document is only relevant on systems with a supported Intel® chipset, with a supported operating system. Ensure that your platform and operating system is properly configured to support Intel® VROC.

Note: The majority of the information in this document is related to software configuration. 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.

Note: This user guide is not an *mdadm* user guide, it will only focus on those commands that are used to support Intel® VROC.

The Intel® VROC provides enterprise RAID support for both NVMe SSDs and SATA devices for enterprise servers, workstations and some high-end desktops based on Intel® Xeon® processors.

- Intel® VROC (VMD NVMe RAID): This product provides an enterprise RAID solution on platforms that support the Intel® VMD technology.
- Intel® VROC (SATA RAID): This product provides an enterprise RAID solution for SATA devices connected to SATA controllers on the Intel® Platform Control Hub (PCH) configured for RAID mode.

1.1 Reference OEM Platform Documentation

Refer to your OEM for a full list of available feature sets. If any of the information in this document conflicts with the support information provided by the platform OEM, the platform documentation and configurations take precedence.

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

1.2 Supported RAID Volumes

Intel® VROC product provides high-performance NVMe RAID capabilities and allows you to create RAID arrays and spanned volumes. Another advanced feature of Intel® VROC is the ability to roam an array between Linux* and Microsoft* Windows host systems.

Intel® VROC for Linux* supports the following RAID levels:

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

1.2.1 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 1-1. RAID 0 Overview

Drives Supported	2 minimum
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. 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.

1.2.2 RAID 1 (Mirroring)

RAID 1 arrays contain two drives where the data is copied to both of the drives 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 1-2. RAID 1 Overview

Drives Supported	2 maximum
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 the capacity of one disk is sufficient and for any application(s) requiring very high availability.

1.2.3 RAID 5 (Striping with Parity)

RAID 5 arrays contain three (minimum) 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 1-3. RAID 5 Overview

Drives Supported	3 minimum
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. As with RAID 0 Striping, although RAID 5 can be scaled to many drives there is a performance sweet spot specific to your implementation.

1.2.4 RAID 10

A RAID 10 array uses four (minimum) or more 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 up to 2 drives to fail while still maintaining access to data.

Note: Double degradation (2 drives failures) support is limited to cases where drives from opposite mirrored set are failed.

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

Table 1-4. RAID 10 Overview

Drives Supported	4 minimum
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.

1.2.5 Intel® Matrix RAID

The Intel® VROC package provides high-performance NVMe RAID capabilities and allows you to create RAID arrays and spanned volumes. Matrix arrays indicate there are up to two RAID volumes in a single RAID container. A container is a collection of devices that are managed as a set. Within a container, there is one set of metadata that describes all the arrays in the container. Intel® Matrix Storage Manager (IMSM) metadata is used to support Intel® Matrix RAID. An important requirement for Matrix RAID is that the volumes within the container span the same set of member drives.

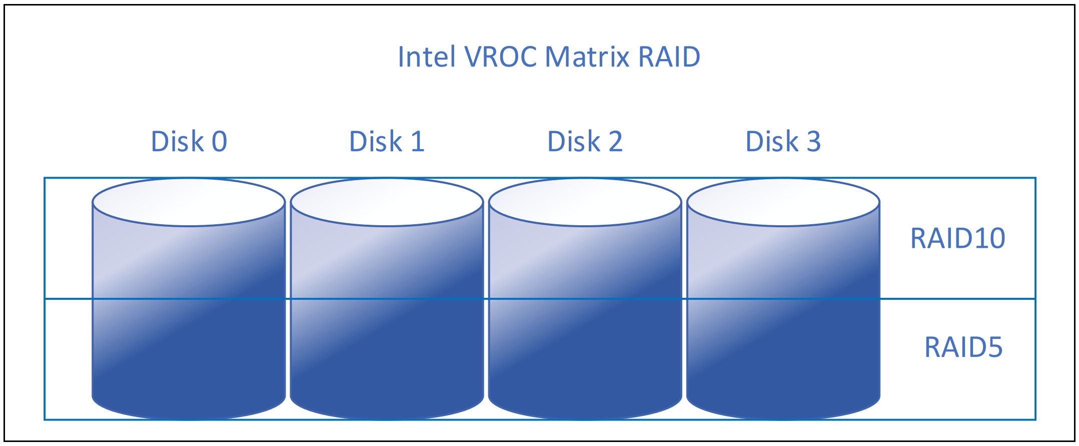
Intel® VROC Matrix RAID allows user to:

1. Fully utilize the RAID member disks' capacity (second array can be used for other application's data storage, otherwise the space could be unused).
2. Use combination of different RAID levels to meet different requirements of redundancy and performance.

Table 1-5. Intel® Matrix RAID Overview

Drives Supported	Same count of drives per each RAID array. Used levels must support the count.
Advantage	Combination of two RAID arrays.
Fault-tolerance	Individual for each level used inside.
Application	Special user configuration.

Figure 1-1. Intel® VROC Matrix RAID



As an example, in the above diagram, on an Intel® VROC enabled system, Intel® VROC Matrix RAID allows you to create both a RAID 10 volume as well as a RAID 5 volume across four drives. Refer to Section 4.2.5 Additional INTEL® VROC RAID Volume Creation Examples to learn how to create Intel® Matrix RAID.

1.3 Supported Linux* OS Distributions

Intel® VROC Linux* driver and tools are open source. It is up to specific Operating System Vendors (OSVs) to pull in Intel® VROC features and patches from upstream Linux* community and integrate into their Linux* OS distributions. For most of Linux* OS distributions, the inbox kernel driver and tools can support Intel® VROC functions well and there is no additional software download required. For some Linux* OS distributions in order to support new generation platforms, it requires an out-of-box Intel® VROC Linux* driver package. For detailed Linux* OS distributions with inbox or out-of-box Intel® VROC support, refer to the Intel® Virtual RAID on CPU (Intel® VROC) Supported Configurations.

1.4 Terminology

Table 1-6. Terminology

Term	Description
API	Application Programming Interface
BIOS	Basic Input/Output System
Array	This term is representative of a *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 or other non-native metadata.
GB	Gigabyte
GiB	Gibibyte (1024 x 1024 x 1024 bytes)
GA	General Access – Operating System release package fully validated by Red Hat Linux*.
HII	Human Interface Infrastructure
Hot-Plug	The unannounced removal and insertion of a drive while the system is powered on.
I/O	Input/Output
Initramfs	Initial RAM File System
IMSM	Intel® Matrix Storage Manager
KB	Kilobyte
KiB	Kibibyte (1024 bytes)
Left-Symmetric	Default layout scheme for RAID 5 configurations. Not supported with IMSM metadata
Left-Asymmetric	Parity bit layout scheme used in RAID 5 configurations.
Matrix RAID	Two different RAID volumes within a single RAID array container.
MB	Megabyte
MiB	Mebibyte (1024 x 1024 bytes)
MD	Linux* kernel Multiple Device driver
Member	A SATA or NVMe drive used within a RAID array.

Term	Description
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 SATA and NVMe SSDs.
NVMe	Non-volatile Memory Express
OS	Operating System
OSV	Operating System Vendor (e.g., Red Hat, SUSE)
Pre-OS	A BIOS component to configure Intel® VROC RAID.
RAID	Redundant Array of Independent Disks: allows data to be distributed across multiple drives to provide data redundancy or to enhance data storage performance.
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.
Hot Spare Drive	Hot spare drive is a disk or group of disks used to automatically or manually, depending upon the hot spare policy, replace a failing or failed disk in a RAID configuration.
RHEL	Red Hat Enterprise Linux*
Intel® RSTe	Intel® Rapid Storage Technology enterprise.
RWH	It stands for RAID5 Write Hole and can cause data integrity issue

Term	Description
SLES	SUSE Linux* Enterprise Server
TB	Terabyte
TiB	Tebibyte (1024 x 1024 x 1024 x 1024 bytes)
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.
Volume initialization	Immediately after a RAID volume has been created, initialization (or resync) commences if the RAID level is 1, 10, or 5 to guarantee volume data integrity
Intel® VROC	Intel® Virtual RAID on CPU
Intel® VROC IC	Intel® VROC integrated Caching

§§

2 Manage Intel® VROC RAID in UEFI HII

Intel® VROC has the capability to manage RAID volumes through platform UEFI HII. A platform BIOS integrated with Intel® VROC PreOS/UEFI driver is able to create a RAID volume for OS installation, delete a RAID volume, add a hot-spare drive, turn on/off locate LED, etc.

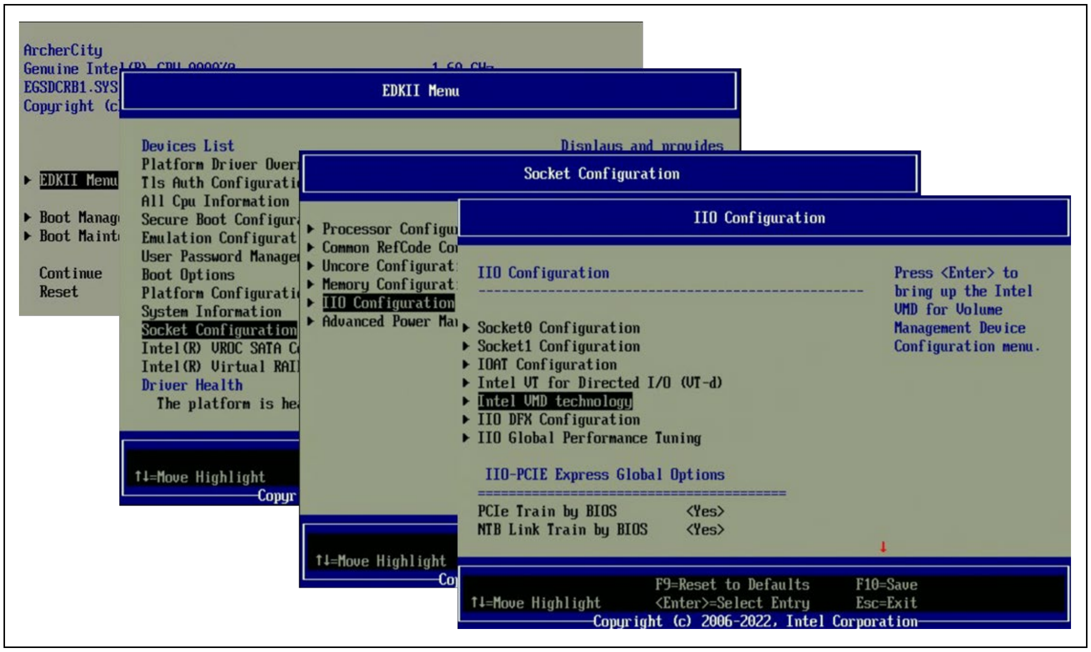
This section illustrates how to enable and create Intel® VROC RAID volumes with instructions and steps based on Intel® Customer Reference Board (CRB). These instructions may differ from platform vendors. Refer to the user guide or instructions supplied by the platform vendor.

2.1 Enabling Intel® VROC (NVMe RAID) in UEFI HII

Intel® VROC (NVMe RAID) is based on Intel® VMD hardware inside Intel® Xeon based processors. Intel® VMD needs to be enabled on the platform BIOS. The following steps illustrate how to enable Intel® VMD in the Intel® CRB UEFI HII.

1. Immediately following POST, select the option that will allow the user to access the BIOS setup menu. This example uses <F2>.
2. For the Intel® CRB reference BIOS, the user will want to use the arrow keys to move the cursor to the **EDKII Menu** (it will become highlighted) and press <Enter>.
3. Using the arrow keys, move the cursor to **Socket Configuration** and press <Enter>.
4. Using the arrow keys, move the cursor to **IIO Configuration** and press <Enter>.
5. Using the arrow keys, move the cursor to **Intel® VMD technology** and press <Enter>.

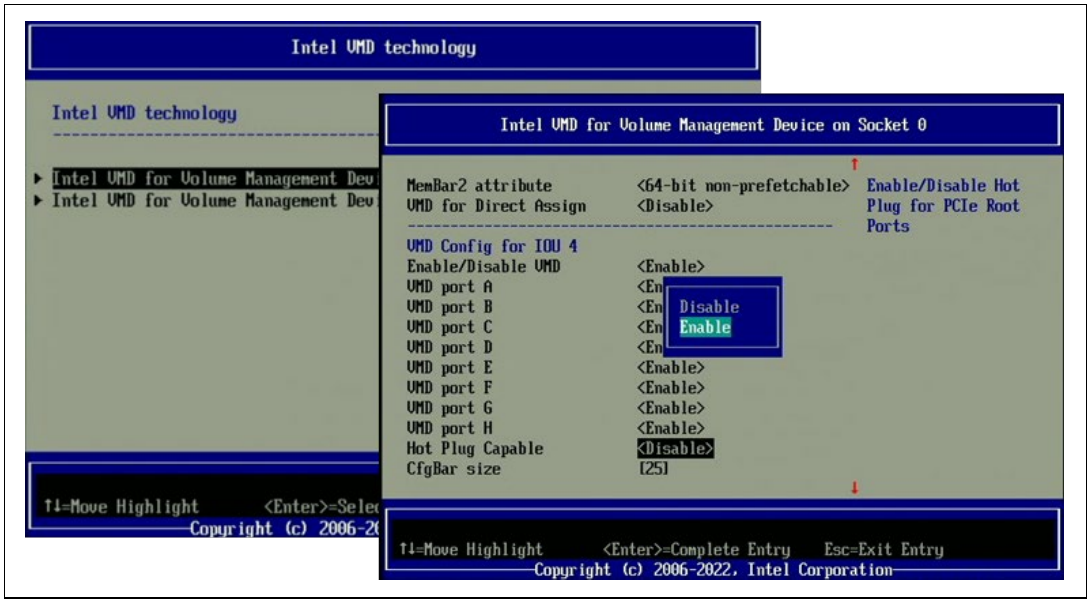
Figure 2-1. Intel® VMD Technology BIOS Setting



```
IIO Configuration

IIO Configuration                                        Press <Enter> to
-------------------------------------------------        bring up the Intel
                                                         VMD for Volume
▸ Socket0 Configuration                                  Management Device
▸ Socket1 Configuration                                  Configuration menu.
▸ IOAT Configuration
▸ Intel VT for Directed I/O (VT-d)
▸ Intel VMD technology
▸ IIO DFX Configuration
▸ IIO Global Performance Tuning

  IIO-PCIE Express Global Options
  ========================================
PCIe Train by BIOS        <Yes>
NTB Link Train by BIOS    <Yes>

                        F9=Reset to Defaults     F10=Save
↑↓=Move Highlight       <Enter>=Select Entry     Esc=Exit
              Copyright (c) 2006-2022, Intel Corporation
```

6. Using the arrow keys, move the cursor to **Intel VMD for Volume Management Device on Socket 0** and press <Enter>.
7. Select the correct **IOUs#** from which NVMe SSDs are attached. In this example, IOU 4 is selected. Enable **VMD, VMD port A-H**, and **Hot Plug capable**, check **MemBar1** size is **[26]** or above.

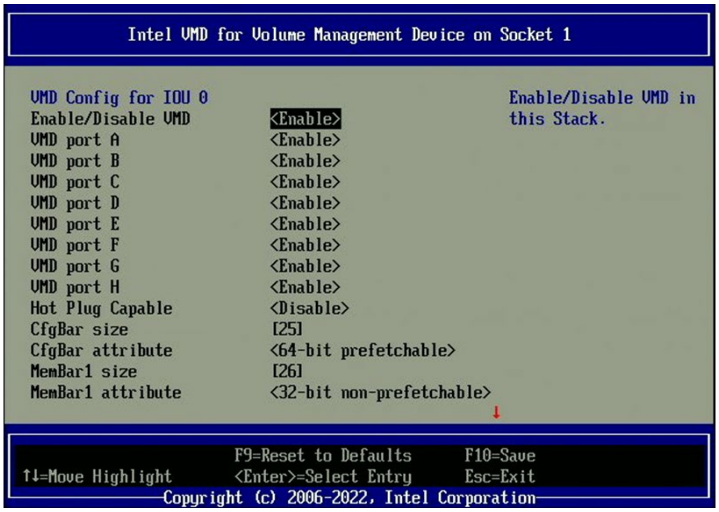
Figure 2-2. Intel® VMD Ports Configuration for Socket 0


Intel VMD technology
Intel VMD technology
Intel VMD for Volume Management Dev
Intel VMD for Volume Management Dev
↑↓=Move Highlight <Enter>=Selec
Copyright (c) 2006-20
Disable
Enable

```
Intel VMD for Volume Management Device on Socket 0

MemBar2 attribute         <64-bit non-prefetchable>   Enable/Disable Hot
VMD for Direct Assign     <Disable>                   Plug for PCIe Root
-------------------------------------------------     Ports
VMD Config for IOU 4
Enable/Disable VMD        <Enable>
VMD port A                <En
VMD port B                <En
VMD port C                <En
VMD port D                <En
VMD port E                <Enable>
VMD port F                <Enable>
VMD port G                <Enable>
VMD port H                <Enable>
Hot Plug Capable          <Disable>
CfgBar size               [25]

↑↓=Move Highlight     <Enter>=Complete Entry     Esc=Exit Entry
              Copyright (c) 2006-2022, Intel Corporation
```

8. Press <Esc> and then move to **Intel VMD for Volume Management Device on Socket 1.** Do the same enabling of VMD functions on the corresponding IOU# ports. In this example, IOU 0 is enabled with VMD function on all the A-H ports.

Figure 2-3. Intel® VMD Ports Configuration for Socket 1



```
           Intel VMD for Volume Management Device on Socket 1

VMD Config for IOU 0                                  Enable/Disable VMD in
Enable/Disable VMD           <Enable>                 this Stack.
VMD port A                   <Enable>
VMD port B                   <Enable>
VMD port C                   <Enable>
VMD port D                   <Enable>
VMD port E                   <Enable>
VMD port F                   <Enable>
VMD port G                   <Enable>
VMD port H                   <Enable>
Hot Plug Capable             <Disable>
CfgBar size                  [25]
CfgBar attribute             <64-bit prefetchable>
MemBar1 size                 [26]
MemBar1 attribute            <32-bit non-prefetchable>
                                                  ↓

                         F9=Reset to Defaults      F10=Save
↑↓=Move Highlight        <Enter>=Select Entry      Esc=Exit
               Copyright (c) 2006-2022, Intel Corporation
```

9. Save settings by pressing <F10> to **Save Changes and Exit**.

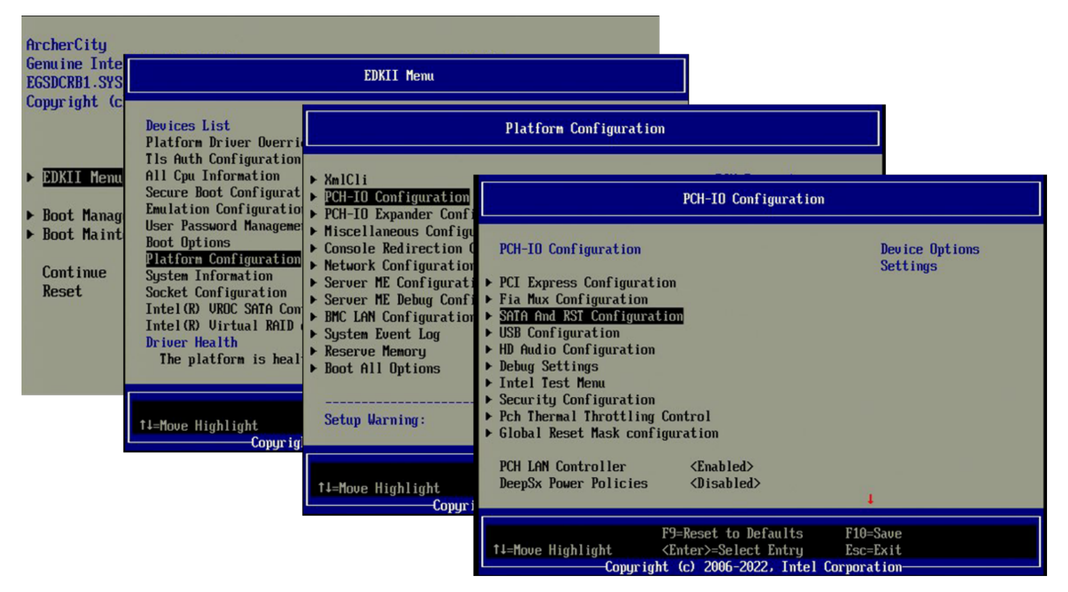
Note: Consult the user's platform BIOS manufacturer documentation for a complete list of options that can be configured.

2.2 Enabling Intel® VROC (SATA RAID) in UEFI HII

Intel® VROC (SATA RAID) is based on Intel® PCH SATA controllers. Intel® SATA controllers need to be enabled and configured as RAID mode on the platform BIOS. The following steps illustrate how to enable Intel® PCH SATA controllers in the Intel® CRB UEFI HII.

1. Immediately following POST, select the option that will allow the user to access the BIOS setup menu. This example uses <F2>.
2. For the Intel® CRB reference BIOS, the user will want to use the arrow keys to move the cursor to the EDKII Menu (it will become highlighted) and press <Enter>.
3. Using the arrow keys, move the cursor to Platform Configuration and press <Enter>.
4. Using the arrow keys, move the cursor to PCH-IO Configuration and press <Enter>.
5. Using the arrow keys, move the cursor to SATA And RST Configuration and press <Enter>.

Figure 2-4. SATA and RST Configuration BIOS Setting



```
ArcherCity
Genuine Inte
EGSDCRB1.SYS
Copyright (c

▶ EDKII Menu

▶ Boot Manag
▶ Boot Maint

  Continue
  Reset
```

```
EDKII Menu

Devices List
Platform Driver Overri
Tls Auth Configuration
All Cpu Information
Secure Boot Configurat
Emulation Configuratio
User Password Manageme
Boot Options
Platform Configuration
System Information
Socket Configuration
Intel(R) VROC SATA Con
Intel(R) Virtual RAID
Driver Health
  The platform is heal

↑↓=Move Highlight
            Copyrig
```

```
Platform Configuration

▶ XmlCli
▶ PCH-IO Configuration
▶ PCH-IO Expander Conf
▶ Miscellaneous Configu
▶ Console Redirection
▶ Network Configuration
▶ Server ME Configurati
▶ Server ME Debug Confi
▶ BMC LAN Configuration
▶ System Event Log
▶ Reserve Memory
▶ Boot All Options

--------------------
Setup Warning:

↑↓=Move Highlight
              Copyri
```

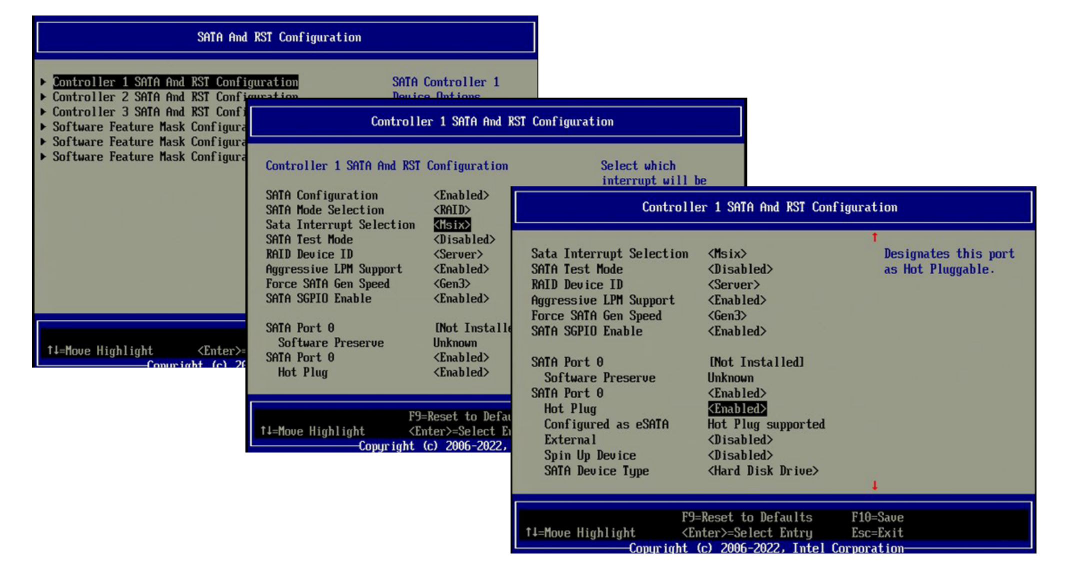
```
PCH-IO Configuration

PCH-IO Configuration                                       Device Options
                                                           Settings
▶ PCI Express Configuration
▶ Fia Mux Configuration
▶ SATA And RST Configuration
▶ USB Configuration
▶ HD Audio Configuration
▶ Debug Settings
▶ Intel Test Menu
▶ Security Configuration
▶ Pch Thermal Throttling Control
▶ Global Reset Mask configuration

PCH LAN Controller          <Enabled>
DeepSx Power Policies       <Disabled>
                                                      ↓
                         F9=Reset to Defaults      F10=Save
↑↓=Move Highlight        <Enter>=Select Entry      Esc=Exit
              Copyright (c) 2006-2022, Intel Corporation
```

6. Using the arrow keys, move the cursor to Controller 1 SATA And RST Configuration and press <Enter>.

Note: In this example, Controller 1 among 3 SATA controllers on Intel® CRB is used. It is with the same steps to enable SATA RAID on other two controllers.

