HCLTech

REST API Guide for Virtualized Infrastructure Manager, Version 6.0

First Published Date: November 18, 2025

HCLTech

Contents

HCLTech

VIM REST API Guide

The following sections explain about how VIM REST API is used to manage NFVI.

HCLTech

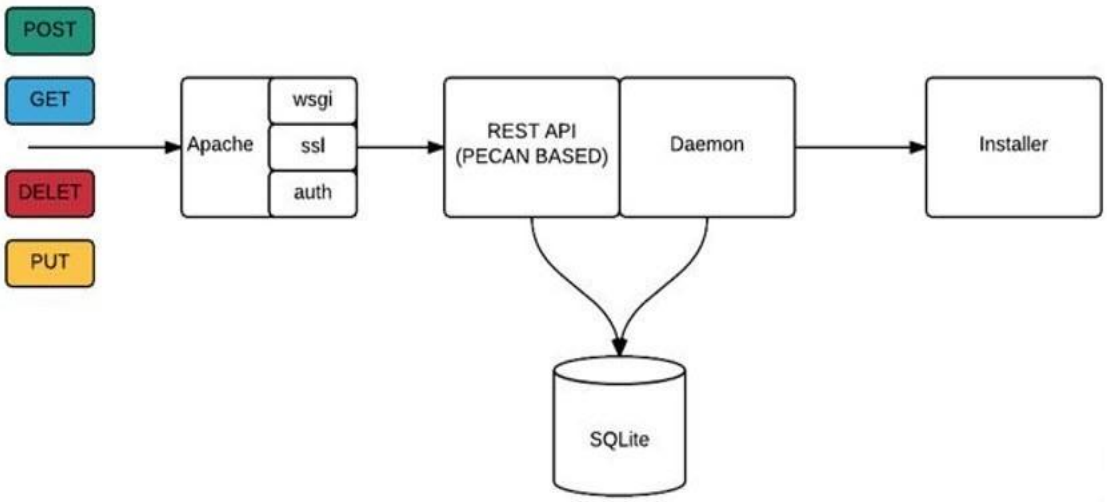
REST API Overview

A Representational State Transfer (REST) API is used to install, expand, and update VIM. Actions performed using the REST APIs are:

- Install VIM on NFVI pods.
- Add and delete pods to and from NFVI during installation.
- Update VIM software.
- Replace controller nodes.
- Perform cloud maintenance operations.

Run cloud validations using Virtual Machine Throughput (VMTP). VMTP is a data path performance measurement tool for OpenStack clouds.

The following figure shows the workflow of VIM REST API.



The VIM REST API security is provided by the Secure Sockets Layer (SSL) on the Apache web server. The *mod_wsgi* running on the Rest API server calls the Pecan-based web application. The Pecan REST API server requires a username and password to authorize the REST API server requests.

Apache handles the authorization of the request to access the Pecan web application. Use the VIM A PI to:

Upload a new setup_data.yaml file to start, stop, or query the state of the installation.

Manage the cloud.

Add/remove compute and Ceph nodes and replace the controller nodes.

Launch VMTP (L2/L3 data plane testing) and CloudPulse.

The VIM REST API is enabled by default in the management node if you are using the supplied VIM buildnode.iso. You can access API server on the br_api interface on port 8445. The authentication is enabled by default on the web service.

You can access the API end points using the following URL format:

```
   https://<Management_node_api_ip>:8445
```

By default, the basic authentication is enabled for the API endpoints in the management node. You can find the authentication credentials in the following file in the management node:

/opt/cisco/ui_config.json

The following code shows a sample ui_config.json file.

```
   {
   "Kibana-Url": "http://10.10.10.10:5601", "RestAPI-Url": "https://
10.10.10.10:8445",
   "RestAPI-Username": "admin",
   "RestAPI-Password": "a96e86ccb28d92ceb1df", "RestDB-Password":
"e32de2263336446e0f57", "BuildNodeIP": "10.10.10
```

HCLTech

VIM REST API Using Curl for IPv6

Prerequisites

1. You need to copy the certificates from the management node to local machine from where you would launch the APIs.
2. Create a folder in local machine and copy the certificates:

```
# mkdir ~/certificates
```

3. Copy REST API CA Certificates (for mercury commands)

```
# scp root@<Management Node>:/var/www/mercury/mercury-ca.crt ~/certificates
```

⚠ The key information that you need are br_api and cloud_api (external_lb_vip_ipv6_address).

4. For each POD, get the REST API credentials:

```
# cat /opt/cisco/ui_config.json
{
"Kibana-Url": "https://[2001:420:293:2433:172:29:85:110]:5601",
"RestAPI-Username": "admin",
"RestAPI-Password": "c9686a8f3ca46644ae95",
"RestAPI-Url":
"https://[2001:420:293:2433:172:29:85:110]:8445", "BuildNodeIP":
"2001:420:293:2433:172:29:85:110". ----> br_api
}
```

Offline Validation using Curl

1. Create offline validation test.

Request

Curl Command:

curl -g -i -X POST -H "Content-Type: application/json" -u admin:<Password> -H "Accept: application/json" --cacert 'C:\<mercury-ca.crt payh>' -d "{\"jsondata\": $(cat setup_data.json)}" https://<Pod IP>:8445/v1/offlinevalidation

Response

```
{"status": "NotValidated", "uuid": "2b8253f4-ad9f-4fbf-b224-
a65bd210392a", "created_at":
"2019-02-28T18:02:36.808740+00:00", "updated_at": null,
"jsondata": "{}"}
```

2. Get the offline validation test result:

Request

```
{"status": "ValidationFailed", "uuid": "2b8253f4-ad9f-4fbf-
b224-a65bd210392a",
"created_at": "2019-02-28T18:02:36+00:00", "updated_at": "2019-
02-28T18:02:57+00:00",
"jsondata": ""}
```

Curl Command:

curl -g -i -X GET -H 'Content-Type: application/json' -u admin:<Password> -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert 'C:<mercury-ca.crt path>' https://<Pod IP>:8445/v1/offlinevalidation/<UUID

Response

HCLTech

Start New Installation

1. Create new setup data before starting new installation, for example

```
curl -g -i -X POST -H 'Content-Type: application/json' -u
admin:46d13357ef15e5482b52 -H 'Accept: application/json' -H
'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt -d '{u'meta': {}, u'name':
u'NEWSETUPDATA', u'jsondata': {<SetupData in JSON Format>}}'
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/setupdata
```

2. To start the installation:

Request:

```
Curl -g -i -X POST -H 'Content-Type: application/json'
admin:46d13357ef15e5482b52 -H 'Accept: application/json' -H
'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt -d '{u'stages': u'vmtp',
u'setupdata': u'8b0d4a46-c67f-4121-99af- 32fde52a82eb'}'
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/install
```

Response:

```
{u'uuid': u'6b02c2ab-441e-471a-9dcc-e771136186e1', u'setupdata':
u'8b0d4a46-c67f-4121-99af-32fde52a82eb', u'vmtpresult': u'',
u'updated_at': None, u'validationstatus': u'', u'currentstatus':
u'Not Available', u'install_logs': u'', u'stages': {u'baremetal':
u'Scheduled', u'bootstrap': u'Scheduled', u'runtimevalidation':
u'Scheduled', u'ceph': u'Scheduled', u'orchestration':
u'Scheduled', u'validation': u'Scheduled', u'hostsetup':
u'Scheduled', u'vmtp': u'Scheduled'}, u'created_at':
u'2019-03-05T05:22:30.986823+00:00'}
```

3.Get active setup data with UUID after installation is started:

Request:

```
curl -g -i -X GET -H 'Content-Type: application/json' -u
admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-
ciscovimclient' --cacert
~/certificates/mercury-ca.crt
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/setupdata
```

```
{"created_at": "2019-03-04 21:35:00+00:00", "updated_at":
"2019-03-04 21:36:24+00:00",
"reboot_required": false, "update_status": false,
"current_op_logs": "", "current_op_status": "diskmgmt_completed",
"insight_monitor_status": "Success", "current_op_name":
"DiskMgmt", "current_op_monitor": ""}
```


Response

It will return to the list format. You must check the status. The status can be Active, Installation Failed,or Installing.


```
{"setupdatas": [{"status": "Active", "uuid": "c5bc5fd9-6f4b-
43e7-a61a-a9d409569943", "jsondata": " {<Setupdata JSON>}",
"meta": "{}", "name": "NEWSETUPDATA"}]}
```


4.Monitoring the installation using OP-information (current operation information):

Request:

```
curl -g -i -X GET -H 'Content-Type: application/json' -u
admin:46d13357ef15e5482b52 -H 'Accept: application/json' -H
'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/op_info
```

Response:

Check for the value of key insight_monitor_status. If it is Running, it indicates that the last operation is still in running state. Once the operation is completed, the value is either Success/Failed based on the result.


```
{u'created_at': u'2019-02-25 18:15:00+00:00', u'updated_at':
u'2019-02-25 18:15:00+00:00',
u'reboot_required': False, u'update_status': False,
u'current_op_logs':
u'https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8008/mercury/ae
3ed699-2ffe-4ae0-a8ab-83ef7fdce008',
u'current_op_status': u'Running', u'insight_monitor_status':
u'Running',
u'current_op_name': u'install_op Orchestration',
u'current_op_monitor':
```


Sample output information after successful completion is given below:

Pod Management Operations

Prerequisites

Before performing any pod management operation, you need to update the setup data using PUT method.

Update Setup Data

1. Get the active setup data UUID using the install API.

Request:

```
curl -g -i -X GET -H 'Content-Type: application/json' -u
admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-
ciscovimclient' --cacert
~/certificates/mercury-ca.crt
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/install
```

Response:

```
{u'installs': {u'uuid': u'6b02c2ab-441e-471a-9dcc-
e771136186e1', u'setupdata': u'8b0d4a46-c67f-4121-99af-
32fde52a82eb', . . .}}
```

2. Send the PUT request on the setup data UUID

```
curl -g -i -X PUT -H 'Content-Type: application/json' -u
admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-
ciscovimclient' --cacert
~/certificates/mercury-ca.crt -d '{u'meta': {}, u'name':
u'NEWSETUPDATA', u'jsondata':
{<Setupdata JSON>}}'
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/setupdata/
```

Note: Change the setup_data.yaml file into json using below steps

Step 1. Get current setupdata

Step 2. Convert setupdata string generated from step 1 into json file format

You can perform the following pod management operations:

- Add Compute
- Add Storage
- Remove Compute
- Remove Storage
- Replace Controller

Add Compute

1. List the nodes.

Request

```
curl -g -i -X GET -H 'Content-Type: application/json' -u
admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-
ciscovimclient' --cacert
~/certificates/mercury-ca.crt
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/nodes
```

Update new setupdata with node manually, follow steps given in the prerequisites to manual update the setupdata

curl -i -X PUT -H "Content-Type: application/json" -H "Accept: application/json" -H "User-Agent: python-ciscovimclient" -u admin:<rest api admin password> --cacert <mercury cert path> -d @<payload in json file format> https://<br_api of the pod>:8445/setupdata/<setupdata uuid>

Run below command to add node into the cluster

curl -i -X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "User-Agent: python-ciscovimclient" -u admin:<Rest API admin password> --cacert <mercury certificate path> -d '{"name": "<Node name>", "skip_vmtp": false}' https://<br_api>:8445/nodes/add_compute

1. POST to nodes to add entry:

Add Storage

1. Follow the steps 1 and 2 given in the add compute as mentioned above
2. Execute below command to add storage into the storage cluster

curl -i -X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "User-Agent: python-ciscovimclient" -u admin:<Rest API admin password> --cacert <mercury certificate path> -d '{"name": "<Node name>", "skip_vmtp": false}' https://<br_api>:8445/nodes/add_compute

3. POST to nodes to add entry:

Remove Compute

List the nodes:

Request:

```
curl -g -i -X GET -H 'Content-Type: application/json' -u
admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-
ciscovimclient' --cacert
~/certificates/mercury-ca.crt
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/nodes
```

Response:

```
{"nodes": [{"status": "Active", "uuid": "1929776f-8b77-4b35-b55c-
0abd6433b989",
"setupdata": "8b0d4a46-c67f-4121-99af-32fde52a82eb", "node_data":
"{\"rack_info\":
{\"rack_id\": \"RackC\"}, \"cimc_info\": {\"cimc_ip\":
\"172.29.172.81\"},
\"management_ip\": \"21.0.0.13\"}", "updated_at": "2019-03-
04T21:42:38+00:00",
"reboot_required": "No", "mtype": " block_storage", "install":
"6b02c2ab-441e-471a-9dcc-e771136186e1", "power_status":
"PowerOnSuccess", "install_logs":
"https://172.31.231.17:8008/mercury/071e79a5-b279-4628-bcf0-
df168152cc42", "created_at":
"2019-03-05T05:42:38+00:00", "name": "compute-3"}, . . . ]}
```


1. Remove the node entry in setup data and update the setup data by following the steps given under prerequisites.

2. . Send delete request on nodes, to remove the storage node for that UUID:

```
   curl -g -i -X DELETE -H 'Content-Type: application/json' -u
admin:46d13357ef15e5482b52
   -H 'Accept: application/json' -H 'User-Agent: python-
ciscovimclient' --cacert
   ~/certificates/mercury-ca.crt -d '{u'force_op': False, u'name':
u'1929776f-8b77-4b35-b55c-0abd6433b989'}'
   https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/nodes/remo
ve_compute
```

Remove Storage

1.Get the UUID of the node to be removed by getting the list of nodes.

2.Remove the node entry in setup data and update the setup data using steps mentioned in the prerequisites.

3.Send delete request on nodes, to remove the storage node for that UUID.

Request

```
   curl -g -i -X GET -H 'Content-Type: application/json' -u
admin:46d13357ef15e5482b52 -H
   'Accept: application/json' -H 'User-Agent: python-
ciscovimclient' --cacert
   ~/certificates/mercury-ca.crt
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/nodes
```

Response:

```
{"nodes"   curl -g -i -X DELETE -H 'Content-Type: application/json' -u
0ddb72bdadmin:46d13357ef15e5482b52
"node_da   -H 'Accept: application/json' -H 'User-Agent: python-
{\"rack_ciscovimclient' --cacert
\"172.29   ~/certificates/mercury-ca.crt -d '{u'force_op': False, u'name':
\"manageu'0b7b2b6e-305c-48e0-b9f3-0ddb72bd3b3f'}'
04T21:42https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/nodes/remove
"install_storage
"6b02c2ab-441e-471a-9dcc-e771136186e1", "power_status": "PowerOnSuccess",
"install_logs":
"https://172.31.231.17:8008/mercury/071e79a5-b279-4628-bcf0-
df168152cc42", "created_at": "2019-03-05T05:42:38+00:00", "name": "Store-
3"}, . . . ]}
```


Replace Controller:

1. Remove the node entry in setup data using steps mentioned in the prerequisites.

   Execute below command to update setupdata

   curl -i -X PUT -H "Content-Type: application/json" -H "Accept: application/json" -H "User-Agent: python-ciscovimclient" -u admin:<Rest api admin passwd> --cacert "<mercury cert file path>" -d @<setup data in json format> https://<br_api ip>:8445/setupdata/<setupdata uuid>

   2.Put nodes to replace entry

   3.Send delete request on nodes, to remove the storage node for that UUID.

   Request:


```
{"nodes": [{"status": "Active", "uuid": "79e43c4c-8cbd-4c81-8c22-
3aec717298e9", "setupdata": "8b0d4a46-c67f-4121-99af-
32fde52a82eb", "node_data": "{\"rack_info\":
{\"rack_id\": \"RackC\"}, \"cimc_info\": {\"cimc_ip\":
\"172.29.172.81\"},
\"management_ip\": \"21.0.0.13\"}", "updated_at": "2019-03-
04T21:42:38+00:00", "reboot_required": "No", "mtype": " control",
"install":
"6b02c2ab-441e-471a-9dcc-e771136186e1", "power_status":
"PowerOnSuccess", "install_logs":
"https://172.31.231.17:8008/mercury/071e79a5-b279-4628-bcf0-
df168152cc42", "created_at": "2019-03-05T05:42:38+00:00", "name":
"gg34-10"}, . . . ]}
```


Fetch Hardware Inventory

Request:

```
   curl -g -i -X GET -H 'Content-Type: application/json' -u
admin:46d13357ef15e5482b52 -H
   'Accept: application/json' -H 'User-Agent: python-ciscovimclient' -
-cacert
   ~/certificates/mercury-ca.crt
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/v1/hwinfo
```


- JSON
  - hwinforesult
    - c37-control-2.cisco.com
      - cimc_ip : "172.26.229.62"
      - firmware
        - serial_number : "FCH2037V3U9"
        - fw_version : "C240M4.3.0.4b.0.0610182318 "
        - fw_model : "UCSC-C240-M4S"
      - storage
        - physical_drive-1
          - status : "Online"
          - vendor : "SEAGATE"
          - interface_type : "SAS"
          - serial_number : "S402LC7Y0000E7093S5G"
          - media_type : "HDD"
          - model : "ST1200MM0088"
          - size : "1143455 MB"
        - physical_drive-2
      - intel_nw_adapters
        - adapter-2
          - num_of_interfaces : "4"
          - product_name : "Cisco(R) Ethernet Converged NIC X710-DA4"
          - mac_address
        - adapter-1
      - memory
        - total_memory : "262144"
        - available_memory : "262144"
      - cpu
        - number_of_cpus : "2"
        - number_of_threads : "48"
        - cores_per_cpu : 12
        - number_of_cores : "24"
      - power_supply
        - power_state : "on"
    - c37-control-1.cisco.com
    - c37-compute-4.cisco.com
    - c37-control-3.cisco.com
    - c37-storage-2.cisco.com
    - c37-storage-1.cisco.com
    - c37-compute-2.cisco.com
    - c37-compute-1.cisco.com
    - c37-compute-3.cisco.com
    - c37-storage-3.cisco.com
    - c37-compute-6.cisco.com
    - c37-compute-5.cisco.com

Glance Image Upload

Use Rest APIs to upload and delete multiple images to/from the cloud. Following is the REST API that are available for usage.

POST /upload

This API is responsible for uploading the image to respective Openstack Cloud.

JSON Payload

```
{
"podsip":[ "172.31.231.17",
"10.30.116.244",
],
"images":[
"xxxxxx.iso", "yyyyyy.qcow2",
]
}
```

Response

```
{"Upload":true}
```

CURL Request

```
curl -s -k -X POST -d '{"upload": {"podsip":["172.23.105.218",
"172.29.85.78"],"images":["buildnode-internal-20606.iso","CentOS-7-
x86_64-GenericCloud-1503.qcow2"]}}'
-H "Auth: <Token>" https://172.29.85.78:9001/upload
```

Following is an example Curl request.

curl -s -k -X DELETE -d '{"upload": {"podsip":["172.23.105.218", "172.29.85.78"],"images":["buildnode-internal-20606.iso","CentOS-7-x86_64-GenericCloud-1503.qcow2"]}}'-H "Auth: " https://172.29.85.78:9001/upload

Delete/Upload

This API is responsible for deleting the image from respective Openstack Cloud.

