OHMS

1 Introduction

Open Hardware Management Services (OHMS) is a stateless software agent responsible for managing individual hardware elements spread across single or multiple physical racks. The hardware elements which OHMS manages are servers, switches, and in the future disaggregated rack architectures.

OHMS can manage OEM servers and switches by publishing a set of interfaces for OEMs to integrate with it. The integration point is the hardware-management API layer (shown in Figure 1 below) using a vendor specific plugin implementation for this layer. At the same time, the plugins can interface to the hardware using the interface supported by the underlying hardware device.

The entire source code has been written in Java 1.7 and would therefore need a corresponding java-virtual-machine to run the binaries.

2 Glossary

This document uses the following terms/acronyms:

Term	***Definition***
API	Application Programming Interface
CIM	Common Information Model
ESXi	Hypervisor developed by VMware
FRU	Field Replaceable Unit
HTTP	Hyper Text Transfer Protocol
HW	Hardware
IB	In Band
IPMI	Intelligent Platform Management Interface
JVM	Java Virtual Machine
NB	Northbound
OEM	Original Equipment Manufacturer
OHMS	Open Hardware Management Services
OOB	Out Of Band
REST	Representational State Transfer
RSD	Rack Scale Design
SB	Southbound
STS	Spring Tool Suite
VM	Virtual Machine

