

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

Technical Product Specification - Linux

March 2017
Version 1.2



Revision History

Revision	Description	Date
1.0	Initial release.	December 2016
1.1	Production Version (PV) Release	March 2017
1.2	Minor Updates	April 2017



Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

No computer system can provide absolute security. Requires an enabled Intel® processor, enabled chipset, firmware and/or software optimized to use the technologies. Consult your system manufacturer and/or software vendor for more information.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com.

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

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

All products, computer systems, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2017 Intel Corporation. All rights reserved. 0716/JL/JT



Table of Contents

Revision History	2
1 Introduction	6
1.1 Intended Use	6
1.2 Intended Audience	6
1.3 Terminology.....	6
2 Product Overview	9
2.1 Intel VROC and Intel RSTe	9
2.2 Supported Operating Systems.....	9
2.3 Supported Platforms/Chipsets/SKUs	10
2.4 New Features	10
3 Intel® Volume Management Device (Intel® VMD)	12
3.1 Multiple VMD Controllers	12
3.2 Intel VMD Method of LED Management	12
3.3 Intel VMD Surprise Removal of NVMe Devices	13
3.4 Intel VMD Error Management	13
4 Intel RSTe Package Components	15
4.1 Intel VROC Package.....	15
4.2 Intel RSTe PreOS Components.....	17
4.3 Intel ASM	20
5 Intel® VROC New Features	21
5.1 Intel VMD.....	21
5.2 Intel VROC Features and Functionality	22
5.3 Intel VROC RAID PreOS Features and Functionality	25
6 Intel RSTe 5.X Common Features.....	41
6.1 Matrix Storage Manager	41
6.2 Intel RSTe 5.X PreOS Features and Functionality	41
6.3 Intel RSTe 5.X Linux Monitoring and Management	46
6.4 MDRAID Sysfs Components	68
7 Intel® RSTe Key Features.....	72
7.1 Intel RSTe PreOS Features and Functionality	72
7.2 Intel RSTe Miscellaneous Features and Functionality	81
8 Unsupported Features.....	82
Appendix A: Related Documentation	83
Appendix B: External Hardware Capability	84
Appendix C: Intel C620 Series Chipset Legacy OROM Boot Option	86
Appendix D: RSTe SATA Port Bitmap Implementation	88



Legacy OROM..... 88

UEFI Driver..... 89

EFI_DEVICE_PATH_PROTOCOL..... 89



1 Introduction

This is the Technical Product Specifications for the Intel® Rapid Storage Technology enterprise (Intel® RSTe) 5.X 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 Intel RSTe Linux* based drivers and tools, the Intel® Accelerated Storage Manager (ASM) web-based remote management tool and the Graphical User Interface (GUI).

1.1 Intended Use

This document is intended to provide high level information on the technical features of the Intel® RSTe 5.X family of products (both Intel VROC and Intel RSTe).

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 RSTe family of products. It contains only information pertaining to the Intel RSTe family of products.

1.3 Terminology

The following acronyms will be used throughout this document.

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
ASM	Intel® Accelerated Storage Manager
BIOS	Basic Input / Output System
BBE	Boot Behind Expander
Chipset	A term used to define a collection of The PNHCI 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



Term	Definition
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	Giga-byte
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 drive when the system is powered on.
Hot Spare Disk	A disk flagged so as 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	Mutiple RAID volumes residing within the same Array
MB	Mega-bytes
NAI	Notification Area Icon
NCQ	Native Command Queuing
NTFS	NT File System
NVMe	Non Volatile Memory PCI Express
ODD	Optical Disk Drive
ODM	Original Design Manufacturer
OEM	Original Equipment Manufacturer
OROM	Option ROM



Term	Definition
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
Volume Roaming	Moving of RAID volumes and all it's array members between two enabled platforms to maintaining data availability or different supported operating systems.
RAID	Redundant Array of Independent Disks
RHEL	Redhat Enterprise Linux
RSTe	Rapid Storage Technology enterprise
RWH	RAID Write Hole
SATA	Serial ATA
sSATA	Secondary 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
UI	User Interface
UEFI	Unified Extensible Firmware Interface
VMD	Intel Volume Management Device
VMD Domain	A grouping of drives behind a single Intel VMD
VROC	Virtual RAID on CPU



2 Product Overview

The Intel RSTe 5.X family of products provide enterprise RAID solution for both NVMe SSD and SATA devices for the enterprise servers, workstations and some high-end desktops.

- Intel Virtual RAID on CPU (Intel VROC) provides an enterprise RAID solution on platforms that support Intel Volume Management Device (VMD).
- Intel RSTe SATA provides an enterprise RAID solution for SATA devices connected to SATA/sSATA Intel Platform Control Hub (PCH) configured for RAID mode.

Intel RSTe 5.X is family of products designed for platforms that support Intel VMD. It is also the high level blanket product reference for both Intel VROC and Intel RSTe. Moving forward in this document, Intel VROC product package will refer to component installed to support Intel VMD enabled platforms while Intel RSTe will refer to the product package installed to support SATA devices (HDDs and SSDs) attached to the PCH (SATA controller) in RAID Mode.

Within the Linux* OS, the primary configuration software to manage Intel RSTe RAID is the **mdadm** application, a native Linux* tool that is used exclusively with RSTe on Linux.

2.1 Intel VROC and Intel RSTe

Intel VROC refers to the product that operates on platforms with Intel VMD. This product is comprised of the PreOS components that support booting from a RAID volume, the Intel VMD driver and the mdadm components to manage Linux RAID volumes using the Intel Matrix Storage Manager (MSM) metadata format. The PreOS is provided by Intel. The Intel VMD driver is developed and maintained by Intel in collaboration with the open source community. The mdadm components are also up streamed to the open source community.

Intel RSTe refers to the product that operates on the platform PCH or SATA/sSATA Controller. This product is comprised of the PreOS components that support booting from a RAID volume and the mdadm component to manage Linux RAID volumes using the Intel MSM metadata format. The PreOS is provided by Intel. The mdadm components are open source and are part of the open source community. The device driver used is the community PCH driver.

An important requirement for utilizing either Intel VROC or Intel RSTe is the need to specify the Intel MSM metadata format. This is done by using the “-e imsm” option when creating an mdadm container

Another feature of the Intel IMSM metadata is the ability to roam a container between Linux and Microsoft Windows* host systems.

Intel RSTe supports the following RAID levels: RAID 0, 1, 5, 10 and Intel® Matrix RAID

Note: Further information will be provided on RSTe as part of the MDRAID architecture

2.2 Supported Operating Systems

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

- Red Hat Enterprise Linux (RHEL) 7.3 (Intel VROC with Intel Package)
- SUSE Linux Enterprise Server (SLES) 12 SP3 (Intel VROC inbox support)



- Red Hat Enterprise Linux (RHEL) 6.7, 6.8, 7.2 and 7.3 (Intel RSTe inbox support)
- SUSE Linux Enterprise Server (SLES) 11 & 12 (Intel RSTe inbox support)

NOTE: Intel VMD has been upstreamed and is included in the Linux 4.10 kernel

2.3 Supported Platforms/Chipsets/SKUs

The Intel RSTe 5.X product package was designed to work with, tested and validated on Intel Customer Reference Boards (CRBs) outlined in this section.

2.3.1 Supported CPU for Intel VROC

CPU	Platform	VMD Device ID	# of VMD
Intel® Xeon® processor E5/E7 v5 Skylake - W	Basin Falls workstation	201D	3 per CPU
Intel® Xeon® processor E5/E7 v5 Skylake – SP	Purley server and workstation	201D	3 per CPU

2.3.2 Supported Chipset SKU Overview

	Platform	RAID controller Device ID	# of ports
Intel® 620 Series chipset	Purley server and workstation	2826 (SATA) 2827 (sSATA)	8 SATA 6 sSATA
Intel® C422 Series chipset	Basin Falls workstation	2826 (SATA)	8 SATA

2.4 New Features

A few of the new features introduced with Intel RSTe 5.X that will apply to both Intel RSTe and Intel VROC are:



- RAID 5 RAID Write Hole Closure
- Support for drives with Native 4K size sector
 - Intel RSTe 5.0 does not support the mixing of Native 4K and 512(e) base drives in a single RAID volume.

With the introduction of the Intel VROC product, there are three modes of operation:

SKU	HW key required	Key features
Pass-thru	Not needed	<ul style="list-style-type: none"> • Pass-thru only (no RAID) • Non-Intel NVMe SSD support • LED Management • Hot Plug Support • RAID 0 support for Intel P3608 NVMe SSDs
Standard	Standard Key	<ul style="list-style-type: none"> • Pass-thru SKU features • RAID 0, 1, 10
Premium	Premium Key	<ul style="list-style-type: none"> • Standard SKU features • RAID 5 • RAID 5 Write Hole Closure

Please consult the user's platform guides to identify which PCIe slots are managed by the CPU.

Please review this entire document for the complete list of new features in the Intel RSTe / Intel VROC family of products.

3 Intel® Volume Management Device (Intel® VMD)

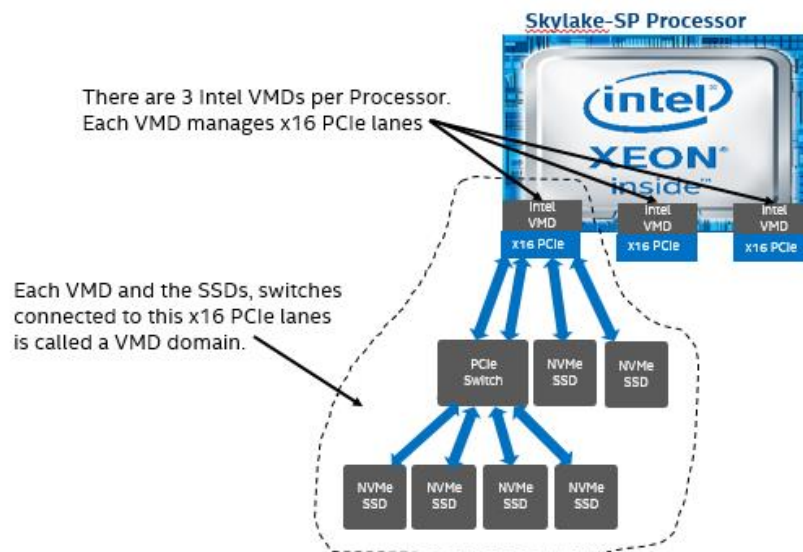
With the introduction of the Intel® Xeon® processor E5/E7 v5 (Sky Lake) product family, one of the key features included is the Intel Volume Management Device (VMD). The Intel VMD is an integrated PCIe endpoint within the CPU root complex. The class code for the Intel VMD device is a RAID controller. Intel VMD driver support is provided with Intel VROC package and includes the following:

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

Each Intel Skylake CPU provides 48 PCIe lanes which is subdivided into 3 domains. Each VMD domain manages x16 lanes. Intel VMD can be turned on/off on x4 lane granularity and supports either NVMe SSD device or PCIe switch device.

3.1 Multiple VMD Controllers

Each Intel® Xeon® processor E5/E7 v5 (Skylake) product family contains 3 Intel VMD controllers.



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.

For details on how to implement the Intel VMD Method, refer to the *LED Management for PCI Express SSDs with Intel RAID* document (IBL Doc ID: 334005-004US).

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

Intel VMD technology handles Advanced Error Reporting (AER) Handling and functions as a crash dump and hibernate target within Linux OS, therefore providing protection to the rest of the PCIe bus, preventing whole system shut down. Upon receiving an interrupt at MSI-X 0 for each root port, the Intel® VMD driver uses a sophisticated algorithm to determine the precise device/volume to apply error handling based on the error logged. The actions taken are specific to the error on the device, and preserve the remaining eco system.

Intel VMD will handle the following errors. If non-fatal error is encountered, the Intel VROC driver will log the error and standard recovery efforts are taken. If fatal error is encountered, the Intel VROC driver will treat the error as a "device surprise hot removed". But system shall continue to operate without hang, reboot, or crash.

Advanced Uncorrectable Error	Severity Register
Data Link Protocol Error	Fatal (1b).
Surprise Down Error	Fatal (1b).
Poisoned TLP	Non-Fatal (0b).
Flow Control Protocol Error	Fatal (1b).
Completion Timeout	Non-Fatal (0b).
Completer Abort	Non-Fatal (0b).
Unexpected Completion	Non-Fatal (0b).
Receiver Overflow	Fatal (1b).
Malformed TLP	Fatal (1b).
ECRC Error	Non-Fatal (0b).
Unsupported Request Error	Non-Fatal (0b).



ACS Violation	Non-Fatal (0b).
Uncorrectable Internal Error	Fatal (1b).
MC Blocked TLP	Non-Fatal (0b).
Atomic Op Egress Blocked	Non-Fatal (0b).
TLP Prefix Blocked	Non-Fatal (0b).



4 Intel RSTe Package Components

This section outlines the individual components that are included in the Intel RSTe 5.X Linux package that is delivered to customers. The specific details on each component are in the following sections of this document.

The Intel RSTe 5.X package is comprised of the Intel VROC package, and the Intel ASM installer.

Support for Intel RSTe (SATA Controller in RAID mode) is already included in the supported Linux distribution general releases.

4.1 Intel VROC Package

The Intel VROC production package is broken into a PreOS component, Intel RSTe VROC Linux ISO image that includes the RPM binary files, the corresponding source code and the Intel ASM package.

4.1.1 Intel VROC PreOS Components

The RSTe VROC PreOS components of:

- VMDVROC_1.efi/VMDVROC_2.efi – These two UEFI device driver files to be included in the platform BIOS. Both of these files are required and work together to properly support the Intel VMD controller when enabled. When these files are properly incorporated into the platform BIOS, the BIOS setup will have the ability to manage NVMe drives (create/delete RAID Volumes) connected to the Intel VMD controller. Other features include:
 - Seeing and reporting (to the BIOS) any NVMe SSD attached to the Intel VMD
 - Installing an OS onto a drive (or RAID Volume) managed by the Intel VMD
 - Recognizing of Intel VROC RAID Upgrade Hardware Keys
 - Proper configuration of the system based of the Intel VROC RAID Upgrade Hardware Key seen
- HWKeyCheckRSTeRS.efi – This utility can be used by OEMs to test Intel VROC RAID Upgrade Hardware Key configuration via a UEFI Shell environment. This help to make sure the system can properly see the Intel VROC RAID Upgrade Hardware Key plugged into the system.
 - This tool must match the major release version of the UEFI VROC driver. To find this release version, boot to the efi shell and type “drivers -sfo” and note the major release version which is the first number (capital X). Look for “Intel(R) VROC with VMD Technology **AA.BB.CC.DDDD**”. Compare this value to what is reported in the Customer Release Notes associated with the latest release. Using a Fat32 formatted USB device, users can download any release of this tool that matches the major release of RSTe UEFI driver found in the previous step. Attach the device to the system under test. Navigate to the file system of the fat32 formatted USB disk. Type “map -r” to find the FS#. Type FS#. Then run the tool (ex: HWKeyCheckRSTeRS.efi).
 - The output from this efi tool can also be saved as a .txt file to attach to an IPS sighting filed against RSTe VROC (ex: HWKeyCheckRSTeRS.efi > HWKeyCheckRSTeRS _Output.txt)
- RCfgRSTeRS.efi – This utility can be used by the OEMs to manage/test Intel VROC capabilities from a UEFI Shell environment. This utility must be copied to and executed from a USB key.