JSON Payload

```
{
"podsip":[ "172.31.231.17",
"10.30.116.244",
],
"images":[
"xxxxxx.iso", "yyyyyy.qcow2",
]
}
```

CURL Request

Following is the example Curl request:

```
curl -s -k -X DELETE -d '{"upload": {"podsip":["172.23.105.218",
"172.29.85.78"],"images":["buildnode-internal-20606.iso","CentOS-7-
x86_64-GenericCloud-1503.qcow2"]}}'
-H "Auth: <Token>" https://172.29.85.78:9001/upload
```

Response

```
{"Delete":true}
```

GET /upload:

This API is responsible for getting the image list from respective Openstack Cloud. Following are the query string parameters to be passed with GET URL:

1. odsip: It is a comma separated string which represents pod IPs, whose Openstack image list needs to be fetched.
2. images: It is a comma separated string which represents Openstack images whose status needs to be fetched.
3. refresh: Takes the value true or false. Used to get updated Openstack images list.

Following are the CURL request examples:

Example 1

```
   curl -s -k -H "Auth: <Token>"
https://172.29.85.78:9001/upload
   This gives the result of pods on which upload/get/delete
operation are performed.
   {
   "uploaded": {
      "172.29.85.78": {
      "opsinprogress": 0,
      "images": null,
      "error": ""
                        },
      "172.23.105.218": {
      "opsinprogress": 0,
      "images": null,
      "error": ""
   }
   }
   }
```

Example 2:

```
   curl -s -k -H "Auth: <Token>"
https://172.29.85.78:9001/upload?"podsip=172.29.85.78"
   {
   "uploaded": {
   "172.29.85.78": {
   "opsinprogress": 0, "images": [
   {
   "OSStatus": "active", "UploadStatus": "UploadSuccess", "ErrStatus":
"",
   "ID": "c50284d7-191a-42ed-a289-9b52d19b9fd5",
   "Name": "buildnode-internal-20606.iso"
   },
   {
   "OSStatus": "active", "UploadStatus": "UploadSuccess", "ErrStatus":
"",
   "ID": "fee44efc-684e-46ac-aa89-b6e785faf1b4", "Name": "CentOS-7-
x86_64-GenericCloud-1503.qcow2"
   }
   ],
   "error": ""
   }
```

Example 3:

```
   curl -s -k -H "Auth: <Token>"
https://172.29.85.78:9001/upload?"podsip=172.29.85.78&refresh=true"
   {
      "uploaded": {
         "172.29.85.78": {
         "opsinprogress": 1,
         "images": null,
         "error": ""
   },
   }
```

Example 4:

```
   curl -s -k -H "Auth: <Token>"
https://172.29.85.78:9001/upload?"podsip=172.29.85.78&
images=buildnode-internal-20606.iso"
   {
      "uploaded": {
      "172.29.85.78": {
      "opsinprogress": 0,
      "images": [
   {
      "OSStatus": "active",
      "UploadStatus": "UploadSuccess",
      "ErrStatus": "",
      "ID": "c50284d7-191a-42ed-a289-9b52d19b9fd5",
      "Name": "buildnode-internal-20606.iso"
   }
   ],
   "error": ""
   }
   }
   }
```

HCLTech

VIM REST API Using Curl for IPv4

Getting REST API Username & Password

Use the following configuration to get REST API Username and Password

```
   cat /opt/cisco/ui_config.json
   {
   "Kibana-Url": "https://172.26.229.73:5601", "RestAPI-Username":
"admin",
   "RestAPI-Password": "cc52dc6d82bde0754ee3", "RestAPI-Url":
"https://172.26.229.73:8445",
   "BuildNodeIP": "172.26.229.73".   ---> br_api
   }
```

Got credentials from /opt/cisco/ui_config.json file

Nodes APIs and Commands

List Nodes

Use the following curl command to get the node's status, power status, reboot status, and mtype I nformation:

```
curl -i -X GET -u admin:**** -H 'Content-Type: application/json' -H 'Accept: application/json'
--cacert /var/www/mercury/mercury-ca.crt https://172.31.231.17:8445/v1/nodes
```

Response:

```
{"nodes": [{"status": ". . . . "name": "Store-2"}]}
```

Power OFF Nodes

To get the power off status of the nodes, use the below command:

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -
H 'Accept:
application/json' --cacert /var/www/mercury/mercury-ca.crt -d
'{'status': 'PowerOff', 'force_op': False, 'name': '<Node UUID>'}'
https://172.31.231.17:8445/v1/nodes/node_power_status
```

You can find the UUID of the node from the list nodes command.

Power ON Nodes

To get the power ON status of the nodes, use the following command:

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -
H 'Accept:
application/json' --cacert /var/www/mercury/mercury-ca.crt -d
'{'status': 'PowerOn', 'force_op': False, 'name': '<Node UUID>'}'
https://172.31.231.17:8445/v1/nodes/node_power_status
```


You can find the UUID of the node from the list nodes command.

Power Status of Nodes

To get the Live status of the nodes, first send POST request to /v1/hwinfoAPI, and then place the GET request on v1/hwinfo/get_nodes_power_status after a minute approximately.

Run the below commands to send the POST request and get the power status:

Request:

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -
H 'Accept: application/json' --cacert /var/www/mercury/mercury-ca.crt
-d '{}' https://172.31.231.17:8445/v1/hwinfo
curl -i -X GET -H 'Content-Type: application/json' -u admin:**** -H
'Accept: application/json'
--cacert /var/www/mercury/mercury-ca.crt
https://172.31.231.17:8445/v1/hwinfo/get_nodes_power_status
```

Response:

```
{'Store-3': {'intended_power_state': 'PowerOnSuccess',
'actual_power_state': 'on'},}}
```

Reboot Node:

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -
H 'Accept: application/json' --cacert /var/www/mercury/mercury-ca.crt
-d
'{'status': 'Reboot', 'force_op': False, 'name': '<Node UUID>'}'
https://172.31.231.17:8445/v1/nodes
/node_power_status
```

Reboot Status:

You can find the UUID of the node from the list nodes command.z

Use the following two commands, to get the reboot status of the node:

```
   curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -
H 'Accept: application/json' --cacert /var/www/mercury/mercury-ca.crt
   -d 'None' https://172.31.231.17:8445/v1/nodes/reboot_status
   curl -i -X GET -H 'Content-Type: application/json' -u admin:**** -H
'Accept: application/json'
   --cacert /var/www/mercury/mercury-ca.crt
https://172.31.231.17:8445/v1/nodes
```

List Openstack Configuration Command

Request:

```
   curl -i -X GET -H 'Content-Type: application/json' -u admin:**** -H
'Accept: application/json'
   --cacert /var/www/mercury/mercury-ca.crt
https://172.31.231.17:8445/v1/openstack_config
```

Response:

```
   {"KEYSTONE_VERBOSE_LOGGING": true, "GNOCCHI_VERBOSE_LOGGING": true,
. . }
```

Cluster Recovery

Command:

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -
H 'Accept:
application/json' --cacert /var/www/mercury/mercury-ca.crt -d
'{'action': {'cluster-recovery': {'run-disk- checks': False}}}'
https://172.31.231.17:8445/v1/misc
```

Response:

```
{'uuid': 'ae3be813-4fae-4510-8467-fab09ac60d2b', 'created_at':
'2019-01-07T08:17:01.229976+00:00', 'updated_at': None,
'operation_status': 'OperationScheduled', 'operation_logs': '',
'operation_name': {'cluster-recovery':
{'run-disk-checks': False}}}
```

Last-Run-Status Command

Command:

```
curl -i -X GET -H 'Content-Type: application/json' -H
'Authorization: ****' -H 'Accept: application/json' - H 'User-Agent:
python-ciscovimclient' --cacert
/var/www/mercury/mercury-ca.crt
https://172.31.231.17:8445/v1/op_info
```

Response:

```
{'created_at': '2019-01-07 08:27:56+00:00', 'updated_at': '2019-01-
07 08:28:03+00:00',
'reboot_required': False, 'update_status': False,
'current_op_logs': 'https://172.31.231.17:8008/mercury/79c402d2-f156-
4ba2-8f17-ec109401a538', 'current_op_status': 'OperationRunning',
'insight_monitor_status': 'Running', 'current_op_name':
'Generate_ssh_keys', 'current_op_monitor':
'Runner_Op_Generate_ssh_keys'}
```

Reconfigure Regenerate Secrets

Command:

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -
H 'Accept: application/json' --cacert /var
/www/mercury/mercury-ca.crt -d
'{'action': {'regenerate_secrets': '****', 'reconfigure': True}}'
https://172.31.231.17:8445/v1/misc
```

Response:

```
{'uuid': '83cf2700-275f-4c18-a900-96c36c4987aa', 'created_at':
'2019-01-07T08:36:19.279425+00:00',
'updated_at': None, 'operation_status': 'OperationScheduled',
'operation_logs': '',
'operation_name': {'regenerate_secrets': '****', 'reconfigure':
True}}
```

Reconfigure Set Password

Request:

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -
H 'Accept: application/json' --cacert /var
/www/mercury/mercury-ca.crt -d '{"action": {"setpassword":
{"HAPROXY_PASSWORD": "*****"}, "reconfigure": true}}'
https://172.31.231.17:8445/v1/misc
```

Response:

```
{"uuid": "16d89b9e-cadc-4467-b1d8-5a8a60171d90", "created_at":
"2020-06-30T16:51:17.316126+00:00",
"updated_at": null, "operation_status": "OperationScheduled",
"operation_logs": "", "operation_name": "{\" setpassword\":
{\"HAPROXY_PASSWORD\": \"****\"}, \"reconfigure\": true}"}
```

Reconfigure Set Openstack Configuration

Command:

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -
H 'Accept:
application/json' --cacert /var/www/mercury/mercury-ca.crt -d
'{'action': {'reconfigure': True,
'setopenstackconfigs':{'GNOCCHI_VERBOSE_LOGGING': True}}}'
https://172.31.231.17:8445/v1/misc
```

Response

```
{'uuid': '5bbbeff7-76df-4444-a38a-8819a8b579e4', 'created_at':
'2019-01-07T08:54:13.733254+00:00', 'updated_at': None,
'operation_status': 'OperationScheduled', 'operation_logs': '',
'operation_name': {'setopenstackconfigs':
{'GNOCCHI_VERBOSE_LOGGING': True}, 'reconfigure': True}}
```

Reconfigure CIMC Password

1. List down the setupdata and find UUID of active setupdata using the following command:

```
curl -i -X GET -H 'Content-Type: application/json' -u admin:**** -H
'Accept: application/json' --cacert /var/www/mercury/mercury-ca.crt
https://172.31.231.17:8445/v1
/setupdata
```

Response:

2. Put the content of setupdata with new CIMC Password using the following command:

```
curl -i -X PUT -H 'Content-Type: application/json' -u
admin:**** -H 'Accept: application/json' --cacert
/var/www/mercury/mercury-ca.crt -d '{'meta': {}, 'name':
'NEWSETUPDATA', 'jsondata':
{'external_lb_vip_address': '172.29.86.9' . . .}, 'uuid':
'3e97381e-4b1c-41a2-9af4-f970a1f1493a'}'
https://172.31.231.17:8445/v1/setupdata/3e97381e-4b1c-41a2-9af4-
f970a1f1493a
```

3.Post on Misc API using the below command:

Request:

```
  curl -i -X POST -H 'Content-Type: application/json' -u
admin:**** -H 'Accept: application/json' --cacert
/var/www/mercury/mercury-ca.crt -d '{'action':
  {'reconfigure_cimc_password': True, 'reconfigure': True}}'
https://172.31.231.17:8445/misc
```

Response:

```
  {'uuid': 'f00e1ae0-5674-4218-b1de-8995c9f9c546', 'created_at':
  '2019-01-07T09:19:40.210121+00:00', 'updated_at': None,
'operation_status': 'OperationScheduled', 'operation_logs': '',
'operation_name':
  {'reconfigure_cimc_password': '****', 'reconfigure': True}}
```

HCLTech

API Resources

- Setupdata
- Install resource
- Nodes
- Replace a controller
- Offline Validation
- Update Secrets
- OpenStack Configs
- Version
- Health of the Management Node
- Hardware Information
- Release Mapping Information

Setupdata

REST wrapper for setupdata. Provides methods for listing, creating, modifying, and deleting setupdata.

Retrieving the setupdata

Resource URI

Verb	URI
GET	/v1/setupdata

Example

JSON Request:

```
GET /v1/setupdata Accept: application/json
```

Curl Command:

curl -g -i -X GET -H 'Content-Type: application/json' -u admin:<Rest API admin password> -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert "<mercury-ca.crt path>" https://<br_api>:8445/setupdata

JSON Response

```
200 OK
Content-Type: application/json
{"setupdatas": [{
"status": "Active",
"name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}]}
```

Creating Setupdata

Resource URI

Verb	URI
POST	/v1/setupdata

Example

JSON Request

```
POST /v1/setupdata Accept: application/json
{ "name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}
```

Curl Command:

curl -g -i -X POST -H 'Content-Type: application/json' -u admin:<Rest API admin passwd> -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert "<mercury-ca.crt path>" -d '{"meta": {}, "name": "NEWSETUPDATA", "jsondata": { }}' https://<br_api>:8445/setupdata

JSON Response

```
201 OK
Content-Type: application/json
{
"status": "Active",
"name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}

400 Bad Request
Content-Type: application/json
{
"debuginfo": null "faultcode":"Client" "faultstring": "Error"
}
409 CONFLICT
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client" "faultstring": "Error"
}
```

Retrieving a Single Setupdata

Example
Resource URI

Verb	URI
GET	/v1/setupdata/(id)

Property:
id—The ID of the setupdata that you want to retrieve.

JSON Request

```
GET /v1/setupdata/123
Accept: application/json
```

Curl Command:curl -g -i -X GET -H 'Content-Type: application/json' -u admin:<Rest API admin password> -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert '<mercury-ca.crt path>' https://<br_api>:8445/setupdata/<setup data uuid>

JSON Response

```
200 OK
Content-Type: application/json
{
"status": "Active",
"name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}
404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Setupdata could not be found."
```

Updating a Setupdata

Resource URI

Verb	URI
PUT	/v1/setupdata/(id)

Property:

id—The ID of the setupdata that you want to update. Example

Convert setupdata response generated from GET request to JSON format then update field required to edit

JSON Request

```
PUT /v1/setupdata/123 Accept: application/json
```

```
$ curl -i -X PUT \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -u admin:<Rest API admin password> \
  --cacert "C:\certificates\mercury-ca.crt" \
  -d '{
   "name": "NEWSETUPDATA",
   "meta": {},
   "uuid": "<setupdata uuid>",
   "jsondata": {
   }
```

<Updated setup data in JSON format>
}' \
https://<br_api>:8445/setupdata/<setupdata uuid>

JSON Response

```
200 OK
Content-Type: application/json
{
"status": "Active",
"name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}
404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Setupdata could not be found."
}
```

Deleting a Setupdata

Resource URI

Verb	URI
DELETE	/v1/setupdata/(id)

Property:

id—The ID of the setupdata that you want to delete. Example

Get active setupdata UUID from GET request

JSON Request

```
DELETE /v1/setupdata/123 Accept: application/json
```

curl -g -i -X DELETE -H 'Content-Type: application/json' -u admin:<Rest API admin password> -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert "<mercury-ca.crt path>" https://<br_api>:8445/setupdata/<setupdata uuid>

JSON Response

```
   204 NO CONTENT Returned on success
   404 NOT FOUND
   Content-Type: application/json
   {
   "debuginfo": null "faultcode": "Client"
   "faultstring": "Setupdata could not be found."
   }
   400 BAD REQUEST
   Content-Type: application/json
   {
   "debuginfo": null "faultcode": "Client"
   "faultstring": "Setupdata cannot be deleted when it is being used
by an installation"
   }
```

Install Resource

REST wrapper for install.
Provides methods for starting, stopping, and viewing the status of the installation process

Return a List of Installation

Resource URI

Verb	URI
GET	/v1/install

Example

JSON Request

```
   GET /v1/install
   Accept: application/json
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {"installs": [{
   "ceph": "Skipped",
   "uuid": "123",
   "setupdata": "345",
   "vmtpresult": "{
   "status": "PASS",
   "EXT_NET": []
   }",
   "barematal": "Success", "orchestration": "Success",
"validationstatus": "{ "status": "PASS", "Software_Validation": [],
"Hardware_Validation": []
   }",
   "currentstatus": "Completed", "validation": "Success", "hostsetup":
"Success", "vmtp": "Skipped"
   }]
   }
```

Create an Installation

Resource URI

Verb	URI
POST	/v1/install

Example

JSON Request

```
   GET /v1/install Accept: application/js
   {
   "setupdata": "123", "stages": [ "validation", "bootstrap",
"runtimevalidation", "barematal", "orchestration", "hostsetup",
   "ceph", "vmtp"
   ]
   }
```

JSON Response

```
   201 CREATED
   Content-Type: application/json
   {
   "ceph": "Skipped",
   "uuid": "123",
   "setupdata": "345",
   "vmtpresult": "{
   "status": "PASS",
   "EXT_NET": []
   }",
   "baremetal": "Success", "orchestration": "Success",
"validationstatus": "{ "status": "PASS", "Software_Validation": [],
"Hardware_Validation": []
   }",
   "currentstatus": "Completed", "validation": "Success", "hostsetup":
"Success", "vmtp": "Skipped"
   }
   409 CONFLICT
   Content-Type: application/json
   {
   "debuginfo": null "faultcode": "Client"
   "faultstring": "Install already exists"
   }
```

Retrieve the Installation

Resource URI

Verb	URI
GET	/v1/install/{id}

Property:

id—The ID of the installation that you want to retrieve. Example

JSON Request

```
GET /v1/install/345
Accept: application/js
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   "ceph": "Skipped",
   "uuid": "123",
   "setupdata": "345",
   "vmtpresult": "{
   "status": "PASS",
   "EXT_NET": []
   }",
   "baremetal": "Success", "orchestration": "Success",
"validationstatus": "{ "status": "PASS", "Software_Validation": [],
"Hardware_Validation": []
   }",
   "currentstatus": "Completed", "validation": "Success", "hostsetup":
"Success", "vmtp": "Skipped"
   }
   404 NOT FOUND
   Content-Type: application/json
   {
   "debuginfo": null "faultcode": "Client"
   "faultstring": "Install doesn't exists"
   }
```

Stop the Installation

Resource URI

Verb	URI
DELETE	/v1/install/{id}

Property:

id—The ID of the installation that you want to stop.

