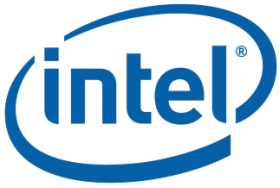


Intel® Rack Scale Design PSME

User Guide
Software v2.2

December 19, 2017

Revision 001

Document Number: 336810-001

Contents

December 19, 2017
Document Number: 336810-001

Appendices

Figures

Tables

Revision History

Revision	Description	Date
001	Initial release.	December 19, 2017

§

1 Overview

The interface specified in this document are based on the Distributed Management Task Force's Redfish™ Interface Specification and schema Table 4.

1.1 Document Scope

This document is a full and detailed documentation of the Pooled System Management Engine (PSME) Software. Information below covers minimal requirements for hardware and software during compilation and runtime. The document contains instructions for compilation, installation, deployment, and configuration of the PSME Software on various supported system environments.

The following topics will be covered in this documentation:

- PSME Software overview.
- Hardware requirements and software prerequisites.
- PSME Software Installation and deployment.
- Hardware and PSME Software Configuration.

1.2 Intended Audience

- Software Vendors (xSVs) of POD Management software, that make use of Intel® Rack Scale Design (Intel® RSD) PSME APIs to discover, compose, and manage the Intel® RSD architecture drawers, regardless of the hardware vendor and/or manage Intel® RSD drawers in a multivendor environment.
- Hardware Vendors (OxMs) of PSME firmware who would like to provide 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 the Drawer.

PSME Software consists of:

- PSME REST server - HTTP server with REST API and JSON data container responsible for gathering and presenting information about assets and available operations on these assets. This application communicates with agents through JSON-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 Assemble Procedure.
  - **PSME Network agent** - responsible for configuration and gathering detailed information about network topology. It also manages the data network ToR switch.
  - **PSME Chassis agent** - responsible for gathering detailed information about the CPP. Communicates with Rack Management Module (RMM).
  - **PSME Pooled NVMe Controller (PNC) agent** - responsible for gathering detailed information about the 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 tgt. This agent connects to the PSME Storage Service.

  - **RMM agent** - Rack Management Module, responsible for managing and gathering detailed information about rack and its power/thermal metrics.
- **PSME Compute agent simulator** - This is 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.
- **PSME Storage service** - It is the PSME REST server with storage-service flag enabled. Provides HTTP server with JSON REST API for storage pool support.

1.4 Supported System Environments

Table 1. Source Compilation

Component	Ubuntu* 16.04	Fedora* 23	ARM xcompiled (buildroot)	CentOS* 7
PSME REST Server	+	+	+	+
PSME Compute SDV	+	-	-	-
PSME Network	-	-	-	+
PSME Storage TGT-LVM	+	-	-	-
PSME Chassis SDV	+	-	-	-
PSME PNC SDV	+	-	-	-
PSME RMM SDV	+	-	+	-
PSME Compute Simulator	-	+	-	-

Table 2. Binaries Working

Component	Ubuntu* 16.04	Arista* EOS
PSME REST Server	+	+
PSME Compute SDV	+	-
PSME Network	-	+
PSME Storage TGT-LVM	+	-
PSME Chassis SDV	+	-
PSME PNC SDV	+	-
PSME RMM SDV	+	-

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

Supported hardware:

- Intel® SDV platform

Supported operating systems:

- Ubuntu* v16.04 LTS
- Arista* EOS
- Fedora* 23 for running Buildroot to prepare the Linux* Utility Image (LUI) for Deep Discovery feature.

The PSME Software compiles 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 will be described for generic hardware and environment with some references to specific cases for supported systems.

1.5 Definition of Terms

Table 3. Terminology

Term	Definition
Blade	Server Board that equates to the SPMF: ComputerSystem
BMC	Baseboard Management Controller
CA	Certificate Authority
CM	Control Module
CPP	Control Plane Processor
GAMI	Generic Asset Management Interface
Intel® SDV	Intel® Software Development Vehicle
Intel® RSD	Intel® Rack Scale Design
IPMI	Intelligent Platform Management Interface
IPMB	Intelligent Platform Management Bus
ISVs	Software Vendors
LAG	Link Aggregation Group
LUI*	Linux* Utility Image
LVM	Logical Volume Management
MDR	Managed Data Region
Module	Physical component housing a blade or switch
NVMe	Non-Volatile Memory Express
OxMs	Hardware Vendors
POD	A physical collection of multiple racks
PODM	POD Manager
PNC	Pooled NVMe Controller
PSME	Pooled System Management Engine
SMBIOS	System Management BIOS
REST	Representational state transfer
RMM	Rack Management Module
TGT	iSCSI target
ToR	Top of Rack network switch
TPM	Trusted Platform Module
UUID	Universally unique identifier
VLAN	Virtual LAN
XML	Extensible Markup Language

1.6 References

Table 4. Reference Documents

Doc ID	Title	Location
336811	Intel® Rack Scale Design Software Reference Kit Getting Started Guide, Software v2.2, Revision 1.0	*http://www.intel.com/intelRSD*
336814	Intel® Rack Scale Design Pod Manager Release Notes, Software v2.2, Revision 1.0	
336815	Intel® Rack Scale Design Pod Manager User Guide, Software v2.2, Revision 1.0	
336816	Intel® Rack Scale Design PSME Release Notes, Software v2.2, Revision 1.0	
336855	Intel® Rack Scale Design PSME REST API Specification, Software v2.2, Revision 1.0	

Doc ID	Title	Location
336856	Intel® Rack Scale Design Storage Services API Specification, Software v2.2, Revision 1.0	
336857	Intel® Rack Scale Design Pod Manager REST API Specification, Software v2.2, Revision 1.0	
336858	Intel® Rack Scale Design Rack Management Module (RMM) API Specification, Software v2.2, Revision 1.0	
336859	Intel® Rack Scale Design Generic Assets Management Interface API Specification, Software v2.2, Revision 1.0	
336860	Intel® Rack Scale Design Firmware Extension Specification, Software v2.2, Revision 1.0	
336861	Intel® Rack Scale Design Architecture Specification, Software v2.2, Revision 1.0	
336862	Intel® Rack Scale Design v2.2 Solid State Drive (SSD) Technical Advisory	
RFC2119	Key words for use in RFCs to Indicate Requirement Levels, March 1997	*https://www.ietf.org/rfc/rfc2119.txt*
SDP0266	Scalable Platforms Management API Specification v1.1.0	*https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.1.0.pdf*
DSP8010	Redfish Schema v2016.3	*https://www.dmtf.org/sites/default/files/standards/documents/DSP8010_2016.3.zip*

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 RFC 2119, Table 4.

1.8 Typographical conventions

Symbol and note convention are similar to typographical conventions used in CIMI specification.

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, "(" and ")", are used to indicate the scope of the operators "?", "*", "+" and "|".
- Ellipses (i.e., "...") indicate points of extensibility. Note that the lack of an ellipses does not mean no extensibility point exists, rather it is just not explicitly called out.

§

2 Key features

This section explains some of the key features of the Intel® RSD PSME software. In order to use some of the features, please follow the instruction in the Intel® RSD Drawer Configuration chapter.

2.1 Basic Discovery

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

The `psme-compute` needs to be provided with the correct addresses and credentials for the BMCs of the managed platforms in the configuration file stored in the `/etc/psme` directory.

Upon the start of the psme-compute service, the compute 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 being filled by the BIOS during boot time and takes a few minutes for this operation to complete.

In Intel® RSD software v2.2, the *Intel® RSD PSME* introduces support for platforms based on the Intel® Xeon® Processor Scalable Family (formerly code-named Purley) with BIOS and Baseboard Management Controller (BMC) implementing the *Intel® RSD v2.2 Firmware Extensions Specification*, refer to Table 4. For these platforms, 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
- FRU information of the system and its chassis
- Discrete FPGAs

2.2 Deep Discovery

Note: In Intel® RSD 2.2 reference code, all discovery performed by the PSME is done out-of-band (OOB). This default OOB discovery enables all RSD POD Manager functionality. The legacy RSD 1.x in-band discovery tool (LUI) is included in the reference code and may be used as an alternative to the OOB discovery.

The following sections describe how to build and configure in-band discovery tool.

2.2.1 Description

There are limitations to gathering hardware details from the BMC over the Intelligent Platform Management Interface (IPMI), especially if the platform is offline. For example, only data about non-system PCI/USB devices is 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 VLAN to the POD Manager.

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

The following steps take place during the Deep Discovery:

1. Basic discovery of the PSME provides resources (Blades) to be deep discovered.

2. The POD Manager selects resources (Blades) to run deep discovery on.
3. Those selected resources (Blades) are booted with a LUI.
4. The LUI exposes data to the POD Manager via a REST API.
5. The POD Manager merges the data obtained from both the PSME and LUI.

A full set of data is made available via the POD Manager REST API. For more information on implementing Deep Discovery, refer to the following steps and the Building Linux* Utility Image (LUI*) section in the Intel® Rack Scale Design 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 that is 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 PXE bootable bzImage: The Buildroot configuration file and kernel configuration file (Intel® RSD SDV platform) are available on the Intel® RSD source repository. The Buildroot tool can be downloaded from: *http://buildroot.uclibc.org/downloads/buildroot-2016.02.tar.gz.*

Buildroot depends on the following utilities:

Table 5. Buildroot dependencies

Utility	Remarks
which	
sed	
make	version 3.81 or any later
binutils	
build-essential	only for Debian* based systems
gcc	version 2.95 or any later
g++	version 2.95 or any later
bash	
patch	
gzip	
bzip2	
perl	version 5.8.7 or any later
tar	
cpio	
python	version 2.6 or any later
unzip	
rsync	
wget	
ncurses5	

It is recommended to perform both steps on the same machine running Fedora* 23 with at least 7 GB of free disk space. The installation requires access to public software repositories on the Internet. Confirm the server network, firewall, and proxy configurations allow the appropriate server access. Installation of the following dependencies on a standard Fedora 23 distribution is sufficient to compile all the necessary components.

```
dnf install bison flex libxslt-devel binutils-devel gcc cmake gcc-c++ cpp ccache
glibc-devel glibc-headers kernel-headers libmpc libstdc++-devel perl-Digest perl-
Digest-MD5 perl-GD libcurl-devel libmicrohttpd-devel libmicrohttpd argtable-devel
```

```
argtable libstdc++-devel libgcc libgomp libstdc++ emacs-filesystem lzo libarchive
texlive-epstopdf texlive-base texlive-epstopdf-bin texlive-kpathsea texlive-kpathsea-
bin texlive-kpathsea-lib ncurses-devel patch libgcrypt-devel gnutls-devel libxml++-
devel perl perl-Thread-Queue perl-Data-Dumper systemd-devel
```

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

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

It is advised to copy the source directory directly to a machine where the build will be made. If the source directory was not copied directly, then make sure that the sources were not modified in the process of copying. It could happen that 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="$RACKSCALE/lui/OS/rootfs"
   ```
3. Compile the psme-rest-server and psme-compute-simulator, then copy them to the `$ROOTFS/usr/bin` directory:
   ```
   cd $RACKSCALE
   mkdir -p build
   cd build/
   export CC=$(which gcc)
   export CXX=$(which g++)
   cmake ..
   make psme-rest-server psme-compute-simulator
   cp bin/psme-rest-server $ROOTFS/usr/bin
   cp bin/psme-compute-simulator $ROOTFS/usr/bin
   ```
4. Copy the `psme-rest-server` shared library dependencies to `$ROOTFS/usr/lib`:
   ```
   cp `ldd $ROOTFS/usr/bin/psme-rest-server | cut -d " " -f 3` $ROOTFS/usr/local/lib
   ```
5. Copy `psme-compute-simulator` shared library dependencies to `$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 were provided with Buildroot:
      ```
      rm $ROOTFS/usr/local/lib/libc.so.*
      rm $ROOTFS/usr/local/lib/libpthread.so.*
      rm $ROOTFS/usr/local/lib/libresolv.so.*
      ```
6. 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
   ```