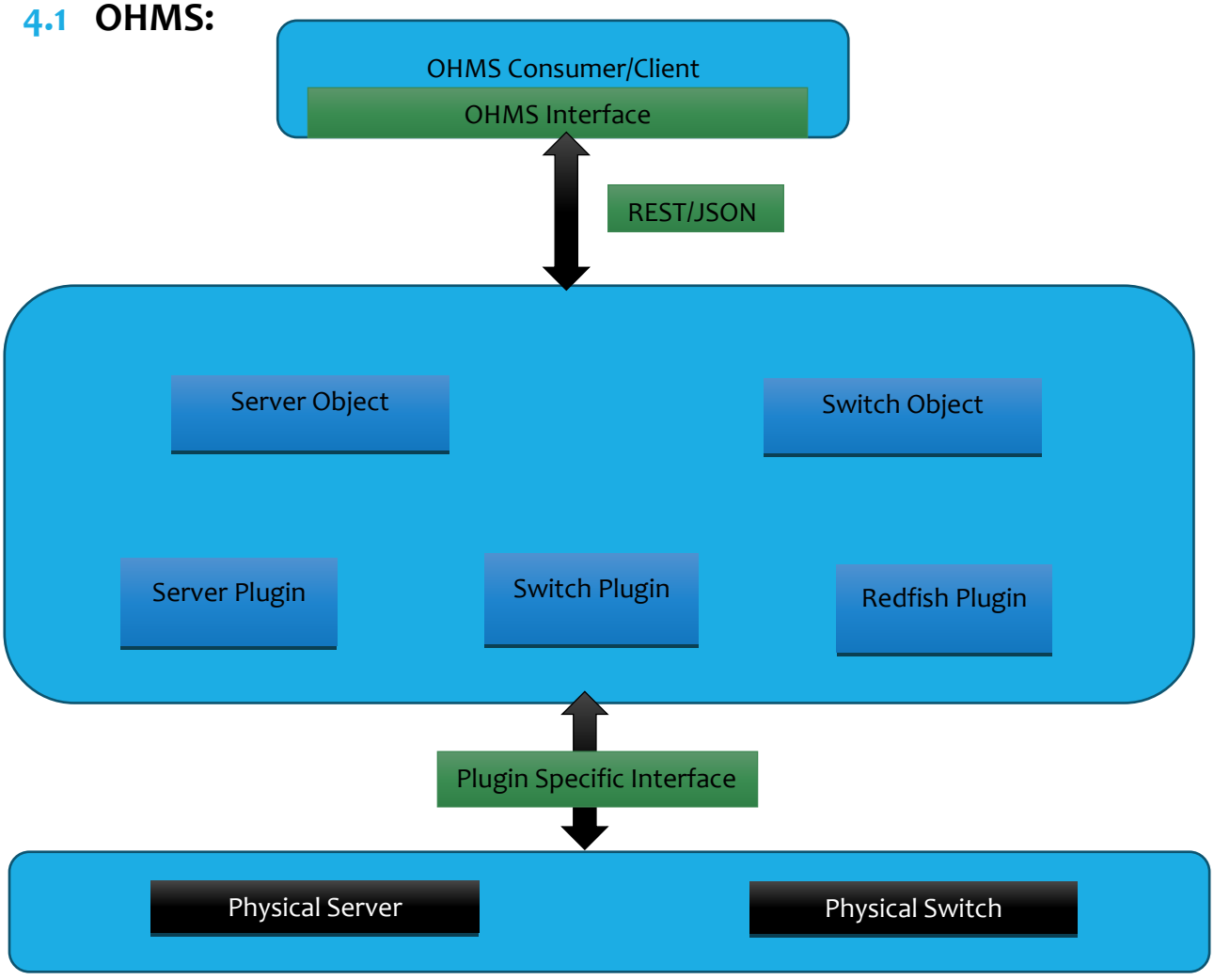
3 Use of Plugins in OHMS

Plugins are software modules that allow for the expansion and customization of the base OHMS software. For the OHMS, users may develop plugins to customize OHMS usage to support servers and switches of their choice. On the northbound side, plugins would comply to the OHMS API and on the southbound side, interface with the specific custom server or switch in question.

The OHMS code available as open source has a sample server plugin as well as a sample switch plugin. Quanta server and Cumulus switch plugins are the example plugins for a server and a switch that are provided as a part of OHMS in the Git repository.

