Intel® Rack Scale Design (Intel® RSD) Pooled System Management Engine (PSME)

User Guide Software v2.3.2

September 2018

Revision 003US

Document Number: 337196-003US

Contents

Figures

Tables

Revision History

Revision	Description	Date
003US	Intel® RSD Storage Services Software v2.3.2: • A Sample implementation of the NVMe-oF initiator script described in the PSME User Guide is publically released. • Added intro statement to Section 2.3 Security Features • Section 2.13.18 revised adding NVMe-Wheel • Added Intro statement to Section 4.1.4 Intel® RSD Software Development Vehicle Drawer Network • Added intro statement to Section 4.2 Networking-Releasted Remarks	September 2018
002US	Intel® RSD Storage Services Software v2.3.1: • Added Cloud Infrastructure Management Interface (CIMI) specification to Table 4. • Updated Section 2.2, Deep Discovery. • Added Note Below Section 2.4.1. • Section 2.10.1.1, defined the Purley code name for Pubic use. • Updated Section 2.13.4 and Figure 3. • Moved Note from Section 2.13.3.4 to 2.13.2.3. • Updated Section 3.1.2, deprecated Deep Discovery feature. • Section 3.2, Changed text "All PSME modules must be built from the previously prepared build directory…" to Note.	July 2018
001US	Initial release for Intel® RSD Storage Services software v2.3.	May 2018

§

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 a full and detailed documentation of the Pooled System Management Engine (PSME) software v2.3.2. Information below covers minimal requirements for hardware and software during compilation and runtime. The 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
- [PSME] Dev environment
- Hardware requirements and software prerequisites
- PSME software installation and deployment
- Hardware and PSME software configuration

1.2 Intended Audience

- Software vendors (xSVs) 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 v2.3.2 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. 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). Communicates with the rack management module (RMM).

- **PSME Pooled Non-Volatile Memory express (NVMe*) Controller (PNC) agent:** responsible for gathering detailed information about the Peripheral Connect Interface express* (PCIe*) storage switch and attaching NVMe* drives to compute hosts.
- **PSME Storage agent:** responsible for preparing, configuring, gathering, and connection of storage logical volume management (LVM) and Linux target framework* (tgt). This agent connects to the PSME Storage Service.
- **RMM agent:** Rack Management Module (RMM) is responsible for managing and gathering detailed information about rack and its power/thermal metrics.
- **PSME NVMe agent:** responsible for managing and gathering detailed information about NVMe volumes attached to hosts through Remote Direct Memory Access (RDMA) Network Interface Controllers (NICs) using NVMe over Fabrics (NVMe-oF)* technology.
- **PSME NVMe discovery agent:** responsible for responding to queries about available NVMe volumes from NVMe-oF initiators.
- **PSME Compute agent simulator:** used to imitate the PSME compute agent with data read from an XML file. The XML file describes hardware (assets layout and details), validates with an XML schema, and sends the information to the PSME REST server.

1.4 Supported System Environments

Table 1. Source Compilation

Component	Ubuntu* v16.04	ARM* xcompiled (Buildroot)	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 NVMe for Intel® Software Development Vehicle	+	-	-
PSME NVMe Discovery for Intel® Software Development Vehicle	+	-	-
PSME Compute Simulator	+	-	-

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 NVMe Discovery for Intel® Software Development Vehicle	+	-

The PSME software is designed and developed to support generic hardware and various operating system 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.

1.4.1 Supported Hardware

- Software Development Vehicle platform
- Supported operating systems:
  - Ubuntu* v16.04 LTS
  - Arista* 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 operating system.

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
API	Application Program Interface
ACLs	Access Control Lists
.deb*	Debian*
Blade	Server board that equates to the `SPMF:ComputerSystem`
BMC	Baseboard Management Controller
BIOS	Basic Input/Output System
CA	Certificate Authority
CEE	Converged Enhanced Ethernet
CHAP	Challenge-Handshake Authentication Protocol
CLI	Command Line Interface
CM	Control Module
CPP	Control Plane Processor
DCBX	Data Center Bridging Capability Exchange
DHCP	Dynamic Host Configuration Protocol
DIMM	Dual In-line memory module
EOS*	Extensible Operating System*
ETS	Enhanced Transmission Selection
FPGAs	Field Programmable Gate Arrays
FRU	Field Replaceable Unit
GAMI	Generic Asset Management Interface
GRUB	GRand Unified Bootloader
HTTPS	Hypertext Transfer Protocol Secure
I^{2}C	I^{2}C, Inter-Integrated Circuit, synchronous multi master/slave serial computer bus
IEEE	Institute of Electrical and Electronics Engineers
Intel® RSD	Intel® Rack Scale Design
IPMB	Intelligent Platform Management Bus
IPMI	Intelligent Platform Management Interface

Term	Definition
iSCSI	Internet Small Computer Systems Interface
JBOD	just a bunch of disks
LLDP	Link Layer Discovery Protocol
LUI	Linux Utility Image
LVM	Logical volume management
LVM	Logical Volume Management
MAC	Media Access Control
MDR	Managed Data Region
Module	Physical component housing a blade or switch
NIC	Network Interface Controller
NTP	Network Time Protocol
NUC	Next Unit of Computing, miniature PC
NVM	Non-Volatile Memory
NVMe*	Non-Volatile Memory Express*
NVMe-oF*	NVMe over Fabrics*
OOB	Out-of-band
OxM	Original Equipment Manufacturer, Original Design Manufacturer
PCI	Peripheral Connect Interface
PCIe*	Peripheral Connect Interface express*
PFC	Priority Flow Control
PNC	Pooled NVMe Controller
POD	A physical collection of multiple racks
PODM	POD Manager
PEM	Privacy Enhanced Mail
PSME	Pooled System Management Engine
PXE	Pre-boot eXecution Environment
OS	Operating system
QoS	Quality of Service
RDMA	Remote Direct Memory Access
RMM	Rack Management Module
RPC	Remote Procedure Call
SLEDs	Single Large Expensive Disks
SMBIOS	System Management Basic Input/Output System
SSDP	Simple Service Discovery Protocol
tgt	target framework
TLV	Type Length Value
TLS	Transport Layer Security
ToR	Top of Rack
QSFP	Quad Small Form-factor Pluggable
USB	Universal Serial Bus
VLAN	Virtual Local Area Network
XML	Extensible Markup Language

1.6 References

Table 4. Reference Documents and Resources

Doc ID	Title	Location
337197	*Intel® Rack Scale Design (Intel® RSD) Conformance and Software Reference Kit Getting Started Guide Software v2.3.2*	*http://www.intel.com/intelRSD*
337198	*Intel® Rack Scale Design (Intel® RSD) POD Manager (PODM) Release Notes Software v2.3.2*	
337199	*Intel® Rack Scale Design (Intel® RSD) POD Manager (PODM) User Guide v2.3.2*	
337200	*Intel® Rack Scale Design (Intel® RSD) Pooled System Management Engine (PSME) Release Notes Software v2.3.2*	
337201	*Intel® Rack Scale Design (Intel® RSD) Firmware Extension Specification Software v2.3.2*	
337202	*Intel® Rack Scale Design (Intel® RSD) Storage Services API Specification Software v2.3.2*	
337203	*Intel® Rack Scale Design (Intel® RSD) Architecture Specification Software v2.3.2*	
337204	*Intel® Rack Scale Design (Intel® RSD) POD Manager (PODM) Representational State Transfer (RESTful) API Specification Software v2.3.2*	
337205	*Intel® Rack Scale Design (Intel® RSD) Rack Management Module (RMM) Representational State Transfer (RESTful) API Specification Software v2.3.2*	
337206	*Intel® Rack Scale Design (Intel® RSD) Generic Assets Management Interface (GAMI) API Software v2.3.2*	
337207	*Intel® Rack Scale Design (Intel® RSD) Pooled System Management Engine (PSME) Representational State Transfer (REST) API Specification Software v2.3.2*	
DSP0263	*Cloud Infrastructure Management Interface (CIMI) specification*	*https://www.dmtf.org/sites/default/files/standards/documents/DSP0263_1.0.1.pdf*
DSP8010	*Redfish* Schema v2016.3 (Compute)*	*https://www.dmtf.org/sites/default/files/standards/documents/DSP8010_2016.3.zip*
DSP8010	*Redfish* Schema v2017.3 Readme*	*https://www.dmtf.org/sites/default/files/DSP8010_2017.3.zip*
DSP0266	*Redfish* Scalable Platforms management API Specification Readme*	*https://www.dmtf.org/sites/default/files/DSP0266_1.4.0.pdf.*
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 v1.0.4*	*https://www.snia.org/sites/default/files/SMI/swordfish/v104/Swordfish_v1.0.4_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*
RFC2119	*Key Words for Use in RFCs to Indicate Requirement Levels, March 1997*	*https://ietf.org/rfc/rfc2119.txt*

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. 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 the 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 Basic Discovery

One of the key features of PSME software is gathering information about hardware from System Management Basic Input/Output System (SMBIOS) and exposing it through the 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.

Upon the start of the `psme-compute` service, compute Single Large Expensive Disks' (SLEDs') BMCs are queried for the information stored in the managed data region (MDR) as well as the power state and telemetry readings of the platform.

Note: The MDR SMBIOS region is filled by the Basic Input/Output System (BIOS) during boot time, and it may take a few minutes for this operation to complete.

For Intel® RSD software v2.3.2, the PSME supports platforms based on the Intel® Xeon® processor Scalable family with BIOS and BMCs that implement the *Intel® Rack Scale Design (Intel® RSD) Firmware Extension Specification* (refer to Table 4). For these platforms, the PSME is currently able to provide information about the following resources exposed by SMBIOS:

- Processors
- Memory DIMMs
- Network interfaces
- Local storage devices
- BIOS version
- Trusted platform modules
- Field Replaceable Unit (FRU) information of the system and its chassis
- Discrete Field Programmable Gate Arrays (FPGAs)

2.2 Deep Discovery

In the Intel® RSD v2.3.2 reference code, all discovery performed by the PSME is done out-of-band (OOB). This default OOB discovery enables all functionalities of the Intel® RSD PODM. This section describes how to build and configure the deprecated Deep Discovery feature, which was the legacy Intel® RSD 1.x way for performing in-band discovery. The Intel® RSD deprecated Deep Discovery legacy code may be removed in future releases.

2.2.1 Description

There are limitations to gathering hardware details from the BMC over an Intelligent Platform Management Interface (IPMI), especially if the platform is offline. For example, only data about non-system Peripheral Connect Interface (PCI)/ Universal Serial Bus (USB) devices are available from successful basic discovery. For this reason, the Deep Discovery feature has been introduced to gather extended blade data. This feature exposes the host details on the PSME REST API through a dedicated virtual local area network (VLAN) to the PODM.

Note: The PSME is not involved in Deep Discovery data propagation.

The following steps take place during Deep Discovery:

1. Basic discovery of the PSME provides resources (systems) to be deep discovered.
2. The PODM selects resources (systems) to run deep discovery on.
3. The selected resources (systems) are booted with a Linux* Utility Image (LUI).
4. The LUI exposes data to the PODM by use of the a REST API.
5. The PODM merges the data obtained from both the PSME and the LUI.

