Intel® Rack Scale Design (Intel® RSD) Pooled System Management Engine (PSME) User Guide Software v2.5

User Guide Software v2.5

July 2019

Revision 001

Document Number: 613314-001

Contents

Tables

Revision History

Revision	Description	Date
001	Initial release for Intel® RSD PSME User Guide Software v2.5	June 2019

§

1.0 Overview

The interfaces specified in the Intel® Rack Scale Design (Intel® RSD) Pooled System Management Engine (PSME) User Guide are based on the *Distributed Management Task Force (DMTF) Redfish* Interface Specification* and schema (refer to Table 4).

1.1 Scope

This document is full and detailed documentation of the Pooled System Management Engine (PSME) software v2.5. The information below covers minimal requirements for hardware and software during compilation and runtime. This document contains instructions to compile, install, deploy, and configure the PSME software on various supported system environments.

The following topics are covered in this documentation:

- PSME software overview
- Hardware requirements and software prerequisites
- PSME software installation and deployment
- Hardware and PSME software configuration

1.2 Intended Audience

- Software vendors (ISVs) of the Pod Manager (PODM) software that make use of Intel® RSD PSME Application Program Interface (API) to discover, compose, and manage Intel® RSD Architecture drawers, regardless of the hardware vendor, and/or manage Intel® RSD drawers in a multi-vendor environment.
- Hardware vendors (OxMs) of PSME firmware for different hardware platforms other than Bulldog Creek Intel® Software Development Vehicle that would like to provide the Intel® RSD PSME API on top of their systems.

1.3 Introduction

The PSME software is a bundle of applications working and communicating with each other to manage and control specific assets in a rack. The PSME software v2.5 consists of:

- **PSME REST server:** HTTP server with Representational State Transfer (REST) API and JavaScript* Object Notation (JSON*) data containers responsible for gathering and presenting information about assets and available operations on these assets. This application communicates with agents through JSON-Remote Procedure Call (RPC) as a transport and the Generic Asset Management Interface (GAMI) as a payload protocol.
- The PSME REST server connects with the following agents:
  - **PSME Compute agent:** responsible for gathering detailed information about compute modules and for controlling hosts. The agent participates in the node assemble procedure.
  - **PSME Network agent:** responsible for configuration and gathering detailed information about the network topology. It also manages the data network top of rack (ToR) switch.
  - **PSME Chassis agent:** responsible for gathering detailed information about the Control Plane Processor (CPP). The agent communicates with the Rack Management Module (RMM).
  - **PSME Field Programmable Gate Array (FPGA)-over Fabrics (FPGA-oF) agent** - responsible for managing FPGA attached to host using FPGA over Fabrics technology.

- **PSME FPGA-oF discovery agent** - responsible for responding to queries about available FPGA from FPGA-oF initiators.
- **PSME FPGA-oF discovery agent** - responsible for responding to queries about available FPGA from FPGA-oF initiators.
- **PSME Pooled Node Controller (PNC) agent:** responsible for gathering detailed information about the Peripheral Connect Interface express* (PCIe*) storage switch and attaching devices (i.e., NVMe* drives) to compute hosts.
- **PSME Internet Small Computer Systems Interface (iSCSI) Storage agent:** responsible for preparing, configuring, gathering, and connection of storage logical volume management (LVM) and target framework `(tgt)` daemon. This agent connects to the PSME Storage Service.
- **PSME RMM agent:** Rack Management Module (RMM) is responsible for managing and gathering detailed information about rack and its power/thermal metrics.
- **PSME GUID Partition Table (GPT) NVMe agent and PSME Storage Performance Development Kit (SPDK) NVMe agent -** responsible for managing and gathering detailed information about NVMe volumes attached to hosts through RDMA NICs using NVMe-oF technology.
- **PSME NVMe discovery agent -** responsible for responding to queries about available NVMe volumes from NVMe-oF initiators.

1.4 Supported System Environments

Table 1. Source Compilation

Component	Ubuntu* v16.04	CentOS* 7
PSME REST Server	+	+
PSME Compute for Intel® Software Development Vehicle	+	-
PSME Network	-	+
PSME Storage for LVM and tgt daemon	+	-
PSME Chassis for Intel® Software Development Vehicle	+	-
PSME PNC for Intel® Software Development Vehicle	+	-
PSME RMM for Intel® Software Development Vehicle	+	-
PSME GPT NVMe for Intel® Software Development Vehicle	+	-
PSME SPDK NVMe for Intel® Software Development Vehicle	+	
PSME NVMe Discovery for Intel® Software Development Vehicle	+	-
PSME FPGA-oF for Intel® Software Development Vehicle	+	-
PSME FPGA-oF Discovery for Intel® Software Development Vehicle	+	-

Refer to Table 2 if you receive pre-built binaries from Intel.

Table 2. Binaries Working

Component	Ubuntu* v16.04	Arista* Extensible Operating System*
PSME REST Server	+	+
PSME Compute for Intel® Software Development Vehicle	+	-
PSME Network	-	+
PSME Storage for LVM and tgt daemon	+	-
PSME Chassis for Intel® Software Development Vehicle	+	-
PSME PNC for Intel® Software Development Vehicle	+	-
PSME RMM for Intel® Software Development Vehicle	+	-
PSME NVMe for Intel® Software Development Vehicle	+	-
PSME GPT NVMe for Intel® Software Development Vehicle	+	-

Component	Ubuntu* v16.04	Arista* Extensible Operating System*
PSME SPDK NVMe for Intel® Software Development Vehicle	+	-
PSME NVMe Discovery for Intel® Software Development Vehicle	+	-
PSME FPGA-oF for Intel® Software Development Vehicle	+	-
PSME FPGA-oF Discovery for Intel® Software Development Vehicle	+	-

The PSME software is designed and developed to support generic hardware and various operating system (OS) solutions. Some steps in the development, configuration, and deployment process can vary for different system environments. Each step, therefore, is described for generic instances with the exception of some detailed instruction for the specific supported system, described next.

Supported hardware:

- Intel® Software Development Vehicle platform

Supported OS:

- Ubuntu* v16.04 Long Term Support (LTS)
- Arista Extensible Operating System (EOS)

The PSME Software should compile and run on every Linux system if required libraries are available and at the proper version for the specific OS.

Throughout this document, all processes will be described for generic hardware and environment with some references to specific cases for supported systems.

1.4.1 Supported Hardware

- Software Development Vehicle platform
- Supported OSs:
  - Ubuntu v16.04 LTS
  - Arista EOS

Note: The PSME software should compile and run on every Linux* system if required libraries are available and at the proper version for the specific OS.

Note: Throughout this document, all processes are described for generic hardware and environment with some references to specific cases for supported systems.

1.5 Terminology

Table 3. Terminology

Term	Definition
AFU	Accelerator Functional Unit
API	Application Program Interface
ACL	Access Control List
Arista EOS	Arista Extensible Operating System
ASCII	American Standard Code for Information Interchange
BIOS	Basic Input/Output System
Blade	Server Board that equates to the SPMF: `ComputerSystem`
BMC	Baseboard Management Controller
CA	Certificate Authority
CEE	Converged Enhanced Ethernet

Term	Definition
CM	Control Module
CLI	Command line interface
CPP	Control Plane Processor
DCBX	Data Center Bridging Capability Exchange
DHCP	Dynamic Host Configuration Protocol
DIMM	Dual In-line memory module
ETS	Enhanced Transmission Selection
FPGA	Field Programmable Gate Array
FRU	Field Replaceable Unit
GAMI	Generic Asset Management Interface
GPT	GUID Partition Table
GRUB	GRand Unified Bootloader
GUID	Globally Unique Identifier
HTTPS	Hypertext Transfer Protocol Secure
I2C	I^{2}C, Inter-Integrated Circuit, synchronous multi master/slave serial computer bus
IEEE	Institute of Electrical and Electronics Engineers
Intel® NUC	Next Unit of Computing, miniature PC
Intel® SDV	Intel® Software Development Vehicle
Intel® RSD	Intel® Rack Scale Design
IPMI	Intelligent Platform Management Interface
IPMB	Intelligent Platform Management Bus
iSCSI	Internet Small Computer Systems Interface
ISV	Independent Software Vendor
JBOD	Just a bunch of disks
LLDP	Link Layer Discovery Protocol
LTS	Long Term Support
LVM	Logical Volume Management
MAC	Media Access Control
MDR	Managed Data Region
Module	The physical component housing a blade or switch
NIC	Network Interface Controller
NTP	Network Timer Protocol
NVM	Non-Volatile Memory
NVMe*	Non-Volatile Memory Express*
NVMe-oF*	NVMe-over Fabrics*
OPAE	Open Programmable Acceleration Engine
OOB	Out-of-band
OS	Operating System
OxM	Original Equipment Manufacturer or Original Design Manufacturer
PCI	Peripheral Connect Interface
PCIe*	Peripheral Connect Interface express*
PEM	Privacy Enhanced Mail
PFC	Priority Flow Control
PNC	Pooled Node Controller
POD	A physical collection of multiple racks
PODM	POD Manager
PSME	Pooled System Management Engine

Term	Definition
PXE	Pre-boot eXecution Environment
QoS	Quality of Service
QSFP	Quad Small Form-factor Pluggable
RDMA	Remote Direct Memory Access
REST	Representational state transfer
RMM	Rack Management Module
RPC	Remote Procedure Call
SDK	Software Development Kit
SMBIOS	System Management BIOS
SPDK	Storage Performance Development Kit
SSDP	Simple Service Discovery Protocol
Symlink	Symbolic link
tgt	Linux daemon for management of iSCSI targets
TLV	Type Length Value
TLS	Transport Layer Security
ToR	Top of the Rack network switch
USB	Universal Serial Bus
UUID	Universally unique identifier
VLAN	Virtual LAN
XML	Extensible Markup Language

1.6 References

Table 4. Reference Documents and Resources

Doc ID	Title	Location
613315	*Intel® Rack Scale Design (Intel® RSD) Conformance and Software Reference Kit Getting Started Guide v2.5*	Note: *https://www.intel.com/content/www/us/en/architecture-and-technology/rack-scale-design/rack-scale-design-resources.html*
613316	*Intel® Rack Scale Design (Intel® RSD) POD Manager (PODM) Release Notes Software v2.5*	
613317	*Intel® Rack Scale Design (Intel® RSD) POD Manager (PODM) User Guide Software v2.5*	
613318	*Intel® Rack Scale Design (Intel® RSD) Pooled System Management (PSME) Release Notes Software v2.5*	
613319	*Intel® Rack Scale Design (Intel® RSD) Architecture Specification Software v2.5*	
613320	*Intel® Rack Scale Design (Intel® RSD) Pod Manager (PODM) Representational State Transfer (REST) API Specification Software v2.5*	
613321	*Intel® Rack Scale Design (Intel® RSD) Rack Management Module (RMM) Representatinal State Transfer (REST) API Specification Software v2.5*	
613324	*Intel® Rack Scale Design (Intel® RSD) Generic Assets Management Interface (GAMI) API Specification v2.5*	
613325	*Intel® Rack Scale Design (Intel® RSD) Pooled System Management Engine (PSME) REST API Specification Software v2.5*	
613329	*Intel® Rack Scale Design Storage Services API Specification Software v2.5*	
N/A	*Intel® Rack Scale Design (Intel® RSD) Conformance Test Suite (CTS) Release Notes*	See Note
596167	*Intel® Rack Scale Design (Intel® RSD) for Cascade Lake Platform Firmware Extension Specification*	*https://cdrdv2.intel.com/v1/dl/getContent/596167*

Doc ID	Title	Location
608298	Field Programmable Gate Array (FPGA) over Fabric Protocol Architecture Specification - rev1.0	https://cdrdv2.intel.com/v1/dl/getContent/608298
DOC-2786	*Default ToS to skprio mapping on Linux**	https://community.mellanox.com/docs/DOC-2786
DSP8010	*Redfish Schema v2018.3*	https://www.dmtf.org/sites/default/files/standards/documents/DSP8010_2018.3.zip
ISO 8601	*Date and time format - ISO 8601*	https://www.iso.org/iso-8601-date-and-time-format.html
N/A	*Swordfish Scalable Storage Management API Specification v1.0.7a*	https://www.snia.org/sites/default/files/SMI/swordfish/v107a/Swordfish_v1.0.7a_Specification.pdf
N/A	*Hypertext Transfer Protocol - HTTP/1.1* *Obsoletes IETF 2145, 2616, and Updates IETF 2817, 2818*	https://tools.ietf.org/html/rfc7230 See RFC 7230-7235.
N/A	*NVM Express over Fabrics Revision v1.0*	http://nvmexpress.org/wp-content/uploads/NVMe_over_Fabrics_1_0_Gold_20160605-1.pdf
N/A	*IEEE Std 802.1Q - 2014*	https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6991462
N/A	*802.1Qbb – Priority-based Flow Control specification*	https://1.ieee802.org/dcb/802-1qbb/
N/A	*802.1Qaz – Enhanced Transmission Selection*	https://1.ieee802.org/dcb/802-1qaz/
N/A	*Aardvark* Interface Library (requires registration)*	https://www.totalphase.com/products/aardvark-software-api/aardvark-api-linux-x86_64-v5.30.zip
N/A	*The GNU Privacy Handbook*	https://www.gnupg.org/gph/en/manual/x56.html
N/A	*Open Programmable Acceleration Engine (OPAE) Driver*	https://github.com/OPAE/opae-sdk/releases/download/1.1.0-2/opae-intel-fpga-driver-1.1.0-2.x86_64.deb
RFC2119	*Key Words for Use in RFCs to Indicate Requirement Levels, March 1997*	https://ietf.org/rfc/rfc2119.txt
N/A	*Intelligent Platform Management Interface Specification*	https://www.intel.com/content/www/us/en/servers/ipmi/ipmi-second-gen-interface-spec-v2-rev1-1.html

Note: Documents referenced in this table which have a Doc ID, but cannot be accessed, can be obtained by calling 1-800-548-4725 or by visiting *www.intel.com/design/literature.htm* obtain a copy.

1.7 Conventions

The key words/phrases "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in *Key Words for Use in RFCs to Indicate Requirement Levels, March 1997*, RFC 2119, refer to Table 4.

1.8 Notes and Symbol Convention

Symbol and note conventions are similar to typographical conventions used in the Cloud Infrastructure Management Interface (CIMI) specification, refer to Table 4. The notation used in JSON* serialization description:

- Values in italics indicate data types instead of literal values.
- Characters are appended to items to indicate cardinality:

  - ? (0 or 1)
  - * (0 or more)
  - + (1 or more)
- Vertical bars, |, denote choice. For example, a|b means a choice between a and b.
- Parentheses, ( ), indicate the scope of the operators ?, *, +, and |.
- Ellipses ("...") indicate points of extensibility. The lack of an ellipsis does not mean no extensibility point exists; rather, it's just not explicitly called out.

§

2.0 Key Features

This section explains some of the key features of the Intel® RSD PSME software. Use of the features requires a correct setup configuration. Instructions are provided in Sections 4.0, Intel® RSD Rack Network Configuration and 5.0, Intel® RSD Drawer Configuration of this user guide.

Note: The PSME applications require exclusive access to managed services. This means that the state of services under PSME management should not be modified by other controllers (managers, command line, APIs, and so forth).

2.1 Out-of-band Discovery

One of the key features of the PSME Compute software is gathering information about hardware from System Management BIOS (SMBIOS) and Advanced Configuration and Power Interface (ACPI) and exposing it through REST API.

The `psme-compute` needs to be provided with correct addresses and credentials to the Baseboard Management Controllers (BMCs) of the managed platforms in the configuration file stored in the `/etc/psme` directory. The credentials should be encrypted using the encrypted binary. The binary can be compiled by following the instructions in Section 3.0, PSME Development Environment. To use the binary, create a protected directory `/etc/psme`.

```
    $sudo mkdir -p /etc/psme
    $sudo chmod 644 /etc/psme
```

Then, use the encrypt binary to encrypt the credentials to the BMCs that the PSME Compute is to communicate with.

```
    $sudo encrypt <BMC_username>
    $sudo encrypt <BMC_password>
```

Place the encrypted credentials in the PSME Compute service configuration section for a particular BMC:

```
{
    "ipv4" : "<bmc_IP_address>",
    "username" : "<encrypted_username>",
    "password" : "<encrypted_password>"
}
```

Upon the start of the `psme-compute` service, compute sleds' BMCs are queried for the information stored in the Managed Data Regions (MDR) as well as the power state and telemetry readings of the platform.

Note: The MDR SMBIOS and ACPI regions are filled by the BIOS during boot time, and it may take up a few minutes for the operation to complete.

In Intel® RSD 2.5, the PSME supports platforms based on the Second Generation Intel® Xeon® Scalable Processors with BIOS and BMCs that implement the Intel® Rack Scale Design (Intel® RSD) for Second Generation Intel® Xeon® Scalable Processors (formerly known as Cascade Lake) Platform Firmware Extension Specification, refer to Table 4. For these platforms, the PSME Compute is currently able to provide information about the following resources exposed by SMBIOS and ACPI:

- processors
- memory DIMMs
- Intel® Optane™ DC Persistent Memory Modules
- network interfaces
- local storage devices
- BIOS version

- Trusted Platform Modules (TPM)
- Field Replaceable Unit (FRU) information of the system and its chassis
- Intel® Speed Select configurations available on the system
- discrete FPGAs

2.2 Speed Select

Some models of the Second Generation Intel® Xeon® Scalable Processors support multiple configurations in which some processor cores are disabled or slowed down to speed up the remaining cores. In Intel® Rack Scale Design v2.5, PSME Compute is able to discover the available configurations. It also provides the administrator with an option to change the current configuration by sending a `PATCH` request to a Computer System resource representing a sled containing such processors.

Note: The `PATCH` request triggers a reboot of the Computer System because the requested configuration can only be applied at boot time.

2.3 Hot Swap

Hardware asset removal is automatically reflected on the REST API. To detect device Hot Swap, the PSME performs periodic hardware monitoring. Therefore, it may take some time (usually up to tens of seconds) before the PSME discovers the hardware change. In some cases (e.g., NVMe drives on the PNC) this time may be longer as the PSME depends on the OS or other software/hardware components. Due to this, if a Hot Plug is quickly followed by a Hot Unplug, then the PSME may not be able to detect the device at all.

2.3.1 Hard Drive Hot Swap

Once the user removes a hard drive from the Just a bunch of disks (JBOD) the following events are triggered:

- Removing a Drive from a Chassis representing such a hard drive (remove event)
- Removing a Drive from a Storage Pool to which it belonged and setting the Storage Pool health to critical (update event)
- Setting Volumes health to critical (update event for every affected Volume placed in the Storage Pool)
- Setting the health of Endpoints pointing to the Volumes to critical (update event for every affected Endpoint)
- Setting the health of the Zones which contain the Endpoints to critical (update event for every affected Zone)

Hard drive reinsertion does not cause the asset critical state to revert to the previous state.

2.3.2 Sled Hot Swap

In the case of sled removal, only a single event is triggered, but all related resources will disappear from the API.

There is a hardware Basic Input/Output System (BIOS) limitation regarding the discovery of information about sled assets. Since enumeration of resources happens at boot time, no changes will be detected after a hardware change (e.g., reinsertion of a Dual In-line memory module (DIMM) into a different slot) without a reboot.

To work around this, the Sled must be rebooted so that full basic discovery succeeds.

After the Sled removal and reinsertion, a power on action should be performed. When an OS boot is completed, the Sled can be powered off. This action is necessary to update BMC data about assets coming from BIOS.

2.3.3 Eventing Limitations

Events are triggered and sent to subscribers in the aforementioned cases. However, the PSME does not send update events for every resource change. If several resources are removed or added in one processing, a single event may be sent - only for the highest-level resource. Finally, there are no events for creation, updates, or deletion of Tasks or Subscriptions.

2.4 Data Network Top-of-rack Switch Management

This section describes the Intel® Rack Scale Design PSME Network service.

2.4.1 Description

The PSME Network agent is responsible for gathering information about the data top of the Rack network switch (ToRS) and managing its ports and Virtual LANs (VLANs). It's features include setting port attributes, creation, deletion and modification of VLANs and management of Quality of Service (QoS) settings. It enables the PODM to discover the topology of the rack by collecting neighbor Media Access Control (MAC) addresses.

2.4.2 Prerequisites

The following sections describe the prerequisites for running the PSME network software.

2.4.2.1 Initial switch configuration

As a prerequisite to using the PSME Network, the initial configuration of the data switch must be performed. The required steps are described in Appendix A, Top-of-Rack Switch Configuration.

2.4.2.2 Dependencies

The PSME Network software depends on several libraries, which must be installed in the OS. Since the package manager provides outdated versions of these libraries, more recent versions are compiled alongside PSME software and must be manually copied to the target OS. The following steps are recommended:

1. Compile the PSME Network software according to the instructions in Section 3.0, PSME Development Environment.
2. `Create /opt/psme/lib` directory in the target OS:
```
    mkdir -p /opt/psme/lib
```
3. The build directory (for example `build.release.gcc.32bit`) contains a `lib` directory with the required libraries. Copy the files matching the following patterns to the `/opt/psme/lib` directory in the target OS.
    - `libcurl.so`
    - `libgcrypt.so`
    - `libgnutls.so`
    - `libgpg-error.so`
    - `libhogweed.so`
    - `libmicrohttpd.so`
    - `libnettle.so`
    - `libgmp.so`

    Using a tool that will preserve the symbolic link (`symlink`) structure of the copied libraries is recommended.

2.4.2.3 Sysdb Profile

A configuration file for EOS Software Development Kit (SDK) can be found in the PSME source tarball as `agent/network/arista-sysdb-profile`. The file should be copied to the target OS to `/usr/lib/SysdbMountProfiles/psmenet`.

2.4.2.4 Executable

The `psme-network` binary created by the compilation should be copied to the target OS under the name `“psmenet”` to match the profile.

2.4.2.5 Daemon Registration

The psmenet binary should be set up as an EOS daemon from Command Line Interface (CLI) configuration mode:

```
    daemon psmenet
    exec /opt/psme/bin/psmenet
    no shutdown
    exit
    write
```

After the write command, the network agent will be started after each EOS reboot - if installed as an extension.

