Intel® Rack Scale Design PSME

User Guide Software Version 2.1.3

May 2017

Revision 003

Document Number: 335460-003

Contents

Revision History

Revision	Description	Date
003	Intermediate release.	May 19, 2017
002	ARM compilation clarified.	March 25, 2017
001	Initial release.	February 9, 2017

1 Overview

The interface specified in this document are based on the Distributed Management Task Force's Redfish™ Interface Specification and schema (see dmtf.org).

1.1 Document Scope

This document is a full and detailed documentation of 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 (ISVs) of POD Management software, that make use of Intel® Rack Scale Design (Intel® RSD) PSME APIs to discover, compose, and manage 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 for different hardware platforms other than Bulldog Creek SDV that 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 Drawer's switch like Red Rock Canyon.
  – **PSME Chassis agent** - responsible for gathering detailed information about 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 LVM and tgt. This agent connects to the PSME Storage Service.

- **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
PSME REST Server	+	+
PSME Compute SDV	-	+
PSME Network RRC	-	+
PSME Storage TGT-LVM	+	-
PSME Chassis SDV	-	+
PSME PNC SDV	+	-

Table 2 Binaries working

Component	Ubuntu 16.04	Fedora 23	ARM xcompiled (under Fedora 21)
PSME REST Server	+	+	+
PSME Compute SDV	-	+	-
PSME Network RRC	-	+	-
PSME Storage TGT-LVM	+	-	-
PSME Chassis SDV	-	+	-
PSME PNC 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:

- ONPSS (Fedora 23)
- Ubuntu 16.04 LTS

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

Throughout this document, all processes 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
CM	Control Module
IPMB	Intelligent Platform Management Bus
CPP	Control Plane Processor
LUI	Linux Utility Image
LVM	Logical Volume Management
Module	Physical component housing a blade or switch
POD	A physical collection of multiple racks
PODM	POD Manager
PNC	Pooled NVMe Controller
PSME	Pooled System Management Engine
REST	Representational state transfer
RMM	Rack Management Module
SDV	Software Development Vehicle
TGT	iSCSI target
ToR	Top of Rack network switch
UUID	Universally unique identifier
VLAN	Virtual LAN
XML	Extensible Markup Language

1.6 References

Table 4 Reference documents

Doc ID	Title	Location
335451	Intel® Rack Scale Design Generic Assets Management Interface API Specification	*Intel.com/intelrsd_resources*
335452	Intel® Rack Scale Design BIOS & BMC Technical Guide	*Intel.com/intelrsd_resources*
335501	Intel® Rack Scale Design Architecture Specification	*Intel.com/intelrsd_resources*
335454	Intel® Rack Scale Design Software Reference Kit Getting Started Guide	*Intel.com/intelrsd_resources*
335455	Intel® Rack Scale Design Pod Manager API Specification	*Intel.com/intelrsd_resources*
335456	Intel® Rack Scale Design Pod Manager Release Notes	*Intel.com/intelrsd_resources*
335457	Intel® Rack Scale Design Pod Manager User Guide	*Intel.com/intelrsd_resources*
335458	Intel® Rack Scale Design PSME REST API Specification	*Intel.com/intelrsd_resources*
335459	Intel® Rack Scale Design PSME Release Notes	*Intel.com/intelrsd_resources*
335460	Intel® Rack Scale Design PSME User Guide	*Intel.com/intelrsd_resources*
335461	Intel® Rack Scale Design Storage Services API Specification	*Intel.com/intelrsd_resources*
335462	Intel® Rack Scale Design Rack Management Module (RMM) API Specification	*Intel.com/intelrsd_resources*
335463	Intel® Rack Scale Design RMM Release Notes	*Intel.com/intelrsd_resources*
335464	Intel® Rack Scale Design Software RMM User Guide	*Intel.com/intelrsd_resources*

Doc ID	Title	Location
DSP0266	Redfish Scalable Platform Management API Specification	*http://dmtf.org/standards/redfish*

1.7 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® Rack Scale Design Drawer Configuration chapter.

2.1 Deep Discovery

2.1.1 Description

The Deep Discovery feature has been introduced to address instances of insufficient blade data from the BMC. The Deep Discovery feature can be used to gather hardware details which are unavailable while the platform is offline such as data from PCI/USB installed devices. This feature exposes the host details on the PSME REST API through a dedicated VLAN to the POD Manager. Note that 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 Linux Utility Image (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 the 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® RSD POD Manager User Guide.

2.1.2 Configuration

The Linux Utility Image 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 (adjusted for the BDC-A and BDC-R 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

Utility	Remarks
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 that 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 p
erl-GD libcurl-devel libmicrohttpd-devel libmicrohttpd jsoncpp-devel jsoncpp argtable-d
evel argtable libstdc++-devel libgcc libgomp libstdc++ emacs-filesystem lzo libarchive
texlive-epstopdf texlive-base texlive-epstopdf-bin texlive-kpathsea texlive-kpathsea-bi
n texlive-kpathsea-lib ncurses-devel uuid-c++-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 or psme-sdv-src-*.tar.gz source packages.

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.

2.1.2.1 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):

   1. Copy the shared library dependencies:

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

   2. 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-com
pute-simulator-configuration.json
cp $RACKSCALE/agent-simulator/compute/deep_discovery.xsd $ROOTFS/etc/psme/deep_dis
covery.xsd
cp $RACKSCALE/application/configuration.json $ROOTFS/etc/psme/psme-rest-server-con
figuration.json
```

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

   1. 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
   ```

   2. 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
   ```

   3. 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
   ```

   4. Set `announce-interval-seconds` variable to 60 seconds:

   ```
   sed -i 's/\"announce-interval-seconds\"[ \t]*:[ \t]*0/\"announce-interval-se
   conds\" : 60/' $ROOTFS/etc/psme/psme-rest-server-configuration.json
   ```

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

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

8. Set executable attribute to $ROOTFS files :

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

2.1.2.2 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/
```

2.2 Security

2.2.1 Overview

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-configuation.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:

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

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

    Add your local ntp server:

    ```
    server IP_OF_YOUR_LOCAL_NTP_SERVER prefer
    ```

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

    2. Start the NTP Daemon

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

    3. 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.

2.2.2 Certificates configuration

2.2.2.1 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 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.

2.3.1 Hot Swap Limitations

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.

2.3.2 Eventing Limitations

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

2.4 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 increased total bandwidth of the link between switches, LAG ensures redundancy between them. More details can be found in the Networking Related Remarks chapter.

2.5 Virtual LAN

The VLAN functionality allows to manage VLANs dynamically using PSME REST API. Detailed VLAN information can be found Networking Related Remarks chapter.

2.6 MultiRack

This functionality allows for managing multiple racks with one POD Manager. The PSME Chassis Agent is responsible for providing Parent ID (Rack ID) and Location Offset (Drawer ID). These values can be set in the PSME Chassis configuration file or can be overwritten by the RMM via IPMB protocol.

2.6.1 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.).

2.7 Pooled NVMe Controller (PNC)

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

2.7.1 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) libossp-uuid16
libsysfs2
```

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 actions later, 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).

In order to make the PSME PNC visible for Intel Rack Scale Design PODM in SSDP discovery mode (not DHCP), set the service root name in 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.

2.8 iSCSI Out of Band Booting

2.8.1 Description

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

iSCSI OOB Booting enables booting from iSCSI Targets without using PXE, and works only in UEFI mode.

This feature was developed with BMC in version 3.0.8 and BIOS in version F20A3A03.03_A.

In order to boot from iSCSI OOB from a given system, the NetworkDeviceFunction in the PSME REST API must be set with correct data about an iSCSI Target, and the BootOverrideTarget must be set to RemoteDrive.

2.8.2 Limitations

- If data about an iSCSI Target is incomplete or points to a non-existent iSCSI Target (a sled cannot connect to the iSCSI Target - PSME will not detect it), then BIOS will attempt to boot from a local drive.
- iSCSI OOB Parameters should not be modified externally (via BIOS/IPMI), only PSME Compute Agent should configure it.
- 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.

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

When the BootOverrideTarget is set to RemoteDrive, then 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 Continuous with BootOverrideTarget:	BIOS Boot Override	OOB iSCSI Parameters	Will boot from
Pxe	Pxe	irrelevant	Pxe
Hdd	Hdd	cleared	Hdd
RemoteDrive	Hdd	set from NetworkDeviceFunction	RemoteDrive

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

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

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

Table 7 How PSME handles possible BootSourceOverrideEnabled Once options

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

2.8.3 NetworkDeviceFunction parameters

Minimal set of NetworkDeviceFunction parameters which should be configured to boot from iSCSI OOB (for the default PODM configuration, in which 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", then following fields will not be updated in BMC's iSCSI OOB Parameters:

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

NetworkDeviceFunction parameters which are not supported:

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

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

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

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

2.8.4 Networking for iSCSI Booting in BDC-RRC

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

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

If you intend to iSCSI OOB boot from the non-default 1G interface, then 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 1G network card of a given system in NetworkDeviceFunction:

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

2.8.5 Known issues in BDC-RRC

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 BootOverrideMode field is omitted, then BootOverrideMode field is reconfigured with the value currently returned by BMC/PSME.

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

2.9 CHAP Authentication

2.9.1 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 8 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

authenticationMethod	chapUsername/Secret	mutualChapUsername/Secret	result	one-way account added	mutual account added
No CHAP section (or CHAP: null)	-	-	OK 20x	-	-

Table 9 PATCH request for editing iSCSI Targets with CHAP

authenticationMethod	previous value of -->	chapUsername/Secret	previous value of -->	mutualChapUsername/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	ERROR 40x	-	-	-	-
Null	*	string	*	null	ERROR 40x	-	-	-	-
Null	*	string	*	string	ERROR 40x	-	-	-	-
OneWay	*	string	*	null/-	OK 20x	X	-	-	X
OneWay	string	-	*	null/-	OK 20x	-	-	-	X
OneWay	*	null/-	*	*/-	ERROR 40x	-	-	-	-
OneWay	*	string	*	string	ERROR 40x	-	-	-	-
Mutual	*	string	*	string	OK 20x	X	X	-	-
Mutual	*	string	string	-	OK 20x	X	-	-	-
Mutual	string	-	*	string	OK 20x	-	X	-	-

authentication Method	previous value of -->	chapUsername/Secret	previous value of -->	mutualChapUsername/Secret	result	one-way account added/updated	mutual account added/updated	one-way account deleted (if exists)	mutual account deleted (if exists)
Mutual	string	-	string	-	OK 20x	-	-	-	-
Mutual	*	null	*	null	ERROR 40x	-	-	-	-
Mutual	*	string	*	null	ERROR 40x	-	-	-	-
Mutual	*	null	*	string	ERROR 40x	-	-	-	-
Tables notes: • „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.									