7. 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
   ```
   c. 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
   ```
   d. 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
   ```
   e. 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"/' $ROOTFS/etc/psme/psme-rest-server-
   configuration.json
   ```
8. Set executable attribute to `$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.uclibc.org/downloads/buildroot-2016.02.tar.gz
   tar -xf buildroot-2016.02.tar.gz
   cd buildroot-2016.02
   ```
2. Update lshw recipe for Buildroot:
   ```
   rm -rf package/lshw
   cp -r $RACKSCALE/lui/package/lshw package/
   ```
3. Create Buildroot initial configuration:
   ```
   make menuconfig
   Choose <Exit>
   Do you wish to save your new configuration?
   Choose <Yes>
   ```
4. Copy (overwrite) LUI Buildroot configuration:
   ```
   cp $RACKSCALE/lui/config/buildroot.config .config
   ```
5. Create Kernel default configuration:
   ```
   make linux-menuconfig
   Choose <Exit>
   ```
6. Copy (overwrite) LUI Kernel configuration:
   ```
   cp $RACKSCALE/lui/config/kernel.config ./output/build/linux-4.4.16/.config
   ```
7. Build image passing `ROOTFS` overlay path, first build may take 1-2h depending on your Internet connection and computing power:
   ```
   make BR2_ROOTFS_OVERLAY=$ROOTFS
   ```
8. Generated image is ready to be copied to POD manager:
   ```
   scp output/images/bzImage rsa@PODM:/opt/pod-manager/wildfly/discovery/
   ```

2.3 Security features

2.3.1 Secure communication over TLS

Every PSME instance can be configured to only accept HTTPS connections secured with TLS protocol. The following points present the details of the feature:

1. The PSME Rest Server service generates a self-signed TLS cert (`server.key`, `server.cert`) on the initial startup and stores it in `/etc/psme/certs` directory.
2. The Certificate Authority (CA) public key, which is used to validate the POD Manager certificate, must be manually installed on the RMM (via USB stick or root scp).
3. The RMM installs the CA public key to all Drawers within the rack over either I2C or an IPMB call.
4. If no RMM is present, then every PSME must be provisioned manually by uploading the CA public key which is used for POD Manager TLS certificate certification. The `ca.crt` file (PEM format) must be placed in the `/etc/psme/certs` directory. The same must be done for PSME Storage (which does not communicate with the RMM). If no RMM present, then the psme-rest-server-`configuration.json` must be modified as follows:
   ```
   "rmm-present" : true -> "rmm-present" : false
   ```
5. For correct PODM certificate verification the system on which PSME runs need to have a correctly configured NTP client:
6. In `/etc/ntp.conf` in servers section:
   ```
   server 0.ubuntu.pool.ntp.org
   server 1.ubuntu.pool.ntp.org
   server 2.ubuntu.pool.ntp.org
   server 3.ubuntu.pool.ntp.org
   ```
   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.
   a. Start the NTP Daemon
   ```
   /etc/init.d/ntp start
   ```
   b. Set initially local date and time
   ```
   ntpdate -u IP_OF_YOUR_LOCAL_NTP_SERVER
   ```
   This command is used to manually synchronize your time with NTP server time. After initial sync NTP client will automatically perform periodic sync with NTP server.

2.3.1.1 Certificates configuration using RMM

To perform certificate deployment automatically, the PSME Chassis agent must be installed on each drawer in the rack.

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

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

When the RMM is present, the certificate (`/etc/rmm/podm.cert`) must be placed on the RMM.

The RMM will then automatically connect to the PSME Chassis and send the certificates which will be available for the PSME REST Server.

2.3.2 System Mode Management

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

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

The sled's mode can be modified by sending a PATCH request to the Computer System resource representing the sled.

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

2.3.3 Trusted Platform Module Management

RSD sleds include Trusted Platform Module 1.2 (TPM) 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. Please note that 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 REST API. In case of sled removal only single event is triggered, but all related resources will disappear on the API. Once the user removes hard drive from the JBOD the following events are triggered:

- Removing physical drive representing such drive (remove event)
- Setting physical volume health to critical (update event)
- Setting volume group health to critical (update event)
- Setting logical volume health to critical (update event for every logical volume placed on removed drive)
- Setting targets which points to logical volume health to critical (update event for every target)

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

2.4.1 Hot Swap Limitations

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

There is a hardware/BIOS limitation regarding discovery of CPU and DIMM information. Since enumeration of resources happens at boot time, no changes will be detected after a hardware change (e.g. reinsertion of a DIMM into a different slot) without reboot.

To work around this, the SLED must be rebooted and then removed and inserted back (or a PSME restart may be performed), so that full basic discovery succeeds.

After the SLED removal and reinsertion, a power on action should be performed. When an OS booting is completed, the SLED could be powered off. The action is necessary to update BMC data about assets coming from BIOS.

2.4.2 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 the Networking Related Remarks chapter.

2.6 Virtual LAN

The Virtual LAN (VLAN) functionality allows the ability to manage VLANs dynamically using the PSME REST API. Detailed VLAN information can be found Networking Related Remarks chapter.

2.7 MultiRack

This functionality allows for managing multiple racks with one POD Manager. The PSME Chassis Agent is responsible for providing a unique location property for each drawer in a MultiRack setup. The 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.

2.7.1 PSME requirements

The PSME requires exclusive access to the managed services. This means that the state of services which are under PSME management cannot be modified by other controllers (managers, command line, APIs, etc.).

2.8 Pooled NVMe Controller (PNC)

Important: It is necessary to apply BKC settings and management host setup before starting the `psme-pnc` and performing any tests.

2.8.1 Service configuration

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

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

A default psme-pnc configuration file (`psme-pnc-configuration.json`) can be found in the PSME source tarball. In order 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 can be generated with:

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

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 Rack Scale Design PODM in SSDP discovery mode (not 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

2.9.1 Description

This feature and its limitations are specific to Intel's SDV.

Due to BIOS implementation, there are two methods for configuring systems to boot from 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.

2.9.2 Limitations

- If data about an iSCSI Target is incomplete or points to a non-existent iSCSI Target (a sled cannot connect to the iSCSI Target - PSME will not detect it), then BIOS will attempt to boot from a local drive.

  Note: `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 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.

Note: BIOS does not have a separate boot option for `RemoteDrive` (`iSCSI OOB`).

When the `BootOverrideTarget` is set to `RemoteDrive`, 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 BootOverrideTarget 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 6. How PSME handles possible BootSourceOverrideEnabled Continuous options

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

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

- If BootSourceOverrideEnabled was previously 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 the BootSourceOverrideEnabled is set to **Once** with the BootOverrideTarget set to RemoteDrive, then after a BootSourceOverrideEnabled Once time-out, or a second power cycle, the system will boot from RemoteDrive.

Note: In the aforementioned cases, a new PATCH for the BootSourceOverrideEnabled Once/Continuous must be sent to alter current boot order.

Table 7. How PSME handles possible BootSourceOverrideEnabled one-time options

BootSourceOverrideEnabled Once with BootOverrideTarget:	Previous BootSourceOverrideEnabled 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

BootSourceOverrideEnabled Once with BootOverrideTarget:	Previous BootSourceOverrideEnabled 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
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`

2.9.3 NetworkDeviceFunction parameters

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

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

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

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

The `NetworkDeviceFunction` parameters whare 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.

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

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

2.9.4 Networking for iSCSI Booting in RSD SDV

The PSME Storage Service provides iSCSI Targets in 10.1.* or 10.2.* network. Change the providing network by configuring the "**portal-interface**" in `/etc/psme/psme-storage-configuration.json file` on the Storage Services (restart of PSME Storage Agent is required).

If you intend to iSCSI OOB boot from the default 10 G interface, then select an interface which is in 10.1.* network as "portal-interface".

If you intend to iSCSI OOB boot from the non-default 1 G interface, make sure that you have selected an interface which is in 10.2.* network as "portal-interface", and that you put a correct `"MACAddress"` of 1 G network card of a given system in `NetworkDeviceFunction:`

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

2.9.5 RSD SDV 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

2.9.6 Known issues in Intel® RSD SDV

PSME Compute Agent reads `BootSourceOverrideEnabled`, `BootOverrideTarget` and `BootOverrideMode` fields from BMC. When these fields are set via PSME, 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 `BootSourceOverrideEnabled` fields are patched, but the `BootOverrideMode` field is omitted, the `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

2.10.1 Overview

The telemetry service is a service which periodically polls the hardware for the telemetry data. The 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 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 of numeric metrics

2.10.2 Configuration

In the appropriate agent config file (e.g. `/etc/psme/psme-compute-configuration.json`), there are settings specific for metric definitions and common settings. These settings compose the telemetry section in the file.

2.10.2.1 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 Purley platform, the only handled metric definition 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 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 (ISO8601 format)
- `calculationAlgorithm` defines calculation algorithm to be used to calculate metrics of average/min/max (`averageOverInterval`, `minOverInterval`, `maxOverInterval`)
- `calculationTimeInterval` defines the time window for `calculationAlgorithm` (ISO8601 format)
- `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 ISO8601 format or a float value representing number of seconds)

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.

2.10.2.2 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 ISO8601 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).

2.10.2.3 Config file example

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

2.10.2.4 Sample computations

Average calculation is done according the equation:

Figure 1. Average equation

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

Assume the hypothetical sensor that returns integer values. Values are read with the `sensingInterval` of 1s ("PT1S"). `calculationTimeInterval` is 7s ("PT7S").

According these assumptions equation simplifies to:

Figure 2. Simplified average equation

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

Table 8. 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	-	**---**	**---**	**-**	**-**
12	4	12	**4**	**4.0**	**4**	**5**
13	5	12..13	**4.5**	**4.5**	**5**	5

Applying the precision reduces number of events, here are:

- 13 events sent for non-rounded values (no `calculationPrecision` property is set)
- 12 events for `calculationPrecision` = 0.5
- 8 events for `calculationPrecision` = 1
- 4 events for `calculationPrecision` = 5

Note: Bold values in Table 8 indicate when resource changed events are sent.

2.10.3 Limitations

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

2.11 CHAP Authentication

2.11.1 Description

This feature and its limitations are based upon Linux SCSI target framework (tgt).

Storage Services enable management of CHAP authentication for iSCSI Targets.

CHAP authentication could be disabled, set to One-Way or Mutual authentication. Type/Username/Password fields in POST/PATCH requests must be configured accordingly to the chosen authentication mode.

Table 9. POST request for creating iSCSI Targets with CHAP

authenticationMethod	chapUsername/ Secret	mutualChapUsername/ Secret	result	one-way account added	mutual account added
Null	null/""/-	null/""/-	OK 20x	-	-
OneWay	string	null/""/-	OK 20x	X	-
Mutual	string	string	OK 20x	X	X
No CHAP section (or CHAP: null)	-	-	OK 20x	-	-

Table 10. PATCH request for editing iSCSI Targets with CHAP

authentication Method	previous value of -->	chapUser name/ Secret	previous value of -->	mutualChap Username/ Secret	Result	one-way account added/ updated	mutual account added/ updated	one-way account deleted (if exists)	mutual account deleted (if exists)
Null	*	null	*	null	OK 20x	-	-	X	X
Null	*	null	null	-	OK 20x	-	-	X	-
Null	null	-	*	null	OK 20x	-	-	-	X
Null	null	-	null	-	OK 20x	-	-	-	-
Null	*	null	*	string	ERRO R 40x	-	-	-	-
Null	*	string	*	null	ERRO R 40x	-	-	-	-
Null	*	string	*	string	ERRO R 40x	-	-	-	-
OneWay	*	string	*	null	OK 20x	X	-	-	X
OneWay	*	string	null	null/-	OK 20x	X	-	-	-
OneWay	*	null/-	*	*/-	ERRO R 40x	-	-	-	-
OneWay	*	string	*	string	ERRO R 40x	-	-	-	-
Mutual	*	string	*	string	OK 20x	X	X	-	-
Mutual	*	string	string	-	OK 20x	X	-	-	-
Mutual	string	-	*	string	OK 20x	-	X	-	-
Mutual	string	-	string	-	OK 20x	-	-	-	-
Mutual	*	null	*	null	ERRO R 40x	-	-	-	-
Mutual	*	string	*	null	ERRO R 40x	-	-	-	-
Mutual	*	null	*	string	ERRO R 40x	-	-	-	-

Tables description:

- „string" - not empty string for username, but possible for password;

- „-" - no field in request;
- null - null value;
- „*" - string/null value;
- Account is updated if another account of the same type already exists.

2.11.2 Limitations

- User is able to patch passwords for CHAP authentication. However, the PSME REST API will always display null values due to security concerns.
- iSCSI Target can only have one One-Way and one Mutual CHAP account assigned.
- CHAP usernames cannot repeat within one Storage Service.
- CHAP usernames and passwords which are set for iSCSI Targets cannot contain spaces.
- Restart of tgt daemon on Storage Services deletes CHAP authentication from all iSCSI Targets, and requires a restart of PSME Storage Agent. CHAP authentication is not automatically restored.
- Tgt targets/CHAP account cannot be modified externally (i.e. via command line), these operations must be performed using PSME.
- Tgt should not have any CHAP accounts which are not assigned to iSCSI Targets when PSME Storage Agent is started.

§

3 PSME Development environment

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

3.1 Requirements

The PSME Software was developed in C++11 language targeting Linux based systems. The whole build process is managed by the CMake tool.

The following software versions are required in order to compile the PSME on Linux based system. Empty "Version" means that the default repository version will suffice:

Software	Version
CMake	≥ 3.4.3
gcc	5.3.1 ≤ ≤ 5.4.0
patch	

The following libraries must be installed prior to PSME compilation:

Software	Version
curl	
microhttpd	
LVM2	

If any of the libraries mentioned above 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 Fedora* 23

Enter the following command to install all Fedora packages:

```
dnf install doxygen gcc cmake gcc-c++ cpp clang glibc-devel glibc-headers kernel-
headers libmpc libstdc++-devel perl-Digest perl-Digest-MD5 perl-GD libcurl-devel
libmicrohttpd-devel libmicrohttpd argtable-devel argtable libstdc++-devel libgcc
libgomp libstdc++ emacs-filesystem lzo libarchive lvm2-libs lvm2-devel libxml++-devel
libnl3-devel libgcrypt-devel gnutls-devel patch mpfr-devel libmpc-devel systemd-devel
ncurses-devel libblkid-devel libblkid unzip gnupg rpm-sign
```

3.1.2 Ubuntu* 16.04 LTS

Enter the following command to install all Ubuntu packages:

```
apt-get 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 cmake cpp ccache \
build-essential linux-libc-dev libmpc-dev libstdc++6 libcurl4-openssl-dev \
libmicrohttpd-dev lcov libxml++2.6-dev libnl-3-dev libnl-route-3-200 libudev-dev \
libgcrypt20-dev libgnutls-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
```

3.1.3 CentOS* 7

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

The following command is 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
```

Note: 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
```

The PSME binaries targeted for Arista EOS need to be compiled with a cross compiler provided by Arista EOS. To obtain it, create a user account at the official Arista website *https://www.arista.com/en/*.

Once the user account has been created log in and 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 `"third_party"` folder:

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

If needed, make sure that you have an Internet connection with proxy servers set for http, https and ftp protocols. If this step was omitted, all required 3rd 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.

Building release version:

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

Building 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 other. For more information, refer to **man cmake**.

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
```

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 command in the build directory:

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

Running unit testing (all build types):

```
ctest
```

Generating documentation:

```
make doc-generate
```

Reading documentation:

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

3.3 Cross compilation process for ARM

Note: Buildroot environment must be built before the PSME cross compilation for the ARM.

For detailed instruction on how to compile and run PSME on MYB-IMX28X reference board go to Appendix 7.

Cross compilation commands to be run in the main PSME directory:

```
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

Use the following command to point to the correct version of CMake and compiler, before starting the compilation:

```
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 Intel® RSD Rack Network Configuration

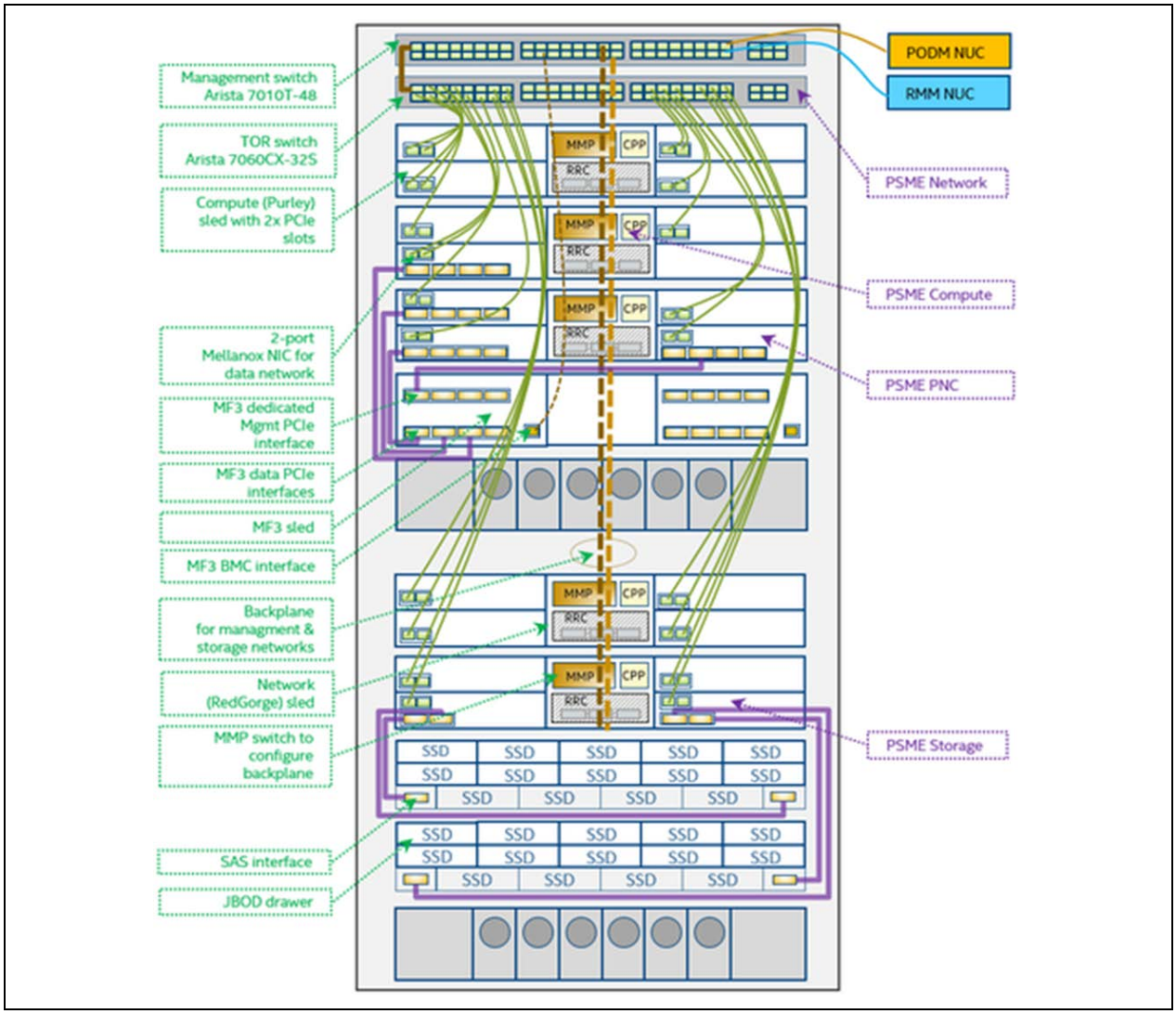
4.1 Overview

This section is specific to the reference design of the Intel® SDV platform networking. The Intel® SDV platform Top-of-Rack switch configuration is included in Appendix C, Top-of-Rack switch configuration.

4.1.1 Intel® SDV Physical layout

Refer to Figure 3 for the physical layout of the Intel® SDV Reference Platform.

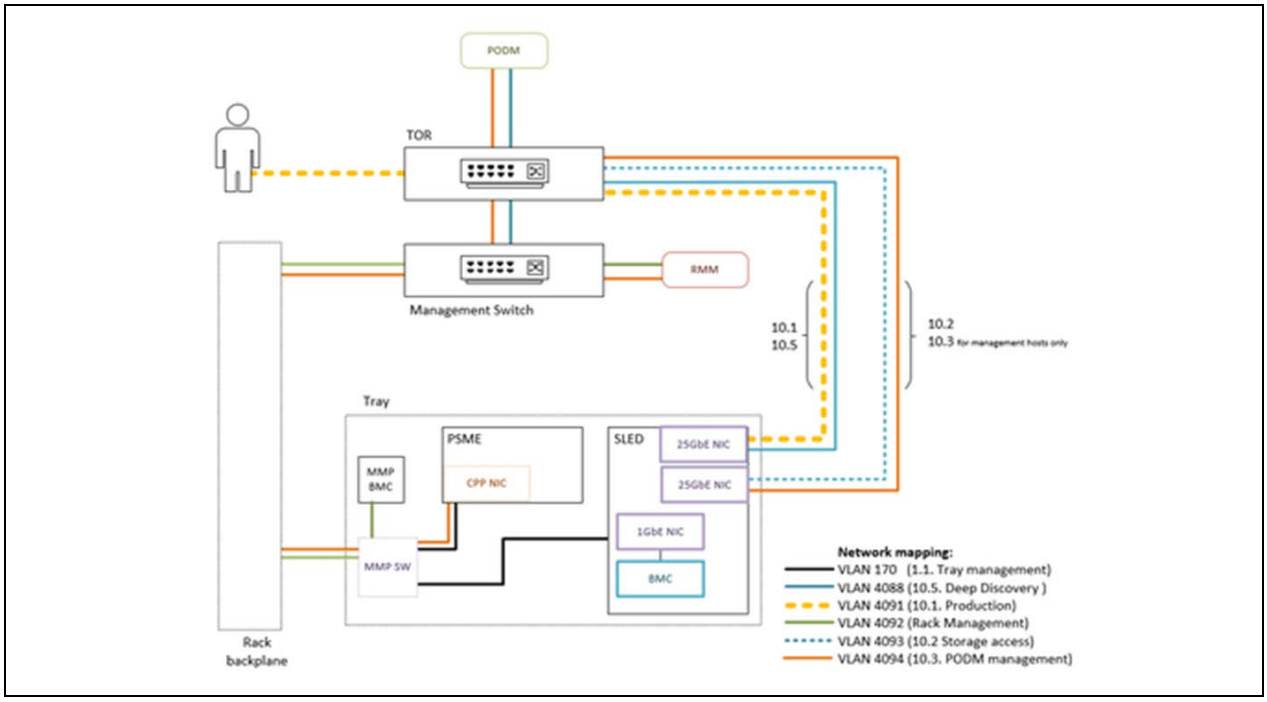
Figure 3. Intel® SDV Reference Platform



4.1.2 Intel® SDV Network Topology

Figure 4 shows the Network Topology of Intel® SDV Platform.

