Hewlett Packard Enterprise

HPE Helion Carrier Grade 4.0 Compute Resource Management Rest API

reference manual

Document Type

CRM Rest API

Helion Carrier Grade 4.0 Compute Resource Management Rest APIs

Logical Reference Architecture

Core Components of CRM:

- CRM Rest API: This service exposes North Bound APIs of CRM. These REST APIs can be invoked by clients like UI, CLI etc to interact with CRM system.
- CRM Queue Process: The POST requests to the CRM Rest API are put into a rabbitmq queue and CRM queue process picks these requests and processes them further.
- DB API Layer: This is a library that provides APIs to interact with CRM database and are used internally by CRM processes. The end user cannot access DBAPIs directly

- VIM API Layer: This is a library that provides APIs to interact with HOS Input Model and are used internally by CRM processes. The end user cannot access VIM APIs directly
- Compute Agent: The agent runs on each Compute and gathers detailed inventory and allocation information for CRM

JSON Template files

The following JSON example files contain the template details. The JSON files can be changed to conform to desired setup details. To fetch a specific template, it is necessary to use the API commands shown in this document. The template will be returned from the API call in JSON format. If it is necessary to alter the configuration templates it is advised to validate the template file to make sure it is still conforms to the JSON file standard (no missing commas, braces, or etc).

Example NIC Template JSON

NIC Template


```
{
    "name": "COMPUTE-INTERFACE-15-TEMP2",
    "porttonetworks": [
        {
            "name": "BONDED_INTERFACE",
            "networktypelist": [
                "PXE",
                "CLM"
            ],
            "propertylist": {},
            "externalnetlist": [],
            "port": {
                "devicelist": [
                    {
                        "busaddress": "0000:04:00.0",
                        "devicename": "hed1"
                    },
                    {
                        "busaddress": "0000:04:00.1",
                        "devicename": "hed2"
                    }
                ],
                "bondpropertylist": {
                    "options": "{'mode': 'active-backup', 'miimon': 200, 'primary':
'hed1'}",
                    "provider": "linux"
                },
                "name": "bond0"
            },
            "tullist": [],
            "forcednetworkgroups": [
                "BLS"
            ],
            "id": null,
            "fcoeinterfaces": [],
            "type": ""
        },
        {
            "name": "BONDED_DPDK_INTERFACE",
```

```
            "networktypelist": [],
            "propertylist": {},
            "externalnetlist": [],
            "port": {
                "devicelist": [
                        {
                           "busaddress": "0000:05:00.0",
                           "devicename": "dpdk0"
                        },
                        {
                           "busaddress": "0000:05:00.1",
                           "devicename": "dpdk1"
                        }
                ],
                "bondpropertylist": {
                    "options": "{'mode': 'active-backup', 'miimon': 200}",
                    "provider": "openvswitch"
                },
                "name": "bond1"

            },
            "tullist": ["VxLAN-VLAN1-TUL"],
            "forcednetworkgroups": [],
            "id": null,
            "fcoeinterfaces": [],
            "type": ""
        },
        {
            "name": "SRIOV_INTERFACE",
            "networktypelist": [],
            "propertylist": {
                "sriov-only": true,
                "vf-count": 10
            },
            "externalnetlist": [],
            "port": {
                "busaddress": "0000:82:00.0",
                "devicename": "hed5"
            },
            "tullist": ["VLAN2-TUL"],
            "forcednetworkgroups": [],
            "id": null,
            "fcoeinterfaces": [],
            "type": ""
    },
        {
            "name": "PCIPT_INTERFACE",
            "networktypelist": [],
            "propertylist": {
                "pci-pt": true
            },
            "externalnetlist": [],
            "port": {
                "busaddress": "0000:82:00.1",
                "devicename": "hed6"
            },
            "tullist": ["VLAN3-TUL"],
            "forcednetworkgroups": [],
            "id": null,
```

```
            "fcoeinterfaces": [],
            "type": ""
        }
    ],
   "templatetype": "NIC",
    "nicmappingid": 1,
    "dpdkdevices": {
        "component-options": [
            {
                "name": "n-dpdk-rxqs",
                "value": 48
            }
        ],
        "eal-options": [
            {
                "name": "socket-mem",
                "value": "2048,0"
            },
            {
                "name": "n",
                "value": 2
            }
        ],
        "devices": [
            {
                "driver": "igb_uio",
                "name": "dpdk0"
            },
                        {
                "driver": "igb_uio",
                "name": "dpdk1"
            }
        ],
        "components": [
            "openvswitch"
        ]
```

```
    },
    "id": null
}
```