2.4.2.6 Service Configuration

A default PSME Network configuration file can be found in the PSME source tarball in `agent/network/configuration.json`. The default settings for QoS setting on the switch and on ports should be modified to fit a particular solution.

2.4.3 Switch Port Management

Reading the following port attributes are supported:

- Administrative and operational port state
- Link speed
- Default VLAN
- Neighbor MAC
- QoS attributes (Data Center Bridging Capability Exchange (DCBX) state, Priority Flow Control (PFC) enabled, PFC enabled priorities, LLDP enabled)

Setting the following port attributes are supported:

- Administrative state
- QoS attributes (DCBX state, PFC enabled, PFC enabled priorities, LLDP enabled)

2.4.3.1 Port Management Limitations

Port Type is not read from the Arista* switch. All discovered ports are assumed to be "Downstream" unless specified otherwise in the PSME network agent configuration file.

2.4.4 Virtual LAN

The VLAN functionality allows the ability to manage VLANs dynamically using the PSME REST API. To configure VLANs on Arista switch ports through the PSME API, special requirements need to be satisfied.

1. Adding/deleting untagged VLANs is not supported.

2. There is exactly one untagged VLAN, and it is set as the primary VLAN on each port. The user cannot change the primary VLAN on a port but can change the ID of the untagged VLAN.
3. Any modification of VLANs on a given port through the PSME API can be verified on the switch CLI using the following commands:

   ```
   interface ethernet <port_id>
   show active
   ```

   or

   ```
   show interfaces ethernet <port_id> switchport
   ```

   Other CLI commands, like shown below, will only show VLANs defined using CLI and will NOT show VLANs configured through PSME API:

   ```
   show vlan
   show interfaces vlans
   ```
4. A VLAN cannot be disabled on a port only added or deleted. The `VLANEnable` API attribute is always returned as "true". The REST API or GAMI API does not support changing the `VLANEnable` API attribute.
5. Name and description cannot be assigned to a VLAN. These parameters are not configurable on the switch. To work around this HW limitation, the REST server assigns the Name attribute automatically. When a VLAN is created through the GAMI API, the supplied Name is discarded. Trying to change that attribute from the REST API or GAMI API is not supported.

2.4.5 Neighbor MAC Discovery

If a device is connected to a switch port and it is generating network traffic, then the PSME Network software is able to detect and expose the MAC address of the device. This feature is enabled only for Downstream ports.

If the device is shut down or the connecting cable is disconnected, the Neighbor MAC is not cleared on the port. As such, the value may represent an outdated state. It may change to

- a new value if a new device is connected and generating network traffic,
- `null` if the same MAC address is detected on another port.

2.4.6 Quality of Service

The following sections provide more details about Quality of Service configuration.

2.4.6.1 Description

The network contains many types of traffic flows. Classification of them is important to differentiate between the RDMA traffic and the other traffic flows. In many scenarios, RDMA traffic should have a higher priority than the other's traffic on the same link.

Quality of Service (QoS) allows the user to configure the network capability to provide better service to selected network traffic over Ethernet. `RoCEv2`-based NVMe-over Fabric (NVMe-oF) is one of the traffic types that require an appropriate level of network resources to be allocated.

QoS is designed as an end-to-end solution. Therefore, all endpoints and switches on the traffic path must be configured properly to achieve the right level of QoS.

The PSME provides APIs to configure some selected QoS parameters on the Leaf-switches. The following QoS parameters can be configured:

- **Priority-Based Flow Control (PFC)** - PFC is defined by *802.1Qbb – Priority-based Flow Control* specification, refer to table 3. It supports the flow control on individual priorities, as specified in the class of service field of the 802.1Q VLAN tag and defined by IEEE P802.1p.

- **Enhanced Transmission Selection (ETS)** - ETS is defined by *802.1Qaz – Enhanced Transmission Selection*, refer to Table 3. It enables dividing traffic according to its 802.1p priority into different priority groups and configures bandwidth for them.
- **Application Protocol mapping** - Application Protocol maps upper-layer applications to one of IEEE 802.1p priority. Endpoints assign a different priority to different traffic based on the protocol type and its port.

2.4.6.2 Remarks

- QoS parameters can be configured through network JSON configuration file (`network-config.json`) which is read and processed by PSME SW during Arista network agent initialization.
- For each QoS parameter specified in the configuration file, the PSME software updates the Arista switch configuration according to the new settings. The interface's parameters can be specified separately, or as the default configuration for all interfaces. Parameters not specified remain unmodified on the switch.
- Enabling DCBX for any interface through the configuration file automatically enables LLDP for this interface and on the switch.
- PFC, ETS and application protocol mapping must be configured for NVMe traffic to guarantee zero packet loss and bandwidth in the network.
- Endpoints can be configured manually by the administrator or automatically through the Arista switch after enabling DCBX functionality on the switch, as an extension of LLDP communication protocol.
- Endpoints must be enabled to accept the configuration sent by the Arista switch through the LLDP protocol.

2.4.6.3 Limitations for Arista switch

- The setting of QoS parameters is limited to interfaces with index "1" in the name (e.g. Ethernet25/1) and with index "2", "3", "4" (e.g. Ethernet25/2, Ethernet25/3, Ethernet25/4) if a QSFP to 4 x SFP+ splitter cable* is plugged in.
- DCBX enable/disable actions are not supported on the switch. DCBX mode must be configured per Ethernet interface only.
- In DCBX IEEE (Institute of Electrical and Electronics Engineers) mode, Application Protocol TLV is not broadcast (in LLDP frames) to the endpoints by the Arista switch. Only PFC and ETS parameters are sent. DCBX CEE (Converged Enhanced Ethernet) mode must be enabled to broadcast Application Protocol mapping as well.

2.4.6.4 QoS configuration on Sleds

For successful operation, QoS requires additional configuration to be performed on compute sleds. Refer to Appendix F, Additional Quality of Service (QoS) configuration for sleds for more details.

2.5 MultiRack

This functionality allows for managing multiple racks with one PODM. The PSME Chassis Agent is responsible for providing a unique location property for each drawer in a multi-rack setup. This property consists of the drawer's parent rack identifier and the drawer's offset within the rack. These values can be set in the PSME Chassis configuration file or can be overwritten by the RMM via Intelligent Platform Management Bus (IPMB) protocol.

2.6 Pooled NVMe Controller (PNC)

This section provides details specific to the Intel® RSD Software Development Vehicle.

2.6.1 Description

The Pooled Node controller allows for attaching end detaching NVMe drives and Intel® PAC cards (with Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA (Intel® PAC with Intel® Arria® 10 GX FPGA)) to compute nodes using PCIe cables.

2.6.2 Prerequisites

As a prerequisite to using the PSME PNC, you should have a setup with the Intel® RSD Software Development Vehicle PCIe Switch with attached NVMe drives. The management host should have a Linux kernel version supporting PCIe device hot swap.

If you are using the Intel® RSD Software Development Vehicle, contact Intel to receive detailed setup configuration instructions.

The PSME PNC software requires the following libraries to be installed in the OS:

```
libmicrohttpd10 libcurl3-gnutls libcurl3 libstdc++6(>=5.3.0)
```

The PSME PNC also requires using the *Aardvark* Software API* and Intel® Open Programmable Acceleration Engine (Intel® OPAE) Driver, refer to Table 4 to download a copy.

It also requires the encrypt binary to be installed in `/usr/bin/` directory. The binary can be compiled by following the instructions in Section 3.0, PSME Development Environment. If you receive pre-built binaries from Intel, then it is provided by the `psme-common` package.

2.6.3 Service Configuration

A default `psme-pnc` configuration file (`psme-pnc-configuration.json`) can be found in the PSME source tarball. To perform successful discovery and any later actions, the following fields in the configuration file must be properly filled (analogically as in the `psme-compute` configuration):

```
"i2c":{
    "interface":"IPMI",
    "username" : "put_username_hash_here",
    "password" : "put_password_hash_here",
    "port" : 623,
    "ipv4" : "put_ipmi_ip_here"
}
```

Where username and password hashes can be generated with

```
$ sudo encrypt <username>
$ sudo encrypt <password>
```

and the ipv4 field is the IP address of the PNC board BMC (not the IP of the sled).

When in doubt, use regular `psme-rest-server` and agents installation guides.

2.7 iSCSI Out of Band Booting

This section describes the management of iSCSI Out of Band Booting using Intel® Rack Scale Design software.

2.7.1 Description

This feature and its limitations are specific to the Intel® RSD Software Development Vehicle.

Due to BIOS implementation, there are two methods for configuring systems to boot from Internet Small Computer Systems Interface (iSCSI) targets, depending on the boot mode setting (Legacy vs. f UEFI). For Legacy, only PXE/iPXE is supported, and in this case system, `BootSourceOverride` parameters should be `PATCH`ed to PXE/Legacy. For UEFI, OOB iSCSI functionality is supported, and `BootSourceOverride` parameters should be `PATCH`ed to RemoteDrive/UEFI. This version also requires `PATCHing` the system's `NetworkDeviceFunction` with the correct data about an iSCSI Target.

2.7.2 Limitations

- If data about an iSCSI Target is incomplete or points to a non-existent iSCSI Target (a SLED cannot connect to the iSCSI Target - PSME will not detect it), then BIOS will attempt to boot from a local drive.
- iSCSI OOB Parameters should not be modified externally (via BIOS/IPMI), only the PSME Compute Agent should configure it.
- The user can choose which network interface should be used for booting from iSCSI by specifying a Media Access Control (MAC) address. If the specified MAC is missing, a default interface will be used. The same will occur if the user sets a non-existent MAC address.
- In this reference PSME feature, only IPv4 is supported.
- If booting from `RemoteDrive` is selected when BIOS is in Legacy mode, then a system will attempt to boot from a local drive.

When the `BootOverrideTarget` is set to `RemoteDrive` through the PSME API, then the PSME Compute Agent:

- sets BIOS's boot to override to "remotely connected Hard Drive",
- in `NetworkDeviceFunction` sets `"DeviceEnabled"` field to "**true**",
- copies iSCSI Target parameters from `NetworkDeviceFunction` to BMC's iSCSI OOB Parameters.

When the `BootOverrideTarget` is set to anything else except `RemoteDrive`, then PSME Compute Agent:

- sets BIOS's boot to override to the selected option,
- in `NetworkDeviceFunction` sets `"DeviceEnabled"` field to "**false**", but other parameters are not modified,
- clears BMC's iSCSI OOB Parameters.

When in `NetworkDeviceFunction` `"DeviceEnabled"` is set to "**false**", then fields in `NetworkDeviceFunction` are not synchronized with the BMC's iSCSI OOB Parameters. If the PSME Compute Agent is restarted when the "`DeviceEnabled`" is set to "**false**", then the `NetworkDeviceFunction` fields will not be remembered.

When in `NetworkDeviceFunction` `"DeviceEnabled"` is set to "**true**", then fields in `NetworkDeviceFunction` are synchronized with BMC's iSCSI OOB Parameters.

The `"DeviceEnabled"` flag in `NetworkDeviceFunction` cannot be overridden, and the flag is only modified by changing the `BootOverride`Target.

Table 5. How the PSME Handles Possible BootSourceOverrideEnabled Continuous Options

BootSourceOverrideEnabled Continuous with BootOverrideTarget:	BIOS Boot Override	OOB iSCSI Parameters	Will boot from
Pxe	Pxe	irrelevant	Pxe
Hdd	Hdd	cleared	Hdd
RemoteDrive	Hdd	set from NetworkDeviceFunction	RemoteDrive

Due to BMC/BIOS limitations, setting `BootSourceOverrideEnabled` to "**Once**" on a system has the following restrictions:

- if previously BootSourceOverrideEnabled was set to "**Continuous**" with the BootOverrideTarget set to **RemoteDrive**, and currently the BootSourceOverrideEnabled is set to "**Once**" with BootOverrideTarget set to **PXE**, then after a BootSourceOverrideEnabled Once time-out, or a second power cycle, the system will boot from the RemoteDrive,
- if previously BootSourceOverrideEnabled was set to **Continuous** with the BootOverrideTarget set to **RemoteDrive**, and currently BootSourceOverrideEnabled is set to **Once** with the BootOverrideTarget set to **Hdd**, then after a BootSourceOverrideEnabled Once time-out, or a second power cycle, the system will boot from the Hdd,
- if the BootSourceOverrideEnabled was previously set to **Continuous** with the BootOverrideTarget set to **Hdd**, and currently the BootSourceOverrideEnabled is set to **Once** with the BootOverrideTarget set to **RemoteDrive**, then after a BootSourceOverrideEnabled Once time-out, or a second power cycle, the system will boot from the RemoteDrive.

Note: In above-mentioned cases, a new PATCH request setting BootSourceOverrideEnabled to **Once/Continuous** must be sent to alter current boot order.

Table 6. How the PSME Handles Possible BootSourceOverrideEnabled Once Options

BootSourceOverrideEnabled Once with BootOverrideTarget:	Previous BootSourceOverrideEnabled Continuous with BootOverrideTarget:	BIOS Boot Override	OOB iSCSI Parameters	Will boot from	After BootSourceOverride Enabled Once time-out or a second power on, will boot from
Pxe	Pxe	Pxe	irrelevant	Pxe	Pxe
Pxe	Hdd	Hdd	cleared	Pxe	Hdd
Pxe	RemoteDrive	Pxe	cleared	Pxe	Hdd
Hdd	Pxe	Hdd	cleared	Hdd	Pxe
Hdd	Hdd	Hdd	cleared	Hdd	Hdd
Hdd	RemoteDrive	Hdd	cleared	Hdd	Hdd
RemoteDrive	Pxe	Hdd	set from NetworkDeviceFunction	RemoteDrive	Pxe
RemoteDrive	Hdd	Hdd	set from NetworkDeviceFunction	RemoteDrive	RemoteDrive
Remote Drive	RemoteDrive	Hdd	set from NetworkDeviceFunction	RemoteDrive	RemoteDrive

2.7.3 NetworkDeviceFunction Parameters

There is a minimal set of NetworkDeviceFunction parameters which should be configured to boot from iSCSI OOB (for the default PODM configuration in which the Initiator IP/netmask/gateway is received from DHCP):

```
"IPAddressType": "IPv4"
"IPMaskDNSViaDHCP": true
"TargetInfoViaDHCP": false
"AuthenticationMethod": "None"
"InitiatorName"
"PrimaryTargetName"
"PrimaryTargetIPAddress"
"PrimaryTargetTCPPort"
"PrimaryLUN"
```

It is required to put a "MACAddress" of a network card from which iSCSI Boot shall be performed for a given system:

```
"Ethernet": {
    "MACAddress" : "AA:BB:CC:DD:EE:FF"
}
```

When "TargetInfoViaDHCP" is set to "true", then following fields will not be updated in BMC's iSCSI OOB Parameters:

```
"PrimaryTargetName"
"PrimaryTargetIPAddress"
"PrimaryTargetTCPPort"
"PrimaryLUN"
```

NetworkDeviceFunction parameters which are not supported:

```
"SecondaryTargetName"
"SecondaryTargetIPAddress"
"SecondaryTargetTCPPort"
"SecondaryLUN"
"SecondaryVLANEnable"
"SecondaryVLANId"
"RouterAdvertisementEnabled"
```

NetworkDeviceFunction only validates types and lengths of input data, and it cannot verify if a complete minimal set of parameters is set, or if a system will be able to connect to a given iSCSI Target.

If "AuthenticationMethod" is not "None", then related CHAP username/password fields should also be set.

The user is able to patch passwords for the CHAP authentication. However, the PSME REST API will always display null values due to security concerns.

2.7.4 Networking for iSCSI Booting in Intel® RSD Software Development Vehicle

PSME Storage Service provides iSCSI Targets in the v10.1.* or v10.2.* network. Change the providing network by configuring "portal-interface" in /etc/psme/psme-storage-configuration.json file on Storage Services (restart of PSME Storage Agent is required).

It is crucial to put in NetworkDeviceFunction parameters and "MACAddress" of a network card working in the same network as "portal-interface".

2.7.5 Intel® RSD Software Development Vehicle Limitations

- The InitiatorName and PrimaryTargetName shall contain the "iqn." prefix followed by up to 200 characters.
- The CHAPUsername and MutualCHAPUsername shall have a maximum of 32 characters.
- The CHAPSecret and MutualSecret shall have between 12 and 16 characters.
- The PrimaryLUN field can be "0" when reading from cleared BMC's iSCSI OOB Parameters.
- The PSME Compute Agent reads BootSourceOverrideEnabled, BootOverrideTarget and BootOverrideMode fields from the BMC. When these fields are set through the PSME, the BMC may display their real values only for a limited time period (60s-120s). After this time the BMC/PSME will return default values of these fields, but the BIOS will remember the previously set configuration.
- If only the BootSourceOverrideEnabled and BootOverrideTarget fields are patched, but the BootOverrideMode field is omitted, then the BootOverrideMode field is reconfigured with the value currently returned by BMC/PSME.
- The BMC by default returns "Legacy" mode after the time-out mentioned in one of the limitations above (60s-120s).

Note: Remember also to patch the field `BootOverrideMode` when patching the `BootSourceOverrideEnabled` and `BootOverrideTarget fields`, and you intend to boot in `"UEFI"` mode.

2.8 Telemetry Service

The telemetry service is a service which periodically polls the hardware for the telemetry data. Gathered data is exposed on the REST API in the form of metric values, resource health states and events for these resources (either resource changed or alerts).

The telemetry service provides metric definitions for all handled metrics. Metric definitions describe metrics and parameters impacting the metric value calculation/presentation:

- Polling interval
- Shoring up health events for particular periods
- Data computations for numeric metrics (averaging/getting minimum or maximum over a specified time window)
- Rounding numeric metrics

2.8.1 Configuration

The appropriate agent config file (for example, `/etc/psme/psme-compute-configuration.json`) can contain the specific settings for each metric definition and common settings for all metric definitions. If the telemetry section is not included in the file, then internal defaults are used.

2.8.1.1 Metric Definition Specific Settings

For each metric definition, there is an object containing properties to be overridden. All settings for a metric definition are stored in an appropriate object in the telemetry section of the config file. The name of the object is identical with the metric definition name. For the Intel® Xeon® Scalable processor platform from Intel (codename Purley), the only handled metric definitions names are:

- `processorConsumedPower`,
- `systemConsumedPower`,
- `processorTemperature`,
- `memoryTemperature`,
- `memoryConsumedPower`,
- `processorMarginToThrottle`,
- `systemProcessorBandwidth`,
- `systemMemoryBandwidth`,
- `systemIOBandwidth`,
- `sledInletTemperature`,
- `sledOutletTemperature`,
- `sledInputACPower`,
- `processorAverageFrequency`,
- `processorHealth`,
- `systemHealth`,
- `memoryHealth`,
- `systemMemoryThrottlingPercent`,
- `memoryLastShutdownSuccess`,
- `memoryPredictedMediaLifeLeftPercent`,

- `memoryAlarmTripsTemperature`,
- `moryControllerTemperature`,
- `memoryUptimeSeconds`,
- `memoryUnsafeShutdownCount`,
- `memoryPowerCycles`,
- `memoryPowerOnTimeSeconds`,
- `memoryCurrentPeriodBlocksRead`,
- `memoryCurrentPeriodBlocksWritten`,
- `memoryCurrentPeriodHostReadRequests`,
- `memoryCurrentPeriodHostWriteRequests`.

The list of available properties, their description, and allowed values, is available in the metadata for `MetricDefinition`.

Note: The first letter of the property name must be changed to lowercase.

Some of these properties have special meanings for metric computation algorithms:

- `sensingInterval` defines a polling interval for the metric (see Table 4, *Date and time format - ISO 8601*),
- `calculationAlgorithm` defines calculation algorithm to be used to calculate metrics of average/min/max `(averageOverInterval`, `minOverInterval`, `maxOverInterval`),
- `calculationTimeInterval` defines the time window for `calculationAlgorithm` (see Table 4, *Date and time format - ISO 8601*),
- `calculationPrecision` defines the precision of calculations (a float value).

There is a special property (neither available on the REST API nor in the metadata):

`shoreupPeriod` defines the period in which events are to be shored up (either ISO 8601 format or a float value representing a number of seconds, refer to Table 4).

Some of the properties cannot be overridden; only predefined values apply. These are:

- `the name` is a key for metric definition identification
- `dataType` defines the type of metric value
- `metricType` defines the logic for particular metric values
- `discreteValues` defines the discrete metric value format (either a single value or an array of values).

All settable properties unconditionally override code-defined values.

2.8.1.2 Common Settings for All Metric Definitions

Common properties are stored directly in the telemetry section of the config file.

There are two common properties:

- `defaultInterval` default value for `sensingInterval` (30s by default),
- the `shoreupPeriod` default value for shoring up events (10s by default).

Both properties might be specified either as an ISO 8601 string or a numeric value (number of seconds).

Common settings apply to metric definitions in which appropriate values are **not set** (these do not override predefined values).

2.8.1.3 Config File Example

```
{
    ....
    "telemetry": {
        "defaultInterval": "PT10S",
        "shoreupPeriod": 30.0,
        "sledInletTemperature": {
            "sensingInterval": 60
        },
        "processorConsumedPower": {
            "calculationPrecision": 10.0,
            "calculationAlgorithm": "AverageOverInterval",
            "calculationTimeInterval": "PT60S"
        },
        "systemHealth": {
            "shoreupPeriod": "PT2M"
        }
    },
    ...
}
```

2.8.1.4 Sample Computations

Average calculation is done according the equation shown in Figure 1.

Figure 1. Average Equation

$$\bar{s} = \sum_{i=k+1}^{m} \frac{(s_{i-1} + s_i)(t_i - t_{i-1})}{2(t_m - t_k)}$$

Let's assume that a hypothetical sensor is returning integer values. The values are read with the `sensingInterval` of 1s ("PT1S"). `calculationTimeInterval` is 7s (`"PT7S"`). According to these assumptions the equation simplifies to the equation shown in Figure 2.

Figure 2. Simplified Average Equation

$$\bar{s} = \frac{\sum_{i=k+1}^{m} s_{i-1} + s_i}{2(m-k)}$$

Bold values in Table 7 indicate when `ResourceChanged` events (a Redfish* event type) are sent.