Figure 4. Network Topology of Intel® SDV Platform



4.1.3 Rack Switches

4.1.3.1 TOR Switch VLAN

Figure 5. Tor Switch VLAN Configuration

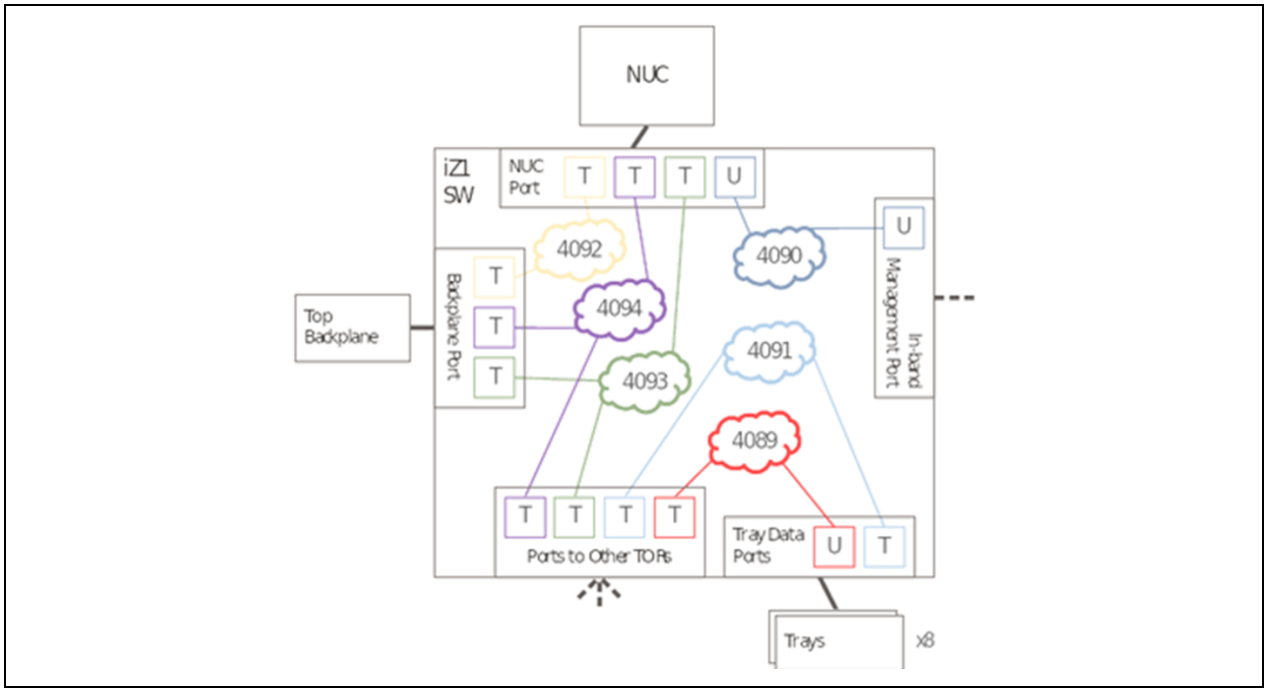
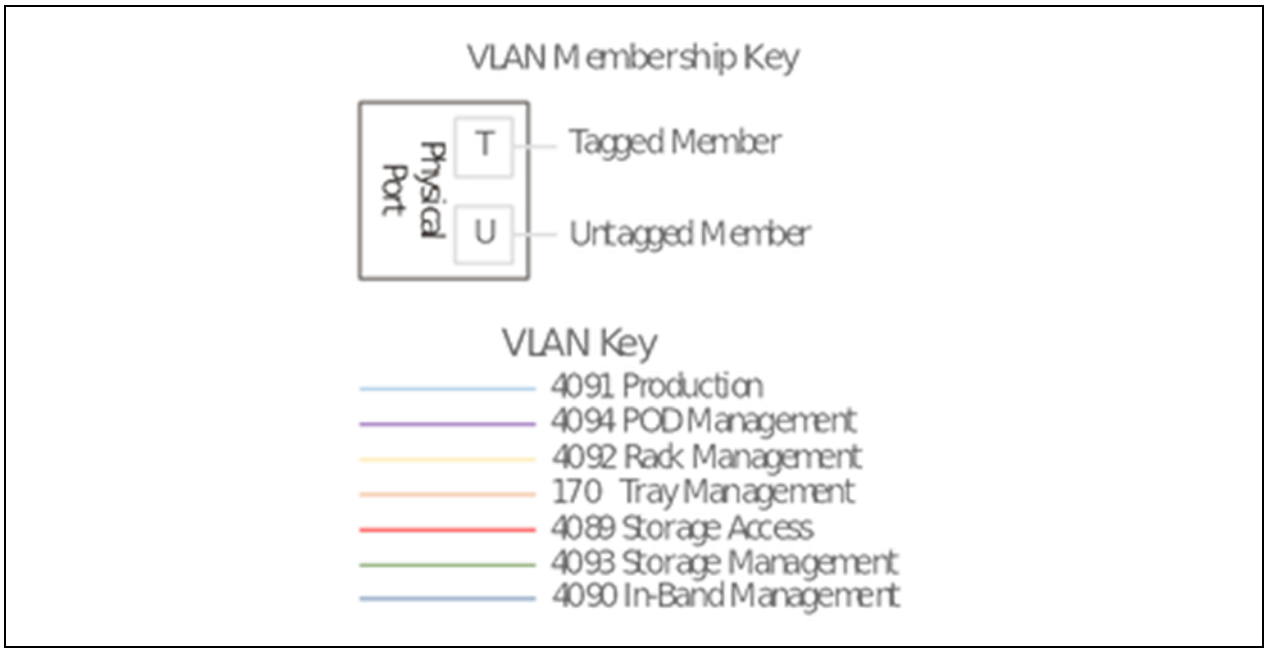


Figure 6. Tor Switch VLAN Configuration VLANs



4.1.3.2 MBP VLAN

Figure 7. MBP VLAN Configuration

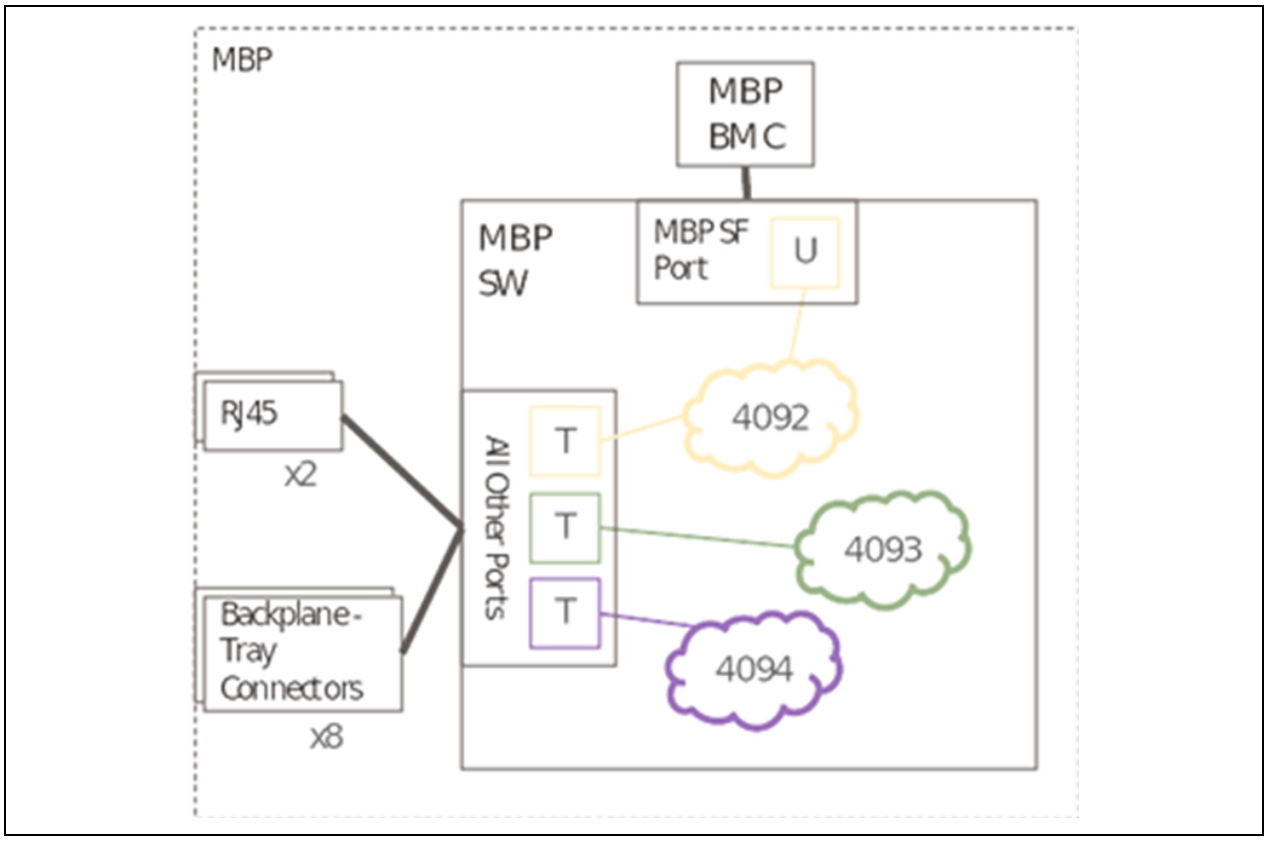
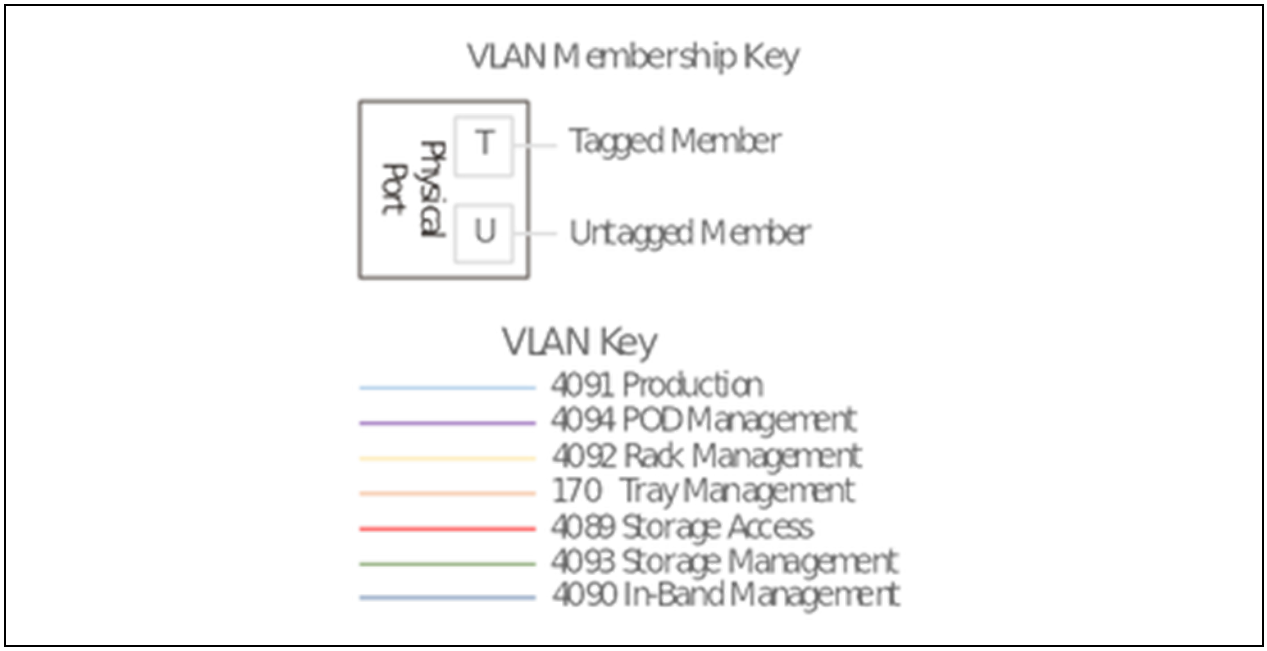


