Graid Technology Inc.

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

Technical Product Specification

Revision 008b

August 2026

Document Number: 335893-007

Copyright ©2021-2026 Graid Technology Inc. All Rights Reserved.

SupremeRAID® is a registered trademark of Graid Technology Inc. All other trademarks and registered trademarks are the property of their respective owners.

Graid Technology reserves the right to make changes without further notice to any products or content herein to improve reliability, function, or design. Neither Graid Technology nor its suppliers makes any warranties as to the accuracy or completeness of any content or information provided herein, all of which are provided on an AS IS basis.

No license to Graid Technology's or any third party's intellectual property rights are conveyed hereunder.

Publication March 19, 2026

Contents

1 Introduction 6
1.1 Intended Use 6
1.2 Intended Audience 6
1.3 Reference OEM Platform Documentation 6
1.4 Terminology 6

2 Product Overview 10
2.1 Intel® VROC Linux* Product Family 10
2.2 Supported Operating Systems 11
2.2.1 Supported Linux* Distributions in the Intel® VROC 9.4 Release 11
2.2.2 Supported Linux* Distributions in the Intel® VROC 9.0/9.1/9.2 Release 11
2.2.3 Supported Linux* Distributions in the Intel® VROC 8.2 Release 11
2.2.4 Supported Linux* Distributions in the Intel® VROC 8.0 Release 12
2.3 New Features 12
2.3.1 New Features Introduced in the Intel® VROC 9.4 Release 12
2.3.2 New Features Introduced in the Intel® VROC 9.2 Release 13
2.3.3 New Features Introduced in the Intel® VROC 8.0 Release 13
2.3.4 New Features Introduced in the Intel® VROC 7.5 Release 15

3 Intel® Volume Management Device 18
3.1 Multiple Intel® VMD Controllers 18
3.2 Intel® VMD Method of LED Management 19
3.3 Intel® VMD Surprise Removal of NVMe Devices 19
3.4 Intel® VMD Error Management in Linux* 20

4 Intel® VROC Linux* Package Components 21
4.1 Intel® VROC Pre-OS Components 21
4.2 Intel® VROC Linux* ISO Images 21
4.3 Intel® VROC Linux* RPM Binary Files 21
4.4 Intel® VROC Linux* Source Code 22
4.5 Intel® VROC Linux* Package Documentation 22

5 Intel® VROC Key Features 23
5.1 Intel® VROC Features and Functionality 23
5.1.1 Intel® VROC (VMD NVMe RAID) Modes of Operation Upgrade Hardware Key for RAID Support 23
5.1.2 Intel® VROC RAID Member Drive Visibility 25
5.1.3 Spanning Intel® VROC RAID Data Volumes across Intel® VMD Domains 25
5.1.4 Intel® VROC (VMD NVMe RAID) on Pass-Thru Boot Drives 25
5.1.5 Intel® VROC (VMD NVMe RAID) Spanned RAID Volumes across Intel® VMD Controllers 26
5.1.6 Intel® VMD NVMe Compliance 26
5.1.7 Intel® VROC Support Maximum Number of Drives in a RAID Volume 26
5.1.8 Intel® VROC Support for Maximum RAID Volumes 26

6 Intel® VROC Common Features 27
6.1 Intel® Matrix Storage Manager 27

6.2 Intel® VROC Pre-OS Features and Functionality 27
6.3 Intel® VROC Linux* Monitoring and Management 28
6.3.1 Intel® VROC Linux* Command Line Tool 28
6.3.2 Intel® VROC Linux* Out-of-Box Driver Installation 28
6.3.3 Monitoring and Logging 28
6.3.4 LED Management 30
6.3.5 Intel® VROC RAID Volume Management 33
6.3.6 RAID Management 35
6.3.7 Intel® RAID Write Hole Closure 43
6.3.8 Intel® VROC Disk Management 46
6.4 MDRAID Sysfs Interface 52

7 Intel® VROC (PCH) Key Features 55
7.1 Intel® VROC (PCH) Miscellaneous Features and Functionality 55
7.1.1 SGPIO on SATA Controller (RAID Mode Only) 55
7.1.2 TRIM Command (RAID 0, 1 and 10) 55

8 Unsupported Features 56
8.1 Linux* Kernel Lockdown and UEFI Secure Boot 56
8.2 Linux* NVMe Multipathing 56
8.3 Preserve User Data While Migrating Non-RAID Disk to Intel® VROC RAID 57

Appendix A Related Documentation 58

A.1 Relevant Specifications 58

Appendix B Hardware Compatibility 59

Figures

Figure 2-1. Overview of Intel® VROC Out-Of-Band Management Solution 14
Figure 3-1. Intel® VMD Architecture 19
Figure 5-1. Intel® VROC RAID Spanning VMD Domains 25

Tables

Table 1-1. Terminology 6
Table 5-1. Intel® VROC Key Features 23
Table 5-2. Intel® VROC SKUs 24
Table 6-1. LED Management Behaviors 30
Table 6-2. Ledmon Options 31
Table 6-3. Migration Capabilities with IMSM 41
Table 6-4. Maximum Number of Drives per RAID Type 42
Table 6-5. Recommended Strip Size for Intel® NVMe SSDs for Optimal Wear Leveling 47
Table 7-1. Intel® VROC PCH Key Features 55

Revision History

Version	Description	Date
001	Initial release	May 2017
002	Intel® VROC 7.5 release	January 2020
003	Intel® VROC 8.0 release	November 2022
004	Initial public release	July 2023
005	Intel® VROC 8.2 release	August 2023
006	Note added in section 3.4	January 2025
007	Intel® VROC 9.2 release Update for NVMe multipath part	June 2025
008	VROC 9.4 release Adding Kylin*	April 2026
008a	Appended note for RAID Level Migration in Intel® VROC 9.4 release.	May 2026
008b	Correct VROC per volume has its maximum drives in table 6.4. Correct that Integrated Caching is deprecated since VROC 7.x from Premium key license Add note between Release 7.5 and 8.0	August 2026

§§

1 Introduction

This document is the Technical Product Specification for the Intel® Virtual RAID on CPU (Intel® VROC) Linux* family of products. The intent of this document is to present the functional requirements (or features) and technical details that make up these products. The features addressed include those for the Pre-Operating System (Pre-OS) components, the Linux* based drivers and corresponding Linux* tools.

1.1 Intended Use

This document is intended to provide high level information on the technical features of the Intel® VROC Linux* family of products that cover both NVMe drives attached to the Intel® Volume Management Device (Intel® VMD) controllers and SATA drives attached to the Platform Controller Hub (PCH).

1.2 Intended Audience

The intended audience of this document is OEMs and ODMs requiring detailed information of the new features and technical specifications of the Intel® VROC Linux* family of products.

1.3 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.

1.4 Terminology

Table 1-1. Terminology

Term	Definition
AER	Advanced Error Reporting
AHCI	Advanced Host Controller Interface
API	Application Programming Interface
ATA	Advanced Technology Attachment
ATAPI	Advanced Technology Attachment Packet Interface
BIOS	Basic Input / Output System

Term	Definition
BBE	Boot Behind Expander
Chipset	Term used to define a collection of integrated components required to make a PC function.
CLI	Command Line Interface
CSMI	Common Storage Management Interface
DMA	Direct Memory Access
Disk Coercion	Applying the coercion scaling factor to the disk to reduce the amount of available capacity on the disk when integrated within a system.
DOS	Disk Operating System
DIPM	Device Initiated Power Management
DSJ	Dirty Stripe Journaling
Drive Roaming	Moving of drives and arrays to different bays or between platforms to maintaining data availability.
Disk's Write Cache	A memory device within a drive, which is allocated for the temporary storage of data before that data is copied to its permanent storage location.
EBDA	Extended BIOS Data Area
EN	Entry Server
EP	Efficient Performance
GB	Gigabyte
GUI	Graphical User Interface
HDD	Hard Disk Drive
HIPM	Host Initiated Power Management
HII	Human Interface Infrastructure
Hot Plug	A term used to describe the removal or insertion of a SATA drive when the system is powered on.
Hot Spare Disk	A disk flagged to be automatically used to rebuild a failed or degraded RAID volume without user interaction.
ICH	Input / Output Controller Hub
IHV	Independent Hardware Vendor
IMSM	Intel® Matrix Storage Manager
I/O	Input / Output
JD	Journaling Drive
LPM	Link Power Management
Matrix RAID	Multiple RAID volumes residing within the same array.
MB	Megabytes
NAI	Notification Area Icon

Term	Definition
NCQ	Native Command Queuing
NTFS	NT File System
NPEM	Native PCIe Enclosure Management
NVMe	Non-Volatile Memory Express
ODD	Optical Disk Drive
ODM	Original Design Manufacturer
OEM	Original Equipment Manufacturer
OOB	Out of Band
OOB-MSM	Out of Band Management Service Module
OROM	Option ROM
OS	Operating System
PCH	Platform Control Hub
Pre-OS	Pre-Operating System Environment (Legacy OROM and/or UEFI)
Port	The point at which a SATA drive physically connects to the SATA controller.
PPL	Partial Parity Logging
PRD	Product Requirements Document
PV	Production Version
Volume Roaming	Moving of RAID volumes and all its array members between two enabled platforms to maintaining data availability or different supported operating systems.
RAID	Redundant Array of Independent Disks
RDC	Resource & Documentation Center
RHEL	Red Hat Enterprise Linux*
RSTe	Rapid Storage Technology enterprise
RWH	RAID Write Hole
SATA	Serial ATA
sSATA	Secondary Serial ATA
tSATA	Third Serial ATA
SES	SCSI Enclosure Service
SGPIO	Serial General Purpose I/O
SMART	Self-Monitoring, Analysis and Reporting Technology: an open standard for developing drives and software systems that automatically monitors a drive's health and reports potential problems.
SLES	SUSE Linux* Enterprise Server
SMIS	Storage Management Initiative Specification
SSD	Solid-State Drive – non-volatile memory

Term	Definition
UI	User Interface
UEFI	Unified Extensible Firmware Interface
Intel® VMD	Intel® Volume Management Device
VMD Domain	A grouping of drives behind a single Intel® VMD
Intel® VROC	Intel® Virtual RAID on CPU

§§

2 Product Overview

The Intel® VROC Linux* family of products provide enterprise RAID solutions for both NVMe SSD and SATA devices directly attached to the enterprise server environment. It's designed and targeted for Intel® Xeon® Processor Family based platforms that support the Intel® VMD technology. The product family includes the following products:

1. Intel® VROC (VMD NVMe RAID) Linux* – This product provides an enterprise RAID solution on platforms that support the Intel® VMD technology. This functionality is developed for and incorporated within Linux* MDRAID.
2. Intel® VROC (SATA RAID) Linux* – This product provides an enterprise RAID solution for SATA devices connected to the SATA controllers on Intel® Platform Control Hub (PCH) configured for RAID mode. This functionality is developed for and incorporated within Linux* MDRAID.

These products can be used independently. For example (unless otherwise restricted by the platform itself), Intel® VMD is not required to be enabled to use Intel® VROC (SATA RAID) Linux*. Likewise, Intel® VROC (VMD NVMe RAID) Linux* can be enabled and used without Intel® VROC (SATA RAID) Linux*.

2.1 Intel® VROC Linux* Product Family

Intel® VROC Linux* refers to the family of products that operate on platforms that support Intel® VMD technology. This product family is comprised of the:

- Intel® VROC Pre-OS components that support creating and booting from a RAID volume. This is provided by Intel via the Intel® VROC Windows* Kit available on Intel® RDC.
- Intel® VMD Linux* driver which is developed and maintained by Intel and upstreamed to the open-source Linux* kernel community.
- Linux* MDRAID driver and *mdadm* components to manage Linux* RAID volumes using the Intel® Matrix Storage Manager (IMSM) metadata format. The Intel® VROC Linux* enhancements to MDRAID and *mdadm* are developed in conjunction with and also upstreamed to the open-source community.