Example Memory Template JSON

Memory Template JSON

```
{
    "name": "COMPUTE-MEMORY-15-TEMP1",
    "templatetype": "MEM",
    "defaultpagesize": "1G",
    "pages": [
        {
            "numanode": 1,
            "pagesize": "2M",
            "pagecount": 8192
        },
        {
            "numanode": 1,
            "pagesize": "1G",
            "pagecount": 20
        },
        {
            "numanode": 0,
            "pagesize": "2M",
            "pagecount": 8192
        },
        {
            "numanode": 0,
            "pagesize": "1G",
            "pagecount": 20
        },
        {
            "numanode": -1,
            "pagesize": "1G",
            "pagecount": 2
        },
        {
            "numanode": -1,
            "pagesize": "2M",
            "pagecount": 10
        }
    ]
}
```

Example Disk Template JSON

Disk Template JSON

```
{
  "name": "COMPUTE-DISKS-15-TEMP1",
  "volume-groups": [
    {
      "name": "hlm-vg",
      "physical-volumes": [
        "/dev/sda_root"
      ],
      "logical-volumes": [
        {
          "name": "root",
          "size": "35%",
          "fstype": "ext4",
          "mount": "/"
        },
        {
          "name": "log",
          "size": "50%",
          "mount": "/var/log",
          "fstype": "ext4",
          "mkfs-opts": "-O large_file"
        },
        {
          "name": "crash",
          "size": "10%",
          "mount": "/var/crashandburn",
          "fstype": "ext4",
          "mkfs-opts": "-O large_file"
        }
      ]
    },
    {
      "name": "vg-comp",
      "physical-volumes": [
        "/dev/sdb"
      ],
      "logical-volumes": [
        {
          "name": "compute",
          "size": "79%",
          "mount": "/var/lib/nova",
          "fstype": "ext4",
          "mkfs-opts": "-O large_file"
        }
      ]
    }
  ]
}
```

Example CPU Template JSON

CPU Template JSON

```
{
    "coreassignment": [
        {
      "numaid": 0,
            "platform": 1,
            "vswitch": 2,
   "vm": 6
        },
        {
      "numaid": 1,
            "platform": 1,
            "vswitch": 1,
   "vm": 6
        }
    ],
    "templatetype": "CPU",
    "name": "COMPUTE-CPU-15-TEMP2"
}
```

CRM Rest APIs

Note 1: All of this API documentation shown below is ONLY intended as a reference for advanced application developers. All normal user interaction with CRM should take place completely through the CRM UI.

Note 2: Prior to running any of the curl commands to invoke any of these API calls, the user must source the /home/stack/keystone.osrc file on a controller node and then run the following sequence of commands:

```
stack@hcg-cp1-controller-m1-clm:~$ source /home/stack/keystone.osrc
stack@hcg-cp1-controller-m1-clm:~$ cat /home/stack/keystone.osrc
# Refer http://docs.openstack.org/developer/python-openstackclient/command-list.html

# Environment variables for keystone v3 API using openstack client

# Keystone requires domain scoped token

unset OS_PROJECT_NAME
unset OS_PROJECT_DOMAIN_NAME

export OS_IDENTITY_API_VERSION=3
export OS_USERNAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PASSWORD=jeUyxlgVZo
export OS_DOMAIN_NAME=Default
export OS_AUTH_URL=https://10.10.62.119:5000/v3        <<<<<<<<<<  Need this
OS_AUTH_URL
export OS_INTERFACE=internal
export OS_CACERT=/etc/ssl/certs/ca-certificates.crt
```

```
stack@hcg-cp1-controller-m1-clm:~$ cat /etc/crm/crm.conf    <<<<<<   look in the
[keystone] section near the end of the file
[keystone]
protocol=https
host=hcg-cp1-vip-KEY-API-clm
admin_port=35357
public_port=5000
version=v2.0
cacert=/etc/ssl/cos/cacert.pem
tenant=services
domain=default
project_name=services
project_domain=Default
username=crm
user_domain=Default
password=DL9Iu8y2y                                        <<<<<<<<<<  Need this password
auth_url=https://hcg-cp1-vip-KEY-API-clm:5000/v3
interface=internal
certificate=/etc/ssl/certs/ca-certificates.crt
compute_version=2

Generate token using this curl command:

curl -i   -H "Content-Type: application/json"   -d '
{ "auth": {
    "identity": {
      "methods": ["password"],
      "password": {
        "user": {
          "name": "crm",
          "domain": { "id": "default" },
          "password": "DL9Iu8y2y"                  <<<<<<<< password from crm.conf
file
        }
      }
    }
  }
}'   https://10.200.79.118:5000/v3/auth/tokens    <<<<<<<<   from OS_AUTH_URL in
keystone.osrc file

HTTP/1.1 201 Created
Date: Fri, 11 Nov 2016 22:03:00 GMT
Server: Apache/2.4.10 (Debian)
X-Subject-Token: 7e0de50fe0694e7488d39dc5203bc1af      <<<<<<<<<<<<<<<  Auth token
needed for all API curl commands
Vary: X-Auth-Token
x-openstack-request-id: req-c39ecd36-436c-4cef-84ac-1bc889db5e8d
Content-Length: 281
Content-Type: application/json
{"token": {"issued_at": "2016-11-11T22:03:00.363541Z", "audit_ids":
["9_zZeWq2QT20p7s6XH_Lag"], "methods": ["password"], "expires_at":
"2016-11-12T02:03:00.363435Z", "user": {"domain": {"id": "default", "name":
"Default"}, "id": "91ae320871934c3a82bf8d23d9656040", "name": "crm"}}}
```

```
stack@hcg-cp1-controller-m1-clm:~$ export
OS_TOKEN="7e0de50fe0694e7488d39dc5203bc1af"
```