A full set of data is made available via the PODM REST API. For more information on implementing Deep Discovery, refer to the following steps and the "Building" section in the *Intel® RSD POD Manager User Guide*, Table 4.

2.2.2 Configuration

The LUI building procedure consists of two steps:

1. Compile and copy the `psme-rest-server` and `psme-compute-simulator` application to the `rootfs` overlay directory available under the PSME source package. The `rootfs` overlay directory, which already contains the python*/bash scripts responsible for gathering hardware data for the `psme-compute-simulator`, is available from the PSME source packages.
2. Prepare the Pre-boot eXecution Environment (PXE) bootable `bzImage`. The buildroot configuration file and kernel configuration file (Intel® RSD Software Development Vehicle platform) are available on the Intel® RSD source repository. The buildroot tool can be downloaded from *http://buildroot.org/downloads/buildroot-2017.02.tar.gz.*

Intel recommends performing both steps on the same machine running Ubuntu* v16 with at least 16 GB of free disk space. The installation requires access to public software repositories on the Internet. Confirm that the server network, firewall, and proxy configurations allow the appropriate server access. The dependencies required to compile the components and run all steps are listed in Section 3.1.2, Ubuntu v16.04 LTS for Deep Discovery.

The output `bzImage` should be copied to the `/opt/pod-manager/wildfly/discovery/` directory on the PODM before starting the pod-manager service.

The source directory relates to an uncompressed directory from the `psme-generic-src-*.tar.gz` or `psme-sdv-src-*.tar.gz` source package.

The source directory should be copied directly to a machine where the build will be made. If the source directory is not copied directly, ensure the sources were not modified in the process of copying. Copying the repository from a non-Linux based operating system could modify end line formatting of the LUI scripts (from LF to CRLF) and make them unusable.

2.2.2.1 Rootfs Overlay Directory Preparation

1. Copy the PSME source directory to a local directory and export its location path to the `$RACKSCALE` variable:

```
export RACKSCALE=/path/to/source/directory/
```

2. Export `rootfs` overlay directory path to `$ROOTFS` variable:

```
export ROOTFS=$RACKSCALE/lui/OS/rootfs
```

3. Compile the `psme-rest-server` and `psme-compute-simulator` targets using instructions from "Compilation" in Section 3.0, PSME Development Environment.
4. Copy the built binaries to the `$ROOTFS/usr/bin` directory:

```
# go to your build directory, for example
# cd $RACKSCALE/build.release
cp bin/psme-rest-server $ROOTFS/usr/bin
cp bin/psme-compute-simulator $ROOTFS/usr/bin
```

5. Copy the `psme-rest-server` shared library dependencies to `$ROOTFS/usr/lib`:

```
cp `ldd $ROOTFS/usr/bin/psme-rest-server | cut -d " " -f 3` $ROOTFS/usr/local/lib
```

6. Copy `psme-compute-simulator` shared library dependencies to `$ROOTFS/usr/lib` (any duplicates from the earlier step can be overwritten):

   a. Copy the shared library dependencies:

   ```
   cp `ldd $ROOTFS/usr/bin/psme-compute-simulator | cut -d " " -f 3`
   $ROOTFS/usr/local/lib
   ```

   b. Remove libraries which will be provided with `buildroot`:

   ```
   rm $ROOTFS/usr/local/lib/libc.so.*
   rm $ROOTFS/usr/local/lib/libpthread.so.*
   rm $ROOTFS/usr/local/lib/libresolv.so.*
   ```

7. Copy configuration files:

```
cp $RACKSCALE/agent-simulator/compute/configuration.json $ROOTFS/etc/psme/psme-
compute-simulator-configuration.json
cp $RACKSCALE/agent-simulator/compute/deep_discovery.xsd
$ROOTFS/etc/psme/deep_discovery.xsd
cp $RACKSCALE/application/configuration.json $ROOTFS/etc/psme/psme-rest-server-
configuration.json
```

8. Copy server certificates signed by authorized Certificate Authority (CA) to authenticate PSME Rest-Server service in the PODM - details in the Section 2.3, Security Features):

```
cp <your_directory_with_certificates>/server.crt $ROOTFS/etc/psme/certs/
cp <your_directory_with_certificates>/server.key $ROOTFS/etc/psme/certs/
```

9. Edit `$ROOTFS/etc/psme/psme-compute-simulator-configuration.json` file:

   a. Set input variable to `/usr/bin/deep_discovery.xml`:

   ```
   sed -i 's/\"input\"[ \t]*: .*/\"input\" :
   \"\/usr\/bin\/deep_discovery.xml\"\,/' $ROOTFS/etc/psme/psme-compute-simulator-
   configuration.json
   ```

   b. Set schema variable to `/etc/psme/deep_discovery.xsd`:

   ```
   sed -i 's/\"schema\"[ \t]*: .*/\"schema\":
   \"\/etc\/psme\/deep_discovery.xsd\"/' $ROOTFS/etc/psme/psme-compute-simulator-
   configuration.json
   ```

10. Edit `$ROOTFS/etc/psme/psme-rest-server-configuration.json file`:

   a. Set client-cert-required variable to false:

   ```
   sed -i 's/\"client-cert-required\"[ \t]*:[ \t]*true/\"client-cert-required\" :
   false/' $ROOTFS/etc/psme/psme-rest-server-configuration.json
   Set announce-interval-seconds variable to 60 seconds:
   sed -i 's/\"announce-interval-seconds\"[ \t]*:[ \t]*0/\"announce-interval-
   seconds\" : 60/' $ROOTFS/etc/psme/psme-rest-server-configuration.json
   ```

b. Set service-root-name variable to LUI Service Root:

```
sed -i 's/\"service-root-name\"[ \t]*:[ \t]*"PSME Service Root"/\"service-root-
name\": \"LUI Service Root"/' $ROOTFS/etc/psme/psme-rest-server-
configuration.json
```

11. Set executable attribute to `$ROOTFS` files :

```
chmod +x $ROOTFS/usr/bin/*
chmod +x $ROOTFS/etc/init.d/*
```

2.2.2.2 Building the LUI

1. Download and unpack `buildroot` to your local directory:

```
wget http://buildroot.org/downloads/buildroot-2017.02.tar.gz
tar -xf buildroot-2017.02.tar.gz
cd buildroot-2017.02
```

2. Create `buildroot` initial configuration:

```
make menuconfig
Choose <Exit>
Do you wish to save your new configuration?
Choose <Yes>
```

3. Copy (overwrite) LUI `buildroot` configuration:

```
cp $RACKSCALE/lui/config/buildroot.config .config
```

4. Create Kernel default configuration:

```
make linux-menuconfig
Choose <Exit>
```

5. Copy (overwrite) LUI Kernel configuration:

```
cp $RACKSCALE/lui/config/kernel.config output/build/linux-4.4.16/.config
```

Build image passing `ROOTFS` overlay path. This step may take 1-2 hours depending on your Internet connection and computing power.

```
make BR2_ROOTFS_OVERLAY=$ROOTFS
```

Optionally, specify the number of parallel jobs '`n`' for faster compilation using `'j'` flag:

```
make BR2_ROOTFS_OVERLAY=$ROOTFS -jn
```

6. Generated image is ready to be copied to PODM:

```
scp output/images/bzImage rsa@PODM:/opt/pod-manager/wildfly/discovery/
```

2.3 Security Features

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

2.3.1 Secure Communication over Transport Layer Security v2.3.2

Certificates described in this section can be easily generated, or you may use existing certificates – refer to the "Certificate Management" section in the *Intel® RSD PODM REST User Guide* (refer to Table 4).

Every PSME instance can be configured to only accept Hypertext Transfer Protocol Secure (HTTPS) connections secured with the Transport layer Security (TLS) protocol. The PSME implements mutual (bidirectional) 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.

Note: 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 certificate signed by an authorized CA (`server.crt`, to authenticate the PSME REST server service in the PODM) 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 PODM 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 CA file must be in Privacy Enhanced Mail (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^{2}C by the means of Intelligent Platform Management Bus (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 PODM 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 over Fabrics* (NVMe-oF*), RMM, and network agent must be configured in such a way (there is no I^{2}C connection to communicate with RMM).

5. The PSME allows for disabling client (the PODM) 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 a Network Time Protocol (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
   ```

b. 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.

c. Start the NTP Daemon:

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

d. Set initially local date and time:

```
ntpdate -u IP_OF_YOUR_LOCAL_NTP_SERVER
```

Use this command to manually synchronize time with the NTP server time. After initial sync the NTP client will automatically perform periodic sync with NTP server.

2.3.2 System Mode Management

A PSME instance, which manages Single Large Expensive Disks (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.

Note: The change will take action only after the system has been rebooted.

2.3.3 Trusted Platform Module Management

Intel® RSD SLEDs include Trusted Platform Module (TPM) v1.2 hardware modules. The PSME 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.

Note: The actual actions will be performed by the BIOS during the next boot.

2.3.3.1 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 are required to enable the cleared TPM.

2.4 Hot Swap

Hardware asset removal is automatically reflected on the REST API. To detect a 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 (for example, Non-Volatile Memory Express* (NVMe*) drives on Pooled NVMe Controller (PNC) this time may be longer as the PSME depends on the operating system or other software/hardware components.

Note: Hot Plug is quickly followed by a Hot Unplug, then the PSME may not be able to detect the device at all.

2.4.1 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)

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

2.4.2 Sled Hot Swap

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

There is a hardware/BIOS limitation regarding discovery of information about SLED assets. Since enumeration of resources happens at boot time, no changes will be detected after a hardware change (for example. reinsertion of a 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.

2.4.3 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.

2.5 Link Aggregation Group (LAG)

The Link Aggregation Group (LAG) functionality allows the ability to aggregate a number of physical links together to make a single high bandwidth data path. It mostly makes sense to configure LAG on interfaces between switches. In addition to the increased total bandwidth of the link between switches, LAG ensures redundancy between them. More details can be found in Section 4.2, Networking-Related Remarks.

2.6 Virtual LAN

The VLAN functionality allows the ability to manage VLANs dynamically using the PSME REST API. Detailed VLAN information can be found in Section 4.2, Networking-Related Remarks.

2.7 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 IPMB protocol.

2.8 Pooled NVMe Controller (PNC)

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

2.8.1 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.

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

2.8.2 Service Configuration

The `psme-pnc` requires the following dependencies to be installed in the OS:

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

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.

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 SLED).

To make the PSME PNC visible for Intel® RSD PODM in Simple Service Discovery Protocol (SSDP) but not Dynamic Host Configuration Protocol (DHCP), set the service root name in the `psme-rest-server-configuration.json` file on the management host as it is shown below:

```
"rest":{
    "service-root-name" : "PSME Service Root"
 }
```

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

2.9 iSCSI Out of Band Booting

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. UEFI). For Legacy, only PXE/iPXE is supported, and in this case system `BootSourceOverride` parameters should be PATCHed to PXE/Legacy. For UEFI, OOB iSCSI functionality is supported and `BootSourceOverride` parameters should be PATCHed to `RemoteDrive`/UEFI. This version also requires PATCHing the system's `NetworkDeviceFunction` with the correct data about an iSCSI Target.

2.9.1 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.

The BIOS does not have a separate boot option for `RemoteDrive` (iSCSI OOB). When the `BootOverrideTarget` is set to `RemoteDrive` through the PSME API, then the PSME Compute Agent:

- Sets BIOS's boot override to Hdd,
- In `NetworkDeviceFunction` sets `"DeviceEnabled"` field to "**true**",
- Copies iSCSI Target parameters from `NetworkDeviceFunction` to BMC's iSCSI OOB Parameters.
- When the `BootOverrideTarge`t is set to anything else except `RemoteDrive`, then PSME Compute Agent:
  - Sets BIOS's boot override to 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 BMC's iSCSI OOB Parameters. If PSME Compute Agent is restarted when `"DeviceEnabled"` is set to "**false**", then `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.

`"DeviceEnabled"` flag in `NetworkDeviceFunction` cannot be overridden, the flag is only modified by changing the `BootOverrideTarget`.

Table 5 shows how the PSME handles possible override options.

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

BootSourceOverrideEnabled Continuous with BootOverrideTarget:	BIOS Boot Override	OOB iSCSI Parameters	Will boot from
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 `BootOverrideTarget` set to `RemoteDrive`, and currently `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 Hdd.
- If previously `BootSourceOverrideEnabled` was set to Continuous with `BootOverrideTarget` set to `RemoteDrive`, and currently `BootSourceOverrideEnabled` is set to Once with `BootOverrideTarget` set to Hdd, then after a `BootSourceOverrideEnabled` Once time-out, or a second power cycle, the system will boot from Hdd.
- If previously `BootSourceOverrideEnabled` was set to Continuous with `BootOverrideTarget` set to Hdd, and currently `BootSourceOverrideEnabled` is set to Once with `BootOverrideTarget` set to `RemoteDrive`, then after a `BootSourceOverrideEnabled` Once time-out, or a second power cycle, the system will boot from `RemoteDrive`.

In aforementioned cases, a new PATCH request setting `BootSourceOverrideEnabled` must be set to Once/Continuous to alter current boot order. Table 6 shows how the PSME handles possible override options.

Table 6. How the PSME Handles Possible BootSourceOverrideEnabled Once Options

BootSourceOverride Enabled Once with BootOverrideTarget:	Previous BootSourceOverride Enabled Continuous with BootOverrideTarget:	BIOS Boot Override	OOB iSCSI Parameters	Will boot from	After BootSourceOverrideEnabled 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
RemoteDrive	RemoteDrive	Hdd	set from NetworkDeviceFunction	RemoteDrive	RemoteDrive

2.9.2 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, 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.

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

2.9.3 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"`.

2.9.4 Intel® RSD Software Development Vehicle Limitations

- `InitiatorName` and `PrimaryTargetName` shall have maximum 200 characters
- `CHAPUsername` and `MutualCHAPUsername` shall have maximum 32 characters

- `CHAPSecret` and `MutualSecret` shall have between 12 and 16 characters
- `PrimaryLUN` field can be "0" when read from cleared BMC's iSCSI OOB Parameters

2.9.5 Known Issues in Intel® RSD Software Development Vehicle

PSME Compute Agent reads `BootSourceOverrideEnabled`, `BootOverrideTarget`, and `BootOverrideMode` fields from BMC. When these fields are set via PSME, the BMC may display their real values only for a limited time period (60s-120s). After this time BMC/PSME will return default values of these fields, but BIOS will remember previously set configuration.

If only `BootSourceOverrideEnabled` and `BootOverrideTarget` fields are patched, but `BootOverrideMode` field is omitted, then `BootOverrideMode` field is reconfigured with the value currently returned by BMC/PSME.

BMC by default returns "Legacy" mode after the time-out mentioned in the first paragraph. Remember also to patch field `BootOverrideMode`, if you are again patching `BootSourceOverrideEnabled` and `BootOverrideTarget` fields, and you intend to boot in `"UEFI"` mode.

2.10 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

2.10.1 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.

2.10.1.1 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`

The list of available properties, their description and allowed values, is available in the metadata for `MetricDefinition`. 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 number of seconds, refer to Table 4).

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

- `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.

2.10.1.2 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),
- `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).

2.10.1.3 Config File Example

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

2.10.1.4 Sample Computations

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

Figure 1. Average Equation

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

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

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

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.

2.10.2 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.

2.11 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 this 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.

2.11.1 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 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 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.

2.12 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

2.13 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.

2.13.1 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.

2.13.1.1 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 operating system:

```
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.

2.13.1.2 Service Configuration and Detection of RDMA Interfaces

The default `psme-nvme` configuration file can be found in the PSME source tarball (in `agent/nvme/configuration.json`). 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 those interfaces are exposed on the APIs that were created by one of the listed drives.

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.

2.13.1.3 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

Note: A "zone" is a logical entity connecting volumes available over the fabric (targets) and hosts (initiators). We need a separate zone for each initiator and its volumes. Also, there can be multiple zones.

2.13.2 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 2.13.2.3, Running the PSME NVMe Discovery Agent on a Target Host.

2.13.2.1 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 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
```

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

2.13.2.2 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"
        }
    ]
}
```

2.13.2.3 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 RESTful Server should specify a different interface because these services should be available on separate IP addresses.

2.13.2.4 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

2.13.3 Provisioning Initiator Hosts

In this Intel® RSD Software v2.3.2 release, an example implementation designed according to the instructions below has been released as the “NVMe-Wheel”. Refer to NVMe-Rev. 003US ReadMe v2.3.2. provided with the NVMe-Wheel Instructions for installing and running to the tool. The *NVMe-Rev. 003US ReadMe v2.3.2* can be found on *Intel.com*.

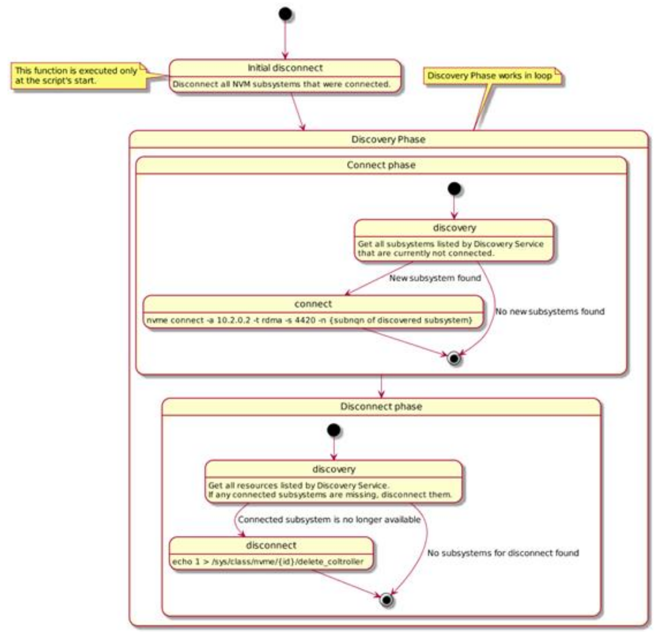
Each initiator host must poll the Discovery Service to ensure that its connections to remote volumes are up-to-date. A tool must be created to perform these actions. This section describes the proposed operation of a tool installed on an initiator host. The tool can be a CRON job script which wraps up the `nvme-cli` ("NVMe management command line interface") utility, which can be obtained from its GitHub* repository at *https://github.com/linux-nvme/nvme-cli*.

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

As presented in the UML diagram shown in Figure 3, 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. A database entry should also contain the sysfs identifier assigned to the subsystem, which can be found in `/sys/class/nvme/`. If the Discovery Service response was missing some subsystems which were previously attached by the script, it would disconnect them using the `sysfs` interface 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
No new subsystems found
connect
nvme connect -a 10.2.0.2 -t rdma -s 4420 -n {subnqn of discovered subsystem}
Disconnect phase
discovery
Get all resources listed by Discovery Service.
If any connected subsystems are missing, disconnect them.
Connected subsystem is no longer available
No subsystems for disconnect found
disconnect
echo 1 > /sys/class/nvme/{id}/delete_coltroller

2.13.4 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 2.14, 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.

Feature	Parameter	Value	Description
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.

2.14 Quality of Service

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 others traffic on the same link.

QoS allows the user to configure the network capability to provide better service to selected network traffic over Ethernet. RoCEv2-based NVMe-oF is one of the traffic types that requires 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 Institute of Electrical and Electronics Engineers (IEEE) 802.1Qbb specification. 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 IEEE 802.1Qaz specification. It enables dividing traffic according to its 802.1p priority into different priority groups and configure 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.

Note: Refer to Table 4 for document *IEEE Std 802.1Q – 2014,* which contains all the specifications.

2.14.1 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 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.

2.14.2 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" (for example, Ethernet25/2, Ethernet25/3, Ethernet25/4) if a Quad Small Form-factor Pluggable (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 mode, Application Protocol Type Length Value (TLV) is not broadcast (in LLDP frames) to the endpoints by the Arista switch. Only PFC and ETS parameters are sent. DCBX Converged Enhanced Ethernet (CEE) mode must be enabled to broadcast Application Protocol mapping as well.

§

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

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

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
```

3.1.2 Ubuntu v16.04 LTS for Deep Discovery

Enter the command shown in this section to install just the Ubuntu* packages required to compile PSME binaries for the deprecated Deep Discovery feature. It also installs the packages required to run `buildroot`.

Note: This command requires root privileges.