Table 7. Computation Results for Sample Set of Readings

T	s	k..m	avg	prec=0.5	prec=1	prec=5
0	null	-	---	---	-	-
1	2	1	**2**	**2.0**	**2**	**0**
2	3	1..2	**2.5**	**2.5**	**3**	**5**
3	6	1..3	**3.5**	**3.5**	**4**	5
4	7	1..4	**4.5**	**4.5**	**5**	5
5	5	1..5	**4.875**	**5.0**	5	5
6	4	1..6	**4.8**	5.0	5	5
7	2	1..7	**4.5**	**4.5**	5	5
8	3	1..8	**4.214...**	**4.0**	**4**	5
9	3	2..9	**4.285...**	**4.5**	4	5
10	4	3..10	**4.142...**	**4.0**	4	5
11	null	-	**---**	**---**	**-**	**-**

T	s	k..m	avg	prec=0.5	prec=1	prec=5
12	4	12	**4**	**4.0**	**4**	**5**
13	5	12..13	**4.5**	**4.5**	**5**	5

Applying the precision reduces the number of events, as follows:

- 13 events sent for non-rounded values (no `calculationPrecision` property is set)
- 12 events for `calculationPrecision` = 0.5,
- Eight events for `calculationPrecision` = 1,
- Four events for `calculationPrecision` = 5.

2.8.2 Limitations

- The telemetry service is fully compliant with the Intel platform codename Purley.
- Configuration is handled by the compute agent only (`/etc/psme/psme-compute-configuration.json file`).
- Metric definition properties cannot currently be modified via REST APIs.

2.9 CHAP Authentication

This feature and its limitations are based upon Linux SCSI target framework (tgt). PSME Storage Services support CHAP authentication during connection to an iSCSI target.

In One-Way mode authentication, the target verifies the initiator's username and password. To enable the one-way authentication mode in PSME Storage Services, the initiator's credentials must be specified in a "Target" Endpoint (i.e., an Endpoint with a Connected Entity with `"Role"` property set to "**Target**").

In Mutual mode authentication, the initiator verifies the target. To enable this mode, the target's credentials must be specified in an `"Initiator"` Endpoint (i.e., an Endpoint with a Connected Entity with `"Role"` property set to `"Initiator"`).

When sending `POST` or `PATCH` request to Endpoint, Username and Password must both contain non-empty strings to enable authentication, or both be null to disable it.

2.9.1 Limitations

- The user is able to `PATCH` passwords for CHAP authentication. However, the PSME REST API will always display Password as null due to security concerns.
- Endpoint Usernames and Passwords cannot contain whitespace characters.
- The Storage Agent stores Endpoint Passwords until it is restarted.
- The Storage Agent preserves Endpoint Usernames on restart by saving them in a database.
- If an Endpoint has a Username and a Password but is not added to any Zone representing an iSCSI Target, then its credentials will not be preserved after the Storage Agent restart.
- If an Endpoint is in a Zone representing iSCSI Target and the Storage Agent is restarted, then its credentials will be cleared when the Endpoint is removed from a Zone.
- Restart of the tgt daemon on Storage Services deletes CHAP authentication from all iSCSI Targets and requires a restart of the PSME Storage Agent. CHAP authentication is not automatically restored (however, PODM can automatically create and assign new credentials to Endpoints).
- The tgt iSCSI target credentials and CHAP accounts must not be modified externally (i.e., via command line), these operations must be performed using PSME.
- The tgt should not have any CHAP accounts which are not assigned to iSCSI Targets when PSME Storage Agent is started.

2.10 Simple Service Discovery Protocol (SSDP)

The PSME REST server can be configured to announce its presence over the network using IP multicast datagrams carrying Simple Service Discovery Protocol (SSDP) presence announcements. In the PSME REST server's configuration, the `ssdp-service` section determines the behavior of PSME's SSDP service, and it contains the options shown in Table 8.

Table 8. Configuration Options for SSDP Service

Option	Type	Description
Enabled	Boolean	specifies if SSDP service should be enabled
announce-interval-seconds	integer	if the interval is non-zero, it specifies the number of seconds between SSDP presence announcements. Zero interval means no announcements will be sent.
Ttl	integer	specifies the time to live value of notifying multicast datagrams

2.11 NVMe-over Fabric (NVMe-oF)

This feature enables attaching NVMe volumes to remote hosts over an Ethernet network using RDMA-capable NICs.

A full NVMe-oF solution consists of:

- One or more hosts providing volumes (the `"targets"`)
- One or more client hosts (the `"initiators")`
- A single discovery service which responds to queries from the initiators about volumes which are available to them for attachment

The PSME solution provides:

- The PSME NVMe agent which manages the target host
- The PSME NVMe Discovery agent which implements the NVMe-oF discovery Service

The user must provision the client hosts with a tool which will periodically query the discovery service to get a list of volumes currently available for attachment and manage connections to these volumes. The section "Provisioning Initiator Hosts" included below describes how such a tool could be designed.

2.11.1 The PSME NVMe Agent for Target Hosts

The PSME NVMe agent is responsible for managing and gathering detailed information about NVMe volumes attached to hosts through remote direct memory access (RDMA) NICs using NVMe-oF technology. The agent runs on a target storage host.

2.11.1.1 Prerequisites

As a prerequisite to using the PSME NVMe agent, you should have a host with attached NVMe drives. The host should have the kernel modules `nvmet,` and `nvmet-rdma` enabled to allow for exposing NVMe targets over Ethernet. Furthermore, it should have an RDMA-capable network interface with appropriate drivers installed and modules enabled. Finally, it should have a Linux kernel supporting RDMA and NVMe features.

The PSME NVMe target service requires the following dependencies to be installed in the OS:

```
libmicrohttpd10 libcurl3 libnl-genl-3-200 libnl-route-3-200
```

If you are using the Intel® RSD Software Development Vehicle, then you can contact Intel to receive detailed setup configuration instructions.

To use PSME GPT NVMe agent: The host should have the kernel modules nvmet, and nvmet-rdma enabled to allow for exposing NVMe targets over Ethernet.

To use PSME SPDK NVMe agent: The SPDK NVMe-oF application `nvmf_tgt` should be run on the host machine. More information about the configuration of SPDK can be found in Section 5.6, PSME SPDK NVMe Storage Service.

2.11.1.2 Service Configuration and Detection of RDMA Interfaces

The default configuration file can be found in the PSME source tarball (in `agent/nvme/configuration.json` for PSME GPT NVMe agent or `agent/spdk/configuration.json` for PSME SPDK NVMe agent). The option nic-drivers in the file contains the list of Linux network drivers which will be used to gather a list of network interfaces that are on a given host. Only the interfaces that were created by one of the listed drivers are exposed on the API.

If an interface is missing from the API, then run the following command to determine the driver name of a network interface:

```
ethtool -i <interface_name>
```

The user should then add the driver name to the list and restart the PSME services.

2.11.1.3 Remarks about the Agent's Operation

- NVMe drives need to be visible in `SYSFS` before the PSME agent is started (i.e., PCIe switch drives need to be bound to the host)
- Each zone can have multiple target endpoints but only one initiator endpoint
- An endpoint cannot be deleted if it is in a zone
- An endpoint can only be in one zone at a time
- The PSME GPT NVMe agent does not allow creating Volume clones or snapshots
- The PSME SPDK NVMe agent allows creating Volume clones and/or snapshots. The limitations of the feature are described in the section "Intel® RSD Drawer configuration : PSME SPDK NVMe Storage Service : Limitations."

2.11.2 The PSME NVMe Discovery Service

The PSME NVMe discovery agent is responsible for responding to queries about available NVMe volumes from NVMe-oF initiators. This service can either be run on a separate host or on the target host. In the latter case, several requirements must be fulfilled to ensure correct operation of both PSME services. They are described in Section 2.11.2.3, Running the PSME NVMe Discovery Agent on a Target Host.

2.11.2.1 Prerequisites

As a prerequisite to using the PSME NVMe discovery service, you should have a host with an RDMA-capable network interface with appropriate drivers installed, and modules enabled (including userspace RDMA support). It should also have a Linux kernel supporting RDMA.

The PSME discovery service is based on `InfiniBand` Verbs interface. It thus, therefore, requires userspace RDMA and Verbs modules to be enabled as well as vendor-specific libraries enabling Verbs on the network interface. The setup may be verified using the `rping` (RDMA ping) utility.

The PSME discovery service also depends on several libraries which need to be installed in the OS:

```
libmicrohttpd10 libcurl3 libnl-genl-3-200 libnl-route-3-200
```

Note: If you are using the Intel® RSD Software Development Vehicle, contact Intel to receive detailed setup configuration instructions.

2.11.2.2 Service Configuration

The default PSME NVMe discovery configuration file can be found in the PSME source tarball (in agent/nvme-discovery/configuration.json).

The discovery-service section in the configuration file must contain the list of network interfaces on which the service will handle queries from initiators, as shown in the sample:

```
"discovery-service": {
    "listener-interfaces": [
        {
            "ofi-provider" : "verbs",
            "trtype" : "rdma",
            "adrfam" : "ipv4",
            "traddr": "127.0.0.1",
            "trsvcid": "4420"
        }
    ]
}
```

If the default configuration file for PSME REST server is used, it must be adjusted in the following way:

```
   "service-root-name" : ["PSME Service Root"] -> "service-root-name"" : ["Discovery
Service Root"]
```

2.11.2.3 Running the PSME NVMe Discovery Agent on a Target Host

If the PSME discovery service is to run along with the PSME NVMe service on the target host, a second instance of the PSME REST Server should be also run which will handle data, requests, and events related solely to the discovery service.

The PSME source tarball contains an example configuration for the second PSME REST Server (in application/discovery-configuration.json). It contains changed port numbers to be used by GAMI and Redfish services. It also specifies a separate service name and database location for the second REST Server.

Note: The network-interface-name in the configuration for the second REST Server should specify a different interface because these services should be available on separate IP addresses.

2.11.2.4 Remarks about the Discovery Agent's Operation

- Each zone can have multiple target endpoints but only one initiator endpoint
- An endpoint cannot be deleted if it is in a zone
- An endpoint can only be in one zone at a time

2.11.3 Provisioning Initiator Hosts

Each initiator host must poll the Discovery Service to ensure that its connections to remote volumes are up-to-date. A tool must be created to perform these actions. This section describes the proposed operation of a tool which would be installed on an initiator host.

The tool can be a CRON job script which wraps up the NVMe management command line interface (nvme-cli) utility, which can be obtained from its repository.

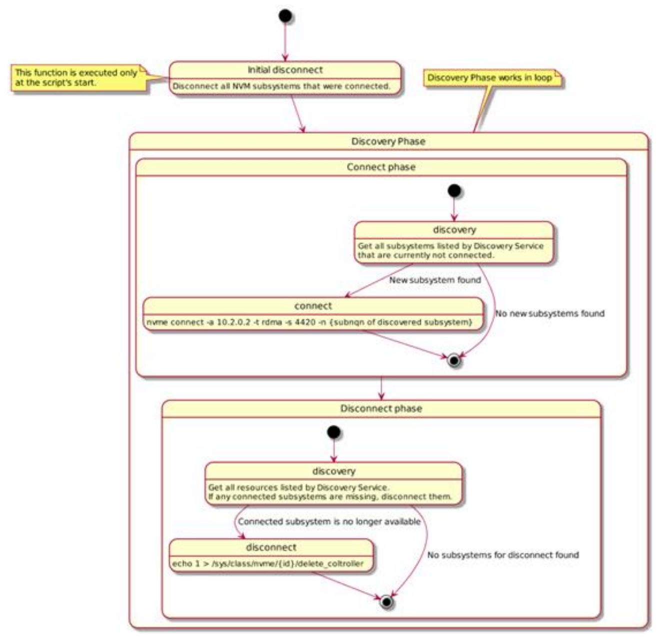
For `nvme-cli` to work properly, the initiator host should have the kernel modules `nvme` and `nvme-rdma` enabled to allow for attaching NVMe targets. Furthermore, it should have an RDMA-capable network interface with appropriate drivers installed and modules enabled. It should also have a Linux kernel supporting RDMA and NVMe features.

Note: If using the Intel® Rack Scale Design Software Development Vehicle, contact Intel to receive detailed setup configuration instructions.

As presented in the UML diagram below, the script would repeatedly poll the Discovery Service and therefore must be provided with the IP address of the discovery service host. The response would contain a list of available NVMe subsystems. After parsing the received information, it would try to attach new subsystems using `nvme-cli`. For each subsystem, if this process succeeds, then information about the subsystem would be stored in its database. If the Discovery Service response was missing some subsystems which were previously attached by the script, it would detach them using nvme disconnect (using `nvme-cli`) and remove their records from its database.

On startup, the script should detach all subsystems recorded in its database using nvme disconnect and clear the database. This behavior shall ensure that duplicate attachments of already mounted subsystems don't occur.

Figure 3. State Diagram for the Proposed NVMe Initiator Script


This function is executed only at the script's start.
Initial disconnect
Disconnect all NVM subsystems that were connected.
Discovery Phase works in loop
Discovery Phase
Connect phase
discovery
Get all subsystems listed by Discovery Service that are currently not connected.
New subsystem found
connect
nvme connect -a 10.2.0.2 -t rdma -s 4420 -n {subnqn of discovered subsystem}
No new subsystems found
Disconnect phase
discovery
Get all resources listed by Discovery Service. If any connected subsystems are missing, disconnect them.
Connected subsystem is no longer available
disconnect
echo 1 > /sys/class/nvme/{id}/delete_coltroller
No subsystems for disconnect found

2.11.4 Recommended Quality of Service Settings for NVMe over-Fabrics

NVMe requires an appropriate level of network resources to be allocated to minimize latency and maximize bandwidth for NVMe traffic. To ensure that level of resources, Quality of Service (QoS) should be configured on all switch interfaces with NVMe traffic.

QoS can be configured through the PSME API by the upper layer (orchestration) software or through the configuration file during PSME network agent initialization. For more details, refer to Section 2.4.6. Quality of Service.

Table 9 and Table 10 show the default configurations that should be applied on the switch.

Table 9. Default QoS Configuration of the Arista* Switch for NVMe-oF Purposes

Feature	Parameter	Value	Description
Priority Flow Control (PFC)	PFC Enabled	True	PFC must be enabled on the switch.
Enhanced Transmission Selection (ETS)	Priority to Priority Group	Priority=5, `PriorityGroup`=1	Priorities must be mapped to Priority Groups.
ETS	Bandwidth Allocation	`PriorityGroup`=1, Bandwidth=50%	Bandwidth percent must be allocated to Priority Groups.
Application Protocol mapping	Application Protocol	Protocol=UDP, Port=4791, Priority=5	Priorities must be assigned to application protocols identified by protocol id and port.
Link Layer Discovery Protocol (LLDP)	LLDP Enabled	True	LLDP must be enabled on the switch.

Table 10. Default QoS Configuration of the Arista Switch Interfaces for NVMe-oF Purposes

Feature	Parameter	Value	Description
Priority Flow Control (PFC)	Enabled	True	PFC must be enabled on the interface.
Priority Flow Control (PFC)	Enabled Priorities	5	Enabled priorities must be specified.
Data Center Bridging Capability Exchange (DCBX)	State	CEE	DCBX mode must be configured.
Link Layer Discovery Protocol (LLDP)	LLDP Enabled	True	LLDP must be enabled on the interface.

2.12 FPGA-over Fabric (FPGA-oF)

This feature enables attaching the FPGA to remote hosts over an Ethernet network using RDMA-capable NICs or TCP.

A full FPGA-oF solution consists of:

- One or more hosts providing volumes (the `"targets"`)
- One or more client hosts (the `"initiators"`)
- A single Discovery Service which responds to queries from the initiators about FPGAs which are available to them for attachment.

The PSME solution provides:

- The PSME FPGA-oF agent which manages the target host.
- The PSME FPGA-oF Discovery agent which implements the FPGA-oF Discovery Service.

The user must provision the client hosts with a tool which will periodically query the discovery service to get a list of FPGAs currently available for attachment. Refer to Section for instructions on how to design such a tool.

2.12.1 The PSME FPGA-oF Agent for Target Hosts

The PSME FPGA-oF agent is responsible for managing FPGAs that can be attached to hosts through RDMA NICs or TCP using FPGA-oF over Fabrics protocol. The agent runs on a separate management host (can be a regular compute host).

2.12.1.1 Prerequisites

As a prerequisite to using the PSME FPGA agents, you should have a host with attached FPGA. Furthermore, the host should have an RDMA-capable or TCP-capable network interface with appropriate drivers installed and modules enabled. Finally, if support for RDMA is enabled, it should have a Linux kernel supporting RDMA.

The PSME FPGA-oF target service requires the following dependencies to be installed in the OS:

```
libmicrohttpd-dev, libcurl4-openssl-dev, libnl-3-200, libnl-route-3-200, libibverbs1,
librdmacm1, libfabric
```

If using the Intel® Rack Scale Design Software Development Vehicle, contact Intel to receive detailed setup configuration instructions.

To use the PSME FPGA-oF agent: The host should have installed the Open Programmable Acceleration Engine (OPAE) driver. The recommended version of the driver is v1.3.0, which can be downloaded from OPAE GitHub: *https://github.com/OPAE/opae-sdk/releases/tag/1.3.0-2*.

2.12.1.2 Service configuration and detection of RDMA interfaces

The default configuration file can be found in the PSME source tarball (in agent`/fpga-of/configuration.json` for PSME FPGA-oF agent. The option `nic-drivers` in the file contains the list of Linux network drivers which will be used to gather a list of network interfaces that are on a given host. Only the interfaces that were created by one of the listed drivers are exposed on the API.

If an interface is missing from the API, then run the following command to determine the driver name of a network interface:

```
ethtool -i <interface_name>
```

Once the driver name of the network interface is determined, then add the driver name to the list and restart the PSME services.

2.12.1.3 Remarks about the Agent's Operation

- Each zone can have multiple target endpoints, but only one initiator endpoint
- An endpoint cannot be deleted if it is in a zone
- An endpoint can only be in one zone at a time

2.12.2 The PSME FPGA-oF Discovery Service

The PSME FPGA-oF discovery agent is responsible for responding to queries about available FPGAs from the FPGA-oF initiators. This service can either be run on a separate host or on the target host. In the latter case, several requirements must be fulfilled to ensure correct operation of both PSME services. They are described in Section 2.12.2.3, Running the PSME FPGA-oF Discovery Agent on a Target Host.

2.12.2.1 Prerequisites

The PSME FPGA-oF discovery agent communicates over the `json-rpc` protocol, so it does not need any special network interface.

The PSME discovery service, however, depends on several libraries which need to be installed in the OS:

```
libmicrohttpd10 libcurl3 libnl-genl-3-200 libnl-route-3-200
```

If you are using the Intel® Rack Scale Design Software Development Vehicle, then you can contact Intel® to receive detailed setup configuration instructions.

2.12.2.2 Service Configuration

The default PSME FPGA-oF discovery configuration file can be found in the PSME source tarball (in `agent/fpga-discovery/configuration.json`).

The `discovery-service` section in the configuration file must contain the port field which describes on which the service will handle queries from initiators.

If the default configuration file for PSME REST server is used, it must be adjusted in the following way:

```
   "service-root-name" : ["PSME Service Root"] -> "service-root-name"" : ["Discovery
Service Root"]
```

2.12.2.3 Running the PSME FPGA-oF Discovery Agent on a Target Host

If the PSME discovery service is to run along with the PSME FPGA-oF service on the target host, a second instance of the PSME `REST` Server should be run too, which will handle data, requests, and events related solely to the discovery service.

The PSME source tarball contains an example configuration for the second PSME `REST` Server (in application/discovery-configuration.json). It contains changed port numbers to be used by the GAMI and Redfish services. It also specifies a separate service name and database location for the second `REST` Server.

Note: The network-interface-name in the configuration for the second `REST` Server should specify a different interface because these services should be available on separate IP addresses.

2.12.2.4 Remarks about the discovery agent's operation

- Each zone can have multiple target endpoints but only one initiator endpoint
- An endpoint cannot be deleted if it is in a zone
- An endpoint can only be in one zone at a time

2.12.3 Provisioning FPGA-oF Initiator Hosts

Each initiator host must poll the Discovery Service to ensure that its connections to remote FPGAs are up-to-date. A tool must be created to perform these actions. This section describes the proposed operation of a tool which would be installed on an initiator host.

The tool can be the FPGA-oF Initiator wheel which communicates with the discovery service over `json-rpc` protocol. The Initiator scripts provide the Discovery Service with its host GUID and put obtained data into a config file, which in turn is read by the FPGA-oF stack on the initiator host.

The FPGA-OF Initiator wheel is avaialbe at: *https://github.com/intel/intelRSD/tree/2.4/PSME/fpgaof-wheel*

The component that reads the config file is the Remote Plugin which will enable the use of remote FPGAs through OPAE library. After the plugin is installed and is provided with target host transport details, any application using a compatible subset of OPAE API should be able to use FPGA just as it was a local device.

The OPAE Remote Plugin can support TCP and RDMA transports. In order to use RDMA, the host should have an RDMA-capable network interface with appropriate drivers installed and modules enabled. It should also have a Linux kernel supporting RDMA feature.

Note: The location of the remote plugin needs to be specified in the /etc/opae/opae.cfg configuration file.

Moreover, to enable the OPAE remote plugin to connect to remote targets, it itself needs to be provisioned with the following:

- Tthe system UUID that can be read from sysfs,
- Tthe location of the file what will be the output of the FPGA-oF Initiator wheel..

If you are using the Intel® Rack Scale Design Software Development Vehicle, then you can contact Intel® to receive detailed setup configuration instructions.

2.13 Security Features

This section describes the security features of the Intel® RSD PSME software.

2.13.1 Secure communication over TLS

Certificates described in this section can be easily generated, or use existing certificates – refer to Table 4, *Intel® RSD POD Manager User Guide*, Section 3.2.2, Key and Certificate management.

Every PSME instance can be configured only to accept HTTPS connections secured with the TLS protocol. The PSME implements mutual (bi-directional) authentication.

The following points present the details of this feature:

1. All certificates (and/or private keys) must be stored in a secure directory. The directory must be placed in the local file system and must be available for the root user only. No `symlinks` are allowed.