The X-Subject-Token line provides the output illustrated above and has to be exported to the shell using the OS_TOKEN variable.

Inventory APIs

From any server that has access to the Controller's Management VIP invoke following APIs through curl:

```
URI:
/computes

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET
https://<clm-vip>:5001/computes

Purpose:
To fetch a list of Computes. Details like nic-list, cpu-info, memor-info are not
included in the JSON by design

Output:
JSON containing Compute List

Input:
None
```

URI:
/computes/<Compute node’s hostname>

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET https://<clm-vip>:5001/computes/<Compute node's Hostname>

Purpose:
To fetch a details of a specific compute including details like nic-list, cpu-info, memor-info etc

Output:
JSON containing Compute details

Input:
None

Known Issue in Alpha:
The nics of the compute that don't have an interface configured, are not returned in the nic inventory of the Compute.

URI:
/computes/<Compute node's Hostname>/cpu

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET https://<clm-vip>:5001/computes/<Compute node's Hostname>/cpu

Purpose:
To get only CPU details of a specific Compute

Output:
JSON containing CPU details of a specific Compute

Input:
None

URI:
/computes/<Compute node's Hostname>/nics

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET https://<clm-vip>:5001/computes/<Compute node's Hostname>/nics

Purpose:
To get only NIC details of a specific Compute

Output:
JSON containing NIC details of a specific Compute

Input:
None

URI:
/computes/<Compute node's Hostname>/disks

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET https://<clm-vip>:5001/computes/<Compute node's Hostname>/disks

Purpose:
To get only disk details of a specific Compute

Output:
JSON containing disk details of a specific Compute

Input:
None

```
URI:
/computes/<Compute node's Hostname>/memory

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET
https://<clm-vip>:5001/computes/<Compute node's Hostname>/memory

Purpose:
To get only memory details of a specific Compute

Output:
JSON containing memory details of a specific Compute

Input:
None
```

- To Validate the API output that is returned as a JSON file review the file content and verify it is correct.

Compute node configuration Template APIs

These API calls illustrate how to work with the compute node configuration templates used in CRM.

NOTE: With a freshly installed Helion Carrier Grade cloud, the compute node configuration fields in the database are not initialized with any meaningful values so any output from the template GET API calls will return empty files. So to access any default templates using the API GET commands below, the user must first use the CRM UI to create a new template or use POST APIs to create new templates. This will initialize the template fields in database with default values. Then when the GET API calls are done below the JSON file that are returns will contain those default configurations for the compute nodes.

Any server that has access to the Controller's Management VIP and has the appropriate keystone authentication token can access the following APIs using the curl commands shown below:

```
URI:
/templates

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET
https://<clm-vip>:5001/templates

Purpose:
To fetch a list of Templates [all types of templates].

Output:
JSON containing Template List

Input:
None
```

NIC Template APIs

Each NIC Template corresponds to how the network interfaces must be laid out for a particular Compute.

From any server that has access to the Controller's Management, invoke following APIs through curl:

```
URI:
/templates/nic

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET
https://<clm-vip>:5001/templates/nic

Purpose:
To fetch a list of NIC Templates.

Output:
JSON containing NIC Template List

Input:
None
```

URI:
/templates/nic

REST Method:
POST

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -d {json} -X POST https://<clm-vip>:5001/templates/nic

Purpose:
To create a NIC Template.

Output:
None

Input:
JSON Containing NIC Templates details.
The contents of the JSON will have to be altered to match the exact setup details.
[Change pci busID as per actual hardware configuration]
[Add 'type' value depending on the interface with the following valid values 'control', 'data_sriov', 'data_pcipt', 'data_vswitch_ovsdpdk', 'data_vswitch_vrs']
[The device name will be auto generated by CRM]
Please find sample json here
[https://wiki.hpcloud.net/download/attachments/64917457/nictemp_ovs.json?api=v2]

Known Issue
Please note - the PCI bus addresses saved as part of the NIC Template are currently not getting used in HOS Input Model.
These bus addresses are only place holders for a future features.