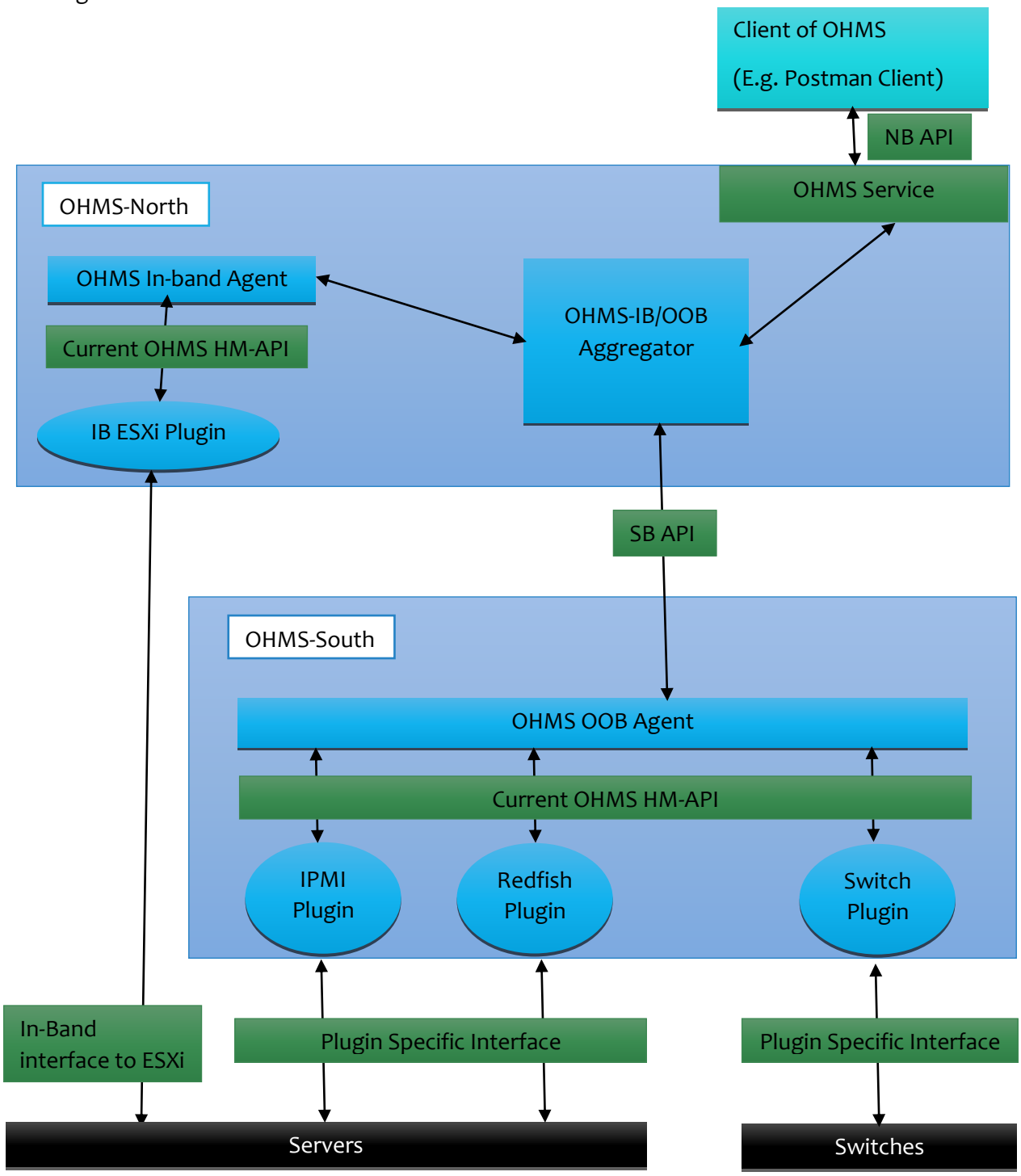
4 Architecture

4.1 OHMS:



OHMS provides a set of generic service APIs for consumption by a client (It can be even a REST client such as Postman). These APIs act on the underlying physical objects, which are encapsulated in a set of software objects viz. server, switch and storage. OHMS internally maintains a hardware management layer to service the API requests. The hardware management layer in turn depends upon vendor/hardware specific plugins to interface to the actual hardware. The two types of plugins used to interface to the actual hardware are in-band (IB) and out-of-band (OOB) plugin used by IB agent and OOB agent respectively.

Another way to look at OHMS would be that internally, OHMS consists of two modules, hms-aggregator and hms-core. Hms-aggregator encompasses the IB agent. hms-core is referred to as OOB agent.



4.2 OHMS-aggregator

Key responsibilities of OHMS aggregator are:

1) Orchestrate request to OOB or IB Agent
   This is one of the key advantages, since the aggregator smoothens out the IB/OOB access and thus the client need not care about the access.
2) Monitor Rack hardware inventory.
3) Cache Rack inventory.
4) Manage/Load Inband Plugin.

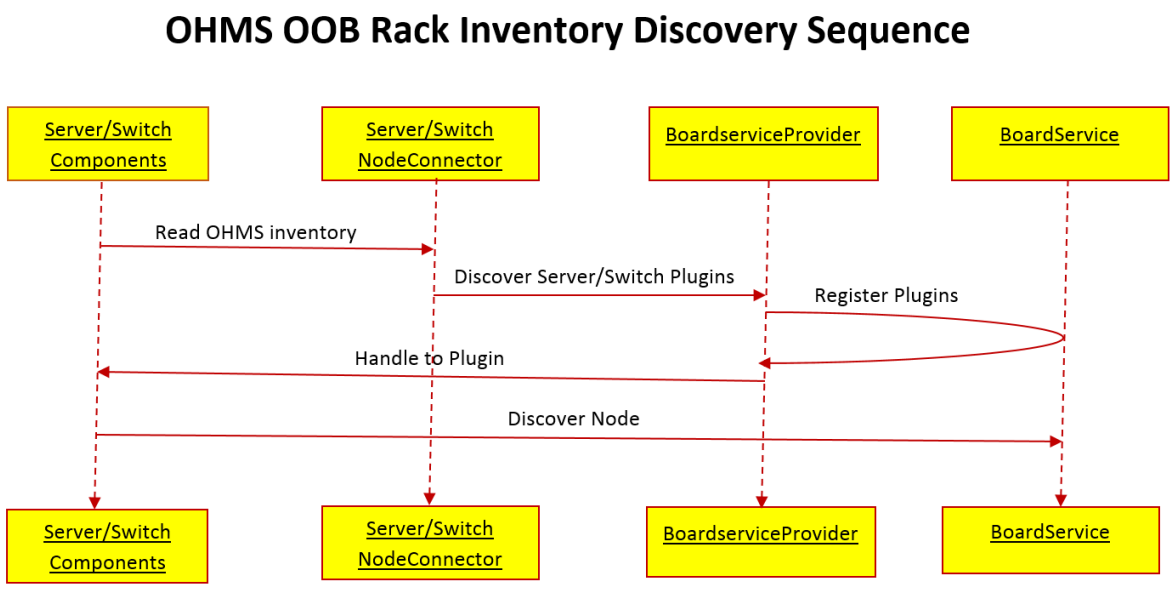
4.3 OHMS OOB Agent (aka: HMS-core)