Example

JSON Request

```
DELETE /v1/install/345
Accept: application/js
```

JSON Response

```
204 NO CONTENT
Content-Type: application/json

404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client" "faultstring":
"Install doesn't exists"
}
```

Nodes

Getting a List of Nodes

Resource URI

Verb	URI
GET	/v1/nodes

Example

Curl Command:

curl -i -X GET -H 'Content-Type: application/json' -H 'Accept: application/json' -H 'CVIM-API-Version: 4.0.0' -H 'User-Agent: python-ciscovimclient' -u admin:<Rest API Password> --cacert "C:\<mercury-ca.crt path>" https://<Pod IP>:8445/nodes

JSON Request

```
Get /v1/nodes
Accept: application/js
```

JSON Response

```
200 OK
Content-Type: application/json
{
"nodes": [ [
"status": "Active",
"uuid": "456",
"setupdata": "123",
"node_data": "{
"rack_info": { "rack_id": "RackA"
},
"cimc_info": {
"cimc_ip": "10.10.10.10"
},
"management_ip": "7.7.7.10"
}",
"updated_at": null, "mtype": "compute",
"install": "345", "install_logs": "logurl",
"created_at":"2016-0710T06:17:03.761152",
"name": " compute-1"
}
]
}
```

Add New Nodes

The nodes are in compute or block_storage type. Before adding the nodes to the system, the name of the nodes and other necessary information like cimc_ip and rackid must be updated in the setupdata object. If the setupdata object is not updated, the post call does not allow you to add the node. Resource URI

Verb	URI
POST	/v1/nodes

Example

CURL Command:
curl -i -X PUT -H "Content-Type: application/json" -H "Accept: application/json" -H "User-Agent: python-ciscovimclient" -u admin:<Password> --cacert </mercury-ca.crt path> -d @payload.json https://<POD IP>:8445/setupdata/<UUID>

JSON Request

```
POST /v1/nodes
Accept: application/js
{
"name" : "compute-5"
}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
"status": "ToAdd",
"uuid": "456",
"setupdata": "123",
"node_data": "{
"rack_info": { "rack_id": "RackA"
},
"cimc_info": { "cimc_ip": "10.10.10.10"
},
"management_ip": "7.7.7.10"
}",
"updated_at": null, "mtype": "compute",
"install": "345", "install_logs": "logurl",
"created_at":"2016-0710T06:17:03.761152",
"name": " compute-1"
}
```

Retrieve Information about a Particular Node

Resource URI

Verb	URI
GET	/v1/nodes{id}

Property:

id—The ID of the node that you want to retrieve.

Example

CURL Command:

curl -g -i -X GET -H 'Content-Type: application/json' -u admin:<Rest Api Password> -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert "<mercury-ca.crt path" https://<POD IP>:8445/v1/nodes

JSON Request

```
POST /v1/nodes
Accept: application/js
```

JSON Response

```
200 OK
Content-Type: application/json
{
"status": "Active",
"uuid": "456",
"setupdata": "123",
"node_data": "{
"rack_info": { "rack_id": "RackA"
},
"cimc_info": { "cimc_ip": "10.10.10.10"
},
"management_ip": "7.7.7.10"
}",
"updated_at": null, "mtype": "compute",
"install": "345", "install_logs": "logurl",
"created_at":"2016-0710T06:17:03.761152",
"name": " compute-1"
}
404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Node doesn't exists"
}
```

Remove a Node

The node to be deleted must be removed from the setupdata object. Once the setupdata object is updated, you can safely delete the node. The node object cannot be deleted until it calls the remote node backend and succeeds.

Resource URI

Verb	URI
DELETE	/v1/nodes{id}

Property:

id—The ID of the node that you want to remove.

Example

CURL Command:
curl -i -X GET -H 'Content-Type: application/json' -H 'Accept: application/json' -H 'CVIM-API-Version: 4.0.0' -H 'User-Agent: python-ciscovimclient' -u admin:<Rest API Password> --cacert "C:\<mercury-ca.crt path>" https://<POD IP>:8445

JSON Request

```
DELETE /v1/nodes/456
Accept: application/js
```

JSON Response

```
204 ACCEPTED
Content-Type: application/json
404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Node doesn't exists"
}
```

To clear the database and delete the entries in the nodes, the delete API is called with special parameters that are passed along with the delete request. The JSON parameters are in the following format.

JSON Request

```
DELETE /v1/nodes/456 Accept: application/js
{
"clear_db_entry":"True"\
}
```

JSON Response

```
204 ACCEPTED
Content-Type: application/json

404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Node doesn't exists"
}
```

Replace a Controller

Resource URI

Verb	URI
PUT	/v1/nodes{id}

Property:

id—The ID of the controller that you want to replace.

Example

> ⚠ This is done only if the node is deleted from the REST API database. The failure reason of the node must be rectified manually apart from the API. True is a string and not a boolean in the preceding line.

CURL Command:

curl -i -X PUT -H "Content-Type: application/json" -H "Accept: application/json" -H "User-Agent: python-ciscovimclient" -u admin: --cacert "C:<mercury-ca.crt pth>" -d '{"name": "<controller node name to be replaced>", "status": "ToReplace", "force_op": false, "skip_vmtp": false}' https://<br_api ip>:8445/nodes/UUID

JSON Request

```
PUT /v1/nodes/456 Accept: application/js
```

JSON Response

```
200 OK
Content-Type: application/json
404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Node doesn't exists"
}
```

Offline Validation

REST wrapper does the offline validation of setupdata. Rest Wrapper only does the Software Validation of the input setupdata.

Create an Offline Validation Operation

Resource URI

Verb	URI
POST	/v1/offlinevalidation

Example

CURL Command:

curl -g -i -X POST -H "Content-Type: application/json" -u admin:<Password> -H "Accept: application/json" --cacert 'C:\<mercury-ca.crt payh>' -d "{\"jsondata\": $(cat setup_data.json)}" https://<Pod IP>:8445/v1/offlinevalidation

JSON Request

```
POST /v1/offlinevalidation Accept: application/json
{
"jsondata": "."
}
```

JSON Response

```
   201 CREATED
   Content-Type: application/json
   {
   "status": "NotValidated",
   "uuid": "bb42e4ba-c8b7-4a5c-98b3-1f384aae2b69", "created_at":
"2016-02-03T02:05:28.384274", "updated_at": "2016-02-
03T02:05:51.880785",
   "jsondata": "{}", "validationstatus": { "status": "PASS",
"Software_Validation": [], "Hardware_Validation": []
   }
   }
```

Retrieve the Results of Offline Validation

Resource URI

Verb	URI
GET	/v1/offlinevalidation

Property:

id—The ID of the node you want to retrieve.

Example

Curl Command:

curl -g -i -X GET -H 'Content-Type: application/json' -u admin:<Password> -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert 'C:<mercury-ca.crt path>' https://<Pod IP>:8445/v1/offlinevalidation/<UUID>

JSON Request

```
   GET /v1/offlinevalidation/789 Accept: application/json
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   "status": " ValidationSuccess",
   "uuid": "bb42e4ba-c8b7-4a5c-98b3-1f384aae2b69", "created_at":
"2016-02-03T02:05:28.384274", "updated_at": "2016-02-
03T02:05:51.880785",
   "jsondata": "{}", "validationstatus": { "status": "PASS",
"Software_Validation": [], "Hardware_Validation": []
   }
   }
```

Update

Start an Update Process

Resource URI

Verb	URI
POST	/v1/update

Parameters:

- fileupload - "tar file to upload"
- filename - "Filename being uploaded"

Example.

JSON Request

```
   curl -sS -X POST --form "fileupload=@Test/installer.good.tgz" --
form "filename=installer.good.tgz" https://10.10.10.8445/v1/update
```

⚠ This curl request is done as a form request.

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   "update_logs": "logurl", "update_status": "UpdateSuccess",
   "update_filename": "installer-4579.tgz", "created_at": "2016-07-
10T18:33:52.698656", "updated_at": "2016-07-10T18:54:56.885083"
   }
   409 CONFLICT
   Content-Type: application/json
   {
   "debuginfo": null "faultcode": "Client" "faultstring": "Uploaded file
is not in tar format"
   }
```

Roll Back an Update

Resource URI

Verb	URI
PUT	/v1/update

Example

JSON Request

```
   PUT /v1/update
   Accept: application/json
   {
   "action":"rollback"
   }
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {

   "update_logs": "logurl", "update_status": "ToRollback",
"update_filename": "installer-4579.tgz", "created_at": "2016-07-
10T18:33:52.698656", "updated_at": "2016-07-10T18:54:56.885083"
   }
```

Commit an Update

Resource URI

Verb	URI
PUT	/v1/update

Example

JSON Request

```
   PUT /v1/update
   Accept: application/json
   {
   "action":"commit"
   }
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   "update_logs": "logurl", "update_status": "ToCommit",
"update_filename": "installer-4579.tgz", "created_at": "2016-07-
10T18:33:52.698656", "updated_at": "2016-07-10T18:54:56.885083"
   }
```

Retrieve the Details of an Update

Resource URI

Verb	URI
GET	/v1/update

Example

JSON Request

```
GET /v1/update
 Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
"update_logs": "logurl", "update_status": "UpdateSuccess",
"update_filename": "installer-4579.tgz",
"created_at": "2016-07-10T18:33:52.698656", "updated_at": "2016-07-
10T18:54:56.885083"
}
```

Secrets

Retrieve the list of secrets that are associated with the OpenStack Setup.

You can retrieve the set of secret passwords that are associated with the OpenStack setup using the preceding api. This gives the list of secrets for each service in OpenStack.

Resource URI

Verb	URI
GET	/v1/secrets

Example

Curl Command

curl -i -X GET \-H "Accept: application/json" \-u admin:<Password> \--cacert /<mercury-ca.crt path> \"https://<Pod Ip:8445>/v1/secrets

JSON Request

GET /v1/secrets

Accept: application/json

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   "HEAT_KEYSTONE_PASSWORD": "xxxx", "CINDER_KEYSTONE_PASSWORD":
"xxxxx",
   ....
   ....
   "RABBITMQ_PASSWORD": "xxxxx"
   }
```

OpenStack Configs

Retrieve the list of OpenStack configs associated with the OpenStack Setup

You can retrieve the set of OpenStack configs associated with the OpenStack setup using the preceding api. This gives the current settings of different configurations such as verbose logging and debug logging for different OpenStack services.

URI
/v1/openstack_config

Curl Command

curl -i -X GET \-H "Content-Type: application/json" \-H "Accept: application/json" \-u admin:<Password>\cacert/<mercuryca.crtpath>\https://<PodIP>:8445/v1/openstack_config

JSON Request

GET /v1/openstack_config

Accept: application/json

JSON Response

```
  200 OK
  Content-Type: application/json
  {
  "CINDER_DEBUG_LOGGING": false, "KEYSTONE_DEBUG_LOGGING": false,
  ....
  ....
  "NOVA_VERBOSE_LOGGING": true
  }
```

Version

Retrieve the version of the Virtualized Infrastructure Manager.

Resource URI

Verb	URI
GET	/v1/version

Example

Curl Command

curl -i -X GET -H 'Accept: application/json' -u admin:<Rest APi admin passwd> --cacert <mercury cert path> https://<br_api ip>:8445/v1/version

JSON Request

```
GET /v1/version Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{"version": "1.9.1"}
```

Health of the Management Node

Retrieve the health of the Management node

This API is used to retrieve the health of the management node. It checks various parameters such as partitions, space and so on. Resource URI

Verb	URI
GET	/v1/health

Example

Curl command:

curl -i -X GET -H 'Content-Type: application/json' -H 'Accept: application/json' -H 'CVIM-API-Version: 4.0.0' -H 'User-Agent: python-ciscovimclient' -u admin:<rest api admin passwd> --cacert "<mercury cert path>" https://<br_api ip>:8445/v1/health

JSON Request

```
GET /v1/health
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
"status": "PASS", "pod_status": { "color": "BLUE", "version":
"<VERSION_NO.>"
},
"insight_version": "<VERSION_NO.>"
}
```

Color signifies the health of the pod for Insight:

- Grey signifies that no installation is kicked off on the pod.
- Green signifies that everything is in Good state and cloud installation is active.
- Blue signifies that some operation is running on the pod.
- Red signifies that the pod is in critical state, and you might need TAC support to recover the pod.
- Amber indicates a warning if a pod management (Add/Remove/Replace) operation failed.

Hardware Information

REST wrapper to do hardware information of setupdata. This returns the hardware information of all hardware available in the setupdata.

Create a HWinfo Operation

Resource URI

Verb	URI
GET	/v1/hwinfo

Example

Curl command:

curl -k -u admin:<rest api admin passwd> -H "Accept: application/json" -H "Content-Type: application/json" -X GET "https://<br_api>:8445/v1/hwinfo"

JSON Request

```
POST /v1/hwinfo Accept: application/json
{
"setupdata":"c94d7973-2fcc-4cd1-832d-453d66e6b3bf"
}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
"status": "hwinfoscheduled",
"uuid": "928216dd-9828-407b-9739-8a7162bd0676",
"setupdata": "c94d7973-2fcc-4cd1-832d-453d66e6b3bf", "created_at": "2017-03-19T13:41:25.488524", "updated_at": null, "hwinforesult": ""
}
```

Retrieve the Results of Hwinfo Operation

Resource URI

Verb	URI
GET	/v1/hwinfo/{id}

Property:

id—The ID of the node you want to query.

Example

Curl command:

curl -k -u admin:<rest api admin passwd> -H "Accept: application/json" -H "Content-Type: application/json" -X GET https://<br_api>:8445/v1/hwinfo

JSON Request

```
   GET /v1/hwinfo/789 Accept: application/json
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   "status": "hwinfosuccess",
   "uuid": "928216dd-9828-407b-9739-8a7162bd0676",
   "setupdata": "c94d7973-2fcc-4cd1-832d-453d66e6b3bf", "created_at": "2017-03-
19T13:41:25.488524", "updated_at": "2017-03-19T13:42:05.087491",
"hwinforesult": "{\"172.29.172.73\": {\"firmware\": ……..
   ……
   ……..
   }
```

Release Mapping Information

This api is used to see the list of Features included and list of options which can be reconfigured in the Openstack Setup.

Retrieve the release mapping information

Resource URI

Verb	URI
GET	/v1/releasemapping

Curl Command

curl -i -X GET -H 'Accept: application/json' -u admin:<rest api admin paswd> --cacert <mercury cert path> https://<br_api ip>:8445/v1/releasemapping

JSON Request

```
   GET /v1/releasemapping Accept: application/json
```

JSON Response

200 OK
Content-Type: application/json [
{
"SWIFTSTACK": {
"feature_status": true,
],
"desc": "swift stack feature"
}
},.....
.....
}

POST Install Operations

The following are the post install operations that can be performed, after the successful installation of OpenStack. It uses a common api.

Example:

1. reconfigure
2. reconfigure -regenerate passwords
3. reconfigure -setpasswords,setopenstack_configs
4. reconfigure -alertmanager_config, -alerting_rules_config
5. check-fernet-keys
6. resync-fernet-keys
7. rotate-fernet-keys

Create a Post install Operation

Resource URI

Verb	URI
POST	/v1/misc

Examples:

JSON Request

```
   POST /v1/misc
   Accept: application/json
   {"action": {"reconfigure": true}}
```

Curl command

curl -i -X POST --insecure -H 'Content-Type: application/json' -u admin:<Rest api admin passwd> -H 'Accept: application/json' -d '{"action": {"reconfigure": true}}' https://<br_api ip>:8445/v1/misc

JSON Response

```
   201 CREATED
   Content-Type: application/json
   {
   "uuid": "7e30a671-bacf-4e3b-9a8f-5a1fd8a46733", "created_at": "2017-03-
19T14:03:39.723914", "updated_at": null,
   "operation_status": "OperationScheduled", "operation_logs": "",
"operation_name": "{"reconfigure": true}"
   }
```

JSON Request

```
   POST /v1/misc
   Accept: application/json
   {"action": {"reconfigure": true, "alertmanager_config": <json_config>}}
```

JSON Response

```
   201 CREATED
   Content-Type: application/json
   {
```

```
   "uuid": "68b67265-8f09-480e-8608-b8aff77e0ec7", "created_at": "2019-01-
09T16:42:11.484604+00:00",
   "updated_at": null,

   "operation_status": "OperationScheduled", "operation_logs": "",
"operation_name": "{"alertmanager_config": <json_config>, "reconfigure": true}"
   }
```

Retrieve a Status of the Post Install Operation

Resource URI

Verb	URI
GET	/v1/misc

Example

JSON Request

```
   GET /v1/misc
   Accept: application/json
```

Curl command:

curl -i -X GET -H "Accept: application/json" -u admin:<rest api admin passwd> --cacert "<mercury cert path>" https://<br_api ip>:8445/v1/misc

JSON Response

```
   201 CREATED
   Content-Type: application/json
   {
   "uuid": "7e30a671-bacf-4e3b-9a8f-5a1fd8a46733", "created_at": "2017-03-
19T14:03:39.723914", "updated_at": "2017-03-19T14:03:42.181180",
   "operation_status": "OperationRunning", "operation_logs":
"xxxxxxxxxxxxxxxx", "operation_name": "{\"reconfigure\": true}"
   }
```

In VIM Rest APIs exist to support NFVBench, query hardware information and to get a list of optional and mandatory features that the pod supports.

Following are the API details:

NFVBench Network Performance Testing

Create NFVBench Run

Starts the network performance test with provided configuration.

Verb	URI
Post	v1/nfvbench/ create_ndr_pdr_test

Example

JSON Request

Curl command:

POST Request URL

```
/v1/nfvbench/create_fixed_rate_test JSON Request:
{"nfvbench_request":
{
"duration_sec": 20, "traffic_profile": [
{
"name": "custom", "l2frame_size": [

"64",
"IMIX", "1518"
]
}
],
"traffic": { "bidirectional": true, "profile": "custom"
},
"flow_count": 1000
}
}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
"status": "not_run", "nfvbench_request":
'{
"duration_sec": 20, "traffic_profile": [
{
"name": "custom", "l2frame_size": [ "64",
"IMIX", "1518"
]
}
],
"traffic": { "bidirectional": true, "profile": "custom"
},
"flow_count": 1000
}',
"created_at": "2017-08-16T06:14:54.219106",
"updated_at": null, "nfvbench_result": "", "test_name": "Fixed_Rate_Test"
}
```

REST API To Create Fixed Rate Test

Verb	URI
Post	v1/nfvbench/ create_ndr_pdr_test

Example

JSON Request

POST Request URL

```
/v1/nfvbench/create_fixed_rate_test JSON Request:
{"nfvbench_request":
{
"duration_sec": 20, "traffic_profile": [
{
"name": "custom", "l2frame_size": [

"64",
"IMIX", "1518"
]
}
],
"traffic": { "bidirectional": true, "profile": "custom"
},
"flow_count": 1000
}
}
```

Curl command:

```
curl -k -u admin:<rest api admin passwd> \ -H "Accept: application/json" \
  -H "Content-Type: application/json" \
  -X POST "https://<br_api>:8445/v1/nfvbench/create_fixed_rate_test" \
  -d '{
        "nfvbench_request": {
          "duration_sec": 20,
          "traffic_profile": [
            {
              "name": "custom",
              "l2frame_size": [
                "64",
                "IMIX",
                "1518"
              ]
            }
          ],
```

```
    "traffic": {
      "bidirectional": true,
      "profile": "custom"
    },
    "flow_count": 1000
  }
  }'
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
"status": "not_run", "nfvbench_request":
'{
"duration_sec": 20, "traffic_profile": [
{
"name": "custom", "l2frame_size": [ "64",
"IMIX", "1518"
]
}
],
"traffic": { "bidirectional": true, "profile": "custom"
},
"flow_count": 1000
}',
"created_at": "2017-08-16T06:14:54.219106",
"updated_at": null, "nfvbench_result": "", "test_name": "Fixed_Rate_Test"
}
```

Status Polling

Polling of NFVbench run status which is one of nfvbench_running, nfvbench_failed, nfvbench_completed.