Figure 8. MBP VLAN Configuration VLANs



4.1.4 SDV Drawer Network

4.1.4.1 MMP VLAN

Figure 9. MMP VLAN Configuration

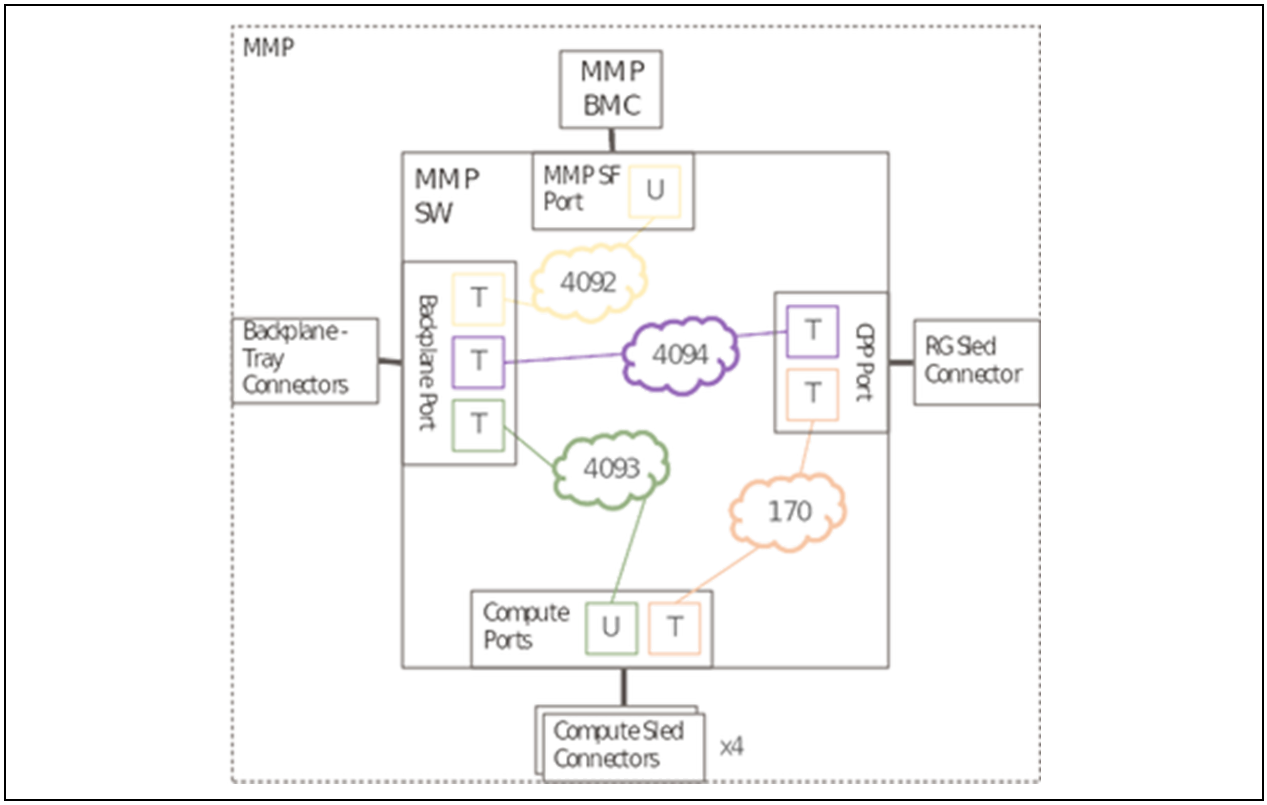
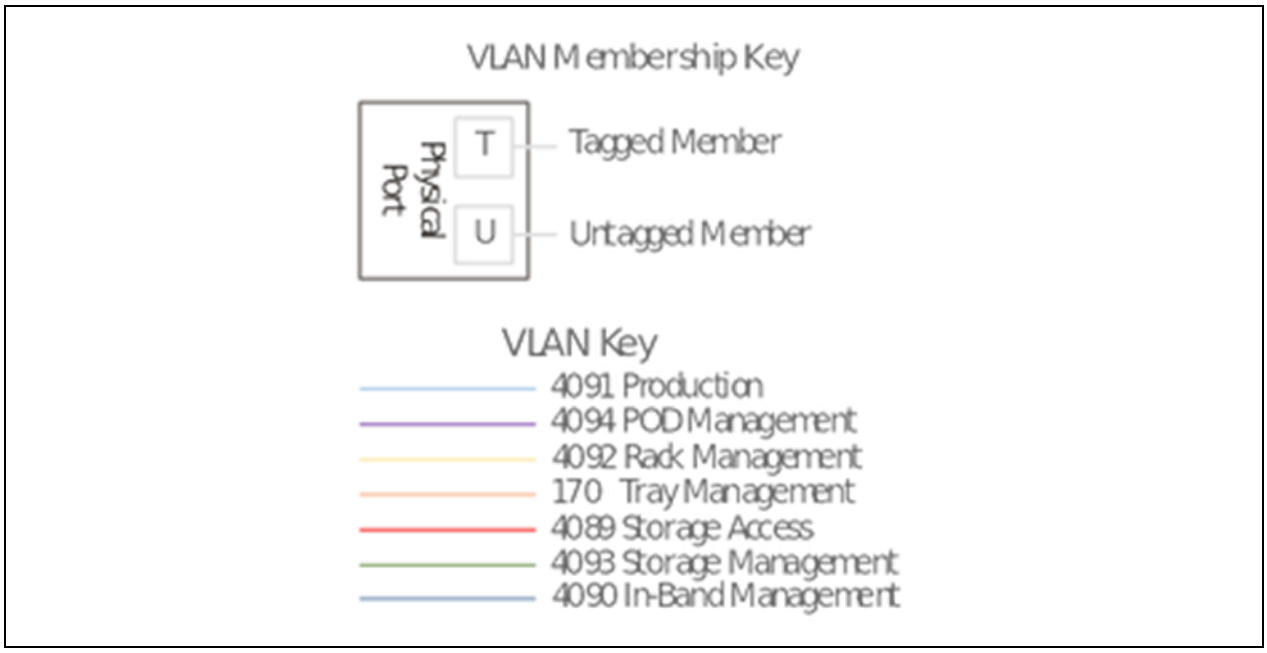


Figure 10. MMP VLAN Configuration VLANs



4.2 Networking Related Remarks

4.2.1 Virtual LAN Configuration

In order to configure VLANs on Arista switch ports through the PSME API, special requirements need to be satisfied.

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

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

   or

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

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

   ```
   show vlan
   show interfaces vlans
   ```

4. A VLAN cannot be disabled on a port. 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.
5. Name and description cannot be assigned to a VLAN. These parameters are not configurable on the switch. To work around this HW limitation, the REST server assigns the Name attribute automatically. When a VLAN is created through the GAMI API, the supplied Name is discarded. Trying to change that attribute from REST API or GAMI API is not supported.

4.2.2 Link Aggregation (LAG)

Note: 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 is supported only for the following:
  - Administrative and operational port state
  - Link speed.

4.2.4 Mesh Topology

The Mesh Topology feature is no longer supported since the introduction of the ToR switch for the data network.

4.2.5 Access Control List (ACL)

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 Intel® RSD Drawer Configuration

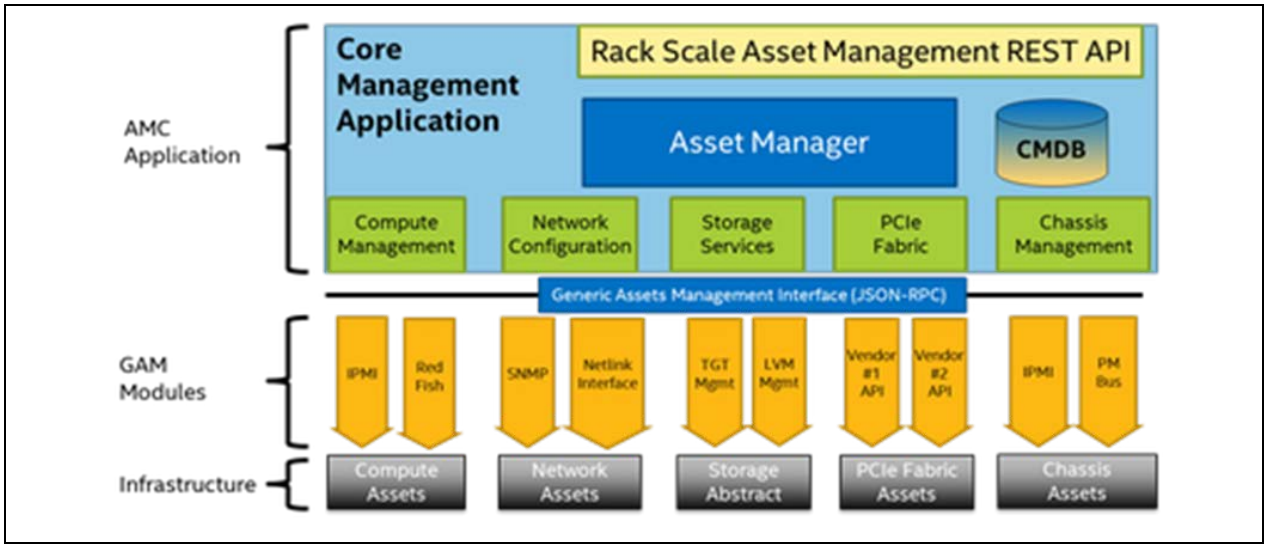
5.1 Hardware configuration

This section is specific to the reference design of the Intel® SDV platform hardware configuration. The Intel® SDV platform hardware configuration is described in the appendix section.

5.2 PSME Base Software

The PSME software consists of two software layers, PSME Rest Server and Generic Asset Management Modules. For the SDV platforms, the user must run the PSME Rest Server, PSME Compute, and PSME Network on each Drawer in Rack. Figure 11, PSME Software Components cover the main software components layout.

Figure 11. PSME Software Components



5.2.1 Running the PSME Components

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

The component can be executed with default values, or an optional configuration file can be passed as a command line argument. Use the following command pattern to run executable:

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

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 11.

Table 11. 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`

Name	Executable name
PSME Network Agent	psme-network
PSME Storage Agent	psme-storage
PSME PNC Agent	psme-pnc
PSME RMM Agent	psme-rmm
PSME Compute Simulator	psme-compute-simulator

5.2.2 PSME configuration file

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

Table 12 shows the location of the PSME module configuration files.

Table 12. PSME software configuration files

Name	Executable name
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 Compute Simulator	<PSME root>/agent-simulator/compute/configuration.json

5.3 PSME Storage Services

5.3.1 Prerequisites

Load PC with Linux OS, Ubuntu 16.04 or higher version.

5.3.2 Configuration

Configure the OS by connecting to iSCSI target and create an LVM structure.

5.3.2.1 Connecting to iSCSI Targets

Set the 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 how to create the LVM structure. Below each step is the code to execute the step

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

```
WARNING: BELOW INSTRUCTION WILL REMOVE ALL EXISTING DATA FROM THE DISK!!!
```

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

2. Create one `VolumeGroup` provided as the first parameter its name (i.e. `main_volume_group`) and as the second one `PhysicalVolumes` (i.e. `/dev/sdY`):
```
vgcreate main_volume_group /dev/sdY
```
3. Add other `PhysicalVolumes` to this `VolumeGroup:`
```
vgextend main_volume_group /dev/sdX
```
4. Create `LogicalVolume` by providing its size, name and `VolumeGroup`:
```
vcreate -L 10G -n base_logical_volume main_volume_group
```
5. Copy your image to `LogicalVolume`:
```
dd if=your_image.raw of=/dev/main_volume_group/base_logical_volume
```
6. If needed, this `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

Bootable attribute of `LogicalVolume` is set in LVM tags. It could be set using the PATCH request on the Logical Drive, or manually.

To manually make a given LV bootable, a "**bootable**" tag has to be added using the following procedure :

1. Add "**bootable**" tag to a given `LogicalVolume`:
```
lvchange --addtag @bootable /dev/main_volume_group/base_logical_volume
```
2. Restart `psme-rest-server` and psme-storage services to discover changes in LVM tags.

To remove the "**bootable**" tag:

3. Remove "bootable" tag from a given `LogicalVolume`:
```
lvchange --deltag @bootable /dev/main_volume_group/base_logical_volume
```
4. Restart `psme-rest-server` and psme-storage services to discover changes in LVM tags.

   LVM tags can be displayed using command:
```
lvs -o lv_name,lv_tags
```

While creating a new LV 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 LogicalVolume

Targets also could be created by using psme-rest-server REST API.