7. Change SATA Mode Selection to <RAID>, Sata Interrupt Selection to <Msix>, SATA SGPIO Enable to <Enable>. Within the same page, change SATA Port 0~7's Hot Plug to <Enabled>.

Figure 2-5. SATA Ports Configuration



```
SATA And RST Configuration

▶ Controller 1 SATA And RST Configuration        SATA Controller 1
▶ Controller 2 SATA And RST Configuration        Device Options
▶ Controller 3 SATA And RST Confi
▶ Software Feature Mask Configura
▶ Software Feature Mask Configura
▶ Software Feature Mask Configura

↑↓=Move Highlight        <Enter>=
           Copyright (c) 20
```

```
Controller 1 SATA And RST Configuration

Controller 1 SATA And RST Configuration              Select which
                                                     interrupt will be
SATA Configuration              <Enabled>
SATA Mode Selection             <RAID>
Sata Interrupt Selection        <Msix>
SATA Test Mode                  <Disabled>
RAID Device ID                  <Server>
Aggressive LPM Support          <Enabled>
Force SATA Gen Speed            <Gen3>
SATA SGPIO Enable               <Enabled>

SATA Port 0                     [Not Installe
  Software Preserve             Unknown
SATA Port 0                     <Enabled>
  Hot Plug                      <Enabled>

                         F9=Reset to Defau
↑↓=Move Highlight        <Enter>=Select En
            Copyright (c) 2006-2022,
```

```
Controller 1 SATA And RST Configuration
                                                     ↑
Sata Interrupt Selection      <Msix>                   Designates this port
SATA Test Mode                <Disabled>               as Hot Pluggable.
RAID Device ID                <Server>
Aggressive LPM Support        <Enabled>
Force SATA Gen Speed          <Gen3>
SATA SGPIO Enable             <Enabled>

SATA Port 0                   [Not Installed]
  Software Preserve           Unknown
SATA Port 0                   <Enabled>
  Hot Plug                    <Enabled>
  Configured as eSATA         Hot Plug supported
  External                    <Disabled>
  Spin Up Device              <Disabled>
  SATA Device Type            <Hard Disk Drive>
                                                     ↓
                         F9=Reset to Defaults      F10=Save
↑↓=Move Highlight        <Enter>=Select Entry      Esc=Exit
              Copyright (c) 2006-2022, Intel Corporation
```

8. Save settings by pressing <F10> to Save Changes and Exit.

2.3 Creating Intel® VROC NVMe RAID Volume in UEFI HII

This section illustrates how to create an Intel® VROC NVMe RAID volume in the Intel® CRB UEFI HII.

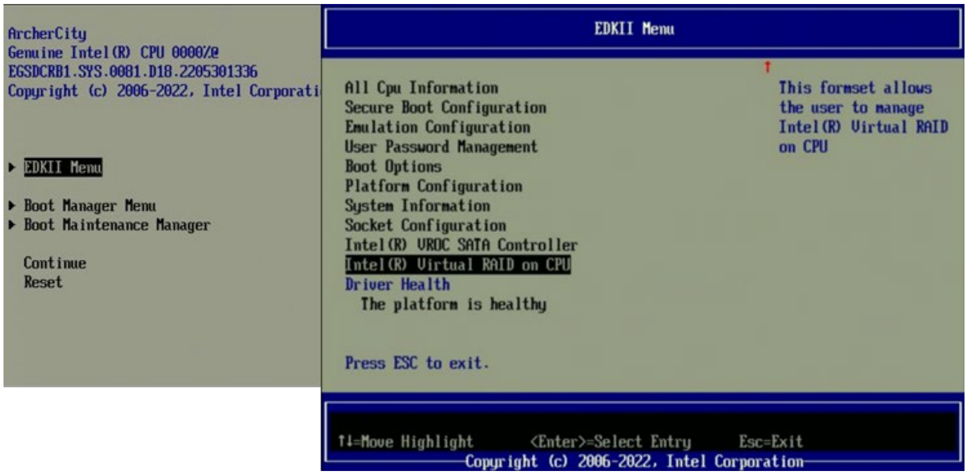
The following assumptions have been made:

- You have enabled Intel® VMD within system BIOS.
- You have sufficient drives of the appropriate size and type to create the RAID volume.
- The drives are connected to the system properly per your vendor's specifications.
- The Pre-OS is able to see all drives.

To create a RAID volume:

1. To enter the Setup Menu, press the appropriate key indicated on the screen during boot up. This example uses <F2>.
2. Navigate to **EDKII Menu** and press <Enter>.
3. Navigate to **Intel Virtual RAID on CPU (Intel® VROC)** and press <Enter>.

Figure 2-6. Intel Virtual RAID on CPU BIOS Setting

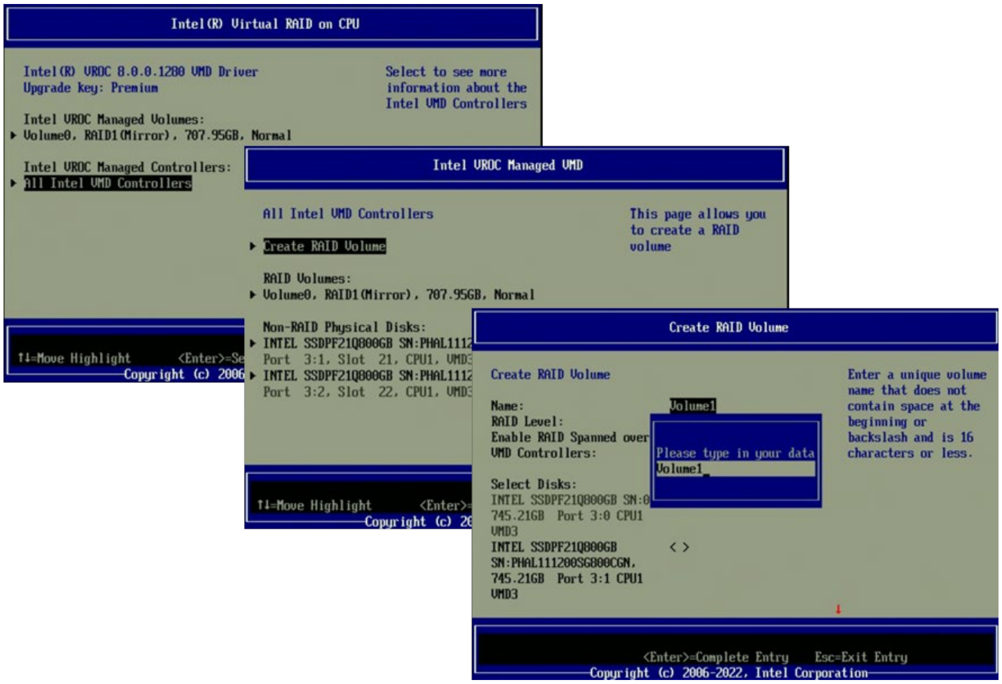


```
ArcherCity
Genuine Intel(R) CPU 0000%@
EGSDCRB1.SYS.0081.D18.2205301336
Copyright (c) 2006-2022, Intel Corporati

▸ EDKII Menu

▸ Boot Manager Menu
▸ Boot Maintenance Manager

  Continue
  Reset
```

```
                         EDKII Menu

All Cpu Information                          This formset allows
Secure Boot Configuration                    the user to manage
Emulation Configuration                      Intel(R) Virtual RAID
User Password Management                     on CPU
Boot Options
Platform Configuration
System Information
Socket Configuration
Intel(R) VROC SATA Controller
Intel(R) Virtual RAID on CPU
Driver Health
  The platform is healthy


Press ESC to exit.

↑↓=Move Highlight      <Enter>=Select Entry      Esc=Exit
            Copyright (c) 2006-2022, Intel Corporation
```

4. Navigate to **ALL Intel VMD Controllers** and press <Enter>, then select **Create RAID Volume** and press <Enter>.
5. Press the <Enter> key to type in a **volume name** or move down the cursor to accept the default name.

Figure 2-7. Create NVMe RAID Volume Option in BIOS



```
Intel(R) Virtual RAID on CPU

Intel(R) VROC 8.0.0.1280 VMD Driver          Select to see more
Upgrade key: Premium                         information about the
                                             Intel VMD Controllers
Intel VROC Managed Volumes:
▸ Volume0, RAID1(Mirror), 707.95GB, Normal

Intel VROC Managed Controllers:
▸ All Intel VMD Controllers

↑↓=Move Highlight      <Enter>=Se
            Copyright (c) 2006
```

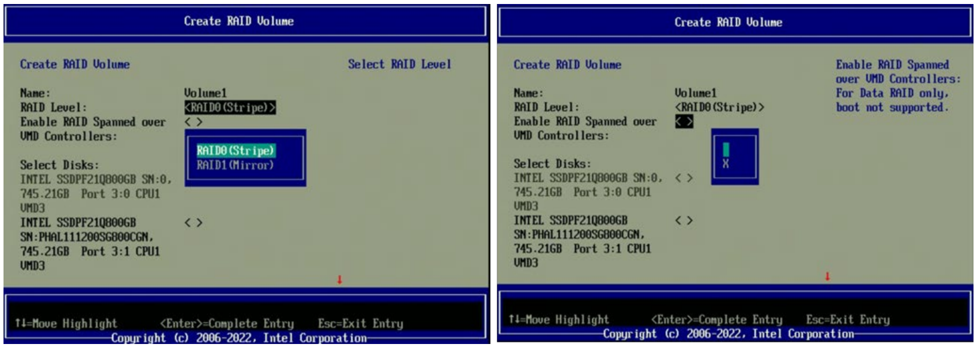
```
Intel VROC Managed VMD

All Intel VMD Controllers                    This page allows you
                                             to create a RAID
▸ Create RAID Volume                         volume

RAID Volumes:
▸ Volume0, RAID1(Mirror), 707.95GB, Normal

Non-RAID Physical Disks:
▸ INTEL SSDPF21Q800GB SN:PHAL1112
  Port 3:1, Slot 21, CPU1, VMD3
▸ INTEL SSDPF21Q800GB SN:PHAL1112
  Port 3:2, Slot 22, CPU1, VMD3

↑↓=Move Highlight      <Enter>=
            Copyright (c) 20
```

```
Create RAID Volume

Create RAID Volume                                 Enter a unique volume
                                                   name that does not
Name:                      Volume1                 contain space at the
RAID Level:                                        beginning or
Enable RAID Spanned over                           backslash and is 16
VMD Controllers:           Please type in your data  characters or less.
                           Volume1_
Select Disks:
INTEL SSDPF21Q800GB SN:0
745.21GB  Port 3:0 CPU1
VMD3
INTEL SSDPF21Q800GB        < >
SN:PHAL111200SG800CGN,
745.21GB  Port 3:1 CPU1
VMD3

      <Enter>=Complete Entry    Esc=Exit Entry
        Copyright (c) 2006-2022, Intel Corporation
```

6. Press the <Enter> key, then select the **RAID level** by pressing the <Enter> key and using the arrow keys to scroll through the available values. Highlight the desired RAID level and press <Enter> to set the RAID level.
7. Only data volumes are supported in this configuration to **Enable RAID Spanned over VMD controllers**, boot volumes are not supported.

   To enable spanned volumes, use the arrow key to highlight the < > bracket and press <Enter>. This will open a small selection menu. Navigate the cursor to the X and press <Enter> to enable volume spanning. To disable, you would set the value back to blank and press <Enter> to save the value.

Figure 2-8. NVMe RAID Volume Options in BIOS

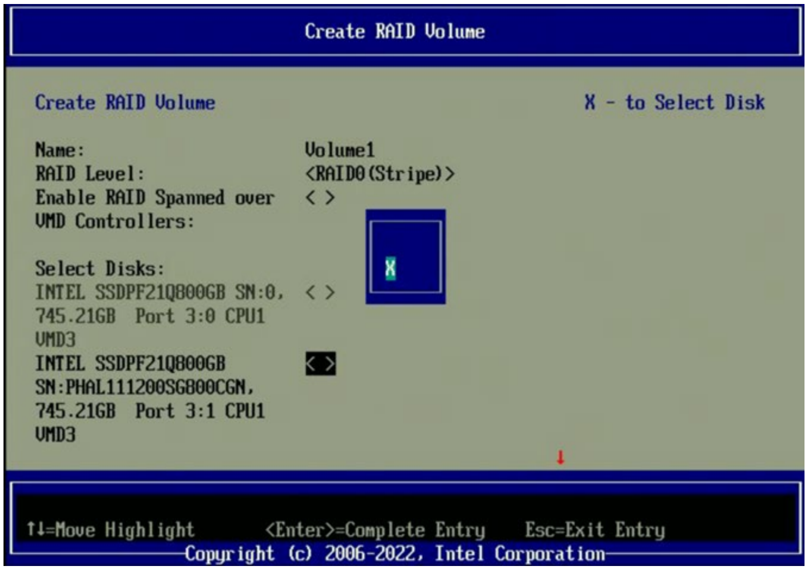


```
Create RAID Volume

Create RAID Volume                              Select RAID Level

Name:                      Volume1
RAID Level:                <RAID0(Stripe)>
Enable RAID Spanned over   < >
VMD Controllers:
                            RAID0(Stripe)
Select Disks:               RAID1(Mirror)
INTEL SSDPF21Q800GB SN:0,
745.21GB  Port 3:0 CPU1
VMD3
INTEL SSDPF21Q800GB        < >
SN:PHAL111200SG800CGN,
745.21GB  Port 3:1 CPU1
VMD3

↑↓=Move Highlight    <Enter>=Complete Entry    Esc=Exit Entry
           Copyright (c) 2006-2022, Intel Corporation
```

```
Create RAID Volume

Create RAID Volume                              Enable RAID Spanned
                                                over VMD Controllers:
Name:                      Volume1              For Data RAID only,
RAID Level:                <RAID0(Stripe)>      boot not supported.
Enable RAID Spanned over   < >
VMD Controllers:
                                 X
Select Disks:
INTEL SSDPF21Q800GB SN:0,  < >
745.21GB  Port 3:0 CPU1
VMD3
INTEL SSDPF21Q800GB        < >
SN:PHAL111200SG800CGN,
745.21GB  Port 3:1 CPU1
VMD3

↑↓=Move Highlight    <Enter>=Complete Entry    Esc=Exit Entry
           Copyright (c) 2006-2022, Intel Corporation
```

8. Press the <Enter> key. Using the arrow keys, select the drives one by one by highlighting the < > bracket on the line next to the drive’s port number. Press <Enter> to open the selection menu which will be set to blank or off status. Navigate to highlight the X and press <Enter> to include the drive within the array.
9. Repeat Step 8 for each drive required within this array.

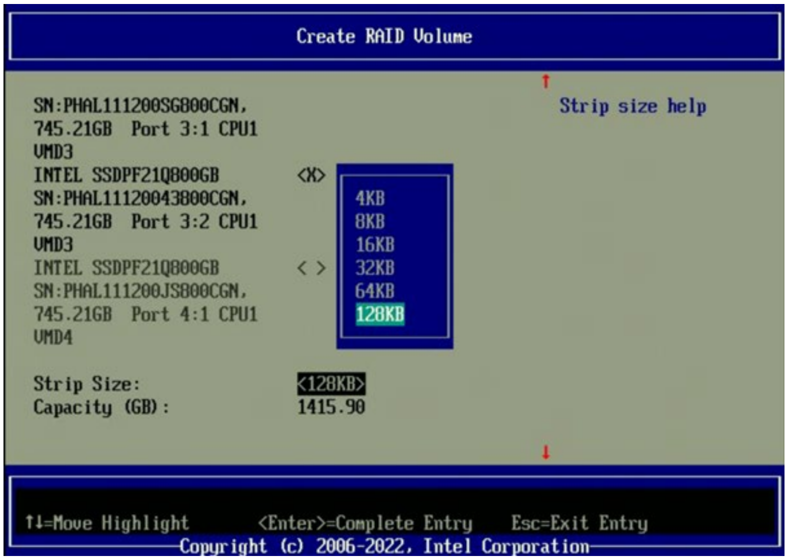
Figure 2-9. Drives Selection for NVMe RAID Volume Creation



```
                         Create RAID Volume

Create RAID Volume                                  X - to Select Disk

Name:                        Volume1
RAID Level:                  <RAID0(Stripe)>
Enable RAID Spanned over     < >
VMD Controllers:
                                      X
Select Disks:
INTEL SSDPF21Q800GB SN:0,    < >
745.21GB  Port 3:0 CPU1
VMD3
INTEL SSDPF21Q800GB          < >
SN:PHAL111200SG800CGN,
745.21GB  Port 3:1 CPU1
VMD3

↑↓=Move Highlight      <Enter>=Complete Entry     Esc=Exit Entry
                Copyright (c) 2006-2022, Intel Corporation
```

10. Unless the user has selected a RAID 1, select the strip size by using the arrow keys and press <Enter> to open the options menu. Utilize the arrow keys to select the desired strip size and press <Enter> to save the value.

Figure 2-10. Strip Size Selection for NVMe RAID Volume Creation

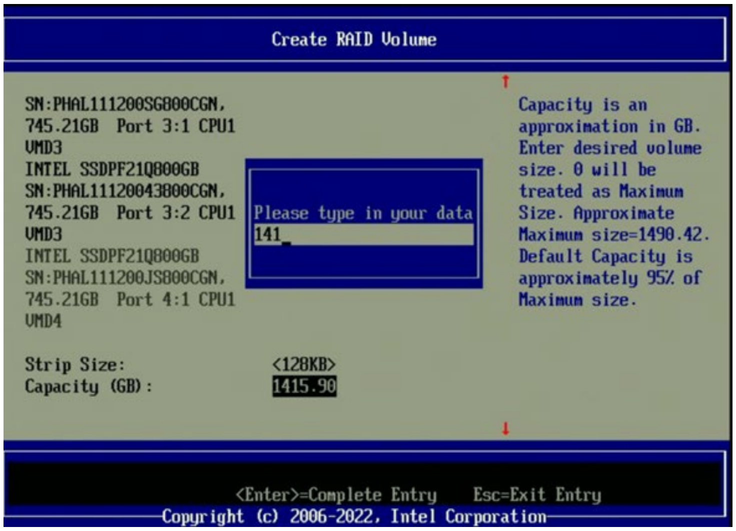


```
                         Create RAID Volume

SN:PHAL111200SG800CGN,                              Strip size help
745.21GB  Port 3:1 CPU1
VMD3
INTEL SSDPF21Q800GB          <X>
SN:PHAL11120043800CGN,            4KB
745.21GB  Port 3:2 CPU1           8KB
VMD3                              16KB
INTEL SSDPF21Q800GB          < >  32KB
SN:PHAL111200JS800CGN,            64KB
745.21GB  Port 4:1 CPU1           128KB
VMD4

Strip Size:                  <128KB>
Capacity (GB):               1415.90

↑↓=Move Highlight      <Enter>=Complete Entry     Esc=Exit Entry
                Copyright (c) 2006-2022, Intel Corporation
```

11. Select the volume capacity and press the <Enter> key. The default value will be displayed as the maximum capacity available with the drives selected. The value is calculated in bytes. A 700GB drive would use the following math: 700 * 1024 = 716000.

Note: Unless specifically selected, the default volume capacity will be calculated based on 95% of the available space of the smallest physical disk. This allows for the variances in the physical disk capacities from different vendors. This feature is called *Disk Coercion* which is a common feature of most RAID solutions.

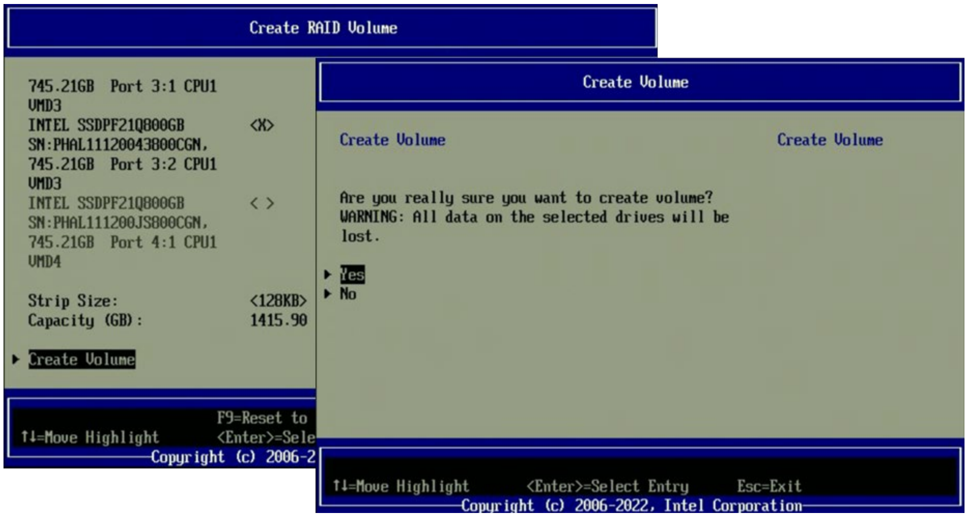
Figure 2-11. Volume Capacity Selection for NVMe RAID Volume Creation



```
Create RAID Volume

SN:PHAL111200SG800CGN,                                  Capacity is an
745.21GB  Port 3:1 CPU1                                 approximation in GB.
VMD3                                                    Enter desired volume
INTEL SSDPF21Q800GB                                     size. 0 will be
SN:PHAL11120043800CGN,                                  treated as Maximum
745.21GB  Port 3:2 CPU1   Please type in your data      Size. Approximate
VMD3                      141_                          Maximum size=1490.42.
INTEL SSDPF21Q800GB                                     Default Capacity is
SN:PHAL111200JS800CGN,                                  approximately 95% of
745.21GB  Port 4:1 CPU1                                 Maximum size.
VMD4

Strip Size:               <128KB>
Capacity (GB):            1415.90

            <Enter>=Complete Entry     Esc=Exit Entry
        Copyright (c) 2006-2022, Intel Corporation
```

12. Navigate to **Create Volume** and press <Enter>. When showing the warning, select **Yes** to create the RAID volume.

Figure 2-12. Confirmation Message to Create NVMe RAID Volume



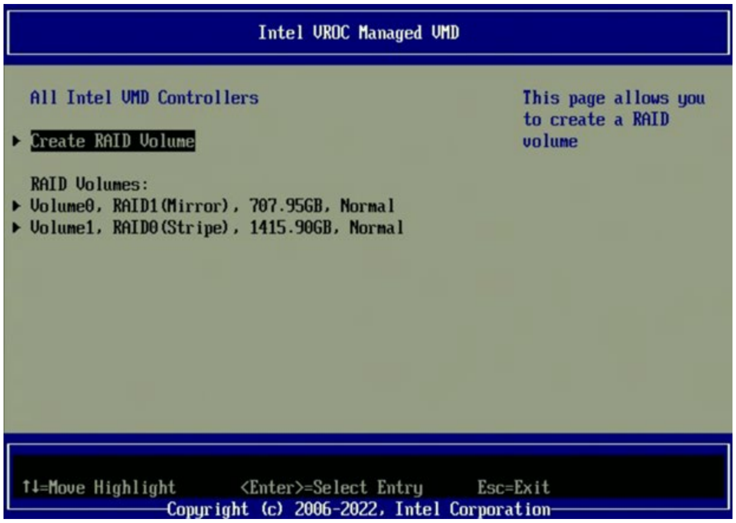
```
Create RAID Volume

745.21GB  Port 3:1 CPU1
VMD3
INTEL SSDPF21Q800GB       <X>
SN:PHAL11120043800CGN,
745.21GB  Port 3:2 CPU1
VMD3
INTEL SSDPF21Q800GB       < >
SN:PHAL111200JS800CGN,
745.21GB  Port 4:1 CPU1
VMD4

Strip Size:               <128KB>
Capacity (GB):            1415.90

▸ Create Volume

                        F9=Reset to
↑↓=Move Highlight       <Enter>=Sele
          Copyright (c) 2006-2
```

```
Create Volume

Create Volume                                   Create Volume

Are you really sure you want to create volume?
WARNING: All data on the selected drives will be
lost.

▸ Yes
▸ No

↑↓=Move Highlight       <Enter>=Select Entry      Esc=Exit
          Copyright (c) 2006-2022, Intel Corporation
```

13. The user will then be returned to the **Intel Virtual RAID on CPU** screen and the newly created RAID volume will be listed just below the text **Intel VROC Managed Volumes**.