- This tool must match the major release version of the UEFI VROC driver. Using a Fat32 formatted USB device, users can download any release of this tool that matches the major release of RSTe UEFI driver found in the previous step. Attach the device to the system under test. Navigate to the file system of the fat32 formatted USB disk. Type "map -r" to find the FS#. Type FS#. Then run the tool (ex: RCFgRSTeRS.efi).
- The output from this efi tool can also be saved as a .txt file to attach to an IPS sighting filed against RSTe VROC (ex: RCFgRSTeRS.efi > RCFgRSTeRS_Output.txt)
- RCmpVROC.efi – This is a debug utility to help verify that the two efi drivers have been properly configured/incorporated into the BIOS. When reporting an issue to Intel, the output from this file will most likely be asked for by the Intel representative.
 - This tool must match the major release version of the UEFI VROC driver. Attach the device to the system under test. Navigate to the file system of the fat32 formatted USB disk. Type "map -r" to find the FS#. Type FS#. Then run the tool (ex: RCmpVROCRS.efi).
 - The output from this efi tool can also be saved as a .txt file to attach to an IPS sighting filed against RSTe VROC (ex: RCmpVROC.efi > RCmpVROC_Output.txt)
- LedToolVMDRSTeRS.efi – This is UEFI base tool that can be used to test LED connectivity to determine if the hardware is setup correctly. The tool provides the ability to simulate a Locate, Fault or Rebuild signal to the enclosure.
 - This tool must match the major release version of the UEFI VROC driver. Using a Fat32 formatted USB device, users can download any release of this tool that matches the major release of RSTe UEFI driver found in the previous step. Attach the device to the system under test. Navigate to the file system of the fat32 formatted USB disk. Type "map -r" to find the FS#. Type FS#. Then run the tool (ex: LedToolVMDRSTeRS.efi /D1 1 /TIMEOUT 2). This will generate a Locate signal for 2 seconds.
 - LedToolVMDRSTeRS.efi [/?] [/HELP] [/L] [/Dn state-number] [/TIMEOUT time]
/?, /HELP Displays Help Screen. Other options are ignored.
/L Lists all detected disks.
/Dn Sends the specified state [0-3] to device selected by 'n'.
Valid state number is from 0 to 3 and they are interpreted as following:
0 = LED_OFF, 1 = LOCATE, 2 = FAULT, 3 = REBUILD
/TIMEOUT After sending the state, waits the specified number of seconds and then turns all leds off.
Example:
"LedtoolVMD.efi /D1 2 /D2 1 /TIMEOUT 2"

This will set state FAULT on device 1 and state LOCATE on device 2. After 2 secs delay, messages will be reset.

4.1.2 Intel VROC ISO Images

The RSTe VROC ISO image (**rste-*version identification_distro identification*.iso**) 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 installation will install the latest device driver, the Intel LED management application and corresponding updates to mdadm. This installation process may also include the installation of the Intel ASM. This is a web based GUI to manage Intel VROC and Intel RSTe RAID volumes.



4.2 Intel RSTe PreOS Components

The RSTe PreOS components are comprised of Legacy Option ROM (OROM) support along with UEFI support. The Intel RSTe PreOS components are:

4.2.1 EFI Support for the SATA Controller

- **SataDriver.efi** – This is the UEFI device driver to support the SATA Controller when configured for RAID mode. This file is to be included in the platform BIOS. When properly incorporated, it will provide the ability to manage SATA drives (create/delete RAID Volumes) connected to the SATA Controller. Other features included:
 - Seeing and reporting (to the BIOS) any SATA drive attached to the SATA Controller when configured in RAID mode.
 - Installing an OS onto a drive (or RAID Volume) managed by the SATA Controller when configured in RAID mode.
- **LedToolSata.efi** – This is UEFI base tool that can be used to test SGPIO LED connections to determine if the hardware is setup correctly. This tool needs to be copied to a USB drive and executed from a UEFI Shell.
 - The tool must match the major release version of the UEFI RSTe driver. To find this release version, boot to the efi shell and type “drivers” and note the major release version which is the first number (capital X). Look for “Intel® RSTe RAID X.x.x.xxxx Drive...”. Using a Fat32 formatted USB device, users can download any release of this tool that matches the major release of RSTe UEFI driver found in the previous step. Attach the device to the system under test. Navigate to the file system of the fat32 formatted USB disk. Type “map –r” to find the FS#. Type FS#. Then run the tool (ex: LedToolSata.efi).
 - Help for SATA/sSATA LedTool:


```
=====
Intel(R) SGPIO Led Testing Utility for Patsburg AHCI Controller
LedTool.efi [/?] [/HELP] [/D:state_number] [/D0-5:state_number] [/TIMEOUT:time]
/?          Same as /HELP. Displays Help Screen. Other options ignored.
/D          Send the specified state to all ports. /D0-9 option will be ignored.
/D0-5       Send the specified state to selected port.
/TIMEOUT    After sending the states waits the specified number of seconds and then turns off all leds
            and performs a resets the Messages sending logic.

NOTE: After a reset, the first sent state will turn on all other LEDs.

Valid state_number is from 0 to 5 and they are interpreted as following:
0 = NO_DISK, 1 = NO_ACTIVITY, 2 = ACTIVITY, 3 = LOCATE, 4 = FAIL, 5 = REBUILD.
Example use: "Ledtool.efi /D2 0 /D0 2 /TIMEOUT 2" this will set state NO_DISK on port 2 and state
ACTIVITY on port 0. After 2 secs delay Messages will be reset.
```
- **RCFgSata.efi** - This utility can be used by the OEMs to manage/test Intel RSTe capabilities from a UEFI Shell environment. This utility must be copied to and executed from a USB key in a UEFI Shell.
 - The tool must match the major release version of the UEFI RSTe driver. Using a Fat32 formatted USB device, users can download any release of this tool that matches the major release of RSTe UEFI driver found in the previous step. Attach the device to the system under test. Navigate to the file system of the fat32 formatted USB disk. Type “map –r” to find the FS#. Type FS#. Then run the tool (ex: RCFgSata.efi).



- The output from this efi tool can also be saved as a .txt file to attach to an IPS sighting filed against RSTe VROC (ex: RCFgSata.efi > RCFgSata_Output.txt)
- RCmpSata.efi - This is a debug utility to help verify that the SATA efi drivers have been properly configures/incorporated into the BIOS. When reporting an issue to Intel, the output from this file will most likely be asked for by the Intel representative. This utility must be copied to and executed from a USB key in a UEFI Shell.
 - The tool must match the major release version of the UEFI RSTe driver. Using a Fat32 formatted USB device, users can download any release of this tool that matches the major release of RSTe UEFI driver found in the previous step. Attach the device to the system under test. Navigate to the file system of the fat32 formatted USB disk. Type "map -r" to find the FS#. Type FS#. Then run the tool (ex: RCmpSata.efi).
 - The output from this efi tool can also be saved as a .txt file to attach to an IPS sighting filed against RSTe VROC (ex: RCmpSata.efi > RCmpSata_Output.txt)
- RClrSata.efi – This is a utility provided that will clear off any RAID metadata from a SATA drive attached to the SATA controller. This will basically wipe out any data that is on the drive that this tool is executed on. So great care must be taken. This utility must be copied to and executed from a USB key in a UEFI Shell.

4.2.2 EFI Support for the sSATA Controller

- sSataDriver.efi – This is the UEFI device driver to support the sSATA Controller when configured for RAID mode. This file is to be included in the platform BIOS. When properly incorporated, it will provide the ability to manage sSATA drives (create/delete RAID Volumes) connected to the sSATA Controller. Other features included:
 - Seeing and reporting (to the BIOS) any SATA drive attached to the sSATA Controller when configured in RAID mode.
 - Installing an OS onto a drive (or RAID Volume) managed by the sSATA Controller when configured in RAID mode.
- LedToolsSata.efi – This is UEFI base tool that can be used to test SGPIO LED connections to determine if the hardware is setup correctly. This tool needs to be copied to a USB drive and executed from a UEFI Shell.
- RCFgsSata.efi - This utility can be used by the OEMs to manage/test Intel RSTe capabilities from a UEFI Shell environment. This utility must be copied to and executed from a USB key in a UEFI Shell.
 - The tool must match the major release version of the UEFI RSTe driver. To find this release version, boot to the efi shell and type "drivers" and note the major release version which is the first number (capital X). Look for "Intel® RSTe RAID X.x.x.xxxx Drive...". Using a Fat32 formatted USB device, users can download any release of this tool that matches the major release of RSTe UEFI driver found in the previous step. Attach the device to the system under test. Navigate to the file system of the fat32 formatted USB disk. Type "map -r" to find the FS#. Type FS#. Then run the tool (ex: RCFgsSata.efi).
 - The output from this efi tool can also be saved as a .txt file to attach to an IPS sighting filed against RSTe VROC (ex: RCFgsSata.efi > RCFgsSata_Output.txt)
- RCmpsSata.efi - This is a debug utility to help verify that the sSATA efi driver has been properly configures/incorporated into the BIOS. When reporting an issue to Intel, the output from this file will most likely be asked for by the Intel representative. This utility must be copied to and executed from a USB key in a UEFI Shell.



- The tool must match the major release version of the UEFI RSTe driver. Attach the device to the system under test. Navigate to the file system of the fat32 formatted USB disk. Type "map -r" to find the FS#. Type FS#. Then run the tool (ex: RCmpsSata.efi).
 - The output from this efi tool can also be saved as a .txt file to attach to an IPS sighting filed against RSTe VROC (ex: RCmpsSata.efi > RCmpsSata_Output.txt)
- RClrsSata.efi – This is a utility provided that will clear off any RAID metadata from a SATA drive attached to the sSATA controller. This will basically wipe out any data that is on the drive that this tool is executed on. So great care must be taken. This utility must be copied to and executed from a USB key in a UEFI Shell.
 - The tool must match the major release version of the UEFI RSTe driver. Attach the device to the system under test. Navigate to the file system of the fat32 formatted USB disk. Type "map -r" to find the FS#. Type FS#. Then run the tool (ex: 'RCfgSata.efi 0 1 2' - it clears metadata in all selected drives).
 - Intel(R) UEFI RAID Clear Metadata Utility for Serial ATA
 - HELP--
 - : Selects disks to have deleted metadatas.
 - List should be delimited by spaces. Evaluation example:
 - 'RCfgSata.efi 0 1 2' - it clears metadata in all selected drives
 - /I Displays Drives Information Screen.
 - /? Displays Help Screen.

4.2.3 Legacy OROM Support for the SATA Controller

- SataOrom.bin - This is the Legacy OROM image to support the SATA Controller when configured for RAID mode. This file is to be included in the platform BIOS. When properly incorporated, it will provide the ability to manage SATA drives (create/delete RAID Volumes) connected to the SATA Controller. Other features included:
 - Seeing and reporting (to the BIOS) any SATA drive attached to the SATA Controller when configured in RAID mode.
 - Installing an OS onto a drive (or RAID Volume) managed by the SATA Controller when configured in RAID mode.
- RCfgSata.exe - This utility can be used by the OEMs to manage/test Intel RSTe capabilities from a DOS Command environment. This utility must be copied to and executed from a DOS bootable USB key with the target environment is booted from this DOS bootable USB key.
- RCmpSata.exe - This is a debug utility to help verify that the SATA Legacy OROM image has been properly configured/incorporated into the BIOS. When reporting an issue to Intel, the output from this file will most likely be asked for by the Intel representative. This utility must be copied to and executed from a DOS bootable USB key with the target environment is booted from this DOS bootable USB key.

4.2.4 Legacy OROM Support for the sSATA Controller

- sSataOrom.bin - This is the Legacy OROM image to support the sSATA Controller when configured for RAID mode. This file is to be included in the platform BIOS. When properly incorporated, it will provide the ability to manage SATA drives (create/delete RAID Volumes) connected to the sSATA Controller. Other features included:



- Seeing and reporting (to the BIOS) any SATA drive attached to the sSATA Controller when configured in RAID mode.
- Installing an OS onto a drive (or RAID Volume) managed by the sSATA Controller when configured in RAID mode.
- RCfgsSata.exe - This utility can be used by the OEMs to manage/test Intel RSTe capabilities from a DOS Command environment. This utility must be copied to and executed from a DOS bootable USB key with the target environment is booted from this DOS bootable USB key.
- RCmpsSata.exe - This is a debug utility to help verify that the sSATA Legacy OROM image has been properly configures/incorporated into the BIOS. When reporting an issue to Intel, the output from this file will most likely be asked for by the Intel representative. This utility must be copied to and executed from a DOS bootable USB key with the target environment is booted from this DOS bootable USB key.

4.3 Intel ASM

The Intel Accelerated Storage Manager project aims to provide REST API capabilities for various, supported Intel storage products.

Intel ASM is complete RESTful solution providing, among others, features like REST API for system and devices information, RAID information through RSTe plugin, authentication and authorization using OAuth 2.0, and web page and application serving through built-in web server. For further information on the Intel ASM, please see Intel Accelerated Storage Manager Linux Administration Guide.



5 Intel® VROC New Features

The Intel VROC 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 for NVMe devices connected to PCIe managed by VMD.

Features	<ul style="list-style-type: none"> • Surprise Hot-Plug • LED Management (VMD Method) • Error Management (VMD First) • RAID Write Hole Closure (RAID 5 w/ Premium activation key) • NVMe RAID Boot • Enhanced GUI • UEFI HII • Drive Roaming • RAID 0/1/5/10 	<ul style="list-style-type: none"> • EFI Extended NVMe Pass-Through Command Support • EFI NVM Express Pass-Through Protocol Support • 4K native HDD • NVMe Deallocate/TRIM • Intel/3rd Party NVMe SSD support Through Intel VROC RAID Upgrade Hardware Key
Utilities	<ul style="list-style-type: none"> • Configuration and Management Utilities 	

5.1 Intel VMD

As previously stated, Intel VMD is a new feature introduced with the Intel® Xeon® processor E5/E7 v5 (Sky Lake) product family. This section is intended to show how to enable and configure this functionality.

5.1.1 Intel VMD Enabling

This set of instructions will cover how to enable Intel VMD in an Intel Customer Reference Board (CCB).

NOTE: Please refer to the instructions provided by the user's platform BIOS vendor because those instructions will most likely be different from these instructions.

Step 1: Immediately following POST, select the option that will allow the user to access the BIOS setup menu. This example uses F2.

Step 2: For the Intel CRB reference BIOS, The user will want to use the arrow keys to move the cursor to **the EDKII Menu** (it will become highlighted) and press <Enter>.

Step 3: Using the arrow keys, move the cursor to **Socket Configuration** and press <Enter>.

Step 4: Using the arrow keys, move the cursor to **Intel® VMD technology** and press <Enter>.

Step 5: Using the arrow keys, move the cursor to **Intel® VMD for Volume Management Device on SocketX**. (X representing the number of the socket the user is to modify).

Step 6: Using the arrow keys, move the cursor to select the desired PStack# to **enable** or **disable** Intel VMD according which hardware configuration is being used.

Step 7: Repeat Step 6 for each PStackX to be **enabled** or **disabled**.

NOTE: Please consult with the user's Platform BIOS manufacturer documentation for a complete list of options that can be configured.



5.1.2 Number of supported VMD domains

The Intel VROC product will support 3 VMD domains per platform CPU on the platform.

5.2 Intel VROC Features and Functionality

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

5.2.1 Intel VROC RAID Modes of Operation Upgrade Hardware Key for RAID Support

The Intel VROC package will support three operational SKUs (or modes) of the platform. They are the Intel VROC Pass-thru SKU, Intel VROC Standard SKU and 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.

