Intel® Rack Scale Design Pod Manager

User Guide
Software Release 1.2

July 2016

Revision 003

Contents

Tables

Revision History

Revision	Description	Date
1.0	Initial release	April 5, 2016

§

1 Introduction

1.1 Scope

This document contains information about the installation and configuration of Intel® Rack Scale Design Pod Manager (PODM) Software Release 1.2.

1.2 Intended audience

The intended audiences for this document include:

- Software Vendors (ISVs) of pod management software, who make use of PODM to discover, compose, and manage drawers, regardless of the hardware vendor and/or manage drawers in a multivendor environment.
- Software Vendors (OxMs) of PSME firmware who would like to provide Intel® Rack Scale Design PODM API on top of their hardware platform.

1.3 Terminology

Table 1 Terminology

Term	Definition
ACL	Access Control List
CA	Certificate Authority
CM	Control Module
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LAG	Link Aggregation Group
LUI	Linux* Utility Image
MMP	Management Midplane
PKCS #12	Personal Information Exchange Syntax Standard
POD	A physical collection of multiple racks
PODM	Pod Manager
PSME	Pooled System Management Engine
Redfish	DMTF standard, for more information refer to: https://www.dmtf.org/standards/redfish
REST	Representational state transfer
RSA	Public key cryptosystem
SSL	Secure Socket Layer
TLS	Transport Layer Security
URI	Uniform Resource Identifier
UUID	Universally Unique Identifier
URL	Uniform Resource Locator

1.4 References

Table 2 Reference documents

Doc ID	Title	Location
332868	Intel® Rack Scale Design GAMI API Specification	*http://intel.com/intelRSD*
332869	Intel® Rack Scale Design Pod Manager REST API Specification	*http://intel.com/intelRSD*

Doc ID	Title	Location
332870	Intel® Rack Scale Design Pod Manager Release Notes	*http://intel.com/intelRSD*
332871	Intel® Rack Scale Design Pod Manager User Guide	*http://intel.com/intelRSD*
332873	Intel® Rack Scale Design PSME REST API Specification	*http://intel.com/intelRSD*
332872	Intel® Rack Scale Design PSME Release Notes	*http://intel.com/intelRSD*
332874	Intel® Rack Scale Design PSME User Guide	*http://intel.com/intelRSD*
332877	Intel® Rack Scale Design RMM REST API Specification	*http://intel.com/intelRSD*
332876	Intel® Rack Scale Design RMM Release Notes	*http://intel.com/intelRSD*
332875	Intel® Rack Scale Design RMM User Guide	*http://intel.com/intelRSD*
332878	Intel® Rack Scale Design Storage Services API Specification	*http://intel.com/intelRSD*
332936	Intel® Rack Scale Design BIOS/BMC Tech Guide	*http://intel.com/intelRSD*
332937	Intel® Rack Scale Design Architectural Requirements Specification	*http://intel.com/intelRSD*
334611	Intel® Rack Scale Design Getting Started Guide	*http://intel.com/intelRSD*
n/a	Scalable Platforms Management API	*http://dmtf.org/standards/redfish*

1.5 Typographical conventions

Notation used in JSON serialization description:

- Values in italics indicate data types instead of literal values.
- Characters are appended to items to indicate cardinality:
  - "*" (wildcard)
  - "?" (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 Requirements & Toolchain

2.1 Requirements

- Machine (with at least 32 GB HDD/SSD storage capacity) with installed and preconfigured Ubuntu* 14.04.4
- Development machine with OpenJDK 1.8 installed
- Internet access must be configured on development machine
- Hostname of PSME services must start with string “psme”, hostname of storage services must start with string “storage” & hostname of deep discovery LUI service must be set to string “lui”. Hostname of RMM service must start with string "rmm"

  Example: psme1, storage1, lui, rmm1
- HTTPS communication with machines managed by POD Manager (i.e. with PSME, storage, LUI and RMM services) requires setting correct system time on machines hosting those services. NTP server must be installed and configured on machine hosting POD Manager. Whereas, NTP client must be installed and configured on machines hosting PSME, storage, LUI and RMM service.

2.2 Toolchain

Component	Name	Version
Compiler	OpenJDK	1.8
Java EE	Full Profile	7.0
Libraries	guava	18.0
	jackson-annotations	2.6.0 and 2.6.3
	jackson-core	2.6.3
	jackson-databind	2.6.3
	jackson-datatype-jsr310	2.6.3
	jackson-jaxrs-base	2.6.3
	jackson-jaxrs-json-provider	2.6.3
	jackson-module-jaxb-annotations	2.6.3
	javassist	3.16.1-GA
	modelmapper	0.7.5
	blueprints-core	2.6.0
	logback-classic	1.1.1
	logback-core	1.1.1
	commons-beanutils	1.8.3
	commons-collections	3.2.1
	commons-digester	1.8.1
	commons-io	2.1
	commons-lang	2.4
	commons-logging	1.2
	commons-validator	1.4.1
Application server	WildFly	9.0.2
Database	OrientDB	2.1.5

§

3 POD Manager Prerequisite

3.1 Core Ubuntu* installation

Download Ubuntu Server 14.04.4. Boot target machine using this image and follow installation instructions.