URI:
/templates/nic/<nictempid>

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET https://<clm-vip>:5001/templates/nic/<nictempid>

Purpose:
To get details of a specific NIC Template

Output:
JSON containing NIC Template
Input:
Please make sure the template id in the URL is of an existing Template ID

```
URI:
/templates/nic/<nictempid>

REST Method:
PUT

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -d {json}
-X PUT https://<clm-vip>:5001/templates/nic/<nictempid>

Purpose:
To edit a NIC Template.

Output:
None

Input:
JSON Containing NIC Templates details. For Format refer to:
Please fetch the JSON by running a GET API for this template and then edit it and
ensure the values for the ids are not changed.
```

```
URI:
/templates/nic/<nictempid>

REST Method:
DELETE

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X DELETE
https://<clm-vip>:5001/templates/nic/<nictempid>

Purpose:
To delete a NIC Template.

Output:
None

Input:
Please make sure the template id in the URL is of an existing Template ID
```

Note: that this API does not change the Compute and simply makes a database association of the templateid and nodeid in Node toTemplate table

```
URI:
/computes/associatetemplate

REST Method:
POST

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -d {json}
-X POST https://<clm-vip>:5001/computes/associatetemplate

Purpose:
To associate a NIC Template on a particular compute.
Please note that this API does not change the Compute and simply makes a database
association of the templateid and nodeid in NodetoTemplate table

Output:
None

Input:
JSON containing nic template id and compute id. For format refer to:
'{"hostname": "hcg-cp1-comp0001-clm", "templateid": 24}'
The contents of the JSON will have to be altered to match the actual setup details.
Please make sure templateid is a valid id of an existing NIC Template
```

NIC template operation examples:

Example1: Get list of all available Templates

```
stack@hcg-cp1-controller-m1-clm:~/json-templates$ curl -i -H "Content-Type:
application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET
https://10.35.2.119:5001/templates
HTTP/1.1 200 OK
Date: Tue, 15 Nov 2016 04:40:46 GMT
Server: Apache/2.4.10 (Debian)
Content-Length: 257
Content-Type: application/json

[{"templatetype": "DISK", "id": 1, "name": "COMPUTE-DISKS-TEMP1"},{"templatetype":
"CPU", "id": 7, "name": "COMPUTE-CPU-TEMP1"},{"templatetype": "MEM", "id": 10,
"name": "COMPUTE-MEMORY-TEMP1"},{"templatetype": "NIC", "id": 13, "name":
"COMPUTE-NIC-TEMP1"}]
```

Example2:- Get list of available NIC Template

```
stack@hcg-cp1-controller-m1-clm:/var/hcg40/crm/restapiserver$ curl -i -H
"X-Auth-Token: $OS_TOKEN" -X GET https://10.80.34.2:5001/templates/nic
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 1984
Server: Werkzeug/0.9.6 Python/2.7.9
Date: Fri, 08 Jul 2016 11:57:11 GMT
Connection: keep-alive
[{"py/object": "crm.model.templatemodel.NicTemplate", "name":
"NICTemplate4-BondedComputeInterfaces", "porttonetworks": [{"py/object":
"crm.model.templatemodel.PortToNetworks", "name": "UNBONDED_INTERFACE",
"networktypelist": ["EXT", "TUL"], "propertylist": {"sriov-only": "true",
"vf-count": 10}, "externalnetlist": [""], "uniqueid": 1, "tullist": [""], "port":
{"py/object": "crm.model.templatemodel.SimplePort", "busaddress": "0000:04:0004",
"devicename": "eth6"}}, {"py/object": "crm.model.templatemodel.PortToNetworks",
"name": "BONDED_INTERFACE", "networktypelist": ["PXE", "CLM"], "propertylist": {},
"externalnetlist": [""], "uniqueid": 4, "tullist": [""], "port": {"py/object":
"crm.model.templatemodel.BondedPort", "devicelist": [{"py/object":
"crm.model.templatemodel.SimplePort", "busaddress": "0000:04:0004", "devicename":
"eth4"}, {"py/object": "crm.model.templatemodel.SimplePort", "busaddress":
"0000:04:0004", "devicename": "eth4"}], "bondpropertylist": {"options": "{mode:
active-backup, miimon: 200}", "provider": "linux"}, "name": "bond1"}}],
"templatetype": "NIC", "nicmappingid": 1, "uniqueid": 1},{"py/object":
"crm.model.templatemodel.NicTemplate", "name": "NICTemplate1-ComputeInterfaces",
"porttonetworks": [{"py/object": "crm.model.templatemodel.PortToNetworks", "name":
"SIMPLE_INTERFACE_TEMPLATE_1", "networktypelist": ["EXTERNAL-VM", "MANAGEMENT"],
"propertylist": {}, "externalnetlist": [""], "uniqueid": 7, "tullist": [""], "port":
{"py/object": "crm.model.templatemodel.SimplePort", "busaddress": "0000:04:0004",
"devicename": "eth0"}}, {"py/object": "crm.model.templatemodel.PortToNetworks",
"name": "SIMPLE__INTERFACE_TEMPLATE_SRIOV1", "networktypelist": ["GUEST"],
"propertylist": {"sriov-only": true, "vf-count": 5}, "externalnetlist": [""],
"uniqueid": 10, "tullist": [""], "port": {"py/object":
"crm.model.templatemodel.SimplePort", "busaddress": "0000:04:0004", "devicename":
"eth1"}}], "templatetype": "NIC", "nicmappingid": 1, "uniqueid": 4}]
```