Note: Other drives or unused portions of drives will be listed under *Non-RAID Physical Disks*. Those may be used to create additional RAID volumes.

Figure 2-13. Intel VROC NVMe RAID Volumes Created



Intel VROC Managed VMD

All Intel VMD Controllers

▸ Create RAID Volume

RAID Volumes:
▸ Volume0, RAID1(Mirror), 707.95GB, Normal
▸ Volume1, RAID0(Stripe), 1415.90GB, Normal

This page allows you to create a RAID volume

↑↓=Move Highlight <Enter>=Select Entry Esc=Exit

Copyright (c) 2006-2022, Intel Corporation

14. To exit the user interface, press <Esc>. Press <Esc> again, the user will be presented with the following message: "*Changes have not saved. Save changes and exit? Press 'Y' to save and exit, 'N' to discard and exit, 'ESC' to cancel*". Press <Y> to save and exit.

Note: Not saving at this time will discard the changes made, including all changes and configuration settings for the RAID array.

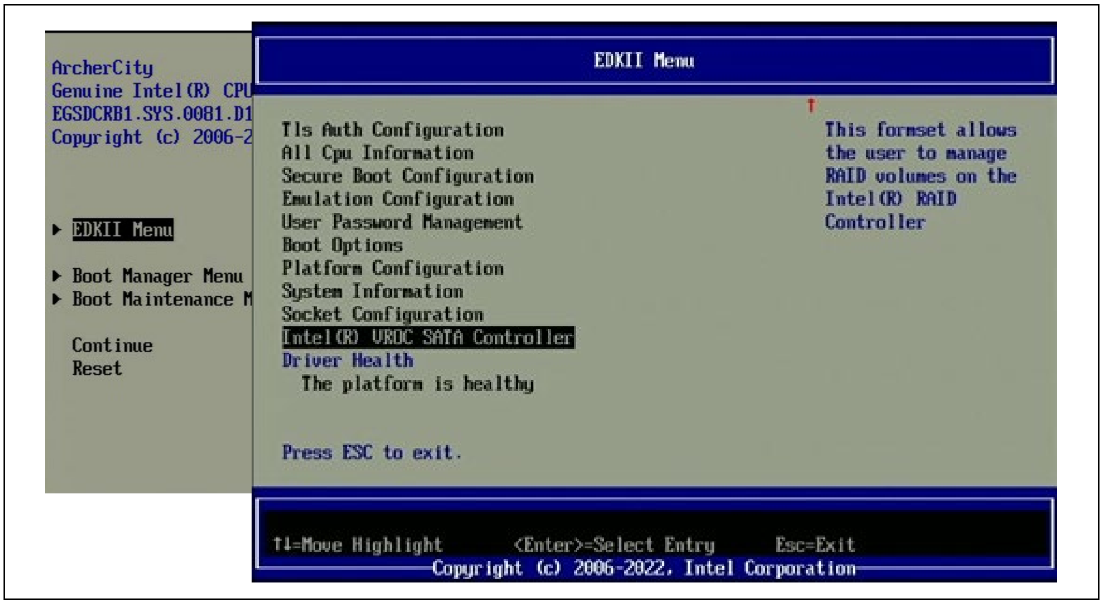
15. To save and reboot, in order to begin the operating system installation, press <Esc> to return to the Main Menu. Navigate to select **Reset** and press <Enter> to reboot the system back to the boot menu.

Note: For RAID levels 1, 5 and 10, the system will not automatically initialize these volumes in the UEFI. This will need to be accomplished once the Operating System has been installed.

2.4 Creating Intel® VROC SATA RAID Volume in UEFI HII

1. Enter into the BIOS configuration setup menu to access the Intel® VROC UEFI HII interface. This example uses <F2>.
2. In this example of Intel® CRB reference BIOS, the user can use the arrow keys to move the cursor to the **EDKII Menu** and press <Enter>. Then, use the arrow keys, move the cursor to **Intel VROC SATA Controller** and press <Enter>.

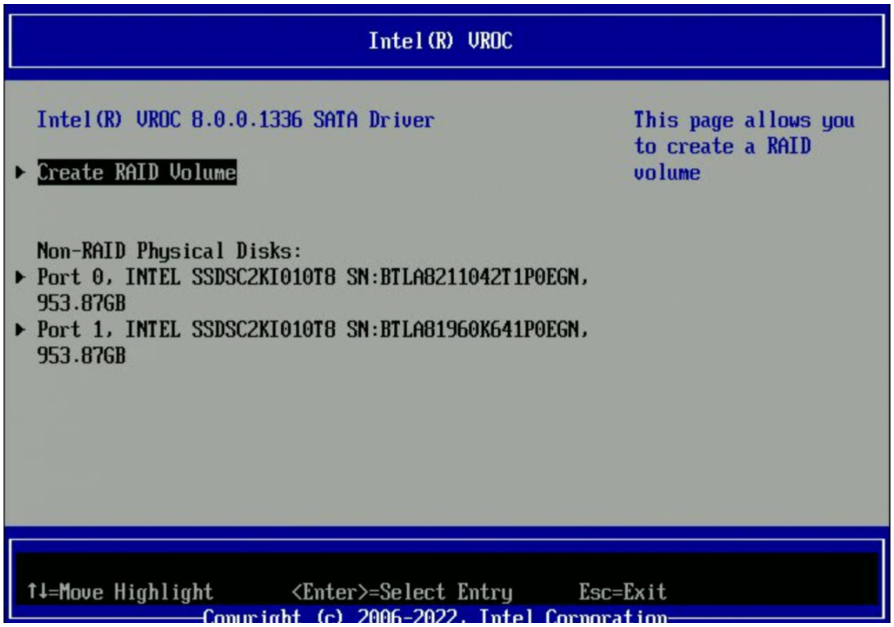
Figure 2-14. Intel VROC SATA Controller BIOS Setting



```
ArcherCity                   EDKII Menu
Genuine Intel(R) CPU
EGSDCRB1.SYS.0081.D1
Copyright (c) 2006-2     Tls Auth Configuration            This formset allows
                         All Cpu Information               the user to manage
                         Secure Boot Configuration         RAID volumes on the
                         Emulation Configuration           Intel(R) RAID
▶ EDKII Menu             User Password Management          Controller
                         Boot Options
▶ Boot Manager Menu      Platform Configuration
▶ Boot Maintenance M     System Information
                         Socket Configuration
  Continue               Intel(R) VROC SATA Controller
  Reset                  Driver Health
                           The platform is healthy

                         Press ESC to exit.

 ↑↓=Move Highlight        <Enter>=Select Entry        Esc=Exit
                 Copyright (c) 2006-2022, Intel Corporation
```

3. Select Create RAID Volume.

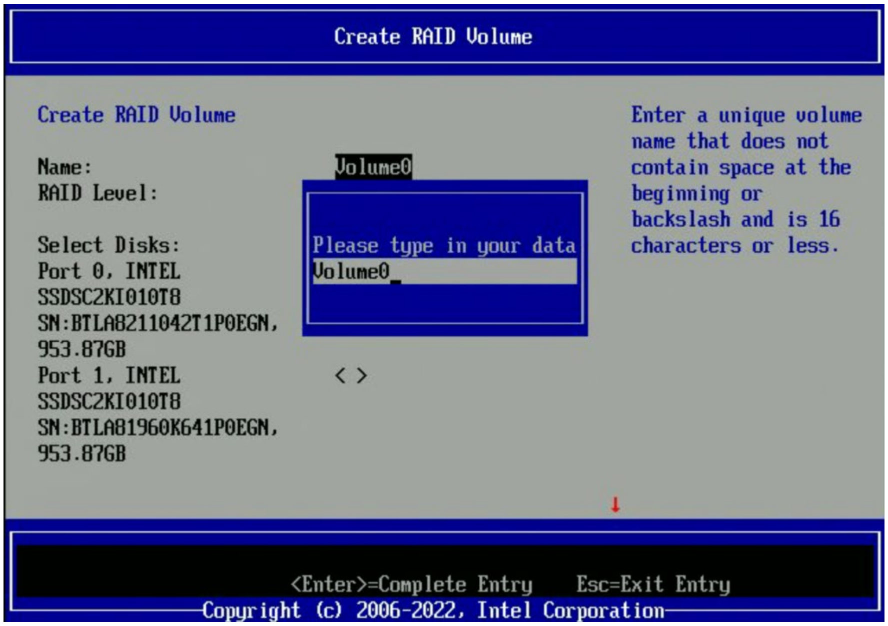
Figure 2-15. Create SATA RAID Volume Option in BIOS



```
                              Intel(R) VROC

  Intel(R) VROC 8.0.0.1336 SATA Driver               This page allows you
                                                      to create a RAID
▶ Create RAID Volume                                  volume


  Non-RAID Physical Disks:
▶ Port 0, INTEL SSDSC2KI010T8 SN:BTLA8211042T1P0EGN,
  953.87GB
▶ Port 1, INTEL SSDSC2KI010T8 SN:BTLA81960K641P0EGN,
  953.87GB

 ↑↓=Move Highlight        <Enter>=Select Entry        Esc=Exit
                 Copyright (c) 2006-2022, Intel Corporation
```

4. Press the <Enter> key to type in a volume name or move down the cursor to accept the default name.

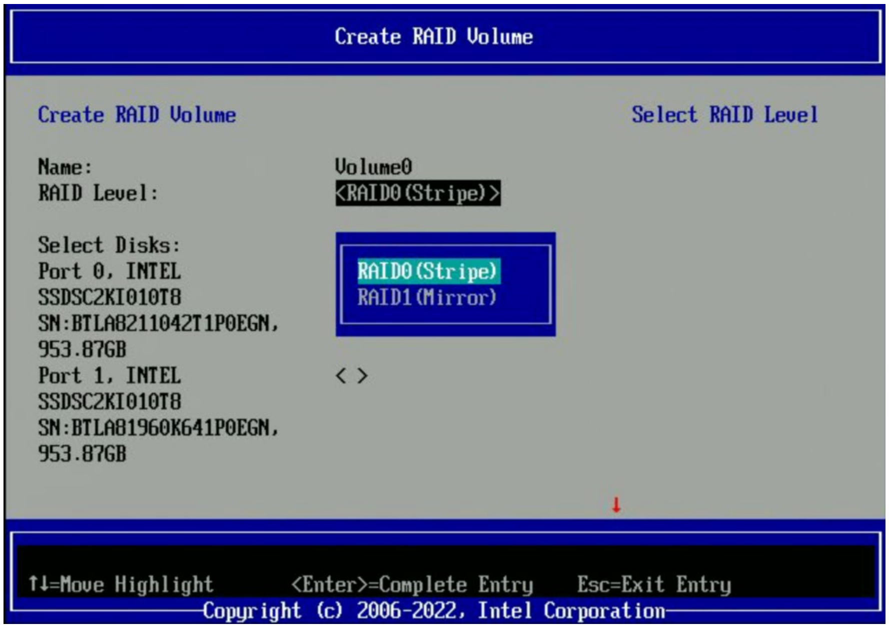
Figure 2-16. Name Option for SATA RAID Volume Creation



```
                         Create RAID Volume

Create RAID Volume                                   Enter a unique volume
                                                     name that does not
Name:                      Volume0                   contain space at the
RAID Level:                                          beginning or
                                                     backslash and is 16
Select Disks:             Please type in your data   characters or less.
Port 0, INTEL             Volume0_
SSDSC2KI010T8
SN:BTLA8211042T1P0EGN,
953.87GB
Port 1, INTEL              < >
SSDSC2KI010T8
SN:BTLA81960K641P0EGN,
953.87GB
                                                 ↓

                  <Enter>=Complete Entry     Esc=Exit Entry
              Copyright (c) 2006-2022, Intel Corporation
```

5. Select the RAID level by pressing the <Enter> key and using the arrow keys to scroll through the available values. Highlight the desired RAID type and press <Enter> to set the RAID type.

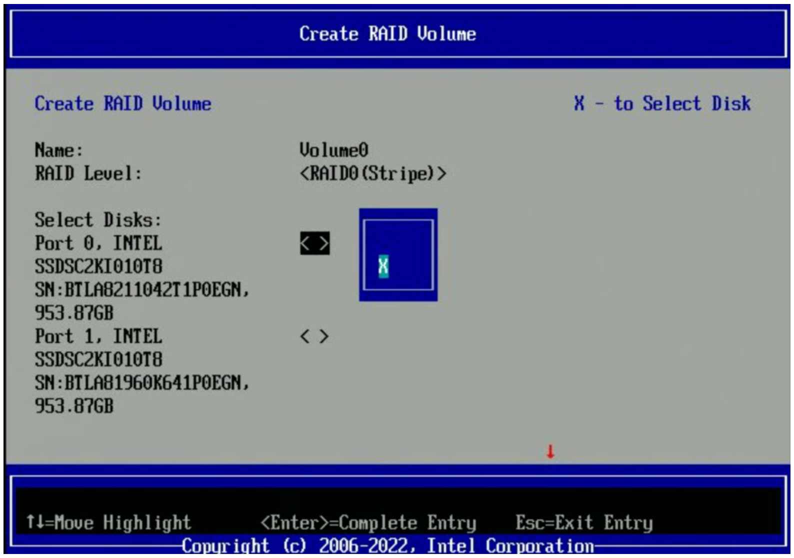
Figure 2-17. RAID Level Option for SATA RAID Volume Creation



```
                         Create RAID Volume

Create RAID Volume                                   Select RAID Level

Name:                      Volume0
RAID Level:                <RAID0(Stripe)>

Select Disks:
Port 0, INTEL                RAID0(Stripe)
SSDSC2KI010T8                RAID1(Mirror)
SN:BTLA8211042T1P0EGN,
953.87GB
Port 1, INTEL              < >
SSDSC2KI010T8
SN:BTLA81960K641P0EGN,
953.87GB
                                                 ↓

↑↓=Move Highlight      <Enter>=Complete Entry     Esc=Exit Entry
              Copyright (c) 2006-2022, Intel Corporation
```

6. Using the arrow keys, select the drives one by one by highlighting the < > bracket on the line next to the drive’s port number. Press <Enter> to open the selection menu, which will be set to blank or off status. Navigate to highlight the X and press <Enter> to include that drive within the array.

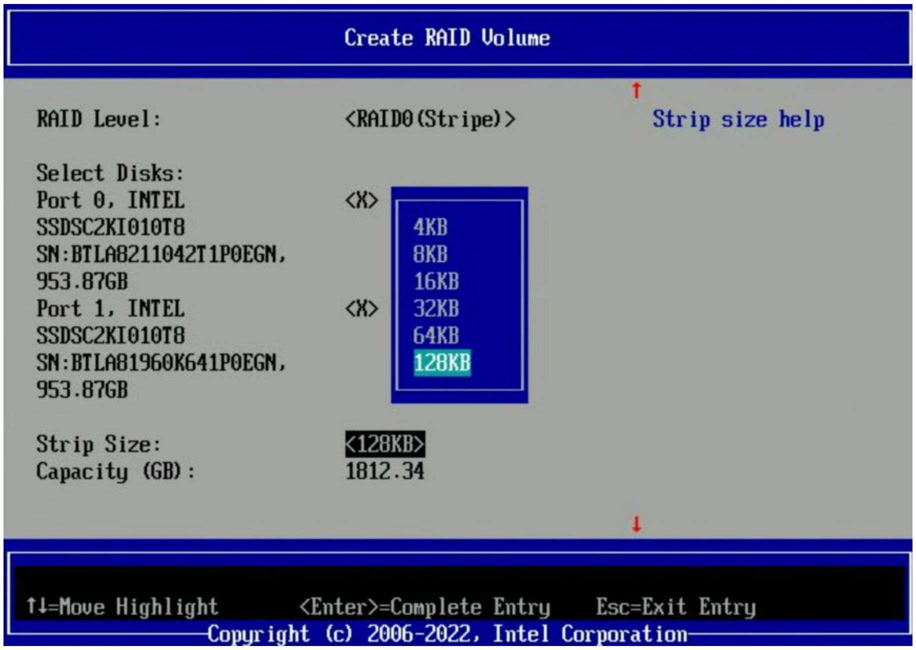
Figure 2-18. Drives Selection for SATA RAID Volume Creation



```
                         Create RAID Volume

Create RAID Volume                                  X - to Select Disk

Name:                        Volume0
RAID Level:                  <RAID0(Stripe)>

Select Disks:
Port 0, INTEL                < >      X
SSDSC2KI010T8
SN:BTLA8211042T1P0EGN,
953.87GB
Port 1, INTEL                < >
SSDSC2KI010T8
SN:BTLA81960K641P0EGN,
953.87GB

↑↓=Move Highlight       <Enter>=Complete Entry     Esc=Exit Entry
                Copyright (c) 2006-2022, Intel Corporation
```

7. Repeat Step 6 for each drive required within this array.
8. Unless the user has selected a RAID 1, select the strip size by using the arrow keys and press <Enter> to open the options menu. Utilize the arrow keys to select the desired strip size and press <Enter> to save the value.

Figure 2-19. Strip Size Selection for SATA RAID Volume Creation

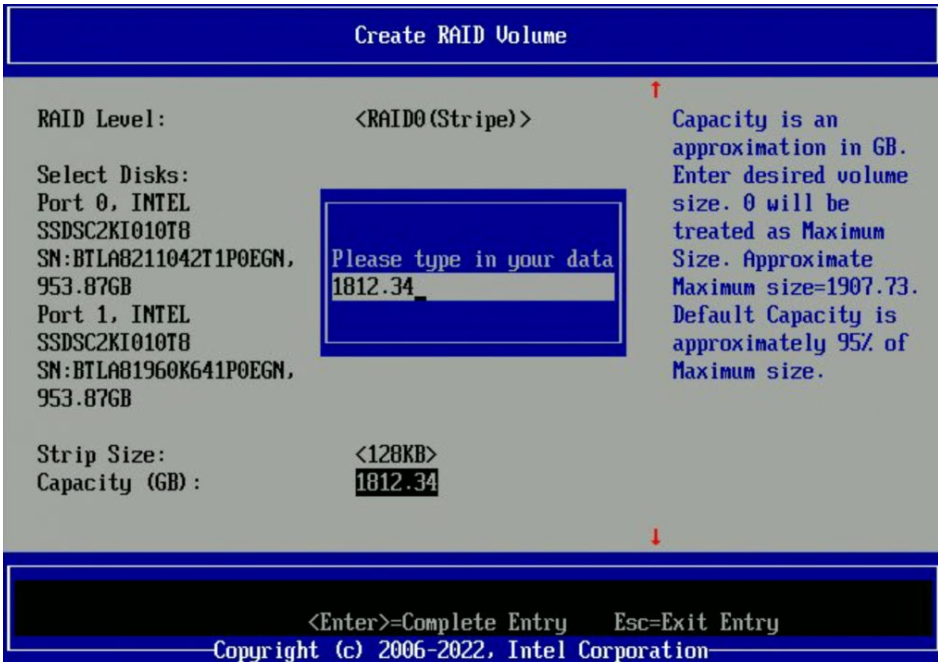


```
                         Create RAID Volume

RAID Level:                  <RAID0(Stripe)>         Strip size help

Select Disks:
Port 0, INTEL                <X>
SSDSC2KI010T8                     4KB
SN:BTLA8211042T1P0EGN,            8KB
953.87GB                          16KB
Port 1, INTEL                <X>  32KB
SSDSC2KI010T8                     64KB
SN:BTLA81960K641P0EGN,            128KB
953.87GB

Strip Size:                  <128KB>
Capacity (GB):               1812.34

↑↓=Move Highlight       <Enter>=Complete Entry     Esc=Exit Entry
                Copyright (c) 2006-2022, Intel Corporation
```

9. Select the volume capacity and press the <Enter> key. The default value will be displayed as the maximum capacity available with the drives selected. The value is calculated in bytes. A 700GB drive would use the following math: 700 * 1024 = 716000.

Note: Unless specifically selected, the default volume capacity will be calculated based on 95% of the available space of the smallest physical disk. This allows for the variances in the physical disk capacities from different vendors. This feature is called *Disk Coercion* which is a common feature of most RAID solutions.

Figure 2-20. Volume Capacity Selection for SATA RAID Volume Creation



Create RAID Volume

RAID Level: <RAID0(Stripe)>

Select Disks:
Port 0, INTEL
SSDSC2KI010T8
SN:BTLA8211042T1P0EGN,
953.87GB
Port 1, INTEL
SSDSC2KI010T8
SN:BTLA81960K641P0EGN,
953.87GB

Please type in your data
1812.34_

Strip Size: <128KB>
Capacity (GB): 1812.34

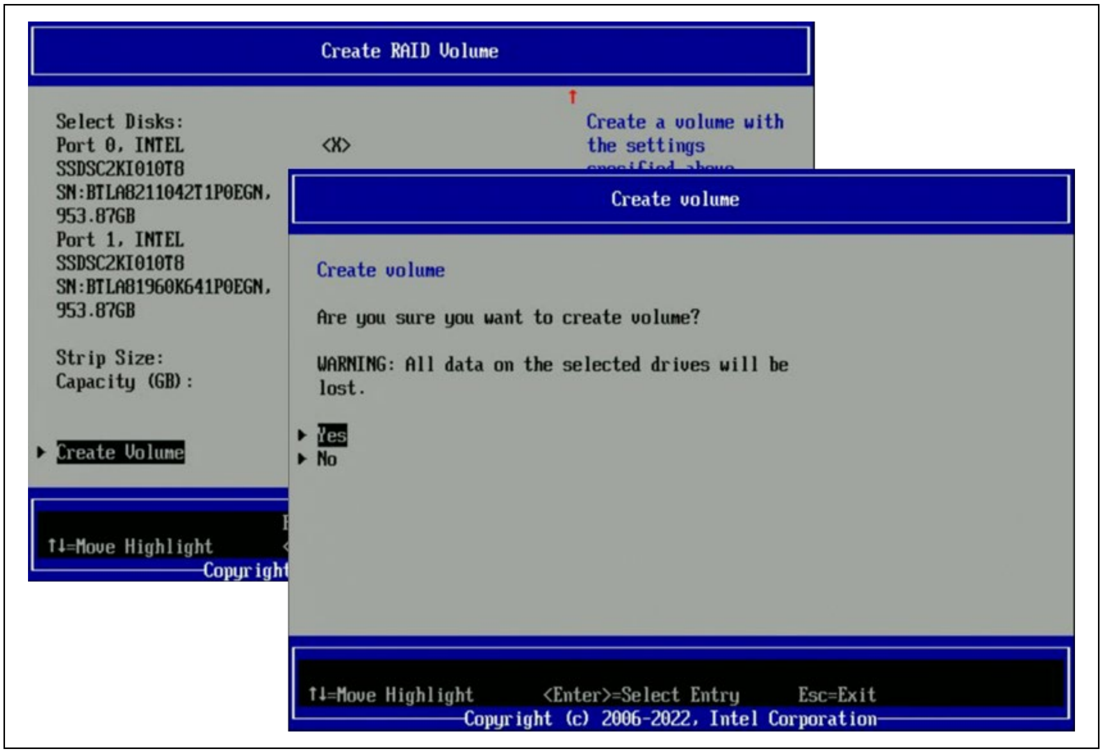
Capacity is an approximation in GB. Enter desired volume size. 0 will be treated as Maximum Size. Approximate Maximum size=1907.73. Default Capacity is approximately 95% of Maximum size.