This Intel VROC RAID Upgrade Hardware Key is available for purchase for use on platforms based on Intel Purley platforms with SKX microarchitecture. The Intel VROC RAID Upgrade Hardware 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 to be loaded when Intel VMD is enabled. Please refer to the user's platform documentation for the location of the Intel VROC RAID Upgrade Hardware Key placement.

NOTE: It is assumed that all Intel VROC supported platforms contain support (physical hardware header) to support the Intel VROC RAID Upgrade Key. Please reference Activation Key Product Architecture Specification Revision 1.2 (or later) for additional information.

5.2.1.1 Intel VROC Pass-thru SKU

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 with the exception of support for RAID 0 on Intel P3608 NVMe SSDs.

5.2.1.2 Intel VROC Standard SKU

The Intel VROC Standard SKU configuration is enabled with the Intel VROC RAID Standard Upgrade Key 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 Intel VMDs that are enabled.

5.2.1.3 Intel VROC Premium SKU

The Intel VROC Premium SKU configuration is enabled with the Intel VROC RAID Premium Upgrade Key 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 Intel VMDs that are enabled.



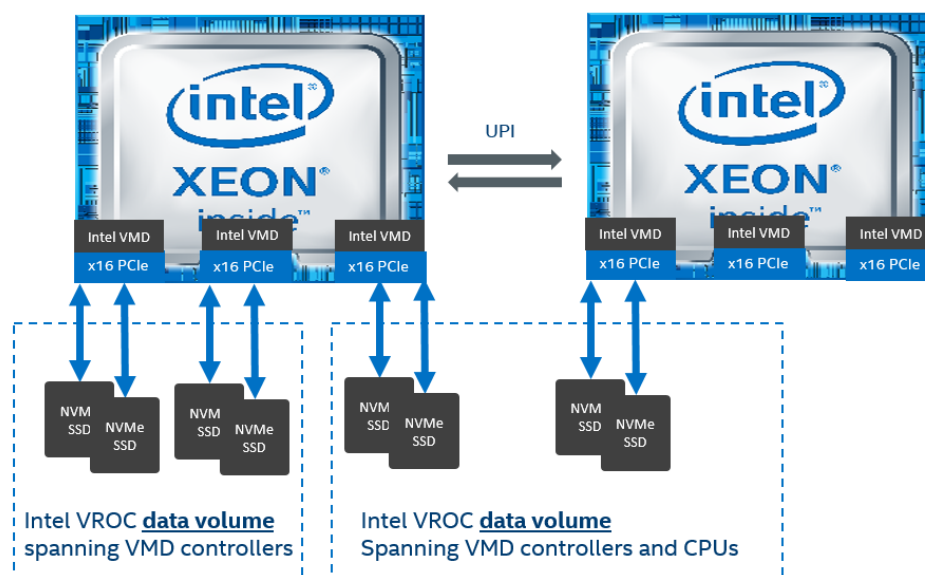
5.2.2 RAID volume controller binding for VMD configuration

Intel VROC product will RAID volumes on NVMe disks connected to a VMD domain shall be reported as disks. These disks shall be reported as connected to the test VMD domain where the volume member disks are connected to for bootable RAID volumes (all member drives on a single VMD domain).

5.2.3 Spanning Intel VROC RAID Data Volumes across VMD Domains

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

*Although supported, spanning across CPUs is generally not recommended. This configuration may incur performance penalty.



5.2.3.1 Intel VROC on Pass-through Boot Drives

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

5.2.3.2 Intel® VROC Spanned RAID Volumes across VMD Controllers

Intel VROC 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 PreOS/BIOS. The following instructions will cover creation of spanned using the BIOS interface.

The following assumptions will be made about the configuration and setup for configuration of RAID volumes that are spanned across VMD Controllers:

1. It is known how to access the Configuration BIOS menus.



2. All pertinent vendor documentation has been reviewed to ensure that the hardware is configured to allow for more than one VMD domain has been established.
3. The Intel VMD has been properly enabled within the BIOS.
4. The appropriate Intel VROC RAID Upgrade Standard/Premium Key has been installed.
5. The appropriate number of drives have been installed to create the desired RAID volume.

Step 1: Enter into the BIOS configuration the 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.

Step 5: Select the RAID level by pressing the <Enter> to access the Menu and scroll through the available values, then press <Enter> key to select the desired RAID type.

Step 6: Ensure that the user's vendor supports spanned RAID volumes. To enable, toggle the status from the default status off, to on.

To enable the RAID to be spanned over multiple controllers, highlight the < >, and press <Enter>. This will open a small menu box with two values, blank and X. Blank indicates a setting that is disabled. To enable RAID spanned over VMD Controllers, highlight X and press the <Enter> key.

Step 7: Using the arrow keys to 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 down arrow 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 selecting the current value and pressing the <Enter> key. This will allow the user to scroll through the available values, then press the <Enter> key to set the desired value. RAID 1 will by default place this setting at 128 KB and cannot be modified.

Step 10: Select the volume capacity and press the <Enter> key. The default value indicates the maximum volume capacity using the selected disks and calculating that at a total of 95% of the value due to disk coercion. To enter a value smaller than the maximum based on RAID type selected, press the <Enter> key. Type in the value of the desired RAID volume, and press <Enter> to save. This value is calculated in bytes. For demonstration, 700GB is 716800. ($700 * 1024 = 716800$). The user may also perform the math based on the base of 1000 instead of base 1024. This is left to the discretion of the administrator.

Step 11: Navigate to and select Create Volume, then press <Enter>.

Step 12: The following message will be displayed: "You have selected NVMe drives that are connected to multiple VMD controllers. Please note that if the user continues and create a RAID volume with drives from multiple VMD controllers that RAID volume will not be bootable in a Linux OS environment. Press 'y' to create, 'n' to discard".

Press <Enter> to create the RAID volume.



Note: The message is a warning regarding Windows and does not apply to Linux. This is a BIOS configuration platform and can be compatible with either Windows Server or Linux distributions. However spanned boot volumes in Windows are not supported.

Step 13: The RAID volume configuration can be verified by looking at the RAID Volume Information screen.

If the RAID volume 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. If there are insufficient disks or space on the other disks to create a RAID volume, the user may receive an error when creating secondary volumes.

Step 14: To exit the user interface, press <Esc>. Press <Esc> again, and the user will be returned to the main menu. Navigate to and select Reset, and press <Enter> to reboot the system and begin Operating System installation.

5.2.4 Intel RSTe NVMe Compliance

The Intel VROC release package adheres to the NVMe Specification version 1.2.

5.2.5 Intel VROC Driver downgrade

If the driver is downgraded to a previous Intel VROC driver version, the Intel VROC RAID volumes will be maintained (not broken). After upgrading back to the latest version, the RAID volumes will be able to become fully operational.

5.2.6 Intel VROC Support Maximum number of Drives in a RAID Volume

The Intel VROC product will support the ability to create RAID volumes from up to 24 NVMe drives connected to the Intel VMD controllers (directly connected or in switches).

5.2.7 Intel VROC Support for Maximum RAID Members

The Intel VROC product will support the ability to create 2 RAID volumes per Array on NVMe drives connected to the Intel VMD controllers. An Array can be created on a minimum of two drives.

5.3 Intel VROC RAID PreOS 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 PreOS component. The Intel VROC RAID PreOS 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:

1. Initialize and enumerate NVMe devices managed by the Intel VMD (when enabled)
2. 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.
3. Provides a RAID management user interface to the BIOS to be able to manage RAID volumes NVMe devices connected to the Intel VMD controller.



5.3.1 Intel® VROC for UEFI

For information on how to incorporate the Intel VROC UEFI images into the platform BIOS, please see [Intel® RSTe 5.X Unified Extensible Firmware Interface \(UEFI\) Drivers](#).

5.3.2 Intel VROC UEFI HII

Intel VROC product support RAID management via the Intel RSTe UEFI HII Protocol. This is only supported in an Intel VROC Enabled configuration. The HII is part of the UEFI Protocol and provides a way for customers to manage RAID Volumes behind the Intel VMD controller in the BIOS environment.

NOTE: Intel VROC does not support nor provide a Legacy Option ROM image as part of the Intel VROC PreOS package.

5.3.3 Intel VROC UEFI Maximum Number of Drives

The Intel VROC UEFI drive supports a maximum number of 24 NVMe across all Intel VMD controllers.

5.3.4 EFI_NVME_EXPRESS_PASS_THRU_PROTOCOL Support

The Intel NVMe API architecture supports a private UEFI NVMe Pass-Thru protocol which will be supported by both Intel® VROC UEFI driver.

This section describes the private protocol GUID, protocol interface, function prototypes, associated structures, as well as an example reference code for using the private Intel UEFI NVMe Pass-Thru Protocol to send a subset of NVMe commands (as listed in section 1.3) to a device enabled behind Intel VMD. It is intended to enable OEM and ODM factory processes for sending necessary UEFI NVMe commands in the UEFI environment on Skylake SP platforms with Intel VMD hardware and Intel VMD-enabled UEFI drivers, including Intel VROC UEFI driver. This private API is intended for OEMs/ODMs using Intel VROC technology for configuring and deploying their platforms with PCIe NVMe devices in a factory UEFI environment.

This private Intel NVMe UEFI Pass-Thru Protocol is to be used with the understanding that it is up to the OEM/ODM to use this API for writing their own factory UEFI applications. The OEM/ODM must be aware of NVMe commands supported on any given PCIe NVMe SSD. It is also the responsibility of the OEM/ODM to write their UEFI applications using this API with ability to verify the opcode of returned commands.

5.3.4.1 Reference Documents

Document	Description	Document Number/Location
NVMe specification 1.0c, 1.2.1	NVMe protocol and commands	http://www.nvmexpress.org/
Unified Extensible Firmware Interface Specification 2.3.1	UEFI Specification	http://www.uefi.org/



5.3.4.2 Supported Admin Commands

List of supported admin commands

ADMIN_GET_LOG_PAGE	0x02	
ADMIN_IDENTIFY	0x06	
ADMIN_SET_FEATURES	0x09	
ADMIN_GET_FEATURES	0x0A	
ADMIN_FIRMWARE_ACTIVATE	0x10	
ADMIN_FIRMWARE_IMAGE_DOWNLOAD	0x11	
ADMIN_FORMAT_NVM	0x80	Command Set Specific
ADMIN_SECURITY_SEND	0x81	Command Set Specific
ADMIN_SECURITY_RECEIVE	0x82	Command Set Specific
Test Command Write	0xD0	5 Admin Command Set - Vendor specific
Test Command Read	0xD1	5 Admin Command Set - Vendor specific

5.3.4.3 Intel UEFI NVMe Pass-thru Protocol API

5.3.4.3.1 API Limitations

- Supported are PCIe NVMe SSDs connected to Intel® VMD-enabled PCIe ports on the Intel® Xeon® processor Sky Lake product family are supported
 - Pass-through Disk:** Single PCIe NVMe device configured as **Pass-through** (stand-alone) device
 - RAID Arrays:** PCIe NVMe devices configured as members of **RAID Arrays**
- Not supported are PCIe NVMe devices not configured using the Intel® VROC UEFI driver

5.3.4.3.2 NVMe Pass Thru Protocol GUID

```
#define NVM_EXPRESS_PASS_THRU_PROTOCOL_GUID \
{\
    0xec51ef5c, 0x2cf3, 0x4a55, {0xbf, 0x85, 0xb6, 0x3c, 0xa3, 0xb1, 0x3f, 0x44 } \
}
```

5.3.4.3.3 NVMe Pass Thru Protocol Interface Structure

```
struct _NVM_EXPRESS_PASS_THRU_PROTOCOL {
    NVM_EXPRESS_PASS_THRU_MODE          *Mode;
    NVM_EXPRESS_PASS_THRU_PASSTHRU      PassThru;
    NVM_EXPRESS_PASS_THRU_GET_NEXT_NAMESPACE  GetNextNamespace;
    NVM_EXPRESS_PASS_THRU_BUILD_DEVICE_PATH    BuildDevicePath;
    NVM_EXPRESS_PASS_THRU_GET_NAMESPACE        GetNamespace;
```



```
};
```

NVMe Pass Thru Mode Structure

```
typedef struct {  
    UINT32    AdapterId;  
    UINT32    Attributes;  
    UINT32    IoAlign;  
    UINT32    HciVersion;  
    UINT64    Timeout;  
    UINT32    MaxNamespace;  
} NVM_EXPRESS_PASS_THRU_MODE;
```

NVM_EXPRESS_PASS_THRU_PASSTHRU Function Prototype

```
typedef  
EFI_STATUS  
(EFI_API *NVM_EXPRESS_PASS_THRU_PASSTHRU)(  
    IN  NVM_EXPRESS_PASS_THRU_PROTOCOL      *This,  
    IN  UINT32                               NamespaceId,  
    IN  UINT64                               NamespaceUuid,  
    IN OUT NVM_EXPRESS_PASS_THRU_COMMAND_PACKET *Packet,  
    IN  EFI_EVENT                             Event OPTIONAL  
);
```

NVMe Pass Thru function sends an NVM Express Command Packet to an NVM Express controller or namespace. UEFI is single threaded with no interrupts; all I/O for all devices is synchronous and blocking. This command does not check the completion status. **It is caller's responsibility to check the command completion status in NVMe response.**

Parameters	Description
This	A pointer to the NVM_EXPRESS_PASS_THRU_PROTOCOL instance.
NamespaceId	NamespaceId is a 32 bit Namespace ID to which the Express HCI command packet will be sent. A value of 0 denotes the NVM Express controller, a value of all 0FFh in the namespace ID specifies that the command packet should be sent to all valid namespaces.



NamespaceUuid	NamespaceUuid is a 64 bit Namespace UUID to which the Express HCI command packet will be sent. A value of 0 denotes the NVM Express controller, a value of all 0FFh in the namespace UUID specifies that the command packet should be sent to all valid namespaces.
Packet	A pointer to the NVM Express HCI Command Packet to send to the NVMe namespace specified by NamespaceId.
Event	This parameter is currently ignored.

Return Values	Description
EFI_SUCCESS	The NVM Express Command Packet was sent by the host. TransferLength bytes were transferred to, or from DataBuffer.
EFI_BAD_BUFFER_SIZE	The NVM Express Command Packet was not executed. The number of bytes that could be transferred is returned in TransferLength.
EFI_NOT_READY	The NVM Express Command Packet could not be sent because the controller is not ready. The caller may retry again later.
EFI_DEVICE_ERROR	A device error occurred while attempting to send the NVM Express Command Packet.
EFI_INVALID_PARAMETER	Namespace, or the contents of NVM_EXPRESS_PASS_THRU_COMMAND_PACKET are invalid. The NVM Express Command Packet was not sent, so no additional status information is available.
EFI_UNSUPPORTED	If SGL mechanism was used, the NVM Express Command Packet was not sent.
EFI_TIMEOUT	A timeout occurred while waiting for the NVM Express Command Packet to execute.

5.3.4.3.4 NVM_EXPRESS_PASS_THRU_GET_NEXT_NAMESPACE Function Prototype

```
typedef
EFI_STATUS
(EFIAPI *NVM_EXPRESS_PASS_THRU_GET_NEXT_NAMESPACE)(
    IN   NVM_EXPRESS_PASS_THRU_PROTOCOL    *This,
    IN OUT UINT32                          *NamespaceId,
    OUT  UINT64                            *NamespaceUuid OPTIONAL
);
```



The NVM_EXPRESS_PASS_THRU_PROTOCOL.GetNextNamespace() function retrieves the next valid namespace Id on this NVM Express controller. If on input a NamespaceId is specified by all 0xFF in the namespace buffer, then the first namespace defined on the NVM Express controller is returned in NamespaceId, and a status of EFI_SUCCESS is returned.