2.9.2 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:

Software	Version
CMake	≥ 2.8.12
clang	= 3.6.0
gcc	= 4.9.2
patch	

The following libraries must be installed prior to PSME compilation:

Software	Version
curl	≥ 7.41.0
microhttpd	≥ 0.9.37
jsonrpccpp	≥ 0.6.0
uuid	≥ 1.6.2
LVM2	≥ 2.02.111
Popt	≥ 1.16
IPMItool	= 1.8.15

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-header
s libmpc libstdc++-devel perl-Digest perl-Digest-MD5 perl-GD libcurl-devel libmicrohttp
d-devel libmicrohttpd jsoncpp argtable-devel argtable libstdc++-devel libgcc libgomp li
bstdc++ emacs-filesystem lzo libarchive uuid-c++-devel lvm2-libs lvm2-devel libxml++-de
vel libnl3-devel libgcrypt-devel gnutls-devel patch mpfr-devel libmpc-devel systemd-dev
el ncurses-devel libsysfs-devel libsysfs 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 li
budev-dev libglibmm-2.4-dev libglib3.0-cil-dev libxml++2.6-dev libgnutls-dev libnl-rout
e-3-dev flex bison valgrind doxygen cmake cpp ccache build-essential linux-libc-dev lib
mpc-dev libstdc++6 libcurl4-openssl-dev libmicrohttpd-dev libjsoncpp-dev lcov libossp-u
```

```
uid-dev libxml++2.6-dev libnl-3-dev libnl-route-3-200 libudev-dev libgcrypt20-dev libgn
utls-dev libsysfs-dev libpopt-dev libusb-dev patch libdevmapper-dev liblvm2-dev unzip l
ibnl-genl-3-dev libblkid-dev debsigs debsig-verify gnupg
```

3.2 Compilation

Uncompress the PSME source code package and create "third_party" folder (if it does not exist) in base source directory.

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 command in the build directory:

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

For detailed instruction how to run PSME on MYB-IMX28X reference board go to appendix.

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
```

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 the appendix section.

4.1.1 Physical layout

4.1.1.1 Intel SDV reference platform

Figure 1 Red Rock Canyon

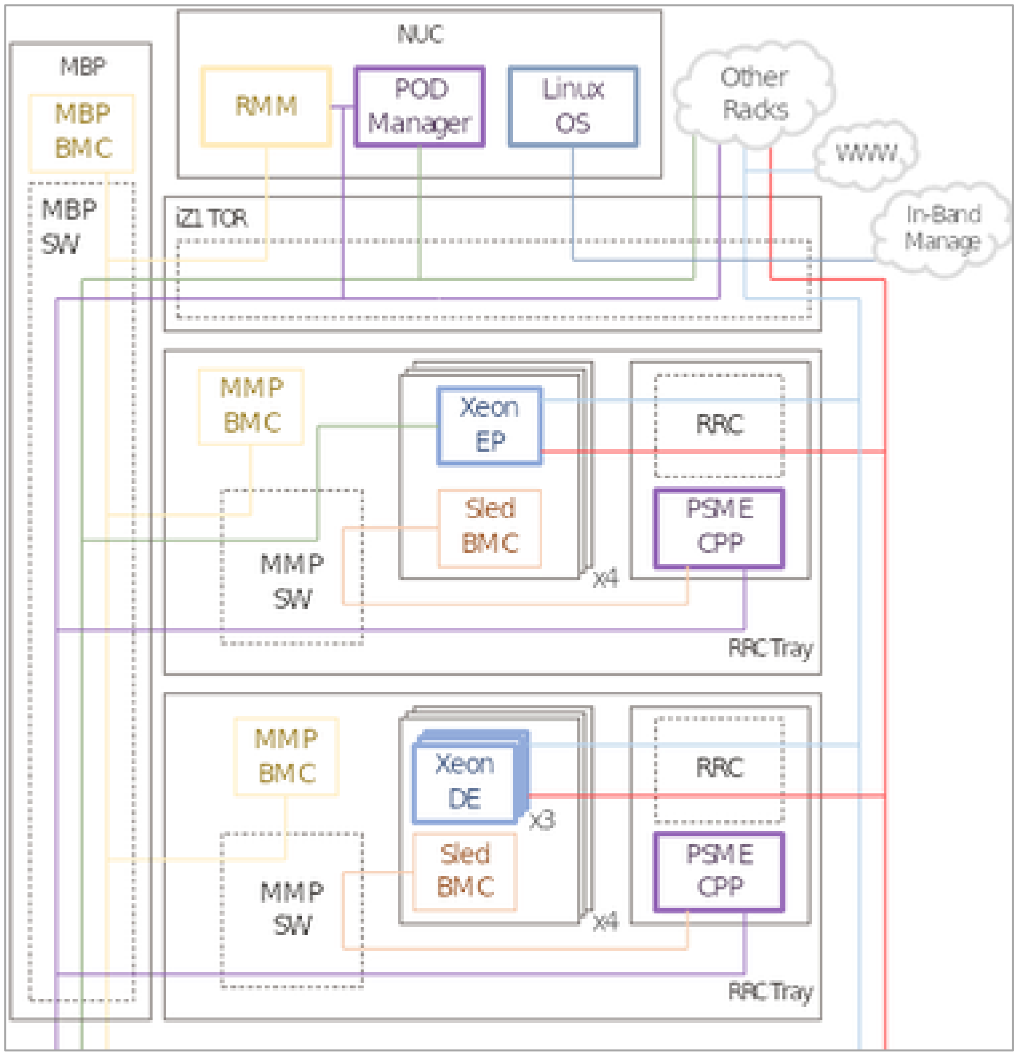
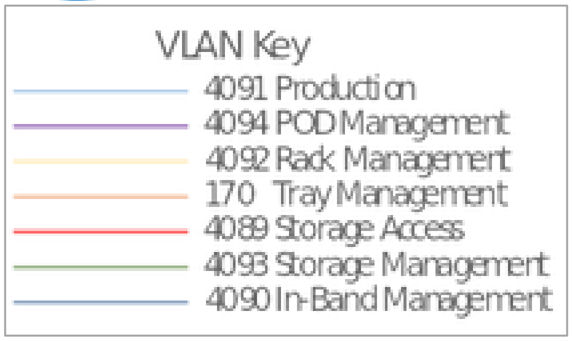


Figure 2 Red Rock Canyon VLANs



4.1.2 Rack Switches

4.1.2.1 TOR Switch VLAN

Figure 3 ToR switch VLAN configuration

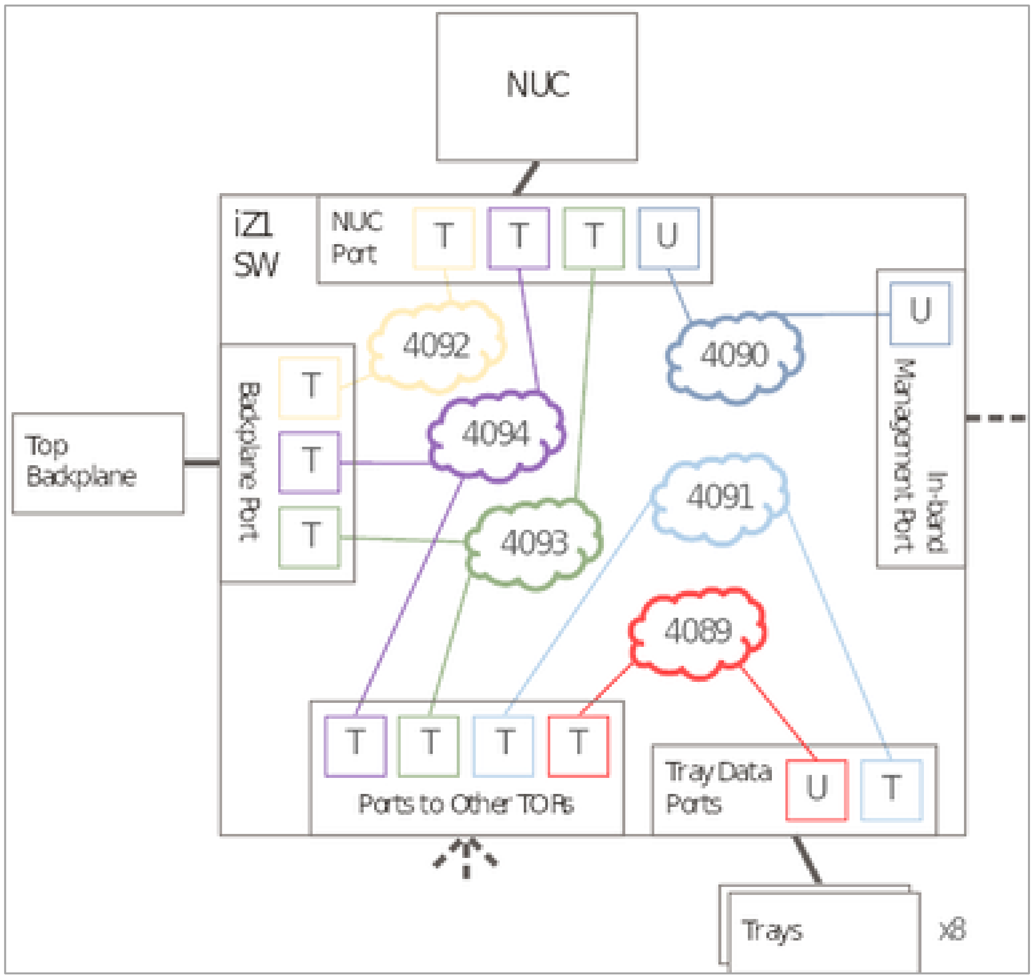
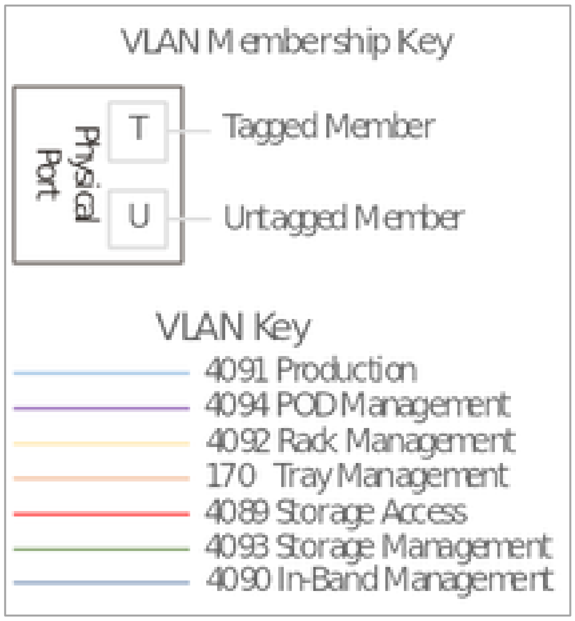