Resource URI

Verb	URI
GET	v1/nfvbench/<test_name>

REST API To Get Fixed Rate Test Result

GET Request URL

```
/v1/upgrade/get_fixed_rate_test_result
```

JSON Request:

```
   Check If NFVbench Test is running
   200 OK
   Content-Type: application/json

   {
   "status": "nfvbench_running",
   "nfvbench_request": '{"traffic": {"bidirectional": true, "profile":
"custom"}, "rate": "1000000pps",
   "traffic_profile": [{"l2frame_size": ["1518"], "name": "custom"}],
"duration_sec": 60, "flow_count": 1000}', "nfvbench_result": ""
   "created_at": "2017-05-30T21:40:40.394274", "updated_at": "2017-05-
30T21:40:41.367279",
   }
```

JSON Response

Check If NFVbench test is completed:

```
   200 OK
   Content-Type: application/json
   {
   "status": "nfvbench_completed",
   "nfvbench_request": '{"traffic": {"bidirectional": true, "profile":
"custom"}, "rate": "1000000pps", "traffic_profile": [{"l2frame_size": ["1518"],
"name": "custom"}], "duration_sec": 60, "flow_count": 1000}',
   "nfvbench_result": '{"status": "PROCESSED", "message": {"date": "2017-08-15
23:15:04", "nfvbench_version": "0.9.3.dev2",   }
   "created_at": "2017-05-30T21:40:40.394274", "updated_at": "2017-05-
30T22:29:56.970779",
   }
```

REST API to create NDR/PDR Test POST Request URL

JSON Request

```
   /v1/nfvbench/create_ndr_pdr_test Accept: application/json
   {"nfvbench_request":
   {
   "duration_sec": 20, "traffic_profile": [
   {
   "name": "custom", "l2frame_size": [ "64",
   "IMIX", "1518"
   ]
   }
   ],
   "traffic": { "bidirectional": true, "profile": "custom"
```

```
   },
   "flow_count": 1000
   }
   }
```

JSON Response

```
   201 CREATED
   Content-Type: application/json
   {
   "status": "not_run", "nfvbench_request":
   '{
   "duration_sec": 20, "traffic_profile": [
   {
   "name": "custom", "l2frame_size": [

   "64",
   "IMIX", "1518"
   ]
   }
   ],
   "traffic": { "bidirectional": true, "profile": "custom"
   },
   "flow_count": 1000
   }'
   "created_at": "2017-08-16T07:18:41.652891",
   "updated_at": null, "nfvbench_result": "", "test_name": "NDR_PDR_Test"
   }
```

REST API To Get NDR/PDR Test Results

GET Request URL

```
   /v1/ nfvbench/get_ndr_pdr_test_result
   JSON Response:
   If NFVbench NDR/PDR test is running:
   200 OK
   Content-Type: application/json
   {
   "status": "nfvbench_running", "nfvbench_request": '{"duration_sec": 20,
"traffic": {"bidirectional": true, "profile": "custom"},
   "traffic_profile": [{"l2frame_size": ["64", "IMIX", "1518"], "name":
"custom"}], "flow_count": 1000}', "nfvbench_result": ""
   "created_at": "2017-08-16T07:18:41.652891", "updated_at": "2017-09-
30T22:29:56.970779",
   }
   If NFVbench NDR/PDR test is completed:
   200 OK
   Content-Type: application/json
   {
   "status": "nfvbench_completed", "nfvbench_request": '{"duration_sec": 20,
"traffic": {"bidirectional": true, "profile": "custom"},
   "traffic_profile": [{"l2frame_size": ["64", "IMIX", "1518"], "name":
"custom"}], "flow_count": 1000}',
```

```
   "nfvbench_result": '{"status": "PROCESSED",...}' "created_at": "2017-08-
16T07:18:41.652891", "updated_at": "2017-09-30T22:29:56.970779",
   }
```

Curl command:

curl -i -X GET -H "Content-Type: application/json" -H "Accept: application/json" -u admin:<rest api admin passwd> --cacert <mercury cert path> "https://<br_api ip>:8445/v1/nfvbench/get_ndr_pdr_test_result"

REST API to Get Node Hardware Information

Rest API helps you to get the hardware information of all the nodes in the:

- POD through CIMC/UCSM.
- Total Memory
- Firmware Info (Model, Serial Number)
- CIMC IP

GET Request URL

```
   /v1/hwinfo Output Response
   {
   "hwinforesult": "{"control-server-2": {"memory": {"total_memory": "131072"},
"firmware": {"serial_number": "FCH1905V16Q", "fw_model": "UCSC-C220- M4S"},
"cimc_ip": "172.31.230.100", "storage": {"num_storage": 4},
"cisco_vic_adapters": {"product_name": "UCS VIC 1225"},
   "cpu": {"number_of_cores": "24"}, "power_supply": {"power_state": "on"}}
   …
   }
```

REST API to Get Mandatory Features Mapping

Curl Command

curl -i -X GET \ -H "Accept: application/json" \-u admin:Password \--cacert /<mercury-ca.crt path>\https://<POD IP>:8445/v1/releasemapping/mandatory_features_mapping

POST Request URL

```
   /v1/releasemapping/mandatory_features_mapping JSON Response:
   {
   "mandatory": { "networkType": { "C": {
   "feature_status": true,
   "values": [{"name": "VXLAN/Linux Bridge", "value": "VXLAN/Linux Bridge"},],
"insight_label": "Tenant Network", "desc": "Tenant Network"
   }, "B": {
   "feature_status": true,
   "values": [{"name": "VXLAN/Linux Bridge", "value": "VXLAN/Linux Bridge"},],
"insight_label": "Tenant Network", "desc": "Tenant Network"
   }
   },
```

```
   "cephMode": {
   "all": { "feature_status": true,
   "values": [{"name": "Central", "value": "Central"},], "insight_label": "Ceph
Mode", "desc": "Ceph Mode"
   }
   },
   "podType": {
   "C": {
   "feature_status": true,
   "values": [{"name": "Fullon", "value": "fullon"},], "insight_label": "POD
Type", "desc": "POD Type"
   }, "B": {
   "feature_status": true,
   "values": [{"name": "Fullon", "value": "fullon"},], "insight_label": "POD
Type", "desc": "POD Type"
   }
   },
   "installMode": { "all": { "feature_status": true,
   "values": [{"name": "Connected", "value": "connected"}, ], "insight_label":
"Install Mode", "desc": "Install Mode"
   }

   }
   },
   "platformType": [{"name": "B-series", "value": "B"}, {"name": "C-series",
"value": "C"}], "postinstalllinks": {
   "view_cloudpulse": {"alwayson": true, "feature_status": true,
"platformtype": "all", "insight_label": "Run VMTP", "desc": "Cloudpluse"},
"password_reconfigure": {"alwayson": true, "feature_status": true,
"platformtype": "all", "insight_label": "Reconfigure Passwords", "desc":
"Reconfigure Passwords"}
```


}

}

REST API to Get Optional Features Mapping

Curl Command:

curl -i -X GET \-H "Accept: application/json" \ -u admin:Password \ --cacert /<mercury-ca.crt path>\
https://<POD IP>:8445/v1/releasemapping/optional_features_mapping

POST Request URL


```
   /v1/releasemapping/optional_features_mapping JSON Response: [
   {
   "SWIFTSTACK": {
   "feature_status": true, "insight_label": "Swiftstack",
"repeated_redeployment": true, "reconfigurable": ["cluster_api_endpoint",
"reseller_prefix", "admin_password", "protocol"],
   "desc": "swift stack feature"
   }
```

```
   },
   {
   "heat": {
   "feature_status": true, "insight_label": "Heat", "repeated_redeployment":
false, "reconfigurable": ["all"], "desc": "Openstack HEAT service"
   }
   },
   .      other features
   ]
```

Cloud Sanity Information

REST wrapper to run cloud-sanity test suites. The cloud-sanity extension to the VIM REST API enables support for managing cloud-sanity test actions.

Create a cloud-sanity Test

Verb	URI
Post	/v1/cloud-sanity/create

Example

Curl Command:

curl -i -X POST /-H "Accept: application/json" \-H "Content-Type: application/json" \-u admin:<Password>\cacert/<mercuryca.crt path>\ https://<Pod IP>:8445/v1/cloudsanity/create \\-d '{"cloudsanity_request": {"command": "create","action": "test","test_name": "cephmon","uuid": "<UU ID of the node>"}}

JSON Request

```
   POST /v1/cloudsanity/create Accept: application/json
'{"cloudsanity_request": {"command": "create", "action": "test", "test_name":
"cephmon", "uuid": ""}}'
   test_name can be all,management,control,compute,cephmon,cephosd
   JSON Response
   201 Created
   {
   'cloudsanity_request': "{u'action': u'test', u'command': u'create', u'uuid':
'5dff1662-3d33-4901-808d-479927c01dde', u'test_name': u'cephmon'}",
'cloudsanity_result': '',
   'created_at': '2018-01-26T20:32:20.436445',
   'status': 'not_run', 'test_name': 'cephmon', 'updated_at': ''
   }
```

Retrieve a Status of the Post Install Operation

Resource URI

Verb	URI
GET	/v1/misc

Example:

CURL Command:

curl -i -X GET \-H "Accept: application/json" \-u admin:<Password> \--cacert /<mercury-ca.crt path> \https://<Pod IP>:8445/v1/misc

JSON Request

```
GET /v1/misc
Accept: application/json
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
"uuid": "7e30a671-bacf-4e3b-9a8f-5a1fd8a46733", "created_at": "2017-03-19T14:03:39.723914", "updated_at": "2017-03-19T14:03:42.181180",
"operation_status": "OperationRunning", "operation_logs": "xxxxxxxxxxxxxxxx", "operation_name": "{\"reconfigure\": true}"
}
```

In VIM Rest APIs exist to support NFVBench, query hardware information and to get a list of optional and mandatory features that the pod supports. Following are the API details:

JSON Response

```
200 OK
Content-Type: application/json [
{
"SWIFTSTACK": {
"feature_status": true,
],
"desc": "swift stack feature"
}
},.....
.............
}
```

List Specific cloud-sanity Test Results

Verb	URI
GET	/v1/cloud-sanity/list/?test_name={all, management, control,compute,cephmon,cephosd}

CURL Command:

curl -i -X GET \-H "Accept: application/json" \-u admin:<Password> \--cacert /<mercury-ca.crt path>\https://<POD IP>:8445/v1/cloudsanity/list/?test_name=cephmon

JSON Request

```
GET /v1/cloudsanity/list/?test_name=cephmon Accept: application/json
```

JSON Response

```
200 OK
{ '5dff1662-3d33-4901-808d-479927c01dde': { 'action': 'test', 'created_at': '2018-01-26 20:32:20',
'status': 'cloudsanity_completed', 'test_name': 'cephmon', 'uuid': '5dff1662-3d33-4901-808d-479927c01dde'},
```

```
   '797d79ba-9ee0-4e11-9d9e-47791dd05e07': { 'action': 'test', 'created_at':
'2018-01-25 12:05:11',
   'status': 'cloudsanity_completed', 'test_name': 'cephmon', 'uuid':
'797d79ba-9ee0-4e11-9d9e-47791dd05e07'}}
```


Show cloud-sanity Test Results

Verb	URI
GET	/v1/cloud-sanity/show/?uuid=<uuid>

CURL Command:

curl -X GET "https://<host>:<port>/v1/cloudsanity/show/?uuid=<uu id>" \-H "Accept: application/json" \-H "Authorization: Bearer <access_token>"

JSON Request

```
   GET /v1/cloudsanity/show/?uuid=d0111530-ee3b-45df-994c-a0917fd18e11
```

JSON Response


```
   200 OK
   { 'action': 'test', 'cloudsanity_request':
   "{u'action': u'test', u'command': u'create',
   u'uuid': 'd0111530-ee3b-45df-994c-a0917fd18e11', u'test_name': u'control'}",
'cloudsanity_result': '{"status": "PROCESSED",
   "message": {"status": "Pass",
   "message": "[PASSED] Cloud Sanity Control Checks Passed", "results":
{"control": {"ping_all_controller_nodes": "PASSED",
"check_rabbitmq_is_running": "PASSED", "check_rabbitmq_cluster_status":
"PASSED", "check_nova_service_list": "PASSED", "ping_internal_vip": "PASSED",
   'created_at': '2018-01-26 18:46:23',
   'status': 'cloudsanity_completed', 'test_name': 'control', 'updated_at':
'2018-01-26 18:47:58',
   "disk_maintenance_raid_health": "PASSED", "check_mariadb_cluster_size":
"PASSED", "disk_maintenance_vd_health": "PASSED"}}}', 'uuid': 'd0111530-ee3b-
45df-994c-a0917fd18e11'}
```


Delete cloud-sanity Test Results

Verb	URI
DELETE	/v1/cloud-sanity/delete/?uuid=<uuid>

Curl Command:

curl -X DELETE "https://<host>:<port>/v1/cloudsanity/delete/?uuid=<uu id>" \-H "Accept: application/json" \-u admin:<password>

JSON Request

```
   GET /v1/cloudsanity/delete/?uuid=444aa4c8-d2ba-4379-b035-0f47c686d1c4
```

JSON Response


```
   200 OK
   {
   "status": "deleted",
```

```
   "message": "UUID 444aa4c8-d2ba-4379-b035-0f47c686d1c4 deleted from
database", "uuid": "444aa4c8-d2ba-4379-b035-0f47c686d1c4", "error": "None"
   }
```


Disk Maintenance information

REST wrapper to query information about RAID disks on Pod nodes. This returns the RAID disk I nformation of all or a selection of RAID disks available in the Pod.

The disk management extension to the VIM REST API enables support for Disk Management actions

Create a Check Disk Operation

Resource URI

Verb	URI
POST	/v1/diskmgmt/check_disks

Example

CURL Command:

curl -k -u admin:<Password> \-H "Accept: application/json" \-H "Content-Type:application/json" \-X POST "https://<Pod IP>:8445/v1/diskmgmt/check_disks" \-d '{"diskmgmt_request": {"command": "create","action": "check-disks","role": "control","locator": "False","json_display": "False","servers": "","uuid": ""}}'

JSON Request

```
   POST /v1/diskmgmt/check_disks Accept: application/json '{"diskmgmt_request":
{"command": "create", "action": "check-disks",
   "role": "control",
   "locator": "False", "json_display": "False", "servers": "", "uuid": ""}}'
```

JSON Response

```
   201 Created
   Content-Type: application/json
   {
   'action': 'check-disks',
   'created_at': '2018-03-08T02:03:18.170849+00:00',
   'diskmgmt_request': "{u'uuid': '0729bdea-cc19-440f-8339-ab21e76be84b',

   u'json_display': u'False', u'servers': u'',
   u'locator': u'False', u'role': u'control', u'action': u'check-disks',
u'command': u'create'}", 'diskmgmt_result': '', 'status': 'not_run',
'updated_at': 'None'
   }
```

Create a Replace Disk Operation

Verb	URI
POST	/v1/diskmgmt/replace_disks

Example

CURL Command:

curl -k -u admin:<Password> \-H "Accept: application/json" \-H "Content-Type: application/json" \-X POST "https://<Pod IP>:8445/v1/diskmgmt/replace_disks" \-d '{"diskmgmt_request": {"command": "create","action": "replace-disks","role": "control","locator": "False","json_display": "False","servers": "","uuid": ""}}'

JSON Request

```
   POST /v1/diskmgmt/replace_disks Accept: application/json
'{"diskmgmt_request": {"command": "create",
   "action": "replace-disks", "role": "control",
   "locator": "False", "json_display": "False", "servers": "", "uuid": ""}}'
```

JSON Response

```
   201 Created
   Content-Type: application/json
   {
   "status": "not_run",
   "diskmgmt_request": "{u'uuid': 'cb353f41-6d25-4190-9386-330e971603c9',
u'json_display': u'False', u'servers': u'', u'locator': u'False', u'role':
u'control',
   u'action': u'replace-disks', u'command': u'create'}", "created_at": "2018-
03-09T12:43:41.289531+00:00",
   "updated_at": "", "diskmgmt_result": "", "action": "replace-disks"}
```

List Check Disk Operation

Verb	URI
GET	/v1/diskmgmt/list/?action={check-disks,replace-disks\&role={all,management,control,compute}

Example

CURL Command:

curl -k -u admin:<Password> \-H "Accept: application/json" \-X GET "https://<POD IP>:8445/v1/diskmgmt/list/?action=check-disks&role=all"

JSON Request

```
   GET /v1/diskmgmt/list/?action=check-disks\&role=all
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   '0be7a55a-37fe-43a1-a975-cbf93ac78893': {'action': 'check-disks',
'created_at': '2018-03-05 14:45:45+00:00',
   'role': 'compute',
   'status': 'diskmgmt_completed', 'uuid':
   '0be7a55a-37fe-43a1-a975-cbf93ac78893'},
   '861d4d73-ffee-40bf-9348-13afc697ee3d': {'action': 'check-disks',
'created_at': '2018-03-05 14:44:47+00:00',
   'role': 'control',
   'status': 'diskmgmt_completed', 'uuid':
   '861d4d73-ffee-40bf-9348-13afc697ee3d'},
   'cdfd18c1-6346-47a2-b0f5-661305b5d160': {'action': 'check-disks',
'created_at': '2018-03-05 14:43:50+00:00',
   'role': 'all',
   'status': 'diskmgmt_completed', 'uuid':
```

```
   'cdfd18c1-6346-47a2-b0f5-661305b5d160'}
   }
   }
```

Show a Completed diskmgmt Operation

Verb	URI
GET	v1/diskmgmt/show/?uuid=<uuid>

Example

CURL Command:

curl -k -u admin:<Password> \-H "Accept: application/json" \-X GET "https://<Pod IP>:8445/v1/diskmgmt/show/?uuid=<UUID>"

JSON Request

```
   GET /v1/diskmgmt/show/?uuid=d24036c6-4557-4c12-8695-a92f6f9315ed
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {'action': 'check-disks',
   'created_at': '2018-03-07 21:46:41+00:00',
   'diskmgmt_request': "{u'uuid': 'd24036c6-4557-4c12-8695-a92f6f9315ed',
u'json_display': False, u'servers': u'f24-michigan-micro-2', u'locator': False,
u'role': u'compute', u'action': u'check-disks', u'command': u'create'}",
   'diskmgmt_result': '{"status": "PROCESSED", "message":
["{\'Overall_Status\': \'PASS\',
   \'Result\': {\'fcfg_disks_results_list\': [], \'spare_disks_results_list\':
[],
   \'raid_results_list\': [{\'RAID level\': \'RAID1\', \'Disk Med\': \'HDD\',
\'server\':
   \'7.7.7.6\', \'RAID type\': \'HW\', \'host\': \'f24-michigan-micro-2\',
\'role\':
   \'block_storage control compute\', \'VD health\': \'Optl\', \'Num VDs\': 1,
\'Num PDs\': 8, \'RAID health\': \'Opt\'}], \'bad_disks_results_list\': [],
   \'rbld_disks_results_list\':
   [], \'add_as_spares_disks_results_list\': []}}"]}', 'role': 'compute',
   'status': 'diskmgmt_completed', 'updated_at': '2018-03-07 21:47:35+00:00',
'uuid': 'd24036c6-4557-4c12-8695-a92f6f9315ed'
   }
```

Delete a Completed diskmgmt Operation

Verb	URI
DELETE	v1/diskmgmt/delete/?uuid=<uuid>

Example

CURL Command:

curl -k -u admin:<Password> \-H "Accept: application/json" \-X DELETE "https://<Pod IP>:8445/v1/diskmgmt/delete/?uuid=<UUID>"

JSON Request

```
DELETE /v1/diskmgmt/delete/?uuid=d24036c6-4557-4c12-8695-a92f6f9315ed
```

JSON Response

```
200 OK
Content-Type: application/json
{
"status": "deleted",
"message": "UUID d24036c6-4557-4c12-8695-a92f6f9315ed deleted from database", "uuid": "d24036c6-4557-4c12-8695-a92f6f9315ed", "error": "None"
}
```

OSD Maintenance Information

REST wrapper to query information about OSD on Pod storage nodes. This returns to the OSD status information of all or a selection of OSDs available in the Pod.