<Enter>=Complete Entry Esc=Exit Entry

Copyright (c) 2006-2022, Intel Corporation

10. Navigate to **Create Volume** and press <Enter>, then select **Yes** to create the RAID volume.

Figure 2-21. Confirmation Message to Create SATA RAID Volume



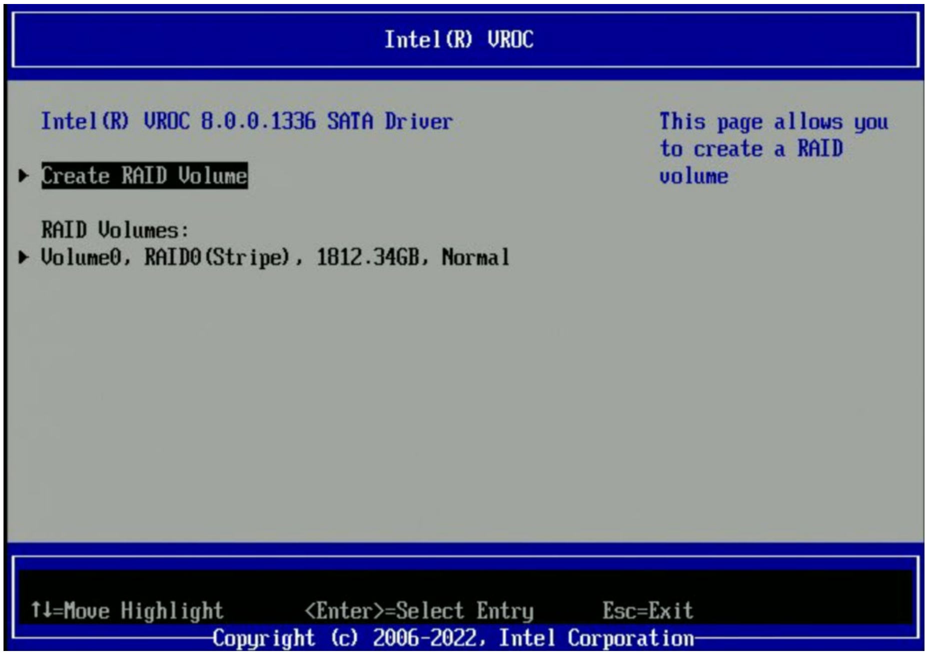
```
Create RAID Volume

Select Disks:                                   Create a volume with
Port 0, INTEL                  <X>              the settings
SSDSC2KI010T8
SN:BTLA8211042T1P0EGN,
953.87GB
Port 1, INTEL
SSDSC2KI010T8
SN:BTLA81960K641P0EGN,
953.87GB

Strip Size:
Capacity (GB):

▸ Create Volume

↑↓=Move Highlight
Copyright
```

```
Create volume

Create volume

Are you sure you want to create volume?

WARNING: All data on the selected drives will be
lost.

▸ Yes
▸ No

↑↓=Move Highlight       <Enter>=Select Entry       Esc=Exit
Copyright (c) 2006-2022, Intel Corporation
```

11. The user will then be returned to the **Intel® VROC SATA Driver** screen and the newly created RAID volume will be listed just below the text **RAID Volumes**.

Note: Other drives or unused portions of drives will be listed under *Non-RAID Physical Disks*. These may be used to create additional RAID volumes.

Figure 2-22. Intel VROC SATA RAID Volumes Created



```
Intel(R) VROC

Intel(R) VROC 8.0.0.1336 SATA Driver                    This page allows you
                                                        to create a RAID
▶ Create RAID Volume                                    volume

  RAID Volumes:
▶ Volume0, RAID0(Stripe), 1812.34GB, Normal

↑↓=Move Highlight        <Enter>=Select Entry       Esc=Exit
          Copyright (c) 2006-2022, Intel Corporation
```

12. To exit the user interface, press <Esc>. Press <Esc> again, the user will be presented with the following message: “*Changes have not saved. Save changes and exit? Press ‘Y’ to save and exit, ‘N’ to discard and exit, ‘ESC’ to cancel*”. Press <Y> to save and exit.

Note: Not saving at this time will discard the changes made, including all changes and configuration settings for the RAID array.

13. To save and reboot, in order to begin the operating system installation, press <Esc> to return to the Main Menu. Navigate to select **Reset** and press <Enter> to reboot the system back to the boot menu.

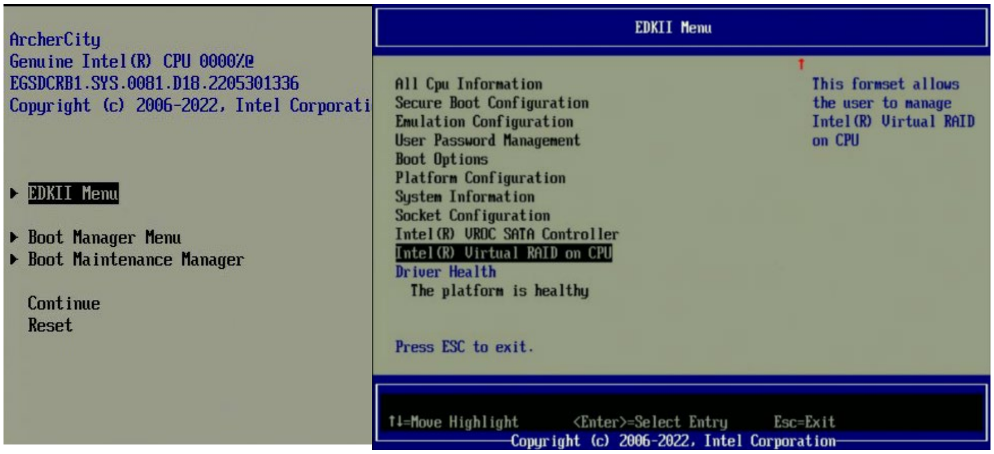
Note: For RAID levels 1, 5 and 10 the system will not automatically initialize these volumes via the UEFI. This will need to be accomplished once the Operating System has been installed.

2.5 Removing RAID Volumes in UEFI HII

In a manner much like what was used to create the volume, the process to remove a RAID volume is very similar.

1. As the system is booting, press the appropriate key to enter the Setup Menu. This example uses <F2>.
2. Navigate to **EDKII Menu** and press <Enter>.
3. Navigate to **Intel Virtual RAID on CPU (Intel® VROC)** and press <Enter>.

Figure 2-23. Intel Virtual RAID on CPU BIOS Setting

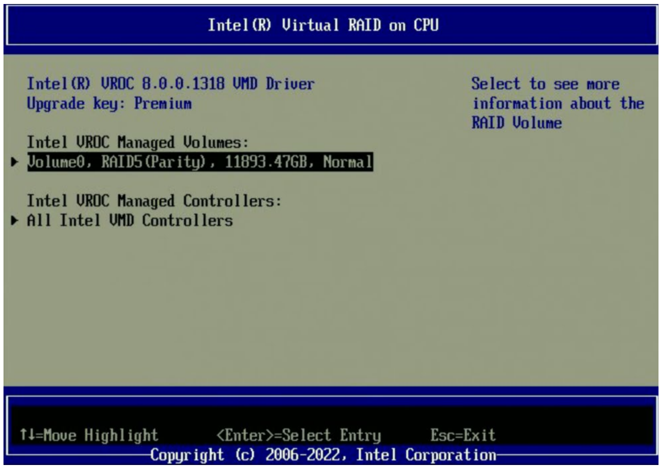


```
ArcherCity
Genuine Intel(R) CPU 0000%@
EGSDCRB1.SYS.0081.D18.2205301336
Copyright (c) 2006-2022, Intel Corporati

▶ EDKII Menu

▶ Boot Manager Menu
▶ Boot Maintenance Manager

  Continue
  Reset
```

```
                         EDKII Menu

All Cpu Information                          This formset allows
Secure Boot Configuration                    the user to manage
Emulation Configuration                      Intel(R) Virtual RAID
User Password Management                     on CPU
Boot Options
Platform Configuration
System Information
Socket Configuration
Intel(R) VROC SATA Controller
Intel(R) Virtual RAID on CPU
Driver Health
  The platform is healthy

Press ESC to exit.

↑↓=Move Highlight     <Enter>=Select Entry     Esc=Exit
          Copyright (c) 2006-2022, Intel Corporation
```

4. Select the RAID volume you want to delete, and press <Enter>.

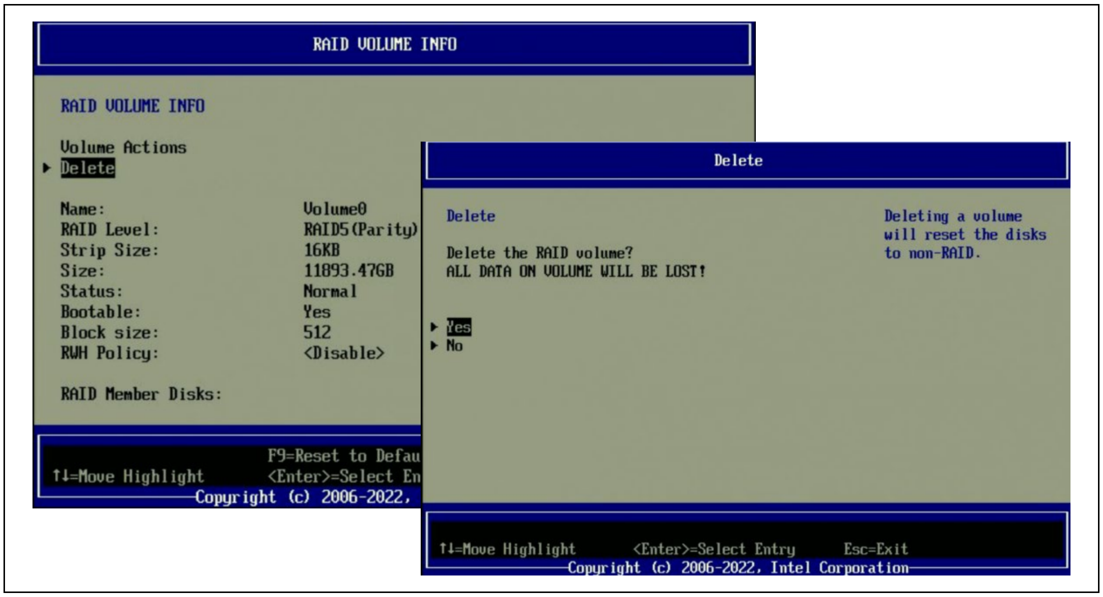
Figure 2-24. Intel VROC Managed Volumes



```
                  Intel(R) Virtual RAID on CPU

 Intel(R) VROC 8.0.0.1318 VMD Driver          Select to see more
 Upgrade key: Premium                         information about the
                                              RAID Volume
 Intel VROC Managed Volumes:
▶Volume0, RAID5(Parity), 11893.47GB, Normal

 Intel VROC Managed Controllers:
▶All Intel VMD Controllers

↑↓=Move Highlight     <Enter>=Select Entry     Esc=Exit
          Copyright (c) 2006-2022, Intel Corporation
```

5. Navigate to **Delete** and press <Enter>, select **Yes** and press <Enter> to confirm the removal of the RAID volume.

Figure 2-25. Confirmation Message to Delete an Intel VROC RAID Volume



```
RAID VOLUME INFO

RAID VOLUME INFO

Volume Actions
▸ Delete

Name:                         Volume0
RAID Level:                   RAID5(Parity)
Strip Size:                   16KB
Size:                         11893.47GB
Status:                       Normal
Bootable:                     Yes
Block size:                   512
RWH Policy:                   <Disable>

RAID Member Disks:

                    F9=Reset to Defau
↑↓=Move Highlight   <Enter>=Select En
          Copyright (c) 2006-2022,
```

```
Delete

Delete                                          Deleting a volume
                                                will reset the disks
Delete the RAID volume?                         to non-RAID.
ALL DATA ON VOLUME WILL BE LOST!


▸ Yes
▸ No

↑↓=Move Highlight     <Enter>=Select Entry     Esc=Exit
        Copyright (c) 2006-2022, Intel Corporation
```

§§

3 Installation of Intel® VROC Linux* OS Drivers and Tools

Intel® VROC Linux* release strategy is upstream to open-source community and have OSV partners to incorporate necessary upstream patches into their Linux* product releases to have the inbox support of Intel® VROC. Meanwhile, Intel® may release Intel® VROC Linux* update (out-of-box) packages in order to support some old Linux* distributions or new features on certain Intel® Xeon platforms. For detailed supported OS list, refer to the Intel® Virtual RAID on CPU (Intel® VROC) Supported Configurations.

This chapter will cover installing of Intel® VROC out-of-box package as well as configuring inbox Intel® VROC components in the Intel® VROC supported Linux* distributions.

3.1 Installation of Intel® VROC Linux* Update Packages

For certain Linux* distributions, Intel® may release out-of-box update packages to support a full functioning Intel® VROC product. This section illustrates how to install the Intel® VROC Linux* out-of-box packages in RHEL OS.

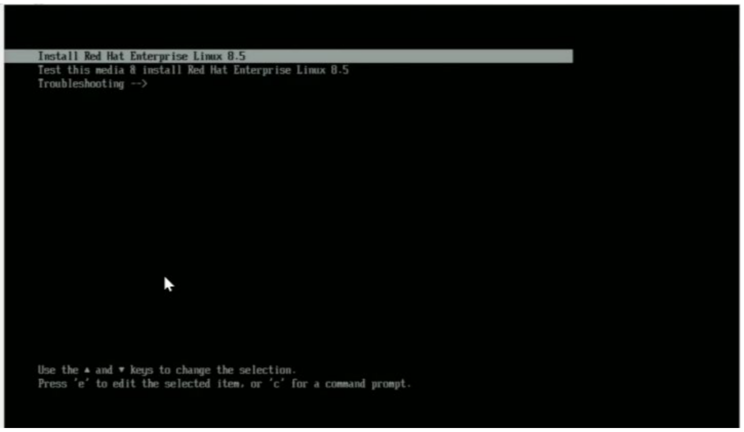
3.1.1 Installing Intel® VMD Replacement Driver during OS Installation

The Intel® VMD replacement driver is included in the Intel® VROC out-of-box package. This replacement driver is used to replace the OS kernel inbox VMD driver to enable or improve Intel® VROC functionalities on certain Intel® Xeon platforms.

The Intel® VMD replacement driver can be installed during the OS installation phase or in the OS environment. The following steps illustrate how to install an Intel® VMD replacement driver during OS installation.

1. Prepare USB drives with RHEL installation ISO and Intel® VROC out-of-box driver ISO package:
   a. Use the **dd** command to create an USB drive with RHEL ISO installation:
   ```
   # dd if=/path/to/<RHEL_OS>.iso of=/dev/sdX status=progress
   ```
   b. Use **dd** command to create a USB drive with Intel® VROC out-of-box ISO package:
   ```
   # dd if=/path/to/<vroc_update_driver>.iso of=/dev/sdX status=progress
   ```
   c. After the USB drive with the VROC driver is prepared, make sure it has the correct label name "*OEMDRV*":
   ```
   # lsblk -o name,label
   NAME          LABEL
   sdb           OEMDRV
   ```
2. Boot your system with the bootable media created in the previous step. When booting we will get the below installation screen, select the "*Install Red Hat Enterprise Linux* 8.5*" option and press <E>.

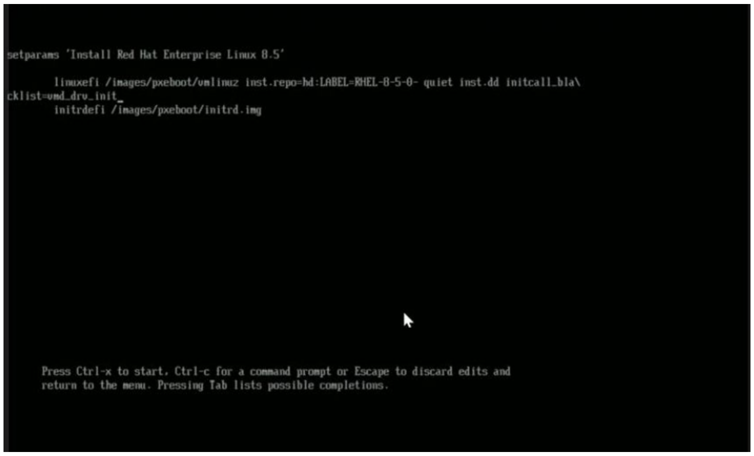
Figure 3-1. Start OS Installation Process



```
Install Red Hat Enterprise Linux 8.5
Test this media 8 install Red Hat Enterprise Linux 8.5
Troubleshooting -->

Use the ▴ and ▾ keys to change the selection.
Press 'e' to edit the selected item, or 'c' for a command prompt.
```

3. Add **`inst.dd initcall_blacklist=vmd_drv_init`** to the end of the kernel boot command line. Press Ctrl + X to continue with the installation.

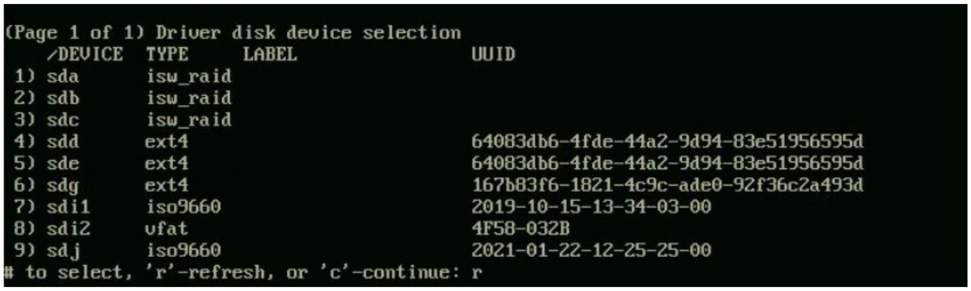
Figure 3-2. Configuring Kernel Boot Parameters



```
setparams 'Install Red Hat Enterprise Linux 8.5'

        linuxefi /images/pxeboot/vmlinuz inst.repo=hd:LABEL=RHEL-8-5-0- quiet inst.dd initcall_bla\
cklist=vmd_drv_init_
        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.
```

4. Select the driver with label "*OEMDRV*" in this example. If you don't see the USB device which is named "E", input <r> to refresh.

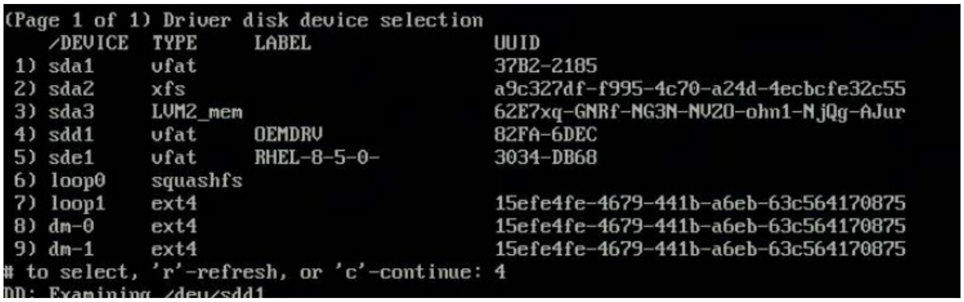
Figure 3-3. Driver Disk Device Selection Before Refresh



```
(Page 1 of 1) Driver disk device selection
    /DEVICE TYPE     LABEL                UUID
 1) sda     isw_raid
 2) sdb     isw_raid
 3) sdc     isw_raid
 4) sdd     ext4                          64083db6-4fde-44a2-9d94-83e51956595d
 5) sde     ext4                          64083db6-4fde-44a2-9d94-83e51956595d
 6) sdg     ext4                          167b83f6-1821-4c9c-ade0-92f36c2a493d
 7) sdi1    iso9660                       2019-10-15-13-34-03-00
 8) sdi2    vfat                          4F58-032B
 9) sdj     iso9660                       2021-01-22-12-25-25-00
# to select, 'r'-refresh, or 'c'-continue: r
```

After refresh, select the device with label "E".