3.2 Configure server internet access

Much of the PODM software installation requires access to public software repositories on the Internet. Confirm that the configurations for the server network, firewall, and proxy allow the appropriate server access.

3.3 Prerequisites

The following sections address prerequisites required to install Pod Manager on Ubuntu.

3.3.1 Adding PPA repository for OpenJDK

Follow the steps below to add a PPA repository for OpenJDK:

1. Add OpenJDK 1.8 repository to */etc/apt/sources.list* file by adding the following line:

   ```
   deb http://ppa.launchpad.net/openjdk-r/ppa/ubuntu trusty main
   ```

2. Download key:

   ```
   sudo su -
   apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 86F44E2A
   ```

   Note: If applicable, user should pass appropriate proxy information to apt-key command as mentioned below:

   "apt-key adv --keyserver-options http-proxy="http://<username>:<password>@<proxy_server>:<proxy_port>" --keyserver keyserver.ubuntu.com --recv-keys 86F44E2A"

3. Update the *apt-get* repositories:

   ```
   sudo apt-get update
   ```

3.3.2 Installing OpenJDK Java development kit

Install OpenJDK 1.8 Java Development Kit package:

```
sudo apt-get install openjdk-8-jdk
java -version
```

3.3.3 Verify OpenJDK installation

Check the Java version using following command:

```
java -version
```

If the command above does not show the default Java version as OpenJDK 1.8, then execute following command to set Java defaults:

```
sudo update-alternatives --config java
```

3.3.4 Additional package installation

Install additional packages using the following commands:

```
sudo apt-get install isc-dhcp-server
```

```
sudo apt-get install openssh-server
sudo apt-get install python
sudo apt-get install tftpd-hpa
sudo apt-get install ntp
sudo apt-get install vlan
sudo apt-get install openjdk-8-jre-headless
sudo apt-get install acl
```

3.3.5 GRUB configuration

1. Configure system to use ethX naming convention for Ethernet interfaces.
2. Edit the */etc/default/grub* file and comment the following variables:

```
#GRUB_HIDDEN_TIMEOUT
#GRUB_HIDDEN_TIMEOUT_QUIET
```

3. 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
```

4. Save the file and apply changes:

```
sudo update-grub
```

5. Restart system for changes to take effect.

3.3.6 NTP configuration

1. Edit /etc/ntp.conf file and add following lines:

```
tos maxdist 16
restrict 10.3.0.0   255.255.252.0 nomodify notrap
restrict 10.2.0.0   255.255.255.0 nomodify notrap
```

2. Restart NTP service:

```
sudo service ntp restart
```

§

4 POD Manager Source Code Compilation

4.1 Acquiring Pod Manager v1.2 source code

Refer to the Software Package Contents section of the Intel® Rack Scale Design Pod Manager Release Notes for the latest posted version of the PODM source code.

4.2 Custom iPXE compilation

1. This step should be carried out on an external machine with access to Internet. The following packages must be installed on Ubuntu 14.04.2:

```
sudo apt-get install build-essential
sudo apt-get install genisoimage
sudo apt-get install git
sudo apt-get install liblzma-dev
```

2. Clone the iPXE repository

```
git clone https://git.ipxe.org/ipxe.git
```

3. Copy the following file from PODM source package to ipxe/src/ directory

```
SW/external/ipxe-dhcp
```

4. Compile iPXE by executing following command from ipxe/src directory

```
make EMBED=ipxe-dhcp
```

5. Transfer bin/undionly.kpxe to target machine.

   Note: In case of iPXE dependency or compilation issues, visit http://ipxe.org/docs.

4.2.1 Custom iPXE installation

1. Create /srv/tftp directory on target machine.

```
sudo mkdir -p /srv/tftp
```

2. On target machine, copy undionly.kpxe to /srv/tftp.
3. Make symlink to podmipxe.0.

```
cd /srv/tftp
sudo ln -s undionly.kpxe podmipxe.0
```

4.3 Compilation and deployment of POD Manager application

1. On the development machine, change directory to the PODM source's root and use gradle to compile the PODM package. This command creates necessary Debian* packages under source root directory which could be used to install PODM.:

```
./gradlew buildDeb
```

   CAUTION! In a rare case, the command can fail with SSL exception which a known issue on Ubuntu with OpenJDK 1.8 arising due to "/etc/ssl/certs/java/cacerts" file NOT being generated properly.

   Note: If applicable, user should pass appropriate proxy information to gradlew script through "-*DproxyHost=<sample proxy> -DproxyPort=<sample port>*" option.

   If the scenario described above occurs, execute the following command:

```
sudo /var/lib/dpkg/info/ca-certificates-java.postinst configure
```

2. Thereafter, re-run command used for compilation.
3. To find all packages generated by above command, please navigate to source root directory and execute:

```
find ./ -iname *.deb
```

4. PODM packages should be installed in order listed in the table below.

Table 3 PODM package order of installation

Package	Required	Description
pod-manager	Yes	Creates user for services. Also contains web server, database, update scripts and configuration for Pod Manager
pod-manager-networking	Yes	Contains configuration for network, DHCP, NTP, TFTP. CAUTION! This package overrides settings for network interfaces in the user's system.

Note: User can install POD Manager .deb packages using following command:

```
cd <PODM .deb package directory>
sudo dpkg -i *.deb
```

§

5 Application Server & Database Configuration

POD Manager requires WildFly application server & OrientDB database. These packages are automatically installed and deployed after installing PODM .deb packages as mentioned in section 4.3.

§

6 Securing POD Manager North Bound API

1. Custom certificates can be configured for secure HTTPS PODM North bound API communication. All certificates are stored in a default keystore located at:

   ```
   /var/lib/pod-manager/keystore.jks
   ```

2. It is possible to change keystore, its password, localization and stored keys. Keystore path, password and alias are provided by an entry in “keystore” xml node contained in file:

   ```
   /opt/pod-manager/wildfly/standalone/configuration/standalone.xml
   ```

3. Detailed information about WildFly SSL connections configuration is provided here:

   ```
   https://docs.jboss.org/author/display/WFLY9/Detailed+Configuration
   ```

4. WildFly supports password hashing for the keystore entry in standalone.xml file using VAULT tool. Detailed information is provided here:

   ```
   https://developer.jboss.org/wiki/JBossAS7SecuringPasswords
   ```

   CAUTION! VAULT uses container to store hashed password. Once the user changes the VAULT container, the configuration files MUST be updated as described in section 7.4.4.

   Keystore creation command example:

   ```
   keytool -genkey -alias <alias> -keyalg RSA -keysize 2048 -keystore
   <keystore_name>
   ```

   Note: Recommended key size is at least 2048 bits. Key with lower size might lead to compatibility issues.

5. Custom certificate import example:

   ```
   keytool -importcert -file <certificate> -keystore <keystore> -alias
   <alias>
   ```

6. For more information please use following command:

   ```
   man keytool
   ```

6.1 Basic authentication management

Default login credentials for POD Manager are:

```
User: admin
Password: admin
```

Basic authentication mechanism can be managed by “add-user” utility

```
/opt/pod-manager/wildfly/bin/add-user.sh
```

§

7 Securing POD Manager South Bound API

POD Manager on southbound API can act as both client and server. Southbound API communication is established using TLSv1.2 protocol.

While establishing secure connection, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 cipher suite is being used. This means that cipher suite implements Elliptic – Curve Diffie – Hellman Ephemeral key exchange. Elliptic – Curve Digital Signature Algorithm, with AES-128 as the block cipher and SHA-256 HMAC for the authentication hash.

More information about this cipher suite can be found here: *https://tools.ietf.org/html/rfc5289*

Important points to note:

- For communication to be properly established by TLS with client authentication at server side, all communicating components MUST have synchronized time. Time must be set up correctly to fulfill certificate validation period. “pod-manager-config” package configures NTP service on PODM to be used for time synchronization between PODM and other services (RMM, PSME, LUI, Storage Service). Each service using PODM as NTP server must have correct NTP client configuration.
- PODM uses “TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256” cipher suite. This means that to create correct key SHA256 with ECDSA signature algorithm must be used.
- Expected format of client certificate CA file or signed client certificate is PKCS #12 (.pfx).

Note: It is assumed that PKCS #12 file with client signed certificate is well formatted container that includes full certification chain.

7.1 POD Manager as server

PODM acts as server while receiving events from clients that are not authenticated since PODM does not have client certificate for PSME service. However, communication between PODM and PSME remains over SSL channel.

7.2 POD Manager as client

PODM acts as a client when obtaining information from external services. PODM is configured to send its own certificate to be authenticated by server. LUI does not authenticate PODM on SB API interface. Also, PODM does not perform server authentication.

PODM uses CA signed certificate. CA certificate should be added to trust store of external service (Ex: RMM/PSME/Storage)

PODM (client) certificate is stored in a keystore file:

```
/var/lib/pod-manager/client.jks
```

CA certificates can be created manually & corresponding keys are stored in:

```
/var/lib/pod-manager/ca
```

CA certificate and key are also packed to file in PKCS #12 format to be later used in other locations.

TLS is turned ON by default & can be turned OFF by modifying following configuration file:

```
/etc/pod-manager/service-connection.json
```

As per requirement, please set “true” or “false” value for each component.

Example to set TLS OFF for PSME:

```
{
  "SslEnabledForRmm" : true,
  "SslPortForRmm" : 8091,
  "DefaultPortForRmm" : 8090,
  "SslEnabledForPsme" : false,
  "SslPortForPsme" : 8443,
  "DefaultPortForPsme" : 8888,
  "SslEnabledForRss" : true,
  "SslPortForRss" : 8443,
  "DefaultPortForRss" : 8888,
  "SslEnabledForLui" : true,
  "SslPortForLui" : 8443,
  "DefaultPortForLui" : 8888
}
```

7.3 Southbound API configuration

PODM southbound API must be properly configured to establish secure connection by satisfying points below:

Client certificate must be added to keystore located at:

```
/var/lib/pod-manager/client.jks
```

Keystore 'client.jks' is being protected by password. To hash password WildFly hashing tool called VAULT is being used. Detailed information can be obtained from: *https://developer.jboss.org/wiki/JBossAS7SecuringPasswords*

CAUTION! VAULT uses container to store hashed password. Once the user changes the VAULT container, the configuration files MUST be updated as described in section 7.4.4.

- Client certificate must be signed by known CA. CA certificate may be self-signed certificate
- To create correct key SHA256 with ECDSA signature algorithm must be used
- All servers that will authenticate PODM must have the CA certificate provided

CAUTION! Certificate distributed with PODM has limited validation period. Check validity period for certificates before using them. System time cannot exceed certificate expiration date. Default exported certificate distributed with PODM is located at:

```
/var/lib/pod-manager/root.crt
```

To check certificate validation period from client.jks file user can use keytool:

```
keytool -v -list -keystore client.jks
```

7.4 Certificate management

Certificates can be created by user or imported to mentioned keystore using tools mentioned in this section.

CAUTION! Certificates distributed with PODM has limited validity. Before you install any certificate check its validity. Also, system time cannot exceed certificate expiration date.

To check certificate validation period from client.jks file, please use keytool:

```
keytool -v -list -keystore client.jks
```

7.4.1 Client certificate management

Script to generate client certificate is located at:

```
/usr/bin/pod-manager-certificate-creation.sh
```

Script must be executed with root privileges.

```
sudo /usr/bin/pod-manager-certificate-creation.sh
```

Print script usage using --help option:

```
sudo /usr/bin/pod-manager-certificate-creation.sh --help
```

Example 1: Create new client certificate and sign it with newly created CA certificate:

```
sudo pod-manager-certificate-creation.sh --ca-certificate
"/C=PL/ST=State/L=locality/O=organization/CN=fqdn_or_ip" --client-certificate
"C=PL,ST=State,L=locality,O=organization,CN=fqdn_or_ip"
```

Above command creates client certificate under /var/lib/pod-manager and CA certificate is created under /var/lib/pod-manager/ca.

Example 2: Create new client certificate and sign it with existing CA certificate:

```
sudo pod-manager-certificate-creation.sh --ca-certificate
/path/to/ca/file/ca.pfx --client-certificate
"C=PL,ST=State,L=locality,O=organization,CN=fqdn_or_ip"
```

Above command creates a new client certificate under /var/lib/pod-manager. Whereas, extracted CA certificate and private key will be copied under /var/lib/pod-manager/ca directory.

Example 3: Create new client certificate and sign it with provided CA certificate - using interactive mode.

```
sudo /usr/bin/pod-manager-certificate-creation.sh --ca-certificate
"/path/to/ca/file/ca.pfx"
```

During this process, the user will be asked to pass client certificate attributes manually. Follow instructions displayed on screen to complete the process. Upon completion, above command creates new client certificate under /var/lib/pod-manager directory and extracted CA certificate & private key are copied under /var/lib/pod-manager/ca.

CAUTION! Do not remove keystore file created after certificate generation process for all above examples. This container is used by PODM application to configure SSL Connection. Additionally, note the escape sequence used to escape “white space” and the existing CA certificate path.

```
/var/lib/pod-manager/client.jks
```

Note: When using TLS configuration on PSMEs, authentication from the PODM to PSMEs is certificate based and PSMEs rest servers will not communicate with clients that do not perform certificate based authentication. To access PSMEs APIs directly, such as with a web browser or curl, it is necessary to export the client certificate from POD Manager for use with other HTTP clients. Following steps describe this process for cURL client:

Generate a PKCS #12 key based on the root key:

```
sudo keytool -importkeystore -srckeystore /var/lib/pod-manager/client.jks -
srcstoretype JKS -destkeystore client.p12 -deststoretype pkcs12
```

Create .pem certificate file using output of the above command:

```
openssl pkcs12 -clcerts -nodes -in client.p12 -out client.pem
```

Now this .pem file can be used by cURL, or imported into a web browser such as Firefox* or Chrome*.

Example usage with cURL:

```
curl -k -I --cert client.pem --cacert /var/lib/pod-manager/root.crt -X GET
https://<psme_ip>:8443/redfish/v1/
```

7.4.2 Certificate import tool

Script to import signed client certificate to container is located at:

```
/usr/bin/pod-manager-certificate-import.sh
```

Expected client certificate format is PKCS #12

Note: Assumption made here is that signed certificate container is of type PKCS #12 and contains certificate chain of root and client certificate.

Script must be executed with root privileges

```
sudo /usr/bin/pod-manager-certificate-import.sh
```

Print script usage via

```
sudo /usr/bin/pod-manager-certificate-import.sh --help
```

Note: For the example below, it is assumed that user signed certificate container is of type PKCS #12 containing full certificate chain.

Example:

```
sudo pod-manager-certificate-import.sh -c /path/to/ca/file/ca.pfx
```

Above command creates container of type JKS with imported client certificate signed by CA certificate, located at:

```
/var/lib/pod-manager/client.jks
```

7.4.3 CA certificate distribution

The CA certificate must be propagated to components that will authenticate PODM. These components are RMM, PSME and Storage Service.

Important points to note:

- Default CA certificates provided with PODM packages are located at /var/lib/pod-manager/root.crt.
- Certificates created using "pod-manager-certificate-creation.sh" script are located at /var/lib/pod-manager/ca/root.crt.

7.4.3.1 RMM & PSME

The CA certificate must be copied to RMM. RMM expects file named “podm.cert”. Rename the CA certificate file if necessary and copy it to:

```
/etc/rmm/podm.cert
```

RMM will then propagate this certificate to all PSME’s in the rack RMM manages.

Note: In cases where RMM is not used, certificate MUST be copied manually to a specific location on each PSME at following location:

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

CAUTION! In above scenario, PSME configuration file “/etc/psme/psme-rest-server-configuration.json” MUST also be updated NOT to expect certificate from RMM as below:

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

7.4.3.2 Storage services

The CA certificate must be copied to Storage Service. Storage service expects file named “ca.crt”. Rename the CA certificate file if necessary and copy it to location below:

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

Note: In cases where RMM is not used, certificate MUST be copied manually to a specific location on each PSME at following location:

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

CAUTION! In above scenario, Storage service PSME REST server configuration file “/etc/psme/psme-rest-server-configuration.json” MUST also be updated NOT to expect certificate from RMM as below:

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

7.4.4 VAULT container update

Vault is used to read the keystore password by PODM while reading the certificate from keystore files, and by script to generate/import certificates to set up the keystore password.

The VAULT container can be changed. For more information refer to:
https://developer.jboss.org/wiki/JBossAS7SecuringPasswords

Once user change VAULT container <vault> entry values (KEYSTORE_URL, KEYSTORE_PASSWORD, KEYSTORE_ALIAS, SALT, ITERATION_COUNT, ENC_FILE_DIR) MUST be updated in two files:

WildFly configuration file located at:

```
/opt/pod-manager/wildfly/standalone/configuration/standalone.xml
```

Scripts configuration file “vault.json” used to generate/import certificates located at:

```
/var/lib/pod-manager/vault/vault.json
```

§

8 Using POD Manager with RMM Service

POD Manager required that RMM software must be installed on external machine which meets following requirements:

- VLAN package should be installed on the OS.
- When running RMM on Ubuntu host, update /etc/modules file by adding following line:

  ```
  8021q
  ```
- Host should be part of VLAN 4092 & 4094.
- Hostname of RMM service must start with string rmm
- Machine should be able to communicate with CM's via USB connection.
- Pod Manager certificate is present under /etc/rmm/ directory. For more information please refer to section 7.4.3

§

9 Storage Service Configuration

Storage service can be configured in two ways:

- Storage service working on drawer's computer system
- Storage service working on external host attached to Rack's storage network

Note: Linux image for storage service host must be prepared with appropriate rules and requirements as described in Intel® Rack Scale Design PSME User Guide.

9.1 Storage service working on drawer's computer system

To have storage service working on drawer's computer system user must burn storage service's image onto computer system's storage device. Next, user should allocate (with local boot option enabled) and assemble new node with this exact computer system.

CAUTION! In case configured storage system is NOT assembled via PODM REST API but simply powered on through e.g. sending IPMI power on request, system may be targeted for deep discovery which would reboot the computer system and leave it in powered off state.

Note: Hostname for storage service must start with string "storage".

9.2 Storage service working on external host attached to Rack's storage network

User can simply attach external host with storage service image to Rack's storage network (possibility of additional TOR configuration).

Note: Hostname for storage service must start with string "storage".

§

10 Accessing POD Manager

The following section describes how to access POD Manager Software.

10.1 Starting OrientDB service

```
sudo service orientdb start
sudo service orientdb status
```

If running, command should return following message:

```
OrientDB server daemon is running with PID: <PID>
```

To access OrientDB web-interface, visit:

```
http://<target_machine_ip>:2480
```

Default login credentials for OrientDB are:

```
User: administrator
Password: rsa
```

10.2 Starting WildFly service

```
sudo service wildfly start
sudo service wildfly status
```

If running, the command should return the following message:

```
* Wildfly Application Server is running with PID <PID>
```

10.3 Accessing POD Manager REST API

POD Manager Service can be accessed at the following address:

```
https://<target_machine_ip>:8443/redfish/v1
```

It is recommended to install JSONView plugin for web browser.

Table 4 Available resources

Resource	URI
Service Root	/redfish/v1
Chassis Collection	/redfish/v1/Chassis
Chassis	/redfish/v1/Chassis/{chassisID}
Computer System Collection	/redfish/v1/Systems
Computer System	/redfish/v1/Systems/{systemID}
(Computer System) Ethernet Interface Collection	/redfish/v1/Systems/{systemID}/EthernetInterfaces
(Computer System) Ethernet Interface	/redfish/v1/Systems/{systemID}/EthernetInterfaces/{nicID}
(Computer System) VLAN Network Interface Collection	/redfish/v1/Systems/{systemID}/EthernetInterfaces/{nicID}/VLANs
(Computer System) VLAN Network Interface	/redfish/v1/Systems/{systemID}/EthernetInterfaces/{nicID}/VLANs/{vlanID}
Processor Collection	/redfish/v1/Systems/{systemID}/Processors
Processor	/redfish/v1/Systems/{systemID}/Processors/{processorID}
Memory Chunks Collection	/redfish/v1/Systems/{systemID}/MemoryChunks
Memory Chunk	/redfish/v1/Systems/{systemID}/MemoryChunks/{chunkID}

Resource	URI
Dimm Config Collection	/redfish/v1/Systems/{systemID}/DimmConfig
Dimm Config	/redfish/v1/Systems/{systemID}/DimmConfig/{dimmConfigID}
Adapter Collection	/redfish/v1/Systems/{systemID}/Adapters
Adapter	/redfish/v1/Systems/{systemID}/Adapters/{adapterID}
Device Collection	/redfish/v1/Systems/{systemID}/Adapters/{adapterID}/Devices
Device	/redfish/v1/Systems/{systemID}/Adapters/{adapterID}/Devices/{deviceID}
Manager Collection	/redfish/v1/Managers
Manager	/redfish/v1/Managers/{managerID}
Network Service	/redfish/v1/Managers/{managerID}/NetworkService
(Manager) Network Interface Collection	/redfish/v1/Managers/{managerID}/EthernetInterfaces
(Manager) Network Interface	/redfish/v1/Managers/{managerID}/EthernetInterfaces/{nicID}
(Manager) VLAN Network Interface Collection	/redfish/v1/Managers/{managerID}/EthernetInterfaces/{nicID}/VLANs
(Manager) VLAN Network Interface	/redfish/v1/Managers/{managerID}/EthernetInterfaces/{nicID}/VLANs/{vlanID}
Storage Service Collection	/redfish/v1/Services
Storage Service	/redfish/v1/Services/{serviceID}
Remote Target Collection	/redfish/v1/Services/{serviceID}/Targets
Remote Target	/redfish/v1/Services/{serviceID}/Targets/{targetID}
Logical Drive Collection	/redfish/v1/Services/{serviceID}/LogicalDrives
Logical Drive	/redfish/v1/Services/{serviceID}/LogicalDrives/{driveID}
Physical Drive Collection	/redfish/v1/Services/{serviceID}/Drives
Physical Drive	/redfish/v1/Services/{serviceID}/Drives/{driveID}
Fabric Switch Collection	/redfish/v1/EthernetSwitches
Fabric Switch	/redfish/v1/EthernetSwitches/{switchID}
Fabric Switch Port Collection	/redfish/v1/EthernetSwitches/{switchID}/Ports
Fabric Switch Port	/redfish/v1/EthernetSwitches/{switchID}/Ports/{portID}
(Switch Port) VLAN Network Interface Collection	/redfish/v1/EthernetSwitches/{switchID}/Ports/{portID}/VLANs
(Switch Port) VLAN Network Interface	/redfish/v1/EthernetSwitches/{switchID}/Ports/{portID}/VLANs/{vlanID}
Composed Node Collection	/redfish/v1/Nodes
Composed Node	/redfish/v1/Nodes/{nodeID}

Please refer Intel® Rack Scale Design Pod Manager API Specification document for detailed description of above resources.

10.4 Verifying POD Manager installation

On the target machine, the Pod Manager application should be accessible at:
https://<target_machine_IP>:8443/redfish/v1

If the application is not available, check for deployment error logs in the following locations:

```
/var/log/wildfly/console.log
/opt/pod-manager/wildfly/standalone/log/server.log
```

Verify whether pre-requisites from section 2, section 3, section 4 and section 5 are met.

While making GET/POST REST calls using a REST client, please remember to user basic access authentication method headers.

§

11 Deep Discovery

11.1 Introduction

In general, PSME is responsible for exposing information about compute modules. Currently, BMC does not provide all the required information. As a workaround, a Linux Utility Image (LUI) can be used to boot the compute modules and perform deep discovery of resources. This image includes a PSME compliant service that POD Manager can read from. POD Manager reads the basic BMC collected data from PSME, then boots LUI on each blade. The data visible in the PSME APIs will only be the BMC collected data, whereas the PODM merges both the basic BMC data from the PSME and the additional data discovered from the LUI environment.

11.2 General requirements, assumptions and limitations

- POD Manager and PSME has a property DiscoveryState.
- During deep discovery process, the blade cannot be used for any other process (for example - allocation).
- Networks/VLANs used:
  - iPXE/Deep discovery: Storage Access Network.
  - LUI → POD Manager: Separate dedicated VLAN.
- To prevent power and bandwidth spikes the deep discovery process is staggered. How staggering is performed is configurable.
- Resource properties should be obtained either from PSME or from deep discovery, but not from both. It is necessary to avoid overwriting data during slow poll process. Expected data sources for specific properties are currently hardcoded, but should be configurable in future. If customization is required, source code has to be changed and application must be rebuilt.

11.3 Building Linux* Utility Image (LUI)

Please refer to Intel® Rack Scale Design PSME User Guide to get information regarding prerequisites.

11.3.1 Preparing rootfs directory

Please refer to Intel® Rack Scale Design PSME User Guide to get information on preparing rootfs directory.

11.3.2 Building LUI

Please refer to Intel® Rack Scale Design PSME User Guide to get information on building LUIs.

11.4 Configuring deep discovery

To enable Deep Discovery POD Manager requires LUI. It should be provided at:

```
/opt/pod-manager/wildfly/discovery/bzImage
```

Refer Intel® Rack Scale Design PSME User Guide for LUI building process.

General configuration of discovery is located at /etc/pod-manager/discovery.json