If on input the value pointed to by NamespaceId is an invalid namespace ID other than 0xFFFFFFFF, then EFI_INVALID_PARAMETER is returned

If NamespaceId is the NamespaceId of the last SSD namespace on the NVM Express controller, then EFI_NOT_FOUND is returned

Parameters	Description
This	A pointer to the NVM_EXPRESS_PASS_THRU_PROTOCOL instance.
NamespaceId	On input, a pointer to a legal NamespaceId for an NVM Express namespace present on the NVM Express controller. On output, a pointer to the next NamespaceId of an NVM Express namespace on an NVM Express controller. An input value of 0xFFFFFFFF retrieves the first NamespaceId for an NVM Express namespace present on an NVM Express controller.
NamespaceUuid	On output, the UUID associated with the next namespace, if a UUID is defined for that NamespaceId, otherwise, zero is returned in this parameter. If the caller does not require a UUID, then a NULL pointer may be passed.

Return Values	Description
EFI_SUCCESS	The NamespaceId of the next Namespace was returned.
EFI_NOT_FOUND	There are no more namespaces defined on this controller.
EFI_INVALID_PARAMETER	NamespaceId is an invalid value other than 0xFFFFFFFF.

5.3.4.3.5 NVM_EXPRESS_PASS_THRU_BUILD_DEVICE_PATH Function Prototype

```
typedef
EFI_STATUS
(EFIAPI *NVM_EXPRESS_PASS_THRU_BUILD_DEVICE_PATH)(
    IN  NVM_EXPRESS_PASS_THRU_PROTOCOL      *This,
    IN  UINT32                               NamespaceId,
    IN  UINT64                               NamespaceUuid,
    IN OUT EFI_DEVICE_PATH_PROTOCOL         **DevicePath
);
```



This function is used to allocate and build a device path node for an NVM Express namespace on an NVM Express controller.

The NVM_EXPRESS_PASS_THRU_PROTOCOL.BuildDevicePath() function allocates and builds a single device path node for the NVM Express namespace specified by NamespaceId.

If the namespace device specified by NamespaceId is not valid, then EFI_NOT_FOUND is returned.

If DevicePath is NULL, then EFI_INVALID_PARAMETER is returned.

If there are not enough resources to allocate the device path node, then EFI_OUT_OF_RESOURCES is returned.

Otherwise, DevicePath is allocated with the boot service AllocatePool(), the contents of DevicePath are initialized to describe the NVM Express namespace specified by NamespaceId, and EFI_SUCCESS is returned.

Parameters	Description
This	A pointer to the NVM_EXPRESS_PASS_THRU_PROTOCOL instance.
NamespaceId	The NVM Express namespace ID for which a device path node is to be allocated and built.
NamespaceUuid	The NVM Express namespace UUID for which a device path node is to be allocated and built.
DevicePath	A pointer to a single device path node that describes the NVM Express namespace specified by NamespaceId. This function is responsible for allocating the buffer DevicePath with the boot service AllocatePool(). It is the caller's responsibility to free DevicePath when the caller is finished with DevicePath.

Return Values	Description
EFI_SUCCESS	The device path node that describes the NVM Express namespace specified by NamespaceId was allocated and returned in DevicePath.
EFI_NOT_FOUND	The NVM Express namespace specified by NamespaceId does not exist on the NVM Express controller.
EFI_INVALID_PARAMETER	DevicePath is NULL.
EFI_OUT_OF_RESOURCES	There are not enough resources to allocate the DevicePath node.

5.3.4.3.6 NVM_EXPRESS_PASS_THRU_GET_NAMESPACE Function Prototype

```
typedef
EFI_STATUS
(EFIAPI *NVM_EXPRESS_PASS_THRU_GET_NAMESPACE)(
    IN  NVM_EXPRESS_PASS_THRU_PROTOCOL      *This,
```



```
IN   EFI_DEVICE_PATH_PROTOCOL    *DevicePath,
OUT  UINT32                      *NamespaceId,
OUT  UINT64                      *NamespaceUuid
);
```

The NVM_EXPRESS_PASS_THRU_PROTOCOL.GetNamespace() function is used to translate a device path node to a Namespace ID and Namespace UUID. This function determines the Namespace ID and Namespace UUID associated with the NVM Express SSD namespace described by DevicePath. If DevicePath is a device path node type that the NVM Express Pass Thru driver supports, then the NVM Express Pass Thru driver will attempt to translate the contents DevicePath into a Namespace ID and UUID. If this translation is successful, then that Namespace ID and UUID are returned in NamespaceId and NamespaceUuid, and EFI_SUCCESS is returned.

Parameters	Description
This	A pointer to the NVM_EXPRESS_PASS_THRU_PROTOCOL instance.
DevicePath	A pointer to the device path node that describes an NVM Express namespace on the NVM Express controller.
NamespaceId	The NVM Express namespace ID contained in the device path node.
NamespaceUuid	The NVM Express namespace contained in the device path node.

Return Values	Description
EFI_SUCCESS	DevicePath was successfully translated to NamespaceId and NamespaceUuid.
EFI_NOT_FOUND	If DevicePath is a device path node type that the Nvm Express Pass Thru driver supports, but there is not a valid translation from DevicePath to a NamespaceId and NamespaceUuid, then EFI_NOT_FOUND is returned.
EFI_INVALID_PARAMETER	If DevicePath, NamespaceId, or NamespaceUuid are NULL, then EFI_INVALID_PARAMETER is returned.
EFI_UNSUPPORTED	If DevicePath is not a device path node type that the NVM Express Pass Thru driver supports, then EFI_UNSUPPORTED is returned.

5.3.4.3.7 Associated Structures

```
typedef struct {
    UINT8          Opcode;
    UINT8          FusedOperation:2;
    #define NORMAL_CMD    0x00
```




```

#define FUSED_FIRST_CMD    0x01
#define FUSED_SECOND_CMD   0x02
UINT8                      Reserved:4;
UINT8                      Psdt:2;
UINT16                     Cid;
} NVME_CDW0;

```

```

typedef struct {
    NVME_CDW0              Cdw0;
    UINT8                  Flags;
    #define CDW10_VALID     0x01
    #define CDW11_VALID     0x02
    #define CDW12_VALID     0x04
    #define CDW13_VALID     0x08
    #define CDW14_VALID     0x10
    #define CDW15_VALID     0x20
    UINT32                 Nsid;
    UINT32                 Cdw10;
    UINT32                 Cdw11;
    UINT32                 Cdw12;
    UINT32                 Cdw13;
    UINT32                 Cdw14;
    UINT32                 Cdw15;
} NVM_EXPRESS_COMMAND;

```

```

typedef struct {
    UINT32                 Cdw0;
    UINT32                 Cdw1;
    UINT32                 Cdw2;
    UINT32                 Cdw3;
} NVM_EXPRESS_RESPONSE;

```

```

typedef struct {

```



```

UINT64      CommandTimeout;
VOID        *TransferBuffer;
UINT32      TransferLength;
VOID        *MetadataBuffer;
UINT32      MetadataLength;
UINT8       QueueId;
NVM_EXPRESS_COMMAND    *NvmeCmd;
NVM_EXPRESS_RESPONSE   *NvmeResponse;
UINT8       ControllerStatus;
} NVM_EXPRESS_PASS_THRU_COMMAND_PACKET;

```

Parameter	Description
CommandTimeout	The timeout in 100 ns units to use for the execution of this NVM Express Command Packet. A Timeout value of 0 means that this function will wait indefinitely for the command to execute. If Timeout is greater than zero, then this function will return EFI_TIMEOUT if the time required to execute the NVM Express command is greater than Timeout.
TransferBuffer	A pointer to the data buffer to transfer between the host and the NVM Express controller for read, write, and bi-directional commands. For all write and non-data commands where TransferLength is 0 this field is optional and may be NULL. If this field is not NULL, then it must be aligned on the boundary specified by the IoAlign field in the EFI_NVM_EXPRESS_PASS_THRU_MODE structure.
TransferLength	On input, the size in bytes of TransferBuffer. On output, the number of bytes transferred to or from the NVM Express controller or namespace.
MetadataBuffer	A pointer to the optional metadata buffer to transfer between the host and the NVM Express controller. For all commands where no metadata is transferred between the host and the controller, this field is optional and may be NULL. If this field is not NULL, then it must be aligned on the boundary specified by the IoAlign field in the EFI_NVM_EXPRESS_PASS_THRU_MODE structure
MetadataLength	On input, the size in bytes of MetadataBuffer. On output, the number of bytes transferred to or from the NVM Express controller or namespace.
QueueId	The type of the queue that the NVMe command should be posted to. A value of 0 indicates it should be posted to the Admin Submission Queue. A value of 1 indicates it should be posted to an I/O Submission Queue.
NvmeCmd	A pointer to an NVM Express Command Packet.
NvmeResponse	The raw NVM Express completion queue entry as defined in the NVM Express Specification.



ControllerStatus	A value that indicates NVMe controller status.
------------------	--

5.3.4.4 Related Definitions

```
//
// If this bit is set, then the NVM_EXPRESS_PASS_THRU_PROTOCOL interface is for directly addressable
// namespaces.
//
#define NVM_EXPRESS_PASS_THRU_ATTRIBUTES_PHYSICAL    0x0001
//
// If this bit is set, then the NVM_EXPRESS_PASS_THRU_PROTOCOL interface is for a single volume logical
// namespace
// comprised of multiple namespaces.
//
#define NVM_EXPRESS_PASS_THRU_ATTRIBUTES_LOGICAL      0x0002
//
// If this bit is set, then the NVM_EXPRESS_PASS_THRU_PROTOCOL interface supports non-blocking I/O.
//
#define NVM_EXPRESS_PASS_THRU_ATTRIBUTES_NONBLOCKIO  0x0004
//
// If this bit is set, then the NVM_EXPRESS_PASS_THRU_PROTOCOL interface supports NVMe command set
// commands.
//
#define NVM_EXPRESS_PASS_THRU_ATTRIBUTES_CMD_SET_NVME 0x0008
//
// QueueId
//
#define NVME_ADMIN_QUEUE                0x00
#define NVME_IO_QUEUE                   0x01
//
// ControllerStatus
//
#define NVM_EXPRESS_STATUS_CONTROLLER_READY    0x00
```



```
#define NVM_EXPRESS_STATUS_CONTROLLER_CMD_ERROR    0x01
#define NVM_EXPRESS_STATUS_CONTROLLER_FATAL        0x02
#define NVM_EXPRESS_STATUS_CONTROLLER_CMD_DATA_ERROR 0x04
#define NVM_EXPRESS_STATUS_CONTROLLER_CMD_ABORT    0x05
#define NVM_EXPRESS_STATUS_CONTROLLER_DEVICE_ERROR 0x06
#define NVM_EXPRESS_STATUS_CONTROLLER_TIMEOUT_COMMAND 0x09
#define NVM_EXPRESS_STATUS_CONTROLLER_INVALID_NAMESPACE 0x0B
#define NVM_EXPRESS_STATUS_CONTROLLER_NOT_READY    0x0C
#define NVM_EXPRESS_STATUS_CONTROLLER_OTHER        0x7F
//
// Status Code - Status Code Type Values
//
#define GENERIC_COMMAND_STATUS          0x00
#define COMMAND_SPECIFIC_STATUS         0x01
#define MEDIA_DATA_INTEGRITY_ERRORS     0x02
#define VENDOR_SPECIFIC                 0x07
//
// Status Code - Generic Command Status Values
//
#define NVME_SC_SUCCESSFUL_COMPLETION    0x00
#define NVME_SC_INVALID_COMMAND_OPCODE  0x01
#define NVME_SC_INVALID_FIELD_IN_COMMAND 0x02
#define NVME_SC_COMMAND_ID_CONFLICT      0x03
#define NVME_SC_DATA_TRANSFER_ERROR      0x04
#define NVME_SC_COMMAND_ABORTED_POWER_LOSS_NOTIF 0x05
#define NVME_SC_INTERNAL_ERROR           0x06
#define NVME_SC_COMMAND_ABORT_REQUESTED  0x07
#define NVME_SC_COMMAND_ABORTED_SQ_DELETION 0x08
#define NVME_SC_ABORTED_FAILED_FUSED_COMMAND 0x09
#define NVME_SC_ABORTED_MISSING_FUSED_COMMAND 0x0A
#define NVME_SC_INVALID_NAMESPACE_OR_FORMAT 0x0B
#define NVME_SC_COMMAND_SEQUENCE_ERROR   0x0C
```



```

#define NVME_SC_INVALID_SGL_SEGMENT_DESCRIPTOR    0x0D
#define NVME_SC_INVALID_NUMBER_SGL_DESCRIPTOR    0x0E
#define NVME_SC_DATA_SGL_LENGTH_INVALID          0x0F
#define NVME_SC_METADATA_SGL_LENGTH_INVALID      0x10
#define NVME_SC_SGL_DESCRIPTOR_TYPE_INVALID      0x11
#define NVME_SC_INVALID_USE_OF_CONTROLLER_MEM_BUF 0x12
#define NVME_SC_PRP_OFFSET_INVALID               0x13
#define NVME_SC_ATOMIC_WRITE_UNIT_EXCEEDED       0x14
//
// Status Code - Generic Command Status Values, NVM Command Set
//
#define NVME_SC_LBA_OUT_OF_RANGE                  0x80
#define NVME_SC_CAPACITY_EXCEEDED                0x81
#define NVME_SC_NAMESPACE_NOT_READY              0x82
#define NVME_SC_RESERVATION_CONFLICT              0x83
#define NVME_SC_FORMAT_IN_PROGRESS               0x84
//
// Status Code - Command Specific Status Values
//
#define NVME_SC_COMPLETION_QUEUE_INVALID          0x00
#define NVME_SC_INVALID_QUEUE_IDENTIFIER          0x01
#define NVME_SC_INVALID_QUEUE_SIZE               0x02
#define NVME_SC_ABORT_COMMAND_LIMIT_EXCEEDED     0x03
#define NVME_SC_ASYNC_EVENT_REQ_LIMIT_EXCEEDED   0x05
#define NVME_SC_INVALID_FIRMWARE_SLOT            0x06
#define NVME_SC_INVALID_FIRMWARE_IMAGE           0x07
#define NVME_SC_INVALID_INTERRUPT_VECTOR         0x08
#define NVME_SC_INVALID_LOG_PAGE                 0x09
#define NVME_SC_INVALID_FORMAT                   0x0A
#define NVME_SC_FW_ACTIVATION_REQUIRES_CONV_RESET 0x0B
#define NVME_SC_INVALID_QUEUE_DELETION           0x0C
#define NVME_SC_FEATURE_IDENTIFIER_NOT_SAVEABLE  0x0D

```



```
#define NVME_SC_FEATURE_NOT_CHANGEABLE      0x0E
#define NVME_SC_FEATURE_NOT_NAMESPACE_SPECIFIC  0x0F
#define NVME_SC_FW_ACTIVATION_REQ_NVM_RESET    0x10
#define NVME_SC_FW_ACTIVATION_REQ_RESET       0x11
#define NVME_SC_FW_ACTIVATION_REQ_MAX_TIME_VIOLATION 0x12
#define NVME_SC_FW_ACTIVATION_PROHIBITED       0x13
#define NVME_SC_OVERLAPPING_RANGE              0x14
#define NVME_SC_NS_INSUFFICIENT_CAPACITY       0x15
#define NVME_SC_NS_IDENTIFIER_UNAVAILABLE      0x16
#define NVME_SC_NS_ALREADY_ATTACHED            0x18
#define NVME_SC_NS_IS_PRIVATE                  0x19
#define NVME_SC_NS_NOT_ATTACHED                0x1A
#define NVME_SC_THIN_PROVISIONING_NOT_SUPPORTED 0x1B
#define NVME_SC_CONTROLLER_LIST_INVALID        0x1C
//
// Status Code - Command Specific Status Values, NVM Command Set
//
#define NVME_SC_CONFLICTING_ATTRIBUTES         0x80
#define NVME_SC_INVALID_PROTECTION_INFO        0x81
#define NVME_SC_ATTEMPTED_WRITE_TO_READ_ONLY  0x82
//
// Status Code - Media and Data Integrity Error Values, NVM Command Set
//
#define NVME_SC_WRITE_FAULT                    0x80
#define NVME_SC_UNRECOVERED_READ_ERROR         0x81
#define NVME_SC_E2E_GUARD_CHECK_ERROR          0x82
#define NVME_SC_E2E_APPL_TAG_CHECK_ERROR       0x83
#define NVME_SC_E2E_REF_TAG_CHECK_ERROR        0x84
#define NVME_SC_COMPARE_FAILURE                0x85
#define NVME_SC_ACCESS_DENIED                  0x86
#define NVME_SC_DEALLOCATE_UNWRITTEN_LB        0x87
```



5.3.4.5 NVMe Pass-Thru Example

The below reference code shows how to use NVMe Pass Thru Protocol to send NVMe identify controller command.