Intel® VROC Linux* also refers to the product that operates on the platforms with PCH SATA Controllers. This product is comprised of the:

- Intel® VROC Pre-OS components that support creating and booting from a RAID volume. This is provided by Intel via the Intel® VROC Windows* Kit available on Intel® RDC.
- Linux* MDRAID driver and *mdadm* components to manage Linux* RAID volumes using the IMSM metadata format. The Intel® VROC Linux* enhancements to MDRAID and *mdadm* are developed in conjunction with and also upstreamed to the open-source community.
- Linux* AHCI driver that supports Intel® PCH SATA controllers.

An important requirement for utilizing Intel® VROC Linux* is the need to specify the Intel® IMSM metadata format. This is done by using the **`--metadata=imsm`** option when

creating the MDRAID container device. Refer to the Intel® Virtual RAID on CPU (Intel® VROC) User Guide for Linux* for more details.

Another feature of the Intel® IMSM metadata is the ability to roam a container between Linux* and Microsoft Windows* host systems. This supports the manufacturing environment where a platform being built can have either Windows* or Linux* or both.

Intel® VROC Linux* supports the following RAID levels: RAID 0, 1, 5, 10 and Intel® Matrix RAID. Refer to the section Intel® VROC RAID Modes of Operation Upgrade Hardware Key for RAID Support for additional information on requirements for Intel® VROC (VMD NVMe RAID) Linux* support.

2.2 Supported Operating Systems

2.2.1 Supported Linux* Distributions in the Intel® VROC 9.4 Release

The Intel® VROC Linux* 9.4 product package is designed to work with, tested and validated on Intel® Customer Reference Boards (CRBs) with the following supported operating systems. Consult the Virtual RAID on CPU (VROC) Linux Release Note.

The overall supporting OS list, please onsult the Virtual RAID on CPU (VROC) Operating Systems Support List.

2.2.2 Supported Linux* Distributions in the Intel® VROC 9.0/9.1/9.2 Release

Consult the Virtual RAID on CPU (VROC) Operating Systems Support List.

2.2.3 Supported Linux* Distributions in the Intel® VROC 8.2 Release

The Intel® VROC Linux* 8.2 product package is designed to work with, tested and validated on Intel® Customer Reference Boards (CRBs) with the following supported operating systems.

- Red Hat Enterprise Linux* Server:
  - RHEL 8.7[1]
  - RHEL 8.8[1]
  - RHEL 9.1[1]
  - RHEL 9.2[1]
- SUSE Linux* Enterprise Server:
  - SLES15 SP5[1]
- Ubuntu Server:

- 20.04.3 LTS

2.2.4 Supported Linux* Distributions in the Intel® VROC 8.0 Release

The Intel® VROC Linux* 8.0 product package is designed to work with, tested and validated on Intel® Customer Reference Boards (CRBs) with the following supported operating systems.

- Red Hat Enterprise Linux* Server:
  - RHEL 8.2
  - RHEL 8.3
  - RHEL 8.4
  - RHEL 8.5
  - RHEL 8.6[1]
  - RHEL 9.0[1]
- SUSE Linux* Enterprise Server:
  - SLES15 SP2
  - SLES15 SP3
  - SLES15 SP4[1]

Note: [1]There are no additional out-of-box driver packages from Intel. The OS inbox has already been validated and full functional for Intel® VROC. Always reference the latest Intel® Virtual RAID on CPU (Intel® VROC) Supported Configurations for details.

2.3 New Features

2.3.1 New Features Introduced in the Intel® VROC 9.4 Release

The Intel® VROC Linux* 9.4 release package introduces some features to support the 6th Gen Intel® Xeon® Scalable Processor Platforms as well as improve the user experience. The following lists the key features introduced in Intel® VROC Linux* 9.4.

2.3.1.1 New Linux* OS supported Kylin*

Kylin* V11.

2.3.2 New Features Introduced in the Intel® VROC 9.2 Release

The Intel® VROC Linux* 9.2 release package introduces some features to support the 6th Gen Intel® Xeon® Scalable Processor Platforms as well as improve the user experience. The following lists the key features introduced in Intel® VROC Linux* 9.2.

2.3.2.1 New update for NVMe multipathing

The Linux* NVMe multipath feature in kernel interaction with Intel® VROC 9.2 mdadm and ledmon are updated in "Linux* NVMe multipathing" section.

2.3.3 New Features Introduced in the Intel® VROC 8.0 Release

The Intel® VROC Linux* 8.0 release package introduces several new features to support the 4th Gen Intel® Xeon® Scalable Processor Platforms as well as improve the user experience. The following lists the key features introduced in Intel® VROC Linux* 8.0.

2.3.3.1 Support of Intel® VMD 3.0

Intel® VMD is a hardware logic inside the Intel® Xeon® Scalable Processor. The 4th Gen Intel® Xeon® Scalable Processors introduce Intel® VMD 3.0, which supports the following key features:

- Up to 64 MSI-X vectors.
- Increase in the number of PCIe lanes that can be controlled by Intel® VMD from 64 to 80.
- Increase in the number of Intel® VMD devices from 5 to 6.
- Intel® VMD Hot Plug support of NVMe devices attached to the Platform Controller Hub (PCH).
- Intel® VROC Out-of-Band Management.

2.3.3.2 Support of 3rd SATA Controller

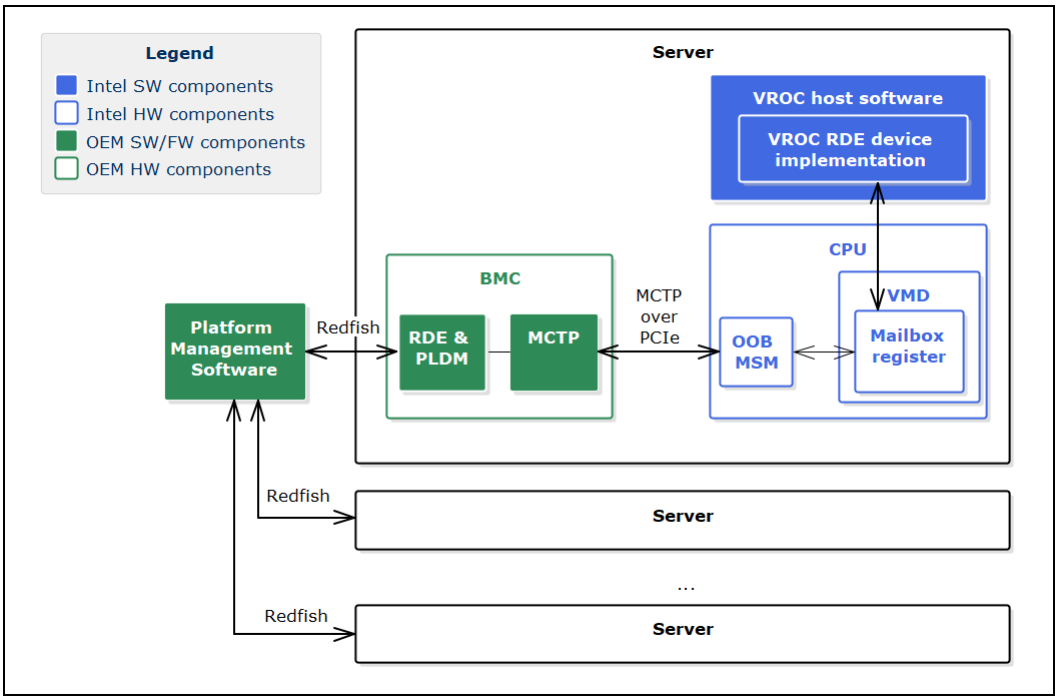
The Intel® VROC Linux* 8.0 release introduces the support for the 3rd SATA controller introduced on the Intel® C741 chipset on the 4th Gen Intel® Xeon® Scalable Processor Platforms. The Intel® C741 chipset has three integrated SATA host controllers that support up to twenty ports with data transfer rates of up to 6Gb/s on all ports. The twenty ports are split into eight ports in two controllers and four ports on the 3rd controller. Utilizing the Flexible I/O Adapter (FIA) on PCH, those twenty SATA ports are multiplexing with DMI, PCIe, GbE and USB port. Therefore, it highly depends on the platform providers to implement how many SATA ports are available on the platform. Intel® VROC Linux* 8.0 can support all the three SATA controllers on the platform.

2.3.3.3 Support of Out-of-Band Management

Intel® VROC Linux* 8.0 introduces the support for Out-of-Band (OOB) Management on platforms based on the Intel® Eagle Stream Customer Reference Board design. The Intel® VROC Out-of-Band Management interface supports NVMe drives and Intel®

VROC (VMD NVMe RAID) volumes only. SATA drives and Intel® VROC (SATA RAID) volumes can be managed through other Intel® VROC (in-band) interfaces. In the Intel® VROC OOB Management solution, the BMC is the entry point for the requests from a management network. It processes the requests (e.g., management command) and sends them to the MCTP endpoint exposed by the Intel® OOB-MSM, via MCTP over PCIe. The Intel® OOB-MSM works as an MCTP bridge, it receives requests from the BMC and sends it to the Intel® VROC OOB host software over the Intel® VMD mailbox register. The Intel® VROC OOB host software component periodically polls the Intel® VMD mailbox register for new requests. The Intel® VROC OOB host software performs the requested action and sends the response to the BMC in the opposite way. A high-level view of the solution is presented in the below figure.

Figure 2-1. Overview of Intel® VROC Out-Of-Band Management Solution



2.3.3.4 Support of PCIe Hot Plug on PCH

Intel® VROC Linux* 8.0 supports the hot plug of NVMe SSDs that are directly attached to the Intel® PCH PCIe lanes managed by Intel® VMD.

Note: This represents a feature change from version 7.5 to 8.0. Please refer to the corresponding sections for full details.

2.3.3.5 Intel® VROC (VMD NVMe RAID) UEFI Maximum Number of Drives

The Intel® VROC (VMD NVMe RAID) 8.0 UEFI driver supports a maximum number of 96 NVMe drives across all Intel® VMD controllers within the supported platform.

2.3.4 New Features Introduced in the Intel® VROC 7.5 Release

The Intel® VROC 7.5 Linux* release package introduces several new features to support the 3rd Gen Intel® Xeon® Scalable Processor Platforms as well as improve the user experience. The key new features introduced are:

- Intel® VMD 2.0 support which includes:
  - Increase in MSI-X vectors to 64.
  - Intel® VMD support for NVMe devices attached to the Platform Controller Hub (PCH).
  - Increase in the number of PCIe lanes that can be controlled by Intel® VMD from 48 to 64.
  - Increase in the number of Intel® VMD devices from 3 to 5.

2.3.4.1.1 Increasing MSI-X Vectors to 64

Intel® VROC 7.5 Linux* introduces support for customer configurations that can support 64 MSI-X vectors. On platforms that support Intel® VMD 1.0, the MSI-X support is limited to 32 MSI-X vectors. For these (Intel® VMD 1.0) platforms, a single Intel® VMD domain can support up to 24 NVMe SSDs. This means that those 24 NVMe SSDs will share a single Intel® VMD MSI-X vector. As the number of vectors, in newer NVMe devices, increase beyond 32, this can result in a platform performance impact. With the introduction of Intel® VMD 2.0, and Intel® VROC 7.5, this increase to 64 MSI-X vectors should help to alleviate this problem. This is because the average dual socket server will have between 48 and 64 cores, which limits the number of MSI-X vectors on a given NVMe SSD to 64, for optimal usage.