Figure 4 ToR switch VLAN configuration VLANs



4.1.2.2 MBP VLAN

Figure 5 MBP VLAN configuration

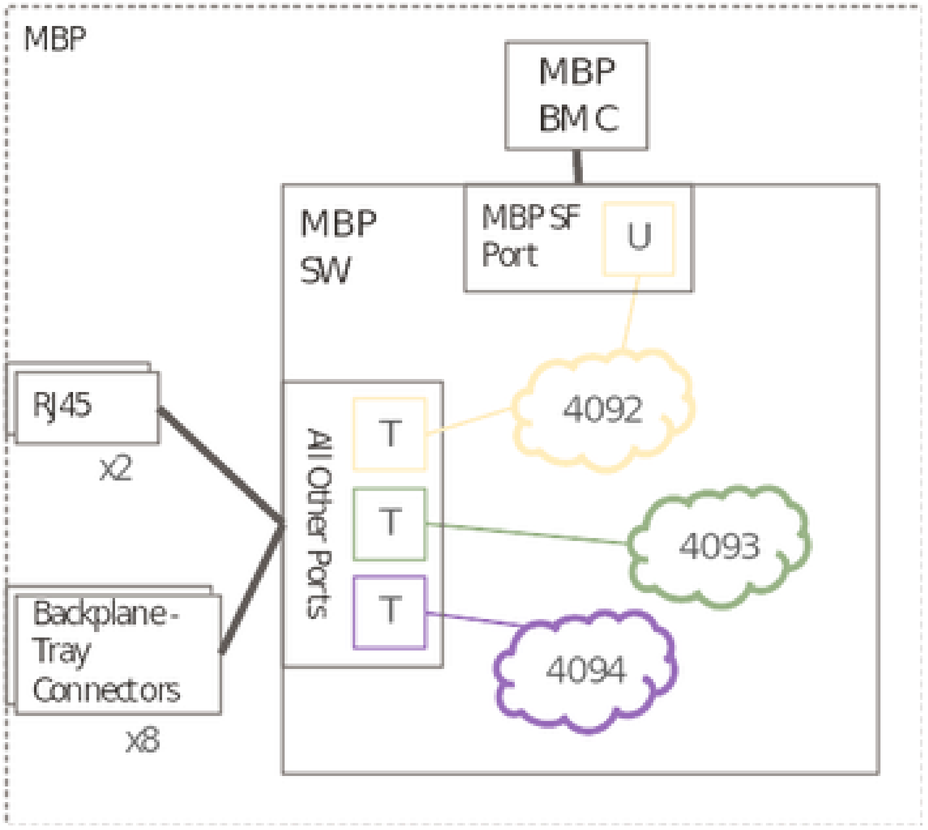
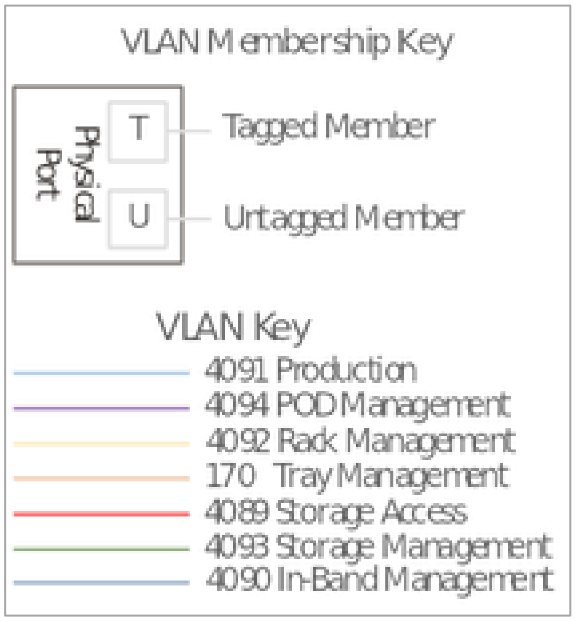


Figure 6 MBP VLAN configuration VLANs



4.1.3 Red Rock Canyon Drawer Network

4.1.3.1 MMP VLAN

Figure 7 MMP VLAN configuration

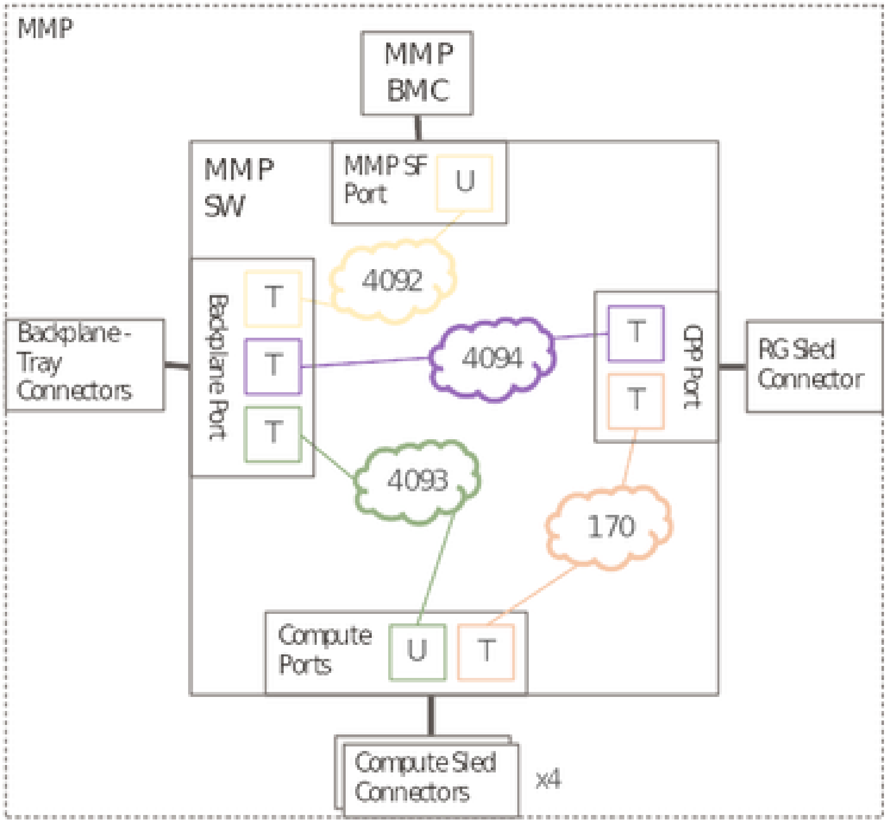
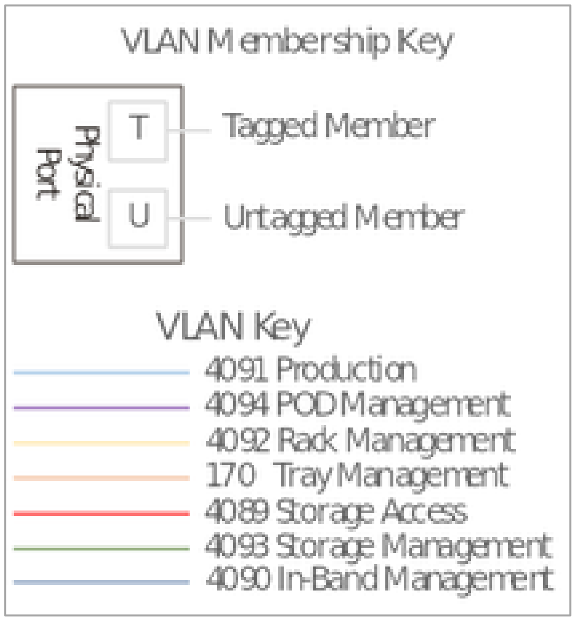