Create an OSD Disk Operation

Verb	URI
POST	/v1/osdmgmt/check_osds

Example

CURL Command:

curl -k -u admin:<Password> \-H "Accept: application/json" \-H "Content-Type: application/json" \-X POST "https://<POD IP>:8445/v1/osdmgmt/check_osds" \-d '{"osdmgmt_request": {"command": "create","action": "check-osds","locator": "False","json_display": "False","servers": "","osd": "None","tag": "all","uuid": ""}}'

JSON Request

```
POST /v1/osdmgmt/osdmgmt/check_osds '{"osdmgmt_request": {"command": "create", "action": "check-osds",
"locator": "False", "json_display": "False", "servers": "", "osd": "None",
"uuid": ""}}'
```

JSON Response

```
201 Created
Content-Type: application/json
{
'action': 'check-osds',
'created_at': '2018-03-08T21:26:15.329195+00:00',
'osdmgmt_request': "{u'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223', u'json_display': u'False', u'servers': u'', u'locator': u'False', u'command': u'create', u'action':
u'check-osds', u'osd': u'None'}", 'osdmgmt_result': '', 'status': 'not_run', 'updated_at': 'None'
}
```

Create a Replace OSD Operation

Verb	URI
POST	v1/osdmgmt/replace_osd

Note:

Login into any controller node and execute “ceph osd tree” and get any osd id from the storage node listed.

Example

CURL Command:

curl -k -u admin:<Password> \-H "Accept: application/json" \-H "Content-Type: application/json" \-X POST "https://<POD IP>:8445/v1/osdmgmt/replace_osd" \d '{osdmgmt_request": {command": "create","action": "replace-osd","locator": "False","json_display": "False","servers": "<Storage server name>","osd": "<osd.Id from “ceph osd tree”>","tag": "all","uuid": ""}}'

JSON Request

```
   POST /v1/osdmgmt/replace_osd Accept: application/json '{"osdmgmt_request":
{"command": "create",
   "action": "replace-osd",
   "locator": "False", "json_display": "False", "servers": "f24-michigan-micro-
1", "osd": "osd.9", "uuid": ""}}'
```

JSON Response

```
   201 Created
   Content-Type: application/json
   {
   "status": "not_run",
   "osdmgmt_request": "{u'uuid': '5140f6fb-dca3-4801-8c44-89b293405310',
u'json_display': u'False', u'servers': u'f24-michigan-micro-1', u'locator':
u'False', u'command': u'create',
   u'action': u'replace-osd', u'osd': u'osd.9'}", "created_at": "2018-03-
09T15:07:10.731220+00:00", "updated_at": null, "action": "replace-osd",
"osdmgmt_result": ""
   }
   }
```

List Check OSD Operation

Verb	URI
GET	v1/osdmgmt/list/? action= {check-osds,replace-osd}

Example

CURI Command:

curl -k -u admin:<Password> \-H "Accept: application/json" \-X GET "https://<POD IP>:8445/v1/osdmgmt/list/?action=check-osds&role=all"

JSON Request

GET /v1/osdmgmt/list/?action=check-osds

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   '4efd0be8-a76c-4bc3-89ce-142de458d844': {'action': 'check-osds',
'created_at': '2018-03-08 21:31:01+00:00',
   'status': 'osdmgmt_running', 'uuid':
   '4efd0be8-a76c-4bc3-89ce-142de458d844'},
   '5fd4f9b5-786a-4a21-a70f-bffac70a3f3f': {'action': 'check-osds',
'created_at': '2018-03-08 21:11:13+00:00',
   'status': 'osdmgmt_completed', 'uuid':
   '5fd4f9b5-786a-4a21-a70f-bffac70a3f3f'},

   '9c64ee52-bed5-4b69-91a2-d589411dd223': {'action': 'check-osds',
'created_at': '2018-03-08 21:26:15+00:00',
   'status': 'osdmgmt_completed', 'uuid':
   '9c64ee52-bed5-4b69-91a2-d589411dd223'}
   }
   }
```

Show a Completed osdmgmt Operation

Verb	URI
GET	v1/osdmgmt/show/?uuid=<uuid>

Example

CURL Command:

curl -k -u admin:<Password> \-H "Accept: application/json" \-X GET "https://<POD IP>:8445/v1/osdmgmt/show/?uuid=POD IP"

JSON Request

```
   GET /v1/osdmgmt/show/?uuid=9c64ee52-bed5-4b69-91a2-d589411dd223
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   'action': 'check-osds',
   'created_at': '2018-03-08 21:26:15+00:00',
   'osdmgmt_request': "{u'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223',
u'json_display': u'False', u'servers': u'', u'locator': u'False', u'command':
u'create', u'action':
   u'check-osds', u'osd': u'None'}",
```

```
   'osdmgmt_result': '{"status": "PROCESSED", "message": ["{\'Overall_Status\': \'PASS\',
   \'Result\': { ommitted for doc }}]}', 'status': 'osdmgmt_completed',
'updated_at': '2018-03-08 21:27:16+00:00', 'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223'
   }
   }
```

Delete a Completed osdmgmt Operation

Verb	URI
DELETE	v1/osdmgmt/delete/?uuid=<uuid>

Example

CURL Command:

curl -k -u admin:<Password> \-H "Accept: application/json" \-X DELETE "https://<POD IP>:8445/v1/osdmgmt/delete/?uuid=<UUID>"

JSON Request

```
   DELETE /v1/osdmgmt/delete/?uuid=9c64ee52-bed5-4b69-91a2-d589411dd223
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   'error': 'None',
   'message': 'UUID 9c64ee52-bed5-4b69-91a2-d589411dd223 deleted from
database', 'status': 'deleted', 'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223'
   }
   }
```

Hardware Management Utility

REST wrapper to control the execution of our query information from the hardware validation utility.

Create a Validate Operation

Verb	URI
POST	/v1/hardwaremgmt/validate

Curl Command:

curl -k -X POST "https://[<br_api ip>]:8445/v1/hardwaremgmt/validate" \-H "Content-Type: application/json" \-u admin:<Rest api admin Password> \-d '{

"hwmgmt_request": {

"command": "create",

```
    "action": "validate",
    "hosts": "None",
    "file": "None",
    "feature_list": "all",
    "uuid": ""
   }
  }'
```

JSON Request

```
   POST /v1/hardwaremgmt/validate '{"hwmgmt_request": {"command": "create",
"action": "validate", "hosts": "None",
   "file": "None", "feature_list": "all", "uuid": ""}}'
```

feature_list is a comma separated list of valid features for the given POD

JSON Response

```
   201 Created
   Content-Type: application/json
   {
   'action': 'validate,'
   'created_at': '2018-03-08T22:01:22.195232+00:00',
   'hwmgmt_request': "{u'feature_list': u'all', u'command': u'create', u'file':
None, u'action': u'validate', u'hosts': None, u'uuid': '89e094d8-b246-4620-
afca- ba3529385cac'}",
   'hwmgmt_result': '',
   'status': 'not_run', 'updated_at': 'None'
   }
```

Create a Validate Operation for Failure

Verb	URI
GET	/v1/hardwaremgmt/resolve_failures

JSON Request

```
   POST /v1/hardwaremgmt/resolve_failures
   {
   "hwmgmt_request": { "command": "create", "action": "resolve-failures",
"hosts": "None",
   "file": "None", "feature_list": "all", "uuid": ""}
   }
```

feature_list is a comma separated list of valid features for the given POD

Curl Command:

curl -k -X GET "https://[br_api ip]:8445/v1/hardwaremgmt/resolve_failures" -u admin:<Rest api admin password> -H "Accept: application/json

JSON Response

```
   201 Created
   Content-Type: application/json
   {

   "status": "not_run",
   "created_at": "2018-03-09T15:47:36.503712+00:00",
   "hwmgmt_request": "{u'feature_list': u'all', u'command': u'create', u'file':
None, u'action': u'resolve-failures', u'hosts': None, u'uuid': '49dc1dc9-3170-
4f68-b152-0f99bd19f7b1'}",
   "updated_at": "",
   "action": "resolve-failures", "hwmgmt_result": ""
   }
```

List a Validate Operation

Verb	URI
GET	v1/hardwaremgmt/list

Curl Command:

curl -k -X GET "https://[br_api ip]:8445/v1/hardwaremgmt/list" \-u admin:<Rest api admin password> \-H "Accept: application/json"

JSON Request

```
   GET /v1/hardwaremgmt/list
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {'89e094d8-b246-4620-afca-ba3529385cac': {'action': 'validate',
'created_at': '2018-03-08 22:01:22+00:00',
   'feature_list': 'all',
   'status': 'hardwaremgmt_completed', 'uuid':
   '89e094d8-b246-4620-afca-ba3529385cac'},
   '9f70e872-a888-439a-8661-2d2f36a4f4b1': {'action': 'validate', 'created_at':
'2018-03-08 20:34:32+00:00',
   'feature_list': 'all',
   'status': 'hardwaremgmt_completed', 'uuid':
   9f70e872-a888-439a-8661-2d2f36a4f4b1'}
   }
```

Show a Completed hardwaremgmt Operation

Verb	URI
GET	/v1/hardwaremgmt /show /?uuid=<uuid>

Curl Command:

curl -k -X GET "https://[br_api ip]:8445/v1/hardwaremgmt/show" \-u admin:<Rest api admin password> \-H "Accept: application/json"Uuid :" <uuid>"

JSON Request

```
GET /v1/hardwaremgmt/show/?uuid=9f70e872-a888-439a-8661-2d2f36a4f4b
```

JSON Response

```
200 OK
Content-Type: application/json
{
'action': 'validate',
'created_at': '2018-03-08 20:34:32+00:00',
'feature_list': 'all',
'hwmgmt_request': "{u'feature_list': u'all', u'hosts': None, u'file': None,
u'action': u'validate', u'command': u'create', u'uuid': '9f70e872-a888-439a-
8661- 2d2f36a4f4b1'}",
'hwmgmt_result': '{"status": "PROCESSED", "message": "Validate of all
completed", "results": {"status": "PASS", "results": [{"status": "PASS",
"name": "CIMC Firmware Version
Check", "err": null}, {"status": "PASS", "name": "All Onboard LOM Ports
Check", "err":
null}, {"status": "PASS", "name": "PCIe Slot: HBA Status Check", "err":
null}, {"status":

"PASS", "name": "Server Power Status Check", "err": null}, {"status":
"PASS", "name": "NFV Config Check", "err": null}, {"status": "PASS", "name":
"Physical Drives Check", "err":
null}, {"status": "PASS", "name": "PCIe Slot(s) OptionROM Check", "err":
null}, {"status": "PASS", "name": "Intel Network Adapter Check", "err":
null}]}}', 'status': 'hardwaremgmt_completed', 'updated_at': '2018-03-08
20:38:02+00:00', 'uuid': '9f70e872-a888-439a-8661-2d2f36a4f4b1'
```

Delete a Completed hardwaremgmt Operation

Verb	URI
DELETE	/v1/hardwaremgmt/delete/?uuid=<uuid>

Curl Command:

curl -k -X DELETE "https://[br_api ip]:8445/v1/hardwaremgmt/delete/?uuid=<uuid>" \-u admin:<Rest api admin password> \-H "Accept: application/json"

JSON Request

```
DELETE /v1/hardwaremgmt/delete/?uuid=9f70e872-a888-439a-8661-2d2f36a4f4b1
```

JSON Response

```
200 OK
Content-Type: application/json
{
'error': 'None',
```

```
   'message': 'UUID 9f70e872-a888-439a-8661-2d2f36a4f4b1 deleted from
database', 'status': 'deleted', 'uuid': '9f70e872-a888-439a-8661-2d2f36a4f4b1'
   }
```

List Password Secrets

Command

```
   curl -i -X GET -H 'Content-Type: application/json' -u admin:**** -H
'Accept: application/json'
   --cacert /var/www/mercury/mercury-ca.crt
https://172.31.231.17:8445/v1/secrets
```

Response

```
   {'HEAT_KEYSTONE_PASSWORD': '***', 'CINDER_KEYSTONE_PASSWORD': '***'
. . }
```

Hardware Management Utility

- Create a Validate Operation
- Create a Validate Operation for Failure
- Create a Validate Operation
- Show a Completed hardwaremgmt Operation
- Delete a Completed hardwaremgmt Operation

REST wrapper to control the execution of or query information from the hardware validation utility.

Create a Validate Operation

Verb	URI
POST	/v1/hardwaremgmt/validate

JSON Request

```
   POST /v1/hardwaremgmt/validate '{"hwmgmt_request": {"command":
"create",
   "action": "validate", "hosts": "None", "file": "None",
   "feature_list": "all", "uuid": ""}}'
   Feature_list is a comma separated list of valid features for the
given
   pod.
```

Curl Command:

curl -k -X POST "https://[<br_api ip>]:8445/v1/hardwaremgmt/validate" \-H "Content-Type: application/json" \-u admin:<Rest api admin Password> \-d '{

"hwmgmt_request": {

"command": "create",

"action": "validate",

"hosts": "None",

"file": "None",

"feature_list": "all",

"uuid": ""

}

}'

JSON Response

```
   201 Created
   Content-Type: application/json
   {
   'action': 'validate',
   'created_at': '2018-03-08T22:01:22.195232+00:00',
   'hwmgmt_request': "{u'feature_list': u'all', u'command': u'create',
u'file': None, u'action': u'validate', u'hosts': None,
   u'uuid': '89e094d8-b246-4620-afca-ba3529385c'}", 'hwmgmt_result':
'',
   'status': 'not_run', 'updated_at': 'None'
   }
```

Create a Validate Operation for Failure

Verb	URI
GET	/v1/hardwaremgmt/resolve_failures

Curl Command:

curl -k -X GET "https://[br_api ip]:8445/v1/hardwaremgmt/resolve_failures" -u admin:<Rest api admin password> -H "Accept: application/json

JSON Request

```
POST /v1/hardwaremgmt/resolve_failures
{
"hwmgmt_request": { "command": "create",
"action": "resolve-failures", "hosts": "None",
"file": "None",
"feature_list": "all", "uuid": ""}
}
feature_list is a comma separated list of valid features for the
given POD
```

JSON Response

```
201 Created
Content-Type: application/json
{

"status": "not_run",
"created_at": "2018-03-09T15:47:36.503712+00:00",

"hwmgmt_request": "{u'feature_list': u'all', u'command':
u'create', u'file': None, u'action': u'resolve-failures',
u'hosts': None, u'uuid': '49dc1dc9-3170-4f68-b152-
0f99bd19f7b1'}",
"updated_at": "",
"action": "resolve-failures", "hwmgmt_result": ""
}
```

List Validate Operation

Verb	URI
GET	v1/hardwaremgmt/list

JSON Request

Curl Command:

curl -k -X GET "https://[br_api ip]:8445/v1/hardwaremgmt/list" \-u admin:<Rest api admin password> \-H "Accept: application/json"

JSON Request

```
GET /v1/hardwaremgmt/list
```

JSON Response

200 OK
Content-Type: application/json
{'89e094d8-b246-4620-afca-ba3529385cac': {'action': 'validate',
'created_at': '2018-03-08 22:
01:22+00:00',
'feature_list': 'all',
'status':
'hardwaremgmt_completed',
'uuid': '89e094d8-b246-4620
afca-ba3529385cac'},
'9f70e872-a888-439a-8661-2d2f36a4f4b1': {'action': 'validate',
'created_at': '2018-03-08 20:
34:32+00:00',
'feature_list': 'all',
'status':
'hardwaremgmt_completed',
'uuid':'9f70e872-a888-439a-8661-2d2f36a4f4b1'}
}

Show a Completed hardwaremgmt Operation

Verb	URI
GET	/v1/hardwaremgmt /show /?uuid=<uuid>

JSON Request

```
GET /v1/hardwaremgmt/show/?uuid=9f70e872-a888-439a-8661-2d2f36a4f4b
```

Curl Command:

curl -k -X GET "https://[br_api ip]:8445/v1/hardwaremgmt/show" \-u admin:<Rest api admin password> \-H "Accept: application/json"

JSON Response

```
200 OK
Content-Type: application/json
{'89e094d8-b246-4620-afca-ba3529385cac': {'action': 'validate',
'created_at': '2018-03-08 22:
01:22+00:00',
'feature_list': 'all', 'status':
'hardwaremgmt_completed',
'uuid': '89e094d8-b246-4620-
afca-ba3529385cac'},
'9f70e872-a888-439a-8661-2d2f36a4f4b1': {'action': 'validate',
'created_at': '2018-03-08 20:
34:32+00:00',
'feature_list': 'all', 'status':
'hardwaremgmt_completed',
'uuid':'9f70e872-a888-439a-8661-2d2f36a4f4b1'}
}
```

Delete a Completed hardwaremgmt Operation

```
   200 OK
   Content-Type: application/json
   {
   'action': 'validate',
   'created_at': '2018-03-08 20:34:32+00:00',
   'feature_list': 'all',

   'hwmgmt_request': "{u'feature_list': u'all', u'hosts': None,
u'file': None, u'action': u'validate', u'command': u'create', u'uuid':
   '9f70e872-a888-439a-8661-2d2f36a4f4b1'}",
   'hwmgmt_result':
   '{"status": "PROCESSED", "message": "Validate of all completed",
"results": {"status": "PASS", "results": [{"status": "PASS", "name":
"CIMC Firmware Version
   Check", "err": null}, {"status": "PASS", "name": "All Onboard LOM
Ports Check", "err":
   null}, {"status": "PASS", "name": "PCIe Slot: HBA Status Check",
"err": null}, {"status": "PASS", "name": "Server Power Status Check",
"err": null}, {"status":
   "PASS", "name": "NFV Config Check", "err": null}, {"status":
"PASS", "name": "Physical Drives Check", "err":
   null}, {"status": "PASS",
   "name": "PCIe Slot(s) OptionROM Check", "err": null}, {"status":
"PASS", "name": "Intel Network Adapter Check", "err": null}]}}',
   'status': 'hardwaremgmt_completed', 'updated_at': '2018-03-08
20:38:02+00:00', 'uuid': '9f70e872-a888-439a-8661-2d2f36a4f4b1'
```

Verb	URI
DELETE	/v1/hardwaremgmt/delete/?uuid=<uuid>

JSON Request

```
   DELETE /v1/hardwaremgmt/delete/?uuid=9f70e872-a888-439a-8661-
2d2f36a4f4b1
```

Curl Command:

curl -k -X DELETE "https://[br_api ip]:8445/v1/hardwaremgmt/delete/?uuid=<uuid>" \-u admin:<Rest api admin password> \-H "Accept: application/json"

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   'error': 'None',
   'message': 'UUID 9f70e872-a888-439a-8661-2d2f36a4f4b1 deleted from
database', 'status': 'deleted',
   'uuid': '9f70e872-a888-439a-8661-2d2f36a4f4b1'
   }
```

HCLTech

Disk and OSD Maintenance

Disk Maintenance information

REST wrapper to query information about RAID disks on Pod nodes. This returns the RAID disk information of all or a selection of RAID disks available in the Pod.

