Cisco VIM REST API

The following topics explain how to use the Cisco VIM REST API to manage Cisco NFVI.

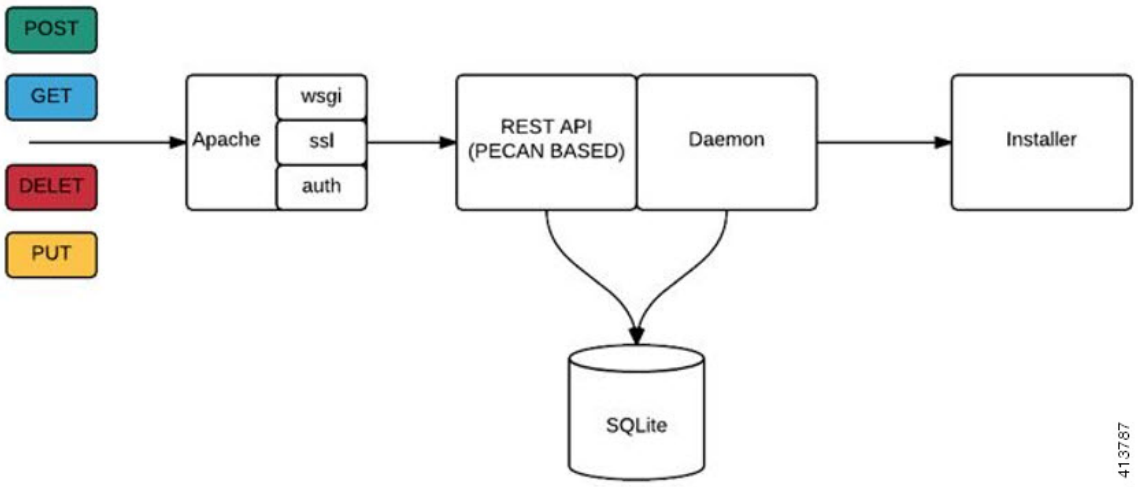
Overview to Cisco VIM REST API

Cisco VIM provides a Representational State Transfer (REST) API that you can use to install, expand, and update Cisco VIM. Actions you can perform using the RESTful API include:

- Install Cisco VIM on Cisco NFVI pods.
- Add and delete pods to and from Cisco NFVI installations.
- Update Cisco VIM software.
- Replace controller nodes.
- Perform cloud maintenance operations.
- Run cloud validations using Virtual Machine ThroughPut (VMTP), a data path performance measurement tool for OpenStack clouds.

The figure below shows the Cisco VIM REST API flow.

Figure 1: Cisco VIM REST API Flow



The Cisco VIM REST API security is provided by the Secure Sockets Layer (SSL) included on the Apache webserver. The Pecan-based web application is called by mod_wsgi, which runs the Rest API server. The Pecan REST API server requires a username and password to authorize REST API server requests. Apache handles the authorization process, which authorizes the request to access the Pecan web application. You can use the Cisco VIM API to upload a new setup_data.yaml file, and start, stop, and query the state of the installation. You can also use it to manage the cloud, add and remove compute and Ceph nodes, and replace the controller nodes. A REST API to launch VMTP (L2/L3 data plane testing) and CloudPulse is also provided.

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

The API end points can be reached with the following URL format:

https://<Management_node_api_ip>:8445

The API endpoint expects a basic authentication which is enabled by default in the management node. The authentication credentials can be found in /opt/cisco/ui_config.json in the management node. Sample ui_config.json contents are as shown below:

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

Cisco VIM REST API Resources

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
```

JSON Response

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

Creating the 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":{
    .......
  }
}
```

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 to be queried.

Example

JSON Request

```
GET /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."
}
```

Updating a setupdata

Resource URI

Verb	URI
PUT	/v1/setupdata/(id)

Property:

id - the id of the setupdata to be updated.

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 to be deleted.