Figure 8 MMP VLAN configuration VLANs



4.1.3.2 Red Rock Canyon CPP NAT

Figure 9 RRC CPP NAT configuration

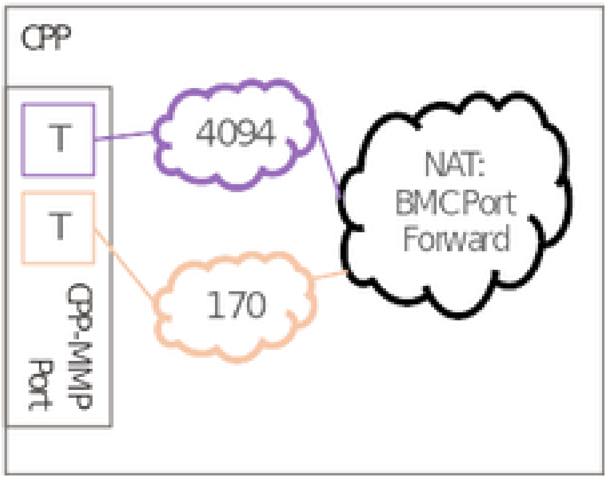
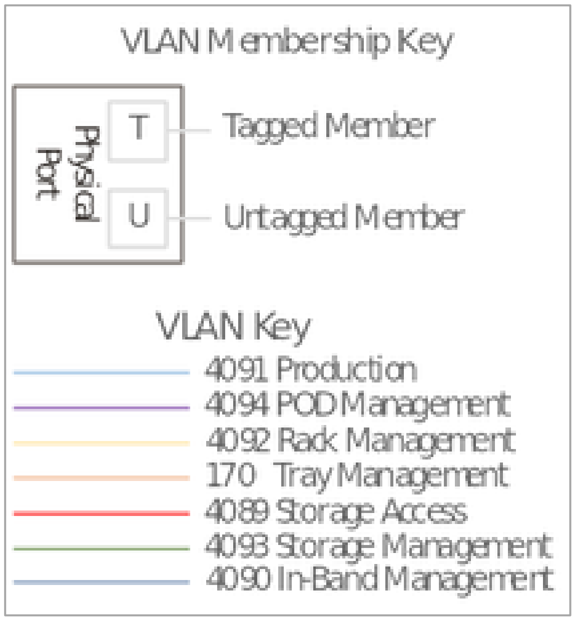


Figure 10 RRC CPP NAT configuration VLANs



4.1.3.3 Red Rock Canyon VLAN

Figure 11 RRC VLAN configuration

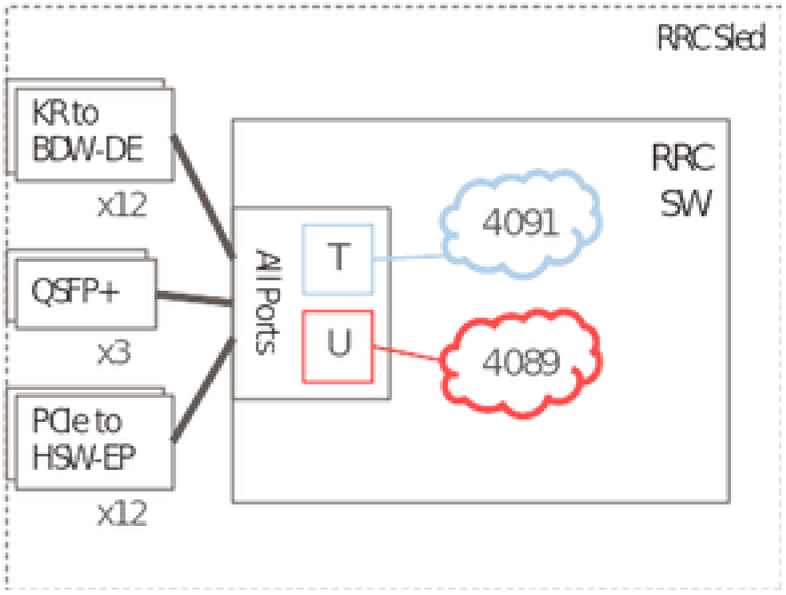
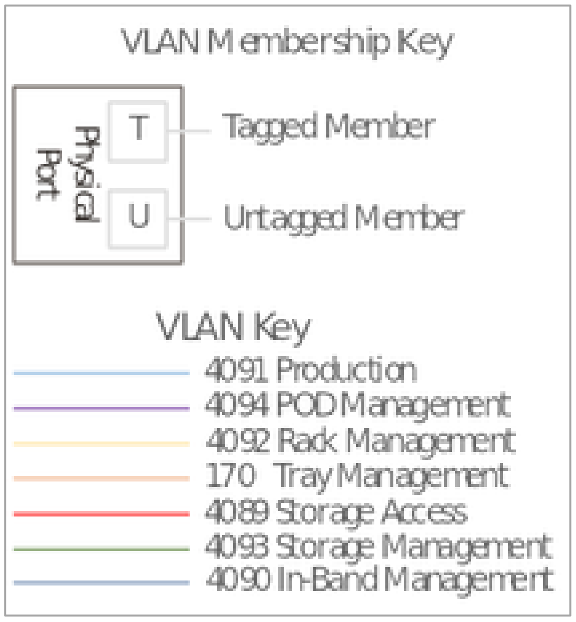


Figure 12 RRC VLAN configuration VLANs



4.2 Networking Related Remarks

4.2.1 Virtual LAN Configuration

In order to configure VLANs on RRC switch ports special requirements need to be satisfied.

1. The user shall not configure PCIe switch ports with a tagged VLAN. Even when using a VLAN subinterface on hosts connected to PCIe ports, untagged VLANs should be added on the switch.

   Note: this will not limit any functionality on that subinterfaces, packets will reach them as they were tagged.

2. It is not possible to manually add or remove VLANs to the CPU port (through the PSME Network API). It is sufficient to add a VLAN on PCIe ports as the network agent will configure CPU port with that VLAN automatically if needed. The VLAN is removed from CPU port when the user removes that VLAN from all PCIe ports.

   Note: configuring VLANs on CPU port is a DCRP requirement.

3. VLAN cannot be disabled on a port. VLANs can be added or deleted only. VLANEnable API attribute is always returned as "true". Trying to change that attribute from REST API is not supported on RRC.
4. Name and description cannot be assigned to a VLAN. These parameters are not configurable on the RRC switch. To work around this HW limitation, the REST server assigns VLANName attribute automatically. Trying to change that attribute from REST API is not supported.

4.2.2 Link Aggregation (LAG)

To configure LAGs on the RRC switch the following requirements and limitations are valid:

1. The LAG functionality is supported for the RRC switch only.
2. Only Static LAG functionality is supported. Note that Dynamic LAG cannot be configured, and changing LAG mode (Static/Dynamic) is not supported.
3. The LAG can have up to 16 port members.
4. The LAG must have at least one port member.
5. The switch port can be a member of one LAG at the same time.
6. The LAG port members must be Upstream type. The switch port type is configurable through the configuration file. DCRP mesh ports are not allowed on a LAG. Note that, for the Intel SDV platform, all ports going to Sleds (1-24, 37-44) are configured as Downstream, ports going outside of Drawer (25 and 29) are by default configured as MeshPort and port 33 configured as Upstream. See DCRP configuration section to configure mesh ports to allow for more Upstream ports, i.e. on the top most drawer.
7. Each LAG port member must have the same link speed.
8. Each LAG port member port must have the administrative state set to Down before adding it to the LAG. Note that, for the BDC-R platform, the PSME network always change port to Down state before adding it to the LAG, and after adding it to the LAG its administrative state is changed to Up.
9. Only one attribute of LAG port members can be changed: administrative state.
10. Only following attributes of LAG can be changed: administrative state and frame size.
11. All VLANs are removed from LAG port member before adding it to the LAG as a member.

12. The initial administrative state of each new created LAG is Down. To get LAG operational, its administrative state has to be set to Up. Note that other initial parameters of new created LAG are the same as default values for the switch ports.
13. The total speed of the LAG port is equal to sum of all LAG port members speed (which are in the Up operational state).
14. The LAG name can have up to 15 characters, and cannot contain whitespace characters.

4.2.3 Port parameters configuration limitations

1. The setting of link speed, frame size and autosense parameters on PCIe ports is not supported.
2. The speed parameter cannot be configured on slave ports if master port is configured as 40G/100G (QSFP) master.
3. The max frame size attribute must be a multiplication of 4. Trying to set this attribute to another value will result in setting the attribute to the closest value that is multiplied by 4.

4.2.4 DCRP Configuration

DCRP configuration file (`/etc/dcrpd.conf`) is automatically configured by dcrpini script located at `/etc/psme/`, before DCRP service is started. To change the default configuration for mesh ports and mesh VLANs user is required to edit following files:

```
/etc/psme/mesh_ports.sh
/etc/psme/mesh_vlans.sh
```

After these files are modified DCRP needs to be restarted, as well as the network agent. Switch reboot is also recommended.

4.2.5 Mesh Topology

The network agent receives the neighbor information from the LLDP agent (Open-LLDP daemon). The agents always communicate between each other (talk using LLDPDU packets) through the mesh links. Thus, the neighbor port information can be retrieved only on this port type. If communications between LLDP agents is lost from some reason (i.e. mesh port went down, wrong speed is set on the port etc.), then the neighbor information will not be provided through the network agent interface. Aging time for neighbor information is about 2 min 20 sec.

4.2.6 Access Control List (ACL)

To configure ACL on the RRC switch, the user should be aware of following:

1. Before binding/unbinding ACL to/from a port, at least one rule needs to be created. Therefore an empty list of ports needs to be passed when creating ACL on a switch.
2. When creating ACL rule, at least one condition needs to be specified in the request.
3. There is no HW support to change already created ACL rules hence the user is requested to delete existing rule and create a new one with new attributes.
4. ACLs are not supported on LAG ports.
5. The number of ACLs that can be created is limited by the switch resources, depending on the number of rules created and by the number and type of conditions that are defined in the rules.

4.2.7 Static MAC

To configure Static MAC on the RRC switch, the user should be aware of following:

1. There is no HW support to change an already created Static MAC entry hence the user is requested to delete the existing entry and create a new one with new attributes.
2. It is not possible to have the same Static MAC entry defined on two or more ports at the same time. Therefore, if adding a static MAC to a port that already exists on another port it will be removed from the existing port.