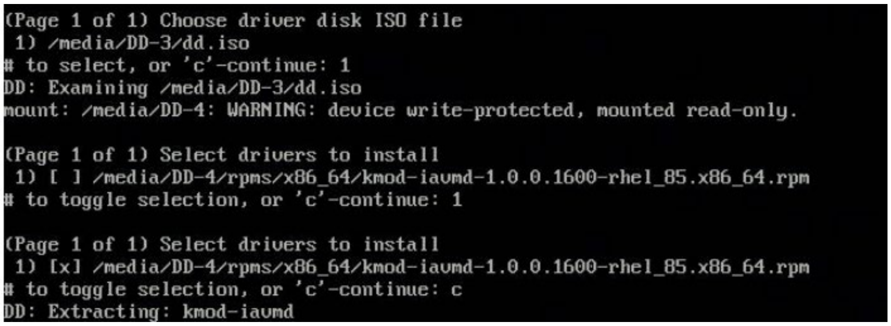
Figure 3-4. Driver Disk Device Selection After Refresh



```
(Page 1 of 1) Driver disk device selection
    /DEVICE  TYPE     LABEL                 UUID
 1) sda1     vfat                           37B2-2185
 2) sda2     xfs                            a9c327df-f995-4c70-a24d-4ecbcfe32c55
 3) sda3     LVM2_mem                       62E7xq-GNRf-NG3N-NVZ0-ohn1-NjQg-AJur
 4) sdd1     vfat     OEMDRV                82FA-6DEC
 5) sde1     vfat     RHEL-8-5-0-           3034-DB68
 6) loop0    squashfs
 7) loop1    ext4                           15efe4fe-4679-441b-a6eb-63c564170875
 8) dm-0     ext4                           15efe4fe-4679-441b-a6eb-63c564170875
 9) dm-1     ext4                           15efe4fe-4679-441b-a6eb-63c564170875
# to select, 'r'-refresh, or 'c'-continue: 4
DD: Examining /dev/sdd1
```

5. Toggle the *kmod-iavmd* package, then input <c> to go back to the main menu.

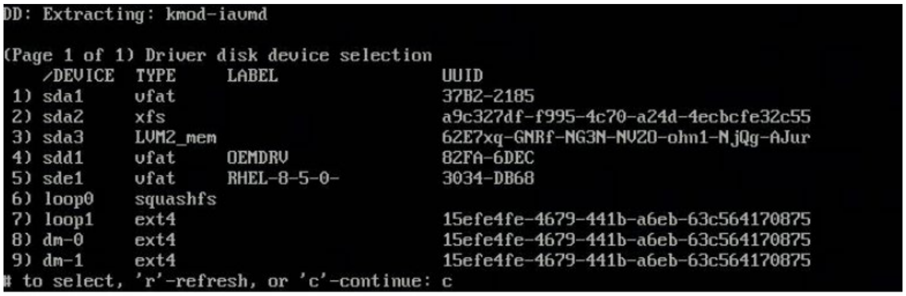
Figure 3-5. Extracting kmod-iavmd Package



```
(Page 1 of 1) Choose driver disk ISO file
 1) /media/DD-3/dd.iso
# to select, or 'c'-continue: 1
DD: Examining /media/DD-3/dd.iso
mount: /media/DD-4: WARNING: device write-protected, mounted read-only.

(Page 1 of 1) Select drivers to install
 1) [ ] /media/DD-4/rpms/x86_64/kmod-iavmd-1.0.0.1600-rhel_85.x86_64.rpm
# to toggle selection, or 'c'-continue: 1

(Page 1 of 1) Select drivers to install
 1) [x] /media/DD-4/rpms/x86_64/kmod-iavmd-1.0.0.1600-rhel_85.x86_64.rpm
# to toggle selection, or 'c'-continue: c
DD: Extracting: kmod-iavmd
```

6. At the main menu, type <c> to continue with the installation.

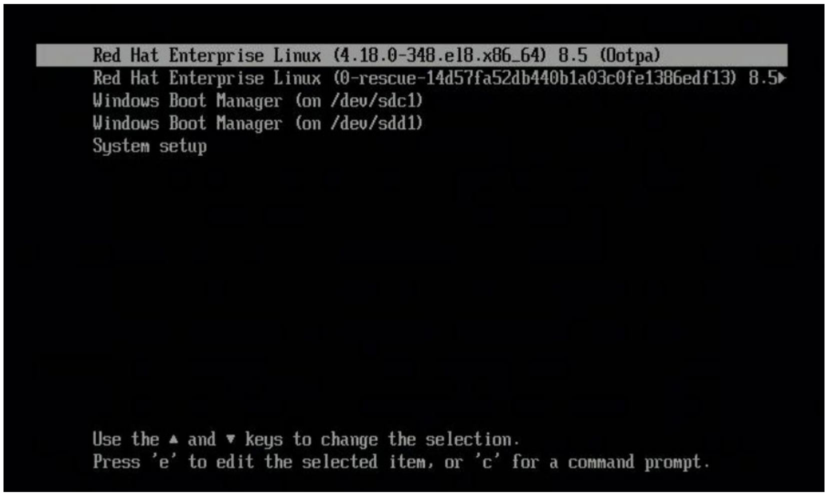
Figure 3-6. Continue OS Installation



```
DD: Extracting: kmod-iavmd

(Page 1 of 1) Driver disk device selection
    /DEVICE  TYPE     LABEL                 UUID
 1) sda1     vfat                           37B2-2185
 2) sda2     xfs                            a9c327df-f995-4c70-a24d-4ecbcfe32c55
 3) sda3     LVM2_mem                       62E7xq-GNRf-NG3N-NVZ0-ohn1-NjQg-AJur
 4) sdd1     vfat     OEMDRV                82FA-6DEC
 5) sde1     vfat     RHEL-8-5-0-           3034-DB68
 6) loop0    squashfs
 7) loop1    ext4                           15efe4fe-4679-441b-a6eb-63c564170875
 8) dm-0     ext4                           15efe4fe-4679-441b-a6eb-63c564170875
 9) dm-1     ext4                           15efe4fe-4679-441b-a6eb-63c564170875
# to select, 'r'-refresh, or 'c'-continue: c
```

7. Follow the normal RHEL installer guidance to complete the OS installation.
8. Reboot the system after the installation. Add **`initcall_blacklist=vmd_drv_init`** in the boot command line before directly booting into the OS. This boot parameter is used to tell the OS to blacklist the inbox VMD driver and load the Intel® VMD replacement driver.
9. Select the "*Red Hat Enterprise Linux**" option and press <E>.

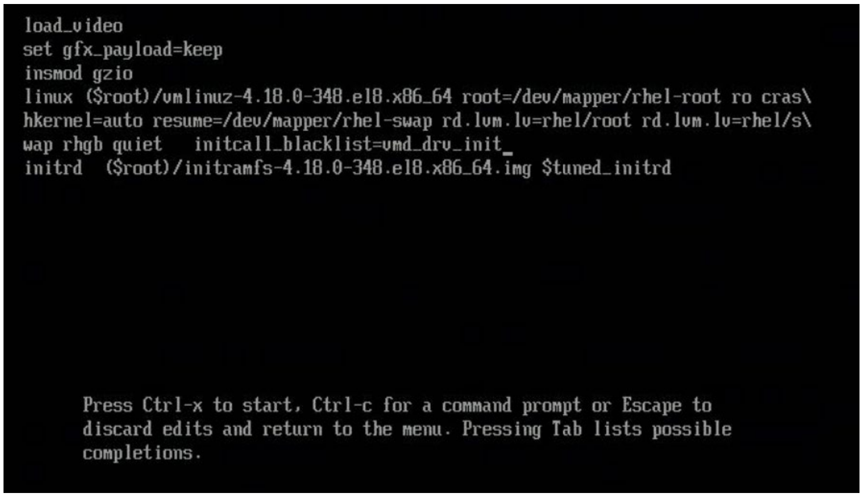
Figure 3-7. Booting Operating System



```
      Red Hat Enterprise Linux (4.18.0-348.el8.x86_64) 8.5 (Ootpa)
      Red Hat Enterprise Linux (0-rescue-14d57fa52db440b1a03c0fe1386edf13) 8.5▸
      Windows Boot Manager (on /dev/sdc1)
      Windows Boot Manager (on /dev/sdd1)
      System setup



      Use the ▴ and ▾ keys to change the selection.
      Press 'e' to edit the selected item, or 'c' for a command prompt.
```

10. Add **`initcall_blacklist=vmd_drv_init`** in the Linux* boot command line, and then press Ctrl + X to start.

Figure 3-8. Configuring OS Boot Parameters



```
load_video
set gfx_payload=keep
insmod gzio
linux ($root)/vmlinuz-4.18.0-348.el8.x86_64 root=/dev/mapper/rhel-root ro cras\
hkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/s\
wap rhgb quiet   initcall_blacklist=vmd_drv_init_
initrd  ($root)/initramfs-4.18.0-348.el8.x86_64.img $tuned_initrd



     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.
```

3.1.2 Installing Intel® VROC out-of-box Drivers in Linux* OS

Section 3.1.1 illustrates the steps to install the Intel® VMD replacement driver during OS installation. This section will cover the steps to install the Intel® VMD replacement driver as well as other Intel® VROC update packages in the Linux* OS.

1. Copy the Intel® VROC update package to the target Linux* OS and mount the iso image to a certain directory. Enter that directory and you will typically find 3 rpm packages as showed in below example.

```
# ls -l rpms/x86_64/
total 828
-r--r--r--. 1 root root 328776 Apr  4 07:23 kmod-iavmd-1.0.0.1600-
rhel_85.x86_64.rpm
```

```
-r--r--r--. 1 root root  83860 Apr  4 07:23 ledmon-0.95-
1.Intel®.7468292.el8.x86_64.rpm
-r--r--r--. 1 root root 431924 Apr  4 07:23 mdadm-4.2-
1.Intel®.9009306.el8.x86_64.rpm
dr-xr-xr-x. 2 root root   2048 Apr  4 07:23 repodata
```

2. Use the **rpm -Uvh --force** command to install all those packages. See the following examples.

   Example 1: This is an example to install the Intel® VMD replacement driver package prefixed by "*kmod-iavmd-*":

   ```
   # rpm -Uvh --force kmod-iavmd-1.0.0.1600-rhel_85.x86_64.rpm
   warning: kmod-iavmd-1.0.0.1600-rhel_85.x86_64.rpm: Header V4 RSA/SHA256
   Signature, key ID c343c1b0: NOKEY
   Verifying...                        ################################ [100%]
   Preparing...                        ################################ [100%]
   Updating / installing...
      1:kmod-iavmd-1.0.0.1600-rhel_85  ################################ [100%]
   ```

   Check the rpm installation status by running the following command:

   ```
   # rpm -qa | grep kmod-iavmd
   kmod-iavmd-1.0.0.1600-rhel_85.x86_64
   ```

   Example 2: This is an example to install Intel® VROC Linux* update package for the *mdadm* utility.

   ```
   # rpm -Uvh --force mdadm-4.2-1.Intel®.9009306.el8.x86_64.rpm
   warning: mdadm-4.2-1.Intel®.9009306.el8.x86_64.rpm: Header V4 RSA/SHA256
   Signature, key ID b7accecb: NOKEY
   Verifying...                         ############################## [100%]
   Preparing...                         ############################## [100%]
   Updating / installing...
      1:mdadm-4.2-1.Intel®.9009306.el8  ############################## [100%]
   ```

   Check the rpm installation status by running the following command:

   ```
   # mdadm --version
   mdadm – v4.2 – 2021-12-30 – Intel®_Build: 1.Intel®.9009306.el8
   ```

   Example 3: This is an example to install Intel® VROC Linux* update package for *ledmon* utility.

   ```
   # rpm -Uvh --force ledmon-0.95-1.Intel®.7468292.el8.x86_64.rpm
   warning: ledmon-0.95-1.Intel®.7468292.el8.x86_64.rpm: Header V4 RSA/SHA256
   Signature, key ID b7accecb: NOKEY
   Verifying...                           ############################ [100%]
   Preparing...                           ############################ [100%]
   Updating / installing...
      1:ledmon-0.95-1.Intel®.7468292.el8  ############################ [100%]
   ```

   Check the rpm installation status by running the following command:

   ```
   # ledmon --version
   Intel(R) Enclosure LED Monitor Service 0.95
   Copyright (C) 2009-2021 Intel Corporation.

   This is free software; see the source for copying conditions. There is NO
   warranty;
   not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

   ledmon[41052] : exit status is STATUS_SUCCESS.
   ```

3. Update the GRUB configuration file to permanently add the following kernel boot parameter to allow the OS to automatically load the Intel® VMD replacement driver instead of the inbox one.

   ```
   initcall_blacklist=vmd_drv_init
   ```

   To achieve that, edit the file **/etc/default/grub**, and add **initcall_blacklist=vmd_drv_init** to the end of the **GRUB_CMDLINE_LINUX*** line. Take the following as an example:

   ```
   # vim /etc/default/grub
   GRUB_CMDLINE_LINUX*=”crashkernel=auto resume=/dev/mapper/rhel-swap
   rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet initcall_blacklist=vmd_drv_init”
   ```

After editing the **/etc/default/grub** file, regenerate the GRUB configuration file by running the following command:

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

3.2 Configuring Intel® VROC in Linux* Distributions with Inbox Support

3.2.1 Configuring Intel® VROC in SUSE Linux* Enterprise Server (SLES) 15

3.2.1.1 Installing SLES 15 SP2 onto Intel® VROC RAID

SLES 15 SP2 installer by default is not able to assemble RAID volumes created in the PreOS/UEFI. When installing SLES 15 SP2 onto an Intel® VROC RAID volume, a boot command line **autoassembly=1** should be added to enable RAID assembly in the installer.

3.2.1.2 Installing ledmon in SLES Family

SLES 15 family OSes don't include the *ledmon* package in the base OS installation. Users should install *ledmon* manually after the OS installation. Use the following command to install *ledmon*:

```
# zypper install ledmon
```

3.2.1.3 Enabling ledmon Service

By default, *ledmon* service is disabled in every supported Linux* distribution. Users should enable it and get it started by running the following command:

```
# systemctl enable --now ledmon.service
```

Check *ledmon* service status by running the following command:

```
# systemctl status ledmon.service
```

§§

4 Intel® VROC RAID Management in Linux*

Intel® VROC RAID management in Linux* is achieved by using Linux* *mdadm* tool. The following sections illustrate the *mdadm* commands with examples to manage Intel® VROC RAID in Linux*. This is not the complete manual for using Linux* *mdadm* tool. Users can reference the manual of *mdadm* for the complete usage.

4.1 Examine System's Intel® VROC RAID Capabilities

Before using Intel® VROC RAID in Linux*, the first step is to check the supported Intel® VROC RAID capability and configurations on the system. Users can use the following command to check what Intel® VROC RAID capabilities are supported as well as what drives are connected on the system.

Example 1: Check the platform's Intel® VROC RAID capability

```
# mdadm --detail-platform
Platform : Intel®(R) Virtual RAID on CPU
Version : 8.0.0.1304
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/pci0000:00/0000:00:17.0 (SATA)
Port7 : /dev/sdj (CVEM948500JX032HGN)
Port3 : /dev/sdd (PEPR304600TX120LGN)
Port4 : /dev/sde (BTPR142501MH120LGN)
Port1 : /dev/sdb (BTPR209202AA120LGN)
Port5 : /dev/sdf (BTPR147300GW120LGN)
Port2 : /dev/sdc (BTPR212500QE120LGN)
Port6 : /dev/sdi (CVEM947301KY032HGN)
Port0 : /dev/sda (BTPR212500UG120LGN)

[…]

Platform : Intel®(R) Virtual RAID on CPU
Version : 8.0.0.1304
RAID Levels : raid0 raid1 raid10 raid5
Chunk Sizes : 4k 8k 16k 32k 64k 128k
2TB volumes : supported
2TB disks : supported
Max Disks : 48
Max Volumes : 2 per array, 24 per controller
3rd party NVMe : supported
I/O Controller : /sys/devices/pci0000:97/0000:97:00.5 (VMD)
NVMe under VMD : /dev/nvme11n1 (PHLN843500CR6P4EGN-2)
NVMe under VMD : /dev/nvme10n1 (PHLN843500CR6P4EGN-1)
I/O Controller : /sys/devices/pci0000:37/0000:37:00.5 (VMD)
NVMe under VMD : /dev/nvme1n1 (PHLE7134002M2P0IGN)
I/O Controller : /sys/devices/pci0000:48/0000:48:00.5 (VMD)
NVMe under VMD : /dev/nvme3n1 (BTLN90550F7B1P6AGN)
NVMe under VMD : /dev/nvme2n1 (PHLF730000Y71P0GGN)
NVMe under VMD : /dev/nvme5n1 (PHLN929100AH1P6AGN)
NVMe under VMD : /dev/nvme4n1 (BTLF7320075H1P0GGN)
I/O Controller : /sys/devices/pci0000:59/0000:59:00.5 (VMD)
NVMe under VMD : /dev/nvme6n1 (PHAB9435004C7P6GGN)
NVMe under VMD : /dev/nvme9n1 (PHAL02860029800LGN)
NVMe under VMD : /dev/nvme8n1 (PHAC0301009X3P8AGN)
NVMe under VMD : /dev/nvme7n1 (PHAL029300AE1P6MGN)
I/O Controller : /sys/devices/pci0000:80/0000:80:00.5 (VMD)
NVMe under VMD : /dev/nvme12n1 (PHLF64120016480AGN)
```

Note: The version represented in this output is the version of the platform's UEFI Intel® VROC, not to be confused with the version of *mdadm* used.

The command returns:

- Intel® VROC capabilities (supported RAID properties and platform details).
- List of I/O controllers associated with Intel® VROC capability.
- List of drives associated with I/O controller.

4.2 Creating Intel® VROC RAID Volume

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

4.2.1 IMSM Container Device Creation

To create an Intel® VROC RAID volume, the container device needs to be created first. A container is a collection of drives that are managed as a set for RAID volume creation with same metadata format. Intel® VROC uses IMSM metadata. A container device with IMSM metadata should be created with the following steps.

Choose the right block device names of the drives that will be used for creating an Intel® VROC RAID. For NVMe drives, Intel® VROC supports creating RAID volume on NVMe drives attached to different VMD controllers. For SATA drives, Intel® VROC supports creating RAID volume on drives attached to the same I/O controller.

Note: Intel® VROC NVMe RAID spanned across different VMD controllers doesn't apply to OS bootable volume.

Create IMSM container device. In this example, "imsm0" is the container name that will be created under the **/dev/md** directory.

Example 2: Create IMSM container

```
# mdadm --create /dev/md/imsm0 --metadata=imsm --raid-devices=4 /dev/nvme0n1 /dev/nvme1n1
/dev/nvme2n1 /dev/nvme3n1
or,
# mdadm --create /dev/md/imsm0 --metadata=imsm --raid-devices=4 /dev/nvme[0-3]n1
```

The command above creates a RAID container with IMSM metadata format. The device node for the container will be **/dev/md/imsm0**. In this example, drives **nvme0n1**, **nvme1n1** **nvme2n1** and **nvme3n1** are used for this RAID container and the total number of drives is 4.

Note: The bash wildcard expressions can be used to specify the range of drives for example: **/dev/nvme[0-3]n1** or **/dev/nvme{0,1,2,3}n1**. Other examples can be found in the bash manual.

Expected output:

```
mdadm: container /dev/md/imsm0 prepared.
```

Note: *mdadm* is able to detect previously existed metadata on the drives (e.g., GPT) and it may ask for confirmation. To avoid that, the option **--run** can be used.

Note: The container name ("imsm0" in this example) is not a permanent name. After the system reboots, the name will be changed to the default **imsmX**, where X is the number assigned sequentially. The permanent name can be defined in the **mdadm.conf** configuration file.

4.2.2 RAID Volume Creation

To create a RAID Volume, the IMSM container device must exist and be active. The following command can be used for raid volume creation. In this example a RAID 5 with name "volume" is created within the "imsm0" container:

```
# mdadm --create /dev/md/volume --level=5 --raid-devices=4 /dev/md/imsm0
```

The command creates a RAID 5 volume within container "imsm0". The RAID 5 volume device node, **/dev/md/volume** in this case, will be created. Four drives (**/dev/nvme[0-3]n1**) assigned to the IMSM container in section 4.2.1 are used for creating the RAID 5 volume with all available space by default.

Expected output:

```
mdadm: array /dev/md/volume started.
```

4.2.3 Intel® Matrix RAID Volume Creation

Intel® Matrix RAID is a combination of two RAID volumes inside one RAID container. The first step is the same as described in section 4.2.1 to create an IMSM container device. After that, the first RAID volume should be created with the **--size** parameter in order to leave free space for the second volume. Both RAID volumes must use the same member drives within the IMSM container.

In this example, a RAID 10 with name "volume0" is created with size of 10GiB, within the "imsm0" container:

```
# mdadm --create /dev/md/volume0 --level=10 --size=10G --raid-devices=4 /dev/md/imsm0
```

Expected output:

```
mdadm: array /dev/md/volume0 started.
```

Now, the second volume can be created.

In this example a RAID 5 with name "volume1" is created, within the same "imsm0" container. The second RAID 5 volume uses the rest of available space on the drives.

```
# mdadm --create /dev/md/volume1 --level=5 --raid-devices=4 /dev/md/imsm0
```

Expected output:

```
mdadm: array /dev/md/volume1 started.
```

4.2.4 RAID Volume Creation Parameters

There are several optional parameters which can be used during Intel® VROC RAID creation. These parameters are tested and verified for Intel® VROC Linux*. The open source *mdadm* utility may offer other parameters which may not be compatible with Intel® VROC.

Table 4-1. RAID Volume Creation Customizable Parameters

Parameter	Short Version	Definition
`--bitmap=`	`-b`	Default is `none`. Can be set to `internal` to enable internal bitmap feature.
`--chunk=`	`-c`	Specifies chunk (strip) size, in kibibytes by default. Meaningful for RAID 0, RAID 5 and RAID 10. Must be power of 2. Optimal chunk size should be considered depending on expected workload profiles.
`--consistency-policy=`	`-k`	Meaningful only for RAID 5. Default is `resync`. Can be set to `ppl` to enable RWH Closure mechanism.
`--force`	`-f`	Used to indicate that non usual operation as intentional. It allows to: • Create container with one drive. • RAID 0 volume with one drive.
`--level=`	`-l`	Specifies the RAID level. The options supported by Intel® VROC RAID are `0`, `1`, `5`, `10`.
`--metadata=`	`-e`	Specifies metadata to be used during creation. Meaningful only for container creation. Must be set to `imsm`.
`--raid-devices=`	`-n`	Number of devices to be used in the array. Must be set as follows: • For RAID1 must be set to `2`. • For RAID10 must be set to `4`.
`--run`	`-R`	Flag used to auto agree if confirmation is required.
`--size=`	`-z`	Specifies the size (by default in kibibyte) of space dedicated on each disk to the RAID volume. This must be a multiple of the chunk size. By default, all available space will be used. Following suffixes can be used to specify the unit of size: • `M` for Megabytes. • `G` for Gigabytes. • `T` for Terabytes.

Note: Intel® VROC support RAID creation only with names defined as `/dev/md/name` or just `name`. Other possibilities are not supported and may cause bugs or unproper behavior.

4.2.5 Additional Intel® VROC RAID Volume Creation Examples

Example 1: Create a 2-drive RAID 0 volume:

```
# mdadm --create /dev/md/imsm0 /dev/nvme[0-1]n1 --raid-devices=2 --metadata=imsm
# mdadm --create /dev/md/md0 /dev/md/imsm0 --raid-devices=2 --level=0
```

Example 2: Create a 1-drive RAID 0 volume with force parameter:

```
# mdadm --create /dev/md/imsm0 /dev/nvme0n1 --raid-devices=1 --metadata=imsm --force
# mdadm --create /dev/md/md0 /dev/md/imsm0 --raid-devices=1 --level=0 --force
```

Example 3: Create a 2-drive RAID 1 volume with 100G size:

```
# mdadm --create /dev/md/imsm0 /dev/nvme[0-1]n1 --raid-devices=2 --metadata=imsm
# mdadm --create /dev/md/md0 /dev/md/imsm0 --raid-devices=2 --level=1 --size=100G
```

Example 4: Create a 3-drive RAID 5 volume:

```
# mdadm --create /dev/md/imsm0 /dev/nvme[0-2]n1 --raid-devices=3 --metadata=imsm
# mdadm --create /dev/md/md0 /dev/md/imsm0 --raid-devices=3 --level=5
```

Example 5: Create a 3-drive RAID 5 volume with 64k chunk size and PPL enabled:

```
# mdadm --create /dev/md/imsm0 /dev/nvme[0-2]n1 --raid-devices=3 --metadata=imsm
# mdadm --create /dev/md/md0 /dev/md/imsm0 --raid-devices=3 --level=5 --chunk=64 --
consistency-policy=ppl
```

Example 6: Create a 4-drive RAID 10 volume:

```
# mdadm --create /dev/md/imsm0 /dev/nvme[0-3]n1 --raid-devices=4 --metadata=imsm
# mdadm --create /dev/md/md0 /dev/md/imsm0 --raid-devices=4 --level=10
```

Example 7: Create a 4-drive Intel® Matrix RAID with RAID 5 and RAID 10 volumes:

```
# mdadm --create /dev/md/imsm0 /dev/nvme[0-3]n1 --raid-devices=4 --metadata=imsm
# mdadm --create /dev/md/md0 /dev/md/imsm0 --raid-devices=4 --level=5 --size=100G
# mdadm --create /dev/md/md1 /dev/md/imsm0 --raid-devices=4 --level=10
```

4.3 Reporting Intel® VROC RAID Information

After an Intel® VROC RAID volume is created, there are multiple ways to get information of RAID volumes in Linux*. This section will show some basic methods to check and get Intel® VROC RAID information.

4.3.1 Block Device of Intel® VROC RAID Volume

After creating an Intel® VROC RAID container or volume device, the Linux* block device is created under the **/dev** directory with name **mdXXX**, where XXX is the number allocated automatically starting from 127. The user defined name specified when creating the RAID will be represented in the **/dev/md** directory as a symbolic link to the **/dev/mdXXX** block device. The following is the example of RAID container and volume devices named as "imsm" and "volume", which are linked to **/dev/md127** and **/dev/md126** block devices respectively.

```
# ls -l /dev/md
total 0
lrwxrwxrwx. 1 root root 8 May 25 15:03 imsm -> ../md127
lrwxrwxrwx. 1 root root 8 May 25 15:03 volume -> ../md126
```

Note: The **/dev/mdXXX** block device name is not a persistent name. The XXX number may change after system reboots. User defined name in the **/dev/md** directory can be persistent when a configuration file is created with the specific array name defined. For details, see Section 4.4 Creating RAID Configuration File.