2.3.4.1.2 Intel® VMD PCH Support

Intel® VROC 7.5 Linux* introduces support for the Intel® VROC (VMD NVMe RAID) management of NVMe SSDs connected to the Platform Controller Hub (PCH). Utilizing the Flex-IO capabilities of the PCH, Intel® VMD 2.0 technology can now take ownership of two of the slots allowing Intel® VROC (VMD NVMe RAID) to control and manage the NVMe SSDs attached to those slots. This is accomplished by utilizing the Function Level Assignment of the PCH PCI functions. Utilizing the BIOS setup menus, the user will be able to enable Intel® VMD on the designated slots, or PCH functions (depending on the BIOS implementation) and the PCH functions will be reassigned to an Intel® VMD (PCH) controller. To be able to accomplish this, there must be NVMe SSDs present on the slot(s) and the slots used must support Slot Implemented Capabilities. Otherwise, Intel® VMD (PCH) cannot be enabled.

When this feature is fully enabled, it will reassign sSATA ports 2-5 to be PCIe lanes managed by Intel® VMD (PCH). This is translated into PCIe root ports 8-11 and Flex I/O ports 14-17. The general configuration is 2- X2 PCIe lanes.

This feature is intended for supporting a simple RAID 1 boot using two NVMe SSDs attached directly to the PCH. However, if the platform supports expanded configurations, using retimer or switch Add-In-Cards (AIC), full Intel® VROC (VMD NVMe RAID) Linux* support can be obtained. Intel® VROC (VMD NVMe RAID) Linux* was only validated with two NVMe SSDs directly attached to the PCH. Any configurations beyond two directly attached NVMe drives is not recommended.

2.3.4.1.3 Intel® VMD (PCH) Pre-OS Support

Intel® VMD (PCH) Pre-OS support is included in the Intel® VROC (VMD NVMe RAID) Pre-OS images, which is part of the Intel® VROC release package. There are no other Pre-OS images required.

2.3.4.1.4 Intel® VMD (PCH) Pass-Thru Boot Support

Intel® VROC 7.5 Linux* provides Intel® VROC (VMD NVMe RAID) Pre-OS support for NVMe SSDs attached to the PCH when Intel® VMD (PCH) is enabled. This will allow any supported Linux* OS to be installed onto and boot from an NVMe device managed by Intel® VROC (VMD NVMe RAID) Linux*. No Intel® VROC upgrade key is required to utilize this feature.

2.3.4.1.5 Intel® VMD (PCH) RAID Boot Support

Intel® VROC 7.5 Linux* provides Intel® VROC (VMD NVMe RAID) Pre-OS support for NVMe SSDs attached to the PCH when Intel® VMD (PCH) is enabled. When an Intel® VROC upgrade key is present and Intel® VMD (PCH) is enabled, the user will be able to use the Intel® VROC (VMD NVMe RAID) Pre-OS HII to setup and manage a RAID volume using the NVMe SSDs attached to the PCH. This will allow any supported Linux* OS to be installed onto and boot from an Intel® VROC (VMD NVMe RAID) volume attached to Intel® VMD (PCH).

2.3.4.1.6 Intel® VMD (PCH) RAID Data Volume Spanning

Once Intel® VMD (PCH) is enabled, Intel® VROC (VMD NVMe RAID) Linux* will treat it like any other Intel® VMD controller. Spanning data RAID volumes are supported as they are currently outlined in this document. This applies to the Intel® VROC Pre-OS tools and Linux* tools. The option to span Intel® VMD controllers is supported, but not recommended.

Note: This is supported but not recommended due to a performance penalty using the PCH.

Note: Boot volume spanning is not supported. All devices used to create a boot volume must reside on the same Intel® VMD domain.

2.3.4.1.7 Intel® VMD (PCH) Designations

The Intel® VROC (VMD NVMe RAID) Linux* management tools (Pre-OS tools and Linux* tools) are used to manage devices attached to the Intel® VMD (PCH) controller, the corresponding information displayed will indicate “PCH” with any device or controller associated with Intel® VMD (PCH).

2.3.4.1.8 Intel® VMD (PCH) Hot Plug Support

Intel® VROC does not support Hot Plug when NVMe SSDs are attached to the Intel® VMD (PCH) controller.

2.3.4.2 Native PCIe Enclosure Management (NPEM)

Intel® VROC 7.5 Linux* introduces support for the Native PCIe Encloser Management (NPEM) standard for LED management in a PCIe 4.0 based environment. This capability is discoverable in each switch-downstream-port. If it is discovered to be present, the Intel® VROC LED utility will use NPEM control, capability, and status registers to visually indicate the various drive and volume states.

2.3.4.3 Limited Out-of-Band Support

Intel® VROC 7.5 Linux* introduces limited Out-of-Band support.

2.3.4.4 Limited Self Encrypted Drives

Intel® VROC 7.5 introduces Self Encrypting Drive (SED) key management support. The implemented key management is only in the UEFI environment but allows secure booting with SEDs into all Intel® VROC OS environments.

2.3.4.5 Intel® VMD Direct Assignment

Intel® VROC 7.5 Linux* introduces support for Intel® VMD Direct Assignment. The Intel® VMD Direct Assignment feature supports a virtualization hypervisor directly assigning an Intel® VMD, and all NVMe SSDs owned by that Intel® VMD controller, to a storage controller Virtual Machine (VM). This allows direct access to the SSDs by the VM, totally by-passing the hypervisor. The storage controller could be Windows* or Linux* OS. The Intel® VMD Direct Assignment feature can improve performance dramatically by removing hypervisor overhead.

§§

3 Intel® Volume Management Device

With the introduction of the Intel® Xeon® Processor Scalable family, one of the key features included is the Intel® Volume Management Device (Intel® VMD). The Intel® VMD is an integrated PCIe endpoint within the CPU root complex. The class code for the Intel® VMD device is a RAID controller. Intel® VMD driver support is provided with the Intel® VROC package and includes the following:

- Multiple Intel® VMD Controllers
- LED Management (VMD Method of LED Management)
- Surprise Hot Plug
- Error Handling

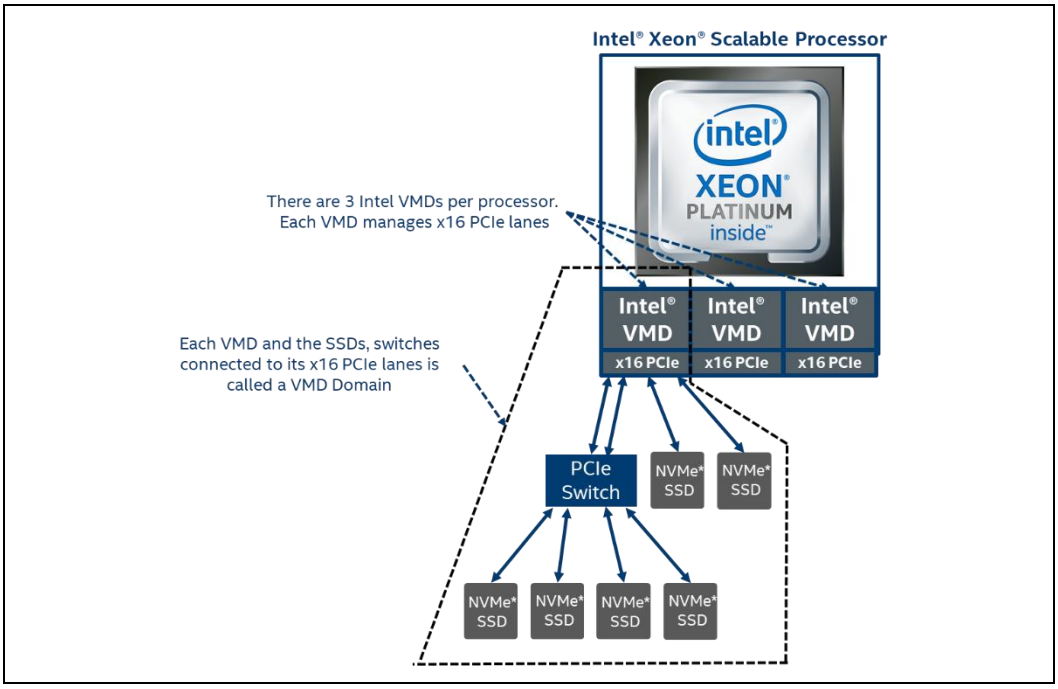
Each Intel® Xeon® Scalable Processor family with Intel® VMD 3.0 provides up to 80 PCIe lanes which is subdivided into five Intel® VMD domains, plus one Intel® VMD domain for the PCH. Each Intel® VMD domain manages x16 PCIe lanes. Intel® VMD can be enabled or disabled on x4 PCIe lane granularity and supports either NVMe SSDs or PCIe switch devices. Intel® VMD is also designed to support the following types of NVMe SSDs:

- PCIe Add in Cards (AIC)
- M.2 form factor drives
- U.2 form factor drives

3.1 Multiple Intel® VMD Controllers

Each Intel® Xeon® Scalable Processor family contains at least three Intel® VMD controllers. The number of Intel® VMD controllers depends on the number of PCIe lanes the Intel® Xeon® Scalable Processor family can support.

Figure 3-1. Intel® VMD Architecture



3.2 Intel® VMD Method of LED Management

Backplane LED management is a critical part of enterprise RAID management. Intel® VMD provides a mechanism to support LED management for NVMe backplanes plugged into an Intel® VMD controller. The mechanism utilizes repurposing the PCIe hot plug register bits (*Power Indicator Control* and *Attention Indicator Control* of the *Slot Control* register) to support IBPI blink patterns defined in the SFF-8489 standard.

Refer to the Intel® Virtual RAID on CPU (Intel® VROC) User Guide for Linux* for details on how to utilize and/or customize the LED management behavior in this product.

Note: LED management outside of a supported backplane environment is not supported.

3.3 Intel® VMD Surprise Removal of NVMe Devices

When Intel® VMD is enabled, surprise insertion and removal of NVMe drives is isolated from the PCIe bus and managed by Intel® VROC. The Intel® VMD takes appropriate action on device enumeration and transport driver device instances on Intel® NVMe PCIe SSDs based on SFF-8639 interface (U.2 form factor only with this release). This includes surprise insertion/removal of 2.5” form factor devices in the OS runtime environment.

3.4 Intel® VMD Error Management in Linux*

Intel® VMD error management in Linux* is implemented in the Linux* PCIe Advanced Error Reporting (AER) driver, which provides the infrastructure to support PCIe AER capability. The errors are classified into two types: *correctable errors* and *uncorrectable errors*. This classification is based on the impacts of those errors, which may result in degraded performance or function failure.

Correctable errors pose no impacts on the functionality of the interface. The PCIe protocol can recover without any software intervention or any loss of data. These errors are detected and corrected by hardware.

Unlike correctable errors, uncorrectable errors impact functionality of the interface. Uncorrectable errors can cause a particular transaction or a particular PCIe link to be unreliable. Depending on those error conditions, uncorrectable errors are further classified into non-fatal errors and fatal errors. Non-fatal errors cause the particular transaction to be unreliable, but the PCIe link itself is fully functional. Fatal errors, on the other hand, cause the link to be unreliable. The PCIe device will automatically send an error message to the PCIe root port above it when the device captures an error. The Root Port, upon receiving an error reporting message, internally processes and logs the error message in its PCIe capability structure. Error information being logged includes storing the error reporting agent's requestor ID into the Error Source Identification registers and setting the error bits of the Root Error Status register accordingly. If AER error reporting is enabled in the Root Error Command register, the Root Port generates an interrupt if an error is detected.

Note that the Intel VMD Linux driver only enables or disable AER on the VMD root-ports. The VMD driver does not include AER recovery or interrupt handling mechanism to be able to stop the Linux AER driver in a case where it is continuously repeating errors in an unending loop.

§§

4 Intel® VROC Linux* Package Components

The support model for Intel® VROC Linux* is through the supported OS Vendors (OSV) publicly available distribution packages. Refer to the Intel® Virtual RAID on CPU (Intel® VROC) Supported Configurations for the full list of supported OSV packages. The intent is to utilize these OSV packages for everything required to install and manage Intel® VROC (in-box).