****Note: Command.Flags = 1.** This is a difference between UEFI 2.5 Public protocol that defines Command.Flags as CDW10_VALID (value of 0x04). This private Intel NVMe Pass Thru protocol must define Command.Flags = 0x01.

```
EFI_GUID gEfiNvmExpressPassThruProtocolGuid = NVM_EXPRESS_PASS_THRU_PROTOCOL_GUID;
```

```
NVM_EXPRESS_PASS_THRU_COMMAND_PACKET CommandPacket;
```

```
NVM_EXPRESS_COMMAND Command;
```

```
NVM_EXPRESS_RESPONSE Response;
```

```
EFI_STATUS Status;
```

```
NVM_EXPRESS_PASS_THRU_PROTOCOL *NvmePassthru
```

```
//Initialize NVMe command packet
```

```
Command.Cdw0.Opcode = 0x6; //NVMe ADMIN IDENTIFY OPCODE
```

```
Command.Nsid = 0;
```

```
//
```

```
// Set bit 0 (Cns bit) to 1 to identify a controller
```

```
//
```

```
Command.Cdw10 = 1;
```

```
Command.Flags = 1;
```

```
CommandPacket.NvmeCmd = &Command;
```

```
CommandPacket.NvmeResponse = &Response;
```

```
CommandPacket.TransferBuffer = Buffer;
```

```
CommandPacket.TransferLength = sizeof (NVME_ADMIN_CONTROLLER_DATA);
```

```
CommandPacket.CommandTimeout = EFI_TIMER_PERIOD_SECONDS (5);
```

```
CommandPacket.QueueId = NVME_ADMIN_QUEUE;
```

```
Status = gBS->LocateHandleBuffer (ByProtocol,
                                &gEfiNvmExpressPassThruProtocolGuid,
                                NULL,
                                &NumPassthruHandles,
                                &NvmePassthruHandleBuffer);
```



```
if (!EFI_ERROR(Status)) {
    for (i = 0; i < NumPassthruHandles; ++i) {
        Status = gBS->OpenProtocol (NvmePassthruHandleBuffer[i],
                                    &gEfiNvmExpressPassThruProtocolGuid,
                                    &NvmePassthru,
                                    ImageHandle,
                                    NULL,
                                    EFI_OPEN_PROTOCOL_GET_PROTOCOL);

        Status = NvmePassthru->PassThru (NvmePassthru,
                                         0,
                                         0,
                                         &CommandPacket,
                                         NULL);

        // Parse NVMe response command packet here to obtain additional NVMe status
    } } // Free Allocated buffer here
```

5.3.5 NVMe Deallocate/TRIM

Intel VROC will support this feature for pass-through drives and RAID 0, 1 and 10. This applies to Intel VROC Enabled platforms.

NOTE: This feature is not an end-user visible feature. There is no Intel RSTe application or user interface control to configure the feature..

Support for the NVMe Deallocate/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 life-span of SSDs.



6 Intel RSTe 5.X Common Features

This section will outline and describe those features that are common between the Intel VROC and Intel RSTe packages.

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

6.1 Matrix Storage Manager

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

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 data on each disk.

- Disks

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

The Intel RSTe 5.X Linux product will be able to support the creation and management of RAID 0, 1, 5, 10 Volumes.

6.2 Intel RSTe 5.X PreOS Features and Functionality

This section will address the usage of the different components and outline the system or platforms requirement need to properly support the usage of the Intel® RSTe 5.X family of products.

6.2.1 Intel® RSTe 5.X Unified Extensible Firmware Interface (UEFI) Drivers

Intel® RSTe 5.X products provide UEFI drivers to support UEFI BIOS environment. The UEFI drivers provide an HII interface to support a BIOS level menu-driven configuration tool that is accessed in the BIOS setup. The Intel RSTe 5.X UEFI images included support Intel VROC and SATA/sSATA controllers set to RAID mode.

6.2.2 UEFI System BIOS and the Intel® RSTe 5.X Package



Intel® RSTe 5.X products provides UEFI drivers for OEMs and their BIOS vendors to integrate into their Intel VROC and Intel RSTe enabled platforms (Not required for AHCI mode platforms).

6.2.2.1 Specification References

This document is not intended to be a go-to document for the UEFI specification. The specification is owned by the UEFI working group and detailed information regarding UEFI can be found in documents published by that organization. The Intel® RSTe 5.X UEFI driver implementation conforms to the UEFI specification and is in compliance with version 2.3.1.

Table 3-1: UEFI Specifications and Location

Specification	Location
UEFI Specification version 2.3.1	(http://www.uefi.org/specsandtesttools)
UEFI Platform Initialization Specification version 1.2	(http://www.uefi.org/specsandtesttools)
UEFI Shell Specification version 2.0	(http://www.uefi.org/specsandtesttools)

6.2.2.2 Intel® RSTe 5.X UEFI User Interface

An **HII-compliant** user interface is provided for the pre-boot configuration of the RAID system.

- The HII UI is integrated within the UEFI driver binary provide
- Per the UEFI specification, we publish the HII UI as string and forms packages
- The HII UI is accessible from within the UEFI BIOS (How the user accesses it from within the BIOS is OEM-dependent upon implementation)
- The text string 'Intel VROC Managed VMD' or 'Intel(R) RSTe' will be displayed as the selection to enter the HII UI
- Some OEMs may want to hard assign where the Intel® RSTe UEFI GUI will be located within their BIOS

The Intel® RSTe 5.X UEFI Driver **FORMSET_GUID** is:

`FORMSET_GUID { 0xd37bcd57, 0xaba1, 0x44e6, { 0xa9, 0x2c, 0x89, 0x8b, 0x15, 0x8f, 0x2f, 0x59 } }`
`{D37BCD57-ABA1-44e6-A92C-898B158F2F59}`

6.2.3 UEFI System BIOS Requirements for Platform Compatibility with Intel® RSTe UEFI

This section covers what the OEM/BIOS Vendor is required to accomplish in order to ensure that the platform is compatible the Intel® RSTe UEFI driver.

6.2.3.1 Required Protocols/Functions to be provided by the UEFI System BIOS

The Intel® RSTe 5.X UEFI drivers require the following protocols/functions to be provided by the BIOS:

EFI_BOOT_SERVICES:

- LocateHandleBuffer
- OpenProtocol
- CloseProtocol
- WaitForEvent



- HandleProtocol
- FreePool
- AllocatePages
- AllocatePool
- InstallMultipleProtocolInterfaces
- UninstallMultipleProtocolInterfaces
- Stall
- CopyMem
- LocateProtocol

EFI_RUNTIME_SERVICES:

- SetVariable
- GetVariable
- GetTime

Other Protocols:

- EFI_ACPI_TABLE_PROTOCOL (or EFI_ACPI_SUPPORT_PROTOCOL (EDK117))

6.2.3.2 Optional Protocols/Functions to be provided by the UEFI System BIOS

If the OEM plans to use the Intel® RSTe HII-compliant UI, then **the following protocols/functions are required to be provided by the BIOS:**

- Form Browser 2 Protocol
- Config Routing Protocol
- HII String Protocol
- HII Database Protocol

6.2.3.3 Protocols Provided by the Intel® RSTe 5.X UEFI Drivers

The Intel® RSTe 5.X UEFI drivers provide the following protocols:

- *Driver Binding Protocol*
- *Component Name Protocol (English only)*
- *Component Name 2 Protocol (English only)*
- *Driver Supported EFI Version Protocol*
- *Device Path Protocol*
- *Config Access Protocol*
- EFI_BLOCK_IO_PROTOCOL
 - For Logical Devices
- EFI_STORAGE_SECURITY_PROTOCOL
 - For Non-RAID disks that support TCG Feature Set
- EFI_EXT_SCSI_PASS_THRU_PROTOCOL:
 - All SCSI commands are supported (for ATAPI devices)



6.2.3.4 How-to-Enable the Platform with Intel® RSTe UEFI Driver/HII_GUI

This section covers what the OEM/BIOS Vendor is required to accomplish in order to ensure that the platform is compatible with the Intel® RSTe UEFI driver.

6.2.3.4.1 Step1: Platform UEFI BIOS

1. Ensure that the UEFI System BIOS meets UEFI Specification 2.3.1 compliance
2. The BIOS must provide the following protocols:
 - EFI_Boot_Services Protocols (see section 3.5.3.1)
 - EFI_Runtime_Services Protocols (see section 3.5.3.1)
 - EFI_HII Protocols** (see section 3.5.3.2) *** Required for the Intel® RSTe UEFI UI

6.2.3.4.2 Step2: Download and Integrate the Intel® RSTe 5.X UEFI Package

1. Download the latest kit from the Intel VIP (Validation Internet Portal) website. From the kit select the PreOS zip file which will contain all of the Intel VROC and Intel RSTe PreOS images and tools. Go to the efi_standalone_rste_rs (Intel VROC) or efi_sata and efi_ssata (Intel RSTe) directories to find the UEFI driver binary file.
2. Select and extract the binary file based on the planned integration method:
 - VMDVROC_1.efi and VMDVROC_2.efi – Include these binaries when building the BIOS image to support Intel VROC.
 - SataDriver.efi – Include this binary when building the BIOS image to support Intel RSTe SATA Controller
 - sSATADriver.efi – Include this binary when building the BIOS image to support Intel RSTe sSATA Controller
3. Use the proper integration tools based on the binary file selected above

6.2.3.4.3 Step3: Verify Compliance

There are tools provided to help verify that this process has been completed successfully. Reference [Intel VROC PreOS Components](#) and [Intel RSTe PreOS Components](#) for these files and instructions.

6.2.3.5 Known Compatibility Issues with the UEFI Self Certification Test (UEFI SCT) tool

The following UEFI 2.3.1 SCT tests will appear as FAIL in reports generated using the “Report Generation” tool of the SCT framework. The “Report Generation” tool is the only method that should be used to determine if tests fail. Do not determine test failing test results by viewing the raw log files. The “Report Generation” tool will discard any test results that failed due to an invalid system configuration.



6.2.3.5.1 Bootable Image Support Test\Block IO Protocol Test

EFI_BLOCK_IO_PROTOCOL.Reset - Reset() returns EFI_SUCCESS with ExtendedVerification being TRUE

- *Test Index:* 5.7.5.1.1
- *Test GUID:* 61EE3A34-62A2-4214-B076-5073B177156C
- *Reason:* The Intel® RSTe UEFI driver does not support Reset – EFI_UNSUPPORTED is returned.

EFI_BLOCK_IO_PROTOCOL.Reset - Reset() returns EFI_SUCCESS with ExtendedVerification being FALSE

- *Test Index:* 5.7.5.1.2
- *Test GUID:* 98530F3D-8BD8-44A1-9D06-08039FDFEC63
- *Reason:* The Intel® RSTe UEFI driver does not support Reset – EFI_UNSUPPORTED is returned.

EFI_BLOCK_IO_PROTOCOL.ReadBlocks - ReadBlocks() returns EFI_SUCCESS with valid parameter

- *Test Index:* 5.7.5.2.1
- *Test GUID:* 9EFE26C2-C565-478A-A0B4-05A8FD2E7E3E
- *Reason:* Test called ReadBlocks() with a BufferSize of 0 so EFI_BAD_BUFFER_SIZE is returned. The UEFI 2.3.1 specification states for ReadBlocks, "The size of the Buffer in bytes. This must be a multiple of the intrinsic block size of the device."

6.2.3.5.2 EXT_SCSI_PASSTHRU_PROTOCOL Test

EFI_ATA_PASS_THRU_PROTOCOL.BuildDevicePath - call BuildDevicePath with NULL DevicePath.

- *Test Index:* 5.7.8.2.1

6.2.3.5.3 HII Test\HII Config Access Protocol Test

HII_CONFIG_ACCESS_PROTOCOL.RouteConfig - RouteConfig() returns EFI_NOT_FOUND if no target was

- *Test Index:* 5.18.6.2.3
- *Test GUID:* 1F99EBC8-0253-455F-88AC-9E2BA6DCD729 found with the routing data.
- *Reason:* Intel® RSTe UEFI driver does not support RouteConfig – EFI_UNSUPPORTED is returned. RouteConfig is not supported so that Intel® RSTe HII form values are only modified by the Intel® RSTe driver itself.

HII_CONFIG_ACCESS_PROTOCOL.RouteConfig - RouteConfig() returns EFI_INVALID_PARAMETER with Configuration been NULL

- *Test Index:* 5.18.6.2.1
- *Test GUID:* 495C99F3-0231-45A5-AFFA-D25C6F9A191C
- *Reason:* is caused by not using EFI HII Configuration Access Protocol (see section 31.4 in UEFI 2.4 specification) in RSTe UEFI Driver. The RSTe software does not use it as it is not necessary.

HII_CONFIG_ACCESS_PROTOCOL.RouteConfig- RouteConfig() returns EFI_SUCCESS with valid parameters

- *Test Index:* 5.18.6.2.4
- *Test GUID:* 1A15DF85-6CC1-43F2-9B86-218BD5FDF4A0
- *Reason:* is caused by not using EFI HII Configuration Access Protocol (see section 31.4 in UEFI 2.4 specification) in RSTe UEFI Driver. The RSTe software does not use it as it is not necessary.



6.3 Intel RSTe 5.X Linux Monitoring and Management

6.3.1 Intel RSTe 5.X Linux Command Line Tool

The main Linux tool that is used for RAID management is mdadm. Please see the Linux “man” pages for instructions on how mdadm is used.

NOTE: DMRAID is not supported

6.3.2 Intel VROC ISO installation

There are two ways of install the Intel VROC ISO images. The first is while the supported OS is being installed on the platform. The second is to use the standard Linux process for installing packages from an ISO image to an existing installed supported Linux OS.