Example3:- Get specific NIC Template passing the template ID

```
stack@hcg-cp1-controller-m1-clm:/var/hcg40/crm/restapiserver$ curl -i -H
"X-Auth-Token: $OS_TOKEN" -X GET https://10.80.34.2:5001/templates/nic/1 HTTP/1.0
200 OK Content-Type: application/json Content-Length: 1118 Server: Werkzeug/0.9.6
Python/2.7.9 Date: Fri, 08 Jul 2016 11:58:39 GMT Connection: keep-alive
{"py/object": "crm.model.templatemodel.NicTemplate", "name":
"NICTemplate4-BondedComputeInterfaces", "porttonetworks": [{"py/object":
"crm.model.templatemodel.PortToNetworks", "name": "UNBONDED_INTERFACE",
"networktypelist": ["EXT", "TUL"], "propertylist": {"sriov-only": "true",
"vf-count": 10}, "externalnetlist": [""], "uniqueid": 1, "tullist": [""], "port":
{"py/object": "crm.model.templatemodel.SimplePort", "busaddress": "0000:04:0004",
"devicename": "eth6"}}, {"py/object": "crm.model.templatemodel.PortToNetworks",
"name": "BONDED_INTERFACE", "networktypelist": ["PXE", "CLM"], "propertylist": {},
"externalnetlist": [""], "uniqueid": 4, "tullist": [""], "port": {"py/object":
"crm.model.templatemodel.BondedPort", "devicelist": [{"py/object":
"crm.model.templatemodel.SimplePort", "busaddress": "0000:04:0004", "devicename":
"eth4"}, {"py/object": "crm.model.templatemodel.SimplePort", "busaddress":
"0000:04:0004", "devicename": "eth4"}], "bondpropertylist": {"options": "{mode:
active-backup, miimon: 200}", "provider": "linux"}, "name": "bond1"}}],
"templatetype": "NIC", "nicmappingid": 1, "uniqueid": 1}]
```

Example4:- Creating NIC Template

Please check the port/pcibus address according to Compute

```
stack@hcg-cp1-controller-m1-clm:~/json-templates$ curl -i -H "Content-Type:
application/json" -H "X-Auth-Token: $OS_TOKEN" -d '@nicpost.json' -X POST
https://10.35.2.119:5001/templates/nic
HTTP/1.1 100 Continue

HTTP/1.1 201 CREATED
Date: Tue, 15 Nov 2016 04:27:26 GMT
Server: Apache/2.4.10 (Debian)
Content-Length: 58
Content-Type: application/json

{"message": "NIC Template created successfully", "id": 13}
```

```
stack@hcg-cp1-hlm-hyp-m1-clm:~$ curl -i -H "Content-Type: application/json" -H
"X-Auth-Token: $OS_TOKEN" -X GET https://10.246.106.27:5001/templates/nic
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 696
Server: Werkzeug/0.11.10 Python/2.7.9
Date: Fri, 09 Sep 2016 21:33:46 GMT
[{"porttonetworks": [{"name": "SIMPLE_INTERFACE_TEMPLATE_1", "networktypelist":
["EXTERNAL-VM", "MANAGEMENT"], "propertylist": {}, "externalnetlist": [""], "id": 4,
"tullist": [""], "port": {"busaddress": "0000:04:0004", "devicename": "hed4"}}],
"templatetype": "NIC", "nicmappingid": 1, "id": 4, "name":
"NICTemplate3-ComputeInterfaces"},{"porttonetworks": [{"name": "UNBONDED_INTERFACE",
"networktypelist": ["EXT", "TUL"], "propertylist": {"sriov-only": "true",
"vf-count": 10}, "externalnetlist": [""], "id": 7, "tullist": [""], "port":
{"busaddress": "0000:04:0004", "devicename": "hed1"}}], "templatetype": "NIC",
"nicmappingid": 1, "id": 61, "name": "NICTemplate4-BondedComputeInterfaces"}]
```

Example5:- Delete NIC Template

```
stack@hcg-cp1-hlm-hyp-m1-clm:~$ curl -i -H "Content-Type: application/json" -H
"X-Auth-Token: $OS_TOKEN" -X DELETE https://10.246.106.27:5001/templates/nic/4
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 48
Server: Werkzeug/0.11.10 Python/2.7.9
Date: Fri, 09 Sep 2016 21:34:44 GMT
{"message": "NIC Template deleted successfully"}
```

Memory Template API

- Memory Template CRUD APIs
- POST, PUT, DELETE, GET, GET LIST APIs for Memory Template

- The Memory Template can be created and Read back.
- The template is being created and stored in the CRM database.
- This template can be applied on a specific compute.