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"
}
```

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": []
      }",
      "baremetal": "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",
       "baremetal",
       "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 install to be queried.

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 install to be stopped.

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

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 will not allow you to add the node.

Resource URI

Verb	URI
POST	/v1/nodes

Example

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 to be queried.

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"
}
```

Remove a node

The node that must be deleted must be removed from the setupdata object. Once the setupdata object is updated, you can safely delete of the node. The node object will not be deleted until it calls the remove node backend and succeeds.

Resource URI

Verb	URI
DELETE	/v1/nodes{id}

Property:

id - the id of the node to be removed.

Example

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"
}
```

For clearing the database and deleting 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"
}
```

Note 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 above line.

Replace a controller

Resource URI

Verb	URI
PUT	/v1/nodes{id}

Property:

id - the id of the controller to be replaced.

Example

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. This will only do S/W 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": ".. .. .."
}
```

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 to be queried.

Example

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
```

Note 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"
 }
```

Rollback 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 associated with the OpenStack Setup

You can retrieve the set of secret password associated with the OpenStack setup using the above api. This gives the list of secrets for each service in OpenStack.

Resource URI

Verb	URI
GET	/v1/secrets

Example

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 above api. This gives the current settings of different configs like verbose logging, debug logging for different OpenStack services.

Resource URI

Verb	URI
GET	/v1/openstack_config

Example

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 Cisco Virtualized Infrastructure Manager.

Resource URI

Verb	URI
GET	/v1/version

Example

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 can be used to retrieve the health of the management node. It checks various parameters like partitions, space and so on.

Resource URI

Verb	URI
GET	/v1/health

Example

JSON Request

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

JSON Response

```
200 OK
Content-Type: application/json
{
    "status": "PASS",
    "BuildNode Validation": {
        "Check Docker Pool Settings": {"status": "Pass", "reason": "None"}
        ….
        ….
    }
}
```

Hardware Information

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

Create a HWinfo operation

Resource URI

Verb	URI
GET	/v1/hwinfo

Example

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 to be queried.

Example

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

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 carried on once the OpenStack installation is carried out successfully. It uses a common api. So only one operation is given as an example below:

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

7 rotate-fernet-keys

Create a post install operation

Resource URI

Verb	URI
POST	/v1/misc

Example

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}"
}
```

Retrieve a status of the post install operation

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}"
}
```

In VIM 2.2, additional Rest APIs are introduced to support NFVBench, query hardware information and to get a list of optional and mandatory features that the pod supports.

Listed below are the details of the API.

NFVBench Network Performance Testing

Create NFVBench Run

Starts network performance test with provided configuration.

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
}
}
```

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 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",
   }
```

REST API to create 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"
}
```

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",

}
```

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

```
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

```
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
]
```

Disk Maintenance information

REST wrapper to query information about RAID disks on Pod nodes. This will return 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

End Point	Type	Valid Args	Valid Arg Values	Example
/diskmgmt	GET	None	None	/v1/diskmgmt/
/diskmgmt/ check_disks	GET	None, args	All, control, compute	/v1/diskmgmt/check_disks/ /v1/diskmgmt/check_disks/? args=control,compute
/diskmgmt/ replace_disks	GET	None, args	All, control, compute	/v1/diskmgmt/replace_disks/ /v1/diskmgmt/replace_disks/? args=control,compute
/diskmgmt/server	GET	server_list, action	List of valid server names, check_disks, replace_disks	/v1/diskmgmt/server/?server_list= srv1,srv2&action=check_disks /v1/diskmgmt/server/?server_list= srv1&action=replace_disks