5 Intel® RSD Drawer configuration

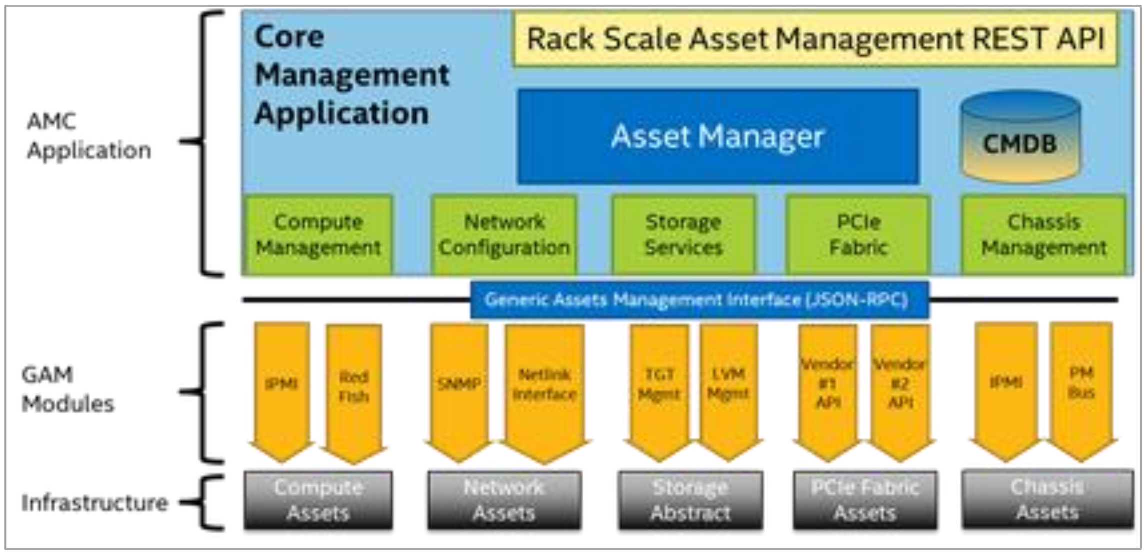
5.1 Hardware configuration

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 consist 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. The following image covers the main software components layout.

Figure 13 PSME software components



5.2.1 Running the PSME Components

Each PSME component can be executed by **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 below table.

Table 10 PSME executables in build output directory

Name	Executable name
PSME REST Server	psme-rest-server
PSME Chassis Agent	psme-chassis

Name	Executable name
PSME Compute Agent	psme-compute
PSME Network Agent	psme-network-fm10000
PSME Storage Agent	psme-storage
PSME PNC Agent	psme-pnc
PSME Compute Simulator	psme-compute-simulator

5.2.2 PSME configuration file

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

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

Table 11 PSME software configuration files

Module	Configuration file
PSME REST Server	<PSME root>/application/configuration.json
PSME Chassis Agent	<PSME root>/agent/chassis/configuration.json
PSME Compute Agent	<PSME root>/agent/compute/configuration.json
PSME Network Agent	<PSME root>/agent/network/configuration.json
PSME Storage Agent	<PSME root>/agent/storage/configuration.json
PSME PNC Agent	<PSME root>/agent/pnc/configuration.json
PSME Compute Simulator	<PSME root>/agent-simulator/compute/configuration.json

5.3 PSME Storage Services

5.3.1 Prerequisites

PC with Linux OS, recommended Ubuntu 16.04 or higher version

5.3.2 Configuration

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

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 providing as the fist parameter its name (i.e. main_volume_group) and as the second one of PhisicalValumes (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:

```
    lvcreate -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 new LVM structure.

5.3.2.3 Bootable attribute of LogicalVolume

Bootable attribute of LogicalVolume is set in LVM tags. In order to make a given LV bootable, a "bootable" tag has to be added:

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.

In order to remove "bootable" tag:

1. Remove "bootable" tag from a given LogicalVolume:

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

2. 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 is inherited from a Master LV (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 /de
    v/main_volume_group/base_logical_volume
    tgtadm --lld iscsi --op bind --mode target --tid 1 -I ALL
```

   Where:

   1. base_logical_volume_target - is name for your target,
   2. /dev/main_volume_group/base_logical_volume - is path to your LogicalVolume,
   3. 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. 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 tgt 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 request on a "Logical Drives Collection" a "Name" field can be sent, but will not be saved. This field is not supported by the PSME Storage Service.

5.4 PSME Chassis

5.4.1 Prerequisites

- PC with Linux OS, recommended Fedora 23

5.4.2 Running the PSME Chassis Agent

Before starting the PSME Chassis, packages that are mentioned as default for Fedora 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

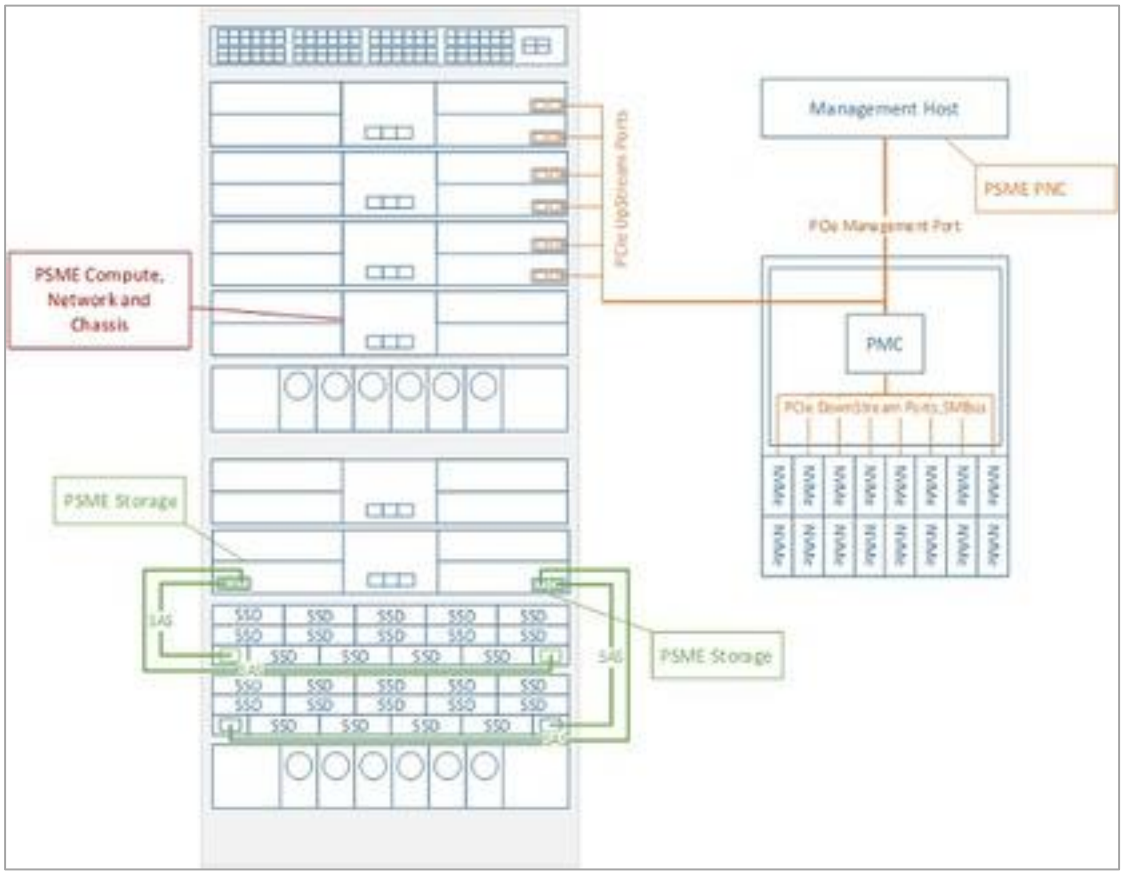
All of the following components must be configured with the latest BKC.

- 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 14 PSME pooled NVMe controller hardware configuration



5.5.3 Installation

5.5.3.1 Ubuntu 16.04

Ubuntu 16.04 Server is recommended operating system for PSME PNC management host. Please follow installation steps described in the install guide available on Ubuntu home page.

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

5.5.3.2 NVM Express SSD Drive

NVMe SSD drives have to be connected to 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 NVMe SSD drive firmware up to date. Some drives may be flashed with old firmware which has no full support for 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 (re
v 01)
04:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (re
v 01)
05:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (re
v 01)
06:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (re
v 01)
07:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (re
v 01)
08:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (re
v 01)
09:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (re
v 01)
0a:00.0 Non-Volatile memory controller: Intel Corporation PCIe Data Center SSD (re
v 01)
```

2. Download the Intel SSD Data Center Tool dedicated for a drive model connected to PCIe switch. The tool is available for download on *Intel Download Center*.
3. The tool is not available in DEB format, but it may be easily converted using Alien. Please see *Alien how to*.
4. When 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

5.6.1 Prerequisites

- PC with Linux OS, recommended Ubuntu 14.04.
- Hostname must be set to `rmm-*`.

5.6.2 Installation

To install RMM, follow the RMM User Guide.

5.6.3 Configuration

In order to assure a functioning RMM with Intel Rack Scale Design, the following steps must be satisfied:

1. GRUB configuration
    1. Edit the `/etc/default/grub` file and comment out the following variables:

    ```
    # GRUB_HIDDEN_TIMEOUT
    # GRUB_HIDDEN_TIMEOUT_QUIET
    ```

    2. 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
```

3. Apply changes by running the following command:

```
sudo update-grub
```

2. VLAN configuration:

   1. Install the VLAN package in amd64 version:

```
http://packages.ubuntu.com/trusty/vlan
```

   2. After installing the VLAN package add it to `/etc/modules`

```
8021q
```

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

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

auto eth0
iface eth0 inet manual

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

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

5.6.3.1 Incorrect system locale settings may make communication over JSON-RPC infeasible

Due to limitations of the jsoncpp library, using a locale identifier other than "`C`" is unsafe for `LC_NUMERIC` environment variable and may prevent the PSME components from communicating. In such a case an error log will appear containing the following line:

```
Exception -32700 : JSON_PARSE_ERROR: The JSON-Object is not JSON-VALID
```

To ensure proper settings follow these steps:

1. Check locale setting with `locale` command:

```
# locale
LANG="pl_PL"
LANGUAGE=
LC_CTYPE="pl_PL"
LC_NUMERIC="pl_PL"
LC_TIME="pl_PL"
LC_COLLATE="pl_PL"
LC_MONETARY="pl_PL"
LC_MESSAGES="pl_PL"
```

```
    LC_PAPER="pl_PL"
    LC_NAME="pl_PL"
    LC_ADDRESS="pl_PL"
    LC_TELEPHONE="pl_PL"
    LC_MEASUREMENT="pl_PL"
    LC_IDENT
```

2. Change the `LC_NUMERIC` variable to "`C`":

```
     # export LC_NUMERIC="C"
```

3. If the locale settings were altered while PSME agents were running, they must be restarted for the changes to take effect.