The disk management extension to the VIM REST API enables support for Disk Management actions

Create a Check Disk Operation

Resource URI

eVerb	URI
POST	/v1/diskmgmt/check_disks

Example

CURL Command:

curl -k -u admin:<Password> \-H "Accept: application/json" \-H "Content-Type:application/json" \-X POST "https://<Pod IP>:8445/v1/diskmgmt/check_disks" \-d '{"diskmgmt_request": {"command": "create","action": "check-disks","role": "control","locator": "False","json_display": "False","servers": "","uuid": ""}}'

JSON Request

```
POST /v1/diskmgmt/check_disks Accept: application/json
'{"diskmgmt_request": {"command": "create",
"action": "check-disks",
"role": "control",
"locator": "False", "json_display": "False", "servers": "",
"uuid": ""}}'
```

JSON Response

```
201 Created
Content-Type: application/json
{
'action': 'check-disks',
'created_at': '2018-03-08T02:03:18.170849+00:00',
'diskmgmt_request': "{u'uuid': '0729bdea-cc19-440f-8339-
ab21e76be84b',
u'json_display': u'False',
u'servers': u'', u'locator': u'False', u'role': u'control',
u'action': u'check-disks', u'command': u'create'}",
'diskmgmt_result': '', 'status': 'not_run', 'updated_at': 'None'
}
```

```
   201 Created
   Content-Type: application/json
   {
   "status": "not_run",
   "diskmgmt_request": "{u'uuid': 'cb353f41-6d25-4190-9386-
330e971603c9',
   u'json_display': u'False', u'servers': u'', u'locator': u'False',
u'role': u'control',
   u'action': u'replace-disks', u'command': u'create'}",
   "created_at": "2018-03-09T12:43:41.289531+00:00",
   "updated_at": "", "diskmgmt_result": "", "action": "replace-disks"}
```

Create a Replace Disk Operation

Verb	URI
POST	/v1/diskmgmt/replace_disks

Example

CURL Command:

curl -k -u admin:<Password> \-H "Accept: application/json" \-H "Content-Type: application/json" \-X POST "https://<Pod IP>:8445/v1/diskmgmt/replace_disks" \-d '{"diskmgmt_request": {"command": "create","action": "replace-disks","role": "control","locator": "False","json_display": "False","servers": "","uuid": ""}}'

JSON Request

```
   POST /v1/diskmgmt/replace_disks Accept: application/json
   '{"diskmgmt_request": {"command": "create",
   "action": "replace-disks", "role": "control",
   "locator": "False", "json_display": "False", "servers": "", "uuid":
""}}'
```

JSON Response

```
201 Created
Content-Type: application/json
{
 "status": "not_run",
 "diskmgmt_request": "{u'uuid': 'cb353f41-6d25-4190-9386-330e971603c9',
                      u'json_display': u'False',
                      u'servers': u'',
                      u'locator': u'False',
                      u'role': u'control',
                      u'action': u'replace-disks',
                      u'command': u'create'}",
```

"created_at": "2018-03-09T12:43:41.289531+00:00",
"updated_at": "",
"diskmgmt_result": "",
"action": "replace-disks"}

List Check Disk Operation

Verb	URI
GET	/v1/diskmgmt/list/?action={check-disks,replace-disks\&role={all,management,control,compute}

Example

Curl Command:

curl -k -u admin:<Rest Api Password> \-H "Accept: application/json" \-X GET "https://<br_api ip>:8445/v1/diskmgmt/list/?action=check-disks&role=all"

JSON Request

```
GET /v1/diskmgmt/list/?action=check-disks\&role=all
```

JSON Response

```
200 OK
Content-Type: application/json
{
'0be7a55a-37fe-43a1-a975-cbf93ac78893': {'action': 'check-disks',
'created_at': '2018-03-05 14:
45:45+00:00',
'role': 'compute', 'status':
'diskmgmt_completed',
'uuid': '0be7a55a-37fe-43a1-
a975-cbf93ac78893'},
'861d4d73-ffee-40bf-9348-13afc697ee3d': {'action': 'check-disks',
'created_at': '2018-03-05 14:
44:47+00:00',
'role': 'control', 'status':
'diskmgmt_completed',
'uuid': '861d4d73-ffee-40bf-
9348-13afc697ee3d'},
'cdfd18c1-6346-47a2-b0f5-661305b5d160': {'action': 'check-disks',
'created_at': '2018-03-05 14:
43:50+00:00',
'role': 'all',
'status': 'diskmgmt_completed',
'uuid': 'cdfd18c1-6346-47a2-b0f5-661305b5d160'}}
}
```

Show a Completed diskmgmt Operation

Verb	URI
GET	v1/diskmgmt/show/?uuid=<uuid>

Example

Curl Command:

JSON Request

```
GET /v1/diskmgmt/show/?uuid=d24036c6-4557-4c12-8695-a92f6f9315ed
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {'action': 'check-disks',
   'created_at': '2018-03-07 21:46:41+00:00',
   'diskmgmt_request': "{u'uuid': 'd24036c6-4557-4c12-8695-
a92f6f9315ed', u'json_display': False,
   u'servers': u'f24-michigan-micro-2', u'locator': False,
   u'role': u'compute', u'action': u'check-disks', u'command':
u'create'}",
   'diskmgmt_result': '{"status": "PROCESSED", "message":
["{\'Overall_Status\': \'PASS\',
   \'Result\': {\'fcfg_disks_results_list\': [],
\'spare_disks_results_list\': [],
   \'raid_results_list\': [{\'RAID level\': \'RAID1\', \'Disk Med\':
\'HDD\', \'server\':
   \'7.7.7.6\', \'RAID type\': \'HW\', \'host\': \'f24-michigan-micro-
2\', \'role\':

   \'block_storage control compute\', \'VD health\': \'Optl\', \'Num
VDs\': 1, \'Num PDs\': 8, \'RAID health\': \'Opt\'}],
   \'bad_disks_results_list\': [], \'rbld_disks_results_list\': [],
\'add_as_spares_disks_results_list\': []}}"]}',
   'role': 'compute',
   'status': 'diskmgmt_completed', 'updated_at': '2018-03-07
21:47:35+00:00',
   'uuid': 'd24036c6-4557-4c12-8695-a92f6f9315ed'
   }
```

Delete a Completed diskmgmt Operation

Verb	URI
DELETE	v1/diskmgmt/delete/?uuid=<uuid>

Example

JSON Request

```
   DELETE /v1/diskmgmt/delete/?uuid=d24036c6-4557-4c12-8695-
a92f6f9315ed
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   "status": "deleted",
   "message": "UUID d24036c6-4557-4c12-8695-a92f6f9315ed deleted from
database", "uuid": "d24036c6-4557-4c12-8695-a92f6f9315ed",
   "error": "None"
   }
```

OSD Maintenance Information

REST wrapper to query information about OSD on Pod storage nodes. This returns to the OSD status information of all or a selection of OSDs available in the Pod.

Create an OSD Disk Operation

Verb	URI
POST	/v1/osdmgmt/check_osds

Example

JSON Request

```
   POST /v1/osdmgmt/osdmgmt/check_osds '{"osdmgmt_request":
{"command": "create",
   "action": "check-osds",
   "locator": "False", "json_display": "False",
   "servers": "",
   "osd": "None",
   "uuid": ""}}'
```

JSON Response

```
   201 Created
   Content-Type: application/json
   {
   'action': 'check-osds',
   'created_at': '2018-03-08T21:26:15.329195+00:00',
   'osdmgmt_request': "{u'uuid': '9c64ee52-bed5-4b69-91a2-
d589411dd223', u'json_display': u'False', u'servers': u'', u'locator':
u'False', u'command': u'create', u'action':
   u'check-osds', u'osd': u'None'}", 'osdmgmt_result': '',
   'status': 'not_run', 'updated_at': 'None'
   }
```

Create a Replace OSD Operation

Verb	URI
POST	v1/osdmgmt/replace_osd

Example

JSON Request

```
   POST /v1/osdmgmt/replace_osd Accept: application/json
   '{"osdmgmt_request": {"command": "create",
   "action": "replace-osd",
   "locator": "False", "json_display": "False", "servers": "f24-
michigan-micro-1", "osd": "osd.9",
   "uuid": ""}}'
```

JSON Response

```
   201 Created
   Content-Type: application/json
   {
   "status": "not_run",
   "osdmgmt_request": "{u'uuid': '5140f6fb-dca3-4801-8c44-
89b293405310', u'json_display': u'False',
   u'servers': u'f24-michigan-micro-1', u'locator': u'False',
   u'command': u'create', u'action': u'replace-osd', u'osd':
u'osd.9'}",
   "created_at": "2018-03-09T15:07:10.731220+00:00",
   "updated_at": null, "action": "replace-osd", "osdmgmt_result":
""
   }
```

List Check OSD Operation

Verb	URI
GET	v1/osdmgmt/list/? action= {check-osds,replace-osd}

Example

JSON Request

```
   GET /v1/osdmgmt/list/?action=check-osds
```

JSON Response

```
200 OK
Content-Type: application/json
{
'4efd0be8-a76c-4bc3-89ce-142de458d844': {'action': 'check-osds',
'created_at': '2018-03-08 21:
31:01+00:00',
'status': 'osdmgmt_running', 'uuid': '4efd0be8-a76c-4bc3-
89ce-142de458d844'},
'5fd4f9b5-786a-4a21-a70f-bffac70a3f3f': {'action': 'check-osds',
'created_at': '2018-03-08 21:
11:13+00:00',
'status':
'osdmgmt_completed',
'uuid':'5fd4f9b5-786a-4a21-
a70f-bffac70a3f3f'},

'9c64ee52-bed5-4b69-91a2-d589411dd223': {'action': 'check-osds',
'created_at': '2018-03-08 21:
26:15+00:00',
'status':
'osdmgmt_completed',
'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223'}
}
}
```

Show a Completed osdmgmt Operation

Verb	URI
GET	v1/osdmgmt/show/?uuid=<uuid>

Example

JSON Request

```
GET /v1/osdmgmt/show/?uuid=9c64ee52-bed5-4b69-91a2-d589411dd223
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   'action': 'check-osds',
   'created_at': '2018-03-08 21:26:15+00:00',

   'osdmgmt_request': "
   {u'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223', u'json_display':
u'False',
   u'servers': u'', u'locator': u'False',
   u'command': u'create', u'action':
   u'check-osds', u'osd': u'None'}",
   'osdmgmt_result': '{"status": "PROCESSED", "message":
["{\'Overall_Status\': \'PASS\',
   \'Result\': { ommitted for doc }}]}', 'status':
'osdmgmt_completed', 'updated_at': '2018-03-08 21:27:16+00:00',
   'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223'
   }
   }
```

Delete a Completed osdmgmt Operation

Verb	URI
DELETE	v1/osdmgmt/delete/?uuid=<uuid>

Example

JSON Request

```
   DELETE /v1/osdmgmt/delete/?uuid=9c64ee52-bed5-4b69-91a2-
d589411dd223
```

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   'error': 'None',
   'message': 'UUID 9c64ee52-bed5-4b69-91a2-d589411dd223 deleted from
database', 'status': 'deleted',
   'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223'
   }

   }
```

Cloud Sanity

- Create a cloud-sanity Test
- List cloud-sanity Test Results

- List Specific cloud-sanity Test Results
- Show cloud-sanity Test Results
- Delete cloud-sanity Test Results

REST wrapper to run cloud-sanity test suites. The cloud-sanity extension to the VIM REST API enables support for managing cloud-sanity test actions.

Create a cloud-sanity Test

Verb	URI
Post	/v1/cloud-sanity/create

Example

Curl Command:

curl -i -X POST /-H "Accept: application/json" \-H "Content-Type: application/json" \-u admin:<Password>\cacert/<mercuryca.crt path>\ https://<Pod IP>:8445/v1/cloudsanity/create \\-d '{"cloudsanity_request": {"command": "create","action": "test","test_name": "cephmon","uuid": "<UUID of the node>"}}

JSON Request

```
POST /v1/cloudsanity/create Accept: application/json
'{"cloudsanity_request": {"command": "create",
"action": "test", "test_name": "cephmon", "uuid": ""}}'
test_name can be all, management, control, compute, cephmon, cephosd
```

JSON Response

```
201 Created
{
'cloudsanity_request': "{u'action': u'test', u'command': u'create',
u'uuid': '5dff1662-3d33-4901-808d-479927c01dde',
u'test_name': u'cephmon'}", 'cloudsanity_result': '',
'created_at': '2018-01-26T20:32:20.436445',
'status': 'not_run', 'test_name': 'cephmon', 'updated_at': ''
}
```

List cloud-sanity Test Results

Verb	URI
GET	/v1/cloud-sanity

JSON Request

```
GET /v1/cloudsanity
```

JSON Response

```
   200 OK
   { '0b91746f-90b4-4355-a748-727c2e5c59c5': {
'action': 'test',

   25 12:08:22',

   'cloudsanity_completed', 'management',
   4355-a748-727c2e5c59c5'},

   '5695cb31-39e4-4be2-9dee-09e7daffc2e7': { 'action':
'test', 01-25 12:03:06',
   'cloudsanity_completed',


   4be2-9dee-09e7daffc2e7'},

   '5dff1662-3d33-4901-808d-479927c01dde': { 'action':
'test',

   26 20:32:20',

   'cloudsanity_completed',


   4901-808d-479927c01dde'},

   '7946255d-df58-4432-b729-20cf16eb5ba5': { 'action':
'test',

   25 12:05:56',

   'cloudsanity_completed',
   4432-b729-20cf16eb5ba5'},
   '797d79ba-9ee0-4e11-9d9e-47791dd05e07': { 'action':
'test',
   'created_at': '2018-01-25 12:05:11',
'created_at': '2018-01-
'status':
'test_name':
'uuid': '0b91746f-90b4-
'created_at': '2018-
'status':
'test_name': 'compute',
'uuid': '5695cb31-39e4-
'created_at': '2018-01-
'status':
'test_name': 'cephmon',
'uuid': '5dff1662-3d33-
'created_at': '2018-01-
'status':
'test_name': 'cephosd',
'uuid': '7946255d-df58-
'status':
```

```
'cloudsanity_completed',

4e11-9d9e-47791dd05e07'},
'962e2c8e-c7b0-4e24-87c1-528cad84002c': { 'action': 'test',

26 18:52:31',

'cloudsanity_completed',

4e24-87c1-528cad84002c'},

'd0111530-ee3b-45df-994c-a0917fd18e11': { 'action': 'test',

26 18:46:23',

'cloudsanity_completed',


45df-994c-a0917fd18e11'}}
'test_name': 'cephmon', 'uuid': '797d79ba-9ee0-

'created_at': '2018-01- 'status':
'test_name': 'control', 'uuid': '962e2c8e-c7b0-

'created_at': '2018-01- 'status':
'test_name': 'control', 'uuid': 'd0111530-ee3b-
```

List Specific cloud-sanity Test Results

Verb	URI
GET	/v1/cloud-sanity/list/?test_name={all, management, control,compute,cephmon,cephosd}

CURL Command:

curl -i -X GET \-H "Accept: application/json" \-u admin:<Password> \--cacert /<mercury-ca.crt path>\https://<POD IP>:8445/v1/cloudsanity/list/?test_name=cephmon

JSON Request

```
GET /v1/cloudsanity/list/?test_name=cephmon Accept:
application/json
```

JSON Response

Show cloud-sanity Test Results

```
200 OK
{ '5dff1662-3d33-4901-808d-479927c01dde': { 'action': 'test',
'created_at': '2018-
01-26 20:32:20',
'status':
'cloudsanity_completed',
'test_name': 'cephmon', 'uuid': '5dff1662-3d33-
4901-808d-479927c01dde'},
'797d79ba-9ee0-4e11-9d9e-47791dd05e07': { 'action': 'test',
'created_at': '2018-01-
25 12:05:11',
'status':
'cloudsanity_completed',
'test_name': 'cephmon', 'uuid': '797d79ba-
9ee0-4e11-9d9e-47791dd05e07'}}
```

Verb	URI
GET	/v1/cloud-sanity/show/?uuid=<uuid>

CURL Command:

curl -X GET "https://<host>:<port>/v1/cloudsanity/show/?uuid=<uu id>" \-H "Accept: application/json" \-H "Authorization: Bearer <access_token>"

JSON Request