   Default directory `/etc/psme/certs` may be changed in the appropriate `psme-rest-server-configuration.json` file:

   ```
   "server": { "connectors" : [ { "use-ssl" : true, "certs-directory" :
   "/etc/psme/certs", ...
   ```

2. For the PSME REST Server, the appropriate private key (`server.key`) and a certificate signed by an authorized CA (`server.crt`, to authenticate the PSME REST-Server service in the POD Manager) must be manually stored in certificate directory.

Note: In the Arista environment, the ROOT filesystem is initialized with each reboot. Certificates are automatically copied from the /mnt/flash/certs directory during package installation to the /etc/psme/certs directory.

3. The certificate signed by the authorized CA (which is used to authenticate the POD Manager in PSME Rest Server services), must be manually stored on the RMM as ca.crt file:

   ```
       cp podm.crt /etc/psme/certs/ca.crt
   ```

   The stored certificate file must be in PEM format.

   To enable the PSME REST Server to work with the PSME Chassis and RMM, the following must be added to the `psme-rest-server-configuration.json` file:

   ```
       "rmm-present" : true
   ```

   The RMM installs this certificate to all Drawers within the rack over I^2C by means of IPMB calls. To perform certificate deployment automatically, the PSME Chassis agent and `cyMUX` service must be installed on each drawer in the rack.

   The default location of the PODM certificate might be changed by the property in `the psme-rmm-configuration.json` file:

   ```
    "certificate-files": { "podm" : "/etc/psme/certs/ca.crt" },
   ```

4. The PSME allows POD Manager authentication without the RMM and/or PSME Chassis Agents.

   The `ca.crt` file must be placed in the certificates directory on each PSME Rest Server:

   ```
       cp podm.crt /etc/psme/certs/ca.crt
   ```

   The `rmm-present` flag must also be disabled (in the `psme-rest-server-configuration.json`):

   ```
       "rmm-present" : false
   ```

Note: Every PSME Storage, NVMe-oF, RMM, and Network agent must be configured in such a way (there is no I2C connection to communicate with RMM).

5. The PSME allows for disabling client (the POD Manager) authentication. To do this, change the SSL connector configuration (in the psme-rest-server-configuration.json file):

   ```
       "client-cert-required": false
   ```

   In this mode, any client would be able to connect to the PSME Rest Server service without authentication.

6. For correct PODM certificate verification, the system (on which the PSME runs) should have an NTP client up and running. The configuration should be as follow:

    a. In `/etc/ntp.conf` in servers section:

    ```
     server 0.rhel.pool.ntp.org iburst
    server 1.rhel.pool.ntp.org iburst
    ```

    Add your local NTP server:

    ```
     server IP_OF_YOUR_LOCAL_NTP_SERVER prefer
    ```

    prefer specifies that this server is preferred over other servers. A response from the preferred server will be discarded if it differs significantly from the other servers' responses.

    b. Start the NTP Daemon.

    ```
     /etc/init.d/ntp start
    ```

    c. Set initially local date and time.

    ```
     ntpdate -u IP_OF_YOUR_LOCAL_NTP_SERVER
    ```

    This command is used to synchronize your time with NTP server time manually. After initial sync, NTP client will automatically perform periodic sync with NTP server.

7. Intel® RSD 2.5 offers no mechanisms for certificate renewal or expiration. Thus, certificate management is left to the administrator. PSME software should be restarted if any of the files with certificates or private keys are updated.

2.13.2 Binding the REST Server to a Network Interface

The PSME REST Server configuration should be adjusted to list the interfaces on which the service should listen for requests. In the standard scenario, the list should contain only the management interface - the one on the same subnet as the PODM software managing the service. To set the interface, change the following section of `/etc/psme/psme-rest-server-configuration.json`:

```
    "network-interface-name": ["enp0s20f0.4094"] -> "network-interface-name":
["your_management_interface"]
```

2.13.2.1 Limitations

If the IP of one interfaces changes while the service is operational, it will not respond to requests on the new IP address. As a workaround, disable the service before changing the network configuration and re-enable it when the changes are complete.

2.13.3 System Mode Management

A PSME Compute instance, which manages sleds on a drawer, allows for the ability to block the operating system running on a sled from updating the sled's firmware. The sled runs in one of the two modes:

- User Mode, which prevents firmware updates
- Admin Mode, in which firmware updates are allowed

The sled's mode can be modified by sending a `PATCH` request to the Computer System resource representing the sled. Please note that the change will take action only after the system has been rebooted.

2.13.4 Trusted Platform Module Management

RSD sleds include Trusted Platform Module 1.2 (TPM) hardware modules. The PSME Compute provides the administrator with the ability to configure the TPM on a sled. The following actions are possible:

- •Enabling or disabling the TPM
- Clearing TPM ownership, which removes all keys stored in the TPM as well as the owner authorization value.

The actions can be requested by sending a `POST` request to the `ChangeTPMState` action on the Computer System resource representing the sled. Please note that the actual actions will be performed by the BIOS during the next boot.

2.13.4.1 Limitations

Clearing the TPM ownership disables the TPM automatically. Therefore, after a request to Clear and Enable a TPM, and a reboot, the TPM will be disabled. Another request to enable the TPM and another reboot is required to enable the cleared TPM.

2.13.5 Redfish Authentication

This section describes how to setup Redfish Authentication in Intel® RSD PSME software.

2.13.5.1 Introduction

Intel® RSD 2.5 PSME software supports configuring an administrator user account for the software. By default, access to the API requires authenticating using the administrator user's credentials. Two modes of authentication are supported:

- HTTP Basic Authentication - authentication by providing username and password in the request header.
- Redfish session token authentication - sending credentials in a `POST` request to create a session token, then providing that token in request header in subsequent requests.

2.13.5.2 Configuration of the PSME Rest Server

The following sections in the psme-rest-server-configuration.json should be adjusted to configure authentication:

- `"username": "root"` - should be filled with the username for the administrator account
- `"password": "put_password_hash_here"` - should be filled with a hash of the password for the administrator account generated using the encrypt binary. The binary can be compiled by following the instructions in Section, 3.0, PSME Development Environment. To generate the hash for a given password, run encrypt with --hash option:
  ```
  $./encrypt example_password --hash
  Then, copy the output to the configuration file.
  ```
- `"authentication-type": "basic-or-session"` - this flag can be modified to disable one or both modes of authentication. The accepted values are basic, session and none.

Note: It is recommended that authentication not be disabled. none flag should be used for testing purposes only.

2.13.6 Intel® Optane™ DC Persistent Memory Erase

The PSME Compute provides the administrator with an option to erase all data stored in Intel® Optane™ DC Persistent Memory Modules to prevent it from being mishandled or stolen by the subsequent users of the same sled. The erasure can be triggered by sending a `POST` request to the `EraseOptaneDCPersistentMemory` action on the Computer System resource representing the sled.

Note: the `EraseOptaneDCPersistentMemory` action triggers a reboot of the Computer System.

2.13.7 PNC Drive Secure Erase

The PNC agents support erasing user data on disaggregated NVMe drives to prevent the data from being accessed by future users. The data on NVMe drives is removed using NVMe admin command.

Securely erase action can be called using the REST API. Refer to *Intel® RSD PSME REST API specification*, Table 4, for more details.

2.13.8 FPGA Secure Erase

PSME FPGA-oF and PNC agents support overwriting the current Green Bitstream (or AFU) to prevent it from being used by subsequent users. The Green Bitstream is overwritten with a default Green Bitstream specified by the user in the configuration file:

```
"secureEraseGBS" : "/etc/opae/default_afu.gbs"
```

It is necessary for the user to provide a path to a valid file. Bear in mind that the Green Bitstreams or AFUs are strictly coupled with the programmed Blue Bits version. Thus, it is recommended to use one of the sample Green Bits packaged with the Intel Acceleration Stack (IAS) release - e.g., nlb_mode_0.gbs, nlb_mode_3.gbs, dma_afu.gbs, hello_afu.gbs, etc.

The IAS can be downloaded from the Intel FPGA Acceleration Hub:
https://www.intel.com/content/www/us/en/programmable/solutions/acceleration-hub/downloads.html.

To trigger the override, the `SecureErase` action should be called on the Processor resource representing the FPGA. Refer to the *Intel® RSD PSME REST API specification*, Table 4, for more details.

2.14 Log Service

Log Service is a service presenting log entries from various software and hardware components. The way that entries are gathered and presented depends on the log type. Gathered entries are exposed on the `REST` API.

2.14.1 Compute Log Service

In Intel® RSD 2.5 the log service functionality is realized by the PSME Compute agent.

The PSME collects data reported in System Event Log (SEL) by the BMC and exposes in the form of Redfish Log Entries.

For more information about the format of a SEL record refer to the *Intelligent Platform Management Interface Specification*, refer to Table 4. The PSME represents the whole SEL as it is returned by the BMC (unrecognized/malformed records are skipped).

2.14.2 Limitations:

- log entries cannot be modified or deleted,
- performance drop might occur with many log entries,
- max number of entries is estimated, not precise (because record size is varying).

§

3.0 PSME Development Environment

The PSME software depends on several libraries and specific OS settings. This section describes precise software versions required to compile and run the PSME software.

3.1 Requirements

The PSME software was developed in `C++14` language targeting Linux based systems. The whole build process is managed by the `CMake` tool. Table 11 shows the software versions required to compile the PSME on Linux based systems.

Table 11. Software Versions to Compile the PSME on Linux

Software	Version
CMake	≥ 3.4.3
gcc	5.3.1 ≤ ≤ 5.4.0
`PATCH`	Default repository version

The libraries shown in Table 12 must be installed prior to the PSME compilation.

Table 12. Libraries to Install Prior to PSME Compilation

Software	Version
curl	Default repository version
microhttpd	Default repository version
LVM2	Default repository version

If any of the libraries are missing, the `CMake` script attempts to download the source package from public software repositories on the Internet and automatically compile it. Confirm that the server network, firewall, and proxy configurations allow the appropriate server access to external software vendor sites.

3.1.1 Ubuntu* v16.04 LTS

Enter the command provided in this section to install all Ubuntu packages.

Note: This command requires root privileges.

```
apt install cmake clang gcc-5 g++-5 libgcrypt20-dev libncurses5-dev \
libnl-3-dev libudev-dev libglibmm-2.4-dev libglib3.0-cil-dev libxml++2.6-dev \
libgnutls-dev libnl-route-3-dev flex bison valgrind doxygen cpp ccache \
build-essential linux-libc-dev libmpc-dev libstdc++6 libcurl4-openssl-dev \
libmicrohttpd-dev lcov libnl-route-3-200 \
libsysfs-dev libpopt-dev libusb-dev patch \
libdevmapper-dev liblvm2-dev unzip libnl-genl-3-dev libblkid-dev debsigs \
debsig-verify gnupg libusb-1.0-0-dev libibverbs-dev librdmacm-dev \
libjson0-dev uuid-dev
```

Download Aardvark* Software API from *https://www.totalphase.com/products/aardvark-software-api/aardvark-api-linux-x86_64-v5.30.zip* and copy it into `RSA/psme/SW/third_party` directory.

Ubuntu v16.04 LTS package repository does not provide necessary `libfabric` version, so download and install `libfabric` from sources:

```
wget https://github.com/ofiwg/libfabric/releases/download/v1.7.0/libfabric-
1.7.0.tar.gz
tar xfv libfabric-1.7.0.tar.gz
```

```
cd libfabric-1.7.0
./configure
make
sudo make install
```

3.1.2 CentOS* v7.3

CentOS* v7.3 has been chosen as a development environment for PSME components running on Arista EOS. This OS comes with long term support and is similar to the Fedora* system, which EOS is based on.

The following command should be used to install all the required packages:

```
sudo yum -y install patch unzip iproute which git openssl wget autogen \
libtool systemd-devel libnl3-devel lvm2-devel glibc.i686 zlib.i686
```

Because CentOS 7* comes with an old version of CMake, it is required to install a newer version manually. Perform the following installation steps:

```
wget https://cmake.org/files/v3.8/cmake-3.8.1-Linux-x86_64.sh
chmod 755 cmake-3.8.1-Linux-x86_64.sh
sudo mkdir -p /opt/cmake
sudo ./cmake-3.8.1-Linux-x86_64.sh --prefix=/opt/cmake --exclude-subdir
```

PSME binaries targeted for Arista EOS need to be compiled with a cross compiler provided by Arista. To obtain it, create a user account at the official Arista website. After you log in, navigate to `Support > Software Download`, and download `arista-fc18-gcc4.9.2.rpm`. Afterwards, install the cross compiler package:

```
rpm -i arista-fc18-gcc4.9.2rpm
```

3.2 Compilation

Decompress the PSME source code package and download all external dependencies to the "`third_party`" folder:

```
cd third_party
./download.sh
cd ..
```

One dependency needs to be downloaded manually. To obtain it, create a user account at the official Total Phase* website, naviage to https://www.totalphase.com. When you're able to log in, navigate to `https://www.totalphase.com/products/aardvark-software-api`, download the Aardvark* Software API v5.30 (Linux 64-bit) and put it in the `third_party` folder.

All PSME modules must be built from the main directory using `make [target]`. First, prepare the build directory using `CMake`. Creating a build directory with `CMake` is a one-time operation.

To build a release version:

```
mkdir build.release
cd build.release
cmake ..
```

To build a debug version:

```
mkdir build.debug
cd build.debug
cmake -DCMAKE_BUILD_TYPE=Debug ..
```

`cmake` enables you to pass additional parameters, like target architecture, compiler, and many others. For more information, refer to `man cmake`.

Note: All PSME modules must be built from the previously prepared build directory (`build.debug` or `build.release`).

Perform the following steps to build the PSME Rest Server, all associated agents, stubs, and simulators:

```
cd <PSME root>/<build directory>
make all -j8
```

Note: Specify the number of parallel jobs `n` for faster compilation using the `j` flag.

To build only a subset of modules, for example only `psme-rest-server` and `psme-chassis`, use the following alternative command:

```
make psme-rest-server psme-chassis -j8
```

To get a list of all possible targets to build, run the following command in the build directory:

```
make -qp | awk -F':' '/^[a-zA-Z0-9][^$#\/\t=]*:([^=]|$)/ {split($1,A,/ /);for(i in
A)print A[i]}'
```

To run unit testing (all build types):

```
ctest
```

To generate documentation:

```
make doc-generate
```

To read documentation:

```
YOUR_WEB_BROWSER doc/html/index.html
```

3.3 Compilation Process for Arista EOS

Before starting the compilation, use the following command to point to the correct version of CMake and compiler:

```
export PATH=/opt/arista/fc18-gcc4.9.2/bin:/opt/cmake/bin:$PATH
```

To build PSME binaries for 32-bit EOS system, use the following command:

```
./build_main.sh -b release -a 32 -c gcc -t psme-rest-server,psme-network
```

§

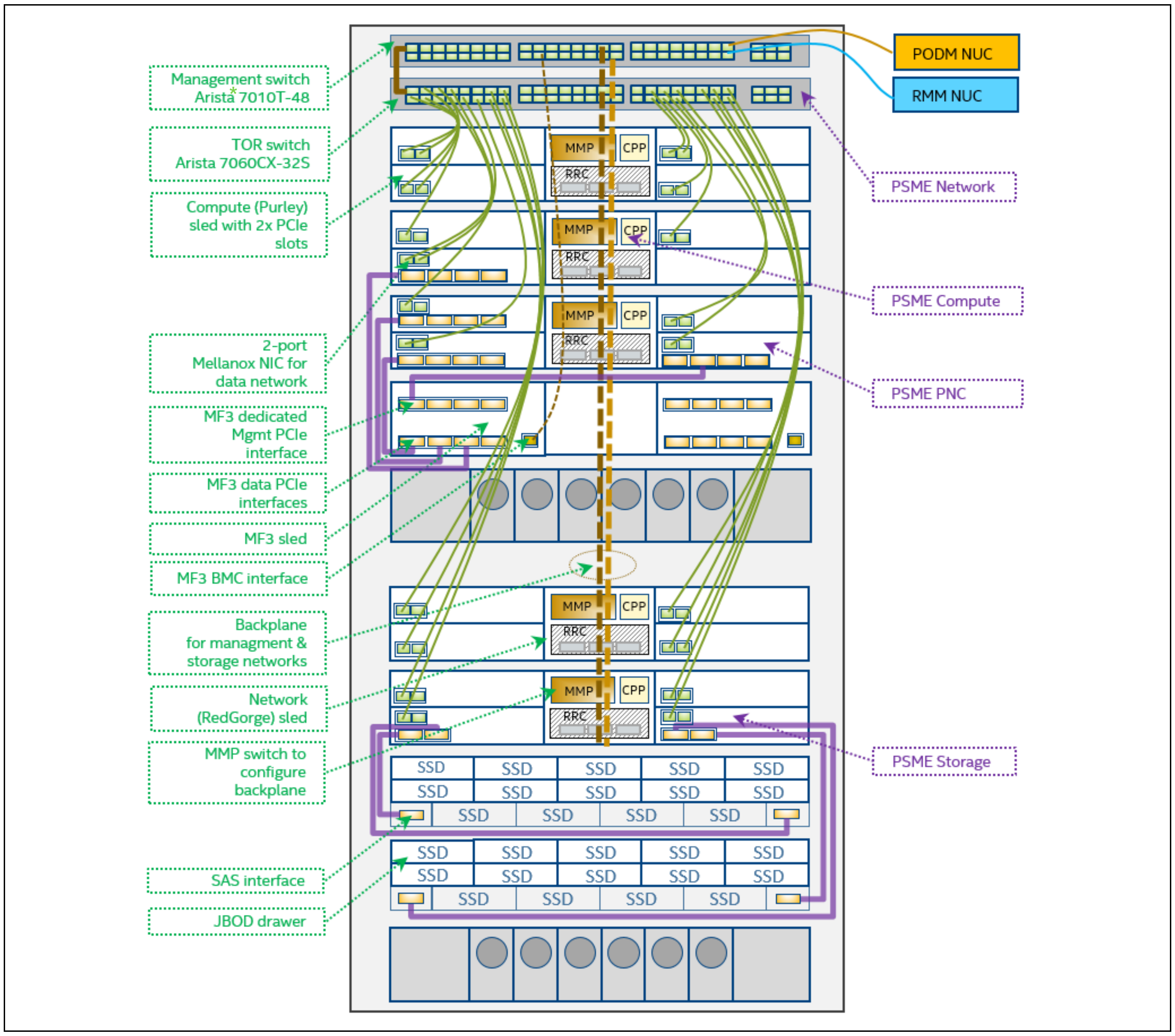
4.0 Intel® RSD Rack Network Configuration

This section is specific to the reference design of the Intel® RSD Software Development Vehicle platform networking. The Intel® RSD Software Development Vehicle platform top-of-rack (ToR) switch configuration is included in the appendices.

4.1.1 Intel® RSD Software Development Vehicle Physical Layout

Figure 4 shows the physical layout of the Intel® RSD Software Development Vehicle Reference Platform.

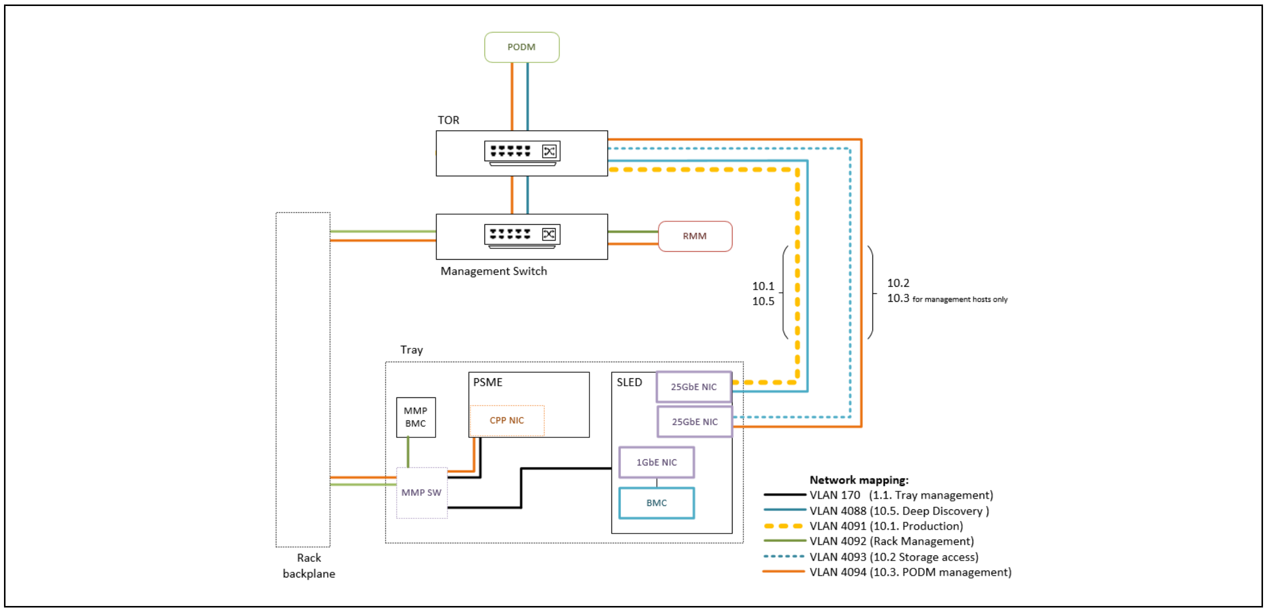
Figure 4. Intel® RSD Software Development Vehicle Reference Platform


Management switch Arista* 7010T-48
TOR switch Arista 7060CX-32S
Compute (Purley) sled with 2x PCIe slots
2-port Mellanox NIC for data network
MF3 dedicated Mgmt PCIe interface
MF3 data PCIe interfaces
MF3 sled
MF3 BMC interface
Backplane for managment & storage networks
Network (RedGorge) sled
MMP switch to configure backplane
SAS interface
JBOD drawer
PODM NUC
RMM NUC
PSME Network
PSME Compute
PSME PNC
PSME Storage
MMP
CPP
RRC
SSD

4.1.2 Intel® RSD Software Development Vehicle Network Topology

Figure 5 shows the Network Topology of the Intel® RSD Software Development Vehicle Platform.