For new platforms that contain new technology (such as Intel® VMD 3.0), these OSV packages may not have the required support. To support these new platforms, a specific OSV based out-of-box package can be made available for OEM/ODMs who may require these new drivers.

This section outlines the individual components that are included in the Intel® VROC Linux* out-of-box packages that can be provided to customers upon request. The specific details on each component are in the following sections of this document.

The Intel® VROC Linux* out-of-box package will include:

- An Intel® VROC Linux* ISO image (when necessary).
- The Intel® VROC RPM binary files.
- All the corresponding source codes.

4.1 Intel® VROC Pre-OS Components

The Intel® VROC Pre-OS is required to be included in the platform BIOS in order to take advantage of the Intel® VROC functionality. Refer to the Intel® Virtual RAID on CPU (Intel® VROC) Technical Product Specification for Windows* for the details on the Intel® VROC Pre-OS components.

4.2 Intel® VROC Linux* ISO Images

The Intel® VROC ISO image is a standard Linux* ISO image that can be used during the OS installation or installed on the system after the OS has been installed. The process will install the latest device driver, the Intel® LED management application and corresponding updates to *mdadm* (e.g., the Intel® VROC components for Intel® VMD (PCH) support). These images are only made available when there is a new platform feature (like Intel® VMD 3.0) and the existing OSV (in-box) packages cannot be used.

4.3 Intel® VROC Linux* RPM Binary Files

The Intel® VROC Linux* out-of-box package will most closely represent this release model. After the OS is installed, the corresponding RPM binary images can be applied. This does require the OS to be installed on the platform.

4.4 Intel® VROC Linux* Source Code

Intel® VROC works closely with the open-source community and is an active contributor to Linux* MDRAID. In staying with GPL licensing, all Intel® VROC source code associated with these packages will be included in the bundle provided to customers.

4.5 Intel® VROC Linux* Package Documentation

With the Intel® VROC out-of-box package provided, there will also be the corresponding documentation, such as the package release notes, a Linux* User Guide, custom package instructions as well as the latest version of the Intel® VROC Linux* Technical Product Specification (TPS).

§§

5 Intel® VROC Key Features

The Intel® VROC Linux* product package is comprised of several components that provide a complete platform solution.

The following is a brief list of the features that are supported on Intel® VROC (VMD NVMe RAID) for NVMe devices attached to the Intel® Volume Management Device (Intel® VMD).

Table 5-1. Intel® VROC Key Features

Key Features	• Surprise Hot-Plug • LED Management (VMD Method) • Error Management (VMD First) • NVMe RAID (0/1/5/10) Boot • UEFI HII RAID Management • RAID Volume Roaming • Self-Encrypting Drive (SED) Key Management • Intel® VMD Direct-Assign for Virtualization • Out-Of-Band Management support • Partial Parity Log for RAID 5 Write Hole Closure • Write-Intent Bitmap
Intel® VROC Linux* Components	• *mdadm* • *ledmon* • Intel® VMD Linux* kernel driver • Linux* kernel md driver

5.1 Intel® VROC Features and Functionality

This section is intended to outline those features that are specific to Intel® VROC Linux*.

5.1.1 Intel® VROC (VMD NVMe RAID) Modes of Operation Upgrade Hardware Key for RAID Support

The Intel® VROC (VMD NVMe RAID) package will support four operational SKUs (or modes) of the platform. They are the Intel® VROC Pass-Thru SKU, Intel® VROC Standard SKU and the Intel® VROC Premium SKU. To enable either the Intel® VROC Standard SKU or the Intel® VROC Premium SKU, an Intel® VROC RAID upgrade key must be purchased and installed in the platform.

These Intel® VROC RAID upgrade keys are available for purchase for use on platforms based on Intel® Xeon® Scalable Processor family platforms. The Intel® VROC RAID upgrade key is a small PCB board that has a security EEPROM, that is read by the Intel® VROC UEFI driver to enable different Intel® VROC software stack features when Intel® VMD is enabled. Refer to the user's platform documentation for the location of the Intel® VROC RAID upgrade key placement.

Note: It is assumed that all Intel® VROC supported platforms contain the required physical hardware header to support the Intel® VROC RAID upgrade key.

Table 5-2. Intel® VROC SKUs

SKU	Description	HW Key Required	Key Features
Intel® VROC Pass-Thru	The expected default platform configuration will be the Intel® VROC Pass-Thru SKU. This SKU is a platform that supports Intel® VROC but does not contain an Intel® VROC RAID upgrade key. In this configuration, the Intel® VROC product will enumerate all NVMe drives (including non-Intel drives) and will expose them to the platform as pass through drives. Intel® VROC RAID functionality is disabled.	Not needed	• All NVMe SSDs supported in Pass-Thru mode only (no RAID) • LED Management • Hot Plug Support
Intel® VROC Standard	The Intel® VROC Standard SKU configuration is enabled with the Intel® VROC RAID Standard upgrade key is installed in the platform. This SKU has the same behavior as the Intel® VROC Pass-Thru SKU with the addition of support for RAID 0, 1 and 10. When the Intel® VROC RAID Standard upgrade key is installed, and Intel® VMD is enabled, the Intel® VROC UEFI HII will enable RAID 0, 1 and 10 management on all supported NVMe drives. This functionality applies only to those PCIe slots that have Intel® VMD enabled.	Standard Key	• Pass-Thru SKU features • RAID 0, 1, 10 (Bootable and Data)
Intel® VROC Premium	The Intel® VROC Premium SKU configuration is enabled with the Intel® VROC RAID Premium upgrade key is installed in the platform. This SKU has the same behavior as the Intel® VROC Standard SKU with the addition of support for RAID 5 and RAID 5 Write Hole Closure. When the Intel® VROC RAID Premium upgrade key is installed, and Intel® VMD is enabled, the Intel® VROC UEFI HII will enable RAID 0, 1, 5 and 10 management on all supported NVMe drives. This functionality applies only to those PCIe slots that have Intel® VMD enabled.	Premium Key	• Standard SKU features • RAID 5 (Bootable and Data) • RAID 5 Write Hole Closure

5.1.2 Intel® VROC RAID Member Drive Visibility

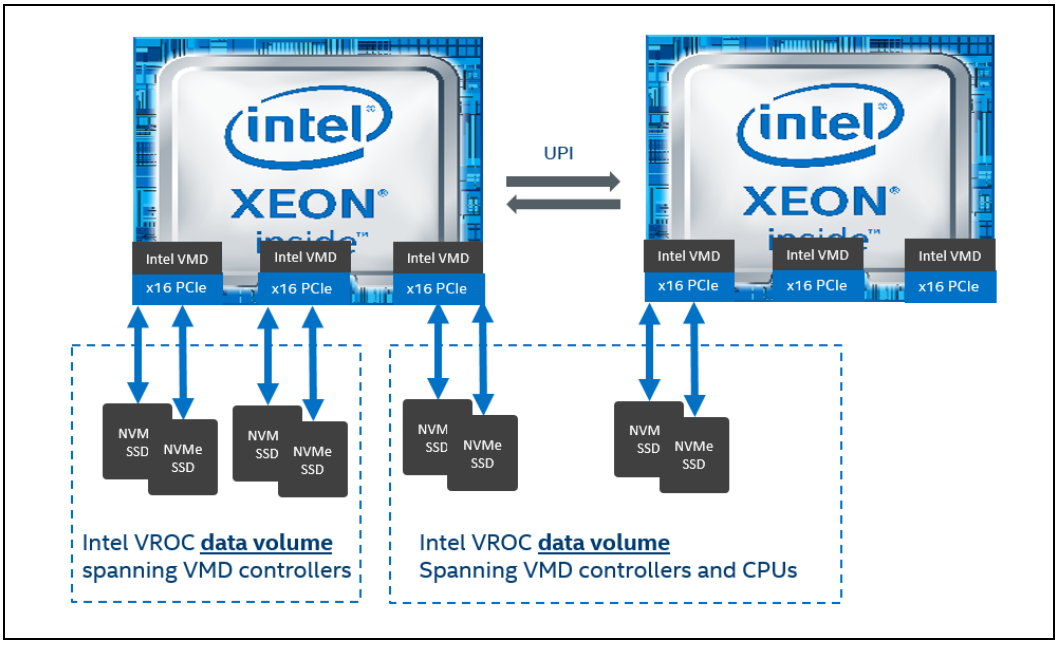
In the Intel® VROC product, when NVMe SSDs are connected to an Intel® VMD controller domain and are established as an Intel® VROC RAID volume (bootable or data), those NVMe disks will be reported to the Linux* environment as disks. Those RAID member disks will be reported as being connected to that Intel® VMD controller domain.

5.1.3 Spanning Intel® VROC RAID Data Volumes across Intel® VMD Domains

Intel® VROC allows data RAID volume on NVMe SSDs that span/reside across Intel® VMD domains. Data RAID volume members can reside on different Intel® VMD domains and be combined into a RAID volume used for storing data.

Although supported, spanning across CPUs is generally not recommended, doing so may incur in performance penalties.

Figure 5-1. Intel® VROC RAID Spanning VMD Domains



5.1.4 Intel® VROC (VMD NVMe RAID) on Pass-Thru Boot Drives

The Intel® VROC product will allow installing to and booting from an OS on a pass-thru NVMe SSD drive (not part of a RAID volume) that is behind an enabled Intel® VMD controller.

5.1.5 Intel® VROC (VMD NVMe RAID) Spanned RAID Volumes across Intel® VMD Controllers

Intel® VROC (VMD NVMe RAID) supports the ability to create an Intel® VROC RAID volume that will span Intel® VMD domains. These volumes can be created at any point before or after the user's system is successfully running Linux* but creating a RAID boot volume should occur in the Pre-OS/BIOS. The following instructions will cover creation of spanned using the BIOS interface.

5.1.6 Intel® VMD NVMe Compliance

The Intel® VMD Linux* driver adheres to the NVMe specification version 1.2.

5.1.7 Intel® VROC Support Maximum Number of Drives in a RAID Volume

The Intel® VROC 8.0 Linux* will support the ability to create RAID volumes from up to 96 NVMe drives connected to the Intel® VMD controllers (directly connected or in switches).

5.1.8 Intel® VROC Support for Maximum RAID Volumes

The Intel® VROC 8.0 Linux* will support the ability to create up to 2 RAID volumes in a RAID array.

§§

6 Intel® VROC Common Features

This section will outline and describe those features that are common between the Intel® VROC support for Intel® VMD controllers and platform PCH controller support.

The following is a summary of the key features of this product that are supported on Intel® VROC (SATA RAID) devices. Not all legacy features below will be supported on Intel® VROC RAID for NVMe devices.

6.1 Intel® Matrix Storage Manager

The Intel® VROC component of *mdadm* is called Intel® Matrix Storage Manager (IMSM). This is the metadata that defines Intel® VROC. Intel® Matrix Storage Manager was the name of the original Intel® RAID solution. It is now called Intel® VROC.

This is a description of each of the RAID configuration elements and how they relate to each other.

Container

A container is a collection of two or more NVMe or SATA drives in a RAID configuration using the IMSM formatted metadata. It is the highest element in the hierarchy of a storage system. The RAID volume is created from the disks the user used to create the container.

Volume

A volume is the storage area on two or more drives whose type dictates the configuration of the data stored. If the user created a volume for data protection, then the user's storage system may include a RAID 1 volume spanning two NVMe or SATA disks, which mirrors the data on each disk.

Disks

A disk (i.e., hard disk, SSD or Intel® NVMe SSD) physically stores data and allows read/write data access. If a disk is used to create a volume, it becomes a container disk because it has been grouped with other disks to form a container.

The Intel® VROC Linux* product will be able to support the creation and management of RAID 0, 1, 5, 10 volumes.