6.3.2.1 For installing Intel VROC components during OS installation:

1. Burn the ISO to a CD/DVD or flash to a USB stick using dd.
2. Insert the CD/DVD or USB stick into platform.
3. Start OS installation.
4. Append "inst.updates=LABEL=RSTE" to the installation options.

NOTE: If the platform has a QS Lewisburgh PCH silicon, there is an extra command required to this line

Append "inst.updates=LABEL=RSTE modprobe.blacklist=qat_c62x

5. Proceed with OS installation.

Alternatively you can put the file "updates.img" from the ISO to a HTTP/FTP/NFS server and add an option to the installer: "inst.updates=<url>".

For more information about RHEL installer boot options see:

https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Installation_Guide/chap-anaconda-boot-options.html

<https://rhinstaller.github.io/anaconda/boot-options.html#inst-updates>

The ISO also contains RPM packages in the "rpms" directory. They can be used to install RSTe components on an already installed system.

6.3.2.2 For installing Intel VROC components after OS installation

The provided ISO image must be copied to the platform. After the contents of the ISO image have been extracted. The user can use the standard “yum install” command to install the RPM files.

This can be done by changing the directory to the “RPMS” director and performing a:

- Yum install *.rpm

The user can also install the individual packages as needed.



6.3.3 Monitoring and Logging

6.3.3.1 mdmon

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

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

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

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

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

6.3.3.2 Configuration File for Monitoring

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

- MAILADDR: This configuration entry allows an E-mail address to be used for alerts. Only one email address should be used.
- MAILFROM: This configuration entry sets the email address to appear from the alert emails. The default from would be the "root" user with no domain. This entry overrides the default.
- PROGRAM: This configuration entry sets the program to run when mdmon detects potentially interesting events on any of the 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>
```

In order 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 a mdadm daemon has been started the following messages can be found in /var/log/messages or the corresponding syslog file the distribution has designated:

```
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 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

6.3.4.1 Intel RSTe SGPIO LED Management

Intel RSTe for the PCH supports LED management through SGPIO signaling. LED management is achieved using the Linux `ledmon` & `ledctl` utilities.

6.3.4.2 Intel VROC LED Management

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



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

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 monitors 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 RSTe 5.X Linux product will provide LED management support for Status LEDs on enclosures that provides either a 2 LED solution or a 3 LED solution.

Ledmon can be run with the following options listed below:

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

- 1) Installing ledmon/ledctl utilities: Execute the following commands to install ledmon

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

- 2) Use ledmod/ledctl utilities

- a) Running ledctl and ledmon concurrently, ledmon will eventually override the ledctl settings. Start the IPMI.

```
# systemctl start ipmi
# ledmon
```

- b) The 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 through the process for the user to follow on a standard supported Linux distribution as a clean installation.

- i. 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
```

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

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

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

```
#!/bin/bash
```

```
# THIS FILE IS ADDED FOR COMPATIBILITY PURPOSES
```

```
#
```

```
# It is highly advisable to create own system services or udev rules
```

```
# to run scripts during boot instead of using this file.
```

```
#
```

```
# In contrast to previous versions due to parallel execution during boot
```

```
# this script will be executed during boot.
```

```
#
```

```
# Please note that you must run 'chmod +x /etc/rc.local' to ensure
```

```
# that this script will be executed during boot.
```

```
#
```

```
touch /var/lock/subsys/local
```

```
ledmon
```



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

Note: Save the file and quit/exit the editor.

6.3.5 Volume Management

6.3.5.1 Volume Creation and Deletion

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

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

1. It is recommended that the user always starts with new drives. To minimize issues with using drives with existing data on them it is recommended that the superblocks be cleared before starting:

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

2. First, 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 4. 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.

3. 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))
```

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



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

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

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
```

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 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 RSTe container.

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

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

Intel RSTe 5.X Linux will prevent OLCE on a Volume if the container it resides in contains another volume of the following levels:

- RAID 1
- RAID 10

Before performing any grow operation (if not already done), one "ENV" variable needs to be set (the variable is case sensitive)

```
# export MDADM_EXPERIMENTAL=1
```

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

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

The example below adds a single drive to the RAID container and then grows the volume(s). Because Intel RSTe 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.

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: Please consult the user's platform documentation for instructions on how to enter into the Intel VROC HII interface.

The following assumptions have been made:

1. It is known how to enter into the appropriate platform BIOS level setup menus



2. The Intel VMD functionality has been enabled
3. The appropriate Intel VROC RAID Upgrade Key has been installed
4. The appropriate number of NVMe SSDs have been plugged into the enabled Intel VMD controller

Step 1: Enter into 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.

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

Step 6: This step is optional, and applies to RAIDs that have volumes connected to more than one 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 <Δ> 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 8 for each drive required in this RAID.

Step 9: Unless the user has selected RAID 1, select the strip size by using the <Δ> 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>.

Step 12: The user will be presented with this screen:

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

The user's RAID configuration has now been saved.

Step 14: To save and reboot to begin 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.

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

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. Mdr RAID 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. The following command adds a spare disk to the designated container.

```
mdadm -a /dev/md0 /dev/sde
```

Note: Visual guide and instructions for configuration of Intel VROC BIOS for RAID with Hot Spare can be located in section 12.1.1.1.

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 RAID5 volume with IMSM metadata where md0 is the IMSM container and md127 is the RAID 5 volume. The RAID 5 volume contains drives `/dev/nvme0n1`, `/dev/nvme1n1`, `/dev/nvme2n1`, `/dev/nvme3n1`

```
Personalities : [raid0] [raid1] [raid5] [raid10]
md127 : active 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 the following command:



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

or from the container by the following command:

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

6.3.6.1.6 RAID Volume Recovery

Intel RSTe 5.X Linux will allow RAID volume recovery in case of failure. The volume will rebuild on the hot spare disk added before removal of broken disk without any problem. File system has to be accessible during this operation.

Intel RSTe 5.X Linux will also allow the first RAID volume in a container to recover from failure condition. The volume will rebuild on the spare disk added before removal of broken disk without any problem. File system has to be accessible during this operation.

Intel RSTe 5.X Linux will support the ability to track volume rebuild progress and resume this operation after accidental break (e.g. after unexpected system reboot).

6.3.6.1.7 RAID Auto Rebuild

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

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

Intel RSTe 5.X Linux will require that a spare disk used for auto-rebuild 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 RSTe 5.X Linux will trigger an auto-rebuild of all volumes in a degraded container.

Intel RSTe 5.X Linux will enable container auto-rebuild spare sharing by default without the need for configuration.

Intel RSTe 5.X Linux will not automatically trigger a rebuild of a volume if another volume in the container does not require a rebuild

Intel RSTe 5.X 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. Mdmmon rebuilds containers only if a failed disk is removed from container.

Intel RSTe 5.X Linux will support volume auto-rebuild to spare drive defined in an container being rebuilt. This operation will start just after MD reports a failure of any drive establishing RAID volume.

Intel RSTe 5.X 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 MD raid array can be used for rebuild if a drive contains metadata consistent with the degraded array.



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

1. Create container "md1" with 3 disks:
mdadm -C /dev/md1 -e imsm -n3 /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1
2. Create RAID1 volume "Volume1" in container "md1", drive /dev/nvme2n1 remains a spare:
mdadm -C /dev/md/Volume1 -l1 -n2 /dev/nvme0n1 /dev/nvme1n1

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>
3. Save configuration file:
mdadm -Es > /etc/mdadm.conf
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 similar below:

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

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

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

ARRAY /dev/md/Volume1 container=267445e7:458c89eb:bd5176ce:c37281b7 member=0
UUID=25025077:fb9cfab:e4ad212d:3e5fce11

POLICY domain=DOMAIN path=* metadata=imsms action=spare-same-slot
5. Make sure mdadm is in monitor mode:
mdadm --monitor --scan --daemonise
6. Fail one of the drives in volume "Volume1", the volume without a spare:
mdadm --fail /dev/md/Volume1 /dev/nvme1n1

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

```
[raid1]
: active raid1 nvme2n1[2] nvme0n1[1]
md1/0 [2/1] [U_]
[===>.....] recovery = X% (updated so-far/total) finish=0.0mins speed=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

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.

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

Intel RSTe 5.X Linux will perform partition/filesystem size verification before begin of 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 RSTe 5.X Linux will support strip size modification performed along with level migration. The following strip sizes are supported 4 kB, 8 kB, 16 kB, 32 kB, 64 kB, 128 kB.

Intel RSTe 5.X Linux will not automatically start a rebuild on a RAID volume within a container that has a RAID volume being migrated.

Intel RSTe 5.X 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 1: Migration Capabilities with IMSM

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



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

Before performing any grow operation (if not already done), one "ENV" variable needs to be set

```
# export MDADM_EXPERIMENTAL=1
```

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>
```

First step is to 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>
```

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

```
# mdadm -G /dev/md0 -n 2
```

```
# cat /proc/mdstat
Personalities : [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>
```

Add a spare disk to the container:

```
# mdadm -a /dev/md0 /dev/sdc
# 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)
      3315 blocks super external:imsm
```

```
unused devices: <none>
```

Migrate from RAID 0 to RAID 5:

```
# mdadm -G /dev/md127 -l 5 --layout=left-asymmetric
# cat /proc/mdstat
Personalities : [raid0] [raid1] [raid5] [raid10]
md127 : active raid5 nvme2n1[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>
```

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

6.3.6.1.9 Freezing Reshape

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

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

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

```
--freeze-reshape
```

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

```
Personalities : [raid5]
md127 : active raid5 nvme2n1[2] nvme1n1[1] nvme0n1[0]
      10485760 blocks super external:-md0/0 level 5, 128k chunk, algorithm 0 [3/3] [UUU]
[>.....] reshape = 2.2% (116736/5242880) finish=501934.9min speed=0K/sec
```

```
md0 : inactive - nvme2n1[2] (S) nvme1n1[1] (S) nvme0n1[0] (S)
      9459 blocks super external:imsm
```

```
unused devices: <none>
```

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



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

or with a volume:

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

6.3.6.1.10 RAID Volume Initialization

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

6.3.6.1.11 Maximum Number of Drives in a RAID Volume

Intel RSTe 5.X Linux will provide support for the maximum number of drives in the following configurations:

- RAID 0 2-15 SATA drives
2-24 NVMe drives
- RAID 1 2 drives
- RAID 5 3-15 SATA drives
3-24 NVMe drives
- RAID10 4 drives

6.3.7 Intel RAID Write Hole Closure

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

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

RAID Write Hole (RWH) is a potential condition when using RAID5 that during the time that a RAID strips is being written, a power loss is encountered. With HWRAID and battery backed DRAM, logging could be used to recover. With Intel RSTe 5.0 and RWH, a journaling SSD can be added to preserve the Partial Parity and reduce the potential data loss issue

6.3.7.1 RAID Write Hole Closure (RWH) Linux

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

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

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



6.3.7.2 RWH PreOS

The RSTe 5.X UEFI drivers will support the able recover 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 RWH Backwards compatibility

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

6.3.7.4 RWH Recovery

The Intel RSTe 5.X 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 RSTe 5.X 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 RSTe 5.X 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 RSTe 5.X 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 Close RWH for VMD-attached NVMe RAID 5 volume without using any additional drive

The Intel RSTe 5.X RWH closure algorithm for RAID 5 volume on NVMe (VMD-attached) will be able to close the RWH without the use of any additional drive (PPL distributed RWH closure mode).

6.3.7.6 Close RWH for SATA RAID 5 volume without using any additional drive

The Intel RSTe 5.X RWH closure algorithm for RAID 5 volume on SATA drives (HDD or SSD) will be able to close the RWH without the use of any additional drive (PPL distributed RWH closure mode).

6.3.7.7 Ability to switch between RWH closure modes for NVMe

The Intel RSTe 5.X product will provide the user with the ability to switch between the following RWH closure modes: PPL distributed mode and Off mode for RAID 5 volumes on NVMe drives.

6.3.7.8 Runtime switching between RWH closure modes

The Intel RSTe 5.X product will provide the user with the ability to switch between the RWH closure modes during the normal OS operation mode.

6.3.7.9 Turning on and off the RWH closure

The Intel RSTe 5.X product will provide the user with the ability to turn off the RWH closure mechanism.



6.3.7.10 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.

6.3.7.11 Ability to switch between RWH closure modes for SATA

The Intel RSTe 5.X 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.

6.3.7.12 Disable on-device cache checkbox in drive properties for NVMe

If a RAID 5 volume has the RWH closure enabled, the Intel RSTe 5.X driver will block the ability to enable on-device cache in the drive properties in Disk Manager for all RAID member drives and Journaling Drive if applicable for RAID volumes on NVMe drives.

6.3.7.13 On device cache for RAID array

If a RAID 5 volume has the RWH closure enabled, the Intel RSTe 5.X CLI will block the ability to enable on-device cache for all devices in a given RAID array.

6.3.7.14 RSTe UEFI for VMD RWH recovery

Intel RSTe 5.X 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.

6.3.7.15 For RWH example:

To create a RAID5 volume with enabled RWH policy:

It is recommended to clear out metadata