Note: After creating an Intel® VROC RAID volume device, the block device of the RAID member drive is still visible and accessible to the user. For that reason, other system utilities (e.g., *lsblk*) may export the device to the user. This is the current Intel® VROC Linux* product design.

4.3.2 Retrieve RAID Status through /proc/mdstat

/proc/mdstat is a Linux* special file for the user to read the status of all the containers and RAID volumes in the system.

```
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4] [raid1]
md126 : active raid5 nvme6n1[3] nvme5n1[2] nvme4n1[1] nvme3n1[0]
      125958144 blocks super external:/md127/0 level 5, 128k chunk, algorithm 0 [4/4]
[UUUU]

md127 : inactive nvme6n1[3](S) nvme3n1[2](S) nvme5n1[1](S) nvme4n1[0](S)
      4420 blocks super external:imsm

unused devices: <none>
```

In the above example, the content of **/proc/mdstat** is presented with Intel® VROC RAID 5 volume (md126) and Intel® VROC IMSM container (md127). Here are some explanations of each field displayed on the md126 and md127 devices.

Table 4-2. Explanation of md126 Global Properties

md126:	active	raid5	nvme6n1[3] nvme5n1[2] nvme4n1[1] nvme3n1[0]
MD block device name	Array state	RAID Level 5	RAID Volume member devices

Table 4-3. Explanation of md126 Additional Properties

125958144 blocks	super external:/md127/0	level 5	128k chunk	algorithm 0 [4/4] [UUUU]
Volume size in sectors.	RAID metadata details with container name and array index.	RAID level 5	Chunk size	Additional information about member disk availability

Note: Some parameters printed in **/proc/mdstat** are determined by the Linux* MD RAID personality, which could be different across different RAID levels.

The next md127 is the container device, which just represents the metadata. Intel® VROC RAID uses IMSM metadata. This can be determined by the string **super external:imsm** printed out in the second line of md127.

4.3.3 Extracting detailed RAID information

The *mdadm* utility offers a special command to extract all RAID array information from the system. To extract all RAID details the following command is used:

```
# mdadm --detail /dev/md126
/dev/md126:
         Container : /dev/md/imsm0, member 0
        Raid Level : raid5
        Array Size : 7814025216 (7.28 TiB 8.00 TB)
     Used Dev Size : 3907012608 (3.64 TiB 4.00 TB)
      Raid Devices : 3
     Total Devices : 3

             State : clean
```

```
    Active Devices : 3
   Working Devices : 3
    Failed Devices : 0
     Spare Devices : 0

            Layout : left-asymmetric
        Chunk Size : 128K

Consistency Policy : resync


              UUID : 3228d4e1:d3d49830:2e9694c9:a2512524
    Number   Major   Minor   RaidDevice State
       0     259       0        0      active sync   /dev/nvme1n1
       1     259       3        1      active sync   /dev/nvme3n1
       2     259       1        2      active sync   /dev/nvme2n1
```

4.3.4 Reading Intel® VROC RAID Metadata

When an Intel® VROC RAID is created, IMSM metadata is stored on each RAID member disk. The Linux* *blkid* utility can understand IMSM metadata, and display drives with IMSM metadata as **`isw_raid_member`**. The following is an example showing four NVMe devices with IMSM metadata stored.

```
# blkid
/dev/nvme3n1: TYPE="isw_raid_member"
/dev/nvme4n1: TYPE="isw_raid_member"
/dev/nvme5n1: TYPE="isw_raid_member"
/dev/nvme6n1: TYPE="isw_raid_member"
```

The user can also read RAID metadata directly on the container device or any RAID member disk, by using the command **`mdadm --examine`**. The following is an example of reading IMSM metadata from the **`/dev/nvme1n1`** device, which is a RAID member disk of a 3 disk RAID 5 volume with name of “vol0” and subarray ID 0.

```
# mdadm --examine /dev/nvme1n1
/dev/nvme1n1:
          Magic : Intel® Raid ISM Cfg Sig.
        Version : 1.3.00
    Orig Family : 7fbd42b1
         Family : 7fbd42b1
     Generation : 00000002
  Creation Time : Wed Jul 20 02:01:37 2022
     Attributes : All supported
           UUID : b6232adc:c3658d33:54b04b77:6cffcdac
       Checksum : 334dcb79 correct
    MPB Sectors : 2
          Disks : 3
   RAID Devices : 1

  Disk00 Serial : LJ832405AB4P0DGN
          State : active
             Id : 00000000
    Usable Size : 7814026766 (3.64 TiB 4.00 TB)

[vol0]:
       Subarray : 0
           UUID : 3228d4e1:d3d49830:2e9694c9:a2512524
     RAID Level : 5
        Members : 3
          Slots : [UUU]
    Failed disk : none
      This Slot : 0
    Sector Size : 512
```

```
     Array Size : 15628050432 (7.28 TiB 8.00 TB)
   Per Dev Size : 7814027264 (3.64 TiB 4.00 TB)
  Sector Offset : 0
    Num Stripes : 30523536
     Chunk Size : 128 KiB
       Reserved : 0
  Migrate State : idle
      Map State : normal
    Dirty State : clean
     RWH Policy : off
      Volume ID : 1

  Disk01 Serial : LJ832405HG4P0DGN
          State : active
             Id : 00000000
    Usable Size : 7814026766 (3.64 TiB 4.00 TB)

  Disk02 Serial : LJ832405AC4P0DGN
          State : active
             Id : 00000000
    Usable Size : 7814026766 (3.64 TiB 4.00 TB)
```

4.4 Creating Intel® VROC RAID Configuration File

After creating RAID volumes, it is necessary to create the configuration file to record the existing RAID volumes' information as well as adding specific RAID management policies. This configuration file allows a consistent RAID volume naming in **/dev/md** after system reboots. This reliable and constant name can be used for system auto-configuration (e.g., in **/etc/fstab** for automounting). The MD device nodes in **/dev** are not consistent and may change depend on system enumeration.

The following command extracts system RAID volume information and stores it to the configuration file **/etc/mdadm.conf**.

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

The configuration file is typically stored at the default location of **/etc/mdadm.conf**. Depending on the OS distribution, there may be different file locations and alternative means of saving the configuration file. It is important to reference the **mdadm.conf** man page to determine what applies to your distribution.

4.5 Intel® VROC 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.

4.5.1 Adjusting Initialization/Resync Speed

Linux* by default has a maximum speed of 200MB as the limitation for volume initialization and rebuilding. This was the community standard set based on spinning platter based hard drives. A consensus has not been reached regarding what the standard should be moved to with the advancements in technology.

This value can be modified to prevent the system from taking an undue amount of time initializing or rebuilding a RAID array on modern storage drives. The series of commands below shows a system that is set at the default value, and how that may be modified.

To see what the current rate of speed the system is using:

```
# cat /sys/block/md0/md/sync_speed_max
200000(system)
```

To modify this to a higher value, such as 5GB for NVMe drives:

```
# echo 5000000 > /sys/block/md0/md/sync_speed_max
```

Note: Different values may be appropriate for different drive types. 5GB is reasonable for typical Data Center NVMe SSD models.

4.6 Adding a Hot Spare Drive

Adding a hot 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 hot spare drive. The hot spare drive can also be used for RAID volume’s capacity expansion (refer to Section 6.5 Online Capacity Expansion for more information). The hot spare drives sit idle during normal operations. Intel® VROC Linux* RAID is using *mdadm* with IMSM metadata, the hot spare drive added to a container is dedicated to that specific container. The following command adds a hot spare drive to the designated container **/dev/md/imsm0**.

```
# mdadm --add /dev/md/imsm0 /dev/<nvmeXn1>
```

Note: The metadata UUID for the member drive inside the container as a hot spare will keep being all zero until the drive is re-assigned to be part of the RAID volume. A hot spare drive with zero UUID may be allocated to a separate container after system reboots.

Note: It is recommended to add hot spare drive one by one separately, due to a known limitation of the *mdadm* tool that multiple hot spare drives may not be properly added in one command.

4.7 Configuring Global Hot Spare

By default, the Intel® VROC Linux* RAID hot spare drive is dedicated to a specific container. Only the Intel® VROC RAID volumes inside that container can be automatically reconstructed to the hot spare drive when redundant RAID volume is degraded. Intel® VROC Linux* RAID allows configuring a global hot spare drive that can be used by any degraded Intel® VROC RAID volume on the system for automatic recovery (rebuilding).

The following command adds the policy with the same domain and the same action for all drives to the *mdadm* configuration file, which enables the global hot spare function that allows the spare drives to move from one container to another for rebuilding:

```
# echo "POLICY domain=DOMAIN path=* metadata=imsm action=spare-same-slot" >>
/etc/mdadm.conf
```

After configuring the policy in the configuration file, the *mdmonitor* service should be restarted to take effect. Refer to Section 5.2.2 for command details to restart the *mdmonitor* service.

4.8 Removing Intel® VROC RAID Volumes

Completely removing Intel® VROC RAID in Linux* requires several steps. The following sections illustrate the *mdadm* commands to stop and remove an Intel® VROC RAID volume in Linux*.

4.8.1 Stopping RAID Volume and Container

The first step is to stop the RAID volume and container devices.

Stopping RAID volume or container is basically removing the Linux* MD block device. The Intel® VROC RAID metadata remains on the RAID member drive and the RAID volume can be assembled again. Intel® VROC RAID volume can be stopped only if it is not in use (for example a filesystem is mounted). The Intel® VROC RAID container device can be stopped only if no member RAID volume is active.

The following command can be used to stop the RAID volume or container device:

```
# mdadm --stop /dev/md/name
```

Expected result:

```
mdadm: stopped /dev/md/name
```

Multiple MD devices can be passed to the command **mdadm --stop**, each one will be stopped individually. The sequence of multiple MD devices should matter.

```
# mdadm --stop /dev/md126 /dev/md127
mdadm: stopped /dev/md126
mdadm: stopped /dev/md127
```

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

```
# mdadm --stop --scan
```

4.8.2 Erasing RAID Metadata

Once the Intel® VROC RAID volume and container devices are stopped correctly, the next step to completely remove Intel® VROC RAID in Linux* is to erase Intel® VROC 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 or spare drive. The RAID volumes and container must be stopped and deactivated to run the erase operation.

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

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

Caution: The metadata erase operation is irreversible and may ruin the RAID volume. Run this operation with caution.

4.8.3 Removing One RAID Volume of Intel® Matrix RAID

Intel® Matrix RAID contains two RAID volumes inside one container. This requires a little more care to remove one of the RAID volumes.

First, both RAID volumes in the Intel® VROC Matrix RAID must be stopped. The following example shows how to stop the Intel® VROC RAID volumes "vol0" and "vol1".

```
# mdadm --stop /dev/md/vol0
# mdadm --stop /dev/md/vol1
```

Next, erase the volume metadata to completely remove the volume. The following example shows how to remove the first RAID volume inside the Intel® Matrix RAID ("imsm0").

```
# mdadm --kill-subarray=0 /dev/md/imsm0
```

Note: **`--kill-subarray`** expects the volume index, which should be either 0 or 1 to represent the first or second volume. Refer to Section 4.3.4 to get the volume's subarray index from RAID metadata.

Note: The RAID volume UUID and index will be changed after subarray deletion. If user deletes index 1 subarray, index 0 will keep in the container. If user deletes index 0 subarray, index 1 will become index 0 after deletion.

4.9 Assembling Intel® VROC RAID Volumes

Intel® VROC RAID volumes can be assembled and activated using the *mdadm* utility.

The following command by default 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. In this case, the RAID volume name under the **`/dev/md`** directory will be added "_0" as suffix.

```
# mdadm --assemble --scan
```

Note: If the configuration file exists with wrong or empty RAID volume/container device information, *mdadm* will refuse to assemble it.

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

First, assemble the container device. The following example assembles the container device **`/dev/md/md0`** using the provided list of drives.

```
# mdadm --assemble /dev/md/md0 --metadata=imsm /dev/<member drives>
```

Then, assemble the RAID volume. The following example shows the assembling of the RAID volume **`/dev/md/md0`** within container **`/dev/md/imsm0`**.

```
# mdadm --assemble /dev/md/md0 /dev/md/imsm0
```

4.10 Creating File System on an Intel® VROC RAID Volume

After the RAID volume has been created, a file system can be created to allow the mounting of the RAID volume. The Linux* *mkfs* utility can be used to create the filesystem. The following example shows how an EXT4 filesystem is created on an Intel® VROC RAID volume “md0”.

```
# mkfs.ext4 /dev/md/md0
```

After the file system has been created, it can be mounted to the location of choice. For example, mount the “md0” volume to the **/mnt/data** directory.

```
# mount /dev/md/md0 /mnt/data
```

When configuring the Linux filesystem table (aka fatab or “/etc/fstab” file) for automount of a filesystem on boot, it is worth mentioning a best practice to add “*_netdev*” or “*noauto,x-systemd.automount*” option for mounting the filesystem onto the Intel® VROC RAID volume. Those two options are intended to delay the mount attempt slightly in order to make more time for the RAID member disks and RAID volume device being discovered and ready before systemd tries to mount the filesystem.

Followings are two examples of adding “*_netdev*” and “*noauto,x-systemd.automount*” options in the “/etc/fstab” file.

Option 1: Add “*_netdev*” mount option:

```
UUID=”c2c9651c-bd6c-4f61-ae9d-2b9dd0fca524” /data/vroc/raid10 ext4 defaults,_netdev 0 0
```

Option 2: Add “*noauto,x-systemd.automount*” mount option:

```
UUID=”c2c9651c-bd6c-4f61-ae9d-2b9dd0fca524” /data/vroc/raid10 ext4 defaults,noauto,x-
systemd.automount 0 0
```

4.11 Removing an Active Intel® VROC RAID Member Disk

Intel® VROC Linux* allows users to hot remove an active drive from a RAID volume. Intel® VROC Linux* supports the surprise hot removal of the physical drive directly from the system without any advance notice to the system. The RAID state will become either degraded or failed depending on the RAID level after removing the RAID member disk. If a RAID volume is failed, the RAID volume logical device will be removed from Linux* and it will not be assembled after system reboots.

As for RAID levels with data redundancy (RAID level 1/5/10), Intel® VROC Linux* also supports a more graceful way to remove an active RAID member disk. This basically requires two steps to remove an active drive from the RAID volume and container.

1. Mark the drive as faulty from the volume. The following command marks the drive **nvme0n1** as faulty in the Intel® VROC RAID volume **/dev/md/volume0**. The faulty drive still stays in the container but not in the RAID volume.

```
# mdadm --fail /dev/md/volume0 /dev/nvme0n1
```

2. Remove the faulty drive from the container. The following command needs to be executed.

```
# mdadm --remove /dev/md/imsm0 /dev/nvme0n1
```

Once the logical drive is removed from the Intel® VROC RAID volume with the commands above, the user can remove the physical drive from the system safely.

Note: Intel® VROC Linux* only allows the user to mark the drive as faulty from the redundant RAID volume which is not in the degraded (or double degraded as for RAID 10) mode. In other words, Intel® VROC Linux* prevents the user from failing a RAID volume.

4.12 Recovery of Intel® VROC 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 the 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 of RAID 10 is not possible.

Intel® VROC supports multiple scenarios of recovering a degraded RAID volume.

4.12.1 Auto Rebuilding to a Hot Spare

If there are spare drives available in the container, rebuild of the RAID volume would automatically commence. The following example shows how a degraded RAID 5 volume is automatically rebuilding (resyncing) to the hot spare drive (**nvme2n1**) pre-configured in the container.

/proc/mdstat is showing the recovery progress as well as the resync speed.

```
# cat /proc/mdstat
Personalities : [raid1] [raid6] [raid5] [raid4] [raid0] [raid10]
md126 : active raid5 nvme2n1[3] nvme3n1[2] nvme1n1[0]
      209715200 blocks super external:/md127/0 level 5, 128k chunk, algorithm 0 [3/2] [U_U]
      [=>...................]  recovery =  9.0% (9460412/104857600) finish=1.1min
speed=1351487K/sec

md127 : inactive nvme3n1[3](S) nvme2n1[2](S) nvme1n1[1](S) nvme0n1[0](S)
      4420 blocks super external:imsm

unused devices: <none>
```

The details of the RAID volume also show the rebuilding/recovering progress onto the spare drive **/dev/nvme2n1**.

```
# mdadm --detail /dev/md126
/dev/md126:
         Container : /dev/md/imsm0, member 0
        Raid Level : raid5
        Array Size : 209715200 (200.00 GiB 214.75 GB)
     Used Dev Size : 104857600 (100.00 GiB 107.37 GB)
      Raid Devices : 3
     Total Devices : 3

             State : clean, degraded, recovering
    Active Devices : 2
   Working Devices : 3
    Failed Devices : 0
     Spare Devices : 1

            Layout : left-asymmetric
        Chunk Size : 128K

Consistency Policy : resync

    Rebuild Status : 24% complete
```

```
              UUID : 162f6668:62b72484:f692ed54:cf74395e
    Number   Major   Minor   RaidDevice State
       0     259        1        0      active sync   /dev/nvme1n1
       3     259        2        1      spare rebuilding   /dev/nvme2n1
       2     259        5        2      active sync   /dev/nvme3n1
```

If there is no spare drive pre-configured in the container, a spare drive can also be manually added to start the rebuild process. Refer to the Section 4.6 Adding a Hot Spare Drive for details of how to add a spare drive. The following example shows how a degraded RAID 5 volume md126 starts rebuilding to the hot spare drive **(/dev/nvme0n1**) automatically when it is manually added.

```
# mdadm --detail /dev/md126
/dev/md126:
         Container : /dev/md/imsm0, member 0
        Raid Level : raid5
        Array Size : 209715200 (200.00 GiB 214.75 GB)
     Used Dev Size : 104857600 (100.00 GiB 107.37 GB)
      Raid Devices : 3
     Total Devices : 2

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

            Layout : left-asymmetric
        Chunk Size : 128K

Consistency Policy : resync


              UUID : 162f6668:62b72484:f692ed54:cf74395e
    Number   Major   Minor   RaidDevice State
       0     259        1        0      active sync   /dev/nvme1n1
       3     259        2        1      active sync   /dev/nvme2n1
       -       0        0        2      removed
[root@localhost Intel®]# mdadm --add /dev/md127 /dev/nvme0n1
mdadm: added /dev/nvme0n1
[root@localhost Intel®]# mdadm --detail /dev/md126
/dev/md126:
         Container : /dev/md/imsm0, member 0
        Raid Level : raid5
        Array Size : 209715200 (200.00 GiB 214.75 GB)
     Used Dev Size : 104857600 (100.00 GiB 107.37 GB)
      Raid Devices : 3
     Total Devices : 3

             State : clean, degraded, recovering
    Active Devices : 2
   Working Devices : 3
    Failed Devices : 0
     Spare Devices : 1

            Layout : left-asymmetric
        Chunk Size : 128K

Consistency Policy : resync

    Rebuild Status : 2% complete


              UUID : 162f6668:62b72484:f692ed54:cf74395e
    Number   Major   Minor   RaidDevice State
       0      259      1        0      active sync   /dev/nvme1n1
       3      259      2        1      active sync   /dev/nvme2n1
       4      259      6        2      spare rebuilding   /dev/nvme0n1
```

4.12.2 Auto Rebuilding to a New Drive

There is a scenario when a failed RAID member drive is replaced with a brand-new drive, it can be configured for the degraded RAID volume to start rebuilding automatically when the brand-new drive is hot inserted to the same slot of the failed drive being replaced. To enable this auto-rebuilding scenario, the following steps should be performed.

1. Follow the instructions from Section 4.7 to configure global hot spare policy in the *mdadm* configuration file, which allows the spare to move from one container to another for rebuilding.

Note: The users need to specify the **MAILADDR** in the *mdadm* configuration file in order to run the *mdmonitor* service (simply run the command **#echo “MAILADDR root” >> /etc/mdadm.conf**).

The configuration file in **/etc/mdadm.conf** will look similar to the following:

```
ARRAY metadata=imsm UUID=60582f12:51325766:88172a8a:8424eb16   spares=1
ARRAY /dev/md/1 container=60582f12:51325766:88172a8a:8424eb16 member=0
UUID=0467835f:e74cb807:0c7b1fe2:84ccf80c
ARRAY metadata=imsm UUID=a4a1b4a1:23f34684:de07ba5b:328edbee
ARRAY /dev/md/2 container=a4a1b4a1:23f34684:de07ba5b:328edbee member=0
UUID=39964136:f3fd21ae:e3cf676c:8f73b3fa
POLICY domain=DOMAIN path=* metadata=imsm action=spare-same-slot
```

2. Follow the below two commands to generate the appropriate *udev* rules after changing the *mdadm* configuration file, and reload the *udev* rules to make it take effect.

```
# mdadm --udev-rules > /etc/udev/rules.d/65-md-bare.rules
# udevadm control --reload
```

3. The generated *udev* rule will be added and look like this:

```
# ACTION=="add", SUBSYSTEM=="block", ENV{ID_PATH}=="*", RUN+="/sbin/mdadm --incremental
$env{DEVNAME}"
```

4. Make sure the *mdmonitor* service is running. Refer to the Section 5.2.2 for commands to check the service status and get it started if it is not running.

Now, it is well configured to enable the auto rebuilding to a new drive when it is hot inserted to replace the failed RAID member drive.