Figure 5. Network Topology of Intel® RSD Software Development Vehicle Platform


PODM
TOR
RMM
Management Switch
10.1
10.5
10.2
10.3 for management hosts only
Tray
PSME
CPP NIC
MMP BMC
MMP SW
SLED
25GbE NIC
25GbE NIC
1GbE NIC
BMC
Rack backplane
Network mapping:
VLAN 170 (1.1. Tray management)
VLAN 4088 (10.5. Deep Discovery)
VLAN 4091 (10.1. Production)
VLAN 4092 (Rack Management)
VLAN 4093 (10.2 Storage access)
VLAN 4094 (10.3. PODM management)

4.1.3 Rack Switches

The following sections describe the VLANs configured on switches in an Intel® RSD Software Development Vehicle Rack.

4.1.3.1 ToR Switch VLAN

Figure 6 and Figure 7 show ToR switch VLAN configurations.

Figure 6. ToR Switch VLAN Configuration

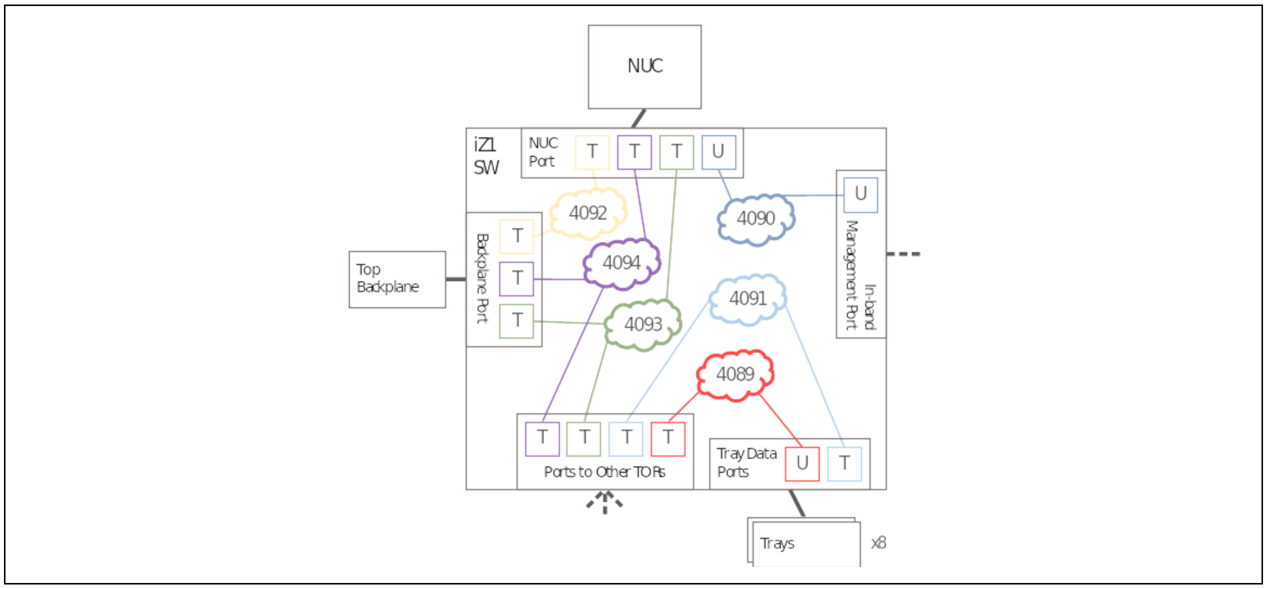
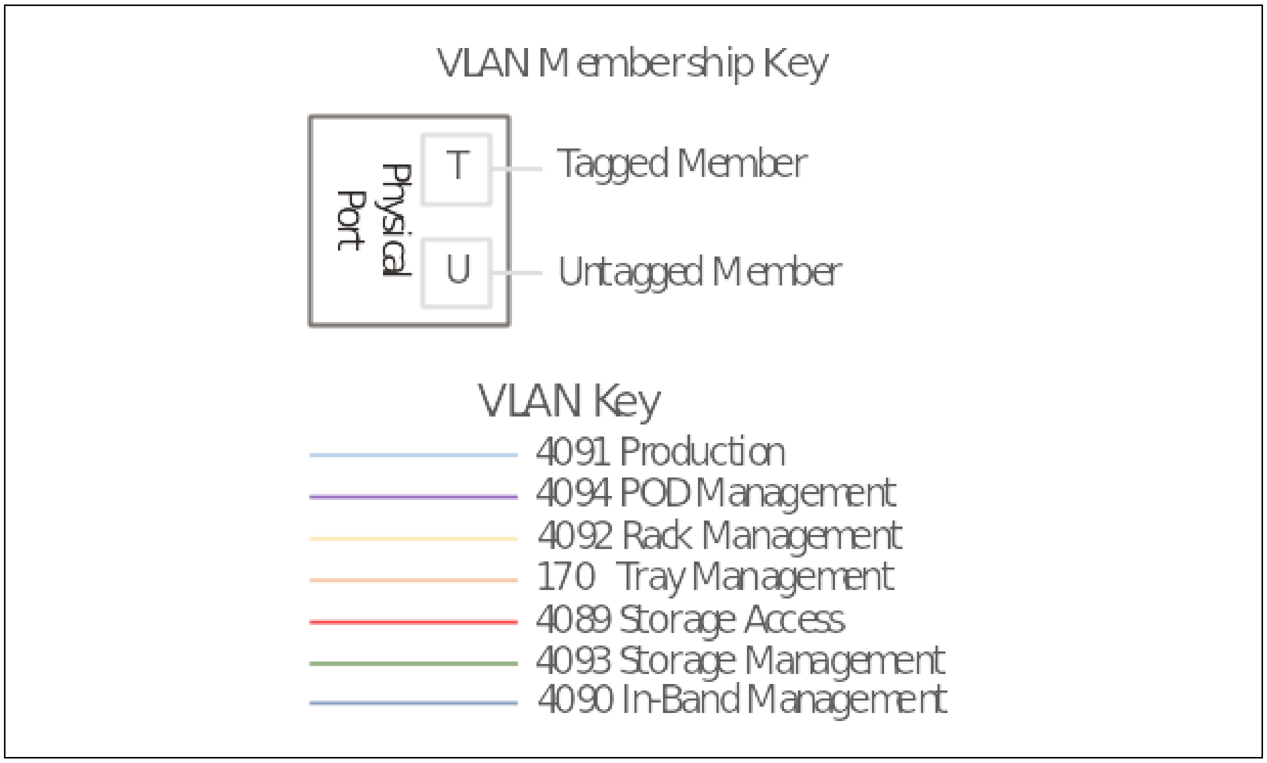


Figure 7. ToR Switch VLAN Configuration VLANs



4.1.3.2 MBP VLAN

Figure 8 and Figure 9 show the MBP VLAN configurations.

Figure 8. MBP VLAN Configuration

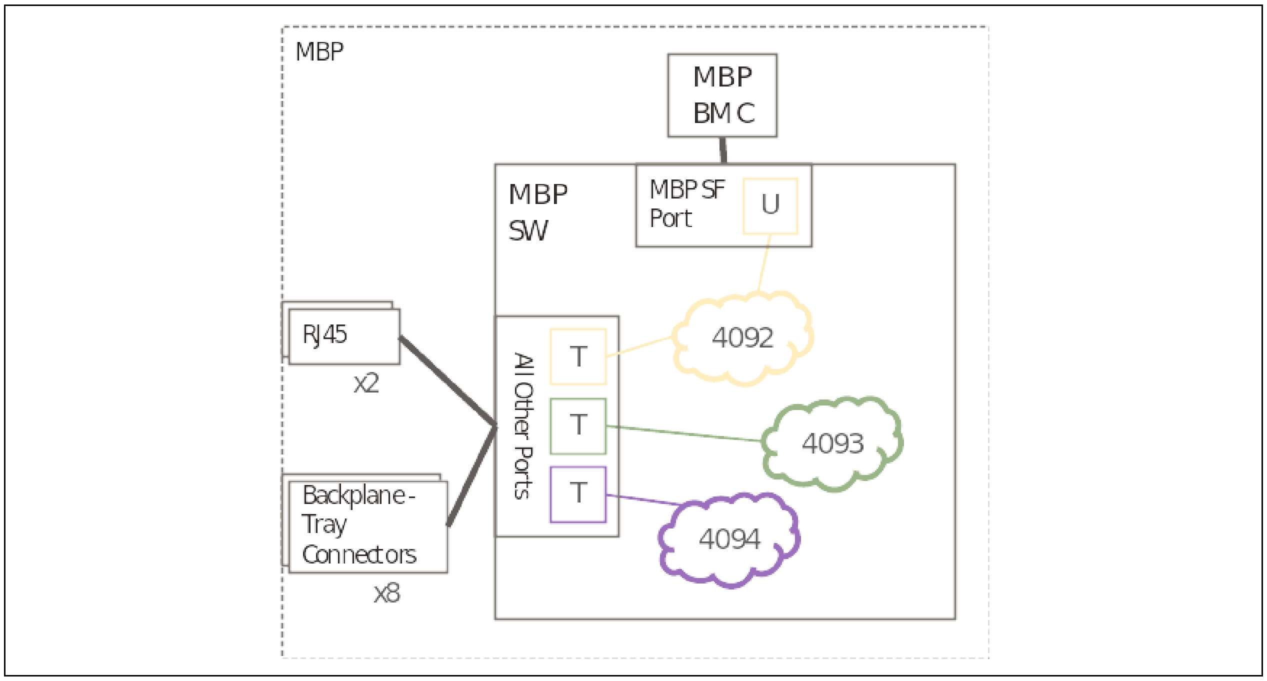
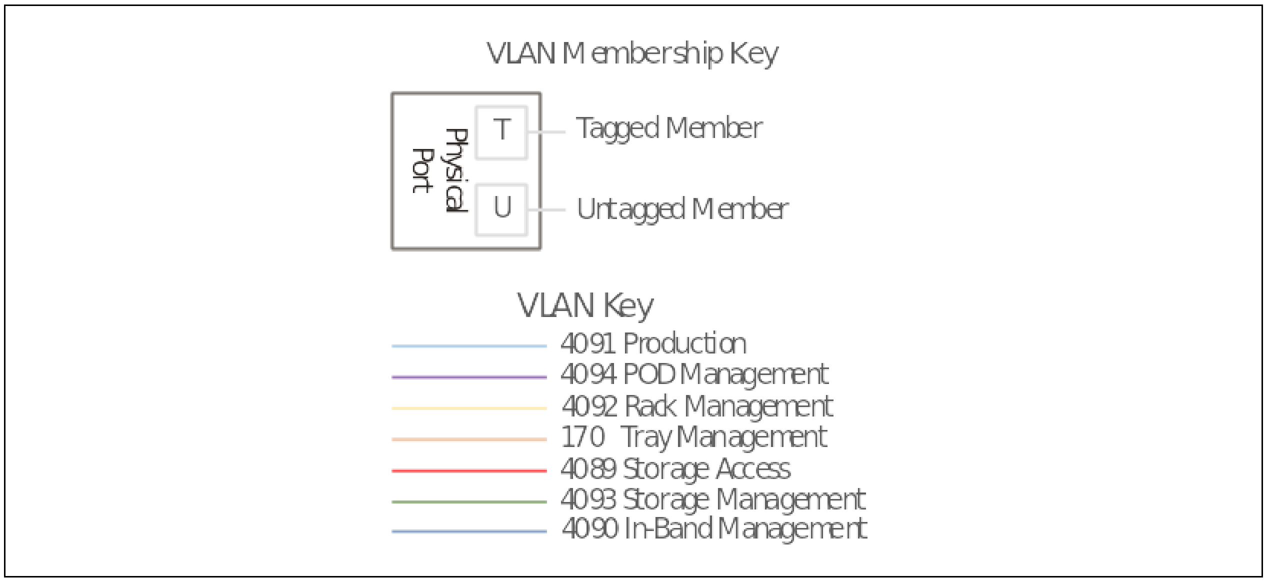


Figure 9. MBP VLAN Configuration VLANs



4.1.4 Intel® RSD Software Development Vehicle Drawer Network

This section describes the topology of the network inside an Intel® RSD Software Development Vehicle drawer.

4.1.4.1 MMP VLAN

Figure 10 and Figure 11 show the MMP VLAN configurations.

Figure 10. MMP VLAN Configuration

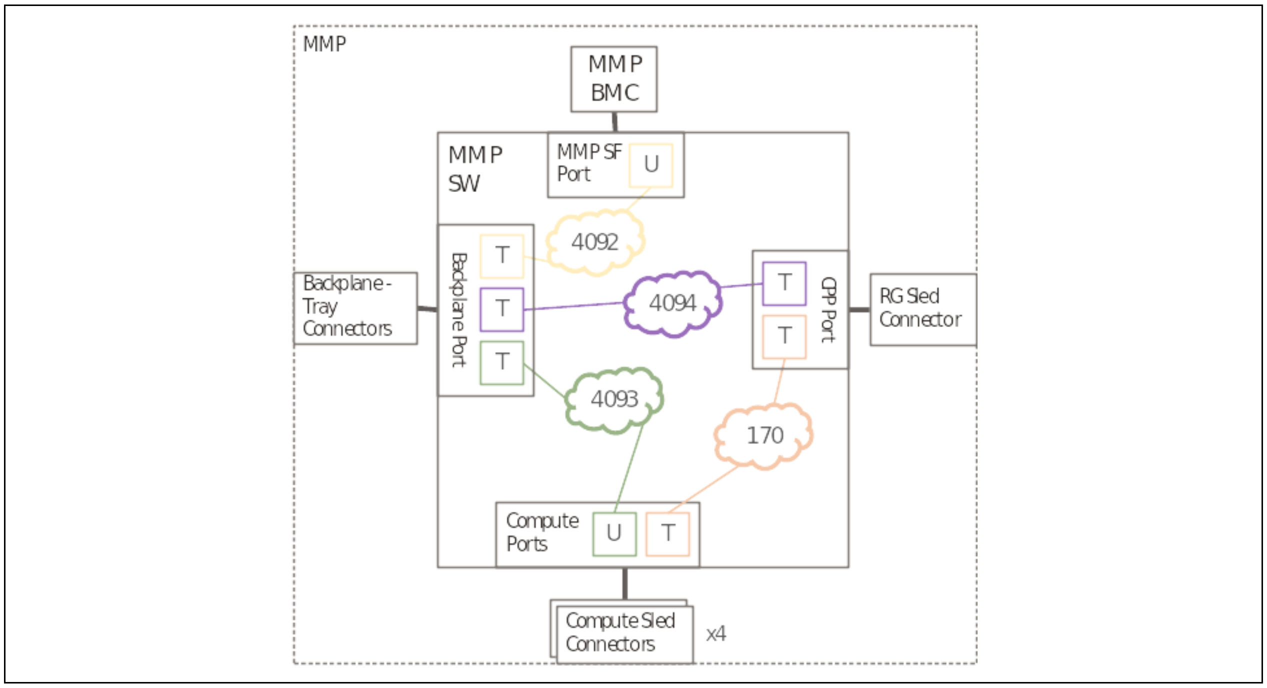
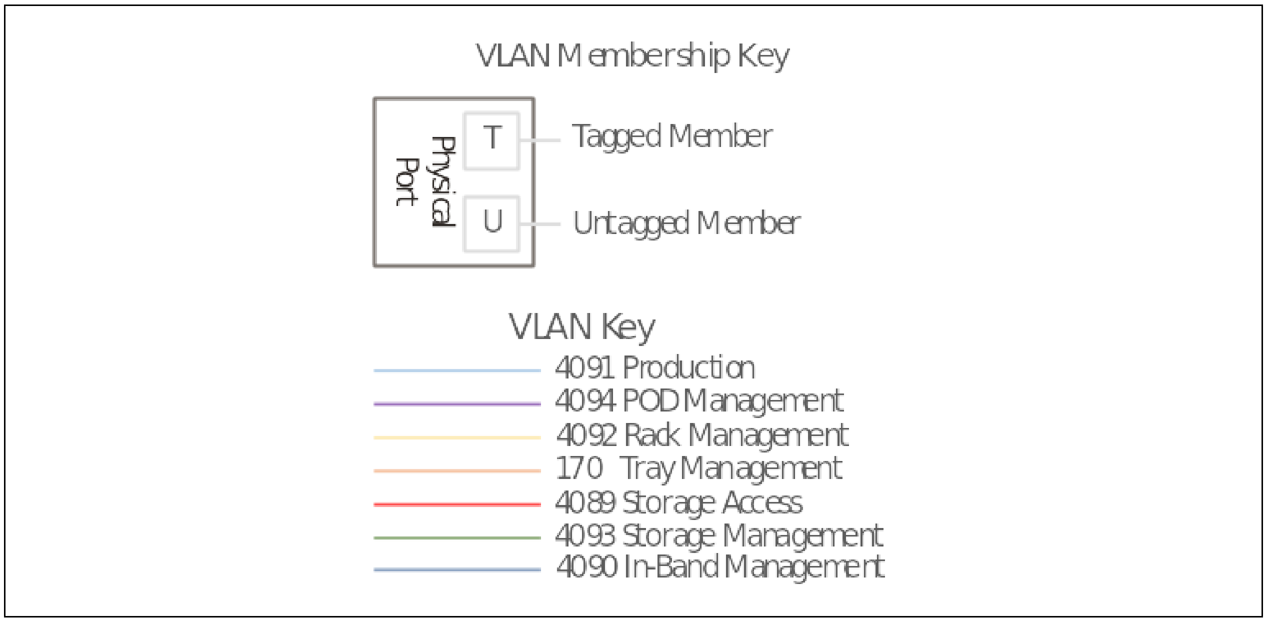


Figure 11. MMP VLAN Configuration VLANs



§

5.0 Intel® RSD Drawer Configuration

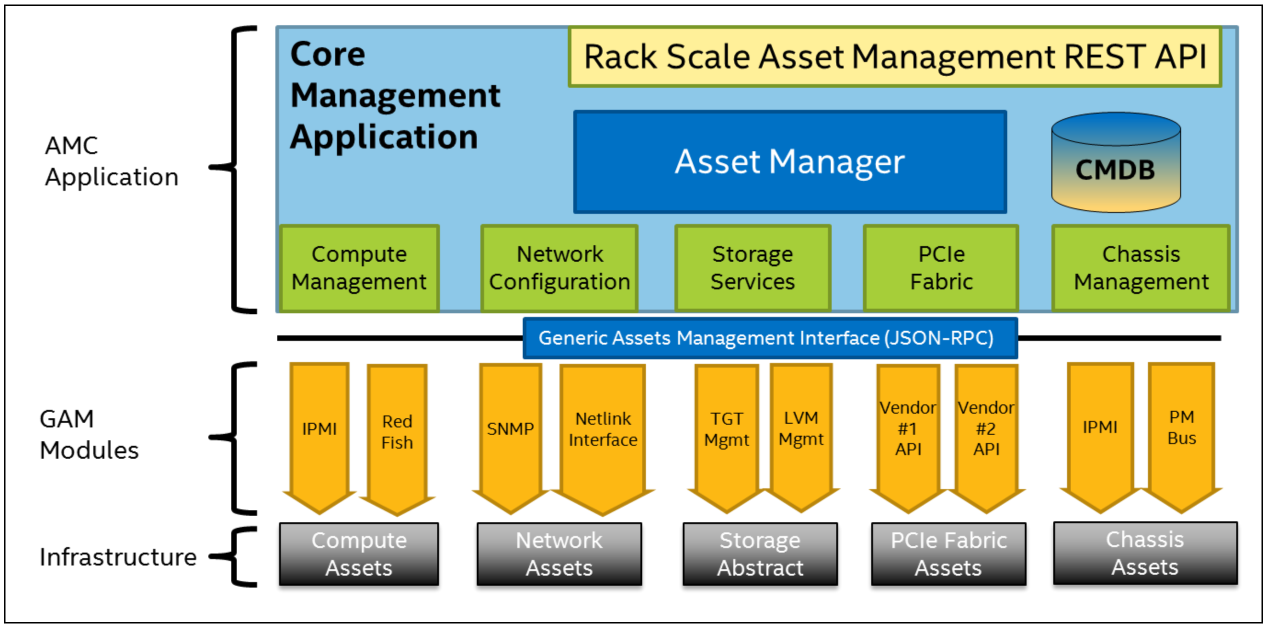
5.1 Hardware Configuration

This section is specific to the reference design of the Intel® RSD Software Development Vehicle platform hardware configuration. The Intel® RSD Software Development Vehicle platform hardware configuration is described in the appendices.

5.2 PSME Base Software

The PSME software consists of two software layers, PSME Rest Server and Generic Asset Management Modules. For the Intel® RSD Software Development Vehicle platforms, the user must run the PSME Rest Server, PSME Compute, and PSME Chassis on each Drawer in Rack. Figure 12 shows the main software components layout.

Figure 12. PSME Software Components



5.2.1 Running the PSME Components

Each PSME component can be executed by the `root` user from any local directory, or optionally run from a non-root account if not binding to a privileged port.

The component should be executed with a configuration file passed as a command line argument. Use the following command pattern to run executable:

```
sudo ./<executable name> <path to configuration>/<configuration file name>.json
```

The following shows an example of the PSME REST server run:

```
sudo ./psme-rest-server ./rest-server-configuration.json
```

The executables are located in the `<PSME root>/build/bin` directory, as shown in Table 13.

Table 13. PSME Executables in Build Output Directory

Name	Executable name
PSME REST Server	`psme-rest-server`
PSME Chassis Agent	`psme-chassis`
PSME Compute Agent	`psme-compute`
PSME Network Agent	`psme-network`
PSME iSCSI Storage Agent	`psme-lvm-iscsi`
PSME PNC Agent	`psme-pnc`
PSME RMM Agent	`psme-rmm`
PSME GPT NVMe Agent	`psme-gpt-nvme`
PSME SPDK NVMe Agent	`psme-spdk-nvme`
PSME NVMe Discovery agent	`psme-nvme-discovery`
PSME FPGA-oF agent	`psme-fpgaof`
PSME FPGA-oF Discovery agent	`psme-fpga-discovery`

5.2.2 PSME Configuration File

The configuration file details and property descriptions can be found in a configuration schema file as a part of the source package. Schema file names are `configuration_schema.json`. Table 14 shows the location of the PSME module configuration files.