```
# mdadm -C /dev/md/ims0 -e imsm -n4 /dev/nvme[0-3]n1
# mdadm -C /dev/md/vol0 -l5 -n4 /dev/nvme[0-3]n1 --rwh-policy=ppl
```

To check the current RWH policy:

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

To disable RWH policy for a running array:

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

To enable RWH policy for a running array:

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



6.3.8 Intel RSTe Disk Management

6.3.8.1 Device Presentation

Intel RSTe 5.X Linux will support the ability to list all of the devices attached to the Intel VMD controller (when enabled) as well as the devices attached to the SATA and sSATA controller.

6.3.8.2 4K Native Sector Sizes Drives

Intel RSTe 5.X 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.

NOTE: This feature is only supported in UEFI mode.

6.3.8.3 Reporting and Notifications

Intel RSTe 5.X Linux will report the status of RAID volumes and containers to the user through the command line

Intel RSTe 5.X Linux trigger an event and/or send an e-mail in case of a situation required to be reported.

Intel RSTe 5.X 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 RSTe 5.X 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 RSTe 5.X 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 RSTe 5.X 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 RSTe 5.X Linux will notify the user to appropriately modify size of partitions existing on the source volume/disk 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 RSTe 5.X 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 is required to properly expand all volumes defined on this array).

6.3.8.4 Hot Plug (Surprise and Managed)

Intel® RSTe 5.X will support the ability to Hot Plug (remove and replace) disk drives on properly configured controllers whether or not I/O is being processed. 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 Solid State drives (SATA or NVMe) drives 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 VROC functionality. This will also include references to the required BIOS settings to ensure that the BIOS is properly configured. Please consult the user's platform design documentation for additional information.

6.3.8.4.1 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.

6.3.8.4.2 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 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. 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. Please consult the user's platform documentation because each BIOS is different and the steps taken to enable this feature may be different.

6.3.8.4.3 Enabling Surprise Hot Plug for PCH

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. Please 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 Hot Spare Disk

Intel RSTe 5.X 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. This support is provided in the Intel RSTe GUI as well as in the Pre-OS images.

6.3.8.5.1 Rebuild on Hot Insert

The Intel RSTe 5.X 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
- if the newly inserted disk is healthy (no recorded SMART event)
- the newly inserted disk is a pass through disk
- the newly inserted disk is a member of a failed volume
- newly inserted disk is a member of a degraded volume

The Intel RSTe 5.X GUI shall give the user the ability to enable and disable rebuild on hot insert.

6.3.8.5.2 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 three ways.

1. The Intel ASM Tools
2. The Intel® RSTe provides Pre-OS environment (UEFI HII) user interface support.
3. The mdadm tool tools

6.3.8.5.3 Using the OS Level User Interface Utility

Step 1: Bring up the Intel ASM tool (See Intel ASM user manual for additional instructions)

Step 2: In the “Devices” pane, select the drive that is to be marked as a spare drive.

Step 3: Navigate to the Disk Properties pane and select “Mark as spare”

Step 4: The warning window will appear to inform the user that all of data on the drive will be deleted. Click on “Yes”

Step 5: Under “Disk Properties” for the drive, the “Usage” is now set to “Spare”.

6.3.8.5.4 Return Hot Spare Back to Normal

Intel RSTe management tools will provide a mechanism for setting a spare back to a pass through disk under Disk Properties.

6.3.8.5.5 Returning a Spare to a Normal Disk in the BIOS

Depending if the RAID volume created by Intel VROC or by Intel, 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® RSTe sSATA Controller or Intel® RSTe 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>. 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.

6.3.8.6 Mark a Disk as Spare in HII

The RSTe 5.X 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® RSTe sSATA Controller or Intel® RSTe 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.

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>.

6.3.8.7 Read Patrol

Intel RSTe 5.X product will provide support for Read Patrol, which checks the RAID volumes for errors that could result in a failure. The checks are done on demand by the user and will verify all sectors of all RAID volumes on SATA and sSATA controllers as well as volumes behind Intel VMD. If an issue is discovered an attempt at corrective action is taken. Read Patrol can be enabled or disabled manually.

NOTE: Read Patrol does not extend to drives marked as Hot Spares.

The RSTe 5.X driver shall execute the Read patrol functionality on RAID volumes that are in a normal state (not degraded rebuilding or migrating). For a redundant volume if the RSTe Driver detects a bad block error the RSTe driver shall attempt to recover from that error by reassigning the bad block and writing the recovered data to the reassigned location. Read patrol will be run on all volumes simultaneously.

For each power-cycle the RSTe 5.X driver will record the event but not store all errors across all RAID volumes being scrubbed. The RSTe 5.X driver will display the following read patrol error information: unrecovered LBAs and its corresponding RAID Volume and physical disk.



The Intel RSTe 5.X management tools will display an option to enable or disable the Read Patrol feature and the feature will be “disabled” by default.

6.3.8.8 Check Pointing

Intel RSTe 5.X 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 RSTe 5.X will be able to track volume migration progress and resume this operation after accidental break (e.g. after unexpected system reboot).

Intel RSTe 5.X will be able to support rebuild and migration check pointing for volumes established on drives.

6.3.8.9 Bad Block Management

Intel RSTe 5.X 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 mark 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 PreOS as well as in the Linux environment.

6.4 MDRAID Sysfs Components

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/

Example:

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

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

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

md127 is imsm0 and md126 is raid1.



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

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

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

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

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

The MD Volume contents look like below:

```
# ls -l /sys/devices/virtual/block/md126/md/
total 0
-rw-r--r-- 1 root root 4096 May 17 13:26 array_size
-rw-r--r-- 1 root root 4096 May 17 13:26 array_state
drwxr-xr-x 2 root root  0 May 18 13:10 bitmap
```



```
--w----- 1 root root 4096 May 18 13:10 bitmap_set_bits
-rw-r--r-- 1 root root 4096 May 17 13:26 chunk_size
-rw-r--r-- 1 root root 4096 May 17 13:26 component_size
-r--r--r-- 1 root root 4096 May 17 13:26 degraded
drwxr-xr-x 2 root root  0 May 17 13:26 dev-nvme1n1
drwxr-xr-x 2 root root  0 May 17 13:26 dev-nvme2n1
-rw-r--r-- 1 root root 4096 May 17 13:26 layout
-rw-r--r-- 1 root root 4096 May 17 13:26 level
-rw-r--r-- 1 root root 4096 May 18 13:10 max_read_errors
-rw-r--r-- 1 root root 4096 May 17 13:26 metadata_version
-r--r--r-- 1 root root 4096 May 18 13:10 mismatch_cnt
--w----- 1 root root 4096 May 17 13:26 new_dev
-rw-r--r-- 1 root root 4096 May 17 13:26 raid_disks
lrwxrwxrwx 1 root root  0 May 17 13:26 rd0 -> dev-nvme1n1
lrwxrwxrwx 1 root root  0 May 17 13:26 rd1 -> dev-nvme2n1
-rw-r--r-- 1 root root 4096 May 18 13:10 reshape_position
-rw-r--r-- 1 root root 4096 May 17 13:26 resync_start
-rw-r--r-- 1 root root 4096 May 17 13:26 safe_mode_delay
-rw-r--r-- 1 root root 4096 May 18 13:10 suspend_hi
-rw-r--r-- 1 root root 4096 May 18 13:10 suspend_lo
-rw-r--r-- 1 root root 4096 May 17 13:26 sync_action
-r--r--r-- 1 root root 4096 May 17 13:26 sync_completed
-rw-r--r-- 1 root root 4096 May 18 13:10 sync_force_parallel
-rw-r--r-- 1 root root 4096 May 18 13:10 sync_max
-rw-r--r-- 1 root root 4096 May 18 13:10 sync_min
-r--r--r-- 1 root root 4096 May 17 13:26 sync_speed
-rw-r--r-- 1 root root 4096 May 18 13:10 sync_speed_max
-rw-r--r-- 1 root root 4096 May 18 13:10 sync_speed_min
```

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

- Array size

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



1048576

- Array state
cat /sys/devices/virtual/block/md126/md/array_state
clean
- Raid level
cat /sys/devices/virtual/block/md126/md/level
raid1
- Strip size
cat /sys/devices/virtual/block/md126/md/chunk_size
65536
- Metadata
cat /sys/devices/virtual/block/md126/md/metadata_version
external:/md127/0

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