Get a Check disk operation

Resource URI

Verb	URI
GET	/v1/diskmgmt

Example

JSON Request

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

JSON Response

```
200 OK
Content-Type: application/json
{
    "add_as_spares_disks_results_list": [],
    "bad_disks_results_list": [],
    "fcfg_disks_results_list": [],
    "raid_results_list": [
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "EMC-Testbed",
            "role": "management",
            "server": "localhost"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-20",
            "role": "control",
            "server": "15.0.0.7"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-21",
            "role": "control",
            "server": "15.0.0.8"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-22",
            "role": "control",
            "server": "15.0.0.5"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
```

```
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-23",
            "role": "compute",
            "server": "15.0.0.6"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-24",
            "role": "compute",
            "server": "15.0.0.10"
        }
    ],
    "rbld_disks_results_list": [],
    "spare_disks_results_list": []
}
```

Get a Check disk operation for compute nodes

Resource URI

Verb	URI
GET	/v1/diskmgmt/check_disks/?args={all,control,compute}

Example

JSON Request

```
GET /v1/diskmgmt/check_disks/?args=compute
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "add_as_spares_disks_results_list": [],
    "bad_disks_results_list": [],
    "fcfg_disks_results_list": [],
    "raid_results_list": [
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-23",
            "role": "compute",
            "server": "15.0.0.6"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-24",
            "role": "compute",
            "server": "15.0.0.10"
        }
    ],
    "rbld_disks_results_list": [],
```

```
    "spare_disks_results_list": []
}
```

Post a replace disk operation

Resource URI

Verb	URI
GET	/v1/diskmgmt/replace_disks/?args={all,control,compute}

Example

JSON Request

```
Get /v1/diskmgmt/replace_disks/?args=compute Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "add_as_spares_disks_results_list": [
        {
            "disk slot": "1",
            "host": "i13-21",
            "replace status": "Success",
            "role": "control",
            "server": "15.0.0.8"
        }
    ]
}
```

Get a check disk operation for a particular server

Resource URI

Verb	URI
GET	v1/diskmgmt/server/?server_list={server_list}&action=check_disks

Example

JSON Request

```
GET
/v1/diskmgmt/server/?server_list=i13-21,i13-23&action=check_disks
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "add_as_spares_disks_results_list": [
        {
            "disk slot": "1",
            "disk state": " UGood",
            "host": "i13-21",
            "role": "control",
            "server": "15.0.0.8"
        }
    ],
    "bad_disks_results_list": [],
    "fcfg_disks_results_list": [],
    "raid_results_list": [
        {
            "Num PDs": 4,
            "Num VDs": 1,
```

```
            "RAID health": "NdAtn",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Dgrd",
            "host": "i13-21",
            "role": "control",
            "server": "15.0.0.8"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-23",
            "role": "compute",
            "server": "15.0.0.6"
        }
    ],
    "rbld_disks_results_list": [],
    "spare_disks_results_list": []
}
```

Perform a replace disk operation for a particular server

Resource URI

Verb	URI
GET	v1/diskmgmt/server/?server_list={server_list}&action={replace_disks}

Example

JSON Request

```
GET
/v1/diskmgmt/server?server_list=i13-21&action=replace_disks
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "add_as_spares_disks_results_list": [
        {
            "disk slot": "1",
            "host": "i13-21",
            "replace status": "Success",
            "role": "control",
            "server": "15.0.0.8"
        }
    ]
}
```

OSD Maintenance information

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

End Point	Type	Valid Args	Valid Arg Values	Example
/osdmgmt	GET	None	None	/v1/osdmgmt/
/osdmgmt/check_osds	GET	None	Detail	/v1/osdmgmt/check_osds/

End Point	Type	Valid Args	Valid Arg Values	Example
osdmgmt/server	GET	server_list, action	List of valid server names, check_osds, replace_osd, osd_name	/v1/osdmgmt/server/?server_list =svr1,svr2&action=check_osds /v1/osdmgmt/server/server_list=svr1&action =replace_osd&osd_name=osd_name