It is also referred as OOB agent. Key responsibilities of OHMS OOB agent are:

1) Provides Hardware Abstraction
2) Define Models to be shared with South Bound partner developed plugins.
3) Define Networking Models for Switches.
4) Manage Partner OOB Plugins.
5) Interface with HMS Aggregator over HTTP via REST.
6) Monitor Hardware Sensor states.
7) Discover Rack Inventory.
8) Perform power operations on Servers/Switches

5 Functional Description (Workflows)

5.1 OHMS OOB Rack Inventory Discovery



ServerNodeConnector: A class which holds all the server nodes in a map with serverId as key.

SwitchNodeConnector: A class which holds all the switch nodes in a map with switchId as key.

BoardServiceProvider: A class which holds .class object of a plugin for each server provided in the inventory file. It holds it in a map with corresponding serverId as key.

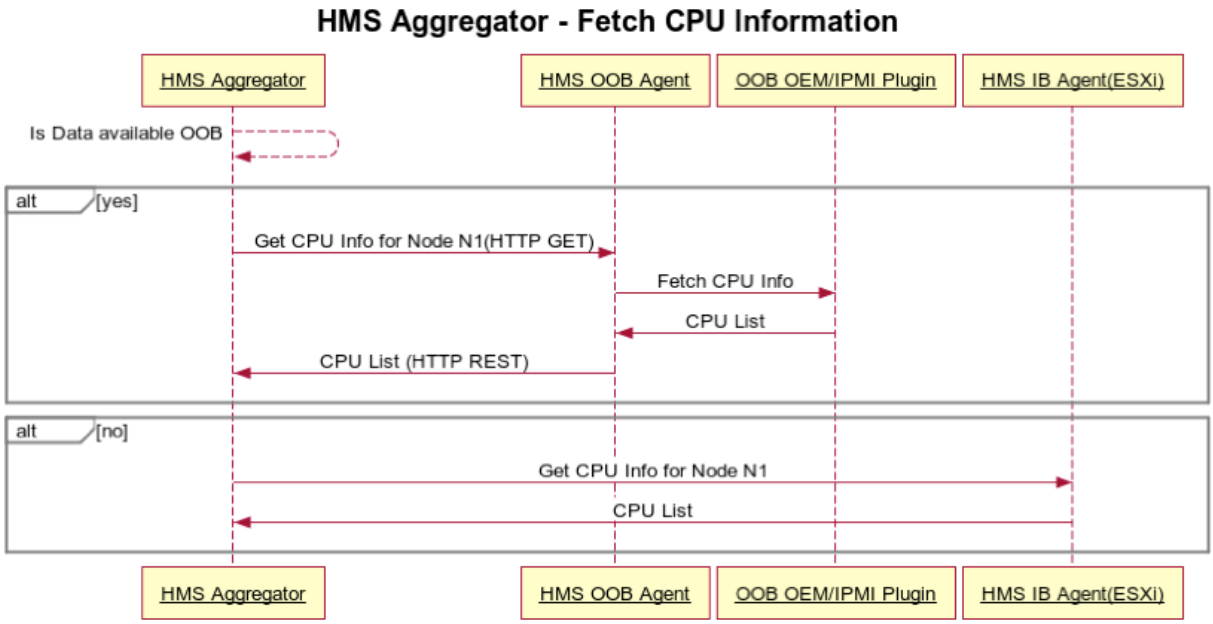
Boardservice: It is an instance of a plugin. With the help of these instances, OHMS talks to corresponding H/W. BoardServiceProvider creates a new instance for each API request with the help of corresponding .class object stored in the map with key as serverId.

NOTE: In IB, we cache the Boardservice itself for each server in a map with key as serverId.

On OHMS boot up, Rack discovery sequence is initiated:

1. Get the rack inventory provided through hms-inventory.json file. Here "rack" indicates scope for the HMS aggregator, IB and OOB agents and not necessarily a single physical rack. It can be a single host or hosts spread across multiple racks.
2. Initiate discovery for servers and switches with the help of Server/Switch NodeConnector. This involves following:
    - 2.1 A plugin for a server is registered by BoardServiceProvider after matching manufacturer and model values present in the hms-inventory.json file as well as in plugins (provided in the form of binaries/jar files).
    - 2.2 Then BoardServiceProvider provides an instance of the plugin to discover a node (Server/Switch).
    - 2.3 Discover the node using the plugin instance.
    - 2.4 Update OHMS models with node discovery status.