1. To create temporary target (existing until reboot or manual deletion) enter:
```
tgtadm --lld iscsi --mode target --op new --tid 1 --targetname
base_logical_volume_target
tgtadm --lld iscsi --mode logicalunit --op new --tid 1 --lun 1 --backing-store
/dev/main_volume_group/base_logical_volume
tgtadm --lld iscsi --op bind --mode target --tid 1 -I ALL
```
   Where:
```
base_logical_volume_target - is name for your target,
/dev/main_volume_group/base_logical_volume - is path to your LogicalVolume,
tid 1 and lun 1 - is target id and Logical Unit Number.
```
2. Restart `psme-rest-server` and `psme-storage` services to discover targets.
3. To make this target permanent:
```
tgt-admin --dump > /etc/tgt/conf.d/base_logical_volume.conf
```

5.3.3 Known Issues

1. In the "**Add iSCSI Target**" POST request "**Name**" and "**Type**" fields can be sent, but will not be saved.

   Note: These fields are not supported by the PSME Storage Service.

2. To create more than 34 iSCSI Targets (i.e. assemble total more than 34 Nodes with remote storage using the same Storage Services) in Ubuntu 16.04, the tasks limit for the target service must be modified:

```
sed -i "/\[Service\]/a TasksMax=infinity" /lib/systemd/system/tgt.service
systemctl daemon-reload
service tgt restart
```

3. In a POST a request on a "**Logical Drives Collection**" and "**Name**" field can be sent, but will not be saved.

   Note: This field is not supported by the PSME Storage Service.

5.4 PSME Chassis

5.4.1 Prerequisites

PC with Ubuntu Linux 16.04

5.4.2 Running the PSME Chassis Agent

Before starting the PSME Chassis, packages that are mentioned as default for Ubuntu development environment must be installed. Next, install CyMUX following the instructions in appendix: Installing CyMUX.

Build Chassis Agent, run and wait a moment while Chassis components are being discovered. You can watch the progress in `journalctl`.

5.5 PSME Pooled NVMe Controller

5.5.1 Prerequisites

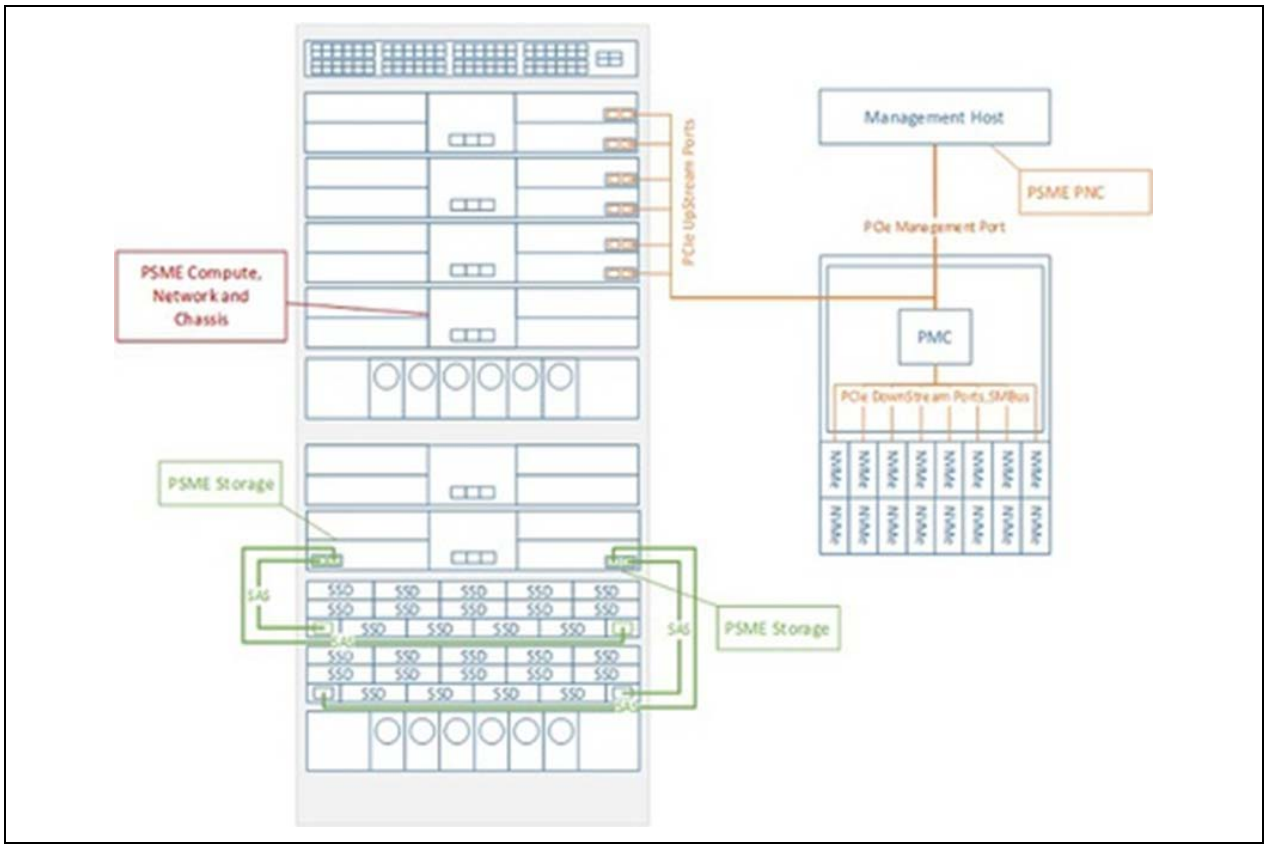
All of the following components must be configured with the latest BKC.

- Intel® SDV platform SLED with PCIe hot swap ready BIOS.
- PCIe switch board
- PCIe Add-in cards.
- Intel NVM Express drives.
- Ubuntu 16.04.

5.5.2 Hardware configuration

PSME Polled NVNe Controller (PNC) consists of two basic hardware components: PCIe switch board and management host (Intel® SDV SLED). The management host is connected to PCIe upstream port 24 of PCIe switch board. The remaining upstream PCIe ports are connected to compute SLEDs. SLEDs and management host must have PCIe retimer card PCIe add-in card connected. NVMe SSD drives are connected to PCIe downstream ports.

Figure 12. PSME Pooled NVMe Controller hardware configuration



5.5.3 Installation

This section provides information on the installation of Ubuntu v16.04 Server OS for the PNC management host, connection of the NVM SSD drives, and the importance of keeping the firmware up to date.

5.5.3.1 Ubuntu* v16.04

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

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

5.5.3.2 NVM Express SSD Drive

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

5.5.3.2.1 Firmware update

It is important to keep the NVMe SSD drive firmware up to date. Some drives can be flashed with old firmware, but will have limited support for the NVM Express Basic Management functionality.

1. Make sure that 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 the PCIe switch. The tool is available for download at the Intel Download Center: *https://downloadcenter.intel.com/*
3. The tool is not available in DEB format, but can be converted using **alien-pkg-convert** software. Refer to the RPM/AlienHowTo avaliable at: *https://help.ubuntu.com/community/RPM/AlienHowto*
4. When the DEB package is ready, install it:

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

5. List all connected drives:

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

6. Load new firmware:

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

5.6 Rack Management Module

This section provides the prerequisites, installation and configuration of the Rack Management Module.

5.6.1 Prerequisites

- PC with Linux OS, recommended Ubuntu 16.04.
- Hostname must be set to rmm-*.

5.6.2 Installation

To install RMM, follow the installation procedures included in the Appendix E, PSME Software installation from packages.

5.6.3 Configuration

Too assure a functioning RMM with the Intel® Rack Scale Design, the following steps must be followed:

1. 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 amd64 version:

```
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 Supported by Intel® SDV MMP BMC

Appendix A provides the IPMI Commands supported by Intel® SDV 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
  <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 PSME Configuration

B.1 MYB-IMX28X board details

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

B.2 Board features

- MYB-IMX28X Evalboard* from MYIR* producer
- ARM926EJ-S (ARMv5), Freescale* i.MX287 processor
- Frequency: 454MHz
- 128 MB DDR2 SDRAM
- 256MB NAND Flash
- 2xETH 10/100 Mb
- Support USB HOST, USB OTG, UART, I2C, SPI, CAN, ADC and others
- Dimensions: 62 mm x 38 mm

B.2.1 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. Boot modes are listed below:

Table 13. Boot modes

Boot mode	BM5	BM4	BM2	BM1	BM0	LCD_RS
USB0 (default)	X	0	0	0	0	1
I2C	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

Corresponding boot mode pins with pull-up (1) or pull-down (0) resistors:

Table 14. Boot mode pins

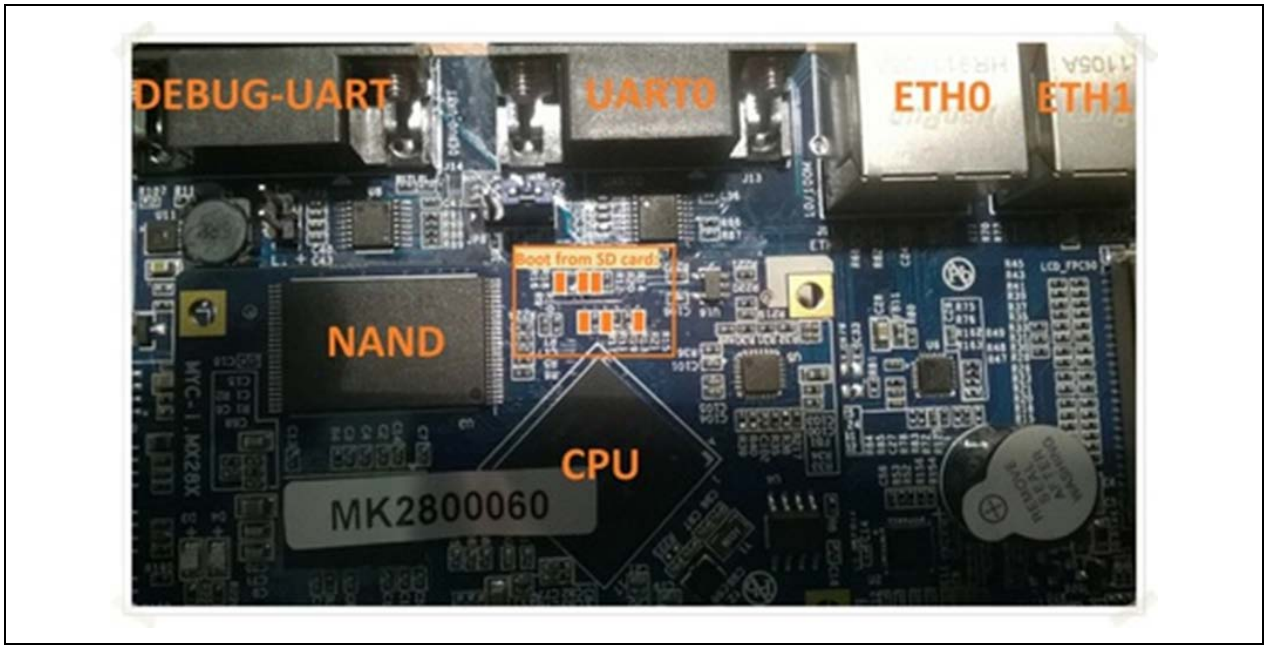
Boot mode	BM5	BM4	BM3	BM2	BM1	BM0	LCD_RS
47k pull-up	x	R11	R12	R13	R14	R15	R10
47k pull-down	R16	R17	R18	R19	R20	R21	x

B.2.2 Booting from SD Card

Booting from the SD card on new evaluation boards requires changing resistors on the BMx pins.

Note: 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 13. Reference Board main components



B.3 Build System Configuration

B.3.1 Environment preparation

1. Clone the buildroot project (these steps are valid for buildroot 2017.08, on other versions some options may be unavailable and other steps may be required):

```
git clone git://git.buildroot.net/buildroot
```

2. Change directory to the buildroot directory:

```
cd buildroot
```

B.3.2 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)
glibc version (2.23)
*** Binutils Options ***
Binutils Version (binutils 2.26.1)
*** GCC Options ***
GCC compiler Version (4.9.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 (Latest version 4.11.4)
    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:

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 (2017.05)
    U-Boot binary format (u-boot.sb)
```

3. Exit from `menuconfig`:

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

B.4 Image creation

B.4.1 Build images

```
$ make -j8
```

Generated directories layout under buildroot:

Table 15. 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)

Minimum required files after build are in output/images directory:

Table 16. Minimum required files

File	Description
`imx28-evk.dtb`	Device Tree Blob for i.MX28x Evaluation Board like MYB-IM28X
`rootfs.tar`	Root file system tarball 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.4.2 PSME cross compilation

1. Copy the buildroot directory to `psme/SW/tools.`
2. Go to the psme/SW directory.
3. Create the directory for build:

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

4. Execute cmake with proper configuration:

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

5. Compile project:

```
$ make -j8
```

B.4.3 Bootloader configuration

Bootloader is used to initialize memory, basic peripherals, and clocks, and load the necessary images and boot kernel. Add header to 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.5 Linux* image configuration

B.5.1 SD Card layout

1. Locate your SD card, typically is locate in `/dev/sdd, /dev/sdb or /dev/mmcblk`:

   ```
   fdisk -l
   ```

2. Create three SD card partitions (e.g. `/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 (e.g. 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 will automatically detect the partition and look for the Device Tree Blob image (`*.dtb`) to load and for the kernel image (`zImage`) to load and boot kernel (if partitions are already mounted, please 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 kernel image and Device Tree Blob to 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 networking
      1) Mount the SD card:
      ```
      mount /dev/sdd3 /mnt/rootfs
      ```
      2) 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
      ```
   g. Unmount the root file system:
   ```
   umount /mnt/rootfs
   ```

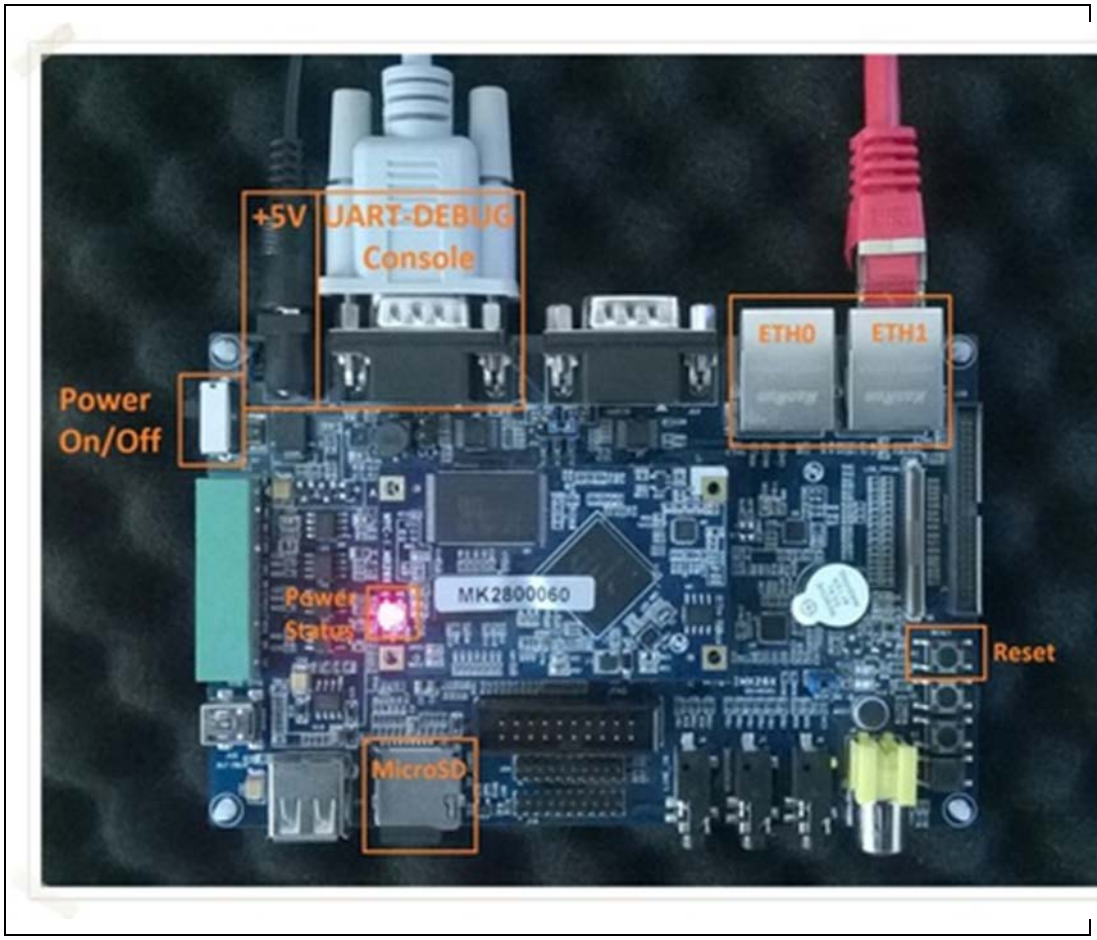
B.6 Evaluation board configuration

B.6.1 Booting

1. Insert the MicroSD card to the MicroSD slot. Note that not all cards are supported and in case of any errors, use different MicroSD card.
2. Connect the RS232 cable to the UART-DEBUG port to be used for 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 14. Reference Board connectors



B.6.2 Serial console settings

Settings for the serial console:

- 115200 baudrate
- 1 bit stop
- No parity
- No hardware flow control
- No software flow control

§

Appendix C Top-of-Rack switch configuration

C.1 Prerequisites

This section is specific to the reference design of the Intel® SDV platform ToR switch configuration.

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

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

C.2 Configuration process for management network ToR (via CLI).

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

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

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. Make sure to use the correct a serial port on the PC
   b. Connection speed is 9600
   c. Make sure to select "Serial"
   d. Open connection and wait for the ToR to respond
3. Log into the ToR system as 'admin' user.
4. Type "enable" to enter privileged mode.
5. The below table specifies the ToR VLAN configuration

Table 17. ToR VLANs configuration

Switch #show vlan	Tagged VLANs on Port	Untagged VLANs on Port	Native 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	48

6. Type "show vlan" to list VLANs.
7. Enter configuration mode:

```
configure
```

8. To create VLAN: type "**vlan 4090**" to create VLAN 4090, then type "**exit**".
9. Do the same steps to create vlan 4088, 4090, 4091, 4092, 4093 and 4094.
10. To verify vlan creation type `"show vlan"`.
11. Or to see ToR port 1 configuration, type `"show interface et1"`.

    To see all interfaces on ToR, type the following:

```
show interface
```

12. Add tagged VLANs 4088, 4090, 4091, 4092, 4093, and 4094 to port 1:
    a. Type:

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

    b. Type "**exit**" to return to config mode.
13. Use the same methods to configure tagged VLANs on all other ports according to the table above.
14. Add untagged and native VLAN 4094 on port 48:
    a. Type:

```
interface Ethernet 48
switchport trunk native vlan 4094
switchport trunk allowed vlan 4094
switchport mode trunk
```

    b. Type "**exit**" to return to config mode.
15. When finished, save the configuration by typing "**write**".

C.2.1 Configuration process for the data network ToR.

It is assumed that 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 will not apply to all four lanes of a port if not connected and the steps will need to be repeated if connected later for each port individually.

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

1. Login as 'admin' user.
2. Enter privileged mode using 'enable' command.
3. Enter configuration mode using '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 first sixteen 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
```

b. 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
```

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

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

6. Configure tagged VLANs on port 33:

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

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

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

§

Appendix D Drawer hardware configuration

Note: In case of I2C communication issues on the Intel® SDV platform make sure to have appropriate rework done.

D.1 Recommended hardware configuration

D.1.1 Rack setup configuration

- x1 NUC
- x1 TOR
- x1 Fabric module
- x4 Purley sleds

D.2 Control Module

Prerequisites:

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

Configuring CM:

Note: Following steps are valid for CM version 1.02 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.

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

   c. Detach screen session using **Ctrl- a – d** command.
   d. Kill the screen session.
3. Use the ipmi tool 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
```

```
If there is no response, 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
```

```
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 exists):

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

8. Store configuration in EEPROM (if required):

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

D.3 Intel® SDV platform

D.3.1 MMP Switch

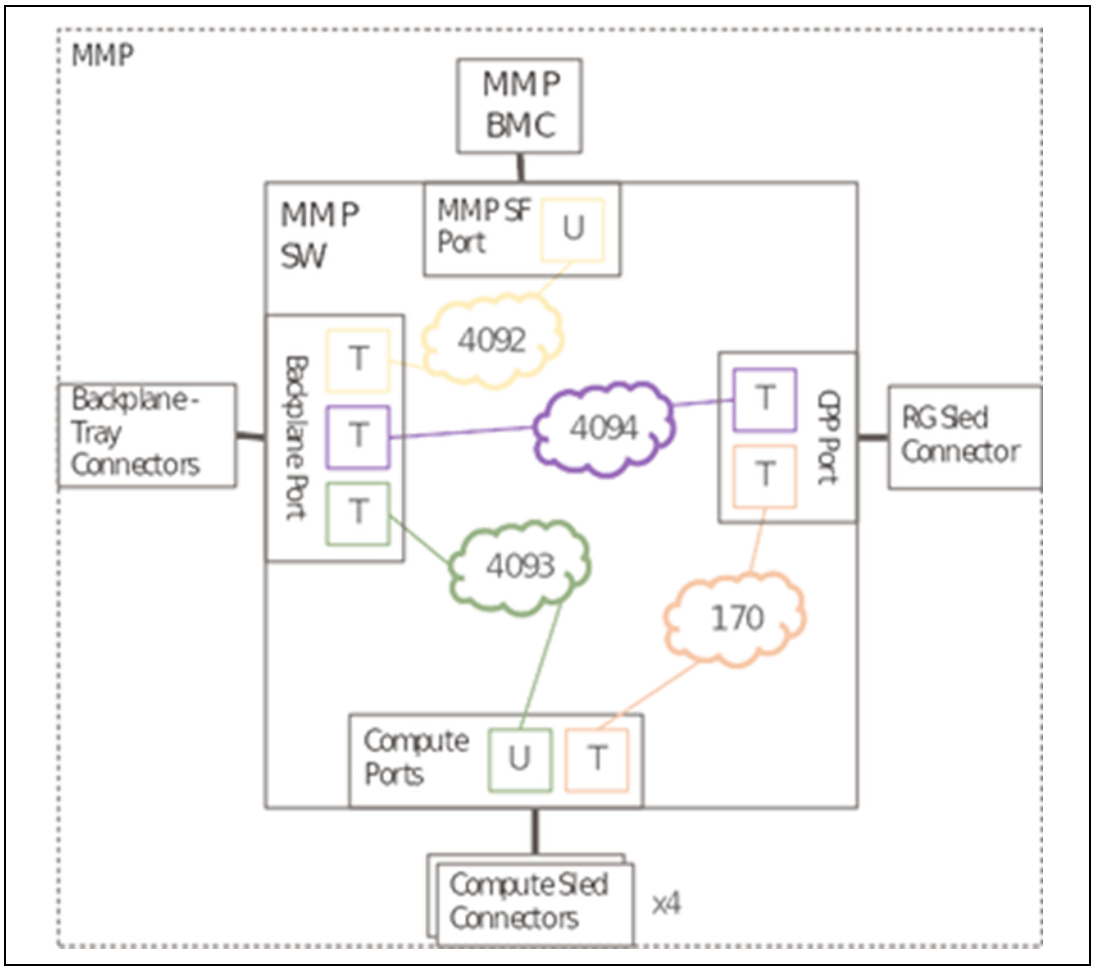
Prerequisites:

- Ensure PODM is installed and network interfaces are configured properly.
- Ensure the ToR switch is configured properly.
- Ensure screen is installed on PODM.
- Ensure the MMP FW version is 1.10 or higher.