Get a OSD disk operation

Resource URI

Verb	URI
GET	/v1/osdmgmt

Example

JSON Request

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

JSON Response

```
200 OK
Content-Type: application/json
{
    "bad_osds_results_list": [],
    "osd_details_results_list": [
        {
            "All OSD status": "All Good",
            "Num OSDs": 5,
            "OSD_detail": [
                {
                    "OSD_id": 0,
                    "OSD_journal": "/dev/sda4",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-0",
                    "OSD_name": "osd.0",
                    "OSD_path": "/dev/sdb1",
                    "OSD_status": "up",
                    "slot_id": 2
                },
                {
                    "OSD_id": 3,
                    "OSD_journal": "/dev/sda5",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-3",
                    "OSD_name": "osd.3",
                    "OSD_path": "/dev/sdc1",
                    "OSD_status": "up",
                    "slot_id": 3
                },
                {
                    "OSD_id": 6,
                    "OSD_journal": "/dev/sda6",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-6",
                    "OSD_name": "osd.6",
                    "OSD_path": "/dev/sdd1",
                    "OSD_status": "up",
                    "slot_id": 4
                },
                {
                    "OSD_id": 9,
```

```
                "OSD_journal": "/dev/sda7",
                "OSD_mount": "/var/lib/ceph/osd/ceph-9",
                "OSD_name": "osd.9",
                "OSD_path": "/dev/sde1",
                "OSD_status": "up",
                "slot_id": 5
            },
            {
                "OSD_id": 12,
                "OSD_journal": "/dev/sda8",
                "OSD_mount": "/var/lib/ceph/osd/ceph-12",
                "OSD_name": "osd.12",
                "OSD_path": "/dev/sdf1",
                "OSD_status": "up",
                "slot_id": 6
            }
        ],
        "host": "i13-27-test",
        "role": "block_storage",
        "server": "15.0.0.4"
    },
    {
        "All OSD status": "All Good",
        "Num OSDs": 5,
        "OSD_detail": [
            {
                "OSD_id": 1,
                "OSD_journal": "/dev/sda4",
                "OSD_mount": "/var/lib/ceph/osd/ceph-1",
                "OSD_name": "osd.1",
                "OSD_path": "/dev/sdb1",
                "OSD_status": "up",
                "slot_id": 2
            },
            {
                "OSD_id": 4,
                "OSD_journal": "/dev/sda5",
                "OSD_mount": "/var/lib/ceph/osd/ceph-4",
                "OSD_name": "osd.4",
                "OSD_path": "/dev/sdc1",
                "OSD_status": "up",
                "slot_id": 3
            },
            {
                "OSD_id": 7,
                "OSD_journal": "/dev/sda6",
                "OSD_mount": "/var/lib/ceph/osd/ceph-7",
                "OSD_name": "osd.7",
                "OSD_path": "/dev/sdd1",
                "OSD_status": "up",
                "slot_id": 4
            },
            {
                "OSD_id": 10,
                "OSD_journal": "/dev/sda7",
                "OSD_mount": "/var/lib/ceph/osd/ceph-10",
                "OSD_name": "osd.10",
                "OSD_path": "/dev/sde1",
                "OSD_status": "up",
                "slot_id": 5
            },
            {
                "OSD_id": 13,
                "OSD_journal": "/dev/sda8",
                "OSD_mount": "/var/lib/ceph/osd/ceph-13",
                "OSD_name": "osd.13",
                "OSD_path": "/dev/sdf1",
                "OSD_status": "up",
                "slot_id": 6
            }
        ],
        "host": "i13-25",
        "role": "block_storage",
```

```
            "server": "15.0.0.11"
        },
        {
            "All OSD status": "All Good",
            "Num OSDs": 5,
            "OSD_detail": [
                {
                    "OSD_id": 2,
                    "OSD_journal": "/dev/sda4",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-2",
                    "OSD_name": "osd.2",
                    "OSD_path": "/dev/sdb1",
                    "OSD_status": "up",
                    "slot_id": 2
                },
                {
                    "OSD_id": 5,
                    "OSD_journal": "/dev/sda5",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-5",
                    "OSD_name": "osd.5",
                    "OSD_path": "/dev/sdc1",
                    "OSD_status": "up",
                    "slot_id": 3
                },
                {
                    "OSD_id": 8,
                    "OSD_journal": "/dev/sda6",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-8",
                    "OSD_name": "osd.8",
                    "OSD_path": "/dev/sdd1",
                    "OSD_status": "up",
                    "slot_id": 4
                },
                {
                    "OSD_id": 11,
                    "OSD_journal": "/dev/sda7",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-11",
                    "OSD_name": "osd.11",
                    "OSD_path": "/dev/sde1",
                    "OSD_status": "up",
                    "slot_id": 5
                },
                {
                    "OSD_id": 14,
                    "OSD_journal": "/dev/sda8",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-14",
                    "OSD_name": "osd.14",
                    "OSD_path": "/dev/sdf1",
                    "OSD_status": "up",
                    "slot_id": 6
                }
            ],
            "host": "i13-26",
            "role": "block_storage",
            "server": "15.0.0.9"
        }
    ]
}
```