  From any server that has access to the Controller's Management VIP invoke following APIs through curl:.

URI:
/templates/memory

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET https://<clm-vip>:5001/templates/memory

Purpose:
To get a list of memory Templates

Output:
JSON containing memory Template List

Input:
None

URI:
/templates/memory

REST Method:
POST

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -d ‘@memorypost.json’ -X POST https://<clm-vip>:5001/templates/memory

Purpose:
To create a memory Templates – make sure attached memorypost.json file copied into the current working directory when running these curl apis

Output:
{"message": "Memory Template created successfully", "id": <template id>}

Input:
JSON containing Memory Template details.

URI:
/templates/memory/<memorytempid>

REST Method:
PUT

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -d '@memoryput.json' -X PUT https://<clm-vip>:5001/templates/memory/<memtempid>

Purpose:
To update the created templates with memeory template id – make sure attached memoryput.json file copied into the current working directory when running the curl apis and change the uniqueid value to match the memorytemplateid

Output:
Acknowledge message with updated ID

Input:
JSON containing Memory Template details.
Please fetch the JSON by running a GET API for this template and then edit it and ensure do not change the values for the ids

URI:
/templates/memory/<memorytempid>

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET https://<clm-vip>:5001/templates/memory/<memorytempid>

Purpose:
To get a specific Memory Template

Output:
JSON containing Memory Template

Input:
Please make sure the template id in the URL and in the JSON file is of an existing Template ID

URI:
/templates/memory/<memorytempid>

REST Method:
DELETE

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X DELETE https://<clm-vip>:5001/templates/memory/<memorytempid>

Purpose:
To delete specific Memory Template

Output:
Acknowledge message with deleted ID

Input:
Please make sure the template id in the URL is of an existing Template ID

Known Issue:
It works but there is known bug which displays error message "{"message": "Memory Template deletion failed"} even though the template has been deleted

URI:
/computes/associatetemplate

REST Method:
POST

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -d {json} -X POST https://<clm-vip>:5001/computes/associatetemplate

Purpose:
To associate a Memory Template on a particular compute

Output:
None

Input:
JSON containing memory template id and compute id. For format refer to:
‘{"hostname": "hcg-cp1-comp0001-clm", "templateid": <template id of type MEM>}’
The contents of the JSON will have to be altered to match the actual setup details.
Please make sure the template id in the JSON file is of an existing Template ID

CPU Template API

- POST, PUT, DELETE, GET, GET LIST APIs for CPU Template

- The CPU Template can be created and Read back. It CANNOT be applied to Computes yet.
- The template is simply being created and stored in the CRM database as of now.
  From any server that has access to the Controller's Management VIP invoke following APIs through curl:

```
URI:
/templates/cpu

REST Method:
GET
Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET
https://<clm-vip>:5001/templates/cpu

Purpose:
To get a list of CPU Templates

Output:
JSON containing CPU Template List

Input:
None
```

```
URI:
/templates/cpu

REST Method:
POST

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -d
{request} -X POST https://<clm-vip>:5001/templates/cpu

Purpose:
To create a CPU Templates – make sure attached cpupost.json file copied into the
current working directory when running the curl apis

Output:
{"message": "CPU Template created successfully", "id": <template id>}

Input:
JSON containing CPU Template details.
```

URI:
/templates/cpu/<cputempid>

REST Method:
PUT

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -d ‘@cpuput.json’ -X PUT https://<clm-vip>:5001//templates/cpu/<cputemplateid>

Purpose:
To update the created templates with cputemplate id– make sure attached cputput.json file copied into the current working directory when running the curl apis and change the uniqueid value to match the cputemplateid

Output:
Acknowledge message

Input:
JSON containing CPU Template details.
Please fetch the JSON by running a GET API for this template and then edit it and ensure to not change the values for the ids

URI:
/templates/cpu/<cputempid>

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET https://<clm-vip>:5001//templates/cpu/<cputempid>

Purpose:
To get a specific CPU Template

Output:
JSON containing CPU Template

Input:
Please make sure the template id in the URL is of an existing Template ID

```
URI:
/templates/cpu/<cputempid>

REST Method:
DELETE

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X DELETE
https://<clm-vip>:5001//templates/cpu/<cputempid>

Purpose:
To delete specific CPU template

Output:
Acknowledge message

Input:
Please make sure the template id in the URL is of an existing Template ID
```

```
URI:
/computes/associatetemplate

REST Method:
POST

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -d {json}
-X POST https://<clm-vip>:5001/computes/associatetemplate

Purpose:
To associate a CPU Template on a particular compute

Output:
None

Input:
JSON containing cpu template id and compute id. For format refer to:
‘{"hostname": "hcg-cp1-comp0001-clm", "templateid": <template id of type CPU>}’
The contents of the JSON will have to be altered to match the actual setup details.
Please make sure the template id in the JSON file is of an existing Template ID.
```

Disk Template APIs

- The DISK Template can be created and Read via the following command. At this version, we only provide the compute node disk template.
- The disk template is simply being created and stored in the CRM database and can use the RESTful API to get and put, update, delete the compute disk template through RESTful API.
- From any server that has access to the Controller's Management VIP and the default port 5001 invoke

following APIs through curl, the output of the RESTful api is JSON format.