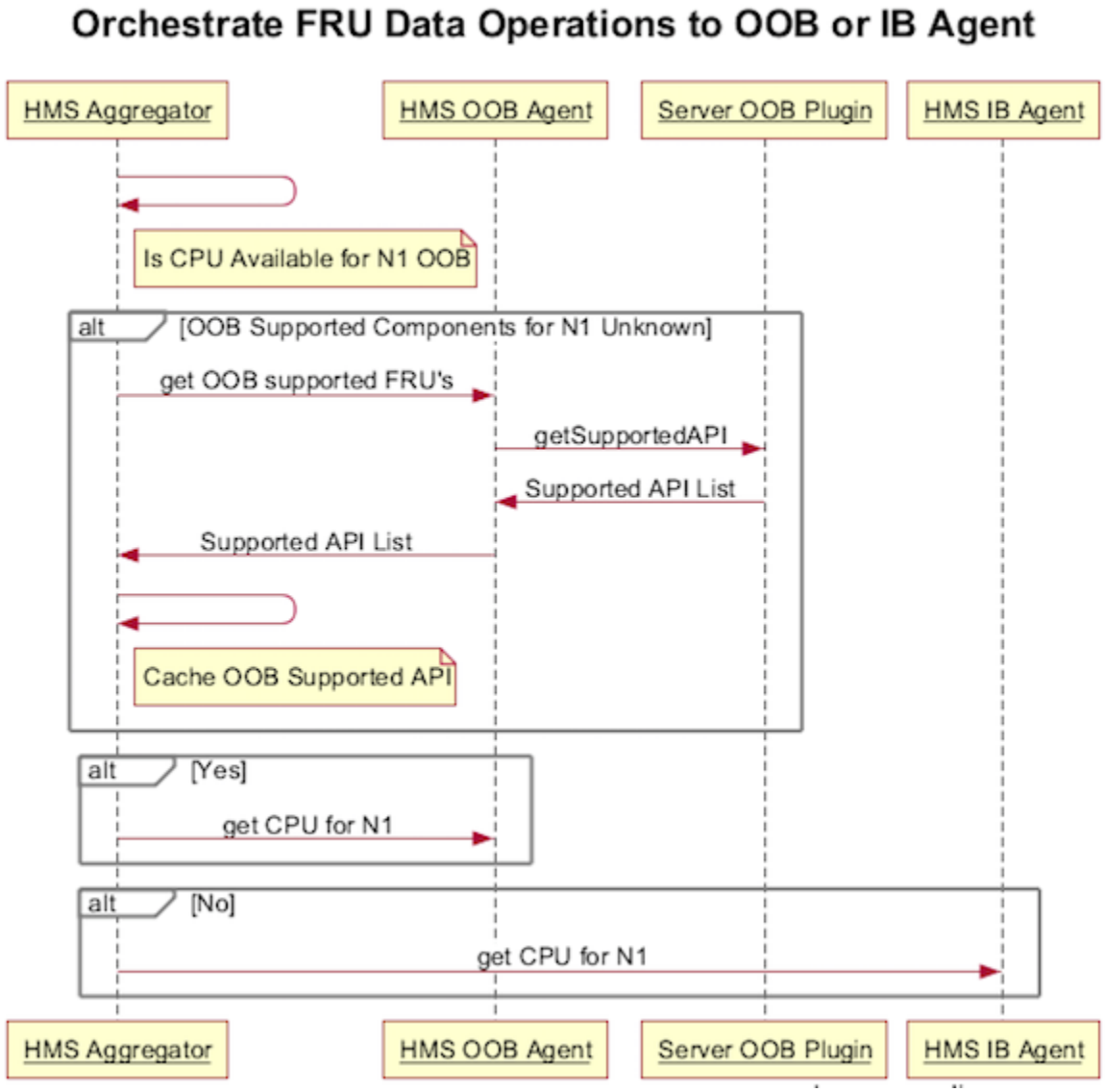
5.2 Fetch FRU information



Sequence to fetch CPU (an FRU) information from hms-aggregator:

1) HMS aggregator checks if Data is available OOB or IB
2) If CPU data is available OOB, get from OOB agent.
3) If CPU data is available IB, get from IB agent.
4) If CPU data is available for both OOB and IB, get from OOB agent.