Table 14. PSME Software Configuration Files

Module	Configuration File
PSME REST Server	`<PSME root>/application/configuration.json`
PSME Chassis Agent	`<PSME root>/agent/chassis/configuration.json`
PSME Compute Agent	`<PSME root>/agent/compute/configuration.json`
PSME Network Agent	`<PSME root>/agent/network/configuration.json`
PSME iSCSI Storage Agent	`<PSME root>/agent/storage/configuration.json`
PSME PNC Agent	`<PSME root>/agent/pnc/configuration.json`
PSME RMM Agent	`<PSME root>/agent/rmm/configuration.json`
PSME GPT NVMe Agent	`<PSME root>/agent/nvme/configuration.json`
PSME SPDK NVMe Agent	`<PSME root>/agent/spdk/configuration.json`
PSME NVMe Discovery Agent	`<PSME root>/agent/nvme-discovery/configuration.json`
PSME FPGA-oF Agent	`<PSME root>/agent/fpga-of/configuration.json`
PSME FPGA-oF Discovery Agent	`<PSME root>/agent/fpga-discovery/configuration.json`

5.3 PSME Storage Services

5.3.1 Prerequisites

A SLED with Linux OS, Ubuntu* v16.04 OS is recommended.

5.3.2 Configuration

Configure the OS by enabling and connecting to iSCSI targets and creating a Logical Volume Management (LVM) structure.

5.3.2.1 Connecting to iSCSI Targets

Set the following variable:

```
"portal-interface" : "interface_for_connection_to_targets"
```

in the `/etc/psme/psme-storage-configuration.json` file to a network interface, which is used to connect to iSCSI targets.

5.3.2.2 Creation of LVM structure

This procedure outlines the steps required to create the LVM structure.

1. For each disk `sdX` (except the system disk), create a `PhysicalVolume`:

Caution: The following instruction will remove all existing data from the disk using the following code:

```
dd if=/dev/zero of=/dev/sdX bs=512 count=1 && hdparm -z /dev/sdX && pvcreate
/dev/sdX
```

2. Create one `VolumeGroup` providing its name (such as `main_volume_group`) as the first parameter and one of the `PhysicalVolumes` (such as `/dev/sdY`) as the second:

```
vgcreate main_volume_group /dev/sdY
```

3. Add other `PhysicalVolumes` to this `VolumeGroup`:

```
vgextend main_volume_group /dev/sdX
```

4. Create a `LogicalVolume` by providing its size, name, and `VolumeGroup`:

```
lvcreate -L 10G -n base_logical_volume main_volume_group
```

5. Copy the OS image to `LogicalVolume`:

```
dd if=your_image.raw of=/dev/main_volume_group/base_logical_volume
```

6. If needed, the `LogicalVolume` can be made read-only:

```
lvchange -pr /dev/main_volume_group/base_logical_volume
```

7. Restart the `psme-rest-server` and `psme-storage` services to discover the new LVM structure.

5.3.2.3 Bootable Attribute of LogicalVolume

The `"bootable"` attribute of `LogicalVolume` is set in LVM tags. It can be set using `PATCH` requests on the logical drive or set manually. To manually make a given LV bootable, a `bootable` tag has to be added, as shown in these steps:

1. Us the following to add a bootable tag to a given `LogicalVolume:`

```
lvchange --addtag @bootable /dev/main_volume_group/base_logical_volume
```

2. Restart the `psme-rest-server` and `psme-storage services` to discover changes in the LVM tags.

To remove a `bootable` tag:

1. To remove a `bootable` tag from a given `LogicalVolume use`:

```
lvchange --deltag @bootable /dev/main_volume_group/base_logical_volume
```

2. Restart the `psme-rest-server` and `psme-storage` services to discover changes in the LVM tags.

   LVM tags can be displayed using the command:

```
lvs -o lv_name,lv_tags
```

   While creating a new LVM using PSME Storage Service, the bootable attribute should be specified in the `POST` request (LVM tags are added automatically).

5.3.2.4 Manual Creation of iSCSI Target from Volume

Starting in Intel® RSD v2.5, it is neither required nor recommended to create iSCSI Targets manually. They should be created using the `psme-rest-server` REST API or `psme-storage` GAMI API.

During initialization, the Storage Agent will try to automatically restore iSCSI Targets that were created via PSME and delete those that were not.

5.3.3 Limitations

- An Initiator Endpoint can have only one connected entity with a role initiator.
- A target Endpoint must have at least one or multiple connected entities with a role target.
- All usernames of Endpoints and iQN Identifiers of target Endpoints must be unique within one Storage Service.
- Target Endpoint connected entity links must be unique within one Storage Service.
- Initiator endpoint connected entity links and access modes must be null.
- An Endpoint cannot be deleted when it is in use (belongs to a zone).
- An Endpoint can belong only to one zone at a time.
- A zone can be deleted regardless of its content.
- A zone can have only one initiator endpoint, but multiple target Endpoints.
- A zone represents an iSCSI target in the tgt daemon, which contains one initiator Endpoint and at least one target Endpoint.

5.3.4 Known Issues

- To create more than 34 iSCSI Targets (for instance, assemble total more than 34 nodes with remote storage using the same Storage Services) in Ubuntu v16.04, the tasks limit for tgt service must be modified, as follows:
```
sed -i "/\[Service\]/a TasksMax=infinity" /lib/systemd/system/tgt.service
systemctl daemon-reload
service tgt restart
```
- Target Endpoints `EntityAccessModes` are stubbed and not synchronized with iSCSI targets. Volumes `AccessCapabilities` are read from LVM logical volumes during `psme-storage` initialization and cannot be set via PSME. New volumes and iSCSI targets created via PSME have by default read/write access capabilities.

5.4 PSME Chassis

This section provides the prerequisites and instructions on how to deploy the PSME Chassis Agent.

5.4.1 Prerequisites

A PC with Ubuntu* Linux* v16.04 OS is recommended.

5.4.2 Running the PSME Chassis Agent

Before starting the PSME Chassis, packages mentioned as default for Ubuntu* development environment must be installed. Next, install `CyMUX` by following the instructions in Appendix G.2, Installing CyMUX.

Build the Chassis Agent, run it, and wait while the Chassis components are being discovered. Watch the progress in `journalctl`.

The PSME Chassis also requires the encrypted binary. The encrypt binary can be compiled by following the instructions in Section 3.0, PSME Development Environment. To use the binary, create a protected directory `/etc/psme.`

```
    $sudo mkdir -p /etc/psme
    $sudo chmod 644 /etc/psme
```

Then, use the encrypt binary to encrypt the credentials to the BMCs that the PSME Chassis is to communicate with.

```
    $sudo encrypt <BMC_username>
    $sudo encrypt <BMC_password>
```

Place the encrypted credentials in the PSME Chassis service configuration section for a particular BMC:

```
{
    "ipv4" : "<bmc_IP_address>",
    "username" : "<encrypted_username>",
    "password" : "<encrypted_password>"
}
```

5.5 PSME Pooled NVMe Controller

This section provides details specific to the Intel® RSD Intel® Software Development Vehicle.

5.5.1 Prerequisites

As a prerequisite to using the PSME PNC, you should have a setup with the Intel® RSD Software Development Vehicle PCIe Switch with attached NVMe drives. The management host should have a Linux kernel version supporting PCIe device hot swap.

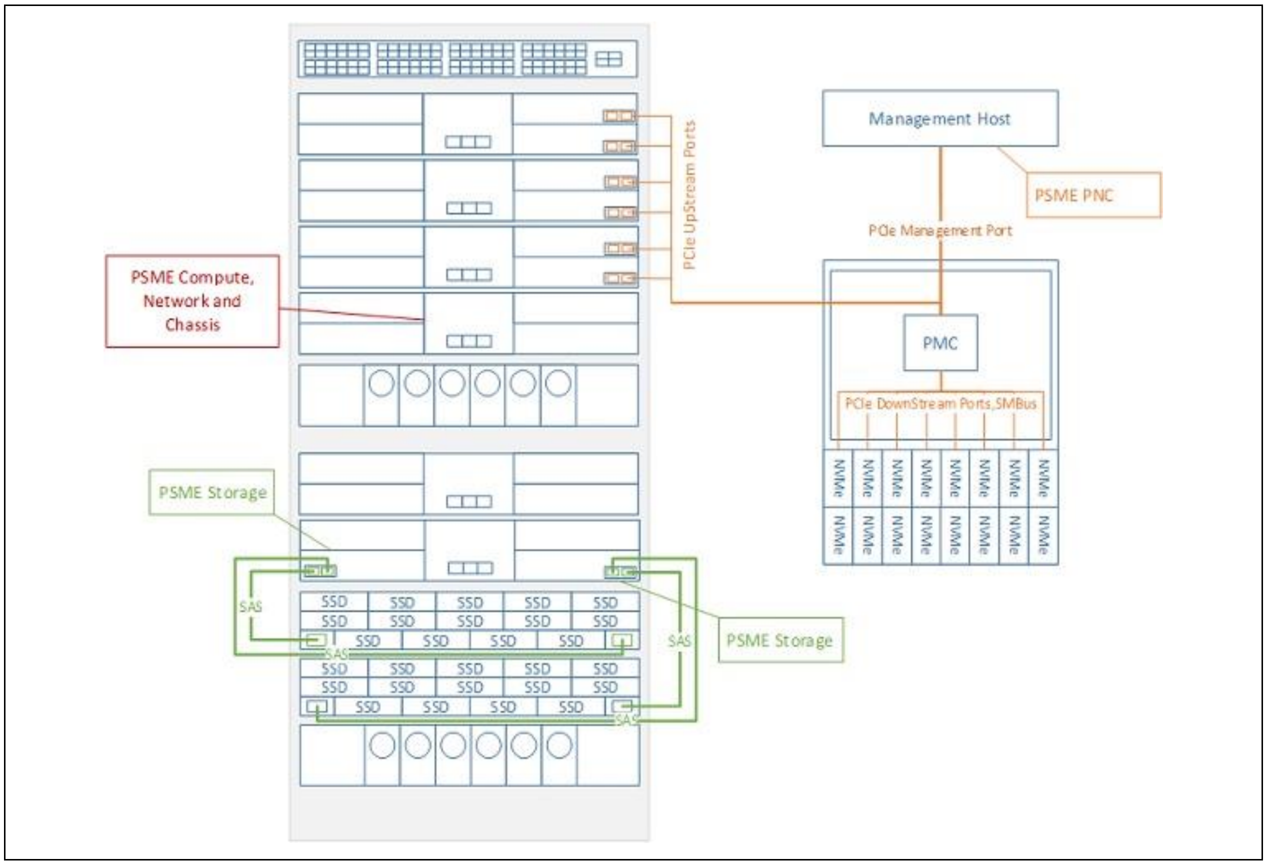
Note: If you are using the Intel® RSD Software Development Vehicle, contact your Intel representative to receive detailed setup configuration instructions.

5.5.2 Hardware Configuration

The PSME PNC consists of two basic hardware components—PCIe switch board and management host. In the Intel® RAD Software Development Vehicle, the management host is a SLED connected to PCIe upstream port 24 of the PCIe switch board. The remaining upstream PCIe ports are connected to compute SLEDs. The SLEDs and management host must have a PCIe retimer add-in card connected. NVMe SSD drives are connected to PCIe downstream ports. Figure 13 shows the PSME PNC hardware configuration.

The PCIe switch boards need to enable PLDM (over I2C) communication with the PCIe slots - it is necessary for Out of band discovery of the FPGAs.

Figure 13. PSME Pooled NVMe Controller Hardware Configuration



5.5.3 Installation

This section provides information about the installing Ubuntu v16.04 Server OS for the PNC management host, connecting the NVM SSD drives, and keeping firmware updated.

5.5.3.1 Ubuntu v16.04

Ubuntu v16.04 Server is the recommended OS for PSME PNC management host. Follow the installation steps described in the install guide available on the Ubuntu home page.

To make the PSME PNC visible for Intel® RSD PODM in DHCP discovery mode (not SSDP), change the OS hostname to begin with `psme` (it must be compatible with regular expression `^psme.*`, for example: `psme, psmeXYZ` or `psme-XYZ`) and enable getting the IP from DHCP for the management interface.

5.5.3.2 NVMe SSD Drive

NVMe SSD drives have to be connected to PCIe switch board downstream ports. The PSME PNC uses SMBus devices exposed by drives to grab FRU information and monitor the drives' status.

5.5.3.2.1 Firmware Update

The NVMe SSD drive firmware must be kept up to date. Some drives may be flashed with old firmware, which does not have full support for NVM Express Basic Management functionality. To update the firmware:

1. Make sure all connected drives are present on the management host OS:

```
     sudo lspci | grep Volatile
03:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (rev
01)
04:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (rev
01)
05:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (rev
01)
06:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (rev
01)
07:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (rev
01)
08:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (rev
01)
09:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (rev
01)
0a:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (rev
01)
```

2. Download the Intel® SSD Data Center Tool dedicated for a drive model connected to PCIe switch. The tool is available for download at the *https://downloadcenter.intel.com*.

Note: The tool is not available in Debian* (.deb*) format, but it may be converted using `alien-pkg-convert` software. Refer to the *https://help.ubuntu.com/community/RPM/AlienHowto*.

3. When the .deb package is ready, install it:

```
dpkg -i isdct-*.deb
```

4. List all connected drives:

```
sudo isdct show -a -intelssd
```

5. Load the new firmware:

```
isdct load -intelssd <drive index>
```

5.5.3.3 Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA (Intel® PAC with Intel® Arria® 10 GX FPGA)

Intel® Programmable Acceleration Card (PAC) cards come with SW and FW stack provided by Intel, called Intel Acceleration Stack (IAS). The IAS suite provides tools for firmware updates, sample green bits (AFUs) and OpenCL runtime. For details on the cards FW update, please refer to the IAS documentation or contact Intel representative.

5.5.4 Known issues

If service or host is restarted while target endpoint pointing to PCIe device (an FPGA or a Drive) is in a zone, the `PCIeDevice` resource associated with the device will not be available until that target endpoint is removed from the zone.

5.6 PSME SPDK NVMe Storage Service

This section provides the prerequisites and instructions on how to deploy the PSME SPDK NVMe Storage Service.

5.6.1 Prerequisites

SLED with Linux OS, Ubuntu v16.04 recommended

5.6.2 Configuration

Configure the SPDK NVMe-oF daemon by running application `(nvmf_tgt`) and creating an SPDK structure.

Note: SPDK installation instructions can be found in Appendix E.2, Step-by-step Installation Instruction.

5.6.2.1 PSME SPDK agent configuration file

PSME SPDK agent's configuration file can be found `in /etc/psme/psme-spdk-nvme-configuration.json` file.

The option `nic-drivers` in the file contains the list of Linux network drivers which will be used to gather a list of network interfaces that are on a given host. Only the interfaces that were created by one of the listed drivers are exposed on the API.

If an interface is missing from the API, then run the following command to determine the driver name of a network interface:

```
ethtool -i <interface_name>
```

The user should then add the driver name to the list and restart the PSME services.

SPDK implements a JSON-RPC 2.0 server that allows for configuration of SPDK components in runtime. The JSON-RPC server listens on UNIX socket which should be set in PSME configuration file in the `spdk-socket` field. The default value is set to the default SPDK socket address `/var/tmp/spdk.sock`.

PSME performs polling of SPDK resources to be up to date with hardware changes. To change the interval of the polling set the `spdk-polling-interval-sec`.

NVMe-oF RDMA service port number can be set in the service-port field.

5.6.2.2 Setup SPDK daemon for RSD Usage

1. Make sure that the `nvmf_tgt` application is running and the console prompt is set to spdk repository directory (spdk by default).
2. To claim NVMe drives from kernel to SPDK user-space run the following command for requested block devices:

```
    sudo ./scripts/rpc.py construct_nvme_bdev -b <bdev-name> -t pcie -a <pci-
address>
```

Where

- `bdev-name` - the name of NVMe controller, assigned by the user. The controller name should contain only alphanumeric ASCII characters with `'-'` and `'_'` characters.
- `pci-address` - PCI address of the device. All devices' addresses are printed to console during installation when `setup.sh` is run (refer to Appendix E, SPDK Installation from Sources).

Example:

```
sudo ./scripts/rpc.py construct_nvme_bdev -b NVMe2 -t pcie -a 0000:3e:00.0
```

All SPDK block devices can be displayed by running:

```
sudo ./scripts/rpc.py get_bdevs
```

3. The Logical Volume Stores are not managed by PSME so the user has to create one store per each drive:

```
    sudo ./scripts/rpc.py construct_lvol_store <bdev-name> <lvs-name>
```

Where

- `bdev-name` - name of NVMe controller from the previous step.
- `lvs-name` - name for the logical volume store, assigned by the user. The logical volume store name should contains only alphanumeric ASCII characters with `'-'` and `'_'` characters.

To print all available Logical Volume Stores run:

```
    sudo ./scripts/rpc.py get_lvol_stores
```

4. Create RDMA transport layer:

```
    sudo ./scripts/rpc.py nvmf_create_transport -t rdma
```

5. Now, PSME agent can be started and it should discover SPDK drives and storage pools structure.

5.6.3 Limitations

- On startup, the PSME SPDK NVMe agent will not restore previously created NVMe targets.
- Volume snapshot cannot be created from other snapshots.
- Volume clones can be created only from snapshots.
- Removed snapshots are hidden in `REST` API until all its clones are removed.
- An Endpoint can have only one Connected Entity with a Role Initiator or Target.
- Target Endpoint Connected Entity Links must be unique within one Storage Service.
- An Endpoint cannot be deleted when it is in use (belongs to a Zone).
- An Endpoint can belong only to one Zone at a time.
- A Zone can have only one Initiator Endpoint, but multiple Target Endpoints.

5.7 PSME FPGA-over Fabric

This section provides details specific to the Intel® Rack Scale Design Software Development Vehicle.

5.7.1 Prerequisites

- SLED with Linux OS, Ubuntu 16.04 recommended
- Open Programmable Acceleration Engine (OPAE) driver, v1.3 or above

5.7.1.1 FPGA

FPGA devices have to be connected to host board downstream PCIe ports.

5.7.1.2 PSME FPGA-oF Agent Configuration File

PSME FPGA-oF agent's configuration file can be found `in /etc/psme/psme-fpgaof-configuration.json` file.

The option `nic-drivers` in the file contains the list of Linux network drivers which will be used to gather a list of network interfaces that are on a given host. Only the interfaces that were created by one of the listed drivers are exposed on the API.

If an interface is missing from the API, then run the following command to determine the driver name of a network interface:

```
ethtool -i <interface_name>
```

The user should then add the driver name to the list and restart the PSME services.

FPGA-oF transport protocols, IP addresses, and ports can be edited in `opae-proxy/transports` sections.

5.7.2 Limitations

- An Endpoint can have only one Connected Entity with a Role Initiator or Target.
- Target Endpoint Connected Entity Links must be unique within service.
- An Endpoint cannot be deleted when it is in use (belongs to a Zone).
- An Endpoint can belong only to one Zone at a time.
- A Zone can have only one Initiator Endpoint, but multiple Target Endpoints.

5.8 Rack Management Module (RMM)

This section provides the prerequisites, installation, and configuration of the RMM.

5.8.1 Prerequisites

- PC with Linux OS, Ubuntu 16.04 recommended
- The hostname must be set to rmm-*
- the following libraries to be installed in the OS:

```
    libmicrohttpd10 libcurl3-gnutls libcurl3
```

5.8.2 Configuration

To ensure a functioning RMM with Intel® RSD, complete the following steps:

1. Grand Unified Bootloader (GRUB) configuration:

   a. Edit the `/etc/default/grub` file and comment out the following variables:

   ```
   # GRUB_HIDDEN_TIMEOUT
   # GRUB_HIDDEN_TIMEOUT_QUIET
   ```

   b. Edit the `/etc/default/grub` file and modify the following variables:

   ```
   GRUB_CMDLINE_LINUX_DEFAULT=""
   GRUB_TERMINAL=console
   GRUB_CMDLINE_LINUX="nomodeset net.ifnames=0 biosdevname=0 acpi_osi="
   GRUB_TIMEOUT=2
   GRUB_RECORDFAIL_TIMEOUT=2
   ```

   c. Apply changes by running the following command:

   ```
   sudo update-grub
   ```

2. VLAN configuration:

   a. Install the VLAN package in the amd64* version from *http://packages.ubuntu.com/trusty/vlan*.
   b. After installing the VLAN package, add it to `/etc/modules`.

   ```
   8021q
   ```

   c. Add VLANs 4092 and 4094 to `/etc/network/interfaces:`

   ```
   # This file describes the network interfaces available on your system
   # and how to activate them. For more information, see interfaces(5).
   # The loopback network interface
   auto lo iface lo inet loopback

   auto eth0
   iface eth0 inet manual
   ```

```
auto eth0.4092
iface eth0.4092 inet static
    address 1.1.1.253
    netmask 255.255.255.0
    vlan-raw-device eth0

auto eth0.4094
iface eth0.4094 inet dhcp
    vlan-raw-device eth0
    post-up ifconfig eth0.4094 mtu 1000
```

5.8.3 Software Setup

To setup the PSME RMM software, make the following adjustments to the service configuration file:

- In `/etc/psme/psme-rest-server-configuration.json` change:

```
     "network-interface-name": ["enp0s20f0.4094"] -> "network-interface-name":
["your_management_interface"]
"rmm-present": true -> "rmm-present": false
```

- Optionally, if needed, in `/etc/psme/psme-rmm-configuration.json` update location offsets of the Rack zones and path to devices used for communication:

```
     "locationOffset": 0 -> "locationOffset": your_location_offset
"device": "/dev/ttyCm1IPMI" -> "device": "your_device_path"
```

5.8.4 Communication with Drawers

The PSME RMM software communicates with PSME Chassis software running on drawers with the following goals:

- to gather telemetric data from a drawer,
- to provision the drawer with the CA certificate for authenticating PODM software,
- to set location information for MultiRack feature,
- to trigger drawer reboot upon receiving a `POST` request for `Chassis.Reset` action.