The DISK Template can only be applied on newly provisioned compute nodes. The steps provided below are required before applying a DISK Template to a specific compute node

```
# As a sample "compute1" is used, which is the name of the compute that can be found
in the servers.yml file. The "hcg-cp1-comp0001-clm" is the hostname of compute1.

# Setup cobbler
source /opt/stack/venv/ansible-hos-4.0.0/bin/activate
cd ~/helion/hos/ansible; ansible-playbook -i hosts/localhost cobbler-deploy.yml -e
hlmuser_password='stack'

# Provision the compute node.
cd ~/helion/hos/ansible; ansible-playbook -i hosts/localhost bm-reimage.yml -e
nodelist=compute1

# Delete partition of the non-sda device on the compute node.
cd ~/scratch/ansible/next/hos/ansible/; ansible-playbook -i hosts/verb_hosts
wipe_disks.yml --limit=hcg-cp1-comp0001-clm

# Steps for deploying mini CRM Agent on newly provision compute node.
cd ~/scratch/ansible/next/hos/ansible/; ansible-playbook -i hosts/verb_hosts
site.yml --tag "generate_hosts_file"
cd ~/scratch/ansible/next/hos/ansible/; ansible-playbook -i hosts/verb_hosts
tls-reconfigure.yml --limit=hcg-cp1-comp0001-clm
cd ~/scratch/ansible/next/hos/ansible/; ansible-playbook -i hosts/verb_hosts
crm-provision-agent-deploy.yml --limit=hcg-cp1-comp0001-lm
cd ~/scratch/ansible/next/hos/ansible/; ansible-playbook -i hosts/verb_hosts
crm-provision-agent-start.yml --limit=hcg-cp1-comp0001-cl
```

```
-----Create Disk Template
URI:
/templates/disk
REST Method:
POST
Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -d
{@jsonfile} -X POST https://<clm-vip>:5001/templates/disk
Purpose:
To create a DISK Template.
Output:
None
Input:
jsonfile Containing DISKTemplates details. For Format refer to:
https://wiki.hpcloud.net/pages/viewpageattachments.action?pageId=58951051
&sortBy=date&highlight=computedisk.json&
The contents of the JSON will have to be altered to match the actual setup details.
```

GET----Get Template
URI:
/templates/disk
REST Method:
GET
Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET https://<clm-vip>:5001/templates/disk
Purpose:
To fetch the compute disk Templates.
Output:
JSON containing disk Template List
Input:
None

GET---Specify the template id whose type is "DISK"
URI:
/templates/disk/<templateid>
REST Method:
GET
Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET https://<clm-vip>:5001/templates/disk/<templateid>
Purpose:
To get details of a specific disk Template
Output:
JSON containing disk Template
Input:
None

PUT------update the template (specify the template id)
URI:
/templates/disk/<templateid>
REST Method:
PUT
Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -d {json} -X PUT https://<clm-vip>:5001/templates/disk/<templateid>
Purpose:
To edit or update a disk Template.
Output:
None
Input:
JSON Containing DISK Templates details.

Please fetch the JSON by running a GET API for this template and then edit it and ensure to not change the values for the ids

URI:
/templates/disk/<templateid>
REST Method:
DELETE
Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X DELETE https://<clm-vip>:5001/templates/disk/<templateid>
Purpose:
To delete a disk Template specify the template-id.
Output:
None
Input:
None

Use the associate the disk apply:
URI:
/computes/associatetemplate

REST Method:
POST

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -d {json} -X POST https://<clm-vip>:5001/computes/associatetemplate

Purpose:
To associate a DISK Template on a particular compute node

Output:
None

Input:
JSON containing disk template id and compute node id. For format refer to:
‘{"hostname": "hcg-cp1-comp0001-clm", "templateid": <template id of type DISK>}’
The contents of the JSON will have to be altrered to match the actual setup details.
Please make sure the template id in the JSON file is of an existing Template ID.

```
URI:
/computes/applydisktemplate
REST Method:
APPLY
Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -d {json}
-X POST https://<clm-vip>:5001/computes/applydisktemplate
Purpose:
To apply a Disk Template on a particular compute node
Output:
None
Input:
JSON containing disk disktemplate id and compute id. For format refer to:
‘{"hostname": "hcg-cp1-comp0001-clm", "templateid": <templateid>}’
The contents of the JSON will have to be altered to match the actual setup details.
```