```
GET /v1/cloudsanity/show/?uuid=d0111530-ee3b-45df-994c-a0917fd18e11
```

JSON Response

```
   200 OK
   { 'action': 'test', 'cloudsanity_request':
   "{u'action': u'test', u'command': u'create',
   u'uuid': 'd0111530-ee3b-45df-994c-a0917fd18e11', u'test_name':
u'control'}",
   'cloudsanity_result':
   '{"status": "PROCESSED",
   "message": {"status": "Pass",
   "message": "[PASSED] Cloud Sanity Control Checks Passed",
"results": {"control": {"ping_all_controller_nodes": "PASSED",
   "check_rabbitmq_is_running": "PASSED",
"check_rabbitmq_cluster_status": "PASSED", "check_nova_service_list":
"PASSED", "ping_internal_vip": "PASSED",
"disk_maintenance_raid_health": "PASSED",
"check_mariadb_cluster_size": "PASSED", "disk_maintenance_vd_health":
"PASSED"}}}}',
   'created_at': '2018-01-26 18:46:23',
   'status': 'cloudsanity_completed', 'test_name': 'control',
   'updated_at': '2018-01-26 18:47:58',
   'uuid': 'd0111530-ee3b-45df-994c-a0917fd18e11'}
```

Delete cloud-sanity Test Results

Verb	URI
DELETE	/v1/cloud-sanity/delete/?uuid=<uuid>

Curl Command:

curl -X DELETE "https://<host>:<port>/v1/cloudsanity/delete/?uuid=<uu id>" \-H "Accept: application/json" \-u admin:<password>

```
   GET /v1/cloudsanity/delete/?uuid=444aa4c8-d2ba-4379-b035-
0f47c686d1c4
```

JSON Response

```
   200 OK
   {
   "status": "deleted",
   "message": "UUID 444aa4c8-d2ba-4379-b035-0f47c686d1c4 deleted from
database", "uuid": "444aa4c8-d2ba-4379-b035-0f47c686d1c4",
   "error": "None"
   }
```

Mandatory/Optional Feature Mapping

- Mandatory Feature Mapping
- Optional Feature Mapping

Mandatory Feature Mapping

```
POST Request URL
/v1/releasemapping/mandatory_features_mapping
```

CURL Command:

curl -i -X GET \-H "Accept: application/json" \-u admin:Password \--cacert /<mercury-ca.crt path>\https://<POD IP>:8445/v1/releasemapping/mandatory_features_mapping

JSON Response:

```
{
"mandatory": { "networkType": { "C": {
                                    "feature_status": true,
                                    "values": [{"name": "VXLAN/Linux Bridge", "value":
"B": {                            "VXLAN/Linux Bridge"},], "insight_label": "Tenant Network",
                                    "desc": "Tenant Network"
},                                  },

                                    "feature_status": true,
                                    "values": [{"name": "VXLAN/Linux Bridge", "value":
                                  "VXLAN/Linux Bridge"},], "insight_label": "Tenant Network",
                                    "desc": "Tenant Network"
                                    }
"cephMode": {
"all": {
"feature_status": true,
"values": [{"name": "Central", "value": "Central"},], "insight_label": "Ceph
Mode",
"desc": "Ceph Mode"
}
                                                },
"podType": {
"C": {
                                    "feature_status": true,
"B": {                              "values": [{"name": "Fullon", "value": "fullon"},],
                                  "insight_label": "POD Type",
},                                  "desc": "POD Type"
                                    },

                                    "feature_status": true,
                                    "values": [{"name": "Fullon", "value": "fullon"},],
                                  "insight_label": "POD Type",
                                    "desc": "POD Type"
                                    }
```

```
   "installMode": {
   "all": {
   "feature_status": true,
   "values": [{"name": "Connected", "value": "connected"}, ], "insight_label":
"Install Mode",
   "desc": "Install Mode"
   }
   }
   },
   "platformType": [{"name": "B-series", "value": "B"}, {"name": "C-series",
"value": "C"}],
   "postinstalllinks": {
   "all",


   "platformtype": Passwords"}
   }

   "view_cloudpulse": {"alwayson": true, "feature_status": true,
"platformtype":

   "insight_label": "Run VMTP", "desc": "Cloudpluse"}, "password_reconfigure":
{"alwayson": true, "feature_status": true,

   "all", "insight_label": "Reconfigure Passwords", "desc": "Reconfigure

   }
```


Optional Feature Mapping

CURL Command:

curl -i -X GET \-H "Accept: application/json" \-u admin:Password \--cacert /<mercury-ca.crt path>\https://<POD IP>:8445/v1/releasemapping/optional_features_mapping

POST Request URL

```
/v1/releasemapping/optional_features_mapping
```

JSON Response:


```
[
{
"heat": {
"feature_status": true, "insight_label": "Heat",
"repeated_redeployment": false, "reconfigurable": ["all"], "desc":
"Openstack HEAT service"
}
},
..... other features
]
```

HCLTech

Testing and Polling

- NFVBench Network Performance Testing
- Create NFVBench Run
- Status Polling
- Get Fixed Rate Test Result
- Execute NDR/PDR Test
- Get NDR/PDR Test Results

NFVBench Network Performance Testingg

Create NFVBench Run

Starts the network performance test with provided configuration.

REST API to create fixed rate test

Verb	URI
Post	v1/nfvbench/ create_ndr_pdr_test

Example

CURL Command:

curl -k -u admin:<Password> \-H "Accept: application/json" \-H "Content-Type: application/json" \-X POST "https://<Pod IP>:8445/

POST Request URL /v1/nfvbench/create_fixed_rate_test

JSON Request

POST Request URL

```
/v1/nfvbench/create_fixed_rate_test JSON Request:
{"nfvbench_request":
{
"duration_sec": 20, "traffic_profile": [
{
"name": "custom",
"l2frame_size": ["64","IMIX", "1518"]
}
],
"traffic": { "bidirectional": true,
"profile": "custom"
},
"flow_count": 1000
}
}
```

JSON Response

```
   201 CREATED
   Content-Type: application/json
   {
   "status": "not_run", "nfvbench_request":
   '{
   "duration_sec": 20, "traffic_profile": [
   {
   "name": "custom",
   "l2frame_size": ["64","IMIX","1518"]
   }
   ],
   "traffic": {"bidirectional": true,
   "profile": "custom"
   },
   "flow_count": 1000
   }',
   "created_at": "2017-08-16T06:14:54.219106",
   "updated_at": null, "nfvbench_result": "", "test_name":
"Fixed_Rate_Test"
   }
```

Status Polling

The polling status of NFVbench status can be nfvbench_running, nfvbench_failed, or nfvbench_completed.

CURL COMMAND:

curl -k -u admin:<Password> \-H "Accept: application/json" \-H "Content-Type: application/json" \-X GET "https://<Pod IP>:8445/v1/nfvbench/Fixed_Rate_Test"

Resource URI

Verb	URI
GET	v1/nfvbench/<test_name>

Get Fixed Rate Test Result

CURL Commoand:

curl -i -X GET -H "Content-Type: application/json" -H "Accept: application/json" -u admin:<Password> --cacert /<mercury-ca.crt path>"https://<PodIP>:8445/v1/nfvbench/get_ndr_pdr_test_result"

GET Request URL

```
   /v1/upgrade/get_fixed_rate_test_result JSON Response:
   Check If NFVbench Test is running
   200 OK
   Content-Type: application/json
   {
   "status": "nfvbench_running",
   "nfvbench_request": '{"traffic": {"bidirectional": true, "profile":
"custom"}, "rate": "1000000pps",
   "traffic_profile": [{"l2frame_size": ["1518"], "name": "custom"}],
"duration_sec": 60, "flow_count": 1000}', "nfvbench_result": ""
   "created_at": "2017-05-30T21:40:40.394274", "updated_at": "2017-05-
30T21:40:41.367279",
   }
```

Check If NFVbench Test is completed

```
   200 OK
   Content-Type: application/json
   {
   "status": "nfvbench_completed",
   "nfvbench_request": '{"traffic": {"bidirectional": true, "profile":
"custom"}, "rate": "1000000pps", "traffic_profile": [{"l2frame_size":
["1518"], "name": "custom"}], "duration_sec": 60, "flow_count":
1000}', "nfvbench_result": '{"status": "PROCESSED", "message":
{"date": "2017-08-15 23:15:04", "nfvbench_version":
   "0.9.3.dev2",    }
   "created_at": "2017-05-30T21:40:40.394274", "updated_at": "2017-05-
30T22:29:56.970779",
   }
```

CURL Command:

curl -k -X POST "https://<Pod IP>:8445/v1/nfvbench/create_ndr_pdr_test" -H "Accept: application/json" -H "Content-Type: application/json" -u admin:<Password> -d @/tmp/req_NDR_PDR_Test.json

Execute NDR/PDR Test

```
POST Request URL
/v1/nfvbench/create_ndr_pdr_test

Accept: application/json
{"nfvbench_request":
{
"duration_sec": 20, "traffic_profile": [
{
"name": "custom",
"l2frame_size": ["64","IMIX","1518"]
}
],
"traffic": {"bidirectional": true, "profile": "custom"},
"flow_count": 1000}}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
"status": "not_run", "nfvbench_request":
'{
"duration_sec": 20,
"traffic_profile": [{"name": "custom", "l2frame_size":
["64","IMIX","1518"]}],
"traffic": {"bidirectional": true, "profile": "custom"},
"flow_count": 1000}' "created_at": "2017-08-16T07:18:41.652891",
"updated_at": null, "nfvbench_result": "", "test_name":
"NDR_PDR_Test"
}
```

Get NDR/PDR Test Results

GET Request URL

```
/v1/ nfvbench/get_ndr_pdr_test_result JSON Response:
```

If NFVbench NDR/PDR test is running

```
200 OK
Content-Type: application/json
{
"status": "nfvbench_running", "nfvbench_request": '{"duration_sec":
20,
"traffic": {"bidirectional": true, "profile": "custom"},
"traffic_profile": [{"l2frame_size": ["64", "IMIX", "1518"],
"name": "custom"}], "flow_count": 1000}',
"nfvbench_result": ""
"created_at": "2017-08-16T07:18:41.652891", "updated_at": "2017-09-
30T22:29:56.970779",
}
```

If NFVbench NDR/PDR test is completed

```
200 OK
Content-Type: application/json
{
"status": "nfvbench_completed", "nfvbench_request":
'{"duration_sec": 20,
"traffic": {"bidirectional": true, "profile": "custom"},
"traffic_profile": [{"l2frame_size": ["64", "IMIX", "1518"],
"name": "custom"}], "flow_count":1000}', "nfvbench_result":
'{"status": "PROCESSED",...}'
"created_at": "2017-08-16T07:18:41.652891", "updated_at": "2017-09-
30T22:29:56.970779",
}
```

HCLTech

Post-Installation Operations

- Create a Post Install Operation
- Retrieve Post Install Operation Status

The following are the post install operations that can be performed, after the successful installation of OpenStack. It uses a common api. Following is an Example:

1. reconfigure
2. reconfigure -regenerate passwords
3. reconfigure -setpasswords,setopenstack_configs
4. reconfigure -alertmanager_config, -alerting_rules_config
5. check-fernet-keys
6. resync-fernet-keys
7. rotate-fernet-keys

Create a Post Install Operation

Resource URI

Verb	URI
POST	/v1/misc

Examples:

JSON Request

```
POST /v1/misc
Accept: application/json
{"action": {"reconfigure": true}}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
"uuid": "7e30a671-bacf-4e3b-9a8f-5a1fd8a46733", "created_at":
"2017-03-19T14:03:39.723914",
"updated_at": null,
"operation_status": "OperationScheduled", "operation_logs": "",
"operation_name": "{"reconfigure": true}"
}
```

JSON Request

```
POST /v1/misc
Accept: application/json
{"action": {"reconfigure": true, "alertmanager_config":
<json_config>}}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
"uuid": "68b67265-8f09-480e-8608-b8aff77e0ec7", "created_at":
"2019-01-09T16:42:11.484604+00:00",
"updated_at": null,
"operation_status": "OperationScheduled", "operation_logs": "",
"operation_name": "{"alertmanager_config": <json_config>,
"reconfigure": true}"
}
```

Retrieve Post Install Operation Status

Resource URI

Verb	URI
GET	/v1/misc

Example

JSON Request

```
GET /v1/misc
Accept: application/json
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
"uuid":
"7e30a671-bacf-4e3b-9a8f-5a1fd8a46733", "created_at": "2017-03-
19T14:03:39.723914", "updated_at": "2017-03-19T14:03:42.181180",
"operation_status": "OperationRunning", "operation_logs":
"xxxxxxxxxxxxxxxx", "operation_name": "{\"reconfigure\": true}"
}
```

HCLTech

Version and Hardware Information

Version

Retrieve the version of the VIM.

Resource URI

Verb	URI
GET	/v1/version

Example

CURL Command:

curl -k -X GET "https://<Pod IP>:8445/v1/version" -H "Accept: application/json" -u admin:<Password>

JSON Request

```
GET /v1/version
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{"version": "1.9.1"}
```

Hardware Information

REST Wrapper returns the hardware information available in the setupdata.

Create a HWinfo Operation

Resource URI

Verb	URI
GET	/v1/hwinfo

Example

CURL Command:

curl -k -u admin:<Password> \ -H "Accept: application/json" \ -H "Content-Type: application/json" \-X POST "https://<Pod IP>:8445/v1/hwinfo" \-d '"setupdata": "<uuid of setup data>"}'

JSON Request

```
POST /v1/hwinfo
Accept: application/json
{
"setupdata":"c94d7973-2fcc-4cd1-832d-453d66e6b3bf"
}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
"status": "hwinfoscheduled",
"uuid": "928216dd-9828-407b-9739-8a7162bd0676",
"setupdata": "c94d7973-2fcc-4cd1-832d-453d66e6b3bf", "created_at":
"2017-03-19T13:41:25.488524", "updated_at": null, "hwinforesult": ""
}
```

Retrieve Hwinfo Operation Results

Resource URI

Verb	URI
GET	/v1/hwinfo/{id}

Property:

id—The ID of the node you want to query.

Example

CURL Cmmmand:

curl -k -X GET "https://<Pod IP>:8445/v1/hwinfo/789" -H "Accept: application/json" -u admin:<Password>

JSON Request

```
GET /v1/hwinfo/789 Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
"status": "hwinfosuccess",
"uuid": "928216dd-9828-407b-9739-8a7162bd0676",

"setupdata": "c94d7973-2fcc-4cd1-832d-453d66e6b3bf", "created_at":
"2017-03-19T13:41:25.488524", "updated_at":
"2017-03-19T13:42:05.087491",
"hwinforesult": "{\"172.29.172.73\": {\"firmware\": ............
............
.................
}
```

Get Node Hardware Information

Rest API helps you to get the hardware information of all the nodes in the pod through CIMC/UCSM.

- Total Memory
- Firmware Info (Model, Serial Number)
- CIMC IP

GET Request URL

```
/v1/hwinfo Output Response
{
"hwinforesult": "{"control-server-2": {"memory": {"total_memory":
"131072"}, "firmware": {"serial_number": "FCH1905V16Q", "fw_model":
"UCSC-C220-M4S"}, "cimc_ip": "172.31.230.100", "storage":
{"num_storage": 4}, "cisco_vic_adapters": {"product_name": "UCS VIC
1225"},
"cpu": {"number_of_cores": "24"}, "power_supply": {"power_state":
"on"}}
...
}
```

OpenStack Setup

- Secrets
- Retrieve the List of Secrets that are Associated with the OpenStack Setup
- OpenStack Configuration
- Retrieve the list of Configurations Associated with the OpenStack Setup
- Release Mapping Information

Secrets

Retrieve the List of Secrets that are Associated with the OpenStack Setup

You can retrieve the set of secret passwords that are associated with the OpenStack setup using the preceding api. This gives the list of secrets for each service in OpenStack.

Resource URI

Verb	URI
GET	/v1/secrets

Example

CURL Command:

curl -i -X GET \-H "Accept: application/json" \-u admin:<Password> \--cacert /<mercury-ca.crt path> \"https://<Pod Ip:8445>/v1/secrets"

JSON Request

```
GET /v1/secrets
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
"HEAT_KEYSTONE_PASSWORD": "xxxx", "CINDER_KEYSTONE_PASSWORD":
"xxxxx",
....
....
"RABBITMQ_PASSWORD": "xxxxx"
}
```

OpenStack Configuration

Retrieve the List of Configurations Associated with the OpenStack Setup

You can retrieve the set of OpenStack configurations associated with the OpenStack setup using the preceding api. This gives the current settings of different configurations such as verbose logging and debug logging for different OpenStack services.

Verb	URI
GET	/v1/secrets

JSON Request

```
GET /v1/openstack_config Accept: application/json
```

CURL Command

curl -i -X GET \-H "Content-Type: application/json" \-H "Accept: application/json" \-u admin:<Password>\cacert/<mercuryca.crtpath>\https://<PodIP>:8445/v1/openstack_config

JSON Response

```
200 OK
Content-Type: application/json
{
"CINDER_DEBUG_LOGGING": false, "KEYSTONE_DEBUG_LOGGING": false,
....
....
"NOVA_VERBOSE_LOGGING": true
}
```

Release Mapping Information

This api is used to see the list of features included and list of options which can be reconfigured in the Openstack Setup.

Retrieve the Release Mapping information

Resource URI

Verb	URI
GET	/v1/releasemapping

CURL Command:

curl -i -X GET \-H "Accept: application/json" \-u admin:<Password> \--cacert /<mercury-ca.crt path>\https://<Pod IP>:8445/v1/releasemapping

JSON Request

```
GET /v1/releasemapping Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json [
{
"SWIFTSTACK": {
"feature_status": true,
],
"desc": "swift stack feature"
}
},.....
.............
}
```

HCLTech

Update

- Start an Update Process
- Roll Back an Update
- Commit an Update
- Retrieve the Details of an Update

Note: HVIM 6.0 release not supporting Update hence we are not updating the documentation

Start an Update Process

Resource URI

Verb	URI
POST	/v1/update

Parameters:

- fileupload - tar file to be uploaded.
- filename - Name of the uploaded file.

Example

JSON Request

```
  curl -sS -X POST --form "fileupload=@Test/installer.good.tgz" --
form "filename=installer.good.tgz" https://10.10.10.8445/v1/update
```

⚠ This curl request is done as a form request.

JSON Response

```
200 OK
Content-Type: application/json
{
"update_logs": "logurl", "update_status": "UpdateSuccess",
"update_filename": "installer-4579.tgz", "created_at": "2016-07-
10T18:33:52.698656", "updated_at": "2016-07-10T18:54:56.885083"
}
409 CONFLICT
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client" "faultstring": "Uploaded
file is not in tar format"
}
```

Roll Back an Update

Resource URI

Verb	URI
PUT	/v1/update

Example

JSON Request