6 Appendix - 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 1-3
4 : MMP BMC
5 : RRC CPP
6 : Uplink (backplane connection)

- Add/Update VLAN (0x30):

```
<base> raw 0x38 0x30 <VLAN MSB> <VLAN LSB> <Member Bitmask> <Tagged Bitmask>
```

- Dump VLANs (0x32):

```
<base> raw 0x38 0x32
```

- Delete VLAN (0x31):

```
<base> raw 0x38 0x31 <VLAN MSB> <VLAN LSB>
```

- Set PVID (0x33):

```
<base> raw 0x38 0x33 <Port #> <VLAN MSB> <VLAN LSB>
```

- Dump PVIDs (0x34):

```
<base> raw 0x38 0x34
```

- Save VLAN Configuration (0x39):

```
<base> raw 0x38 0x39
```

7 Appendix - MYB-IMX28X reference board PSME Configuration

7.1 MYB-IMX28X board

7.1.1 Board features

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

7.1.2 Booting modes

The Processor boot mode can be selected from the BM5-BM0 pins and the LCD_RS pin. For more details, see MYC-IMX28X Datasheet and/or i.MX28x Reference Manual. Boot modes are listed below:

Table 12 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:

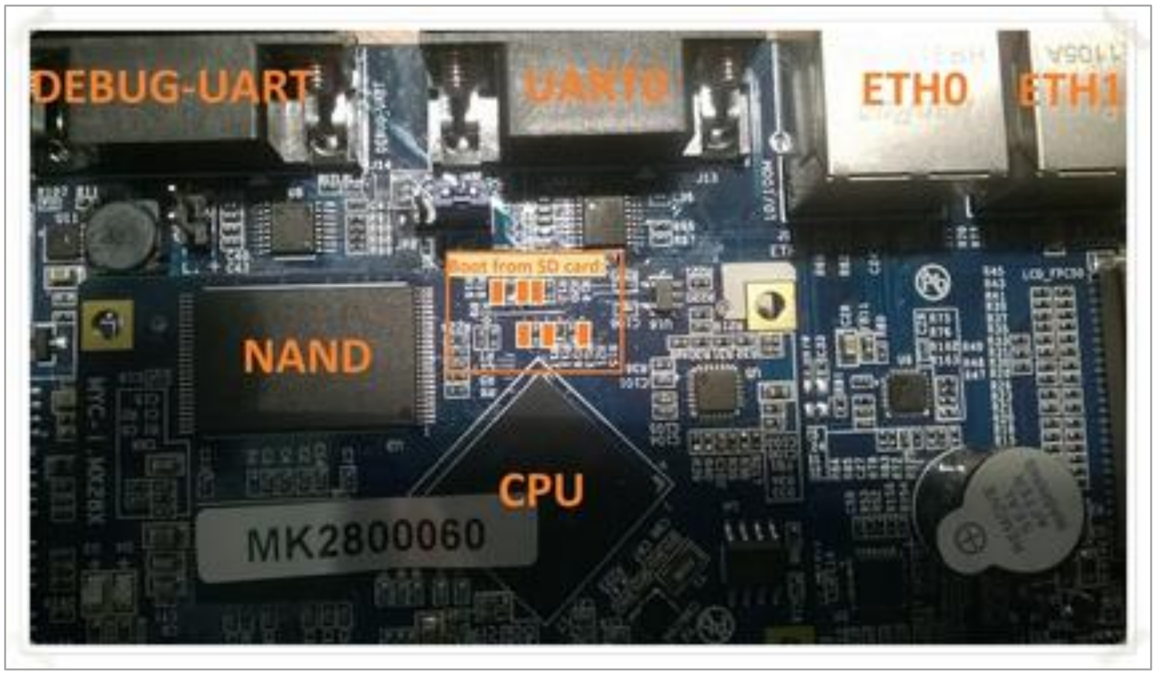
Table 13 Boot mode pins

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

7.1.3 Booting from SD Card

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

Table 14 Reference board main components



7.2 Build system configuration

7.2.1 Environment preparation

1. Clone the buildroot project:

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

2. Change directory to the buildroot directory:

```
cd buildroot
```

7.2.2 Kernel configuration

1. Configure buildroot using menuconfig:

```
make menuconfig
```

2. Edit the following sections:

    1. 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)
    ```

    2. Build options

    ```
    No need to change
    ```

3. Toolchain

```
Toolchain type (Buildroot toolchain)
*** Kernel Header Options ***
Kernel Headers (Linux 3.19.x kernel headers)
C library (glibc)
glibc version (2.20)
*** Binutils Options ***
Binutils Version (binutils 2.24)
*** GCC Options ***
GCC compiler Version (4.9.x)
[*] Enable C++ support
[*] Enable MMU support
```

4. 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
```

5. Kernel

```
[*] Linux Kernel
    Kernel version (3.19.3)
    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
```

6. Target packages
   Enable OpenSSH only for developing

```
-*- BusyBox
    Networking application --->
        [*] openssh
```

7. Filesystem images

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

8. Bootloaders

```
[*] U-Boot
(mx28evk) U-Boot board name
    U-Boot Version (2015.1)
    U-Boot binary format (u-boot.sb)
```

3. Exit from menuconfig:

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

7.3 Image creation

7.3.1 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

7.3.2 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.cm
ake -DENABLE_HTTPS=OFF ..
```

5. Compile project

```
$ make -j8
```

7.3.3 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-2015.04/tools/mxsboot sd ./output/images/u-boot.sb ./output/images
/u-boot.sd
```

7.4 Linux image configuration

7.4.1 SD Card layout

1. Locate your SD card which typically is located in `/dev/sdd` 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 1GB `/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:

    1. 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:

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

    If partitions are already mounted, please unmount them first.

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

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

3. Write bootloader to bootloader partition:

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

4. 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
```

5. 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
```

6. Prepare networking
   a. Mount the SD card:

   ```
   mount /dev/sdd3 /mnt/rootfs
   ```

   b. Edit the interface file:

   ```
   vi /mnt/rootfs/etc/network/interfaces
   ```

      i. 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
      ```

   c. Unmount the root file system:

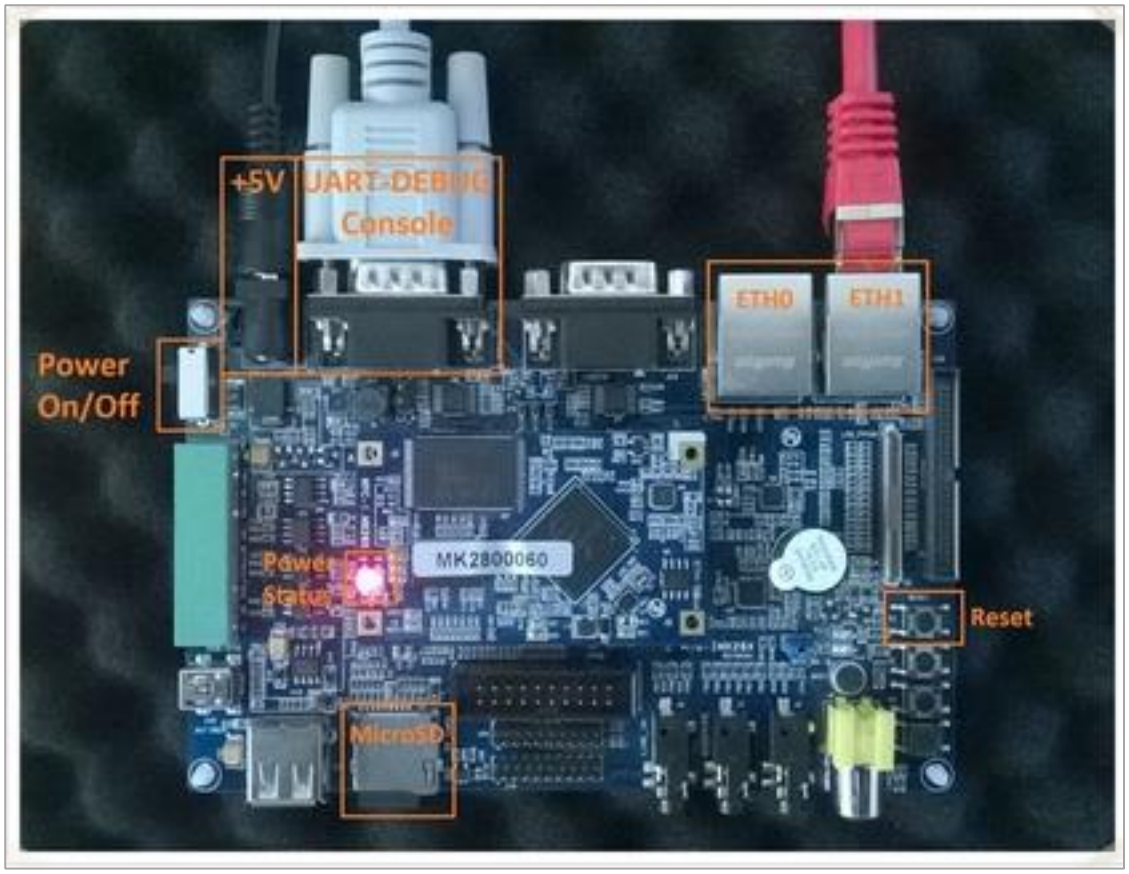
   ```
   umount /mnt/rootfs
   ```

7.5 Evaluation board configuration

7.5.1 Booting

1. Insert the MicroSD card to the MicroSD slot.
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 15 Reference board connectors



Reference Board connectors

7.5.2 Serial console settings

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

8 Appendix - Top-of-Rack switch configuration

8.1 Prerequisites

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

The provided ToR switch has 52 ports. Ports 1-48 are 1/10g SFP+ ports, while ports 49-52 are 10/40g QSFP ports. The ToR is configured with various VLANs for connectivity between rack components.

8.2 Configuration process (via CLI)

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):
    1. Make sure to use the correct a serial port on the PC
    2. Connection speed is 115200
    3. Make sure to select "Serial"
    4. Open connection and wait for the ToR to respond
3. Log into the ToR system.
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
4088	1, 49, 53, 57, 61	-
4090	1, 2	-
4091	1	49, 53, 57, 61
4092	1, 2	-
4093	1, 2, 49, 53, 57, 61	-
4094	1, 2	-

6. Type "show vlan" to list VLANs.
7. To create VLANs: type "vlan-database", then type "vlan 4090" to create VLAN 4090.
8. Do the same steps to create vlan 4088, 4090, 4091, 4092, 4093 and 4094.
9. To verify vlan creation, type "exit", then "show vlan"
10. Or to see ToR port 1 configuration, type "show interface xe1".
    1. To see all interfaces on ToR, type