Network configuration:

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

Figure 15. Network Configuration



Configuring MMP:

Assuming the CM IP address is known, the MMP switch can be configured by sending the IPMI commands to the CM using the rack management network (`1.1.1.x`) on the `PODM NUC`. If the CM IP address is not known, then follow "Configuring CM" Section D.2. Steps 1-3.

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

```
rsa@pod-manager:~$ ipmitool -I lanp -U admin -P admin -H 1.1.1.1 -b 0 -t
0x24 mc info
Device ID                 : 37
Device Revision           : 0
Firmware Revision         : 1.04
IPMI Version              : 2.0
Manufacturer ID           : 7244
Manufacturer Name         : Unknown (0x1C4C)
```

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

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

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

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

3. Check VLAN configuration:

```
   ipmitool -I lanp -U admin -P admin -H 1.1.1.1 -b 0 -t 0x24 raw 0x38 0x32
00 04 0f fc 50 40 0f fd 4f 40 0f fe 60 60 00 aa 2f 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 2f - vlan aa (170) - Tray (drawer) Management
```

4. Remove VLAN 1 (if exists):

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

5. Set PVIDs:

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

6. Check PVIDs:

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

7. Store configuration in EEPROM (if required):

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

D.3.2 Drawer OS Configuration.

For the PSME compute and chassis working properly, install the drawer using Ubuntu16.04 OS. Details of the network configuration are covered in Secton 4, Intel® RSD Rack Network Configuration.

D.4 Remote iSCSI Target Blade boot

D.4.1 Prerequisites

Make sure the following items are up to date:

- Make sure the BMC and BIOS are up-to-date.
- Make sure the Intel® SDV platform networks are 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

E.1 PSME Software packages – introduction

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

- A package for each binary listed in Table 2.
- A PSME Common package for each host, is required by other PSME packages. It is a set of common shared libraries and executables for the PSME provided by RPM/`.deb` packages.

  Note: The common package must be installed before other packages.

- The PSME Network Configuration package is used for drawer configuration.

Note: For more information, Refer Section E.4, PSME network configuration package.

Detailed information about using specific packages is provided in the following sections.

E.2 Update of configuration files from packages

During PSME package removal a current configuration file is renamed to `{component-name}-configuration.json.old` in the `/etc/psme/` directory.

During PSME package installation the `*.old` configuration file is compared with the configuration file from the package. If files are different, then the old configuration file has the `*.old` suffix removed and the new configuration file is renamed `{component-name}-configuration.json.new`. In this case a package installer should display a warning, that the User shall manually merge changes from `*.new` configuration file to the old one. If files are identical, then the old configuration file has `*.old` suffix removed and the new configuration file is not installed.

When the PSME package is updated, both aforementioned removal and installation steps are applied.

E.3 PSME Software Ubuntu 16.04 packages

E.3.1 Installation

1. The following PSME binary `*.deb` installation files can be built from sources or acquired from a pre-built binary file:
   - PSME Common
   - PSME Compute
   - PSME Chassis
   - PSME Rest Server
2. Copy all PSME `*.deb` files to Drawer.
3. Login into Drawer as a root.
4. Install CyMUX following the instructions provided on the project's GitHub® page: *https://github.com/01org/intelRSD/tree/master/tools/CyMUX*
5. Install dependencies for PSME Common, PSME Compute, PSME Chassis and PSME Rest Server:
   ```
   apt-get install libmicrohttpd10 libcurl3 libcurl3-gnutls openssl
   ```
6. Install the PSME using deb on the Drawer:

```
# 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
```

7. Change the hostname to begin with "psme" (it must be compatible with regular expression "^psme.*"), for example "psme-drawer-1":

```
# hostnamectl set-hostname --static "psme-drawer-1"
```

8. Reboot

```
# reboot
```

E.3.2 Update

Prerequisites:

System with installed PSME Common, PSME Compute, PSME Chassis and PSME Rest Server from deb packages.

Update PSME:

1. Copy all PSME `*.deb` files to Drawer.
2. Login into the Drawer as root.
3. Install CyMUX following the instructions provided on the project's GitHub page: *https://github.com/01org/intelRSD/tree/master/tools/CyMUX*
4. Update the PSME using the debs on the Drawer:

```
# 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
```

5. Modify the configuration files to match the setup, since the existing ones are not being overwritten during the update. By default, new configuration files are saved with suffix '.new'.
6. Restart updated services:

```
# service psme-rest-server restart
# service psme-compute restart
# service psme-chassis restart
```

E.4 PSME network configuration package

E.4.1 Overview

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 Figure 5, 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 POD manager).

It is intended to be installed only on the CPP (`drawer/tray`), where the `psme-compute`, `psme-chassis` and `psme-rest-server` services run.

E.4.2 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 16.04 packages

E.5.1 Installation

1. Check network connectivity (set proxy if needed).
2. Switch to the root account.
3. Download debs of PSME Common, PSME RMM and PSME Rest Server in the same version as the rest of packages.
4. Install dependencies for PSME Common, PSME RMM and PSME Rest Server:

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

5. Install PSME RMM and PSME Rest Server:

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

6. Install PSME Rest Server CA's certificate in `/etc/psme/certs/ca.crt`.
7. Install PODM certificate in `/etc/psme/certs/podm.cert`.
8. 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": "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"
```

   If PODM certificate is not located in `/etc/psme/certs/podm.cert`, its location has to be updated in RMM the configuration file `/etc/psme/psme-rmm-configuration.json`:

```
"podm": "/etc/psme/certs/podm.cert" -> "podm": "your_podm_certificate_location"
```

9. Reboot the system to finish RMM configuration. Services will start automatically after reboot.
10. After installation services may be started manually:

```
service psme-rest-server start
service psme-rmm start
```

E.5.2 Update

1. Copy the PSME Common, PSME Rest Server and PSME RMM debs to OS.
2. Switch to the root account.
3. Update the PSME packages:

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

4. 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"
```

   Update RMM configuration file `/etc/psme/psme-rmm-configuration.json` so that it includes any desired optional changes.

5. Restart the services (note that in this case, system reboot is not required):

```
service psme-rmm restart
service psme-rest-server restart
```

E.6 Storage Services Ubuntu 16.04 packages

E.6.1 Installation

1. Check network connectivity (set proxy if needed).
2. Download debs of PSME Common, PSME Storage and PSME Rest Server in the same version as the rest of packages.
3. Install dependencies for PSME Common, PSME Storage and PSME Rest Server:

```
apt-get install libmicrohttpd-dev libcurl4-openssl-dev tgt lvm2 liblvm2app2.2
```

4. Install PSME Storage and PSME Rest Server:

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

5. Edit configuration files.

   In the `/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"
```

6. User should place CA's certificate in `/etc/psme/certs/ca.crt` for PSME Rest Server.

7. Start services:
```
service psme-rest-server start
service psme-storage start
```
8. Wait for the Storage components to be discoverd discovered. Watch the progress in the console: `journalctl -fu psme-storage`.
9. To make Storage Services visible for Intel® Rack Scale Design PODM, change the OS hostname to begin with "**storage**" (it must be compatible with regular expression "`^storage.*`", for example "storage", "storageXYZ" or "storage-XYZ") and enable by getting IP from DHCP for the management interface.
10. If Storage Service does not appear on PODM, then execute this command on PODM machine:
```
service pod-manager restart
```

E.6.2 10.6.2 Update

1. Copy the PSME Common, PSME Rest Server and PSME Storage debs to OS.
2. Switch to the root account.
3. Update PSME packages:
```
dpkg -i psme-common-{version}.deb
dpkg -i psme-rest-server-{version}.deb
dpkg -i psme-storage-{version}.deb
```
4. Edit the configuration files. Located in the `/etc/psme/psme-rest-server-configuration.json` change as shown below:
```
"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 the `/etc/psme/psme-storage-configuration.json` change interface is used as Portal IP (for connection to TGT targets):
```
"portal-interface" : "eth0" -> "portal-interface" :
"interface_for_connection_to_targets"
```
5. Restart the services:
```
service psme-rest-server restart
service psme-storage restart
```
6. Wait while Storage components are being discovered. Watch the progress in a file:
```
/var/log/psme-storage.log
```

E.7 PSME PNC Ubuntu 16.04 packages

E.7.1 Installation

1. Check network connectivity (set proxy if needed).
2. Download suitable debs of PSME PNC and PSME Rest Server.
3. Install dependencies for PSME Storage and PSME Rest Server:
```
apt-get install libmicrohttpd-dev libcurl4-openssl-dev
```
4. Install PSME PNC and PSME Rest Server:
```
dpkg -i psme-common-{version}.deb
dpkg -i psme-pnc-{version}.deb
```

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

5. Edit configuration files.

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

```
"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"
```

6. Place the CA's certificate in `/etc/psme/certs/ca.crt` for PSME Rest Server.
7. Reset management host and switch board.

E.8 PSME packages for Arista EOS

E.8.1 Installation

1. One way to install the PSME RPM packages for Arista is to use BASH shell and follow standard Fedora installation methods:

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

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

2. Installation from CLI (it is assumed all packages are copied to /tmp).
   a. Enter configuration mode:

```
enable
configure
```

   b. For each PSME RPM package, copy and install as 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.8.2 Update

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

```
daemon psmenet
shutdown
```

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

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

2. If packages were installed using CLI, remove the old extensions first and then enter 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 as outlined in Section E.8.1, Installation.

```
Copy installed extensions to boot extensions.
```

E.8.3 Configuring and starting PSME services

1. PSME REST server needs to be started from `systemd` after each installation and reboot:
   ```
   systemctl start psme-rest-server
   ```
2. PSME network agent needs to be configured as EOS daemon from CLI configuration mode:
   ```
   daemon psmenet
   exec /opt/psme/bin/psmenet
   no shutdown
   exit
   write
   ```
   After the write command, the network agent will be started after each EOS reboot if installed as an extension.
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. 3.The following table specifies the structure of the "ports" section of the configuration:

Table 18. 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.9 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.9.1 Signing a package procedure

1. To sign a `.deb package`, use the command below:
   ```
   dpkg-sig -s builder <deb package>
   ```
2. 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
   ```
3. To sign an .rpm package use this command:
   ```
   rpm --addsign <RPM package>
   ```
4. Once the packages are signed, use the following guide to exchange your GPG key with the recipient:

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

E.9.2 Checking signatures procedure

1. Before checking a signature of a `.deb` package may need to import the GPG public key used during package signing using:
   ```
   gpg --import <gpg public key file>
   ```
2. To verify a signature in a `.deb` package, run following command:
   ```
   dpkg-sig -c <psme package>.deb
   ```
3. On a Fedora system, import the GPG public key file using following command:
   ```
   sudo rpm --import <GPG public key file>
   ```
4. To check the signature in an .rpm file run:
   ```
   rpm --checksig <PSME package>.rpm
   ```

§

Appendix F Miscellaneous

F.1 Compilation code coverage and sanitizer build versions

Building with code coverage (only GCC):

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

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

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

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

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

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

```
make code-coverage
```

Reading code coverage results:

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

F.2 Certificates configuration without RMM

Certificates usually are populated by the RMM and PSME Chassis. In the event that none of the components are installed, certificates can be manually deployed on the PSME REST Server:

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

And the PSME REST Server must have the `rmm-present` flag disabled:

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

To disable certificate validation, perform the following steps:

1. Change the ssl connector in the `psme-rest-server-configuration.json` file:
   ```
   "client-cert-required": false
   ```
2. The Drawer ID must be set manually in the PSME Chassis by setting:
   ```
   "chassis": "locationOffset": DRAWER_ID.
   ```

F.3 Installing CyMUX

1. Go to *https://github.com/01org/intelRSD/tree/master/tools/CyMUX* to build and install the CyMUX service from the source.
2. Alternatively, install cyMUX dependency from the package using:
   ```
   $ dpkg -i cymux-{version}.deb
   ```

§