If the drawer is offline, or CyMUX software is not installed, or PSME Chassis software is not running, the features will not be available for a particular drawer.

The communication between the PSME RMM and the PSME Chassis is bridged through the CM and the MMP. As such, it occasionally timeouts. If Chassis.Reset action fails with an error message indicating a timeout, and it can be safely retried.

§

Appendix A Top-of-Rack Switch Configuration

This section contains the instructions for configuration of the Top-of-Rack switches in an Intel® RSD Software Development Vehicle rack.

A.1 Prerequisites

This section is specific to the reference design of the Intel® RSD Software Development Vehicle platform ToR switch configuration.

There are two ToR switches installed in the Intel® RSD Software Development Vehicle rack, one for the management network and one for the data network. The first one will only have static configuration explained later in this chapter. The second one will be managed by PSME but requires some default configuration done from CLI. Both ToRs are configured with various VLANs for connectivity between rack components.

The management switch is Arista 7010 Series*, while the data network switch is Arista 7060CX Series*.

A.2 Configuration Process for Management Network ToR (via CLI)

It is assumed that the following ports of the management ToR switch are used to connect different components:

1. Port 1: NUC with PODM.
2. Port 3: CM.
3. Port 5: RMM.
4. Port 48: the management port of the data network ToR switch.
5. Port 49: the Ethernet port 33 of the data network ToR switch.

The VLANs for each port are listed in the table below:

Table 15. ToR VLANs configuration

Switch #show vlan	Tagged VLANs on Port	Access VLANs on Port
4088	1, 49	-
4090	1, 3	-
4091	1, 49	-
4092	1, 3, 5	-
4093	1, 3, 49	-
4094	1, 3, 5, 49	48

Follow these steps to configure the switch:

1. Use the supplied DB9 to RS45 cable to connect the serial port of the ToR to a PC serial port.
2. To communicate with the ToR, open `putty.exe` (or another tool for communication via a serial port and set up the serial line and connection speed):
   a. Make sure to use the correct a serial port on the PC.
   b. Connection speed is 9600.
   c. Make sure to select "**Serial**".
   d. Open connection and wait for the ToR to respond,
3. Log into the ToR system as the `'admin'` user.

4. Type `"enable"` to enter privileged mode.
5. Type show vlan to list VLANs.
6. Enter configuration mode

```
    configure
```

7. Create all VLANs listed in the table "ToR VLANs configuration" above. For example, to create vlan 4090, type vlan 4090 to create VLAN 4090, then type exit.
8. To verify vlan creation type show vlan
9. Or to see ToR port 1 configuration, type show interface et1.

   To see all interfaces on ToR, type

```
    show interface
```

10. Add tagged VLANs to ports according to the table above. For example for port 1 VLANs 4088, 4090, 4091, 4092, 4093, and 4094 should be added:

    a. Type

    ```
      interface Ethernet 1
    switchport trunk allowed vlan 4088,4090-4094
    switchport mode trunk
    ```

    b. Type exit to return to config mode.

11. Add access VLANs to ports according to the table. For example, to add VLAN 4094 on port 48:

    a. Type

    ```
        interface Ethernet 48
    switchport access vlan 4094
    ```

    b. Type **exit** to return to config mode.

12. When finished, save the configuration by typing **write**.

A.3 Configuration Process for the Data Network ToR.

It is assumed that the first eight QSFP ports of the switch are already connected to all sleds in the rack using break-out cables like 4x25Gbps.

Note: The configuration will not apply to all four lanes of a port if not connected, and the steps will need to be repeated if connected later for each port individually.

Use the same way to access CLI of this ToR switch like in the previous chapter.

1. Login as '**admin**' user.
2. Enter privileged mode using the '`enable`' command.
3. Enter the configuration mode using the `'configure'` command.
4. Create VLANs in the VLAN database

   a. `DeepDiscovery` VLAN (for network 10.5)

   ```
     vlan 4088
   exit
   ```

   b. Production network VLAN (for network 10.1)

   ```
     vlan 4091
   exit
   ```

   c. Storage VLAN (for network 10.2)

```
 vlan 4093
exit
```

5. Configure VLANs on all QSFP ports.

   a. Configure VLANs for data network (10.1, 10.5) on interfaces connected to the first Mellanox's interface on the sled.

   ```
    interface ethernet <list of interfaces>
   switchport trunk allowed vlan 4088,4091
   switchport mode trunk
   switchport trunk native vlan 4091
   exit
   ```

   b. Configure VLANs for storage network (10.2) on interfaces connected to the second Mellanox's interface on the sled.

   ```
    interface ethernet <list of interfaces>
   switchport trunk allowed vlan 4093
   switchport mode trunk
   switchport trunk native vlan 4093
   exit
   ```

   c. Configure VLANs for management network (10.3) on interfaces connected to sleds working as management hosts (e.g., PNC management host).

   ```
    interface ethernet <list of interfaces>
   switchport trunk allowed vlan 4094
   switchport mode trunk
   switchport trunk native vlan 4094
   exit
   ```

6. Configure tagged VLANs on port 33.

```
         interface Ethernet 33
    switchport trunk allowed vlan 4088,4091,4093,4094
    switchport mode trunk
    exit
```

7. Enable Arista eAPI JSON-RPC management interface.

   Arista EOS offers multiple programmable interfaces for applications. One of them is the EOS API (eAPI). It allows applications to have programmatic control over EOS. Once the API is enabled, the switch accepts commands using Arista's CLI syntax, and responds with machine-readable output and errors serialized in JSON, served over HTTP.

   Arista PSME agent uses the eAPI management interface to configure Quality of Service parameters. It must be configured before starting PSME software.

   a. Configure HTTP server:

   ```
    management api http-commands
   protocol http localhost
   no shutdown
   exit
   ```

   b. Verify HTTP server status:

   ```
   show management api http-commands
   ```

8. Save the currently running configuration to the startup configuration.

```
copy running-config startup-config
```

Appendix B Appendix - Drawer Hardware Configuration

This section provides the steps for establishing networking to an Intel® RSD Software Development Vehicle drawer through the Control Module. Refer to these instructions in the event of issues with communication over I^{2}C.

B.1 Recommended Rack Setup

- x1 NUC
- x1 TOR
- x1 Fabric module
- x4 sleds based on the Second Generation Intel® Xeon® Scalable Processors

B.2 Control Module

B.2.1 Prerequisites

- Ensure PODM is installed and network interfaces configured properly
- Ensure that the TOR switch is configured properly
- Ensure screen tool is installed on PODM or RMM

B.2.2 Configuring CM

The following steps are valid for CM v1.02 or higher only. Earlier drops are preconfigured.

1. Log on to PODM or RMM
2. Get the IP address of the CM in a proper (upper or lower) power zone.
   a. Attach to the CM serial console (for lower power zone use Cm2)

```
 sudo screen /dev/ttyCm1Console
```

   b. Get the network configuration - type **net** shown in the CM console.

```
 > net show
IP      :       1.1.1.1
Mask    :       255.255.255.255
Gateway :       255.255.255.255
MAC     :       2c:60:0c:67:2a:04
```

   c. Detach screen session - **Ctrl- a - d**
   d. Kill screen session.
3. Use ipmitool to check the network connection to the CM:

```
rsa@pod-manager:~$ ipmitool -I lanp -U admin -P admin -H 1.1.1.1 mc info
Device ID                 : 37
Device Revision           : 0
Firmware Revision         : 1.03
IPMI Version              : 2.0
```

```
Manufacturer ID              : 7244
Manufacturer Name            : Unknown (0x1C4C)
Product ID                   : 12621 (0x314d)
Product Name                 : Unknown (0x314D)
Device Available             : yes
Provides Device SDRs         : yes
Additional Device Support :
    Sensor Device
    FRU Inventory Device
    Chassis Device Aux Firmware Rev Info      :
    0x00
    0x00
    0x00
    0x00
```

Note: If there is no response, then check the network configuration on PODM and TOR switch.

4. Configure VLANs (`0x30 <VLAN MSB> <VLAN LSB> <Member Bitmask MSB> <Member Bitmask LSB> <Tagged Bitmask MSB> <Tagged Bitmask LSB>)`:

   Assuming base = `ipmitool -I lanp -U admin -P admin -H 1.1.1.1`

```
<base> raw 0x38 0x30 0x0F 0xFC 0x07 0xff 0x06 0xff
<base> raw 0x38 0x30 0x0F 0xFD 0x06 0xff 0x06 0xff
<base> raw 0x38 0x30 0x0F 0xFE 0x06 0xff 0x06 0xff
```

5. Check VLAN configuration:

```
ipmitool -I lanp -U admin -P admin -H 1.1.1.1 raw 0x38 0x32
00 03 0f fe 06 ff 06 ff 0f fd 06 ff 06 ff 0f fc  07 ff 06 ff
```

   Explanation:

```
00 03 - 3 vlans
0f fc 07 ff 06 ff - vlan ffc (4092) - Rack Management
0f fd 06 ff 06 ff - vlan ffd (4093) - Storage Management
0f fe 06 ff 06 ff - vlan ffe (4094) - POD Management
```

6. Configure PVID for CM port

```
<base> raw 0x38 0x33 8 0x0F 0xFC
```

Note: After issuing this command, communication with the CM may be lost - depending on the actual TOR and POD VLAN 4092 settings.

7. Remove VLAN 1 (if it exists):

```
<base> raw 0x38 0x31 0 1
```

8. Store configuration in EEPROM (if required):

```
ipmitool -I lanp -U admin -P admin -H 1.1.1.1 raw 0x38 0x39
```

B.3 Intel® RSD Software Development Vehicle platform

This section provides additional information about configuring the Intel® RSD Software Development Vehicle platform.

B.3.1 MMP Switch

This section provides the configuration about configuring the MMP Switch.

B.3.1.1 Prerequisites

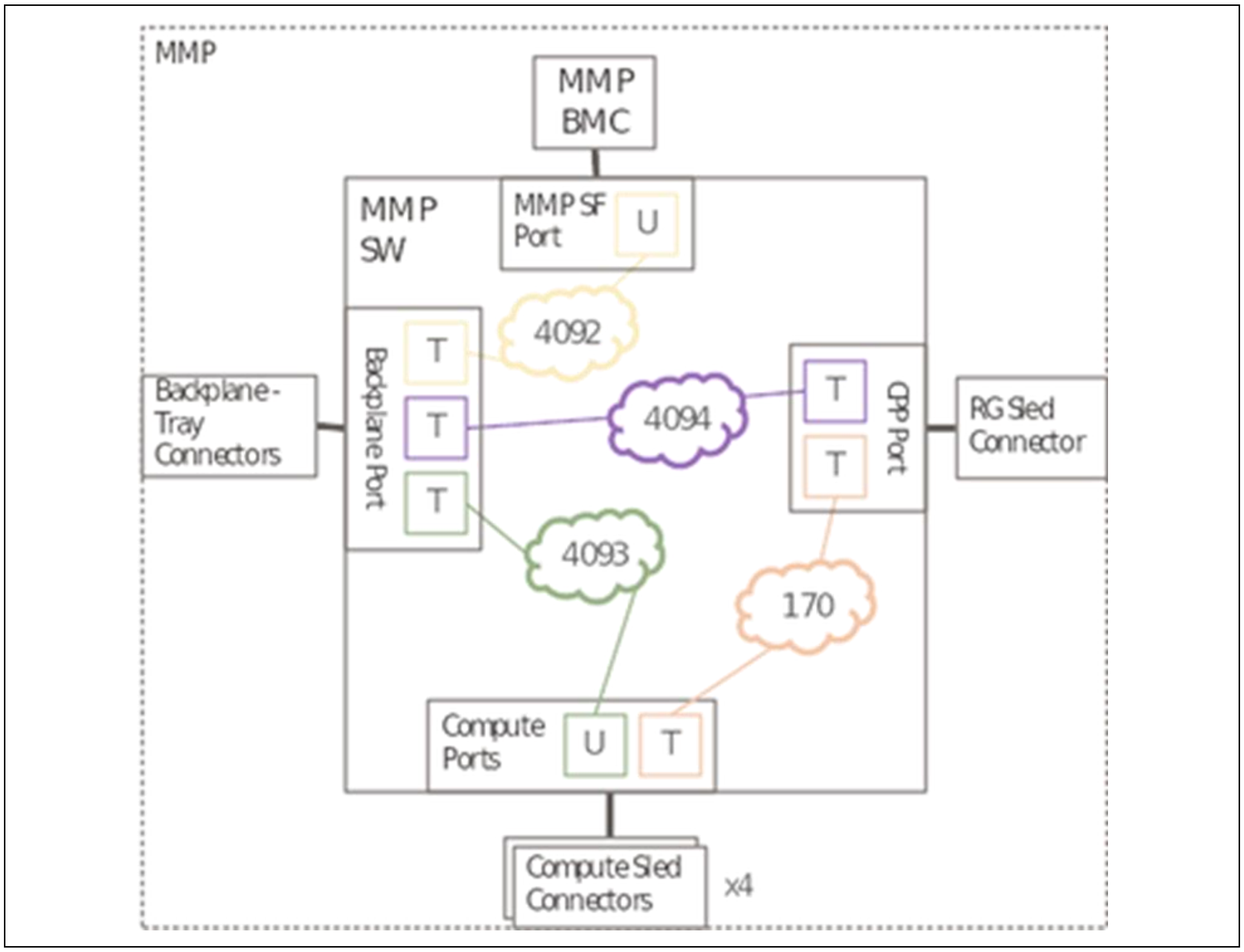
- Ensure PODM is installed and network interfaces are configured properly

- Ensure the ToR switch is configured properly
- Ensure screen is installed on PODM
- Ensure the MMP FW version is v1.10 or higher

B.3.1.2 Network Configuration

The MMP switch supplies all management network connections for sleds, CPP, and MMP BMC. The required network configuration is presented in the diagram:

Figure 14. Network Configuration



B.3.1.3 Configuring MMP

Assuming that the CM IP address is known, the MMP switch can be configured by sending IPMI commands to the CM using the rack management network (1.1.1.x) on the PODM NUC. If the CM IP address is not known, then follow steps 1-3 from the "Configuring CM" section above.

1. Check the network connection to MMP - specify a drawer in the power zone with option -b x, where x can be 0, 2, 4 or 6 for drawer 1 to 4 accordingly. FW revision can be checked now.

```
rsa@pod-manager:~$ ipmitool -I lanp -U admin -P admin -H 1.1.1.1 -b 0 -t
0x24 mc info
Device ID                    : 37
Device Revision              : 0
```

```
Firmware Revision         : 1.04
IPMI Version              : 2.0
Manufacturer ID           : 7244
Manufacturer Name         : Unknown (0x1C4C)
Product ID                : 12621 (0x314d)
Product Name              : Unknown (0x314D)
Device Available          : yes
Provides Device SDRs      : no
Additional Device Support :
    FRU Inventory Device
    IPMB Event Receiver
    IPMB Event Generator
    Chassis Device
Aux Firmware Rev Info     :
    0x00
    0x00
    0x00
    0x00
```

2. Configure VLANs (0x30 <VLAN MSB> <VLAN LSB> <Member Bitmask> <Tagged Bitmask>):

   Assuming base = `ipmitool -I lanp -U admin -P admin -H 1.1.1.1 -b 0 -t 0x24`

```
<base> raw 0x38 0x30 0x0F 0xFC 0x50 0x40
<base> raw 0x38 0x30 0x0F 0xFD 0x4F 0x40
<base> raw 0x38 0x30 0x0F 0xFE 0x60
<base> raw 0x38 0x30 0x00 0xAA 0x2F
```

3. Check VLAN configuration:

```
ipmitool -I lanp -U admin -P admin -H 1.1.1.1 -b 0 -t 0x24 raw 0x38 0x32
00 04 0f fc 50 40 0f fd 4f 40 0f fe 60 60 00 aa 2f
```

   Explanation:

```
00 04 - 4 vlans
0f fc 50 40 - vlan ffc (4092) - Rack Management
0f fd 4f 40 - vlan ffd (4093) - Storage Management
0f fe 60 60 - vlan ffe (4094) - POD Management
00 aa 2f - vlan aa (170) - Tray (drawer) Management
```

4. Remove VLAN 1 (if it exists):

```
<base> raw 0x38 0x31 0 1
```

5. Set PVIDs:

```
<base> raw 0x38 0x33 0 0x0F 0xFD
<base> raw 0x38 0x33 1 0x0F 0xFD
<base> raw 0x38 0x33 2 0x0F 0xFD
<base> raw 0x38 0x33 3 0x0F 0xFD
<base> raw 0x38 0x33 4 0x0F 0xFC
<base> raw 0x38 0x33 5 0x0F 0xFE
<base> raw 0x38 0x33 6 0x0F 0xFE
```

6. Check PVIDs:

```
ipmitool -I lanp -U admin -P admin -H 1.1.1.1 -b 0 -t 0x24 raw 0x38 0x34
07 0f fd 0f fd 0f fd 0f fd 0f fc 0f fe 0f fe
```

7. Store configuration in EEPROM (if required):

```
ipmitool -I lanp -U admin -P admin -H 1.1.1.1 -b 0 -t 0x24 raw 0x38 0x39
```

B.3.2 Drawer OS Configuration.

For the PSME compute and chassis to work properly, the drawer should be installed with Ubuntu v16.04 OS. Details of the network configuration are covered in to Section 4.0, Intel® RSD Rack Network Configuration.

B.4 Remote iSCSI Target Blade Boot

This section provides the requirements for configuring an Intel® RSD Software Development Vehicle platform for the iSCSI Out of Band boot feature.

B.4.1 Prerequisites

- Be sure that BMC and BIOS are up-to-date.
- Be sure that Intel® RSD Software Development Vehicle platform networks are properly configured. SLED has access to storage management (10.2.0.x) and public (10.1.0.x) network.

§

Appendix C PSME Software Installation from Packages

This section provides the instructions for the installation of PSME software from packages.

C.1 PSME Software Packages - Introduction

The section is applicable for those who have access to RPM/DEB packages with PSME Software. The following PSME packages are available:

- a package for each binary is listed Table 2., Binaries Working, Detailed information about using particular packages is provided in the following sections.
- PSME Common packages, required by other PSME packages. A common package is a set of shared libraries and executables for the PSME provided by RPM/DEB packages. Note that appropriate psme-common package must be installed before other packages.
- The PSME Network Configuration package, used for drawer configuration. For more information, refer to Appendix C.4., PSME Network Configuration Package.

C.2 Package Installation

All package operations require `root` privileges. All required packages must be copied to the target system. Network connectivity must be configured correctly (e.g., the proxy must be set if needed).

All required dependencies are tracked during installation and must be met. All required system dependencies must be installed prior to the PSME installation. The necessary steps are described in "Install required system dependencies" chapters in this appendix.

In addition, PSME packages track services configuration changes and configure systemd services.

C.2.1 Initial package(s) Installation

If the PSME was not installed before or had been purged from the system, default configuration files are installed. Default configuration must be refined to match the setup. All additional steps are described in the "Edit configuration files" chapters in this appendix.

After adapting the configuration to suit the user's needs, the administrator should either reboot the platform to configure the system settings and start the PSME services automatically or start them manually (by means of systemctl command).

C.2.2 Certificate Management

Each PSME REST server service must have the proper PSME certificate and private key stored in the certificates directory. If these are not installed by the administrator prior to the first installation, then default ones will be installed in the default directory `/etc/psme/certs`.

PODM's CA certificate is not installed automatically. It must be installed manually for services which do not communicate with RMM (see additional remarks related to "rmm-present" configuration flag).

C.2.3 Upgrade Package(s)

During upgrades, PSME packages (installed previously) must be overwritten with new ones. All installed packages should have the same version.

Existing configuration files {component-name}-configuration.json are preserved. If the newly installed configuration files are different than previous ones, then the new is saved to {component-name}-`configuration.json.new` extensions and a warning is logged. All required changes must be manually merged by the administrator.

If PSME services were running before the upgrade, they are automatically restarted after the installation, but with the old configuration files. Therefore if any changes need to be made, all services will need to be restarted after the configuration is updated.

C.2.4 Removal of Installed Package(s)

During PSME package removal, current configuration files are renamed to {component-name}-`configuration.json.old`. These files would be used as default by the next installation. If PSME packages are purged, however, these files are removed.

C.3 PSME Compute Ubuntu v16.04 Packages

PSME Compute must be installed on the drawer; it communicates with Rack Management Module by means of I^{2}C channel. PSME Compute receives PODM CA's certificate from RMM, so it is not necessary to install this certificate locally.

The following PSME binary installation files must be built from sources or acquired from pre-built binaries:

```
-   PSME Common (`psme-common-{version}.deb`)

-   PSME Chassis (`psme-chassis-{version}.deb`)

-   PSME Compute (`psme-compute-{version}.deb`)

-   PSME Rest Server (`psme-rest-server-{version}.deb`)
```

C.3.1 Installation

1. Install `CyMUX` following the instructions Appendix G.2, Installing CyMUX.
2. Install the required system dependencies:
   ```
   apt-get install libmicrohttpd10 libcurl3 libcurl3-gnutls openssl
   ```
3. Install the packages:
   ```
   dpkg -i psme-common-{version}.deb
   dpkg -i psme-chassis-{version}.deb
   dpkg -i psme-compute-{version}.deb
   dpkg -i psme-rest-server-{version}.deb
   ```
4. Change the hostname, to begin with, `"psme"` (it must be compatible with regular expression `"^psme.*"`):
   ```
   hostnamectl set-hostname --static "psme-drawer-1"
   ```
5. Reboot the platform:
   ```
   reboot
   ```

C.4 PSME Network Configuration Package

This package is intended to simplify the process of network configuration within the rack, and thus its installation is optional. If the rack is already configured according to Section 4.0, Intel® RSD Rack Network Configuration, then this step may be skipped.

The package contains network configuration files for VLANs 170 (network v1.1.2.0 for communication with BMCs, MMPs and CMs) and 4094 (network v10.3.0.0 for communication with PODM).

It is intended to be installed only on the CPP (drawer/tray), where PSME Compute service runs.