5.3 HMS Aggregator Orchestrator

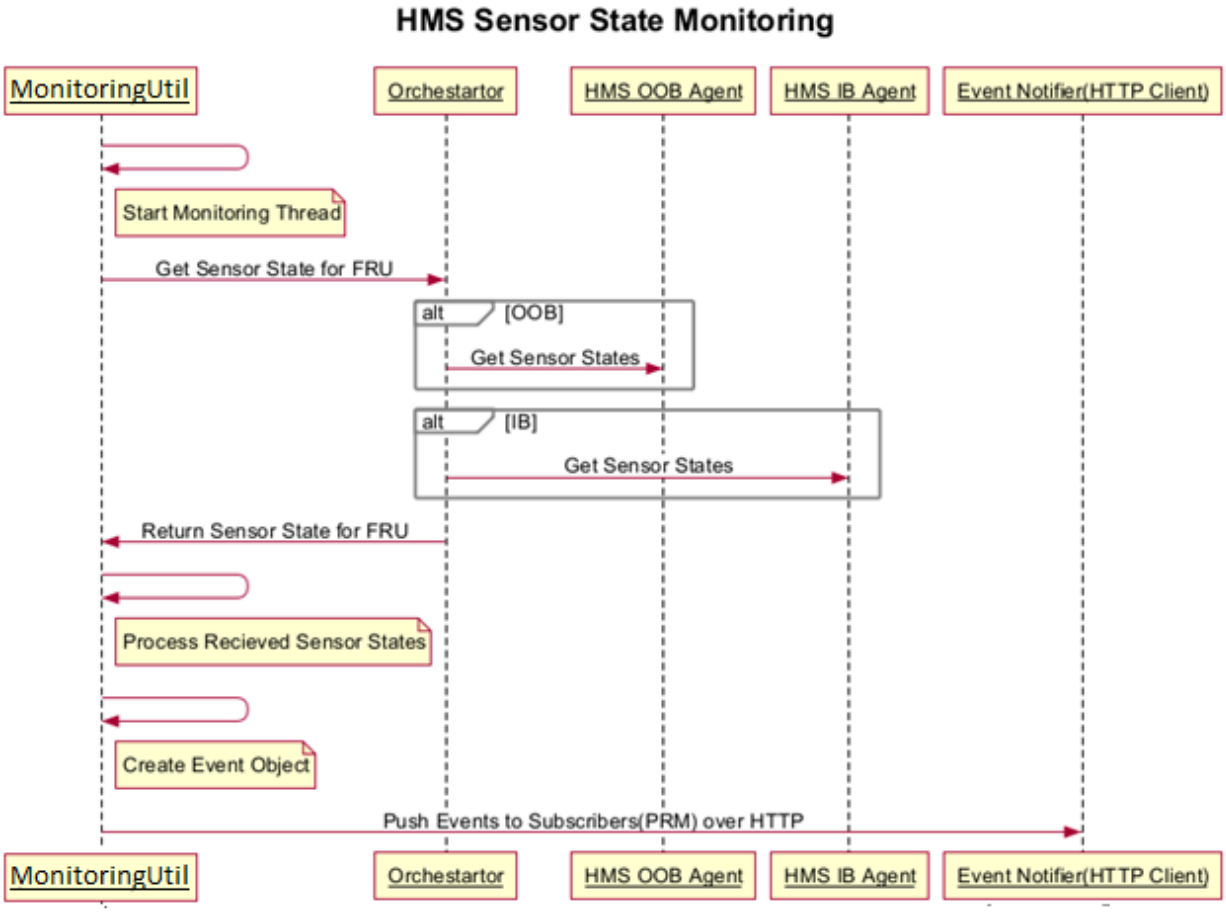


Orchestrator sequence for fetching FRU data from OOB agent or IB agent:

1) Check in cache if FRU information is available with OOB
2) If cache is not having any data, start sequence to cache this information.
3) HTTP request to OOB agent to get OOB supported FRUs
4) OOB fetches the FRUs that are supported by the plugin that has been implemented.
5) Orchestrator caches the same information for future use.
6) If FRU is supported by OOB, FRU information is retrieved from OOB agent over HTTP
7) If FRU is not supported by OOB, the same is fetched from IB agent.