```
apt install bc wget unzip cmake python libudev-dev libgcrypt-dev \
libxml++2.6-dev libblkid-dev liblvm2-dev libnl-3-dev libnl-route-3-dev \
libnl-genl-3-dev libmicrohttpd-dev libcurl4-gnutls-dev libncurses-dev \
libibverbs-dev librdmacm-dev
```

3.1.3 CentOS* 7

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

The following command should be used to install all 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

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

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

An Internet connection with proxy servers set is required if http, https, and ftp protocols are needed. If this step is omitted, all required third-party components are downloaded automatically during the build.

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: You can 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 Cross Compilation Process for ARM

The `buildroot` environment must be built before the PSME cross compilation for ARM*. For detailed instruction on how to compile and run PSME on MYB-IMX28X reference board, see Appendix B, MYB-IMX28X Reference Board Configuration for PSME. Cross compilation commands to be run in the main PSME directory, as follows:

```
mkdir build.arm
cd build.arm
cmake -DCMAKE_TOOLCHAIN_FILE={PSME_source_root_dir}/cmake/Platform/Linux-buildroot-
arm.cmake ..
make -j8 psme-rest-server
```

3.4 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
```

§

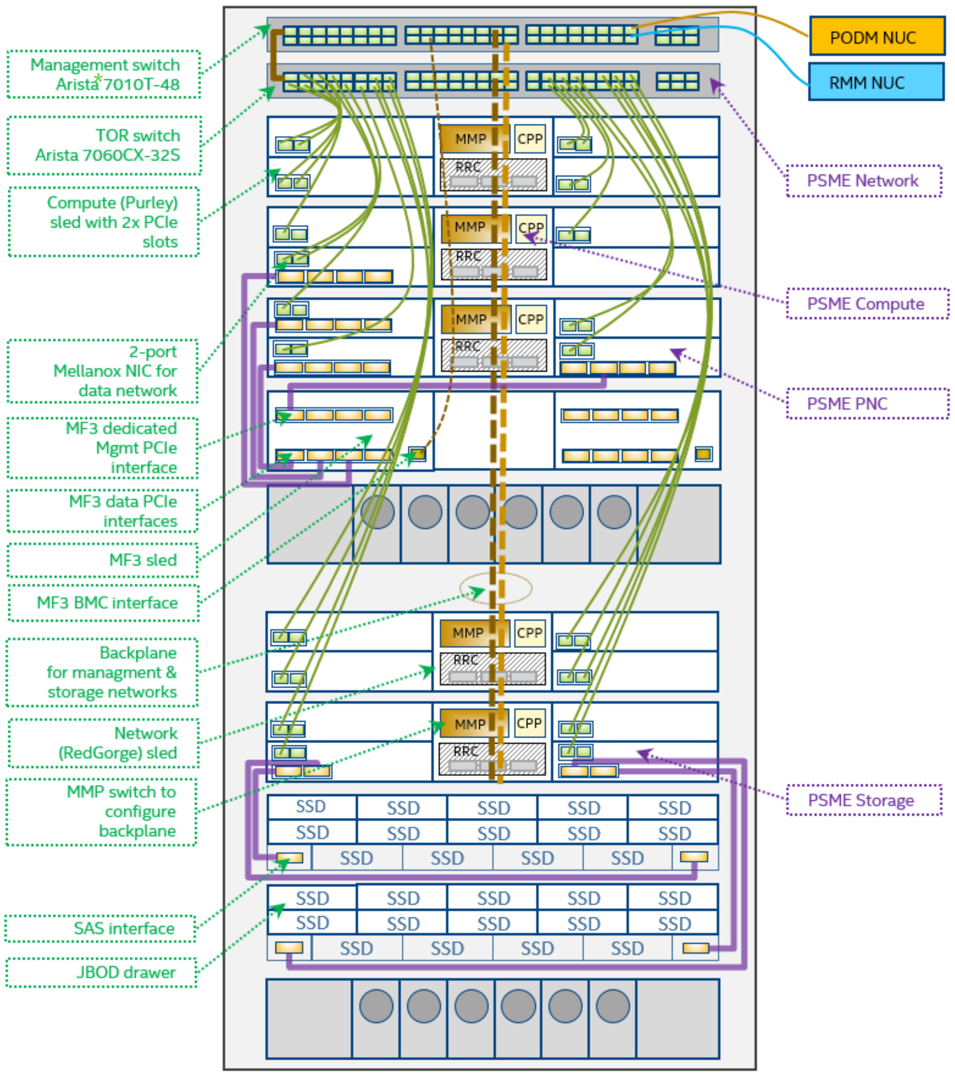
4.0 Intel® RSD Rack Network Configuration

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.

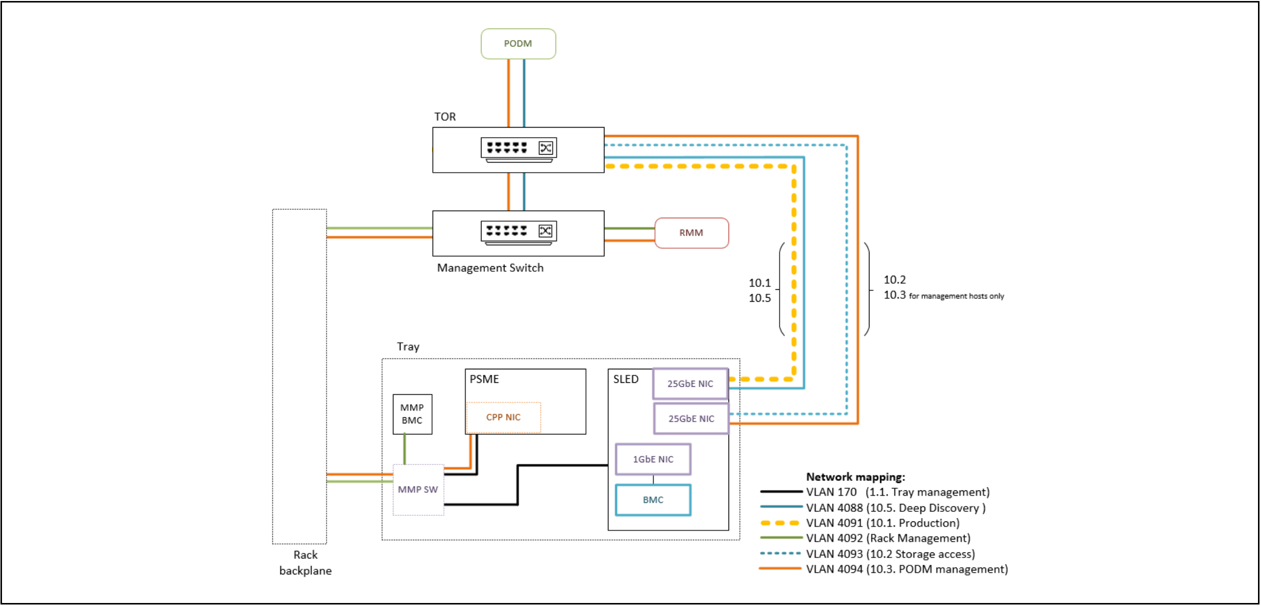
Figure 4. Intel® RSD Software Development Vehicle Reference Platform


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

4.1.3.1 ToR Switch VLAN

Figure 6 and Figure 7 show ToR switch VLAN configurations.

Figure 6. ToR Switch VLAN Configuration

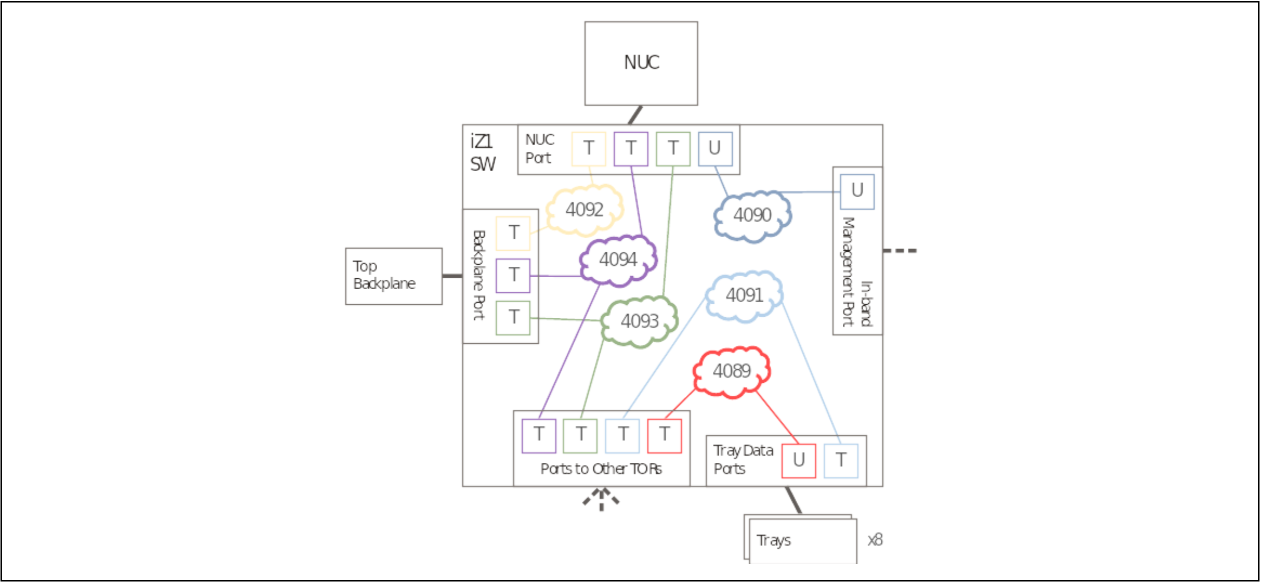
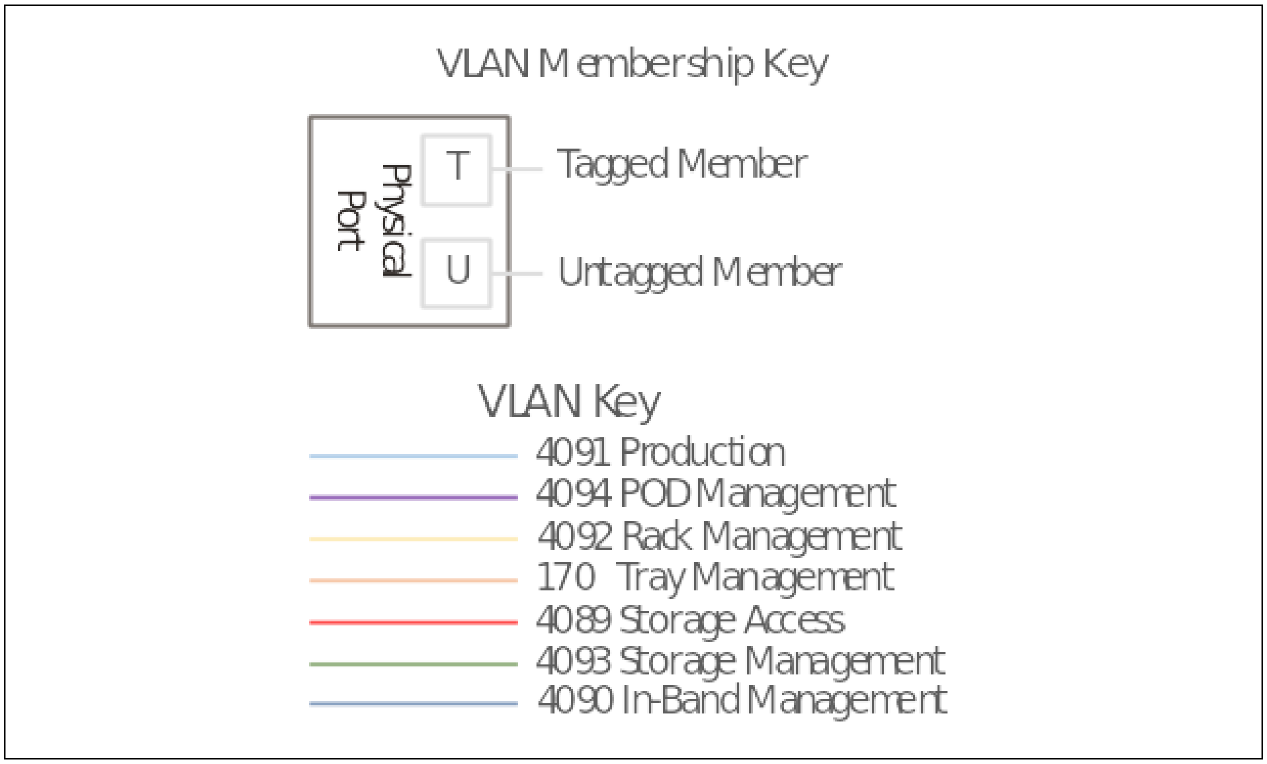


Figure 7. ToR Switch VLAN Configuration Membership Key



4.1.3.2 MBP VLAN

Figure 8 and Figure 9 show the MBP VLAN configurations.