Perform a check OSD operation for a particular server

Resource URI

Verb	URI
GET	/v1/osdmgmt/server/?server_list={server_list}&action={check_osds}

Example

JSON Request

```
GET
/v1/diskmgmt/server/?server_list=i13-26&action=check_osds
Accept: application/json
```

JSON Response

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

{
    "bad_osds_results_list": [],
    "osd_details_results_list": [
        {
            "All OSD status": "All Good",
            "Num OSDs": 5,
            "OSD_detail": [
                {
                    "OSD_id": 2,
                    "OSD_journal": "/dev/sda4",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-2",
                    "OSD_name": "osd.2",
                    "OSD_path": "/dev/sdb1",
                    "OSD_status": "up",
                    "slot_id": 2
                },
                {
                    "OSD_id": 5,
                    "OSD_journal": "/dev/sda5",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-5",
                    "OSD_name": "osd.5",
                    "OSD_path": "/dev/sdc1",
                    "OSD_status": "up",
                    "slot_id": 3
                },
                {
                    "OSD_id": 8,
                    "OSD_journal": "/dev/sda6",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-8",
                    "OSD_name": "osd.8",
                    "OSD_path": "/dev/sdd1",
                    "OSD_status": "up",
                    "slot_id": 4
                },
                {
                    "OSD_id": 11,
                    "OSD_journal": "/dev/sda7",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-11",
                    "OSD_name": "osd.11",
                    "OSD_path": "/dev/sde1",
                    "OSD_status": "up",
                    "slot_id": 5
                },
                {
                    "OSD_id": 14,
                    "OSD_journal": "/dev/sda8",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-14",
                    "OSD_name": "osd.14",
                    "OSD_path": "/dev/sdf1",
                    "OSD_status": "up",
                    "slot_id": 6
                }
            ],
            "host": "i13-26",
            "role": "block_storage",
            "server": "15.0.0.9"
        }
    ]
}
```

Perform a replace OSD operation for a particular server

Resource URI

Verb	URI
GET	/v1/osdmgmt/server/?server_list={server_name}&action=replace_osd&osd_name={osd_name}

Example

JSON Request

```
GET
/v1/diskmgmt/server/?server_list=i13-25&action=replace_osd&osd_name=osd.10
Accept: application/json
```

JSON Response

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

{
'osd_replace_details_results_list': [
        {   'hdd_slot': 5,
                        'host_name': 'i13-25',
                        'journal_mnt': '/dev/sda4',
                        'new_dev_uuid': 'UUID=94480b3e-5698-4d6a-b715-0613be41cff5',
                        'new_mount': '/var/lib/ceph/osd/ceph-10',
                        'new_osd_id': 10,
                        'new_path': '/dev/sde1',
                        'old_osd_id': 10,
                        'status_msg': 'Successfully deleted, removed and replaced OSD
osd.10 from server i13-25'
                    }
              ]
}
```