```
show interface
```

11. Add tagged vlan 4088, 4090, 4091, 4092, 4093, and 4094 to port 1:
    1. Type

```
configure
```

    2. Then type

```
interface xe1
```

3. Type

```
switchport vlan add 4090, 4091, 4092, 4093, 4094 tagged
```

4. The console should look something like this:

```
config-f xe1
#switchport vlan add 4088, 4090, 4091, 4092, 4093, 4093 tagged
```

5. Type "exit" to return to config mode. Then type "exit" again to return to the main directory

12. Add tagged VLAN to port 2:

   1. Type

```
configure
```

   2. Then type

```
interface xe2
```

   3. Type

```
 switchport vlan add 4088, 4090, 4092, 4093, 4094 tagged
```

   4. What you typed should look something like this

```
config-f xe2
#switchport vlan add 4088, 4090, 4092, 4093, 4094 tagged
```

13. Use the same methods to configure all other ports

   1. Note: ports 49, 53, 57, 61 are special. To configure these ports, use "qe49", "qe53", "qe57", "qe61" instead of "xe".

14. All configured ports need to be removed from vlan 1 (all ports are configured to be on vlan1 by system default)

   1. To remove a port 1 from vlan 1 type

```
no switchport vlan add 1
```

   2. The console display should look like this:

```
(config-if xe1)
#no switchport vlan add 1
```

   3. Use this method and repeat on all other ports

15. Change the speed of ports 49, 53, 57, 61 to 10G (BDC-A) or 40G (BDC-R)

   1. For example, to change port 49 speed from 1G to 10G:

      a. type

```
configure
```

      b. then type

```
interface qe49
```

      c. type

```
speed 40000
```

    d. the console display should look like this:

```
switch (config-if qe49)
#speed 40000
```

16. Then to confirm the change, in the enabled mode, type

```
show interface qe49
```

    to print the interface details on the console.

17. Also to confirm that the associated QSFP ports are disabled:

```
show interface xe50
```

    It should show that speed is 0.

18. Now change port 1-6 speed from 10G to 1G
    1. For example, to change port 5 speed from 10G to 1G:
        a. Type

        ```
        configure
        ```

        b. Then type

        ```
        interface xe5
        ```

        c. Type

        ```
        speed 1000
        ```

19. When finished, save the configuration by typing "save configuration".

8.3 Known issues

In the event of connection problems between hosts in the tagged network, it is advised to change Maximum Transmission Unit to 1536 for ports 1-48.

1. For example, to change port 1 MTU to 1536:
    1. type:

    ```
    configure
    ```

    2. then type

    ```
    interface xe1
    ```

    3. then type

    ```
    max-frame-size 1536
    ```

    4. the console display should look like this:

    ```
    switch (config-if xe1)
    #max-speed-size 1536
    ```

9 Appendix - Drawer hardware configuration

In case of I2C communication issues on the Intel SDV platform please make sure that you have appropriate rework done.

9.1 Recommended hardware configuration

9.1.1 Rack setup configuration

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

9.2 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

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.
    1. Attach to the CM serial console (for lower power zone use Cm2)

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

    2. 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
    ```

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

```
rsa@pod-manager:~$ ipmitool -I lanp -U admin -P admin -H 1.1.1.1 mc info
Device ID                  : 37
Device Revision            : 0
Firmware Revision          : 1.03
IPMI Version               : 2.0
Manufacturer ID            : 7244
Manufacturer Name          : Unknown (0x1C4C)
```

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

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
```

9.3 Intel SDV platform

9.3.1 MMP Switch

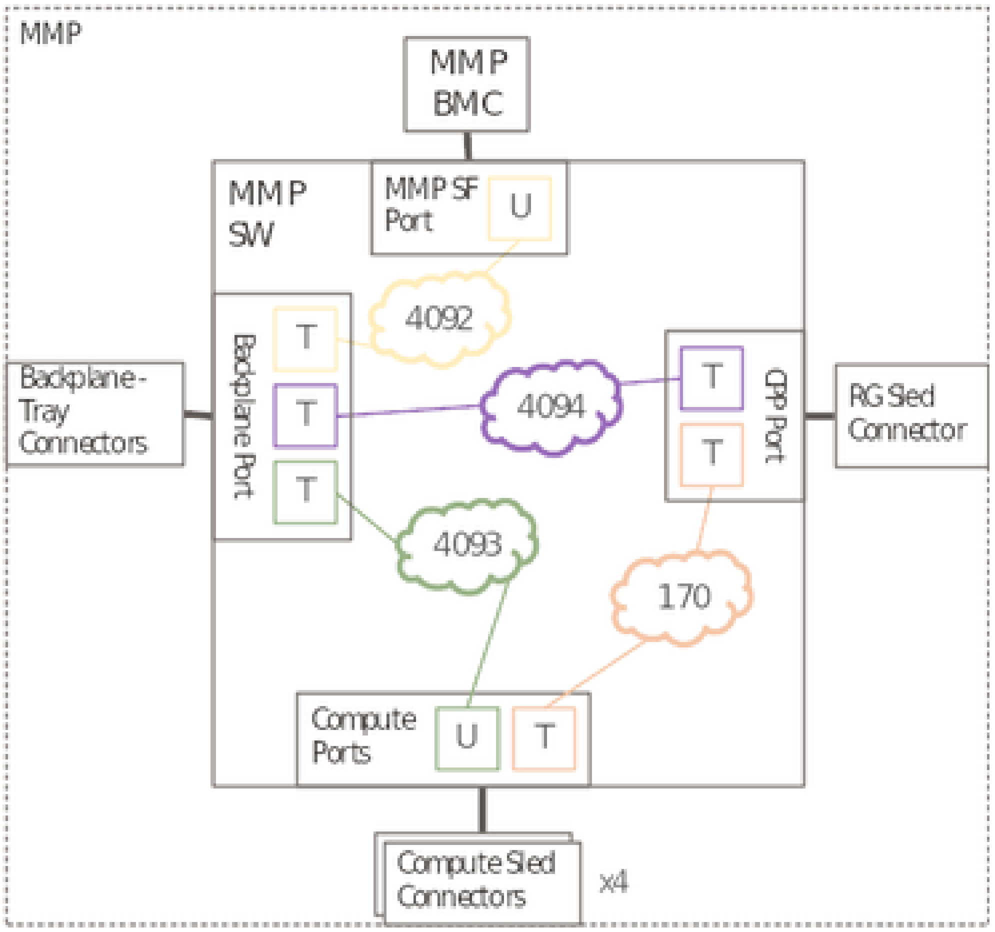
Prerequisites

- 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 16 Network configuration



Network Configuration

Configuring MMP

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

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

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

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

2. Configure VLANs (0x30 <VLAN MSB> <VLAN LSB> <Member Bitmask> <Tagged Bitmask>):
   Assuming base = `ipmitool -I lanp -U admin -P admin -H 1.1.1.1 -b 0 -t 0x24`

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

9.3.2 ONPSS Software Configuration.

To build and install Intel© Open Network Platform Switch Software (ONPSS) please follow instructions in ONPSS2_BuildGuide to build OS image and then ONPSS2_InstallGuideBDC to install it. These documentation are distributed together with ONPSS releases.

9.3.3 Default Red Rock Canyon Switch Configuration.

The left (sw0p25) and the center (sw0p29) QSFP+ connectors on the front of Red Rock Canyon Drawer are set to work as 100Gbit ports by default. Those ports should be used to connect to a neighboring switch. The right QSFP+ connector (sw0p33) is set to 40Gbit and should be connected to the ToR. The port speed is configured using system-networkd configuration files located at `/etc/system/network` and applied after OS is booted.

Note: any changes done to sw0p33 port speed using ethtool command or PSME may be overwritten by networkd at any time.

9.4 Remote iSCSI Target Blade boot

9.4.1 Prerequisites

- Please be sure that BMC and BIOS are up-to-date.
- Please be sure that Intel SDV platform networks are properly configured. SLED has access to storage management (10.2.0.x) and public (10.1.0.x) network.

9.4.2 BIOS Configuration

The PSME doesn't support UEFI boot. Due to this, user has to change UEFI boot mode in SLED BIOS to Legacy mode.

1. Connect to SLED via SOL or Serial Debug connector.
2. Enter to setup using F2 or delete.
3. Change UEFI boot mode to Legacy.
4. Save changes and restart.

10 Appendix - PSME Software installation from packages

10.1 PSME Software packages availability

The section is applicable for those who has access to RPM/DEB packages with PSME Software. Table below shows which packages are available under supported operating systems.