Note: The new disk must be “bare”. That means that at least the first and last 4KB space must be all zero. Users can follow the below example to make the NVMe disk **nvme0n1** as the “bare” drive.

```
# dd if=/dev/zero of=/dev/nvme0n1 bs=512 count=8 oflag=direct
# dd if=/dev/zero of=/dev/nvme0n1 bs=512 count=8 seek=$(($(cat /sys/block/nvme0n1/size) -
8)) oflag=direct
```

§§

5 Intel® VROC RAID Logging and Monitoring in Linux*

5.1 Intel® VROC RAID Logging in Linux*

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

5.1.1 Use "demsg" to Retrieve Kernel Logging

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

```
# dmesg
[Thu Aug  4 09:19:52 2022] md/raid1:md126: not clean -- starting background reconstruction
[Thu Aug  4 09:19:52 2022] md/raid1:md126: active with 2 out of 2 mirrors
[Thu Aug  4 09:19:52 2022] md126: detected capacity change from 0 to 107374182400
[Thu Aug  4 09:19:52 2022] md: resync of RAID array md126
[Thu Aug  4 09:21:36 2022] md: md126: resync done.
[Thu Aug  4 09:21:43 2022] md126: detected capacity change from 107374182400 to 0
[Thu Aug  4 09:21:43 2022] md: md126 stopped.
[Thu Aug  4 09:21:43 2022] md: md126 stopped.
[Thu Aug  4 09:21:43 2022] md: md127 stopped.
[Thu Aug  4 09:23:14 2022] md126: detected capacity change from 0 to 16003123642368
[Thu Aug  4 09:23:38 2022] md126: detected capacity change from 16003123642368 to 0
[Thu Aug  4 09:23:38 2022] md: md126 stopped.
[Thu Aug  4 09:23:38 2022] md: md127 stopped.
[Fri Aug  5 01:52:54 2022] md/raid:md126: not clean -- starting background reconstruction
[Fri Aug  5 01:52:54 2022] md/raid:md126: device nvme3n1 operational as raid disk 2
[Fri Aug  5 01:52:54 2022] md/raid:md126: device nvme0n1 operational as raid disk 1
[Fri Aug  5 01:52:54 2022] md/raid:md126: device nvme1n1 operational as raid disk 0
[Fri Aug  5 01:52:54 2022] md/raid:md126: raid level 5 active with 3 out of 3 devices,
algorithm 0
[Fri Aug  5 01:52:54 2022] md126: detected capacity change from 0 to 214748364800
[Fri Aug  5 01:52:54 2022] md: resync of RAID array md126
[Fri Aug  5 01:54:36 2022] md: md126: resync done.
[Fri Aug  5 01:54:54 2022] md/raid:md126: Disk failure on nvme0n1, disabling device.
                           md/raid:md126: Operation continuing on 2 devices.
[Fri Aug  5 01:54:54 2022] md: recovery of RAID array md126
[Fri Aug  5 01:56:41 2022] md: md126: recovery done.
[Fri Aug  5 02:00:20 2022] md/raid:md126: Disk failure on nvme3n1, disabling device.
                           md/raid:md126: Operation continuing on 2 devices.
[Fri Aug  5 02:00:50 2022] md: recovery of RAID array md126
[Fri Aug  5 02:02:46 2022] md: md126: recovery done.
```

5.1.2 Use "journalct" to Retrieve System Journal Logs

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

```
Aug 05 01:52:55 localhost.localdomain kernel: md/raid:md126: not clean -- starting
background reconstruction
Aug 05 01:52:55 localhost.localdomain kernel: md/raid:md126: device nvme3n1 operational as
raid disk 2
Aug 05 01:52:55 localhost.localdomain kernel: md/raid:md126: device nvme0n1 operational as
raid disk 1
```

```
Aug 05 01:52:55 localhost.localdomain kernel: md/raid:md126: device nvme1n1 operational as
raid disk 0
Aug 05 01:52:55 localhost.localdomain kernel: md/raid:md126: raid level 5 active with 3 out
of 3 devices, algorithm 0
Aug 05 01:52:55 localhost.localdomain kernel: md126: detected capacity change from 0 to
214748364800
Aug 05 01:52:55 localhost.localdomain systemd[1]: Starting MD Metadata Monitor on
/dev/md127...
Aug 05 01:52:55 localhost.localdomain systemd[1]: Started MD Metadata Monitor on
/dev/md127.
Aug 05 01:52:55 localhost.localdomain kernel: md: resync of RAID array md126
Aug 05 01:52:58 localhost.localdomain dhclient[6077]: DHCPDISCOVER on ens785f1 to
255.255.255.255 port 67 interval 3 (xid=0x3b9ab34d)
Aug 05 01:53:01 localhost.localdomain dhclient[6077]: DHCPDISCOVER on ens785f1 to
255.255.255.255 port 67 interval 5 (xid=0x3b9ab34d)
Aug 05 01:53:06 localhost.localdomain dhclient[6077]: DHCPDISCOVER on ens785f1 to
255.255.255.255 port 67 interval 8 (xid=0x3b9ab34d)
Aug 05 01:53:14 localhost.localdomain dhclient[6077]: DHCPDISCOVER on ens785f1 to
255.255.255.255 port 67 interval 15 (xid=0x3b9ab34d)
Aug 05 01:53:29 localhost.localdomain dhclient[6077]: DHCPDISCOVER on ens785f1 to
255.255.255.255 port 67 interval 21 (xid=0x3b9ab34d)
Aug 05 01:53:50 localhost.localdomain dhclient[6077]: DHCPDISCOVER on ens785f1 to
255.255.255.255 port 67 interval 9 (xid=0x3b9ab34d)
Aug 05 01:53:59 localhost.localdomain dhclient[6077]: No DHCPOFFERS received.
Aug 05 01:53:59 localhost.localdomain dhclient[6077]: No working leases in persistent
database - sleeping.
Aug 05 01:54:37 localhost.localdomain kernel: md: md126: resync done.
Aug 05 01:54:55 localhost.localdomain kernel: md/raid:md126: Disk failure on nvme0n1,
disabling device.
                                              md/raid:md126: Operation continuing on 2
devices.
Aug 05 01:54:55 localhost.localdomain udisksd[2823]: Unable to resolve
/sys/devices/virtual/block/md126/md/dev-nvme0n1/block symlink
Aug 05 01:54:55 localhost.localdomain kernel: md: recovery of RAID array md126
```

5.1.3 Use Syslog File: /var/log/messages

Below is an example snippet of what the log may look like in **/var/log/messages**:

```
Aug  4 09:05:10 localhost kernel: md126: detected capacity change from 0 to 16003123642368
Aug  4 09:05:23 localhost kernel: md126: detected capacity change from 16003123642368 to 0
Aug  4 09:05:23 localhost kernel: md: md126 stopped.
Aug  4 09:05:23 localhost kernel: md: md127 stopped.
Aug  4 09:06:56 localhost kernel: md/raid:md126: not clean -- starting background
reconstruction
Aug  4 09:06:56 localhost kernel: md/raid:md126: device nvme2n1 operational as raid disk 3
Aug  4 09:06:56 localhost kernel: md/raid:md126: device nvme3n1 operational as raid disk 2
Aug  4 09:06:56 localhost kernel: md/raid:md126: device nvme0n1 operational as raid disk 1
Aug  4 09:06:56 localhost kernel: md/raid:md126: device nvme1n1 operational as raid disk 0
Aug  4 09:06:56 localhost kernel: md/raid:md126: raid level 5 active with 4 out of 4
devices, algorithm 0
Aug  4 09:06:56 localhost kernel: md126: detected capacity change from 0 to 322122547200
Aug  4 09:06:56 localhost systemd[1]: Starting MD Metadata Monitor on /dev/md127...
Aug  4 09:06:56 localhost systemd[1]: Started MD Metadata Monitor on /dev/md127.
Aug  4 09:06:56 localhost kernel: md: resync of RAID array md126
Aug  4 09:09:15 localhost kernel: md: md126: resync done.
Aug  4 09:09:24 localhost kernel: md126: detected capacity change from 322122547200 to 0
Aug  4 09:09:24 localhost kernel: md: md126 stopped.
Aug  4 09:09:24 localhost systemd[1]: mdmon@md127.service: Succeeded.
Aug  4 09:09:24 localhost kernel: md: md126 stopped.
Aug  4 09:09:24 localhost kernel: md: md127 stopped.
Aug  4 09:10:23 localhost kernel: md/raid10:md126: not clean -- starting background
reconstruction
Aug  4 09:10:23 localhost kernel: md/raid10:md126: active with 4 out of 4 devices
Aug  4 09:10:23 localhost kernel: md126: detected capacity change from 0 to 214748364800
Aug  4 09:10:23 localhost systemd[1]: Starting MD Metadata Monitor on /dev/md127...
Aug  4 09:10:23 localhost systemd[1]: Started MD Metadata Monitor on /dev/md127.
Aug  4 09:10:23 localhost kernel: md: resync of RAID array md126
Aug  4 09:12:00 localhost kernel: md: md126: resync done.
Aug  4 09:16:32 localhost kernel: md126: detected capacity change from 214748364800 to 0
Aug  4 09:16:32 localhost kernel: md: md126 stopped.
Aug  4 09:16:32 localhost systemd[1]: mdmon@md127.service: Succeeded.
Aug  4 09:16:32 localhost kernel: md: md126 stopped.
Aug  4 09:16:32 localhost kernel: md: md127 stopped.
Aug  4 09:19:53 localhost kernel: md/raid1:md126: not clean -- starting background
reconstruction
Aug  4 09:19:53 localhost kernel: md/raid1:md126: active with 2 out of 2 mirrors
Aug  4 09:19:53 localhost kernel: md126: detected capacity change from 0 to 107374182400
Aug  4 09:19:53 localhost systemd[1]: Starting MD Metadata Monitor on /dev/md127...
Aug  4 09:19:53 localhost systemd[1]: Started MD Metadata Monitor on /dev/md127.
Aug  4 09:19:53 localhost kernel: md: resync of RAID array md126
Aug  4 09:21:37 localhost kernel: md: md126: resync done.
Aug  4 09:21:44 localhost kernel: md126: detected capacity change from 107374182400 to 0
Aug  4 09:21:44 localhost kernel: md: md126 stopped.
Aug  4 09:21:44 localhost systemd[1]: mdmon@md127.service: Succeeded.
Aug  4 09:21:44 localhost kernel: md: md126 stopped.
Aug  4 09:21:44 localhost kernel: md: md127 stopped.
```

5.2 Intel® VROC RAID Monitoring in Linux*

Once the Intel® VROC RAID volume is created and started, a RAID monitoring service (daemon) called *mdmonitor* is started automatically to monitor Intel® VROC RAID events and trigger actions if configured.

5.2.1 Use mdadm Monitoring Daemon

mdadm monitoring can be started with the following command:

```
# 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.

It is mandatory to define the email address in the **mdadm.conf** file before starting the *mdadm* monitoring. The following command shows an example to configure the email address in the configuration file.

```
# echo “MAILADDR root” >> /etc/mdadm.conf
```

5.2.2 Use systemctl for RAID Monitoring

The *mdmonitor* daemon is configured to work with *systemd*. To see if the *mdmonitor* services are running as anticipated, you may use the following command to check the status:

```
# systemctl status mdmonitor.service
```

The following command can be used to manually start the service if you find that it is not running:

```
# systemctl start mdmonitor.service
```

Alternatively, the service can be restarted by running the following command:

```
# systemctl restart mdmonitor.service
```

The service can also be enabled to start immediately following each reboot. Use this command:

```
# systemctl enable mdmonitor.service
```

To stop the service, the following command is used:

```
# systemctl stop mdmonitor.service
```

5.3 Intel® VROC RAID Alerts in Linux*

Intel® VROC reports the following RAID alerts through the monitoring service in Linux*. Users can develop their own program to hook the monitoring service to receive and handle those RAID alerts.

Table 5-1. Intel® VROC RAID Alerts in Linux*

VROC Alert/Event	Severity	Description
Fail	Critical	A member drive of RAID is faulty.
FailSpare	Critical	The spare drive which was being rebuilt to has failed.
DeviceDisappeared	Critical	A RAID volume is disappeared (or removed).
DegradedArray	Critical	A newly noticed RAID array appears to be degraded.
RebuildStarted	Warning	A degraded RAID started rebuilding (recovery).
RebuildNN	Warning	Notification of rebuilding progress, NN is two-digit number (e.g., 20, 40, …) which indicates the rebuild has passed that many percent of the total.
RebuildFinished	Warning	The rebuilding of a degraded RAID is completed or aborted.
SparesMissing	Warning	One or more spare drives defined in the **`mdadm.conf`** file is disappeared (or removed).
SpareActive	Information	The spare drive which was being rebuilt to has been successfully rebuilt and made active.
NewArray	Information	A new RAID array has been detected.
MoveSpare	Information	A spare drive has been moved from one array to another.

5.4 Develop a Program to Handle Intel® VROC Alerts

The Intel® VROC RAID monitoring service allows any user-defined program to be hooked to receive and handle the RAID alerts. The system administrator should configure the **`/etc/mdadm.conf`** file to add the user-defined program which will be called by the monitoring service whenever an alert is detected.

The Intel® VROC RAID monitoring service will call the user-defined program and pass two or three parameters to it when an alert occurs. The first parameter is the event name. The second parameter is the RAID volume device name. There will be the third parameter when the event is related to the spare device or RAID member device.

The following is the bash script example of a user-defined program, which shows how to handle the alerts the Intel® VROC engine reports. In this example, the program simply prints the event messages to the **/tmp/vroc_alerts.log** file when it receives an event. The user can develop their own method of handling those Intel® VROC alerts.

```
#!/bin/bash

event=$1
md_device=$2
device=$3

case $event in
    DegradedArray)
        msg="$md_device is running in the Degraded MODE"
        ;;
    DeviceDisappeared)
        msg="$md_device has disappeared"
        ;;
    Fail)
        msg="$md_device had a failed member device: $device"
        ;;
    FailSpare)
        msg="$md_device: Spare device ($device) FAIL during rebuild"
        ;;
    RebuildStarted)
        msg="Recovery/Rebuilding of $md_device has started"
        ;;
    Rebuild??)
        msg="$md_device REBUILD is now $(echo $event|sed 's/Rebuild//')% complete"
        ;;
    RebuildFinished)
        msg="Rebuild of $md_device is completed or aborted"
        ;;
    SpareActive)
        msg="$device has become an ACTIVE COMPONENT of $md_device"
        ;;
    NewArray)
        msg="$md_device has been detected"
        ;;
    MoveSpare)
        msg="SPARE device $device has been MOVED to a new array :$md_device"
        ;;
    SparesMissing)
        msg="$md_device is MISSING one or more SPARE devices"
        ;;
    TestMessage)
        msg="TEST MESSAGE generated for $md_device"
        ;;
esac

# In this example, we just send the event message to the tmp log.
echo "[$(date -u)] $msg" >> /tmp/vroc_alerts.log
```

The system administrator can place the user-defined program into the **/usr/sbin** directory. For example: **/usr/sbin/vroc_linux_events_handler.sh**.

Then, edit the **/etc/mdadm.conf** file and add the following line. This is the step to enable the Intel® VROC RAID monitoring service to call this **PROGRAM** when the RAID alert occurs.

```
PROGRAM /usr/sbin/vroc_linux_events_handler.sh
```

Here is the example of the **mdadm.conf** file with the user-defined program added:

```
# cat /etc/mdadm.conf
ARRAY metadata=imsm UUID=f69f9275:68fce440:3420da7a:48e2a723
ARRAY /dev/md/vol0 container=f69f9275:68fce440:3420da7a:48e2a723 member=0
UUID=06c1975e:2c160226:ef62cbc6:b42e4570
POLICY domain=DOMAIN path=* metadata=imsm action=spare-same-slot
PROGRAM /usr/sbin/vroc_linux_events_handler.sh
```

The following is the example of messages the above example program printed to the log file when receiving the Intel® VROC alerts:

```
# cat /tmp/vroc_alerts.log
[Tue Feb 21 01:59:28 UTC 2023] Rebuild of /dev/md/vol0 is completed or aborted
[Tue Feb 21 01:59:28 UTC 2023] /dev/md/vol0 has disappeared
[Tue Feb 21 02:13:47 UTC 2023] /dev/md/vol0 REBUILD is now 21% complete
[Tue Feb 21 02:29:53 UTC 2023] /dev/md/vol0 REBUILD is now 40% complete
[Tue Feb 21 02:43:53 UTC 2023] /dev/md/vol0 REBUILD is now 60% complete
[Tue Feb 21 02:56:54 UTC 2023] /dev/md/vol0 REBUILD is now 80% complete
[Tue Feb 21 03:10:08 UTC 2023] Rebuild of /dev/md/vol0 is completed or aborted
[Wed Feb 22 02:44:28 UTC 2023] /dev/md/vol0 had a failed member device: /dev/nvme7n1
[Wed Feb 22 02:47:01 UTC 2023] Recovery/Rebuilding of /dev/md/vol0 has started
```

§§

6 Intel® VROC RAID Advanced Usages in Linux*

This chapter describes various additional commands used for Intel® VROC RAID advanced management in Linux*. The advanced usages of Intel® VROC RAID in Linux* include Online Capacity Expansion (OCE), RAID level migration, changing RAID chunk size, as well as enabling Partial Parity Log (PPL) for RAID 5 write hole closure and enabling write-intent bitmap function. Some advanced changes to Intel® VROC RAID may result in a RAID reshape or resync process in the background. Even though the data remains intact for all those advanced usages, it is strongly recommended to back up the user data before performing those advanced operations.

6.1 Changing RAID Volume Name

Any changes to the RAID volume should be performed when the volume status is inactive.

First, refer to Section 4.3.2 to check the status of the Intel® VROC RAID volume. If the RAID volume status is active, refer to Section 4.8.1 to stop it.

Then, use the following command to change the Intel® VROC RAID volume name. In this example, **/dev/md/imsm** is the container device that this change is applied to. The name of subarray 0 is updated to be "vol".

```
# mdadm --update-subarray=0 --update=name --name=vol /dev/md/imsm
```

Note: The **--update-subarray=** parameter expects the index of the volume inside the container, which is starting from 0. If there is more than one volume in the container, the index should be set accordingly. The subarray index can be found from the Intel® VROC RAID metadata (see Section 4.3.4).

Expected result:

```
mdadm: Updated subarray-0 name from /dev/md/imsm, UUIDs may have changed
```

Note: If the RAID configuration file is created by following the commands in Section 4.4, it is necessary to change the name and UUID in the configuration file manually.

Lastly, the RAID volume with the updated name can now be assembled. Refer to Section 4.9 for the commands to assemble a RAID volume.

6.2 Enabling and Disabling PPL for Intel® VROC RAID 5 RWH Protection

RAID Write Hole (RWH) is a fault scenario, related to parity-based RAID 5. It occurs when a power-failure/crash and a drive failure occur at the same time or very close to each other. Unfortunately, these system crashes and disk failures are correlated events that may cause silent data corruption.

Partial Parity Log (PPL) is a feature available for Intel® VROC RAID 5 arrays designated to close RWH. Additionally, with PPL enabled resync of the array is not needed after dirty shutdown.

PPL is disabled by default unless it is explicitly enabled when creating RAID 5 using *mdadm* commands. Check Section 4.2.5 for the example of creating RAID 5 with PPL enabled. It is also supported to enable PPL for an active RAID 5 volume. The following command demonstrates how to enable PPL for a RAID 5 volume named **/dev/md/vol**:

```
# mdadm --grow --consistency-policy=ppl /dev/md/vol
```

On success, the above command doesn't return any output. The result can be verified by checking the details of the volume (see Section 4.3.3). As for a successful enabling of PPL, the "Consistency Policy" in the volume details should be set to "ppl".

To disable PPL on an active RAID 5 volume (**/dev/md/vol**), the following command can be used:

```
# mdadm --grow --consistency-policy=resync /dev/md/vol
```

On success, the above command doesn't return any output. The result can be verified by checking the details of the volume (see Section 4.3.3). As for a successful disabling of PPL, "Consistency Policy" should be set to "resync".

Note: The default Consistency Policy is "resync".

6.3 Enabling and Disabling Write-Intent Bitmap

Bitmap is a RAID feature that records blocks of data to which the changes are made. This way there is no need to sync the whole array but just the blocks that changed. This feature can be useful when system reboots after dirty shutdown. Bitmap works only on RAID levels with redundancy.

The bitmap feature can be enabled when creating the RAID volume. The following is an example to create an Intel® VROC RAID 5 volume with bitmap feature enabled:

```
# mdadm --create /dev/md/imsm0 /dev/nvme[0-2]n1 --raid-devices=3 --metadata=imsm
# mdadm --create /dev/md/r5 /dev/md/imsm0 --raid-devices=3 --level=5 --bitmap=internal
```

The bitmap feature can also be enabled or disabled on an active Intel® VROC RAID volume. The first step is to stop the active RAID volume because it is an offline operation. Remember to save the RAID configuration beforehand (see Section 4.4) in order to assemble the volume again afterwards. The following example lists the commands to enable/disable the bitmap feature.

This is an example to enable write-intent bitmap on an active RAID volume:

```
# mdadm --stop /dev/md/<volume_name>
# mdadm --update-subarray=<subarray_index> --update=bitmap /dev/md/<container_name>
# mdadm --assemble /dev/md/<volume_name>
```

Note: The **--update-subarray=** parameter expects the index of the volume inside the container, which is starting from 0. If there is more than one volume in the container, the index should be set accordingly. The subarray index can be found from the Intel® VROC RAID metadata (see Section 4.3.4).

This is an example to disable write-intent bitmap on an active RAID volume:

```
# mdadm --stop /dev/md/<volume_name>
# mdadm --update-subarray=<subarray_index> --update=no-bitmap /dev/md/<container_name>
# mdadm --assemble /dev/md/<volume_name>
```

Note: *mdadm* GROW operations cannot be executed on volumes with bitmap enabled. Bitmap must be disabled first, execute grow operation and enable bitmap again.

Note: Bitmap and PPL cannot be enabled simultaneously for Intel® VROC RAID 5.

There are several ways to check whether bitmap is enabled or not:

1. By reading the **/proc/mdstat** file:

```
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md126 : active (auto-read-only) raid5 nvme1n1[2] nvme3n1[1] nvme2n1[0]
      1953513472 blocks super external:/md127/0 level 5, 128k chunk, algorithm 0 [3/3]
[UUU]
      bitmap: 0/8 pages [0KB], 65536KB chunk

md127 : inactive nvme1n1[2](S) nvme2n1[1](S) nvme3n1[0](S)
      15603 blocks super external:imsm

unused devices: <none>
```

   a. When bitmap is enabled, an additional line starting with bitmap is shown. The bitmap information in the **/proc/mdstat** file is as follows:

bitmap:	0/8 pages [0KB],	65536KB chunk
Bitmap line header	How many pages are allocated and used, as well as the allocated size	Size of the chunk data mapped into each bit in the bitmap

2. By checking RAID volume details (see Section 4.3.3). In the detailed information of a RAID volume, the enabled bitmap information is shown as follows:

```
Intent Bitmap : Internal
Consistency Policy : bitmap
```

   a. If bitmap is disabled, the “Intent Bitmap” field is not visible, and “Consistency Policy” field shows “resync” or “ppl”.

3. By reading the RAID metadata (see Section 4.3.4). If the bitmap is enabled, the following can be observed in the RAID metadata:

```
RWH Policy : Write-intent bitmap
```

6.4 Changing RAID Chunk Size

Caution: Even though the data remains intact, it is strongly recommended to back up the user data before changing the RAID chunk size.

A RAID array’s chunk size defines the smallest amount of data per write operation that should be written to each individual disk. Depending on the specific requirements of a user workload as well as the drive’s specification, users may need to change the RAID chunk size to fully optimize the RAID performance.

Intel® VROC Linux* supports changing the RAID chunk size on RAID level 0 and 5 with the supported value from 4KB to 128KB. The default chunk size is 128KB.

To change the chunk size of the RAID volume, the following command can be used. In the following example, the chunk size of a RAID volume (**/dev/md/volume**) is changed to 64KB.

```
# mdadm --grow /dev/md/volume --chunk=64
```

By default, the chunk size is specified in kilobytes. Adding the `M` specifier signifies megabytes. Provided chunk size must be a power of 2, with minimal chunk size being 4KB.

Expected output:

```
mdadm: level of /dev/md/volume changed to raid4
```

The RAID reshaping process will be performed in the background right after executing the above command. When changing the chunk size of RAID 0, a temporary RAID 4 is used during the reshaping process. The migration result can be verified by checking the **`/proc/mdstat`** file (see Section 4.3.2) or reading the RAID volume details (see Section 4.3.3).

Note: Specifying the chunk size is not applicable for RAID 1 and RAID 10.

6.5 Online Capacity Expansion (OCE)

The Online Capacity Expansion (OCE) feature allows the RAID capacity to increase. Online indicates that the operation can be performed while the RAID volume is actively used, without the need to stop the application or RAID volume. This avoids having down time from taking the RAID volume offline and ensure the business continuity. There are two kinds of capacity expansion of an Intel® VROC RAID volume:

1. Increasing the RAID volume capacity to the maximum drive capacity or a larger value than the initial one.
2. Adding one or more drives to expand the RAID volume capacity.

Caution: Data should always be backed up before performing any OCE operation.

Note: Capacity decreasing is not supported by Intel® VROC Linux*.

Note: If the RAID volume’s consistency policy is not the default “resync”, in other words it is “bitmap” or “ppl”, it should be changed to “resync” in order to perform the OCE. After OCE operation, it can be changed back. Refer to Section 6.2 and Section 6.3 for details about PPL and Bitmap.

6.5.1 OCE on Existing RAID Member Disks

By default, when creating an Intel® VROC RAID volume by using *mdadm* commands, the maximum available capacity of the drives will be used if no specific **`--size`** option is added. Users can also create an Intel® VROC RAID volume with designated capacity by adding the **`--size`** option in the *mdadm* command line. Refer to Section 4.2 for details on creating an Intel® VROC RAID.

The expansion of the RAID volume capacity can only be applied to the RAID level 1, 5, and 10 with free space available on each RAID member disk. If there are multiple RAID volumes inside one RAID container, only the last RAID volume can perform OCE. The following shows examples of different options to increase the RAID volume capacity.

Size for capacity expansion can be specified in two ways. Either by using “max”, where volume will grow to the maximum available space or by explicitly using a designated size.

Examples of increasing Intel® VROC RAID volume capacity:

```
# mdadm --grow /dev/md/vol1 --size=max
# mdadm --grow /dev/md/vol2 --size=200G
# mdadm --grow /dev/md/vol3 --size=1T
```

Note: The "size" specified in the above examples is the size on each RAID member disk. The "resync" operation to the expanded space will start right away after executing the above commands.

6.5.2 OCE by Adding New Drives

Intel® VROC Linux* supports OCE by adding new drives on following RAID levels:

- RAID 0
- RAID 5
- Intel® Matrix RAID with RAID 0 and RAID 5
- Intel® Matrix RAID with two RAID 0
- Intel® Matrix RAID with two RAID 5

The OCE operation needs to be done in two steps.

First, follow the instructions in Section 4.6 to add new spare drives to the container. Second, perform the OCE to grow the number of RAID member devices. The following command shows an example of an OCE operation to grow the number of RAID member devices to 4 for each RAID volume in the container named as **/dev/md/imsm**:

```
# mdadm --grow --raid-devices=4 /dev/md/imsm
```

On success, the command doesn't return any output, and the RAID reshaping process will be performed right away in the background. Users can follow the instructions in Section 4.3 to monitor the RAID reshaping status.

Note: To reduce the reshape impact on performance, only one RAID volume can be reshaped at a time. If Intel® VROC Matrix RAID is configured, the reshape on one volume will be delayed and will not be visible in the volume details.

Note: RAID level 0 is temporarily changed to RAID level 4 during the reshape operation.

6.6 RAID Level Migration

The RAID level migration feature allows to change the level of the RAID volume online. All data remains intact.

Caution: Even though the data remains intact, it is strongly advised for the data to be backed up before performing migration operations.

The following table shows the available migration support with Intel® IMSM metadata. You must have the appropriate number of drives necessary for the level you are converting to as spare drives. For information on the required number of disks for each RAID level refer to Section 1.2.

Table 6-1. 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

Note: 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.

6.6.1 RAID 1 to RAID 0 Migration

Two steps are required to migrate from RAID 1 to RAID 0.

1. Migrate from 2-disk RAID 1 to 1-disk RAID 0. The following command shows an example of migrating from a 2-disk RAID 1 volume (**`/dev/md/volume`**) to 1-disk RAID 0:

```
# mdadm --grow /dev/md/volume --level=0
```

The command will return immediately with the following output. The migration result can be verified by checking the **`/proc/mdstat`** file (see Section 4.3.2) or reading the RAID volume details (see Section 4.3.3).