```
{
  "MaxBladesCountPerDrawerBeingDeepDiscovered": 1,
  "DeepDiscoveryEnabled": true,
  "DiscoveryIntervalSeconds": 600
}
```

11.4.1 Deep discovery configuration notes

- It is possible to enable or disable deep discovery. If disabled, all data would be read from PSME
- It is possible to configure the number of blades per drawer that could be deep discovered at the same time. It could be set to lower value to prevent overcurrent and power spikes. Setting this property to higher value will yield an overall shorter time needed to perform deep discovery on all blades.
- It is possible to trigger deep discovery process manually. A manual trigger will be queued along with automatically triggered deep discovery event.
- The “MaxComputerSystemsCountPerDrawerBeingDeepDiscovered” configuration property defines threshold value per Drawer for both manual and automatic deep discovery events.

§

12 Link Aggregation Group (LAG)

Link aggregation group is a technique used in a high-speed-backbone network to enable the fast and inexpensive transmission of bulk data. The best feature of link aggregation is its ability to enhance or increase network capacity while maintaining a fast transmission speed and not changing any hardware devices, thus reducing cost.

12.1 Creating LAG

A LAG can be created by combining at least one physical uplink port resulting in an additional virtual port. To create new LAG using POD Manager REST API it is necessary to create and supply proper JSON template to POD Manager by performing a HTTP POST request on Ethernet Switch Port Collection resource located at:

```
/redfish/v1/EthernetSwitches/{switchID}/Ports/{portID}
```

Sample JSON template:

```
{
    "PortId" : "LagPort",
    "PortMode" : "LinkAggregationStatic",
    "Links" : {
        "PortMembers" : [{
                "@odata.id" : "/redfish/v1/EthernetSwitches/1/Ports/10"
            }, {
                "@odata.id" : "/redfish/v1/EthernetSwitches/1/Ports/11"
            }
        ]
    }
}
```

Table 5 Link aggregation group creation elements

Key	JSON type	Allowed values	Required	Notes, limitations
PortId	String		Yes	Switch port unique identifier. CAUTION! The value has to be maximum 16 characters long and must not contain whitespaces.
PortMode	String	"LinkAggregationStatic", "LinkAggregationDynamic"	Yes	Port working mode. Currently only LinkAggregationStatic mode is supported.
PortMembers	Array[Link]		Yes	Array of ports being member of LAG. Must be placed in Links object. There must be at least one port. All ports contained in this array must have: - "PortClass": "Physical", - "PortType": "Upstream", - The same speed. None of these ports can be a member of another LAG. Empty array [ ] will be interpreted as absence of this requirement key.

Response after LAG has been successfully created should contain location to newly created Switch Port:

```
HTTP/1.1 201 Created
Location: <PROTOCOL>://<IP>:<PORT>/redfish/v1/EthernetSwitches/1/Ports/99
```

12.2 Modifying LAG

To modify LAG using POD Manager REST API it is necessary to create and supply proper JSON template to POD Manager by performing a HTTP PATCH request on existing Switch Port resource located at following URI on POD Manager Service:

```
/redfish/v1/EthernetSwitches/{switchID}/Ports/{portID}
```

Mentioned Switch Port must be a proper LAG:

- "PortClass" must be set to "Logical".
- "PortMode" must be set to "LinkAggregationStatic".
- "PortMembers" array cannot be empty.

In following example, port 12 is added as a member to the LAG along with changing additional properties.

Sample JSON template:

```
{
    "AdministrativeState" : "Up",
    "LinkSpeedMbps" : 40000,
    "FrameSize" : 1500,
    "Autosense" : false,
    "Links" : {
        "PrimaryVLAN" : {
            "@odata.id" : "/redfish/v1/EthernetSwitches/1/Ports/99/VLANs/1"
        },
        "PortMembers" : [{
                "@odata.id" : "/redfish/v1/EthernetSwitches/1/Ports/10"
            }, {
                "@odata.id" : "/redfish/v1/EthernetSwitches/1/Ports/11"
            }, {
                "@odata.id" : "/redfish/v1/EthernetSwitches/1/Ports/12"
            }
        ]
    }
}
```

Table 6 Link aggregation group modification elements

Key	JSON type	Allowed values	Required	Notes, limitations
AdministrativeState	String	"Up", "Down"	No	Port link state forced by user.
LinkSpeedMbps	Number	Nonnegative number	No	Port speed.
FrameSize	Number	Nonnegative number	No	MAC frame size in bytes.
Autosense	Boolean	true, false	No	Indicates if the speed and duplex is automatically configured by the NIC
PrimaryVLAN	Link		No	Link to VLAN available on this port (only these VLANs are acceptable).
PortMembers	Array[Link]		No	Array of ports being member of LAG. Must be placed in Links object. All ports contained in this array must have: "PortClass": "Physical", "PortType": "Upstream", the same speed. None of these ports can be a member of another LAG.

Key	JSON type	Allowed values	Required	Notes, limitations
				If PortMembers array is not present in PATCH request, list of port members will not be changed. Otherwise, there must be at least one port.

All of above properties are optional. If not provided, they will not be changed.

Response after success LAG modification:

```
HTTP/1.1 204 No Content
```

12.3 Removing LAG

To remove LAG using PODM REST API it is necessary to perform a HTTP DELETE request on existing Switch Port resource located on POD Manager Service at:

```
/redfish/v1/EthernetSwitches/{switchID}/Ports/{portID}
```

12.4 Limitations

- LAGs can be created only from non-LAG ports.
- LAGs can be created only on upstream physical ports that have matching speeds.
- Creating a LAG on selected ports removes those ports from VLAN memberships.

§

13 Verifying OrientDB and WildFly Service Status

13.1 Verifying OrientDB status

Check whether DB is running:

```
sudo service orientdb status
```

If it is running, the above command should return following message:

```
* OrientDB Database is running with pid <PID>
```

If it is not running, start the service:

```
sudo service orientdb start
```

To access the OrientDB web interface, visit the following link:

```
http://<target_machine_ip>:2480
```

13.2 Verifying WildFly status

Check whether the WildFly service is running:

```
sudo service wildfly status
```

If it is running, the above command should return following message:

```
WildFly application Server is running with pid <PID>
```

If it is not running, start the service:

```
sudo service wildfly start
```

§

14 Events

14.1 Overview

Pod Manager is able to subscribe to events and receive them from Redfish compliant external services when specific resource is removed, added or changed.

14.2 Configuration

Configuration for events is located at /etc/pod-manager/events.json

```
{
    "EventSubscriptionIntervalSeconds" : 90,
    "NetworkInterfaceNameForEventsFromPsme" : "eth0.4094",
    "NetworkInterfaceNameForEventsFromRmm" : "eth0.4094",
    "NetworkInterfaceNameForEventsFromRss" : "eth0.4093",
    "PodManagerIpAddressForEventsFromPsme": "127.0.0.1",
    "PodManagerIpAddressForEventsFromRss": "127.0.0.1",
    "PodManagerIpAddressForEventsFromRmm": "127.0.0.1"
}
```

- EventSubscriptionIntervalSeconds – how often PODM checks if it is subscribed to service. In case of subscription absence, PODM subscribes to this service
- NetworkInterfaceNameForEventsFrom* – name of network interface used to communicate with specified service. It is used to determine PODM IP address for event subscription in PSME/Storage Service/RMM services. When PodManagerIpAddressForEvents* is present, this parameter is ignored
- PodManagerIpAddressForEventsFrom* – PODM IP address used for event subscription in PSME/Storage Service/RMM service. This parameter is optional

14.3 Impact on composed nodes status

As a result of event handling, Composed Node State may be set to “Offline” and Health to “Critical” when:

- Associated Computer System’s or RemoteTarget’s State is no longer “Enabled”
- Associated Computer System or RemoteTarget is removed

14.4 POD Manager service root UUID configuration

The PODM Northbound API service root UUID is stored in the */var/lib/pod-manager/service-root-uuid.json* configuration file. PODM will generate a UUID and store it at this location by default. The user can change or set a specific UUID of PODM service root by editing this file.

Note: This file must contain a proper UUID of PODM service root for PODM to function as expected. Sample UUID format:

```
{
  "UUID": "00000000-0000-0000-0000-000000000000"
}
```

§

15 POD Manager Database Cleanup

Pod Manager retains its asset inventory upon restart. This affects Pod Manager Startup, Deep Discovery, and functionality related to allocation and performing actions on resources.

In case avoid issues in above scenarios, it is recommended to clear Pod Manager Database upon Pod Manager Startup using following script:

```
sudo /usr/bin/pod-manager-clean-database-on-next-startup
```

CAUTION! After executing PODM database clean script, PODM would lose knowledge of all composed nodes. Workloads running on composed nodes would be killed and PODM would perform deep discovery on all resources within the rack.

§

Appendix A

Reserved VLANs

There is a possibility to restrict usage of some vlans by changing configuration file located in /etc/pod-manager/allocation.json

Example file looks like:

```
{
  "ReservedVlanIds": [1, 170, 4088, 4091, 4094]
}
```

Where 1, 170, 4088, 4091, 4094 are VLANs which are reserved. Reserved VLANs have following implications:

- Allocation JSON could not contain such VLANs and such requests results in an error
- Reserved VLANs are not deleted during allocation
- Reserved VLANs are not deleted during disassembly

Protecting storage service on POD Manager

Storage Service discovered via Pod Manager may be a Composed Node allocated and assembled via Pod Manager (this is a standard use case) but it may also be a running workload inside a managed rack that is left powered on by administrator to achieve storage sustainability (this is a corner case). While fully handling standard use case the Pod Manager handles this corner case partially by additionally protecting a Computer System that was identified as a host for such Storage Service.

In case a Storage Service is already up and running in a rack when Pod Manager starts, an attempt is made to identify the Computer System that hosts Storage Service based on the MAC address of the discovered Storage Service. If Pod Manager encounters a Computer System with a matching MAC address, the Computer System is internally marked as storage service host and marked as allocated.

Marking the Computer System allocated makes it unavailable for allocation and for automatic/manual deep discovery triggering.

MAC address of Storage Service is obtained from DHCP hook scripts (writing to /tmp/leases file). Pod Manager keeps a list of MAC addresses associated with Storage Services in its StorageGuard mechanism and performs the MAC address matching on every discovered Computer System.

If this operation succeeds, you should see in log file /var/log/pod-manager/pod-manager-application.log an INFO level entry saying:

```
"Computer system […] is now marked as storage service host"
```

§