Figure 8. MBP VLAN Configuration

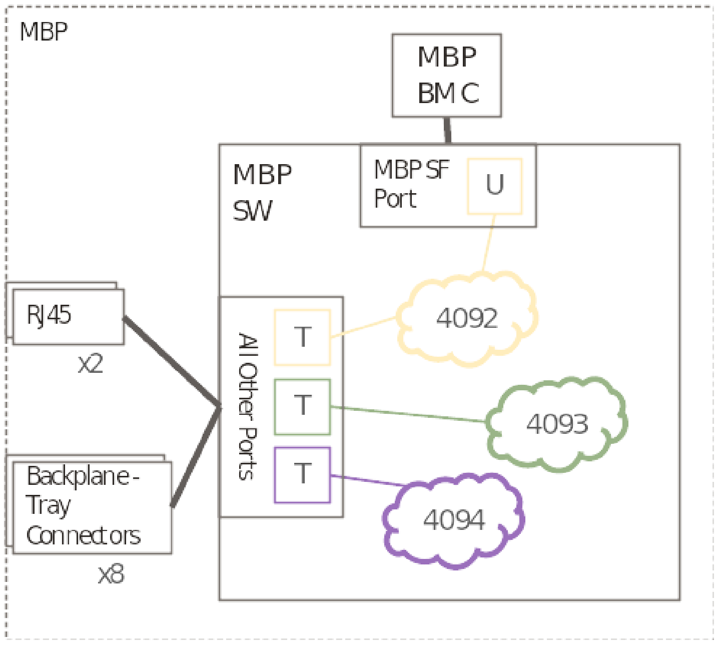
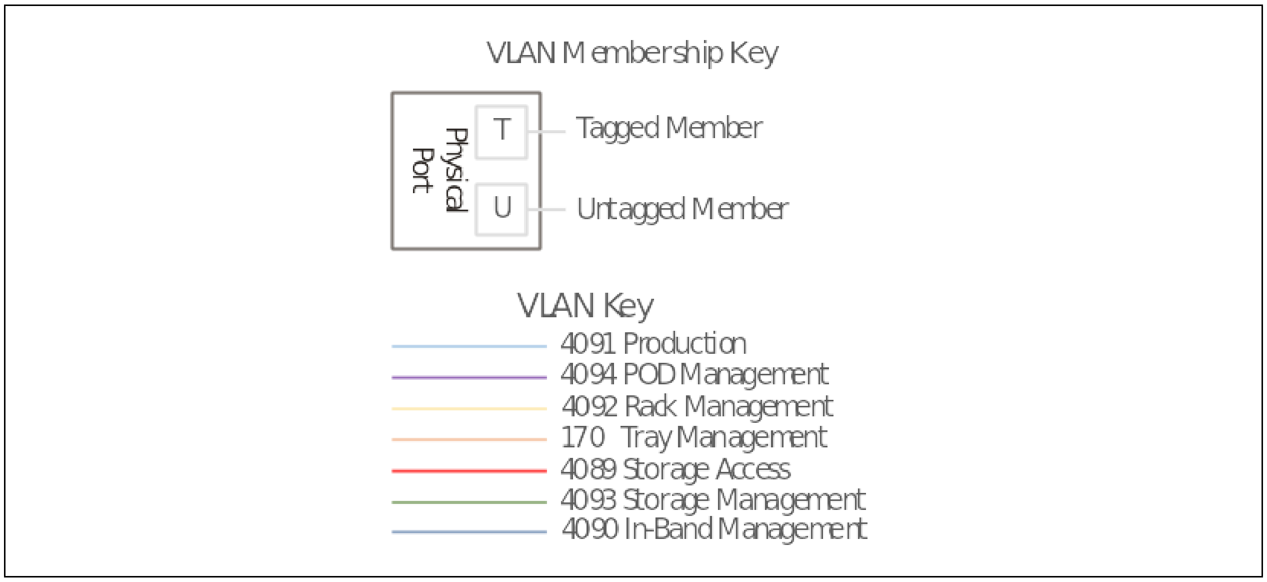


Figure 9. MBP VLAN Configuration Membership Key



4.1.4 Intel® RSD Software Development Vehicle Drawer Network

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

4.1.4.1 Vehicle Drawer Network for MMP VLAN

Figure 10 and Figure 11 show the MMP VLAN configurations.

Figure 10. Vehicle Drawer Network for MMP VLAN Configuration

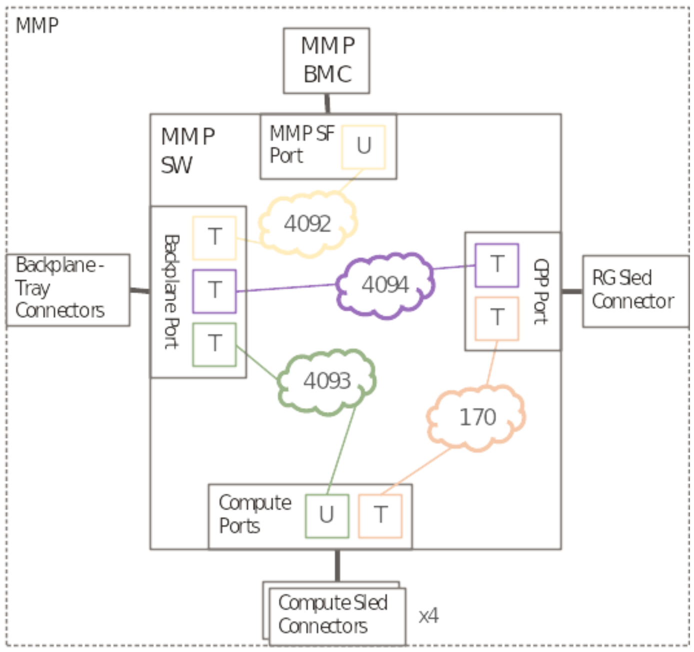
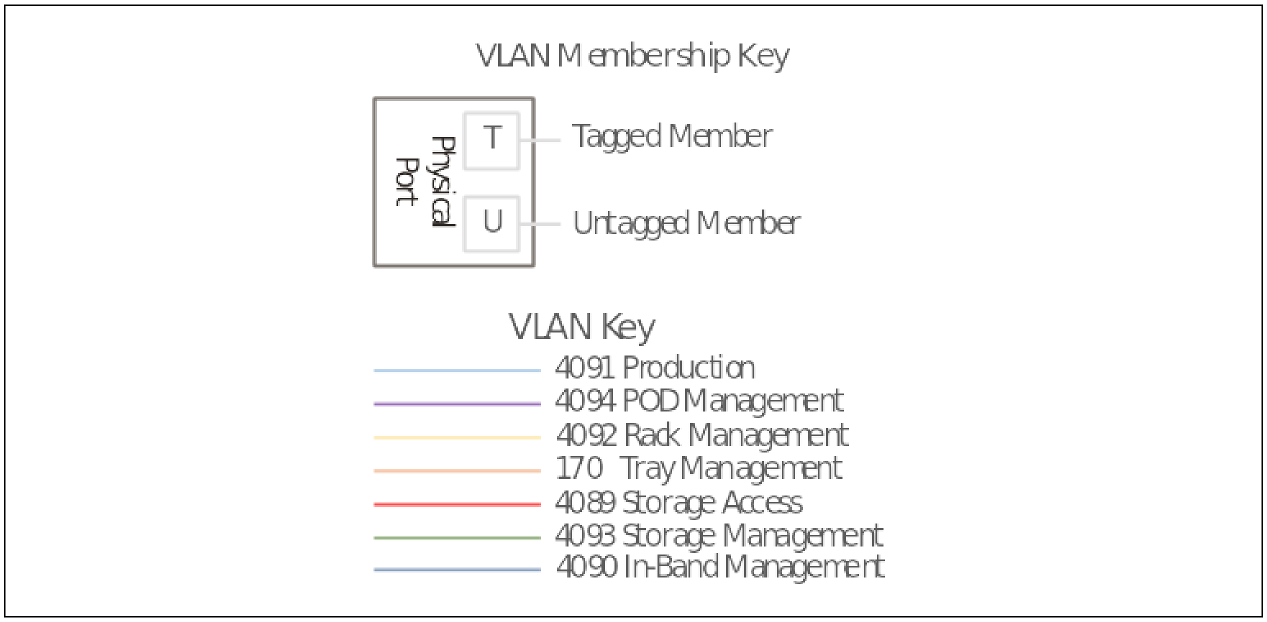


Figure 11. Vehicle Drawer Network for MMP VLAN Membership Key



4.2 Networking-Related Remarks

This sections contains additional instructions, limitations and other remarks related to Intel® RSD network management.

4.2.1 Virtual LAN Configuration

To configure VLANs on Arista* switch ports through the PSME API, the following special requirements need to be satisfied:

- Adding/deleting untagged VLANs is not supported.
- 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.
- Any modification of VLANs on a given port through the PSME API can be verified on the switch command line interface (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
  ```
- A VLAN cannot be disabled on a port. VLANs can be added or deleted only. The `VLANEnable` API attribute is always returned as "`true`". The REST API or GAMI API does not support changing the `VLANEnable` API attribute.

- 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 GAMI API, the supplied Name is discarded. Trying to change that attribute from REST API or GAMI API is not supported.

4.2.2 Link Aggregation (LAG)

Link aggregations (LAGs) are not supported by the PSME network agent running on an Arista switch.

4.2.3 Port Parameters Configuration Limitations

- Only setting of administrative state is supported.
- Reading port attributes are supported for the following:
  - Administrative and operational port state
  - Link speed

4.2.4 Mesh Topology

With the introduction of the ToR switch, the data network Mesh Topology feature is no longer supported.

4.2.5 Access Control List (ACL)

Access Control Lists (ACLs) are not supported by the PSME network agent on the Arista switch.

4.2.6 Static MAC

Static MACs are not supported by the PSME network agent.

§

5.0 Intel® RSD Drawer Configuration

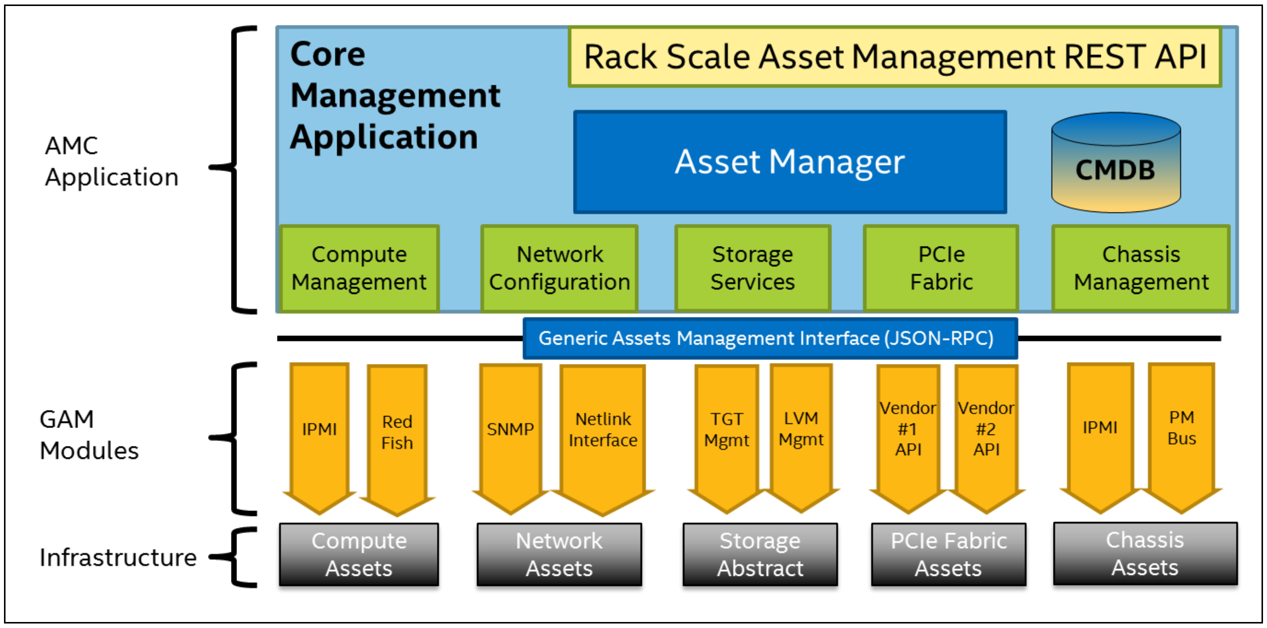
5.1 Hardware Configuration

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 consist 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 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 Storage Agent	`psme-storage`
PSME PNC Agent	`psme-pnc`
PSME RMM Agent	`psme-rmm`
PSME NVMe Agent	`psme-nvme`
PSME Compute Simulator	`psme-compute-simulator`
PSME NVMe Discovery agent	`psme-nvme-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 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 NVMe Agent	`<PSME root>/agent/nvme/configuration.json`
PSME NVMe Discovery Agent	`<PSME root>/agent/nvme-discovery/configuration.json`
PSME Compute Simulator	`<PSME root>/agent-simulator/compute/configuration.json`

5.3 PSME Storage Services

5.3.1 Prerequisites

A PC with Linux OS, Ubuntu* v16.04 operating system 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. 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. Remove a bootable tag from a given LogicalVolume:

```
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 v2.3.2, it is neither required nor recommended to manually create iSCSI Targets. 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 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

5.4.1 Prerequisites

A PC with Ubuntu* Linux* v16.04 operating system 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 F F.2, Installing `CyMUX`.

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

5.5 PSME Pooled NVMe Controller

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

5.5.1 Prerequisites

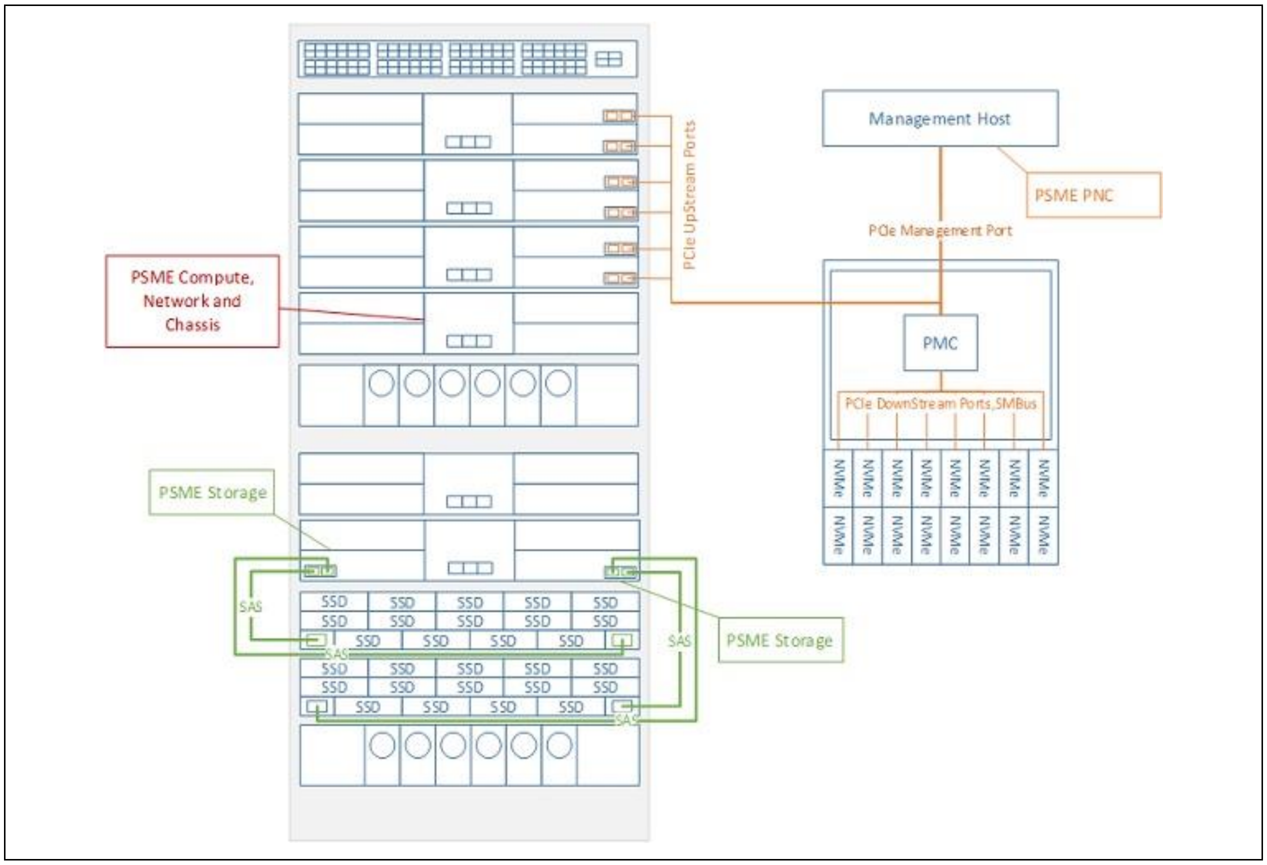
As a prerequisite to using the PSME PNC, you should have a setup with the Intel® RSD Software Development Vehicle PCIe Switch with attached NVMe drives. The management host should have a Linux kernel version supporting PCIe device hot swap.

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.

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 operating system 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 operating system 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 operating system:

```
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>
```

5.6 Rack Management Module (RMM)

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

5.6.1 Prerequisites

A PC with Linux OS, Ubuntu v16.04 operating system is recommended. The hostname must be set to `rmm-*`.

5.6.2 Installation

To install RMM, follow the installation steps included in PSME Software Installation from Packages.

5.6.3 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
   ```

§

Appendix A IPMI Commands

Appendix A 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 number in commands and bit numbers in bitmasks:

```
0-3 : Sled BMC 0-3
4   : MMP BMC
5   : RRC CPP (not used by PSME SDV 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 B MYB-IMX28X Reference Board Configuration for PSME

Appendix B provides the MYB-IMX28X reference board PSME Configuration.

B.1 MYB-IMX28X Board

The MYB-IM28X is the reference board used for PSME cross compilation for ARM.

B.1.1 Board Features

- MYB-IMX28X Evalboard from MYIR* producer
- ARM926EJ-S* (ARMv5*), Freescale* i.MX287 processor
- Frequency: 454 MHz
- 128 MB DDR2 SDRAM
- 256 MB NAND Flash
- 2xETH 10/100 MB
- Support USB HOST, USB OTG, UART, I^{2}C, SPI, CAN, ADC, and others
- Dimensions: 62 mm x 38 mm

B.1.2 Booting Modes

The processor boot mode can be selected from the BM5-BM0 pins and the LCD_RS pin. For more details, refer to the MYC-IMX28X* datasheet and/or i.MX28x Reference Manual. Table 15 lists the boot modes. Table 16 shows the corresponding boot mode pins with pull-up (1) or pull-down (0) resistors.

Table 15. Boot Modes

Boot mode	BM5	BM4	BM2	BM1	BM0	LCD_RS
USB0 (default)	X	0	0	0	0	1
I^{2}C	0	0	0	0	1	1
NAND	0	0	1	0	0	1
SD Card	0	1	0	0	1	1
JTAG	0	0	1	1	0	1
SPI Flash	0	0	0	1	0	1

Note: In Table 15, *X* means that the state of the pin doesn't matter for that boot mode.

Table 16. Boot Mode Pins

Boot mode	BM5	BM4	BM3	BM2	BM1	BM0	LCD_RS
47k pull-up	x	R11	R12	R13	R14	R15	R10
47k pull-down	R16	R17	R18	R19	R20	R21	x

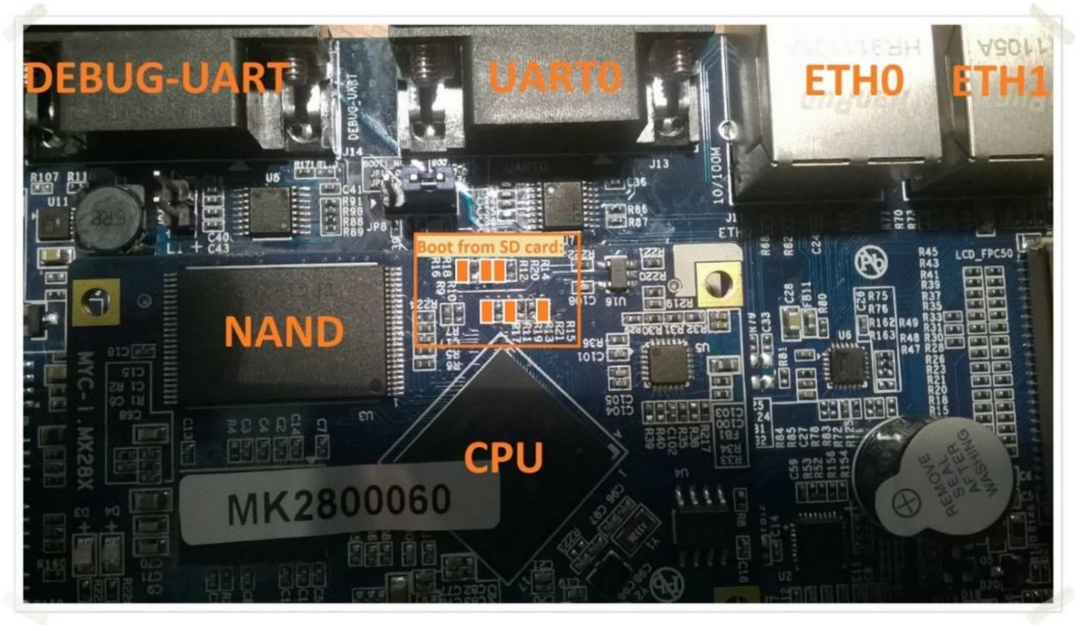
Note: In Table 16, *X* means that it doesn't matter for that pin whether any resistor is soldered to the board.