6.2 Intel® VROC Pre-OS Features and Functionality

With the introduction of Intel® VMD and Intel® VROC, one of the key features is the ability to install an OS to and boot from a RAID volume. A key component in this feature is the Intel® VROC RAID Pre-OS. The Intel® VROC RAID Pre-OS is a set of binary images that are compiled into the platform BIOS and provide a method by which the BIOS environment will be able to do the following:

- Initialize and enumerate NVMe devices managed by the Intel® VMD controller (when enabled).

- Scan for the Intel® VROC RAID upgrade key and properly configure the Intel® VROC software to properly manage the SKU the system is configured for.
- Provides a RAID management user interface to the BIOS to manage RAID volumes and NVMe devices connected to the Intel® VMD controller.

Refer to the Intel® Virtual RAID on CPU (Intel® VROC) Technical Product Specification for Windows* for the details on the Intel® VROC Pre-OS components.

6.3 Intel® VROC Linux* Monitoring and Management

6.3.1 Intel® VROC Linux* Command Line Tool

Intel® VROC uses *mdadm* tool to manage RAID in Linux* OS. Reference the Intel® Virtual RAID on CPU (Intel® VROC) User Guide for Linux* for how to use *mdadm* to manage Intel® VROC RAID in Linux*.

Note: Linux* DMRAID is not supported.

6.3.2 Intel® VROC Linux* Out-of-Box Driver Installation

There are two ways of installing the Intel® VROC out-of-box drivers. The first is leverage the Driver Update Disk (DUD) feature to load and install the out-of-box drivers during the OS installation phase. The second is to use the standard Linux* process for installing packages from an ISO image to an existing installed supported Linux* OS. Refer to the Intel® Virtual RAID on CPU (Intel® VROC) User Guide for Linux* for more details.

6.3.3 Monitoring and Logging

6.3.3.1 mdmon

The *mdadm* monitor, *mdmon*, is automatically started when MDRAID volumes are activated by *mdadm* through creation or assemble. However, the daemon can be started manually as follows:

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

The **`--all`** parameter can be used in place of the container name to start monitoring for all active containers.

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

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

6.3.3.2 Configuration File for Monitoring

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

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

6.3.3.3 Examples of Monitored Events in syslog

```
Personalities : [raid5]

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

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

To monitor all RAID containers, an *mdadm* daemon can be started using the following command:

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

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

```
myhost mdadm[9863]: NewArray event detected on md device /dev/md127
myhost mdadm[9863]: NewArray event detected on md device /dev/md0
```

When a spare drive has been added:

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

When an OLCE command is finished:

```
myhost mdadm[9863]: RebuildFinished event detected on md device /dev/md127
```

When a drive fails:

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

When a rebuild finishes:

```
myhost mdadm[9863]: RebuildFinished event detected on md device /dev/md127
myhost mdadm[9863]: SpareActive event detected on md device /dev/md127
```

When all MD devices are stopped:

```
myhost mdadm[9863]: DeviceDisappeared event detected on md device /dev/md127
myhost mdadm[9863]: DeviceDisappeared event detected on md device /dev/md0
```

6.3.3.4 Logging

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

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

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

6.3.4 LED Management

Intel® VROC supports the ability to perform basic LED management of the status LEDs within SFF-8485 (or equivalent) compliant backplane and with the blink patterns as defined in SFF-8489. The LEDs management will occur in the following behavior:

Table 6-1. LED Management Behaviors

Activate LED	Used to activate the status LED of a specific drive at 1Hz.
Fail LED	Used to identify a present drive in a failed state (or missing RAID volume member) by turning the status LED on.
Rebuild	Used to identify the specific drive that is being brought into a rebuilding RAID volume by blinking the status LED at 4Hz.
Volume Initializing	Used to indicate that the RAID volume is initializing by blinking all the status LED of the RAID volume members by blinking the status LEDs at 4Hz.

Note: 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.

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

6.3.4.1 Intel® VROC (SATA RAID) SGPIO LED Management

Intel® VROC (SATA RAID) for the PCH supports LED management through SGPIO signaling. Intel® VROC LED Management in Linux* is achieved by using the Linux* *ledmon* and *ledctl* utilities.

6.3.4.2 Intel® VROC LED Management in Linux*

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

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

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 services monitor all RAID volumes. There is no method to specify individual volumes to monitor. These utilities have only been verified with Intel® storage controllers.

The Intel® VROC Linux* product will provide LED management support for status LEDs on enclosures that provides either a two-LED solution or a three-LED solution.

Ledmon can be run with the following options listed below:

Table 6-2. Ledmon Options

Option	Usage
-c or **--config-path=**	Sets the configuration file path. This overrides any other configuration files (although the utility currently does not use a config file). The **/etc/ledcfg.conf** is shared by *ledmon* and *ledctl* utilities.
-l or **--log-path**	Sets the path to a log file. This overrides **/var/log/ledmon.log**.
-t or **--interval=**	Sets the time interval in seconds between scans of the sysfs. A minimum of 5 seconds is set.
--quiet, **--error**, **--warning**, **--info**, **--debug**, **--all**	Specifies verbosity level of the log. **quiet** means no logging at all, and **all** means to log everything. The levels are given in order. If user specifies more than one verbose option, the last option comes into effect.
-h or **--help**	Prints help text and exits.
-v or **--version**	Prints version and license information, then exits.

The following example shows the steps to install and use the *ledmon*/*ledctl* utility:

Step 1: Execute the following commands to install *ledmon*:

```
# yum install ledmon-<latest version>.rpm
```

Running *ledctl* and *ledmon* concurrently, *ledmon* will eventually override the *ledctl* settings.

Step 2: Start the IPMI with the following commands:

```
# systemctl start ipmi
```

```
# ledmon
```

The following command will blink the drive LED on the device node **/dev/nvme0n1**:

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

The following command will turn off the locate LED:

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

6.3.4.3 Setting Up ledmon to Auto-Start

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

Step 1: Confirm *ledmon* is present/started on your Linux* installation. This command will start the daemon if not already started:

```
# ledmon
```

If it is running, the following is the expected return text:

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

Step 2: To ensure that the *ledmon* daemon starts on each reboot, open the file **/etc/rc.local** using your favorite editor (e.g., VIM or VI).

```
# vi /etc/rc.local
** Insert/add ‘ledmon’ to the final line of the file as shown below.
#!/bin/bash
# THIS FILE IS ADDED FOR COMPATIBILITY PURPOSES
#
# It is highly advisable to create own system services or udev rules
# to run scripts during boot instead of using this file.
#
# In contrast to previous versions due to parallel execution during boot
# this script will be executed during boot.
#
# 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.

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

6.3.5 Intel® VROC RAID Volume Management

6.3.5.1 Volume Creation and Deletion

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

The following shows an example of how to create a RAID 5 array with four Intel® NVMe SSDs.

Note: It is recommended that the user always starts with new drives. If using drives with existing data on them, it is recommended that the superblocks are cleared before starting:

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

Step 1: A container of Intel® MSM metadata must be created:

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

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

Step 2: Next, a RAID 5 volume is created.

```
#mdadm -C /dev/md/Volume0 /dev/md0 –n 4 –l 5
mdadm: array /dev/md/Volume0 started.
```

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

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

-n Number of active devices in array.

-x Number of spare (extra) devices in initial array.

-c Specifies the chunk (strip) size in Kilobytes. The default is 128KiB. This value must be a multiple of 4KiB. Optimal chunk size should be considered depending on expected workload profiles.

-l Specifies the RAID level. The options are 0, 1, 5, and 10.

-z Specifies the size (in Kilobytes) of space dedicated on each disk to the RAID volume. This must be a multiple of the chunk size. For example:

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

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

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

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

Step 3: After the RAID volume has been created, a file system can be created in order to allow the mounting of the RAID volume. The volume can also be partitioned further as the user requires/desires.

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

Note: RAID 5 volumes are immediately initialized. As such the initialization process should be monitored (using **cat /proc/mdstat**) and it should be allowed to complete before continuing:

Step 4: After the file system has been created, it can be mounted to the location of choice:

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

6.3.5.2 Volume Assembly

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

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

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

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

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

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

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

6.3.5.3 Stopping the Volumes

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

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

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

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

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

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

6.3.5.4 Online Capacity Expansion

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

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

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

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

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

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

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

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

6.3.6 RAID Management

6.3.6.1 RAID Operations

The process of Linux* RAID management using *mdadm* is the same for SATA drives attached to the PCH controller as it is for NVMe drives attached to the Intel® VMD.

Note: Do not create RAID volumes with mixed media (SATA and NVMe SSDs).

6.3.6.1.1 Using the Intel® VROC HII to Create a RAID Volume

The following are instructions on how to create a RAID volume using the Intel® VROC Pre-OS HII interface. This procedure should only be used for a newly built system or reinstall of the operating system. The user should use *mdadm* from the Linux* operating system for creating RAID volumes after the operating system is up and running.

Note: Consult the user's platform documentation for instructions on how to enter the Intel® VROC HII interface.

The following assumptions have been made:

- It is known how to enter the appropriate platform BIOS level setup menus.
- The Intel® VMD functionality has been enabled.
- The appropriate Intel® VROC RAID upgrade key has been installed.
- The appropriate number of NVMe SSDs have been plugged into the enabled Intel® VMD controller.

To create a RAID volume, do the following:

Step 1: Enter the BIOS configuration setup menu to access the Intel® VROC HII user interface.

Step 2: Navigate to and select **Intel® Virtual RAID on CPU**.

Step 3: Navigate to and select **Create RAID Volume**.

Step 4: Type in a volume name and press the <Enter> key or press the <Enter> key to accept the default name. Keep volume names to standard alpha-numeric characters. The use of excessive special characters may cause problems.

Step 5: Select the RAID level by pressing <Enter> to access the menu and using the <△> or <▽> keys to scroll through the available values, then press the <Enter> key to select the desired RAID type.

Step 6: Optional for RAIDs that have volumes connected to more than one Intel® VMD controller: To enable the RAID to be spanned over multiple controllers, highlight the < >, and press <Enter>. This will open a small menu where the default is set as blank, indicating that this feature is not enabled. To enable, use the <▽> key, and press <Enter>.

Step 7: Using the <△> or <▽> keys, highlight the drives one by one by selecting the < > bracket on the line next to that drive's port number. Press <Enter> to open the selection menu which will be set initially to blank. Use the <□> key to highlight the X and press <Enter> to enable as part of the RAID.

Step 8: Repeat Step 7 for each drive required in this RAID.

Step 9: Unless the user has selected RAID 1, select the strip size by using the <△> or <▽> keys to scroll through the available values, then press the <Enter> key.

Step 10: Select the volume capacity and press the <Enter> key. The default value indicates the maximum volume capacity using the selected disks. This value is

calculated in bytes. For demonstration 700GB is 716800 (700 * 1024 = 716800).

Step 11: Use the <∇> key to select **Create Volume** and press <Enter>. If the user's RAID created successfully, it will be listed under the heading of **RAID Volumes**. Other disks or portions of volumes that were not included within the RAID will be listed as part of the selections under **Non-RAID Physical Disks**. These may be used to create additional RAID volumes.

Step 12: To exit the user interface, press <Esc>. Press <Esc> again. The user will be presented with the message: *Changes have not saved. Save Changes and exit? Press 'Y' to save and exit, 'N' to discard and exit. 'ESC' to cancel*.

Step 13: Press <Y> to save and exit. The user's RAID configuration has now been saved.

Step 14: To save and reboot to begin the OS installation, press <Esc> to return to the Main Menu. Highlight **Reset**, and press <Enter> to reboot the system back to the boot menu.

6.3.6.1.2 Creating RAID Configuration File

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

6.3.6.1.3 Adding Hot Spares

Adding a spare disk allows immediate reconstruction of the RAID volume when a device failure is detected. MDRAID will mark the failed device as *bad* and start reconstruction with the first available spare disk. The spare disk can also be used to grow the RAID volume. The spare disks sit idle during normal operations. When using *mdadm* with IMSM metadata, the spare disk added to a container is dedicated to that specific container.