```
# cat /proc/mdstat
Personalities : [raid1]
md127 : active raid1 nvme0n1[1] nvme1n1[0]
      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>
```



7 Intel® RSTe Key Features

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

Name	Key Features	
RAID	• Bad Block Management**	• SGPIO • Partial Parity Logging (PPL)**

**Items that are supported on both Intel RSTe PCH/SATA and Intel VROC

7.1 Intel RSTe PreOS Features and Functionality

7.1.1 Intel® RSTe RAID Legacy Option ROMs

Intel® RSTe will provide Legacy RAID OROM images that support standard Int13 BIOS environments. These images provide a BIOS level menu-driven configuration tool for managing RSTe RAID volumes outside of an OS. There are individual images for each of the PCH controllers on the Intel® C620 and C230 series chipset platforms:

- There is a single RSTe RAID Legacy Option ROM image that will support the AHCI/SATA controller for each of the mentioned chipsets.
- For the Intel® C610/C620 series chipsets there is also a RAID Legacy Option ROM for the sSATA controller.
- The RSTe Legacy OROM will support 48bit LBAs for pre-boot platform IO. This ensures that the RSTe OROM is capable of providing native OROM boot support for 2TB disks and RAID volumes. It is up to the platform BIOS to provide a means of sending IO that utilizes the full LBA address space.

7.1.1.1 Using the Intel® Rapid Storage Technology enterprise Legacy Option ROM User Interface

Step 1: Upon re-boot, the user will see the option ROM status message on the screen – press CTRL-I to enter the Intel Rapid Storage Technology enterprise option ROM user interface.

Step 2: In the Main Menu, select option #1 'Create RAID Volume'. Enter the name the user wants to use for the RAID volume, then press Enter.

Step 3: Select the RAID Volume Name and the RAID Level

Step 4: Press Enter to select the disks to be used by the array that the volume will be created on. The key to select the disks will be shown at the bottom of the screen, usually this is the <Space> bar. Press Enter when done

Step 5: Select the strip size (128 KB is the default for RAID 5) by using the arrow keys, then press Enter when done.

Step 6: Enter the size for the RAID volume in gigabytes. The default value will be the maximum size. If the user specifies a smaller size, the user will be able to create a second volume in the remaining space using the same procedure. When done select arrow down to "Create Volume" and press enter to complete.

Step 7: After this is done, exit the Option ROM user interface.



7.1.1.2 Legacy OROM Reset Disks to Non-RAID

When the 'Reset Disks to Non-RAID' option is selected in the Intel RSTe Legacy OROM, the RAID metadata will be removed from the disk, returning it back to a pass-through drive.

7.1.1.3 Legacy OROM Rebuild Messaging

If the Intel RSTe Legacy OROM detects that one or more RAID volumes are in the 'Rebuilding' state then a message will be displayed stating that the RAID Volume is in a rebuild state.

7.1.1.4 Legacy OROM Banner when locked HDs are detected

The Intel RSTe Legacy OROM will present a banner to the user if a locked HDD is detected in the system when it is powering on or if the system is resuming from a hibernate (S4).

7.1.1.5 Legacy OROM boot from password protected disk

When a system disk contains password protection the Intel RSTe Legacy OROM will only boot from the disk if unlocked.

7.1.1.6 Legacy OROM boot from password protected volume

When a system volume contains member disks with password protection the Intel RSTe Legacy OROM will only boot from the RAID Volume after all of the member disks are unlocked. If any member disk is password protected, the system may not boot properly.

7.1.1.7 Legacy OROM RAID Volume limit

The Intel RSTe Legacy OROM environment supports a maximum of 4 RAID volumes on the SATA/sSATA controller

7.1.1.8 Legacy OROM Dirty Shutdown Recovery

The Intel RSTe Legacy OROM for SATA and sSATA will be able to recover from the RAID 5 volume invalid state caused by Dirty Shutdown condition if all RAID5 volume disk members are enumerated during system boot.

7.1.1.9 Boot Device Name

The Intel RSTe Legacy Option ROM will provide the name of any bootable device recognized by the Option ROM to the system BIOS

7.1.1.10 Migration Support

The Intel RSTe Legacy Option ROM will support I/O access to a RAID volume that has a migration in progress.

7.1.1.11 Using the BCFS to Differentiate Platform SKUs

Intel® RSTe PreOS has support for BIOS Control Feature Set (BCFS) to enable OEMs the opportunity to customize the Intel® RSTe feature offerings. OEMs can enable/disable the desired features per platform SKU directly in their BIOS code. By clearing or setting the corresponding bits of the '**Intel RSTe Feature Capabilities**' register in the Intel chipset's SATA controller MMIO space, OEMs now have greater flexibility in determining what Intel® RSTe features will be supported per platform model/SKU.



The following sections explain the use of each of the bits in the BCFS, also known as the Software Feature Mask bits.



BCFS bit number	Bit meaning	Values		Additional info
0	Enable/disable Raid0	Raid type disabled	0x0	If you disable all raid levels – all BCFS settings will be set back to default and OROM UI delay will be set to 2 seconds
		Raid type enabled	0x1	
1	Enable/disable Raid1	Raid type disabled	0x0	
		Raid type enabled	0x1	
2	Enable/disable Raid10	Raid type disabled	0x0	
		Raid type enabled	0x1	
3	Enable/disable Raid5	Raid type disabled	0x0	
		Raid type enabled	0x1	
4	RESERVED			
5	Enable/disable UI	Feature disabled	0x0	If you set it to 0 (disable) bits 10-14 are ignored
		Feature enabled	0x1	
6	Enable/disable unlock of HDD in OS	OS can unlock HDDs OS cannot unlock HDDs	0x1 0x0	
7	Enable/disable LED SGPIO	Feature disabled Feature enabled	0x0 0x1	
8	eSATA RAID usage	Allow all volume types to span internal and external ports	0x0	
		Allow only R1 volume type to span internal and external ports. Other volume types – allow all eSATA-internal disks only if there are enough of them to create the volume	0x1	
9	reserved			
10 - 12	Delay on UI splash screen	2 seconds	0x000	Default setting is 0x000: 2 seconds.
		4 seconds	0x001	
		6 seconds	0x010	



		8 seconds	0x011	
		10 seconds	0x100	
		15 seconds	0x101	
		30 seconds	0x110	
		60 seconds	0x111	
13 - 14	Mode of showing UI	Show if error or >=2 disks	0x00	Default setting 0x00: show if there are 2 or more disks connected or error occurred
		Show only if error	0x01	
		Never show UI	0x10	
		Show always	0x11	
15	RESERVED			

Note: This document does not cover details on how to setup a system BIOS. For that level of information please contact the user's platform's BIOS vendor or the user's Intel field representative to put the user in contact with the appropriate Intel BIOS support personnel.

7.1.1.11.1 Configuring the Platform's RAID Related Features

When the BIOS has set the SATA Controller's mode to RAID, the following bits of the 'Intel RSTe Feature Capabilities' register in the Intel chipset's SATA controller MMIO space will determine what RAID levels will be supported on the platform SKU:

Note: Clearing all RAID level related bits to '0' is an unsupported configuration. The Intel® RSTe Legacy OROM will ignore the BIOS settings and enable all RAID levels.

7.1.1.12 BCFS Bit Setting

7.1.1.12.1 Example Configuration

To configure a platform SKU that offers **only** RAID levels 0 and 10, the bits must be configured as follows:

Bit 0 == 1 (default)

Bit 1 == 0

Bit 2 == 1 (default)

Bit 3 == 0



7.1.1.12.2 Configuring the Behavior of the OROM UI and Banner

There are three (3) bits that control the behavior of the Intel® RSTe Legacy OROM UI and the Banner Splash Screen that are displayed during POST at system boot-up. Use the following bit configurations to determine whether or not the splash screen will be displayed during post and if so, how long the delay will be before the system continues the boot process:

Bits	Type	Reset/ Default	Description
<u>11:10</u>	<u>RWO</u>	<u>0h</u>	<p>OROM UI Normal Delay (OUD): Values of these two bits specify the delay of the OROM UI Splash Screen in a normal status.</p> <p>00 – 2 secs (default and previous value)</p> <p>01 – 4 secs</p> <p>10 – 6 secs</p> <p>11 – 8 secs</p> <p>Note: If bit 5 == 0, then these values are disregarded</p> <p>Comment: Allow OEM to lengthen normal timeout of OROM splash screen so user has more time to hit CTRL+I on keyboard.</p>
<u>5</u>	<u>RWO</u>	<u>1h</u>	<p>Intel RSTe OROM UI (RSTOROMUI): If set to '1' then the OROM UI is shown. Otherwise, no OROM banner or information will be displayed if all disks and RAID volumes are Normal.</p>

7.1.1.12.3 Example Configuration

To configure a platform SKU that enables the OROM Banner Splash Screen to be displayed for 6 seconds, the bits must be configured as follows:

Bit 5 == 1 (default)

Bit '10' == 0 (default)

Bit '11' == 1

7.1.1.12.4 Configuring Intel® RSTe UI Capabilities

There are a few capabilities within the Intel® RSTe UI that is controlled by the BCFS bits.

HDD unlock and Delay on UI splash screen are controlled with the following bits:

Bits	Type	Reset/ Default	Description
----------------------	----------------------	------------------------------------	-----------------------------



<u>6</u>	<u>RWO</u>	<u>0h</u>	HDD Unlock (HDDLK): If set to '1', then HDD password unlock is enabled in the OS.
<u>10:12</u>	<u>RW</u>	<u>0x000</u>	Delay on UI splash screen: 2 seconds 0x000 4 seconds 0x001 6 seconds 0x010 8 seconds 0x011 10 seconds 0x100 15 seconds 0x101 30 seconds 0x110 60 seconds 0x111

7.1.1.12.5 Example Configuration

<u>Bits</u>	<u>Type</u>	<u>Reset</u>	<u>Description</u>
15:14	RO	0h	Reserved.
13:12	RWO	0h	Reserved
<u>11:10</u>	<u>RWO</u>	<u>0h</u>	OROM UI Normal Delay (OUD): Values of these bits specify the delay of the OROM UI Splash Screen in a normal status. 00 – 2 secs (default and previous value) 01 – 4 secs 10 – 6 secs 11 – 8 secs If bit 5 == 0, then these values are disregarded <u>Comment: Allow OEM to lengthen normal timeout of OROM splash screen so user has more time to hit CTRL+I on keyboard.</u>
<u>9</u>	<u>RWO</u>	<u>0h</u>	Reserved
<u>8</u>	<u>RWO</u>	<u>0h</u>	eSATA RAID usage: If set to '1', allow only R1 volume type to span internal and external ports. Other volume types – allow all esata-internal disks only if there are enough of them to create the volume If cleared to '0', then allow all volume types to span internal and external ports.



7	<u>RWO</u>	<u>0h</u>	Enable/disable LED SGPIO: If set to "1", then LED SGPIO support is enabled.
6	<u>RWO</u>	<u>0h</u>	Enable/disable unlock of HDD in OS: If set to "1", then unlock of HDD in OS is enabled.
5	<u>RWO</u>	<u>1h</u>	Intel® RSTe OROM UI (RSTOROMUI): If set to '1' then the OROM UI is shown. Otherwise, no OROM banner or information will be displayed if all disks and RAID volumes are Normal.
4	<u>RWO</u>	<u>0h</u>	Reserved
3	<u>RWO</u>	<u>1h</u>	RAID 5 Enable (R5E): If set to '1', then RAID5 is enabled
2	<u>RWO</u>	<u>1h</u>	RAID 10 Enable (R10E): If set to '1', then RAID10 is enabled
1	<u>RWO</u>	<u>1h</u>	RAID 1 Enable (R1E): If set to '1', then RAID1 is enabled
0	<u>RWO</u>	<u>1h</u>	RAID 0 Enable (R0E): If set to '1', then RAID0 is enabled

To configure a platform SKU to not allow unlocking passwords from the Intel® RSTe UI and to allow the UI to activate the disk/port LEDs, the bits must be configured as follows:

Bit 6 == 0 (default)

7.1.1.13 RSTe UEFI Dirty Shutdown Recovery

The Intel RSTe UEFI drive will be able to recover from the RAID 5 volume invalid state caused by Dirty Shutdown condition occurrence if all RAID 5 disk members are enumerated during system boot.

7.1.2 Determining the Version of the Intel® RSTe UEFI Driver

There are three ways to determine the version of the Intel® RSTe UEFI driver(s) integrated into the system BIOS. Use the following procedure to determine the version.

7.1.2.1 Using the UEFI Shell

When the platform BIOS is configured to boot from the UEFI Intel® RSTe UEFI, to obtain the driver version or to verify that the UEFI driver is loaded just enter into the BIOS setup or press the hot key to enter into the Boot Option menu. Boot into a UEFI Shell environment.

Shell:>Drivers

The Intel® RSTe UEFI driver will be shown along with version, where xx.x.x.xxxx will be replaced with the actual UEFI OROM Version e.g.:

"CD 0000000B B - - 1 2 Intel® RSTe xx.x.x.xxxx SATA Driver"



7.1.2.2 Intel RSTe Device Information Display in UEFI

The Intel® RSTe product release package includes the support for the RSTe UEFI driver to report the physical port for devices connected to the SATA/sATA controllers. In some of the releases of the Intel® RSTe UEFI driver, the device information provided (i.e. to the UEFI shell environment) was based off of the enumeration values created during the discovery of the devices attached. To support a manufacturing environment that relies of this information to identify the physical port the device is connected too, the UEFI driver now reports out the physical port value instead of the enumerated value. The data is displayed as a set of 3 values in the following order: X-Y-Z

X – 0: passthru disk, 1: volume

Y– PHY number: 1 (phy0), 2 (phy1), 4 (phy3), 8 (phy3), 16 (phy4), 32 (phy6), 64 (phy7)

Z – disk number on PHY (in case of an expander)

7.1.2.3 Intel RSTe UEFI

This example shows how to configure a bootable RAID volume using PCH with SATA drives for UEFI. This procedure should only be used for a newly-built system or reinstall of the operating system.

Step 1: To enter the setup and configuration mode of the BIOS, press the key indicated for entering setup and select boot options when the following menu appears:

Step 2: Navigate to Intel® RSTe SATA Controller, and press <Enter>.

Step 3: Highlight Create RAID Volume, and press <Enter>.

Step 4: Highlight the Volume Name field. If the user wishes to change this value from the default, press <Enter>. Modify the name to the preferred name, and press <Enter> to save the new name.

Step 6: Highlight the RAID Level. Default setting is RAID0. To change this to another RAID configuration, press <Enter> and select desired RAID level, then press <Enter> to save the user's selection.

Step 7: To select the drives that will be part of the RAID, highlight the < > next to the desired drive and press <Enter>. A small menu will pop up with the setting currently at a value of blank, indicating not enabled. To include this drive within the RAID, highlight the X and press <Enter> to enable this drive as part of the array.

Step 8: Repeat Step 7 with each drive needed until all desired drives have been included within the RAID.

Step 9: Unless the user is setting up a RAID1, the user will be allowed to adjust the Strip Size for the configuration. RAID1 will be set at 128 KB, and cannot be adjusted. To modify this setting, highlight the present value, press <Enter>, and select the preferred value. Press <Enter> to save the new strip size.

Step 10: The capacity of a RAID Volume will initially display the maximum value allowed. This can be modified to a smaller value to create additional RAID volumes. If the user wishes to alter this value, select the Capacity and press <Enter>. Enter the desired value up to the maximum and press enter to save. The value will not change if it exceeds allowed capacity.

Step 11: To finalize creation of the RAID volume, highlight Create Volume, and press <Enter>.

Step 12: The user will see the user's created volume on the following screen. To save the configuration, press <Esc> and the user will be presented with the following message. "Changes have not been saved. Save Changes and exit? Press 'Y' to save and exit, 'N' to discard and exit, 'ESC' to cancel." Press <y> to save and exit.

Note: This step is extremely important, failure to save at this point will cause the user's RAID to fail to be saved. The user must save in order for the RAID in order for this process to properly complete.



Step 13: After saving the user's RAID volume, the system will require a reboot if before the RAID volume may be used for a boot or data volume. The user may finish other configuration items now, but if the user is ready to proceed with OS installation, this is the time to perform a reboot.

Step 14: Now that system boot has been completed, the user may format the RAID volume and begin system installation.

7.2 Intel RSTe Miscellaneous Features and Functionality

7.2.1 SGPIO on SATA Controller (RAID Mode Only)

Intel RSTe product will support enclosure management, compliant to **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.

7.2.2 TRIM Command (RAID 0, 1 and 10)

NOTE: This feature is not an end-user visible feature. There is no Intel® RSTe application or user interface control to configure the feature. Registry settings are provided for OEM use.

Support for the TRIM command allows the OS to pass information to the Solid State Disk (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 life-span of SSDs.



8 Unsupported Features

This section will outline some (by no means all) of the unsupported features.

- Intel RSTe 5.X Linux product will not support creating a single disk RAID volume



Appendix A: Related Documentation

Relevant Specifications

UEFI Specifications 2.4 (http://uefi.org)
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 1.0 Specification (http://www.serialata.org)
SATA II Specification (http://www.serialata.org)
SATA 3 (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)

Relevant Documentation

CDI / IBL#	Title/Location
Reference Documents	
441979	Intel® 6 Series Chipset/ Intel® C600 series chipset Platform Controller Hub (PCH) BIOS Specification Update – NDA
30051	RS – Intel® 6 Series Chipset/ Intel® C600 series chipset Platform Controller Hub (PCH) BIOS Spec <i>Contact the user's Intel FAE to get access to this document</i>
Kit 33272	Intel® Server Platform Services Alpha SPS_02.01.01.009.0 NOTE: This package is the Intel® Server Platform Services Manageability Engine Firmware for Intel® C600 series chipset Product Line - Alpha Full Release and contains key tools such as FITc and fpt for the Intel® C600 series chipset This document can be downloaded from ARMS/VIP
558771	Skylake Server Processor External Design Specification (EDS)
546955	Skylake Platform Controller Hub (SKL PCH-H) External Design Specification (EDS)
547817	Lewisburg Platform Controller Hub External Design Specification
566449	KabyLake Platform Controller Hub H External Design Specification V2_Rev2_0



Appendix B: External Hardware Capability

Enterprise SATA Drives

The following table identifies the current list of SATA drives used in validation and is subject to change without notice. Contact the user's factory representative for questions on any specific hardware item.

Vendor	Family	Model Name/Number
Fujitsu	A160 (2.5") 7200 RPM FDE Option Extended Duty	MHZ2080BK
Hitachi	Ultrastar A7k1000 (3.5") 7.2rpm	
Seagate	Barracuda 7200.10 Serial ATA	
Seagate	Barracuda 7200.11 Serial ATA	
Seagate	Barracuda ES	
Western Digital		WD1002FAEX
Western Digital		WD6000HLHX

NVMe Drives

The following table identifies the non-Intel NVMe SSD used in validation and is subject to change without notice. Contact the user's factory representative for questions on any specific hardware item.

Vendor	Model Name/Number
Samsung	SM951
Samsung	SM961
Samsung	PM961
Samsung	PM953
Samsung	PM963
Toshiba	XG3
Hitachi	Ultrastar SN100
Hitachi	Ultrastar SN200



Expanders and Enclosures

The following table identifies the NVMe expander and switches used in validation and is subject to change without notice. Contact the user's factory representative for questions on any specific hardware item.

Vendor	Model Name/Number
AIC	XJ1100
Xyratex	RS1603X
Supermicro	CSE-M28x
Promise	Vtrak E-Class E310
	Vtrak J-class
Supermicro	SC836E1-R800V



Appendix C: Intel C620 Series Chipset Legacy OROM Boot Option

INT 15 / AX=F300h / BX=0002h (Get RSTe OROM Boot Info)

Description:

Through this function, BIOS provides user-settable RSTe boot information to the RSTe legacy OROM driver. These values are visible to the user through the BIOS Setup menus. The menu options should be linked to legacy OROM selections in the PCH-IO section.

Inputs:

AX = F300h (Function)
BX = 0002h (Sub Function)
EAX = 0000F300h
EBX = 4F450002h ('OE') + Sub Function
EDX = 424F4F54h ('BOOT')

Normal Outputs:

CF = clear if successful
EAX = 424F4F54h ('BOOT')
BL = legacy_orm_boot_controller_selection:

Due to limited shadow RAM and EBDA space, and the fact that a platform may require multiple OROMs be loaded for other functions, there might not be enough runtime space for both the RSTe SATA RAID controller OROM and the RSTe sSATA RAID controller OROM to provide int13h support simultaneously. Even so, each RSTe OROM still needs to initialize so that it can configure each controller based on platform dependencies and store data needed by the OS drivers in the Shadow RAM area even if it does not provide full int13h runtime support. Thus, through this setup option BIOS can avoid the runtime space conflict by allowing the user to select the boot controller according to the following values:

0000h = No runtime space restrictions. BIOS indicates that both RSTe SATA and sSATA runtime code should provide full int13h support for RSTe devices.

(NOTE: The BIOS should allow this option if it knows that there is room in shadow RAM for both OROMs' runtime code. If the BIOS can always guarantee this condition, then it does NOT need to make Legacy OROM boot controller selection visible to the user in BIOS setup.)

0001h = The sSATA controller is selected as boot controller. BIOS will load RSTe SATA OROM first, but the SATA OROM will only initialize and then leave pertinent RAID configuration information for the SATA OS RAID Driver in runtime space. The RSTe sSATA OROM will initialize, relocate to runtime space, and provide full int13h support for sSATA attached devices.



0002h = The SATA controller is selected as boot controller. BIOS will load RSTe sSATA OROM first, but the sSATA OROM will only initialize and then leave pertinent RAID configuration and sSATA OEM parameter information for the sSATA OS RAID Driver in runtime space. The RSTe SATA OROM will initialize, relocate to runtime space, and provide full int13h support for SATA controller attached devices.

0003h = Neither the SATA nor sSATA controller is selected as boot controller. Boot support is being provided by another device. BIOS will load both RSTe OROMs, but each will only initialize and leave pertinent RAID configuration and sSATA OEM parameter information for the RSTe OS RAID Drivers in runtime space. There will NOT be int13h support for RSTe devices.

Error Outputs:

CF = set on error

AH = error code

= 86h Function Not Supported = (boot_controller_selection = 0000h assumed)



Appendix D: RSTe SATA Port Bitmap Implementation

Legacy OROM

RSTe overloaded offset 0x08 of the PNP header, with a bitmap of which ports make up the PNP header.

E.g., a RAID5 volume whose member disks located on SATA ports 0,2,3 would have a value of 0x0D (0000_1101b) and a pass-thru disk on port 4 would have a value of 0x10 (0001_0000b).

Offset	Size	Value	Description
00h	1	'\$'	Signature byte 1
01h	1	'P'	Signature byte 2
02h	1	'N'	Signature byte 3
03h	1	'P'	Signature byte 4
04h	1	01h	Structure revision
05h	1	Varies	Length (in 16 byte blocks)
06h	2	Varies	Offset of next header (0000h if none)
08h	1	00h	Reserved
09h	1	Varies	Checksum
0Ah	4	Varies	Device identifier
0Eh	2	Varies	Offset to manufacturer string (optional)
10h	2	Varies	Offset to product name string (optional)
12h	3	Varies	Device type code
15h	1	Varies	Device indicators
16h	2	Varies	Boot Connection Vector (BCV)
18h	2	Varies	Disconnect Vector (DV)
1Ah	2	Varies	Bootstrap Entry Vector (BEV)
1Ch	2	0000h	Reserved
1Eh	2	Varies	Static resource information vector



UEFI Driver

The RSTe UEFI driver, in an effort to provide similar functionality as in the legacy OROM, has implemented the Port Number value in the Device Path as a bitmap representing the physical disk connections that the Logical Disk represents. The LSB (least significant bit) represents port 0 and increases linearly. E.g. a single pass through disk on SATA port 3 (assuming the SATA ports are enumerated 0 – X) is represented by 0000_1000b (or 0x08).

EFI_DEVICE_PATH_PROTOCOL

For each logical disk that is exposed by the SATA RAID UEFI Driver, an EFI_DEVICE_PATH_PROTOCOL shall be created.

The Device Path Protocol for each logical disk shall be appended to the PCI SATA Controller Device Path.

The fields of the EFI_DEVICE_PATH_PROTOCOL shall be filled out differently depending on whether the device is an ODD or an HDD.

EFI_DEVICE_PATH_PROTOCOL Field	ATAPI (ODD)	HDD/Volume – Logical Device
Type	3 (Messaging Device Path)	3 (Messaging Device Path)
Sub-Type	18 (SATA)	18 (SATA)
Length	10	10
HBA Port Number	Port ID bitmap (bit #n set if device is on port #n)	Port ID bitmap (bit #n set if logical device contains device ID #n) Lowest Significant Bit (LSB) represents port 0.
Port Multiplier Port Number	0x8000 (directly connected)	0x8000 (directly connected)
Logical Unit Number	0	0 for passthrough devices, myVolRaidDevNum for RAID volumes, which is the n th volume created on the array.