```
PUT /v1/update
Accept: application/json
{
"action":"rollback"
}
```

JSON Response

```
200 OK
Content-Type: application/json
{

"update_logs": "logurl", "update_status": "ToRollback",
"update_filename": "installer-4579.tgz", "created_at": "2016-07-
10T18:33:52.698656", "updated_at": "2016-07-10T18:54:56.885083"
}
```

Commit an Update

Resource URI

Verb	URI
PUT	/v1/update

Example: JSON Request

```
PUT /v1/update
Accept: application/json
{
"action":"commit"
}
```

JSON Response

```
200 OK
Content-Type: application/json
{
"update_logs": "logurl", "update_status": "UpdateSuccess",
"update_filename": "installer-4579.tgz",
"created_at": "2016-07-10T18:33:52.698656", "updated_at": "2016-07-
10T18:54:56.885083"
}
```

Retrieve the Details of an Update

Resource URI

Verb	URI
GET	/v1/update

Example

JSON Request

```
GET /v1/update
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "update_logs": "logurl",
    "update_status": "UpdateSuccess",
    "update_filename": "installer-4579.tgz",
  "created_at": "2016-07-10T18:33:52.698656",
    "updated_at": "2016-07-10T18:54:56.885083"
}
```

Install Resource

Retrieve Information About a Particular Node

Resource URI

Verb	URI
GET	/v1/nodes{id}

Property:

id—The ID of the node that you want to retrieve.

Example

JSON Request

```
POST /v1/nodes
Accept: application/js
```

JSON Response

```
200 OK
Content-Type: application/json
{
"status": "Active",
"uuid": "456",
"setupdata": "123",
"node_data": "{
"rack_info": { "rack_id": "RackA"
},
"cimc_info": { "cimc_ip": "10.10.10.10"
},
"management_ip": "7.7.7.10"
}",
"updated_at": null, "mtype": "compute",
"install": "345", "install_logs": "logurl",
"created_at":"2016-0710T06:17:03.761152",
"name": " compute-1"
}
404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Node doesn't exists"
}
```

HCLTech

Install Resource

REST wrapper for install provides methods for starting, stopping, and viewing the status of the installation process.

Return a List of Installation

Resource URI

Verb	URI
GET	/v1/install

Example

JSON Request

```
GET /v1/install
Accept: application/json
```

curl -g -i -X GET -H 'Content-Type: application/json' -u admin:<Rest API admin user password> -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert "<mercury-ca.crt file path>" https://<br_api>:8445/v1/install

JSON Response

```
200 OK
Content-Type: application/json
{"installs": [{
"ceph": "Skipped",
"uuid": "123",
"setupdata": "345",
"vmtpresult": "{
"status": "PASS",
"EXT_NET": []
}",
"baremetal": "Success", "orchestration": "Success",
"validationstatus": "{ "status": "PASS", "Software_Validation": [],
"Hardware_Validation": []
}",
"currentstatus": "Completed", "validation": "Success", "hostsetup":
"Success", "vmtp": "Skipped"
}]
}
```

HCLTech

Create an Installation

Resource URI

Verb	URI
POST	/v1/install

Fresh install HVIM in the pod using below process

1. Upload setupdata file (payload.json) before installation start, take setupdata json file from current installation

2. Install HVIM in 8 steps process

curl -i -X POST -H 'Content-Type: application/json' -H 'Accept: application/json' -H 'CVIM-API-Version: 4.0.0' -H 'User-Agent: python-ciscovimclient' -u admin:<Rest API admin password> --cacert "mercury-ca.crt file path" -d @payload.json https://<br_api>:8445/setupdata

curl -i -X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "CVIM-API-Version: 4.0.0" -H "User-Agent: python-ciscovimclient" -u admin:<Rest API admin password> --cacert "<mercury-ca.crt certificate path>" -d '{"setupdata": "<setupdata UUID taken from above POST method>", "stages": ["validation", "bootstrap", "runtimevalidation", "baremetal", "orchestration", "hostsetup", "ceph", "vmtp"]}' https://<br_api>:8445/v1/install

Example

JSON Request

```
   201 CREATED
   Content-Type: application/json
   {
   "ceph": "Skipped",
   "uuid": "123",
   "setupdata": "345",
   "vmtpresult": "{
   "status": "PASS",
   "EXT_NET": []
   }",
   "baremetal": "Success", "orchestration": "Success",
"validationstatus": "{ "status": "PASS", "Software_Validation": [],
"Hardware_Validation": []
   }",
   "currentstatus": "Completed", "validation": "Success", "hostsetup":
"Success", "vmtp": "Skipped"
   }
   409 CONFLICT
   Content-Type: application/json
   {
   "debuginfo": null "faultcode": "Client"
   "faultstring": "Install already exists"
   }
```

JSON Response

```
   GET /v1/install Accept: application/js
   {
   "setupdata": "123", "stages": [ "validation", "bootstrap",
"runtimevalidation", "baremetal", "orchestration", "hostsetup",
   "ceph", "vmtp"
   ]
   }
```

Retrieve the Installation

Resource URI

Verb	URI
GET	/v1/install/{id}

Property:

id—The ID of the installation that you want to retrieve.

Example

JSON Request

```
   GET /v1/install/345
   Accept: application/js
```

curl -i -X GET -H "Content-Type: application/json" -u admin:<Rest API admin password> -H "Accept: application/json" --cacert "<mercury-ca.crt file path>" https://<br_api>:8445/v1/install/<UUID of the install GET request>

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   "ceph": "Skipped",
   "uuid": "123",
   "setupdata": "345",
   "vmtpresult": "{
   "status": "PASS",
   "EXT_NET": []
   }",
   "baremetal": "Success", "orchestration": "Success",
"validationstatus": "{ "status": "PASS", "Software_Validation": [],
"Hardware_Validation": []
   }",
   "currentstatus": "Completed", "validation": "Success", "hostsetup":
"Success", "vmtp": "Skipped"
   }
   404 NOT FOUND
   Content-Type: application/json
   {
   "debuginfo": null "faultcode": "Client"
   "faultstring": "Install doesn't exists"
   }
```

Stop the Installation

Resource URI

Verb	URI
DELETE	/v1/install/{id}

Property:

id—The ID of the installation that you want to stop.

Example JSON Request

```
   DELETE /v1/install/345
   Accept: application/js
```

curl -i -X DELETE -H "Content-Type: application/json" -H "Accept: application/json" -H "User-Agent: python-ciscovimclient" -u admin:<Rest API admin password> --cacert "<mercury-ca.crt file path>" https://<br_api>:8445/v1/install/<install UUID>

JSON Response

```
   204 NO CONTENT
   Content-Type: application/json
   404 NOT FOUND
   Content-Type: application/json
   {
```

```
   "debuginfo": null "faultcode": "Client" "faultstring": "Install doesn't
exists"
   }
```

Nodes and Replace Controller

Nodes

Getting a List of Nodes

Resource URI

Verb	URI
GET	/v1/nodes

Example

JSON Request

```
Get /v1/nodes
Accept: application/js
```

curl -i -X GET -H 'Content-Type: application/json' -H 'Accept: application/json' -H 'CVIM-API-Version: 4.0.0' -H 'User-Agent: python-ciscovimclient' -u admin:<Rest API admin password> --cacert "<mercury-ca.crt path>" https://<br_api>:8445/nodes

JSON Response

```
200 OK
Content-Type: application/json
{
"nodes": [ [
"status": "Active",
"uuid": "456",
"setupdata": "123",
"node_data": "{
"rack_info": { "rack_id": "RackA"
},
"cimc_info": {
"cimc_ip": "10.10.10.10"
},
"management_ip": "7.7.7.10"
}",
"updated_at": null, "mtype": "compute",
"install": "345", "install_logs": "logurl",
"created_at":"2016-0710T06:17:03.761152",
"name": " compute-1"
}
]
}
```

Add New Nodes

The nodes are in compute or block_storage type. Before adding the nodes to the system, the name of the nodes and other necessary information like cimc_ip and rackid must be updated in the setupdata object. If the setupdata object is not updated, the post call does not allow you to add the node.

Verb	URI

Steps to add node into the cluster

1. Get setup json response from GET request, convert or beautify the json string, save it in payload.json file

2. Update modified setup_data.yaml file using PUT request

3. Send remove request using POST request

Resource URI

POST	/v1/nodes
PUT	setupdata/uuid

Example

JSON Request

```
POST /v1/nodes
Accept: application/js
{
"name" : "compute-5"
}
```

curl -i -X PUT -H "Content-Type: application/json" -H "Accept: application/json" -H "User-Agent: python-ciscovimclient" -u admin:<Rest API admin passwd> --cacert "mercury-ca.crt path" -d @payload.json https://<br_api>:8445/setupdata/<setup data uuid>

curl -i -X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "User-Agent: python-ciscovimclient" -u admin: --cacert <mercury-ca.crt> -d '{"name": "<Node name to be removed", "skip_vmtp": false}' https://:8445/nodes/add_compute

JSON Response

```
201 CREATED
Content-Type: application/json
{
"status": "ToAdd",
"uuid": "456",
"setupdata": "123",
"node_data": "{
"rack_info": { "rack_id": "RackA"
},
"cimc_info": { "cimc_ip": "10.10.10.10"
},
"management_ip": "7.7.7.10"
}",
"updated_at": null, "mtype": "compute",
"install": "345", "install_logs": "logurl",
"created_at":"2016-0710T06:17:03.761152",
"name": " compute-1"
}
```

Remove a Node

The node to be deleted must be removed from the setupdata object. Once the setupdata object is updated, you can safely delete the node. The node object cannot be deleted until it calls the remote node backend and succeeds.

Resource URI

Verb	URI
DELETE	/v1/nodes{id}

Property:

id—The ID of the node that you want to remove. Example

Follow below procedure to remove node from the cluster using rest api call

1. Get setupdata UUID from install API
2. Get setupdata with the collected setupdata uuid from above install API.
3. Get list of nodes already deployed in the pod
4. Update the modified setupdata
5. Delete node from the podcurl -g -i -X GET -H 'Content-Type: application/json' -u admin: -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert "<mercury-ca.crt file path>" https://:8445/install

curl -g -i -X GET -H 'Content-Type: application/json' -u admin: -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert "<mercury-ca.crt file path>" https://:8445/setupdata/

curl -g -i -X GET -H 'Content-Type: application/json' -u admin: -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert "<mercury-ca.crt file path>" https://:8445/nodes

curl -i -X PUT -H "Content-Type: application/json" -H "Accept: application/json" -H "User-Agent: python-ciscovimclient" -u admin: --cacert "<mercury-ca.crt file path>" -d @setup_data_comp1-remove.json https://:8445/setupdata/

JSON Request

```
DELETE /v1/nodes/456 Accept: application/js
```

curl -g -i -X DELETE -H 'Content-Type: application/json' -u admin: -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert "<mercury-ca.crt file path>" -d '{"force_op": false, "name": ""}' https://:8445/nodes/remove_compute

JSON Response

```
204 ACCEPTED
Content-Type: application/json
404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Node doesn't exists"
}
```

To clear the database and delete the entries in the nodes, the delete API is called with special parameters that are passed along with the delete request. The JSON parameters are in the following format.

JSON Request

```
DELETE /v1/nodes/456 Accept: application/js
{
"clear_db_entry":"True"\
}
```

JSON Response

```
204 ACCEPTED
Content-Type: application/json

404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Node doesn't exists"
}
```

Health of the Management Node

This API is used to retrieve the health of the management node. It checks various parameters such as partitions, space and so on. Resource URI

Verb	URI
GET	/v1/health

⚠ This is done only if the node is deleted from the REST API database. The failure reason of the node must be rectified manually apart from the API. True is a string and not a boolean in the preceding line.

Example

JSON Request

```
GET /v1/health
Accept: application/json
```

curl -i -X GET -H 'Content-Type: application/json' -H 'Accept: application/json' -H 'CVIM-API-Version: 4.0.0' -H 'User-Agent: python-ciscovimclient' -u admin:<Rest API admin password> --cacert "<mercury-ca.crt file path>" https://<br_api>:8445/v1/health

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   "status": "PASS", "pod_status": { "color": "BLUE",
"version": "<VERSION_NO.>"
   },
   "insight_version": "<VERSION_NO.>"
   }
```

Color signifies the health of the pod for Insight:

- Grey signifies that no installation is kicked off on the pod.
- Green signifies that everything is in Good state and cloud installation is active.
- Blue signifies that some operation is running on the pod.
- Red signifies that the pod is in critical state, and you might need TAC support to recover the pod.
- Amber indicates a warning if a pod management (Add/Remove/Replace) operation failed.

Replace a Controller

Resource URI

Verb	URI
PUT	/v1/nodes{id}

Property:

id—The ID of the controller that you want to replace.

Example

JSON Request

```
   PUT /v1/nodes/456 Accept: application/js
```

curl -i -X PUT -H "Content-Type: application/json" -H "Accept: application/json" -H "User-Agent: python-ciscovimclient" -u admin: --cacert "<mercury-ca.crt file path>" -d '{"name": "", "status": "ToReplace", "force_op": false, "skip_vmtp": false}' https://:8445/nodes/

JSON Response

```
   200 OK
   Content-Type: application/json
```

```
404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Node doesn't exists"
}
```

Setupdata and Offline Validation

Setupdata

REST wrapper for setupdata. Provides methods for listing, creating, modifying, and deleting setupdata.

Retrieving the Setupdata

Resource URI

Verb	URI
GET	/v1/setupdata

Example

Curl Command:

curl -g -i -X GET -H 'Content-Type: application/json' -u admin:<rest api password> -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert "C:\certificates\<mercury-ca.crt path>" https://<br_api ip>:8445/setupdata

JSON Request

```
GET /v1/setupdata
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{"setupdatas": [{
"status": "Active",
"name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}]}
```

Creating Setupdata

Resource URI

Verb	URI
POST	/v1/setupdata

Example

Curl Command:

curl -g -i -X POST -H "Content-Type: application/json" -u admin:<Rest Api password> -H "Accept: application/json" --cacert 'C:\certificates\<mercury-ca.crt path>' "name": "GG34", "uuid": "uuid", "meta": {"user": "root"} -d "{\"jsondata\": " https://<br_api ip>:8445/v1/offlinevalidation

JSON Request

```
POST /v1/setupdata Accept: application/json
{ "name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}
```

JSON Response

```
201 OK
Content-Type: application/json
{
"status": "Active",
"name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}

400 Bad Request
Content-Type: application/json
{
"debuginfo": null "faultcode":"Client" "faultstring": "Error"
}
409 CONFLICT
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client" "faultstring": "Error"
}
```

Retrieving a Single Setupdata

Resource URI

Verb	URI
GET	/v1/setupdata/(id)

Property:

id—The ID of the setupdata that you want to retrieve. Example

JSON Request

```
GET /v1/setupdata/123 Accept: application/json
```

Curl Command:

curl -g -i -X GET -H 'Content-Type: application/json' -u admin:<Rest Api Password> -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert 'C:\certificates\i11tb3\<mercury-ca.crt path>' https://<br_api ip>:8445/setupdata/<uuid>

JSON Response

```
200 OK
Content-Type: application/json
{
"status": "Active",
"name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}
404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Setupdata could not be found."
}
```

Updating a Setupdata

Resource URI

Verb	URI
PUT	/v1/setupdata/(id)

Property:

id—The ID of the setupdata that you want to update.

Example

JSON Request

```
PUT /v1/setupdata/123 Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
"status": "Active",
"name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}
404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Setupdata could not be found."
}
```

Deleting a Setupdata

Resource URI

Verb	URI
DELETE	/v1/setupdata/(id)

Property:

id—The ID of the setupdata that you want to delete. Example

JSON Request

```
DELETE /v1/setupdata/123 Accept: application/json
```

JSON Response

```
204 NO CONTENT Returned on success
404 NOT FOUND
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Setupdata could not be found."
}
400 BAD REQUEST
Content-Type: application/json
```

```
   {
   "debuginfo": null "faultcode": "Client"
   "faultstring": "Setupdata cannot be deleted when it is being used by an
installation"
   }
```

Offline Validation

REST wrapper does the offline validation of setupdata. Rest wrapper does only the software Validation of the input setupdata.

Create an Offline Validation Operation

Resource URI

Verb	URI
POST	/v1/offlinevalidation

Example

JSON Request

```
   POST /v1/offlinevalidation
   Accept: application/json
   {
   "jsondata": "."
```

curl -g -i -X POST -H "Content-Type: application/json" -u admin: -H "Accept: application/json" --cacert '' -d "{"jsondata": $(cat setup_data.json)}" https://:8445/v1/offlinevalidation

JSON Response

```
   201 CREATED
   Content-Type: application/json
   {
   "status": "NotValidated",
   "uuid": "bb42e4ba-c8b7-4a5c-98b3-1f384aae2b69", "created_at":
"2016-02-03T02:05:28.384274", "updated_at": "2016-02-
03T02:05:51.880785",
   "jsondata": "{}", "validationstatus": { "status": "PASS",
"Software_Validation": [], "Hardware_Validation": []
   }
   }
```

```
   201 CREATED
   Content-Type: application/json
   {
   "status": "NotValidated",
   "uuid": "bb42e4ba-c8b7-4a5c-98b3-1f384aae2b69", "created_at":
"2016-02-03T02:05:28.384274", "updated_at": "2016-02-
03T02:05:51.880785",
   "jsondata": "{}", "validationstatus": ""
```

Retrieve the Results of Offline Validation

Resource URI

Verb	URI
GET	/v1/offlinevalidation

Property:

id—The ID of the node you want to retrieve.

Example

JSON Request

```
   GET /v1/offlinevalidation/789 Accept: application/json
```

Curl Command:

curl -g -i -X GET -H 'Content-Type: application/json' -u admin:<Password> -H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert 'C:\certificates\<mercury-ca.crt path>' https://<br_api ip>:8445/v1/offlinevalidation/<uuid>

JSON Response

```
   200 OK
   Content-Type: application/json
   {
   "status": " ValidationSuccess",
   "uuid": "bb42e4ba-c8b7-4a5c-98b3-1f384aae2b69", "created_at": "2016-02-
03T02:05:28.384274", "updated_at": "2016-02-03T02:05:51.880785",
   "jsondata": "{}", "validationstatus": { "status": "PASS",
"Software_Validation": [], "Hardware_Validation": []
   }
   }
```

HCLTech

Copyrights

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE LICENSE AGREEMENT SIGNED BETWEEN THE PARTIES.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS” WITH ALL

FAULTS. HCLSOFTWARE AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL HCLSOFTWARE OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF HCLSOFTWARE OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

HCLSOFTWARE Logo is trademark of HCL Technologies Ltd., and/or its affiliates in the U.S. and other countries.

Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between HCLSOFTWARE and any other company.