B.1.3 Booting from SD Card

Booting from the SD card on new evaluation boards requires changing resistors on the BMx pins. Figure 14 shows the main components on the reference board.

Note: The default settings force the processor to boot from NAND/USB0 (based on JP8 jumper). The JP8 jumper should be removed or may be set to **BOOT2** when using SD card boot mode. The JP10 jumper must be removed.

Figure 14. Reference Board Main Components



B.2 Build System Configuration

This section covers the Preparation and Kernel Configuration.

B.2.1 Environment Preparation

1. Download the required packages for `buildroot` and PSME compilation:

```
sudo apt install bc g++ wget cmake unzip python ncurses-dev \
pkg-config libcurl4-gnutls-dev libmicrohttpd-dev
```

2. Download `buildroot`:

```
wget http://buildroot.org/downloads/buildroot-2017.11.2.tar.gz
tar -xf buildroot-2017.11.2.tar.gz
```

3. Change directory to the `buildroot` directory:

```
cd buildroot-2017.11.2
```

B.2.2 Kernel Configuration

1. Configure `buildroot` using menuconfig:

```
make menuconfig
```

2. Edit the following sections:

   a. Target options

```
Target Architecture (ARM (little endian))
Target Binary Format (ELF)
Target Architecture Variant (arm926t)
Target ABI (EABI)
Floating point strategy (Soft float)
ARM instruction set (ARM)
```

   b. Build options

```
No need to change
```

   c. Toolchain

```
Toolchain type (buildroot toolchain)
*** Toolchain buildroot Options ***
C library (glibc
*** Kernel Header Options ***
Kernel Headers (Linux 4.11.x kernel headers)
*** Binutils Options ***
Binutils Version (binutils 2.27)
*** GCC Options ***
GCC compiler Version (gcc 5.x)
[*] Enable C++ support
*** Toolchain Generic Options ***
[*] Enable MMU support
```

   d. System configuration

```
(psme) System hostname
(Welcome to PSME) System banner
[*] Run a getty (login prompt) after boot
getty options --->
        (ttyAMA0) TTY port
        Baudrate (115200)
        (vt100) TERM environment variable
[*] remount root filesystem read-write during boot
```

   e. Kernel

```
[*] Linux Kernel
    Kernel version (Custom version)
(4.11.4) Kernel version
    Kernel configuration (Using an in-tree defconfig file)
(mxs) Defconfig name
Kernel binary format (zImage)
[*] Build a Device Tree Block (DTB)
    Device tree source (Use a device tree present in the kernel)
(imx28-evk) Device Tree Source file names
[] Install kernel image to /boot in target
```

   f. Target packages

Caution: Enable `OpenSSH` only for development.

```
-*- BusyBox
    Libraries --->
        Networking --->
            [*] libcurl
            [*]   curl binary
            [*] libmicrohttpd
            [*]   https support
    Networking application --->
        [*] openssh
```

g. Filesystem images

```
[*] tar the root filesystem
```

h. Bootloaders

```
[*] U-Boot
(mx28evk) U-Boot board name
    U-Boot Version (Custom version)
    (2017.05) U-Boot version
    U-Boot binary format --->
        [*] u-boot.sb
```

3. Exit from menuconfig:

```
< Save >
< Exit >
```

B.3 Image Creation

B.3.1 Build Images

Build the system image, optionally specifying the number of parallel jobs '`n`':

```
$ make [-jn]
```

Table 17 shows the generated directories layout under `buildroot`. Table 18 shows the minimum required files after builds are in the output/images directory.

Table 17. Directories Layout

Directory	Description
`output/build`	Binaries and libraries for the host or the target
`output/host`	Host tools for target developing like toolchain, tools
`output/images`	Target images like kernel, bootloader,*.dtb, root file systems
`output/target`	Not packed root file system for the `target, useful` for PSME build system in cross compilation mode (using `cmake/Platform/Linux-buildroot-arm.cmake file`)

Table 18. Minimum Required Files

File	Description
`imx28-evk.dtb`	Device Tree Blob for i.MX28x* Evaluation Board like MYB-IM28X*
`rootfs.tar`	Root file system tarbal for the target
`u-boot.sb`	Bootloader. Serial binary format (*.sb)is required for the Freescale* i.MX processors
`zImage`	A compressed version of the Linux kernel image that is self-extracting

B.3.2 PSME Cross Compilation

1. Set environment variable pointing to the `buildroot` directory:

```
export BUILDROOT_DIR=/path/to/buildroot-2017.11.2
```

2. Navigate to the psme/SW directory.
3. Create a directory for the build:

```
$ mkdir build && cd build
```

4. Execute `CMake` with the proper configuration. Use the full path of the configuration file, as shown:

```
$ cmake -DCMAKE_TOOLCHAIN_FILE={...}/psme/SW/cmake/Platform/Linux-buildroot-
arm.cmake ..
```

5. Compile the project, optionally specifying the number of parallel jobs '`n`':

```
$ make [-jn]
```

B.3.3 Bootloader Configuration

Bootloader is used to initialize memory, basic peripherals, and clocks, and load the necessary images and boot kernel. Add a header to the bootloader image for Freescale* i.MX processors for SD cards:

```
./output/build/uboot-2017.05/tools/mxsboot sd ./output/images/u-boot.sb
./output/images/u-boot.sd
```

B.4 Linux Image Configuration

B.4.1 SD Card Layout

1. Locate your SD card (typically, it is located in `/dev/sdd, /dev/sdb or /dev/mmcblk)`:

```
fdisk -l
```

2. Create three SD card partitions (such as `/dev/sdd`)—one for the bootloader, one for the kernel image, and one for root file system:

```
fdisk /dev/sdd
Command: o
Command: n
    Select: p
    Partition number: 1
    First sector:
    Last sector: +8M
Command: n
    Select: p
    Partition number: 2
    First sector:
    Last sector: +8M
Command: n
    Select: p
    Partition number: 3
    First sector:
    Last sector:
Command: t
    Partition number: 1
    Hex code: 53
Command: t
    Partition number: 2
    Hex code: b
Command: t
    Partition number: 3
    Hex code: 83
Command: a
    Partition number: 1
Command: w
```

3. Check disk partition schema (such as for 1 GB `/dev/sdd`):

```
fdisk /dev/sdd
Command: p
Device     Boot Start     End  Sectors  Size Id Type
/dev/sdd1  *      2048    18431    16384    8M 53 OnTrack DM6 Aux3
```

```
/dev/sdd2       18432   34815   16384    8M  b W95 FAT32
/dev/sdd3       34816 1888255 1853440 905M 83 Linux

Command: q
```

4. Preparing partitions:
   a. The second partition is used for the kernel image and Device Tree Blob. It must be formatted as FAT. u-boot automatically detects the partition and looks for the Device Tree Blob image (*.dtb) to load, the kernel image (`zImage`) to load, and boot kernel. (If partitions are already mounted, unmount them first):

   ```
   mkfs.vfat /dev/sdd2
   ```

   b. The third partition is used as root file system:

   ```
   mkfs.ext2 /dev/sdd3
   ```

   c. Write bootloader to bootloader partition:

   ```
   dd if=output/images/u-boot.sd of=/dev/sdd1
   ```

   d. Copy the kernel image and Device Tree Blob to the boot partition:

   ```
   mkdir /mnt/boot
   mount /dev/sdd2 /mnt/boot
   cp ./output/images/imx28-evk.dtb /mnt/boot
   cp ./output/images/zImage /mnt/boot
   umount /mnt/boot
   ```

   e. Unpack the root file system tarball to the target `rootfs` partition:

   ```
   mkdir /mnt/rootfs
   mount /dev/sdd3 /mnt/rootfs
   tar xvf ./output/images/rootfs.tar -C /mnt/rootfs
   umount /mnt/rootfs
   ```

   f. Prepare for networking

   ```
   Mount the SD card:
   mount /dev/sdd3 /mnt/rootfs

   Edit the interface file:
   vi /mnt/rootfs/etc/network/interfaces

   Setup the Ethernet interfaces to DHCP:
       auto eth0
       iface eth0 inet dhcp
           hwaddress ether 00:04:00:AA:BB:CC
       auto eth1
       iface eth1 inet dhcp
           hwaddress ether 00:04:00:DD:EE:FF
   Unmount the root file system:
   umount /mnt/rootfs
   ```

B.5 Evaluation Board Configuration

Figure 15 shows the reference board connectors.

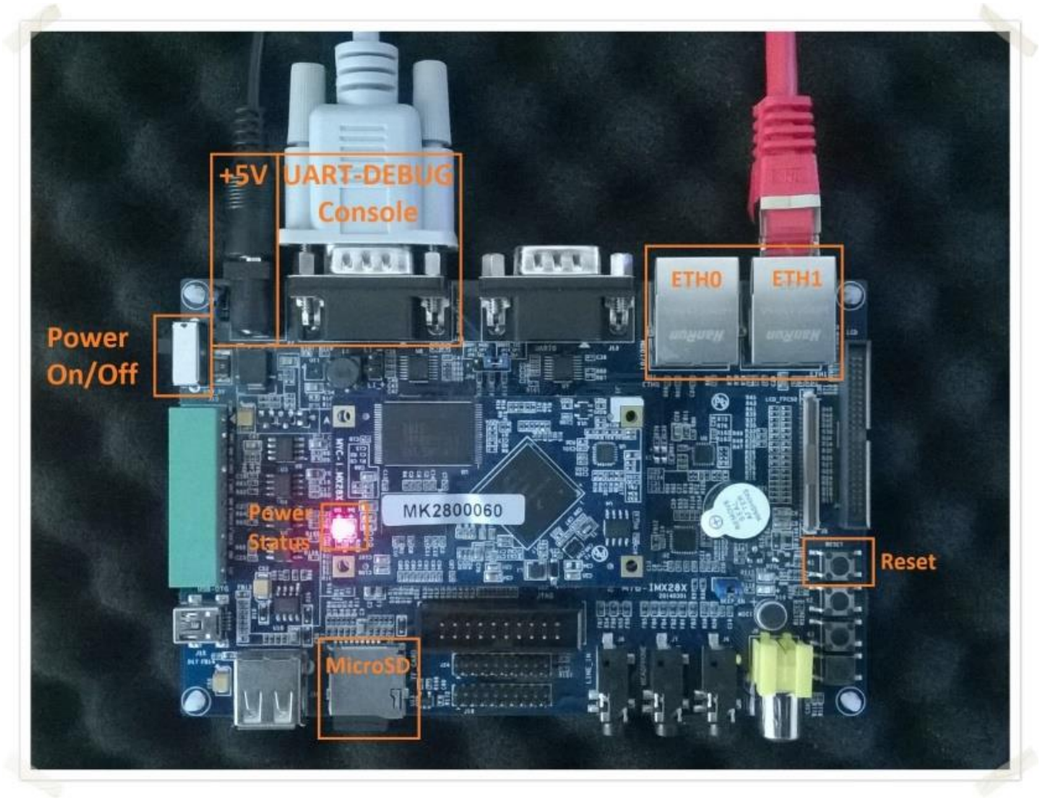
B.5.1 Booting

1. Insert the MicroSD* card to the MicroSD slot.

   Note: Not all cards are supported, and, in case of errors, use a different MicroSD card.