Table 18 Package availability

Component	Ubuntu 16.04	Fedora 23
PSME REST Server	+	+
PSME Compute SDV	-	+
PSME Network RRC	-	+
PSME Storage TGT-LVM	+	-
PSME Chassis SDV	-	+
PSME PNC SDV	+	-

Before the PSME installation from packages a **PSME Common package** should be installed. It is a set of common shared libraries and executables for the PSME provided by RPM/DEB packages.

10.2 PSME Software Fedora 23 packages

10.2.1 Installation

1. The following PSME binary *.rpm installation files can be built from sources or acquired from a pre-built binary file:
   - PSME Common
   - PSME Compute
   - PSME Network (FM10000 for Red Rock Canyon)
   - PSME Rest Server
2. Copy all PSME *.rpm files to Drawer.
3. Login into Drawer as a root.
4. Install CyMUX following the instructions in appendix: Installing CyMUX.
5. Install the PSME using rpm on Drawer:

```
# rpm -i psme-common-*.rpm
# rpm -i psme-compute-*.rpm
# rpm -i psme-network-*.rpm
# rpm -i psme-rest-server-*.rpm
# rpm -i psme-chassis-*.rpm
```

6. 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"
```

7. Reboot

```
# reboot
```

10.2.2 Update

Prerequisites

- System with installed PSME Common, PSME Compute, PSME Network and PSME Rest Server from rpm packages.

Update PSME:

1. Copy all PSME *.rpm files to Drawer.
2. Login into the Drawer as root.
3. Install CyMUX following the instructions in appendix: Installing CyMUX.
4. Update the PSME using the rpm on Drawer:

```
# rpm -U psme-common-*.rpm
# rpm -U psme-compute-*.rpm
# rpm -U psme-network-*.rpm
# rpm -U psme-rest-server-*.rpm
# rpm -U psme-chassis-*.rpm
```

5. Modify the configuration files to match your setup or replace them with previous versions if they are compatible. If `psme-network-configuration.json` file has been edited on disk, but is not actually different from one RPM to another then the edited version will be silently left in place. If a base configuration structure file has been updated, RPM installation will rename old edited file version with a .rpmsave suffix and new configuration file will be installed.

```
# cp /etc/psme/psme-compute-configuration.json.old /etc/psme/psme-compute-configur
ation.json
# cp /etc/psme/psme-rest-server-configuration.json.old /etc/psme/psme-rest-server-
configuration.json
# cp /etc/psme/psme-chassis-configuration.json.old /etc/psme/psme-chassis-configur
ation.json
```

6. Restart services if you updated psme-compute or psme-rest-server:

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

7. Reboot system if you updated psme-network:

```
# reboot
```

10.3 Storage Services Ubuntu 16.04 packages

10.3.1 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 libjsoncpp-dev libmicrohttpd-dev libossp-uuid-dev libcurl4-openssl
-dev libsysfs2 tgt lvm2 liblvm2app2.2
```

4. Install PSME Storage and PSME Rest Server:

```
dpkg -i psme-common-*.deb
dpkg -i psme-storage-*.deb
dpkg -i psme-rest-server-*.deb
```

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

```
"network-interface-name" : "enp0s20f0.4094" -> "network-interface-name" : "your_ma
nagement_interface"
"rmm-present": true -> "rmm-present": false
```

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

```
"portal-interface" : "eth0" -> "portal-interface" : "interface_for_connection_to_t
argets"
```

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 a moment while Storage components are being discovered. You can watch the progress in the console: `journalctl -fu psme-storage`.
9. In order to make Storage Services visible for Intel Rack Scale Design PODM, change the operating system hostname to begin with "storage" (it must be compatible with regular expression "^storage.*", for example "storage", "storageXYZ" or "storage-XYZ") and enable getting IP from DHCP for your management interface.
10. If Storage Service does not appear on PODM, then execute this command on PODM machine:

```
service pod-manager restart
```

10.3.2 Update

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

```
dpkg -i psme-common-*.deb
dpkg -i psme-rest-server-*.deb
dpkg -i psme-storage-*.deb
```

4. Edit the configuration files. In `/etc/psme/psme-rest-server-configuration.json` change:

```
"network-interface-name" : "enp0s20f0.4094" -> "network-interface-name" : "your_ma
nagement_interface"
"rmm-present": true -> "rmm-present": false
```

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

```
"portal-interface" : "eth0" -> "portal-interface" : "interface_for_connection_to_t
argets"
```

5. Restart the services:

```
service psme-rest-server restart
service psme-storage restart
```

6. Wait a moment while Storage components are being discovered. You can watch the progress in a file:

```
    /var/log/psme-storage.log
```

10.4 PSME PNC Ubuntu 16.04 packages

10.4.1 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 libjsoncpp-dev libmicrohttpd-dev libossp-uuid-dev libcurl4-openssl
    -dev libsysfs2
```

4. Install PSME PNC and PSME Rest Server:

```
    dpkg -i psme-common-*.deb
    dpkg -i psme-pnc-*.deb
    dpkg -i psme-rest-server-*.deb
```

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

```
    "network-interface-name" : "enp0s20f0.4094" -> "network-interface-name" : "your_ma
    nagement_interface"
    "rmm-present": true -> "rmm-present": false
```

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

```
    "network-interface-name" : "eth0" -> "network-interface-name" : "your_management_i
    nterface"
```

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

10.5 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
```

10.5.1 Signing a package

To sign a .deb package use the command below:

```
debsigs --sign=origin -k <key ID> <deb package>
```

Before signing an .rpm package you need to configure .rpmmacros file as follows:

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

To sign an .rpm package use this command:

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

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
```

10.5.2 Checking signatures

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

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

Get the fingerprint of the key:

```
gpg --fingerprint
```

Get the last 16 characters (8 bytes) of gpg-fingerprint and remove spaces. Create system keyring directory:

```
sudo mkdir -p /usr/share/debsig/keyrings/<fingerprint>
```

Import public key to the system keyring:

```
sudo gpg --no-default-keyring --keyring /usr/share/debsig/keyrings/<fingerprint>/psme.g
pg --import <GPG public key>
```

Create directory for the policy document:

```
sudo mkdir -p /etc/debsig/policies/<fingerprint>
```

Create XML policy document. Use the following example:

```
<?xml version="1.0"?>
<!DOCTYPE Policy SYSTEM "http://www.debian.org/debsig/1.0/policy.dtd">
<Policy xmlns="http://www.debian.org/debsig/1.0/">
    <Origin Name="rmm" id="EBDEEB785B35B559" Description="PSME package"/>
    <Selection>
        <Required Type="origin" File="psme.gpg" id="EBDEEB785B35B559"/>
    </Selection>
    <Verification MinOptional="0">
        <Required Type="origin" File="psme.gpg" id="EBDEEB785B35B559"/>
    </Verification>
</Policy>
```

Save the file in the policies directory you created under the name psme.pol. Replace IDs with the fingerprint of your public key.

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

```
sudo debsig-verify <psme package>.deb
```

On Fedora system the procedure is less complicated. Import the GPG public key file:

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

To check the signature in an .rpm file run:

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

11 Appendix - Network agent configuration

11.1 Fedora 23 packages

PSME Network Agent should not be compiled on Ubuntu. Only Fedora 23 is officially supported by ONPSS.

The PSME Network Agent requires additional packages for compilation. If you cannot download them from locations below, then contact ONPSS team for access.

Install a package needed to build ACL code:

```
wget http://ons-archive.jf.intel.com/pub/onpss/repos/fedora/23/fm10kd/ww2016.08.sdnd/fm
10kd-devel-0.5.0-201602091212.git966044bc.fc23.x86_64.rpm
rpm --nodeps -i fm10kd-devel-0.5.0-201602091212.git966044bc.fc23.x86_64.rpm
```

11.1.1 Access Control List

Please refer to the ONPSS2 API Guide for information about rule priorities and limitation on number of ACLs that can be created.

11.2 Configuration parameters

The following Port and VLAN parameters can be configured through the network agent configuration.json file. If a given parameter is not specified then it is set to default value.

Table 19 Ports parameter

Name	Description	Possible values	RRC Switch supported	RRC Switch default
id	Name of switch port to be configured	Port identifiers as string	Yes	-
portType	Port type	"Downstream", "Upstream", "MeshPort", "Unknown"	"Downstream", "Upstream"	"Upstream"
link technology	Port Link technology	"Ethernet", "PCIe", "Unknown"	"Ethernet", "PCIe"	"Ethernet"
link state	Port link administrative state	"Up", "Down"	All	"Down"
ethmod	Port ethernet mode	"Default", "Mode1000BaseKX", "Mode1000BaseX", "Mode10GBaseCR", "Mode10GBaseCX4", "Mode10GBaseKR", "Mode10GBaseKX4", "Mode10GBaseSR", "Mode24GBaseCR4", "Mode24GBaseKR4", "Mode2500BaseX", "Mode40GBaseCR4", "Mode40GBaseKR4", "Mode40GBaseSR4", "Mode6GBaseCR", "Mode6GBaseKR", "ModeAN73", "ModeDisabled", "ModeSGMII", "ModeXAUI", "ModeXLAUI"	No	-
autoneg	Port auto-negotiation	"Default", "SGMII", "Clause37", "Clause73"	No	-

Table 20 VLANs

Name	Description	Possible values	RRC Switch supported	RRC Switch default
id	VLAN identifier	1-4095	No	-
tagged ports	List of tagged Ports to be added to the VLAN specified by "id" field	1-MAX_PORTS as string	No	-
untagged ports	List of untagged Ports to be added to the VLAN specified by "id" field.	1-MAX_PORTS as string	No	-

Table 21 Public VLANs

Name	Description	Possible values	RRC Switch supported	RRC Switch default
-	List of public VLAN identifiers	1-MAX_PORTS as integer	No	-

12 Appendix - Miscellaneous

12.1 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
```

12.2 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.
```

12.3 Installing CyMUX

1. Go to *https://github.com/01org/intelRSD/tree/master/tools/CyMUX* to build CyMUX from source.
2. Install cyMUX dependency:

```
    $ rpm -i libcyusbserial-1.0-0.x86_64.rpm
```