In case the OOB and IB agents support FRU information then the OOB agent would be used.

5.4 OHMS hardware monitoring



MonitoringUtil: A class (present in hms-aggregator module) responsible for monitoring the inventory.

Orchestrator: HMS- aggregator plays the role of an orchestrator as evident from section 5.3 above.

1) HMS Aggregator Initiate Monitoring Sequence
2) Every 10 minutes a node and its FRU sensor are monitored (10 minutes is default but configurable)
3) On receiving request to get sensor state Orchestrator validates if FRU sensor are available OOB or IB
4) If OOB Sensor states are fetched from HMS OOB Agent over HTTP else In Band Agent is used to fetch sensor data.
5) Received sensor states are processed (compares threshold, sensor state change since last fetch and classify into critical, error, warning etc.)
6) Corresponding Event Objects are created that are understood by upper layer (in this case client) and then pushed over HTTP.

6 Build and Start OHMS

6.1 Building OHMS

1) On Windows, install the following software components viz. git bash, Spring Tool Suite (STS), JAVA 7, apache-maven and set up the environment variables as required by each of these components.
2) Open git bash terminal and run the command: git clone <repository-url>
   For E.g. git clone https://github.com/vmware/OHMS.git MyOHMS

   This will download the whole repository in a directory named MyOHMS.
3) Change the current directory to the newly created "MyOHMS".
4) Check if the bash shell is showing the current branch as "master".
5) Run the command: git checkout development
   This will change the current branch to development.
6) Now to compile the codebase, run the command: mvn clean compile
   (Assuming maven has been set up by adding proper environment variables)
7) One can also directly go ahead and build the codebase by running the command: mvn clean install
8) Once it has been done successfully, open STS and import the project as an existing maven project.

6.2 Running OHMS (From Developer point of view)

1) Update the hms-inventory.json file (present in modules/hms-core/config) with the proper host details that one is going to use.
   It basically involves changing oobIpAddress, oobUsername, oobPassword, ibIpAddress, ibUsername, ibPassword, boardInfo:manufacturer and boardInfo:model.
2) Update hms.properties
   This is slightly tricky since the codebase tries to find this file at two locations.

   Case I: When hms.properties file is present in {user.home}/VMware/vRack/hms-config directory. Open it and change the value of hms.ib.inventory.location to any pathname you want.

   Case II: When hms.properties file isn't present in the above mentioned location. Then go to modules/hms-aggregator/src/main/resources/. Open hms.properties present there and change the value of hms.ib.inventory.location to any pathname you want. The pathname will be used to look for the inventory file by hms-aggregator and if not found then will create the same at that location. Now, go to git bash shell and traverse to modules/hms-aggregator and run the command: mvn clean install

3) Now go back to STS, do a refresh on the project and run the file HmsApp.java (present in hms-core/src/main/java) as a java application. If running successfully then open any browser or REST client (such as postman) and invoke the URL: http://127.0.0.1:8448/api/1.0/hms/host/
It should display all the servers present in the updated hms-inventory.json file

4) Finally, run hms-aggregator as Run on server. If running successfully then invoke the URL: http://127.0.0.1:8080/hms-aggregator/api/1.0/hms/host
It should again display all the servers present in the hms_ib_inventory.json file

Congrats, you have OHMS codebase up and running!

7 OOB agent REST Endpoints

These are termed SB (Southbound) APIs. The aggregator talks to OOB agent and gets the OOB data by invoking these APIs.