6.3.6.1.4 Failing an Active Drive

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

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

6.3.6.1.5 Removing Failed Drive(s)

Below is the output of **`/proc/mdstat`** with an active RAID 5 volume with IMSM metadata where **`md0`** is the IMSM container and **`md127`** is the RAID 5 volume. The RAID 5 volume contains the drives **`/dev/nvme0n1`**, **`/dev/nvme1n1`**, **`/dev/nvme2n1`**, **`/dev/nvme3n1`**.

```
Personalities : [raid0] [raid1] [raid5] [raid10]
md127 : active raid5 nvme0n1[3] nvme1n1[2] nvme2n1[1] nvme3n1[0]
      39999488 blocks super external:/md0/0 level 5, 128k chunk, algorithm 0 [4/40]
[UUUU]

md0 : inactive - nvme0n1[3] (S) nvme1n1[2] (S) nvme2n1[1] (S) nvme3n1[0] (S)
```

```
      11285 blocks super external:imsm

unused devices: <none>
```

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

```
Personalities : [raid0] [raid1] [raid5]
md127 : active raid5 nvme1n1[2](S) nvme0n1[1](S) nvme3n1[0](S)
      39999488 blocks super external:/md0/0 level 5, 128k chunk, algorithm 0 [5/4]
[_UUUU]

md0 : inactive - - nvme0n1[3]  nvme1n1[2]  nvme3n1[0]
      1045 blocks super external:imsm

unused devices: <none>
```

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

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

Or removed from the container by running the following command:

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

6.3.6.1.6 RAID Volume Recovery

Intel® VROC 8.0 Linux* will also allow the first RAID volume in a container to recover from a failure condition. The volume will rebuild on the spare disk added before removal of broken disks without any problem. The file system must be accessible during this operation.

Intel® VROC 8.0 Linux* will support the ability to track the volume rebuild progress and resume this operation after accidental break (e.g., after unexpected system reboot).

6.3.6.1.7 RAID Auto Recovery

Auto-rebuild allows a RAID volume to be automatically rebuilt when a drive fails. There are 2 different scenarios when this can happen:

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

Intel® VROC 8.0 Linux* will require that a spare disk used for auto-rebuilding (recovery) must have consistent metadata with the array to be rebuilt. The minimum consistency requirement is that a spare drive has a valid IMSM metadata array.

Intel® VROC 8.0 Linux* will trigger an auto-rebuild of all volumes in a degraded container.

Intel® VROC 8.0 Linux* will enable container auto-rebuild spare sharing by default without the need for configuration.

Intel® VROC 8.0 Linux* will not automatically trigger a rebuild of a volume if another volume in the container does not require a rebuild.

Intel® VROC 8.0 Linux* will support RAID volume auto-rebuild to move an eligible spare disk to the degraded volume even if another volume in the same container is not eligible for rebuild. *Mdmon* rebuilds containers only if a failed disk is removed from the container.

Intel® VROC 8.0 Linux* will support volume auto-rebuild to spare drive defined in a container being rebuilt. This operation will start just after MD reports a failure of any drive establishing a RAID volume.

Intel® VROC 8.0 Linux* will support volume auto-rebuild to hot-plugged drive. The IMSM spare disks present in the system that are not part of any active MDRAID array can be used for rebuilding if a drive contains metadata consistent with the degraded array.

For the second scenario above, the following example is presented below:

Step 1: Create a container **md1** with three disks:

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

Step 2: Create a RAID 1 volume **Volume1** in the container **md1**. The drive **/dev/nvme2n1** remains as a spare:

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

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

 md1 : - inactive nvme2n1[2] (S) nvme0n1[1] (S) nvme1n1[0] (S)
      3315 blocks super external:imsm

unused devices: <none>
```

Step 3: Save configuration file:

```
# mdadm –Es > /etc/mdadm.conf
```

Step 4: Add the policy with the same domain and the same action for all drives to the configuration file, which allows the spare to move from one container to another for rebuild:

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

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

```
ARRAY metadata=imsm UUID=67563d6a:3d253ad0:6e649d99:01794f88 spares=1
ARRAY /dev/md/Volume2 container=67563d6a:3d253ad0:6e649d99:01794f88 member=0
UUID=76e507f1:fadb9a42:da46d784:2e2166e8
ARRAY metadata=imsm UUID=267445e7:458c89eb:bd5176ce:c37281b7
ARRAY /dev/md/Volume1 container=267445e7:458c89eb:bd5176ce:c37281b7 member=0
UUID=25025077:fba9cfab:e4ad212d:3e5fce11
POLICY domain=DOMAIN path=* metadata=imsms action=spare-same-slot
```

Step 5: Make sure *mdadm* is in monitor mode:

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

Step 6: Fail one of the drives in the volume **`Volume1`**, the volume without a spare:

```
# mdadm --fail /dev/md/Volume1 /dev/nvme1n1
```

The spare drive **`/dev/nvme2n1`** should be automatically used:

```
 [raid1]
: active raid1 nvme2n1[2] nvme0n[1]
md1/0 [2/1] [U_]
 [====è…..] recovery = X% (updated so-far/total) finish=0.0mins peed=144298K/sec
: inactive nvme2n1[2] (S) nvme1n1[1] (S) nvme0n1[0] (S)
      5363 blocks super external:imsm
```

When the rebuild has completed:

```
: active raid1 nvme2n1[2] nvme0n[1]
md1/0 [2/1] [UU]
 finish=0.0minspeed=144298K/sec
: inactive nvme2n1[2] (S) nvme1n1[1] (S) nvme0n1[0] (S)
      5363 blocks super external:imsm
```

Note: After system reboot, the spare drive will be moved to an individual container.

6.3.6.1.8 RAID Level Migration

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

Caution: Even though the data remains intact, always perform a data back-up before performing migration operations.

Intel® VROC Linux* will perform partition/filesystem size verification before beginning the following migration types:

- From non-RAID drive to RAID 1 volume.
- From non-RAID drive to RAID ready drive (i.e., single drive RAID 0).

MBR and GPT partitioning systems shall be supported.

Intel® VROC Linux* will support strip size modification performed along with level migration. The following strip sizes are supported: 4kB, 8kB, 16kB, 32kB, 64kB, 128KB.

Intel® VROC Linux* will not automatically start a rebuild on a RAID volume within a container that has a RAID volume being migrated.

Intel® VROC Linux* will disable automatic spare rebuild migration by default.

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

Table 6-3. 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 (with appropriate number of drives) 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.

Note: RAID level migration is not supported for RAID volumes configured in a multi-volume IMSM container (Matrix RAID configuration).

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

```
# cat /proc/mdstat
Personalities : [raid0] [raid1] [raid5] [raid10]
md127 : active raid1 nvme1n1[1] nvme0n1[0]
      102400 blocks super external:/md0/0 [2/2] [UU]

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

unused devices: <none>
```

Step 1: Migrate from RAID 1 to RAID 0:

```
# mdadm –G /dev/md127 –l 0
# cat /proc/mdstat
Personalities : [raid0] [raid1] [raid5] [raid10]
md127 : active raid0 nvme1n1[1] nvme0n1[0]
      102400 blocks super external:/md0/0 64k chunks

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

unused devices: <none>
```

Step 2: Use Online Capacity Expansion to go from one-disk RAID 0 to two-disk RAID 0:

```
# mdadm -G /dev/md0 -n 2
# cat /proc/mdstat
Personalities : [raid0] [raid1] [raid5] [raid10]
md127 : active raid0 nvme0n1[0] nvme1n1[1]
      204800 blocks super external:/md0/0 64k chunks

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

unused devices: <none>
```

Step 3: Add a spare disk to the container:

```
# mdadm -a /dev/md0 /dev/sdc
# cat /proc/mdstat
Personalities : [raid0] [raid1] [raid5] [raid10]
md127 : active raid0 mvme0n1[0] nvme1n1[1]
      204800 blocks super external:/md0/0 64k chunks

md0 : inactive - mvme0n1[0] (S) nvme1n1[1] (S)
      3315 blocks super external:imsm

unused devices: <none>
```

Step 4: Migrate from RAID 0 to RAID 5:

```
# mdadm -G /dev/md127 -l 5 --layout=left-asymmetric
# cat /proc/mdstat
Personalities : [raid0] [raid1] [raid5] [raid10]
md127 : active raid5 nvme2n1[3] nvme1n1[2] nvme0n1[1]
      204800 blocks super external:/md0/0 level 5, 64k chunk, algorithm 0 [3/3]
[UUU]

md0 : inactive - nvme2n1[3] (S) nvme1n1[2] (S) nvme0n1[1] (S)
      3315 blocks super external:imsm

unused devices: <none>
```

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

6.3.6.1.9 RAID Volume Initialization

Newly created volumes can be used immediately (no reboot required), protecting newly written data, and creating parity data concurrently.

6.3.6.1.10 Maximum Number of Drives in a RAID Volume

Intel® VROC 8.0 Linux* will provide support for the maximum number of drives in the following configurations. This is aligned with Intel® VROC 8.0 Pre-OS capability as well as platform capability.

Table 6-4. Maximum Number of Drives per RAID Type

RAID Level	Maximum Number of Drives for NVMe RAID	Maximum Number of Drives for SATA RAID
RAID 0	24	8
RAID 1	2	2
RAID 5	24	8
RAID 10	4	4

6.3.7 Intel® RAID Write Hole Closure

Intel® VROC supports the ability to close the RAID Write Hole scenario in RAID 5 configurations. This applies to Intel® VROC enabled platforms.

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

Intel® VROC Linux* implements distributed Partial Parity Logging to close the RAID Write Hole scenario. The Partial Parity Logging (PPL) can be enabled when creating Intel® VROC RAID 5 in UEFI or Linux* environment. This feature can also be enabled or disabled through the *mdadm* utility after Intel® VROC RAID 5 is created. Reference the Intel® Virtual RAID on CPU (Intel® VROC) User Guide for Linux* for the detailed command lines.

6.3.7.1 RAID Write Hole Closure (RWH) Linux*

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

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

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

6.3.7.2 RAID Write Hole Closure (RWH) Pre-OS

The Intel® VROC UEFI drivers will support the recovery from the RAID 5 volume invalid state caused by a RWH condition for all enumerated RAID 5 volumes during system boot.

6.3.7.3 RAID Write Hole Closure (RWH) Backwards Compatibility

For RAID 5 volumes on drives created in previous versions of the product, Intel® VROC 8.0 will be able to leverage the older RWH closure mechanisms.

6.3.7.4 RAID Write Hole Closure (RWH) Recovery

The Intel® VROC drivers will be able to recover from the RAID 5 volume invalid state caused by RWH condition occurrence for all RAID 5 volumes in the system which were exposed to I/O interruption (e.g., dirty shutdown).

The Intel® VROC drivers will be able to recover from the RAID 5 volume invalid state caused by RWH condition occurrence when the RAID 5 volume is discovered by the driver after hot plug of all the member drives except the failed drive.

The Intel® VROC drivers will be able to recover from the RAID 5 volume invalid state caused by RWH condition occurrence when the RAID 5 volume is discovered by the driver during driver load.

The Intel® VROC drivers will be able to recover from the RAID 5 volume invalid state caused by RWH condition occurrence when the RAID 5 volume is discovered by the driver after enabling all the member drives except the failed drive in the device management utility.

6.3.7.5 Disable On-Device Cache for NVMe

The Intel® VROC RWH closure protection feature is intended to be used with the NVMe onboard volatile cache disabled. Enter the NVMe SSD drive properties to disable on-device cache before proceeding to enable the Intel® VROC RWH closure protection feature. If a RAID 5 volume has the RWH closure enabled, with the Intel® VROC Linux* package installed, if an attempt is made to enable on-device cache on a RAID 5 member drive a warning message is added to syslog that PPL is meant to be used with on-device volatile cache disabled.

6.3.7.6 RAID 5 RWH Closure Feature PPL Distribution Mode

The Intel® VROC 8.0 RWH closure feature implementation on Intel® VROC/SATA RAID 5 volumes will be able to close the RAID 5 RWH condition without the use of additional drives. This is referred to as the PPL distributed RWH closure mode.

6.3.7.7 Runtime Switching Between RWH Closure Protection Feature Modes

The Intel® VROC Linux* product will provide the user with the ability to enable or disable the RWH closure protection feature during the normal OS operation mode.

Intel® VROC Linux* will provide support to enable/disable the RWH protection feature in the Linux* command line tool on existing RAID 5 volumes in the system. The options will support the ability to disable the feature (Off) or define the Partial Parity Log (PPL) configuration (PPL or multiple PPLs).

6.3.7.8 Turning On and Off the RWH Closure Protection Feature

The Intel® VROC Linux* product will provide the user with the ability to turn off the RWH closure mechanism. Intel® VROC Linux* will provide support to enable or disable the RWH protection feature in the Linux* command line tool when a RAID 5 volume is being created. The options will support the ability to disable the feature (Off) or define the Partial Parity Log (PPL) configuration (with Intel® VROC 8.0 installed, multiple PPL usage is automatic). There is no longer **`--rwh-policy`** parameter. Instead, there is a **`--consistency-policy`** parameter (or **`-k`** as short version).

Create example:

```
mdadm --create /dev/md/volume -l5 --size=1G --consistency-policy=ppl -n3 /dev/sd[a-c]
```

To turn on/off PPL during runtime (array is assembled and started):

```
mdadm --grow /dev/md/volume --consistency-policy=X
```

where **X** is **ppl** (ppl on) or **resync** (ppl off).

6.3.7.9 Interrupted PPL Write - RWH Recovery

If the PPL write request has been interrupted and PPL was not fully written, the RWH recovery will not be performed for this particular RAID 5 I/O request.

6.3.7.10 Ability to Switch Between RWH Closure Modes for SATA

The Intel® VROC Linux* product will allow the user the ability to switch between the following RWH closure modes during normal OS operation mode: PPL distributed mode and Off for RAID 5 volumes on SATA drives.

6.3.7.11 Intel® VROC UEFI for VMD RWH Recovery

Intel® VROC 8.0 UEFI driver for VMD will be able to recover from the RAID 5 volume invalid state caused by RWH condition occurrence for all enumerated RAID 5 volumes during system boot.

6.3.7.12 Intel® VROC RWH Closure Restrictions

Intel® VROC 8.0 will block the expansion of a RAID 5 volume that is being protected by the RWH closure feature.

Intel® VROC 8.0 will block the changing of the strip size of a RAID 5 volume that is being protected by the RWH closure feature.

Intel® VROC 8.0 will block adding a drive to an existing RAID 5 volume that is being protected by the RWH closure feature.

When the system discovers a RAID 5 volume with *Journaling Drive* RWH Closure protection enabled, the Intel® VROC 8.0 will set the RWH Closure protection feature to *Off* when a RAID 5 volume is discovered with a journaling drive.

6.3.7.13 Intel® VROC Linux* RWH Example

To create a RAID 5 volume with enabled RWH policy (it is recommended to clear out metadata):

```
# mdadm -C /dev/md/imsm0 –e imsm -n4 /dev/nvme[0-3]n1
# mdadm -C /dev/md/vol0 -l5 -n4 /dev/nvme[0-3]n1 --consistency-policy=ppl
```

To check the current RWH policy:

```
# mdadm -D /dev/md/vol0
```

To enable RWH policy for a running array:

```
# mdadm --rwh-policy=ppl /dev/md/vol0
```

To disable RWH policy for a running array:

```
# mdadm -G /dev/md/volume --consistency-policy=resync
```

6.3.8 Intel® VROC Disk Management

6.3.8.1 Intel® NVMe Wear Leveling Recommendations

NVMe Wear Leveling refers to techniques used to prolong the service life of NVMe drives. This section will outline the recommended configurations (number of drives vs strip size) to maximize Wear Leveling on Intel® NVMe SSDs when configured as part of a RAID 5 volume. When creating an Intel® VROC RAID 5 volume, several configuration parameters can be selected, and the number of drives used along with the strip size chosen can have an impact on the wear leveling. The following table outlines the different options for number of drives vs strip size to achieve the optimal wear leveling on Intel® NVMe SSDs.

6.3.8.2 Device Presentation

Intel® VROC 8.0 Linux* will support the ability to list all the devices attached to the Intel® VMD controller (when enabled) as well as the devices attached to the SATA and sSATA controller.

6.3.8.3 4K Native Sector Sizes Drives

Note: This feature is only supported in UEFI mode.

Intel® VROC Linux* will support the following drives:

- 512 and 512b sector size drives.
- Native 4K sector size drives for SATA/sSATA controllers and NVMe SSDs for Intel® VROC.

Note: The only limitation is the inability to use the mixed 4K and 512(b) in one RAID volume (creation, migration, expansion, etc.).

6.3.8.4 Reporting and Notifications

Intel® VROC Linux* will report the status of RAID volumes and containers to the user through the command line.

Intel® VROC Linux* trigger an event and/or send an e-mail in case of a situation required to be reported.

Intel® VROC Linux* will notify the user (and prevent the creating of a RAID volume) when an unsupported number of components are identified in the command line.

Intel® VROC Linux* will notify the user (and prevent the creating of a RAID volume) when the same device is already identified as a RAID member.

Intel® VROC Linux* will notify the user (and prevent the creating of a RAID volume) when the same device is already identified as a member of a different RAID volume.

Intel® VROC Linux* will notify the user (and prevent the activation of a RAID volume) if the RAID metadata has a Bad Block Management log entry.

Intel® VROC Linux* will notify the user to appropriately modify size of partitions existing on the source volume/drive if the existing partitions cannot be fully moved during volume migration due to lack of space on the destination volume (e.g., metadata should be added).

Intel® VROC Linux* will notify the user to provide another drive to successfully complete an expanding operation if a given drive cannot be used for expansion of the particular array (e.g., drive smaller than what is required to properly expand all volumes defined on this array).

Table 6-5. Recommended Strip Size for Intel® NVMe SSDs for Optimal Wear Leveling

Drives \ Strip Size	4	8	16	32	64	128
3	Optimal	Optimal	Optimal	Optimal	Optimal	Optimal
4	Optimal	Optimal	Optimal	Optimal	Suboptimal	Suboptimal
5	Optimal	Optimal	Optimal	Optimal	Optimal	Optimal
6	Optimal	Optimal	Optimal	Optimal	Optimal	Suboptimal
7	Optimal	Optimal	Optimal	Optimal	Optimal	Optimal
8	Optimal	Optimal	Optimal	Suboptimal	Suboptimal	Suboptimal
9	Optimal	Optimal	Optimal	Optimal	Optimal	Optimal
10	Optimal	Optimal	Optimal	Optimal	Optimal	Suboptimal
11	Optimal	Optimal	Optimal	Optimal	Optimal	Optimal
12	Optimal	Optimal	Optimal	Optimal	Suboptimal	Suboptimal
13	Optimal	Optimal	Optimal	Optimal	Optimal	Optimal
14	Optimal	Optimal	Optimal	Optimal	Optimal	Suboptimal
15	Optimal	Optimal	Optimal	Optimal	Optimal	Optimal
16	Optimal	Optimal	Suboptimal	Suboptimal	Suboptimal	Suboptimal
17	Optimal	Optimal	Optimal	Optimal	Optimal	Optimal
18	Optimal	Optimal	Optimal	Optimal	Optimal	Suboptimal
19	Optimal	Optimal	Optimal	Optimal	Optimal	Optimal
20	Optimal	Optimal	Optimal	Optimal	Suboptimal	Suboptimal
21	Optimal	Optimal	Optimal	Optimal	Optimal	Optimal
22	Optimal	Optimal	Optimal	Optimal	Optimal	Suboptimal
23	Optimal	Optimal	Optimal	Optimal	Optimal	Optimal
24	Optimal	Optimal	Optimal	Suboptimal	Suboptimal	Suboptimal

Note: It is left to the customer to determine the most effective combination of parameters (number of drives vs. strip size) to achieve their desired performance goals, usage models and drive endurance.

6.3.8.5 Hot Plug (Surprise and Managed)

Intel® VROC Linux* will support the ability to hot plug (remove and replace) disk drives on properly configured controllers whether I/O is being processed or not. To be able to take advantage of this feature, the system and BIOS must have this feature enabled.

Hot Plug, also referred to as hot swap, is a feature that allows SSDs (SATA or NVMe) to be removed or inserted while the system is powered on and running under a Linux* operating system. As an example, Hot Plug may be used to replace a failed drive that is in an externally accessible drive enclosure.

This will be expanded to reference PCH and Intel® VROC functionality. This will also include references to the required BIOS settings to ensure that the BIOS is properly configured. Consult the user's platform design documentation for additional information.

6.3.8.5.1 How to Enable the BIOS for Hot Plug

This series of instructions will guide users through the Intel® BIOS configuration for enabling a RAID with a spare drive addition. This allows for configurations such as a data or file storage service using multiple drives to have backup drives ready for another feature, auto rebuild to assist in recovery and use much more swiftly.

6.3.8.5.2 Enabling Surprise Hot Plug for Intel® VROC

The following steps are an example of how to enable surprise hot plug in the Intel® CRB platform BIOS:

Step 1: From the boot options menu, select the option that will allow the user to enter the BIOS setup.

Step 2: With the **EDKII Menu** highlighted, press <Enter>.

Step 3: Navigate to **Socket Configuration**, then press <Enter>.

Step 4: Navigate to the selection that reads **PCIe Hot Plug**. Press <Enter> to open the options menu. The user may toggle this setting between disabled and enabled.

Step 5: Select **Enable**, and press <Enter> to set selection.

Note: This will enable the feature for all NVMe drives that are associated with the system. This is a distinction from Hot Plug as it works with PCH drives. Consult the user's platform documentation because each BIOS is different, and the steps taken to enable this feature may be different.

6.3.8.5.3 Enabling Surprise Hot Plug for PCH

The following steps are an example of how to enable surprise hot plug in the Intel® CRB platform BIOS:

Step 1: From the boot options menu, select the option that will allow the user to enter the BIOS setup.

Step 2: With the **EDKII Menu** highlighted, press <Enter>.

Step 3: Navigate to **Platform Configuration**, and press <Enter>.

Step 4: Navigate to **PCH Configuration**, and press <Enter>.

Step 5: Navigate to **PCH SATA** or **PCH SSATA Configuration** (as appropriate) and press <Enter>.

Note: Each port may be individually enabled for Hot Plug. The user may turn this feature on for all or none as appropriate as the administrator. Consult the user's platform documentation because each BIOS is different, and the steps taken to enable this feature may be different.

6.3.8.6 Hot Spare Disk

Intel® VROC Linux* will support the ability to set a drive as a hot spare that would automatically be used to rebuild a failed or degraded RAID volume without any user interaction.

6.3.8.6.1 Rebuild on Hot Insert

Intel® VROC Linux* will provide support for initiating a RAID volume rebuild process with the hot insert of a drive into the same physical location as the failed or missing drive. The rebuild process will begin only when:

- The inserted disk meets the size requirements for the array containing the degraded RAID volume.
- The remaining healthy member disks are of the same type of drive.
- The newly inserted disk is a non-RAID disk.
- The newly inserted disk is a member of a failed volume.
- The newly inserted disk is a member of a degraded volume.

To enable this feature, the properly configure policies must be added to **`/etc/mdadm.conf`**. Reference the Intel® Virtual RAID on CPU (Intel® VROC) User Guide for Linux* for detailed configuration.

6.3.8.6.2 Define a Hot Spare

Marking a disk as a *spare* allows the user to designate an available disk as the default destination for automatic volume rebuilds in the event of a failed, missing or "at risk" (SMART event) array disk.

The action is only available for non-system disks in a normal state. The spare disk must be connected to the same controller as the disk that it is supporting. The maximum number of spare disks is determined by the maximum number of disks supported by the controller.

Setting a disk as a spare disk can be accomplished in two ways:

1. The Intel® VROC provides Pre-OS environment (UEFI HII) user interface support.
2. The *mdadm* command line tool.

6.3.8.6.3 Return Hot Spare Back to Normal

Intel® VROC management tools will provide a mechanism for setting a spare back to a pass-thru disk under Disk Properties.

6.3.8.6.4 Returning a Spare to a Normal Disk in the BIOS

Depending on the RAID volume created by Intel® VROC or by PCH, the process is fairly similar. The destination will vary slightly based on where the RAID was created. The assumption is that this information is known.

Step 1: Enter the setup and configuration mode of the BIOS.

Step 2: Navigate to **EDKII Toolkit** and press the <Enter> key.

Step 3: Navigate to **Intel ® Virtual RAID on CPU** and press the <Enter> key. This may be where the user deviates and selects the option **Intel® VROC sSATA Controller** or **Intel® VROC SATA Controller**.

Step 4: The non-RAID disk will be listed below the RAID volume on the screen. Highlight the target drive, then press <Enter>.

Step 5: This page is the Physical Disk Information. Highlight the option that reads **Reset to non-RAID**, then press <Enter>.

Step 6: The user will be presented with a message reading *Remove RAID structure on disk?* Highlight the option reading **Yes**, then press the <Enter> key.

The disk is no longer a spare for the RAID volume. This can be verified by highlighting the non-RAID drive and viewing the Physical Disk Information page. **Mark as Spare** is available as an option again.

6.3.8.7 Mark a Disk as Spare in HII

The Intel® VROC product will provide an option to mark a disk as spare in the UEFI HII UIs.

The assumption has been made that the appropriate additional drive(s) that are to be made into spare(s) have been physically installed on the system. The installed drives are assumed to meet the size requirements to be identified as spare drives for the RAID array they will be associated to.

Step 1: Enter the setup and configuration mode of the BIOS.

Step 2: Navigate to **EDKII Toolkit** and press the <Enter> key.

Step 3: Navigate to **Intel® Virtual RAID on CPU** and press the <Enter> key. Alternate destinations may be **Intel® VROC sSATA Controller** or **Intel® VROC SATA Controller**.

Step 4: The non-RAID disk will be listed below the RAID volume in a category below. Highlight the target drive desired and press the <Enter> key.

Step 5: This screen presents the Physical Disk Information page. Depending on the nature of the RAID array, there may be two options available. *Mark as Spare* and *Mark as Journal*. For the intended purpose of marking as a spare drive, highlight **Mark as Spare**, and press the <Enter> key.

Step 6: A window will be displayed asking *Are you sure the user wants to mark the disk as Spare? Marking disk as Spare will remove all data on the disk.* To proceed, highlight **Yes**, and press <Enter>.

6.3.8.8 Read Patrol

The Intel® VROC product will provide support for Read Patrol. Read Patrol checks for physical disk errors that could lead to drive failures. These checks usually include an attempt of corrective action.

You can enable Read Patrol scan by writing the string **`check`** to **`/sys/block/md###/md/sync_action`**, simply run:

```
$ echo [check/repair] > /sys/block/md###/md/sync_action
```

where **`md###`** is the MD device name (like **`/dev/md126`**).

The number of potential errors is stored in **`/sys/block/md###/md/mismatch_cnt`**. The **`repair`** setting does both checking and repairing.

The above action performs Read Patrol only once. A cron job could be used to run Read Patrol periodically. Such script is present in RHEL6.x releases.

6.3.8.9 Check Pointing

Intel® VROC will provide the ability to perform Check Pointing to be able to track forward progress on read patrol, array rebuilds and volume migration if interrupts occur. After resuming, the operation will restart from the last valid stage reached.

Intel® VROC will be able to track volume migration progress and resume this operation after accidental break (e.g., after unexpected system reboot).

Intel® VROC will be able to support rebuild and migration check pointing for volumes established on drives.

6.3.8.10 Bad Block Management

Intel® VROC product will provide support for Bad Block Management.

In the course of rebuilding a degraded RAID volume, where one of the member disks has failed or been removed and is being replaced by a spare drive, the redundant contents of the other drive(s) are read and then used to reconstruct data to be written to the spare drive. In case a read failure occurs sometime during this rebuild process,

the data to be written to the spare will not be available and therefore lost. In this scenario, rather than marking the entire RAID volume as failed, we can mark only those sectors on the spare that are known to have indeterminate data, in a log of such bad sectors. This bad block management log can be used to reflect error status whenever any attempts are made to access those sectors of the spare.

This feature is supported in the Pre-OS as well as in the Linux* environment.

6.4 MDRAID Sysfs Interface

The MDRAID subsystem has sysfs components that provide information or can be used to modify behavior and performance. All MDRAID devices present in the system are shown in **/sys/block/**.

For 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 follows:

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

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

MD devices in **/sys/block** are symbolic links pointing to the **/sys/devices/virtual/block**. All MD devices are in the **md** subdirectory in 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    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 the 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 these 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
4096
```

Metadata

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

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

```
# cat /proc/mdstat
Personalities : [raid1]
md127 : active raid1 nvme0n1[1] nvme1n1[0]
      1048576 blocks super external:/md0/0 128K chunk,   [2/2] [UU]
md0 : inactive - nvme0n1[1] (S) nvme1n1[0] (S)
      2210 blocks super external:imsm
unused devices: <none>
```

For additional information on the sysfs interface, reference https://www.kernel.org/doc/html/v4.12/admin-guide/md.html.

§§

7 Intel® VROC (PCH) Key Features

The following is a summary of the key features of this product that are supported on Intel® VROC PCH/SATA devices. Not all legacy features below will be supported on Intel® VROC RAID for NVMe devices.

Table 7-1. Intel® VROC PCH Key Features

Name	Key Features	
RAID	• Bad Block Management	• SGPIO • Partial Parity Logging (PPL)[1]

Note: [1] Items that are supported on both Intel® VROC PCH/SATA and Intel® VROC.

7.1 Intel® VROC (PCH) Miscellaneous Features and Functionality

7.1.1 SGPIO on SATA Controller (RAID Mode Only)

The Intel® VROC (PCH) product will support enclosure management, compliant to the SFF-8485 standard (Specification for Serial GPIO Bus) as well as SFF-8489 (Specification for Serial GPIO IBPI (International Blinking Pattern Interpretation)), to identify drive location or unit failures on the SATA or sSATA controllers.

7.1.2 TRIM Command (RAID 0, 1 and 10)

Note: This feature is not an end-user visible feature. There is no Intel® VROC (PCH) application or user interface control to configure the feature.

Support for the TRIM command allows the OS to pass information to the Solid-State Drive (SSD) that identifies sectors that can be deleted. The SSD will then go through and clear out that information in the background thereby minimizing the chances of an "overwriting" process happening at crucial times. The SSD is also free to do some additional optimizations with those sectors (e.g., an SSD can pre-erase any sector that has been TRIM'ed). The TRIM command improves the long-term write performance and the lifespan of SSDs.

§§

8 Unsupported Features

This section will outline some (by no means all) of the unsupported features of Intel® VROC Linux* products.

8.1 Linux* Kernel Lockdown and UEFI Secure Boot

The Linux* Kernel Lockdown feature is intended to strengthen the boundary between the User ID (UID) 0 and the kernel. This is an optional kernel security feature firstly introduced in Linux* v5.4. If UEFI secure boot is enabled, the kernel lockdown feature is automatically enabled.

The upstream patch set of kernel lockdown is available in the following link: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=aefcf2f4b58155d27340ba5f9ddbe9513da8286d.

When kernel lockdown is enabled, it may prevent user-space applications from accessing the platform hardware directly. It may also prevent unsigned out-of-tree drivers being loaded.

Intel® VROC Linux* products currently do not plan on supporting the Linux* Kernel Lockdown feature. Currently, the Intel® VROC Linux* products operate under the assumption that the Linux* Kernel Lockdown feature and UEFI secure boot are disabled. To verify the default configuration of the Linux* Kernel Lockdown feature, consult the Linux* distribution documentation and/or consult the Linux* OSVs.

8.2 Linux* NVMe Multipathing

Linux* native NVMe multipathing was first introduced in kernel v4.15, as an efficient and native implementation in the kernel NVMe driver. Linux* NVMe multipathing provides path failover and path load sharing for multipath-capable NVMe drives.

In some of the Linux* distributions (e.g., RHEL 9.x, SLES 15), the native NVMe multipathing is enabled by default.

Intel VROC Linux product revisions older than Intel VROC Linux version 9.1 do not support the Linux* native NVMe multipathing feature. Enabling the Linux* native NVMe multipathing feature will cause Intel® VROC functions (e.g., LED management, OOB management) to not function properly under Linux*, and Intel recommends disabling Linux native NVMe multipathing by adding the **nvme_core.multipath=N** option to the kernel command line when using these Intel VROC Linux product revisions.

Starting from Intel VROC Linux product revision 9.1, Intel® VROC Linux products still do not support Linux multipath features, but the core Intel VROC Linux product functions will now work properly when using NVMe drives with multipath enabled. The Intel® VROC Linux product treats multipath drives as single-path drives.

Multipath drive support in the Intel® VROC Linux product is introduced in the following releases of mdadm and ledmon which are listed in the table below. The user can still disable multipathing by adding the **nvme_core.multipath=N** option to the

kernel command line if desired, particularly due to the limitation described in the following paragraph.

Note:

1. This feature is not fully tested by Intel® VROC. If the user encounters issue with multipath capable drive, the user can still disable multipathing by adding the "nvme_core.multipath=N" option to the kernel command line as the workaround.

2. This feature has the limitation described below:
   Limitation: Auto rebuild* and hot insert to the same slot is not supported when using global spare drives which are multipath capable drives with multipath enabled in Linux.
   *The auto rebuild case applies to the spare in a container which doesn't include the target volume.

Source Code	Release
mdadm	>= 4.2
ledmon	>= 1.0.0

Base OS Inbox Support	OS version
mdadm	
RHEL	>= RHEL 8.5 or >= RHEL 9.0
SLES	>= SLES15 SP4
Ubuntu	>= 22.04
ledmon	
RHEL	>= RHEL 9.5
SLES	>= SLES15 SP7 Note: SLES15 SP6 inbox is 0.97
Ubuntu	Not supported Note: Ubuntu 24.04 inbox is 0.97

8.3 Preserve User Data While Migrating Non-RAID Disk to Intel® VROC RAID

Intel® VROC Linux* products currently do not plan to support preserving the user data while migrating non-RAID disk to Intel® VROC RAID.

§§

Appendix A Related Documentation

A.1 Relevant Specifications

Specification	Location
UEFI Specifications 2.4	http://uefi.org
UEFI Platform Initialization Specification version 1.2	http://www.uefi.org/specsandtesttools
UEFI Shell Specification version 2.0	http://www.uefi.org/specsandtesttools
ATA Command Set 2	http://www.t13.org/Documents/UploadedDocuments/docs2009/d2015r2-ATAATAPI_Command_Set_-_2_ACS-2.pdf
ATA8-ACS-8	http://www.t13.org/Documents/UploadedDocuments/docs2007/D1699r4c-ATA8-ACS.pdf
SATA I Specification	http://www.serialata.org
SATA II Specification	http://www.serialata.org
SATA III Specification	http://www.sata-io.org/documents/SATA-Revision-3.0-Press-Release-FINAL-052609.pdf
Serial Attached SCSI - 2 (SAS-2)	http://www.t10.org
NVM Express Revision 1.2 Specification	https://nvmexpress.org/wp-content/uploads/NVM_Express_1_2_Gold_20141209.pdf

§§

Appendix B Hardware Compatibility

Refer to the latest Intel® Virtual RAID on CPU (Intel® VROC) Supported Configurations.

§§