The following PSME binary installation files must be built from sources or acquired from pre-built binaries:

```
-   Intel® RSD Software Development Vehicle Network Configuration (`psme-network-
config-{version}.deb`)
```

C.4.1 Installation

1. Install the package.

   ```
   dpkg -i psme-network-config-{version}.deb
   ```

2. To enable network configuration changes after package installation, restart the network service

   ```
   systemd-networkd.service restart
   ```

   or reboot the system.

   ```
   reboot
   ```

C.5 Rack Management Module Ubuntu v16.04 Packages

RMM software must be installed on a computer connected to CM(s) by USB cables.

The following PSME binary installation files must be built from sources or acquired from pre-built binaries:

```
-   PSME Common (`psme-common-{version}.deb`)

-   PSME Rack Management Module (`psme-rmm-{version}.deb`)

-   PSME Rest Server (`psme-rest-server-{version}.deb`)
```

C.5.1 Installation

1. Install required system dependencies:

   ```
   apt-get install libmicrohttpd10 libcurl3
   ```

2. Install the packages:

   ```
   dpkg -i psme-common-{version}.deb
   dpkg -i psme-rmm-{version}.deb
   dpkg -i psme-rest-server-{version}.deb
   ```

3. Edit configuration files:

   In `/etc/psme/psme-rest-server-configuration.json` change:

   ```
   "network-interface-name": ["enp0s20f0.4094"] -> "network-interface-name":
   ["your_management_interface"]
   "rmm-present": true -> "rmm-present": false
   ```

   Optionally, if needed, in `/etc/psme/psme-rmm-configuration.json` update location offsets of the Rack zones and path to devices used for communication:

   ```
   "locationOffset": 0 -> "locationOffset": your_location_offset
   "device": "/dev/ttyCm1IPMI" -> "device": "your_device_path"
   ```

4. Reboot the system to finish RMM configuration.

   ```
   Devices will be configured properly, and service will start automatically after
   reboot.
   ```

C.6 Storage Services Ubuntu v16.04 Packages

The following PSME binary installation files must be built from sources or acquired from pre-built binaries:

```
-   PSME Common (`psme-common-{version}.deb`)

-   PSME Storage (`psme-storage-{version}.deb`)

-   PSME Rest Server (`psme-rest-server-{version}.deb`)
```

C.6.1 Installation

1. Install required system dependencies:
```
apt-get install libmicrohttpd-dev libcurl4-openssl-dev tgt lvm2 liblvm2app2.2
```
2. Install the packages:
```
dpkg -i psme-common-{version}.deb
dpkg -i psme-storage-{version}.deb
dpkg -i psme-rest-server-{version}.deb
```
3. Edit configuration files:

   In `/etc/psme/psme-rest-server-configuration.json` change:
```
"network-interface-name": ["enp0s20f0.4094"] -> "network-interface-name":
["your_management_interface"]
"rmm-present": true -> "rmm-present": false
```
   Optionally in `/etc/psme/psme-storage-configuration.json` change interface which is used as Portal IP (for connection to tgt daemon targets):
```
"portal-interface" : "eth0" -> "portal-interface" :
"interface_for_connection_to_targets"
```
4. Change the hostname to begin with `"storage"` (it must be compatible with regular expression `"^storage.*"`):
```
hostnamectl set-hostname --static "storage-1"
```
   DHCP client for the management interface must be enabled.

5. Start services:
```
service psme-rest-server start
service psme-storage start
```

C.7 PSME PNC Ubuntu v16.04 Packages

PSME PNC must be installed on the management host which is connected to PCI switch board.

The following PSME binary installation files must be built from sources or acquired from pre-built binaries:

```
-   PSME Common (`psme-common-{version}.deb`)

-   PSME PNC (`psme-pnc-{version}.deb`)

-   PSME Rest Server (`psme-rest-server-{version}.deb`)
```

C.7.1 Installation

1. Install required system dependencies:
```
apt-get install libmicrohttpd-dev libcurl4-openssl-dev
```

2. Install the packages:

```
dpkg -i psme-common-{version}.deb
dpkg -i psme-pnc-{version}.deb
dpkg -i psme-rest-server-{version}.deb
```

3. Edit configuration files:

   In `/etc/psme/psme-rest-server-configuration.json` change:

```
"network-interface-name" : ["enp0s20f0.4094"] -> "network-interface-name" :
["your_management_interface"]
"rmm-present": true -> "rmm-present": false
```

   In `/etc/psme/psme-pnc-configuration.json` change:

```
"network-interface-name" : "eth0" -> "network-interface-name" :
"your_management_interface"
```

4. Restart management host and switch board.

C.8 PSME FPGA-oF Target Ubuntu v16.04 Packages

PSME FPGA-oF must be installed on the target host which has connected FPGA.

The following PSME binary installation files must be built from sources or acquired from pre-built binaries:

```
-   PSME Common (`psme-common-{version}.deb`)

-   PSME FPGA-oF (`psme-fpgaof-{version}.deb`)

-   PSME Rest Server (`psme-rest-server-{version}.deb`)
```

C.8.1 Installation

1. Install required system dependencies:

```
apt-get install libmicrohttpd-dev, libcurl4-openssl-dev, libnl-3-200, libnl-route-
3-200, libibverbs1, librdmacm1

wget https://github.com/ofiwg/libfabric/releases/download/v1.7.0/libfabric-
1.7.0.tar.gz
tar xfv libfabric-1.7.0.tar.gz
cd libfabric-1.7.0
./configure
make
sudo make install
```

2. Install the packages:

```
dpkg -i psme-common-{version}.deb
dpkg -i psme-fpgaof-{version}.deb
dpkg -i psme-rest-server-{version}.deb
```

3. Edit configuration files:

   In `/etc/psme/psme-rest-server-configuration.json` change:

```
"network-interface-name" : ["enp0s20f0.4094"] -> "network-interface-name" :
["your_management_interface"]
"rmm-present": true -> "rmm-present": false
```

   In `/etc/psme/psme-fpgaof-configuration.json`

   Update `secureEraseGBS` to define path to default bitstream used for reconfiguration of FPGA acceleration slot during `Secure Erase`.

Update `transports` section to define protocols, IP addresses, and ports used for communication with initiator host.

```
     "opae-proxy": {
    "transports": [
        {
            "protocol": "TCP",
            "ipv4": "127.0.0.1",
            "port": 8447
        },
        {
            "protocol": "RDMA",
            "ipv4": "127.0.0.1",
            "port": 8448
        }
    ]
}
```

Optionally update nic-drivers field accordingly to required drivers for NICs that are used on the host.

4. Restart PSME FPGA-oF agent.

```
service psme-fpgaof restart
```

C.9 PSME NVMe Target Ubuntu v16.04 Packages

The following PSME binary installation files must be built from sources or acquired from pre-built binaries:

```
-   PSME Common (`psme-common-{version}.deb`)

-   PSME NVMe Target (`psme-nvme-target-{version}.deb`)

-   PSME Rest Server (`psme-rest-server-{version}.deb`)
```

C.9.1 Installation

1. Install required system dependencies:

```
apt-get install libmicrohttpd10 libcurl3 libnl-genl-3-200 libnl-route-3-200
```

2. Change the hostname to begin with `"storage"` (it must be compatible with regular expression "^storage.*"):

```
hostnamectl set-hostname --static "storage-1"
DHCP client for the management interface must be enabled.
```

3. Install the packages:

```
dpkg -i psme-common-{version}.deb
dpkg -i psme-nvme-target-{version}.deb
dpkg -i psme-rest-server-{version}.deb
```

4. Edit configuration files:

In `/etc/psme/psme-rest-server-configuration.json` change:

```
"network-interface-name" : ["enp0s20f0.4094"] -> "network-interface-name" :
["your_management_interface"]
"rmm-present": true -> "rmm-present": false
```

In `/etc/psme/psme-nvme-target-configuration.json` optionally update `nic-drivers` field accordingly to required drivers for NICs that are used on the host.

C.10 PSME Discovery Ubuntu v16.04 Packages

The following PSME binary installation files must be built from sources or acquired from pre-built binaries:

```
-   PSME Common (`psme-common-{version}.deb`)

-   PSME NVMe Discovery (`psme-nvme-discovery-{version}.deb`)

-   PSME FPGA Discovery (`psme-fpga-discovery-{version}.deb`)

-   PSME Discovery Server (`psme-nvme-discovery-server-{version}.deb`)
```

C.10.1 Installation

1. Install the required system dependencies:

```
apt-get install libmicrohttpd10 libcurl3 libnl-genl-3-200 \
libnl-route-3-200 libibverbs1 librdmacm1
```

   On a host with Mellanox* ConnectX-3/ConnectX-3 Pro interfaces install:

```
apt install libmlx4-1
apt install libmlx5-1
```

2. If the PSME NVMe Discovery is installed on the same host as a PSME NVMe Target, then set the hostname according to Appendix C.9, PSME NVMe Target Ubuntu v16.04 Packages above. However, if the PSME NVMe Discovery packages are installed on a separate host, then change the operating system hostname to `discovery-service`:

```
hostnamectl set-hostname --static "discovery-service"
```

   DHCP client for the management interface must be enabled.

   There should be only one PSME NVMe Discovery host in a rack.

3. Install PSME NVMe Discovery, PSME FPGA Discovery and PSME Rest Server:

```
dpkg -i psme-common-{version}.deb
dpkg -i psme-nvme-discovery-{version}.deb
dpkg -i psme-fpga-discovery-{version}.deb
dpkg -i psme-nvme-discovery-server-{version}.deb
```

4. Edit configuration files:

   In `/etc/psme/psme-discovery-server-configuration.json` change:

```
"network-interface-name" : ["eth2"] -> "network-interface-name" :
["your_management_interface"]
```

Note: If the PSME NVMe Discovery is installed on the same host as a PSME NVMe Target or a PSME FPGA-oF Target, the network-interface-name in /`psme-discovery-server-configuration.json` should be a different interface than the interface used by the Target's Rest Service (network-interface-name in `/etc/psme/psme-rest-server-configuration.json`). Both services should be available on separate IP addresses.

In /etc/psme/psme-nvme-discovery-configuration.json change:

```
     "discovery-service": {
    "listener-interfaces": [
        {
            "ofi-provider" : "verbs",
            "trtype" : "rdma",
            "adrfam" : "ipv4",
            "traddr": "127.0.0.1", -> "traddr" : "ipv4 address of your RDMA
interface"
            "trsvcid": "4420"
        }
    ]
}
```

C.11 PSME packages for Arista* EOS

The following PSME binary installation files must be built from sources or acquired from pre-built binaries:

```
-   PSME Common (`psme-common-arista-{version}.rpm`)

-   PSME Network (`psme-network-arista-{version}.rpm`)

-   PSME Rest Server (`psme-rest-server-arista-{version}.rpm`)
```

C.11.1 Installation

1. Store certificates (PODM CA's, REST server certificate chain and REST server private key) in `/mnt/flash/certs/ directory`.
2. To install the PSME RPM packages for Arista, use the BASH shell and follow standard Fedora* installation methods:

```
rpm -i psme-*-arista-*.rpm
```

Note: Packages installed using this method are not preserved after reboot.

3. Installation from CLI (it is assumed all packages are copied to /tmp).

   a. Enter configuration mode:

   ```
   enable
   configure
   ```

   b. For each PSME RPM package, copy and install as an extension:

   ```
   copy file:/tmp/<psme...>.rpm extension:
   extension <psme...>.rpm
   ```

   c. To have all packages installed after reboot, run the command:

   ```
   copy installed-extensions boot-extensions
   ```

C.11.2 Update

Before attempting to update PSME software on the switch, please stop the PSME network agent from CLI configuration mode:

```
    daemon psmenet
    shutdown
```

1. When packages were installed using BASH follow the standard Fedora update methods:

```
rpm -U psme-*-arista-*.rpm
```

2. If packages were installed using CLI you need to remove old extensions first and then install new ones.

   a. For each old RPM package uninstall the extension and delete it:

   ```
   no extension <psme...>.rpm
       delete extension:<psme...>.rpm
   ```

   b. Install new packages like in the previous section.
   c. Copy installed extensions to boot extensions.

C.11.3 Configuring and Starting PSME Services

1. PSME REST server needs to be started from systemd after each installation and reboot:

   ```
   systemctl start psme-rest-server
   ```

2. PSME network agent needs to be configured as EOS daemon from CLI configuration mode:

   ```
   daemon psmenet
   exec /opt/psme/bin/psmenet
   no shutdown
   exit
   write
   ```

   After the write command, the network agent will be started after each EOS reboot if installed as an extension.

C.12 Package Signatures

A GPG key pair is needed to sign Linux packages. The following command can be used to check existing keys in the system:

```
gpg --list-key
```

To create a new key pair use the following command (note it will take a while to finish):

```
gpg --gen-key
```

C.12.1 Signing a Package

To sign a .deb package, use the command below:

```
dpkg-sig -s builder <deb package>
```

Before signing a `.rpm` package, configure the `.rpmmacros` file as follows:

```
%_signature gpg
%_gpg_path <full path to .gnupg file, i.e. /root/.gnupg>
%_gpg_name <key ID>
%_gpgbin /usr/bin/gpg
```

To sign a `.rpm` package use this command:

```
rpm --addsign <RPM package>
```

Once the packages are signed, refer to the *GNU Privacy Handbook*, Table 4 to exchange the GPG key with the recipient.

C.12.2 Checking Signatures

Before checking a signature of a `.deb` package, the GPG public key that was used during package signing may need to be imported:

```
gpg --import <gpg public key file>
```

To verify a signature in a .deb package, run following command:

```
dpkg-sig -c <psme package>.deb
```

On an Arista EOS system, import the GPG public key file using the following command:

```
sudo rpm --import <GPG public key file>
```

To check the signature in a `.rpm` file run:

```
rpm --checksig <PSME package>.rpm
```

§

Appendix D IPMI commands supported by Intel® RSD Software Development Vehicle MMP BMC

This appendix provides the IPMI commands supported by Intel® RSD Software Development Vehicle MMP BMC.

```
<base> = "ipmitool -I lan -U admin -P admin -H <CM IP> -b <Bridge #> -t 0x24 "
```

`<Bridge #> = 0,2,4,6` for trays 1,2,3,4 in a power zone.

Port Numbers for use as a number in commands and bit numbers in bitmasks:

```
0-3 : Sled BMC 0-3

4   : MMP BMC

5   : RRC CPP (not used by PSME Software Development Vehicle solution)

6   : Uplink (backplane connection)
```

- Add/Update VLAN (0x30):
  ```
  <base> raw 0x38 0x30 <VLAN MSB> <VLAN LSB> <Member Bitmask> <Tagged Bitmask>
  ```
- Dump VLANs (0x32):
  ```
  <base> raw 0x38 0x32
  ```
- Delete VLAN (0x31):
  ```
  <base> raw 0x38 0x31 <VLAN MSB> <VLAN LSB>
  ```
- Set PVID (0x33):
  ```
  <base> raw 0x38 0x33 <Port #> <VLAN MSB> <VLAN LSB>
  ```
- Dump PVIDs (0x34):
  ```
  <base> raw 0x38 0x34
  ```
- Save VLAN Configuration (0x39):
  ```
  <base> raw 0x38 0x39
  ```

§

Appendix E SPDK Installation from Sources

This section provides the step-by-step instructions for installing SPDK on a storage host.

E.1 System Dependencies

1. Make sure that all modules and libraries are installed in the operating system:

   Intel® Rack Scale Design SDV uses Mellanox NICs (ConnectX-3/ConnectX-3 Pro or Mellanox ConnectX-4/ConnectX-4 Lx).

   For Mellanox ConnectX-3 series run:

```
sudo apt install libmlx4-1
modprobe mlx4_core
modprobe mlx4_ib
```

   For Mellanox ConnectX-4 series run:

```
sudo apt install libmlx5-1
modprobe mlx5_core
modprobe mlx5_ib
```

2. Some kernel modules are required to load at boot time. Add the following lines to `/etc/modules` file:

```
nvmet
nvmet-rdma
mlx5_ib
rdma_ucm
ib_ucm
ib_uverbs
nvme
nvme_rdma
```

3. Reboot host machine:

```
reboot
```

4. Verify correctness of the Mellanox drivers installation. The following command should list the required modules:

```
lsmod | grep mlx

mlx5_ib                163840  0
ib_core                212992  10
ib_iser,ib_cm,rdma_cm,nvme_rdma,ib_uverbs,iw_cm,mlx5_ib,ib_ucm,rdma_ucm,nvmet_rdma
mlx5_core              339968  1 mlx5_ib
devlink                 28672  1 mlx5_core
ptp                     20480  2 igb,mlx5_core
```

5. Verify correctness of the RDMA modules installation. The following command should list the required modules:

```
lsmod | grep rdma

nvme_rdma               28672  0
nvme_fabrics            20480  1 nvme_rdma
rdma_ucm                28672  1
ib_uverbs               65536  8 ib_ucm,rdma_ucm
nvmet_rdma              24576  0
rdma_cm                 57344  4 ib_iser,nvme_rdma,rdma_ucm,nvmet_rdma
iw_cm                   49152  1 rdma_cm
ib_cm                   45056  2 rdma_cm,ib_ucm
```

```
ib_core                  212992  10
ib_iser,ib_cm,rdma_cm,nvme_rdma,ib_uverbs,iw_cm,mlx5_ib,ib_ucm,rdma_ucm,nvmet_rdma
nvmet                     49152  1 nvmet_rdma
configfs                  40960  3 rdma_cm,nvmet
nvme_core                 53248  7 nvme_fabrics,nvme_rdma,nvme
```

E.2 Step-by-step Installation Instructions for SPDK

The Storage Performance Development Kit is distributed only as source code, the user has to download the SPDK repository and install its dependencies.

```
git clone https://github.com/spdk/spdk
cd spdk
```

1. Current implementation of Intel® RSD PSME agent uses v19.01.1 release of the SPDK. The user needs to checkout the release version of the SPDK repository using the following command:

```
git checkout tags/v19.01.1
```

2. Then install submodules and install the full set of dependencies required to build and develop SPDK:

```
git submodule update --init
sudo scripts/pkgdep.sh
```

   In case of errors during packages installation, be sure that all your local packages are up-to-date:

```
apt-get update
apt-get upgrade
```

3. Build SPDK daemon with RDMA support:

```
./configure --with-rdma
 make
```

4. To turn on extra log level, add the following flag to the previous step:

```
./configure --with-rdma --enable-debug
```

Note: If there were errors related to the DPDK during migrations from previous SPDK versions, the DPDK could need to be refreshed by the following command:

```
 rm -rf ./dpdk/
 git submodule update --init
```

5. Once completed, confirm the build is working by running the unit tests:

```
./test/unit/unittest.sh
```

6. Before running SPDK, the hugepages must be allocated and devices need to be bound to SPDK:

```
sudo NRHUGE=2048 scripts/setup.sh
```

   (optional) To bind devices back from SPDK to the kernel run:

```
sudo scripts/setup.sh reset
```

7. Finally run SPDK NVMf Target daemon (nvmf_tgt):

```
sudo app/nvmf_tgt/nvmf_tgt
```

   To enable extra log, add the following argument to the previous step:

```
sudo app/nvmf_tgt/nvmf_tgt -L all
```

§

Appendix F Additional Quality of Service (QoS) configuration for sleds

This section contains the instructions for Quality of Service configuration of the sleds in an Intel® RSD Software Development Vehicle rack.

F.1 Prerequisites

1. Mellanox* OFED is installed.
2. LLDPAD service is started. This can be verified by running:

```
     service lldpad status
```

F.2 Configuration Process for QoS on Compute Sleds.

1. Ensure that RoCEv2 mode is set for all interfaces.

   Get a list of all Mellanox interfaces using `ibdev2netdev` script

```
ibdev2netdev

# example output:
mlx5_0 port 1 => eth94s0f0 (Up)
mlx5_1 port 1 => enp94s0f1 (Up)
```

   then, for each interface, set RoCEv2 mode:

```
cma_roce_mode -d mlx5_0 -p 1 -m 2
cma_roce_mode -d mlx5_1 -p 1 -m 2
```

2. Read the LLDP application configuration, e.g.

```
lldptool -t -i enp94s0f0 -V APP -c app

# example output:
APP=(prio,sel,proto)
0:({L2-priority},3,{protocol-id}) peer hw (set)
```

   where: L2-priority, `protocol-id` - sent by the Ethernet switch through DCBX protocol.

3. Set RoCE default ToS of RDMA Connection Manager applications, e.g.

```
cma_roce_tos -d mlx5_0 -t [ToS]
```

   where: `ToS` - one of Type of Service.

   For more detailed information refer to Table 4, *Default ToS to skprio mapping on Linux**.

4. Map kernel priority to egress QoS L2 priority, e.g.

```
     vconfig set_egress_map enp94s0f0.600 {skprio} {L2-priority}

# example output:
......
Device: enp94s0f1
INGRESS priority mappings: 0:0  1:0  2:0  3:0  4:0  5:0  6:0 7:0
EGRESS priority mappings: **{skprio}**:**{L2-priority}**
```

   where: `skprio` - Linux priority mapped to ToS configured in the previous step.

§

Appendix G Miscellaneous

This appendix contains additional information relevant to PSME and Intel® Rack Scale Design.

G.1 Compilation Code Coverage and Sanitizer Build Versions

Building with code coverage (only GCC):

```
mkdir build.coverage
cd build.coverage
cmake -DCMAKE_BUILD_TYPE=Coverage ..
```

Building with address/memory sanitizer (only GCC, libasan has to be installed):

```
mkdir build.asanitize
cd build.asanitize
cmake -DCMAKE_BUILD_TYPE=asanitize ..
```

Building with thread sanitizer (only GCC, libtsan has to be installed):

```
mkdir build.tsanitize
cd build.tsanitize
cmake -DCMAKE_BUILD_TYPE=tsanitize ..
```

Running code coverage, which will also run unit tests to collect code coverage traces:

```
make code-coverage
```

Reading code coverage results:

```
YOUR_WEB_BROWSER code_coverage/html/index.html
```

G.2 Installing CyMUX

1. Go to *https://github.com/01org/intelRSD/tree/master/tools/CyMUX* to build and install CyMUX service from source.
2. Alternatively, install `cyMUX` dependency from package:

```
        $ dpkg -i cymux-{version}.deb
```

§