7.1 GET requests

http://{OOB_Agent_ip}:8448/api/1.0/hms/nodes/

http://{ OOB_Agent_ip}:8448/api/1.0/hms/about

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}/

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}/powerstatus

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}/supportedAPI

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}/bootoptions/

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}/bmcusers/

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}/cpuinfo/

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}/memoryinfo

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}/storageinfo

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}/storagecontrollerinfo

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}/nicinfo/

http://{ OOB_Agent_ip}:8448/api/1.0/hms/event/host/{host_id}/CPU/

http://{ OOB_Agent_ip}:8448/api/1.0/hms/event/host/{host_id}/MEMORY/

http://{ OOB_Agent_ip}:8448/api/1.0/hms/event/host/{host_id}/STORAGE/

http://{ OOB_Agent_ip}:8448/api/1.0/hms/event/host/{host_id}/STORAGE_CONTROLLER/

http://{ OOB_Agent_ip}:8448/api/1.0/hms/event/host/{host_id}/SYSTEM/

http://{ OOB_Agent_ip}:8448/api/1.0/hms/event/host/{host_id}/NIC

http://{ OOB_Agent_ip}:8448/api/1.0/hms/event/host/HMS/

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/ports

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/portsbulk

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/ports/{port_id}

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/lacpgroups

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/vlans

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/vlansbulk

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/vlans/{vlan_name}

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/vxlans

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/vlans/{vlan_name}/vxlans

http://{ OOB_Agent_ip}:8448/api/1.0/hms/event/switches/{switch_id}/SWITCH

http://{ OOB_Agent_ip}:8448/api/1.0/hms/event/switches/{switch_id}/SWITCH_PORT

7.2 PUT requests

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}?action=power_up

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}?action=power_down

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}?action=power_cycle

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}?action=hard_reset

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}?action=cold_reset

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}/chassisidentify/

E.g. application/json

{“identify”: true, “interval”: 15}

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}/bootoptions/

E.g. application/json

{

“bootFlagsValid”: true,

“bootOptionsValidity”: “Persistent”,

“biosBootType”: “Legacy”,

“bootDeviceType”: “External”,

“bootDeviceSelector”: “PXE”,

“bootDeviceInstanceNumber”: 2

}

http://{ OOB_Agent_ip}:8448/api/1.0/hms/host/{host_id}/selinfo/

E.g. application/json

{“direction” : “RecentEntries”, “recordCount” : 5, “selTask” : “SelDetails”}

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/lacpgroups

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/vlans

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/vlans/{vlan_name}

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/vlans/{vlan_name}/vxlans

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/ports/{port_id}

http://{ OOB_Agent_ip}:8448/api/1.0/hms/switches/{switch_id}/reboot

Here OOB_Agent_ip can be replaced by the IP of the machine on which OOB agent is running.

8 Aggregator REST Endpoints

These are termed NB APIs. Any client of OHMS can call these APIs and get the H/W related data, both IB and OOB.

8.1 GET requests

http://{Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/about/

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/nodes/

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}/powerstatus/

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}/cpuinfo/

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}/memoryinfo

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}/storageinfo

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}/storagecontrollerinfo

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}/nicinfo

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}/selftest

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/{host_id}/bmcusers

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/{host_id}/bootoptions

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/event/host/{host_id}/CPU

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/event/host/{host_id}/MEMORY

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/event/host/{host_id}/STORAGE

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/event/host/{host_id}/STORAGE_CONTROLLER

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/event/host/{host_id}/SYSTEM

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/event/host/{host_id}/NIC

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/event/host/HMS/

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/switches

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/switches/{switch_id}

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/switches/{switch_id}/ports

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/switches/{switch_id}/portsbulk

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/switches/{switch_id}/ports/{port_id}

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/switches/{switch_id}/lacpgroups

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/switches/{switch_id}/vlans

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/switches/{switch_id}/vlansbulk

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/switches/{switch_id}/vlans/{vlan_name}

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/switches/{switch_id}/vxlans

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/switches/{switch_id}/vlans/{vlan_name}/vxlans

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/event/switches/{switch_id}/SWITCH

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/event/switches/{switch_id}/SWITCH_PORT

8.2 PUT requests

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}/bootoptions/

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}/chassisidentify

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}/selinfo

Request Body: { “direction” : “RecentEntries”, “recordCount” : 5, “selTask” : “SelDetails”}

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}?action=power_down

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}?action=power_up

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}?action=power_cycle

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}?action=hard-reset

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/host/{host_id}?action=cold_reset

http://{ Inband_Agent_ip}:8080/hms-aggregator/api/1.0/hms/switches/{switch_id}

Here Inband_Agent_ip can be replaced with the IP of the machine on which hms-aggregator is running.

NOTE: For finding the corresponding request mapping in the codebase, one can explore the packages services and controller in hms-core and hms-aggregator respectively.