2. Connect the RS232 cable to the UART-DEBUG port to be used for the console.
3. Connect the Ethernet cable to one of the two Ethernet ports.
4. Connect the +5V DC power supply cable.
5. Power on the board.

Figure 15. Reference Board Connectors



B.5.2 Serial Console Settings

- 115200 baud rate
- 1 bit stop
- No parity
- No hardware flow control
- No software flow control

§

Appendix C ToR Switch Configuration

Appendix C contains the instructions for configuration of the ToR switches in an Intel® RSD Software Development Vehicle rack.

C.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 has the static configuration explained later in this section. The second one is 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*, and the data network switch is Arista 7060CX*.

C.2 Configuration of Management Network ToR with Command-Line Interface (CLI)

The configuration assumes the following ports of the management ToR switch are used to connect different components:

- **Port 1:** NUC with PODM
- **Port 3:** Control Module (CM)
- **Port 5:** RMM
- **Port 48:** Management port of the data network ToR switch
- **Port 49:** Ethernet port 33 of the data network ToR switch

Table 19 shows the management configuration and ports.

Table 19. Management 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 CON 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 serial line and connection speed):
   a. Use the correct serial port on the PC:
      1. Set the connection speed to 9600
      2. Select **Serial**.
      3. Open a connection and wait for the ToR to respond.

3. Log into the ToR system as '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. For example, to create vlan 4090, type `vlan 4090`, and then type `exit`.
8. To verify VLAN creation, type `show vlan`.
9. 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. 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 configuration 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 configuration mode.

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

C.3 Configuration Process for the Data Network ToR.

The configuration process assumes the first eight QSFP ports of the switch are already connected to all SLEDs in the rack using break-out cables, like 4x25 Gbps.

Note: The configuration does 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.

Access the CLI of this ToR switch in the same way as described in the previous section.

1. Login as 'admin' user.
2. Enter privileged mode using the `enable` command.
3. Enter 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 SLED:

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

   a. Configure VLANs for storage network (10.2) on interfaces connected to the second Mellanox's interface on SLED:

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

   b. Configure VLANs for management network (10.3) on interfaces connected to SLEDs working as management hosts (for example, 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. Save the currently running configuration to the startup configuration.
8. Copy `running-config startup-config`.

§

Appendix D Drawer Hardware Configuration

Appendix D provides the steps for establishing networking to an Intel® RSD Software Development Vehicle drawer through the Control Module. Refer to these instructions to address issues with communication over I^2C. If there are I^2C communication issues on the Intel® RSD Software Development Vehicle platform, ensure that the appropriate rework is done.

D.1 Recommended Hardware Configuration

D.1.1 Rack Setup Configuration

- x1 NUC
- x1 TOR
- x1 Fabric module
- x4 SLEDs based on the Intel® Xeon® processor Scalable family

D.2 Control Module

D.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.

D.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 show` in the CM console, as shown:
   ```
   > 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 the screen session - Ctrl- a – d.
   d. Kill the 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 no response is received, 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 the configuration in EEPROM (if required):

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

D.3 Intel® RSD Software Development Vehicle Platform

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

D.3.1 MMP Switch

This section provides the information configuring the MMP Switch.

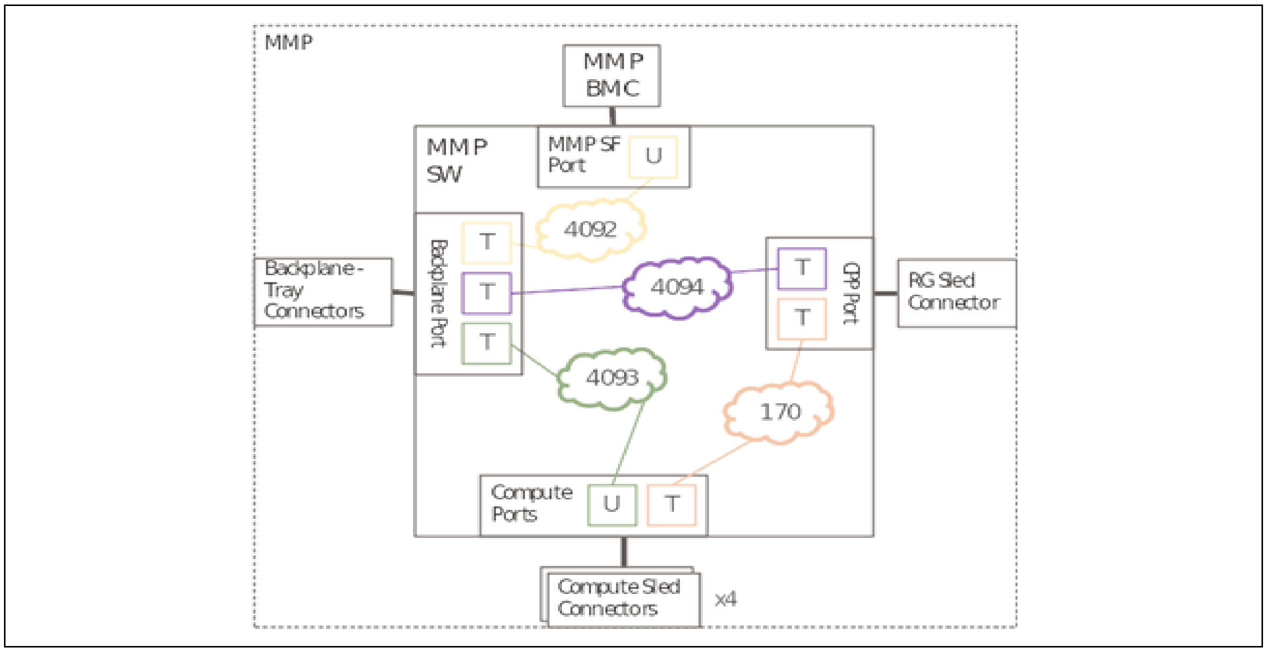
D.3.1.1 Prerequisites

- 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 is v1.10 or higher.

D.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 Figure 16.

Figure 16. Network Configuration



D.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, follow Steps 1 through 3 in Appendix D.2.2, Configuring CM.

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` through `4` accordingly. The firmware 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>`).
3. Assuming the base equals `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
```

4. Check the 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
```

5. Remove VLAN 1 (if it exists):

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

6. Set the 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
```

7. Check the 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
```

8. Store the configuration in EEPROM (if required):

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

D.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 Section 4.0 Intel® RSD Rack Network Configuration.

D.4 Remote iSCSI Target Blade Boot

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

D.4.1 Prerequisites

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

§

Appendix E PSME Software Installation from Packages

Appendix E provides the instructions for installing PSME software from packages.

E.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 listed in 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: The appropriate `psme-common` package must be installed before other packages.
- The PSME Network Configuration package, used for drawer configuration. For more information, refer to the Appendix E.4, PSME Network Configuration Package section.

E.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 (for example, proxy must be set if needed).

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

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

E.2.1 Initial Packages 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" section 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 the means of `systemctl` command).

E.2.2 Certificate Management

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

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

E.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, the new files are saved with `{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 need to be restarted after the configuration is updated.

E.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.

E.3 PSME Compute Ubuntu* v16.04 Packages

PSME Compute must be installed on the drawer, it communicates with Rack Management Module by the 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)`

E.3.1 Installation

1. Install `CyMUX` following the instructions in section Installing CyMUX from Miscellaneous.
2. Install 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
```

E.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 1.1.2.0 for communication with BMCs, MMPs and CMs) and 4094 (network 10.3.0.0 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)`

E.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
```

E.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)`

E.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 the 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
"service-root-name": "PSME Service Root" -> "service-root-name": "Root Service"
```
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 the RMM configuration. Devices should be configured properly and service should start automatically after reboot.

E.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)`

E.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
"service-root-name": "PSME Service Root" -> "service-root-name": "RSS Service Root"
```

   Optionally in `/etc/psme/psme-storage-configuration.json` change interface which is used as Portal IP (for connection to tgt 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 the services:

```
service psme-rest-server start
service psme-storage start
```

E.7 PSME PNC Ubuntu v16.04 Packages

PSME PNC must be installed on the management host that 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`)

E.7.1 Installation

1. Install the 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 the 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 the management host and switch board.

E.8 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)`

E.8.1 Installation

1. Install the 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
```

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

E.9 PSME NVMe 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 NVMe Discovery Server (`psme-nvme-discovery-server-{version}.deb)`

E.9.1 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
   ```

   On a host with Mellanox ConnectX-4/ConnectX-4 Lx interfaces install:

   ```
   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 section 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 and PSME Rest Server:

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

4. Edit the 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, the network-interface-name in /psme-discovery-server-configuration.json should be a different interface than the interface used by the PSME NVMe Target's Rest Service (network-interface-name in `/etc/psme/psme-rest-server-configuration.json`). Both services should be available on separate IP addresses.

5. 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"
           }
       ]
   }
   ```

E.10 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)`

E.10.1.1 Installation

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

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

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

E.10.1.2 Update

Before attempting to update PSME software on the switch, 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, 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.

E.10.1.3 Configuring and Starting PSME Services

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

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

2. The 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 starts after each EOS reboot, if installed as an extension.

3. 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. Table 20 specifies the structure of the "ports" section of the configuration.

Table 20. Ports Configuration

Key	Description	Possible Values	Arista* Switch Supported	Arista* Switch Default
id	Identifier of a switch port to be configured	Port identifiers as string	Yes	-
`portType`	Port type	`"Downstream", "Upstream", "MeshPort", "Unknown"`	`"Downstream", "Upstream"`	`"Downstream"`

E.10.1.4 Enabling Arista* eAPI JSON - Remote Procedure Call (RPC) management interface

Arista EOS offers multiple programmable interfaces for applications. One of them is 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 QoS parameters. It must be configured before starting PSME software.

1. Configure HTTP server:

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

2. Verify HTTP server status:

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

E.11 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
```

E.11.1.1 Signing a Package

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

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

Before signing an `.rpm package`, configure `.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 an .rpm package, use the following command:

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

Once the packages are signed, use the following guide to exchange the GPG key with the recipient:

https://www.gnupg.org/gph/en/manual/x56.html

E.11.1.2 Checking Signatures

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

```
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 following command:

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

To check the signature in an .rpm file, run:

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

§

Appendix F Miscellaneous

Appendix F contains additional information relevant to PSME and Intel® RSD Software v2.3.2.

F.1 Compilation Code Coverage and Sanitizer Build Versions

To build with code coverage (only GCC):

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

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

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

To build with thread sanitizer (only GCC, libtsan* has to be installed):

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

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

```
make code-coverage
```

To read code coverage results:

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

F.2 Installing CyMUX

To install `CyMUX`, go to *https://github.com/01org/intelRSD/tree/master/tools/CyMUX* to build and install `CyMUX` service from source. Alternatively, install `CyMUX` dependency from a package, as follows:

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

§