```
mdadm: level of /dev/md/volume changed to raid0
```

2. Use OCE to migrate 1-disk RAID 0 to 2-disk RAID 0. The RAID reshaping process will be performed in the background. Refer to Section 6.5.2 for details about OCE instructions. The following command shows an example of growing 1-disk RAID 0 to 2-disk RAID 0 in the **`/dev/md/imsm`** container:

```
# mdadm --grow /dev/md/imsm --raid-devices=2
```

6.6.2 RAID 10 to RAID 0 Migration

RAID 10 is a combination of two RAID 1 used in RAID 0. Therefore, it has the similar process of RAID 1 to RAID 0 migration. Two steps are required to migrate from RAID 10 to RAID 0.

1. Migrate from 4-disk RAID 10 to 2-disk RAID 0. The following command shows an example of migrating from a 4-disk RAID 10 volume (**`/dev/md/volume`**) to 2-disk RAID 0:

```
# mdadm --grow /dev/md/volume --level=0
```

The command will return immediately with the following output. The migration result can be verified by checking the **`/proc/mdstat`** file (see Section 4.3.2) or reading the RAID volume details (see Section 4.3.3).

```
mdadm: level of /dev/md/volume changed to raid0
```

2. Use OCE to migrate 2-disk RAID 0 to 4-disk RAID 0. The RAID reshaping process will be performed in the background. Refer to Section 6.5.2 for details about OCE instructions. The following command shows an example of growing 2-disk RAID 0 to 4-disk RAID 0 in the **`/dev/md/imsm`** container:

```
# mdadm --grow /dev/md/imsm --raid-devices=4
```

6.6.3 RAID 0 to RAID 10 Migration

Intel® VROC RAID 10 supports 4 drives only. Therefore, it only supports migrating from a 2-disk RAID 0 to 4-disk RAID 10 because RAID 10 is a nested RAID 0 over two RAID Two steps are required to migrate 2-disk RAID 0 to 4-disk RAID 10.

1. Add two spare drives to the container. The following commands show an example of adding two spare drives (**nvme0n1** and **nvme1n1**) to the container device **/dev/md/imsm0** for a 2-disk RAID 0 volume:

```
# mdadm --add /dev/md/imsm0 /dev/nvme0n1
# mdadm --add /dev/md/imsm0 /dev/nvme1n1
```

2. Migrate 2-disk RAID 0 to 4-disk RAID 10. The following command shows an example of migrating a 2-disk RAID0 (**/dev/md/volume**) to 4-disk RAID 10:

```
# mdadm --grow /dev/md/volume --level=10
```

Expected output:

```
mdadm: level of /dev/md/volume changed to raid10
```

The RAID recovery (resyncing) process will be performed in the background right after executing the above command. The migration result can be verified by checking the **/proc/mdstat** file (see Section 4.3.2) or reading the RAID volume details (see Section 4.3.3).

6.6.4 RAID 0 to RAID 5 Migration

The migration from RAID 0 to RAID 5 requires a hot spare drive being added in the container beforehand. Intel® VROC RAID 5 requires the minimum of 3 drives (see Section 1.2.3). Therefore, the minimum supported disk number for RAID 0 to RAID 5 migration is migrating from 2-disk RAID 0 to 3-disk RAID 5. Of course, it can further migrate to RAID 5 with more than 3 member drives by OCE. The following example shows how to migrate from a 2-disk RAID 0 to 3-disk RAID 5.

1. Add one spare drive to the container. The following commands show an example of adding one spare drive (**nvme0n1**) to the container device **/dev/md/imsm0** for a 2-disk RAID 0 volume:

```
# mdadm --add /dev/md/imsm0 /dev/nvme0n1
```

2. Migrate 2-disk RAID 0 to 3-disk RAID 5. The RAID 5 layout must be set to left-asymmetric, which is supported by Intel® VROC. The following command shows an example of migrating a 2-disk RAID 0 (**/dev/md/volume**) to 3-disk RAID 5:

```
# mdadm --grow /dev/md/volume --level=5 --layout=left-asymmetric
```

Expected output:

```
mdadm: level of /dev/md/volume changed to raid5
```

The RAID reshaping process will be performed in the background right after executing above command. The migration result can be verified by checking the **/proc/mdstat** file (see Section 4.3.2) or reading the RAID volume details (see Section 4.3.3).

Note: Intel® VROC only supports the left-asymmetric layout of RAID 5.

6.6.5 RAID 1 to RAID 5 Migration

Migration from 2-disk RAID 1 to 3-disk RAID 5 can be achieved in two steps.

1. Migrate from 2-disk RAID 1 to 2-disk RAID 0, which is described in Section 6.6.1.
2. Migrate from 2-disk RAID 0 to 3-disk RAID 5, which is described in Section 6.6.4.

Users can further grow 3-disk RAID 5 to a larger RAID 5 array with more drives through OCE, which is described in Section 6.5.2.

6.6.6 RAID 10 to RAID 5 Migration

Migration from 4-disk RAID 10 to 3-disk RAID 5 can be achieved in two steps.

1. Migrate from 4-disk RAID 10 to 2-disk RAID 0, which is described in Section 6.6.1.
2. Migrate from 2-disk RAID 0 to 3-disk RAID 5, which is described in Section 6.6.4.

Users can further grow 3-disk RAID 5 to a larger RAID 5 array with more drives through OCE, which is described in Section 6.5.2.

§§

7 Intel® VROC LED Management in Linux*

Intel® VROC LED management in Linux* is achieved using the Linux* *ledmon* and *ledctl* utilities. These utilities have only been verified with Intel® storage controllers.

ledmon can be run as a daemon to constantly monitor the status of drives and Intel® VROC RAID volumes and set the drive LEDs appropriately. Only a single instance of the daemon should be running at a time. On all the modern installations of Linux* OS, *ledmon* will be active and running as a *systemd* service. Should there not be *systemd* active on your installation, you will have to manually add *ledmon* to the **/etc/rc.local** file.

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.

Refer to the following document in GitHub for additional details: ledmon - Intel® LED monitor service for storage enclosures.

7.1 Installing ledmon Package

The *ledmon* package is by default included in all modern Linux* distributions. If it is not by default installed during Linux* OS installation process, the user can manually install it by following the user guide of a specific Linux* distribution. If there is a specific out-of-box release from Intel, follow Intel® VROC Linux* Release Note to install the out-of-box *ledmon* package.

7.2 Configuring ledmon

Note: *ledmon* daemon is typically implemented as a *systemd* service in modern Linux* distributions to allow the auto startup for every system boot.

7.2.1 Configuring ledmon with systemd

Note: The *ledmon* service is loaded by the *systemd* when the system boots up. This section describes how to use the command to check if the `ledmon.service` was loaded from *systemd*.

Check if the `ledmon.service` is running:

```
# systemctl status ledmon.service
```

If the service is not running, double check if the **ledmon.service** file exist in the **/usr/lib/systemd/system** directory. If not, create the new file **/usr/lib/systemd/system/ledmon.service** with the following content:

```
[Unit]
Description=Enclosure LED Utilities
```

```
[Install]
WantedBy=multi-user.target
[Service]
Type=simple
User=root
ExecStart=/usr/sbin/ledmon --foreground
Restart=on-failure
```

Start the *ledmon systemd* service if it is inactive:

```
# systemctl start ledmon.service
```

Enable the *ledmon systemd* service for automated startup of *ledmon* after each OS boots:

```
# systemctl enable ledmon.service
```

7.2.2 Configuring ledmon with /etc/rc.local

If the *systemd* service is not available in your Linux* distribution, to ensure that the *ledmon* daemon starts on each reboot, open the file **/etc/rc.local** using your favorite editor program (e.g., VIM or VI). Insert/add **ledmon** to the final line of the file as shown below:

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

7.3 Using ledctl Utility

Running *ledctl* and *ledmon* can be done concurrently, however, the *ledmon* application has the highest priority when accessing LEDs than other programs. It means some patterns set by *ledctl* may have no effect (except the “locate” pattern).

ledctl is a standalone tool used for controlling LEDs behaviors on the drive slots in a server platform. This is a one-shot command to trigger different types of LED behaviors. The available LED patterns for Intel® VROC are “locate”, “locate_off”, “rebuild”, “failure”, and “off (normal)”. Users must have root privileges to use this tool.

To turn on a locate LED or trigger a “locate” LED pattern, the following command can be used. In the following example, the locate LED of the **/dev/nvme0n1** device is triggered.

```
# ledctl locate=/dev/nvme0n1 --listed-only
```

Note: The **`--listed-only`** option is added to specify the change of the LED status is only applied to the device listed in the command line. Otherwise, the rest of devices in the system will be set LED pattern to normal.

To turn off a locate LED, the following command can be used:

```
# ledctl locate_off=/dev/nvme0n1 --listed-only
```

To trigger a "rebuild" LED pattern, the following command can be used:

```
# ledctl rebuild=/dev/nvme0n1 --listed-only
```

To trigger a "failure/fault" LED pattern, the following command can be used:

```
# ledctl failure=/dev/nvme0n1 --listed-only
```

To turn off the status LED, or set the status LED back to normal, the following command can be used:

```
# ledctl off=/dev/nvme0n1 --listed-only
```

The *ledctl* tool supports changing status LED behaviors on multiple devices at a time. In a single *ledctl* command line, use blank space to separate different LED patterns, and use comma to separate different block device names. The following command shows an example of triggering locate LED on the **`nvme0n1`** and **`nvme1n1`** devices, rebuild LED on the **`nvme2n1`** device and fault LED on the **`nvme3n1`** device.

```
# ledctl locate=/dev/nvme0n1,/dev/nvme1n1 rebuild=/dev/nvme2n1 failure=/dev/nvme3n1 --
listed-only
```

7.4 LED Activity During Hot-Plug Events

After hot-removing a disk and re-inserting it in the same slot, the fault LED may blink for 10 seconds. This is the expected behavior because *ledmon* imposes fail state to slot's LEDs once the drive is removed from the system, but the design backplanes require drive presence in the slot for LED to blink. So, the fail state is always there since the drive is removed, but LED start blinking only if the drive is in the slot (once the new drive is inserted). *Ledmon* will change the state to normal once the hot-plug event is handled.

7.5 Advanced LED Management

Intel® VROC Linux* will support the ability to perform basic LED management configuration of the status LEDs on compatible backplanes. Advanced LED management provides a mechanism for users to customize LED default behavior based on the **`ledmon.conf`** configuration file. The following table includes few example options on different LED status default behaviors. These advanced LED management capabilities can be set in the **`ledmon.conf`** configuration file.

Table 7-1. The Enhanced LEDs Management Capabilities

Event/Parameter	Behavior	Configuration Options	Default Setting
Skip/exclude controller **BLACKLIST**	*Ledmon* will exclude scanning controllers listed on the blacklist. When whitelist is also set in the configuration file, the blacklist will be ignored.	Controller in the blacklist will be excluded from the scanning.	Support all controller
RAID volume is initializing **or verifying or verifying and fixing** **BLINK_ON_INIT**	Rebuild pattern on all drives in RAID volume (until initialization/verify/ verify and fix finishes).	True/Enabled (on all drives) False/Disabled (no drives)	True/Enabled
Set ledmon scan interval **INVERVAL**	The value is given in seconds. Defines the time interval between *ledmon sysfs* scans.	10s (5s is maximum)	10s
RAID volume is rebuilding **REBUILD_BLINK_ON_ALL**	Rebuild pattern on a single drive to which RAID volume rebuilds.	False/Disabled (on one drive) True/Enabled (on all drives)	False/Disabled
RAID volume is migrating **BLINK_ON_MIGR**	Rebuild pattern on all drives in RAID volume (until migration finishes).	True/Enabled (on all drives) False/Disabled (no drives)	True/Enabled
Set ledmon debug level **LOG_LEVEL**	Corresponds with the `–log-level` flag from *ledmon*.	Acceptable values are `quiet`, `error`, `warning`, `info`, `debug`, `all` 0 means `quiet` and 5 means `all`.	2
Set manage RAID member or All **RAID_MEMBRES_ONLY**	If this flag is set to true *ledmon* will limit monitoring only to drives that are RAID members	False / (all RAID member and PT) True / (RAID member only)	False
Limited scans only to following controllers **WHITELIST**	*Ledmon* will limit changing LED state to controllers listed on whitelist.	Limit changing LED state in whitelist controller.	No limit

Intel® VROC LED Management only applies to drives that reside within a supported drive backplane (NVMe and/or SATA). Drives that are connected either by an I/O cable, PCIe add-in card or plugged directly into the motherboard (M.2) will not have LED Management support.

Intel® VROC LED Management does not include drive activity LED management (only status LEDs).

Ledmon can be run as a daemon to constantly monitor the status of drives and software RAID and set the drive LEDs appropriately. The following table lists the additional options of *ledmon* to achieve some advanced LED management features, like accept user defined configuration from a configuration file, enable different levels of debug prints, as well as save logs into user's defined log file, etc.

Usage:

```
# ledmon [OPTIONS]
```

Table 7-2. Ledmon Options Listed

Options	Usage

`-c`
`--config-path=`
	Sets the configuration file path. This overrides any other configuration files. (Although the utility currently does not use a configuration file). The **`/etc/ledcfg.conf`** file is shared by *ledmon* and *ledctl* utilities.

`-l` `--log-path`	Sets the path to a log file. This overrides **`/var/log/ledmon.log`**.

`-t`
`--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` `--help`	Prints help text and exits.
`-v` `--version`	Prints version and license information, then exits.

§§

Appendix A mdadm Quick Start Guide

Usage:

```
# mdadm [mode] <raiddevice> [options] <component-devices>
```

Assemble – Assemble previously created array into an active array

`# mdadm -A -s`	Scan and assembly all the RAID volume based on config file.
`# mdadm -A /dev/md/imsm0 -e imsm /dev/ <member drives>`	Assemble *imsm0* container manually if without config file.
`# mdadm -A /dev/md/md0 /dev/md/imsm0`	Assemble *md0* array manually if without config file.

Create/Grow/Misc – Create new array with metadata/change active size or number of active devices

`# mdadm -C /dev/md/imsm0 /dev/nvme[0-3]n1 -n 4 -e imsm`	Create *imsm0* container with 4 *nvme0-3* drives.
`# mdadm -C /dev/md/md0 /dev/md/imsm0 -n 4 -l 5`	Create *md0* as 4DR5 in *imsm0* container.
`# mdadm -C /dev/md/md0 /dev/md/imsm0 - n 4 -l 5 --consistency-policy=ppl`	Create *md0* as 4DR5 in *imsm0* container with RWH enabled.
`# mdadm -C /dev/md/md0 /dev/md/imsm0 -n 4 -l 5 -z $((100*1024*1024))`	Create 4DR5 with 100GB size for each drive. 300GB size created with this example command.
`# mkfs.ext4 /dev/md/md0`	Create ext4 filesystem in *md0* RAID volume.
`# mount /dev/md/md0 /mnt/<mountpoint>`	Mount *md0* RAID volume in the file system.
`# mdadm -a /dev/md/imsm0 /dev/<nvmeXn1>`	Add hot-spare drive to *imsm0* container.
`# mdadm -Ebs > /etc/mdadm.conf`	Export device metadata to *mdadm.conf* file.
`# mdadm -S /dev/md/md0`	Stop and deactivate *md0* array.
`# mdadm -S /dev/md/imsm0`	Stop and deactivate *imsm0* container.
`# mdadm -S -s`	Stop and deactivate all.
`# mdadm -f /dev/md/md0 /dev/nvme0n1`	Fail *nvme0n1* drive in *md0* array.
`# mdadm -r /dev/md/imsm0 /dev/nvme0n1`	Detached *nvme0n1* drive in imsm0 container.

`# mdadm --zero-superblock /dev/nvme0n1`	Clear and zero the superblock in *nvme0n1* drive.
`# mdadm -G /dev/md/md0 -l 0`	Grow existing 2DR1 to 1DR0.
`# mdadm -G /dev/md/imsm0 -n 2`	Grow 2DR0 to container.
`# mdadm -a /dev/md/imsm0 /dev/nvme2n1`	Add one more drive into container.
`# mdadm -G /dev/md/md0 -l 5 --layout=left-asymmetric`	Grow 2DR0 to 3DR5.
`# mdadm -G /dev/md/md0 --size=128G`	Grow component size *md0* to the 128GB.
`# mdadm -G /dev/md/md0 --consistency-policy=resync`	Switch RWH closure policy to resync on the running md0.
`# mdadm -G /dev/md/md0 --consistency-policy=ppl`	Enable RWH closure mechanism on the running *md0*.

Volume, Member, Hardware Information

`# mdadm --detail-platform`	Print detail of platform's RAID capabilities.
`# mdadm -D /dev/md/md0`	Print detail of *md0* device.
`# mdadm -E /dev/nvme0n1`	Print contents of metadata store in *nvme0n1*.
`# cat /proc/mdstat`	List all active *md* devices with information.

Follow/Monitor/Misc – Monitor one or more devices on state changes

`# systemctl status mdmonitor.service`	Check *mdmonitor* service.
`# systemctl start mdmonitor.service`	Start *mdmonitor* service.
`# systemctl stop mdmonitor.service`	Stop *mdmonitor* service.
`# systemctl enable mdmonitor.service`	Enable *mdmonitor* service for next boot auto start.

§§

Appendix B 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 as shown below:

```
# 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
```

Note: **md127** is **imsm0** and **md126** is **raid1**.

MD devices in **/sys/block** are symbolic links pointing to **/sys/devices/virtual/block**. All MD devices are in the **md** subdirectory in the **/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 the following 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 the **/proc/mdstat** file 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>
```

§§

Appendix C RAID Monitoring Parameters and Events

The following table lists additional command line parameters that can be passed to *mdadm* at startup.

Table C-1. Parameters for mdadm in Monitor Mode

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 a specified file.
`--no-sharing`	`N/A`	This inhibits the functionality for moving spare drives between arrays.

Note: The *mdmonitor* service has to be re-started if *mdmonitor.service* or the *mdadm.conf* file is changed.

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

Table C-2. 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.

Event Name	Description
`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 the **`--test`** flag was used.

Note: **[1]** The events indicated cause an email to be sent. These can also be sent to the *syslog* file.

§§

Appendix D ledmon.conf

NAME

`ledmon.conf` - Configuration file for Intel® Enclosure LED Utilities.

DESCRIPTION

The *ledmon* configuration file allows you to use advanced settings and functions of *Intel® Enclosure LED Utilities*. The global location of the configuration file is **`/etc/ledmon.conf`**. Instead of a global configuration file, you can specify a local config file using the **`-c`** option when running *ledmon*.

SYNTAX

One line should keep exactly one option and value in the configuration file in format: **`OPTION=VALUE`**. Any word that begins with a hash sign (**#**) starts a comment and that word together with the remainder of the line is ignored. Empty lines are allowed. Either single quotes (**'**) or double quotes (**"**) should not be used.

Values that are considered as truth: **`enabled`**, **`true`**, **`yes`**, **`1`**.

Values that are considered as false: **`disabled`**, **`false`**, **`no`**, **`0`**.

See also the examples section.

List of configurable options:

BLACKLIST - Ledmon will exclude scanning controllers listed on blacklist. When whitelist is also set in config file, the blacklist will be ignored. The controllers should be separated by comma (,) character.

BLINK_ON_INIT - Related with RAID Initialization (resync), Verify (check), and Verify and Fix (repair) processes. If value is set to true, status LEDs of all member drives will blink with proper pattern. If RAID volume is under sync process. If value is set to false, processes like init or verifications will not be reported by LEDs. The default value is **`true`**.

BLINK_ON_MIGR - RAID can be migrated between some levels or strip sizes and the flag is related with these processes. Also RAID Grow operation will be reported along with this flag. If value is set to true, status LEDs of all member drives will blink with proper pattern if RAID volume is under reshape. If value is set to false, listed actions will not be reported by LEDs. The default value is **`true`**.

INTERVAL - The value is given in seconds. Defines time interval between *ledmon sysfs* scan. The minimum is 5 seconds the maximum is not specified. The default value is 10 seconds.

LOG_LEVEL - Corresponds with **`--log-level`** flag from *ledmon*. Log level **`QUIET`** means no logging at all and **`ALL`** means to log everything. The default log level is **`WARNING`**. Acceptable values are **`quiet`**, **`error`**, **`warning`**, **`info`**, **`debug`**, **`all`**. Value also can be set by integer number - **`0`** means **`quiet`** and **`5`** means **`all`**.

LOG_PATH - Sets a path to local log file. If this option is specified, the global log file **`/var/log/ledmon.log`** is not used.

RAID_MEMBERS_ONLY - If flag is set to true, *ledmon* will limit monitoring only to drives that are RAID members. The default value is `false`.

REBUILD_BLINK_ON_ALL - Flag is related with RAID rebuild process. When value is set to false, only the drive that the RAID is rebuilding to will be marked with appropriate LED pattern. If value is set to true, all drives from RAID that is during rebuild will blink during this operation.

WHITELIST - *Ledmon* will limit changing LED state to controllers listed on whitelist. If any whitelist is set, only devices from list will be scanned by *ledmon*. The controllers should be separated by comma (,) character.

EXAMPLES

Excluding one controller from *ledmon* scans, changing log level and scans interval:

```
LOG_LEVEL=all
INTERVAL=5
#Exclude disks from SATA controller
BLACKLIST=/sys/devices/pci0000:00/0000:00:17.0
```

Blink only on RAID members, blink on all disks during rebuild and ignore init phase:

```
RAID_MEMBERS_ONLY=true
BLINK_ON_INIT=false
REBUILD_BLINK_ON_ALL=true
```

LICENSE

Copyright © 2009-2017 Intel® Corporation.

This program is distributed under the terms of the GNU General Public License as published by the Free Software Foundation. See the built-in help for details on the License and the lack of warranty.

§§