Disk template operation examples:

Example1:- Get list of available Disk Template

disk_post.json

```
stack@hcg-cp1-controller-m1-clm:/var/hcg40/crm/restapiserver$ curl -i -H
"X-Auth-Token: $OS_TOKEN" -X GET https://10.80.34.2:5001/templates/disk

[{"volume_groups": [{"logical_volume": [{"name": "root", "mount": "/", "fstype":
"ext4", "mkfs_opts": "", "uniqueid": 1, "size": "35%"}, {"name": "log", "mount":
"/var/log", "fstype": "ext4", "mkfs_opts": "-O large_file", "uniqueid": 4, "size":
"50%"}, {"name": "crash", "mount": "/var/crash", "fstype": "ext4", "mkfs_opts": "-O
large_file", "uniqueid": 7, "size": "10%"}], "name": "hlm-vg", "uniqueid": 1,
"physical_volume": [{"uniqueid": 1, "disk_name": "/dev/sda_root"}]},
{"logical_volume": [{"name": "compute", "mount": "/var/lib/nova", "fstype": "ext4",
"mkfs_opts": "-O large_file", "uniqueid": 10, "size": "95%"}], "name": "vg-comp",
"uniqueid": 4, "physical_volume": [{"uniqueid": 4, "disk_name": "/dev/sdc"}]}],
"templatetype": "DISK", "name": "COMPUTE-DISKS-TEST1", "uniqueid":
1},{"volume_groups": [{"logical_volume": [{"name": "root", "mount": "/", "fstype":
"ext4", "mkfs_opts": "", "uniqueid": 13, "size": "35%"}, {"name": "log", "mount":
"/var/log", "fstype": "ext4", "mkfs_opts": "-O large_file", "uniqueid": 16, "size":
"50%"}, {"name": "crash", "mount": "/var/crash", "fstype": "ext4", "mkfs_opts": "-O
large_file", "uniqueid": 19, "size": "10%"}], "name": "hlm-vg", "uniqueid": 7,
"physical_volume": [{"uniqueid": 7, "disk_name": "/dev/sda_root"}]},
{"logical_volume": [{"name": "compute", "mount": "/var/lib/nova", "fstype": "ext4",
"mkfs_opts": "-O large_file", "uniqueid": 22, "size": "92%"}], "name": "vg-comp",
"uniqueid": 10, "physical_volume": [{"uniqueid": 10, "disk_name": "/dev/sdb"}]}],
"templatetype": "DISK", "name": "COMPUTE-DISKS-new", "uniqueid":
4},{"volume_groups": [{"logical_volume": [{"name": "root", "mount": "/", "fstype":
"ext4", "mkfs_opts": "", "uniqueid": 25, "size": "35%"}, {"name": "log", "mount":
"/var/log", "fstype": "ext4", "mkfs_opts": "-O large_file", "uniqueid": 28, "size":
"50%"}, {"name": "crash", "mount": "/var/crash", "fstype": "ext4", "mkfs_opts": "-O
large_file", "uniqueid": 31, "size": "10%"}], "name": "hlm-vg", "uniqueid": 13,
"physical_volume": [{"uniqueid": 13, "disk_name": "/dev/sda_root"}]},
{"logical_volume": [{"name": "compute", "mount": "/var/lib/nova", "fstype": "ext4",
"mkfs_opts": "-O large_file", "uniqueid": 34, "size": "92%"}], "name": "vg-comp",
"uniqueid": 16, "physical_volume": [{"uniqueid": 16, "disk_name": "/dev/sdb"}]}],
"templatetype": "DISK", "name": "COMPUTE-DISKS-songbo", "uniqueid":
7},{"volume_groups": [{"logical_volume": [{"name": "root", "mount": "/", "fstype":
"ext4", "mkfs_opts": "", "uniqueid": 37, "size": "35%"}, {"name": "log", "mount":
"/var/log", "fstype": "ext4", "mkfs_opts": "-O large_file", "uniqueid": 40, "size":
"50%"}, {"name": "crash", "mount": "/var/crash", "fstype": "ext4", "mkfs_opts": "-O
large_file", "uniqueid": 43, "size": "10%"}], "name": "hlm-vg", "uniqueid": 19,
"physical_volume": [{"uniqueid": 19, "disk_name": "/dev/sda_root"}]},
{"logical_volume": [{"name": "compute", "mount": "/var/lib/nova", "fstype": "ext4",
"mkfs_opts": "-O large_file", "uniqueid": 46, "size": "92%"}], "name": "vg-comp",
"uniqueid": 22, "physical_volume": [{"uniqueid": 22, "disk_name": "/dev/sdb"}]}],
"templatetype": "DISK", "name": "COMPUTE-DISKS-songbo-2", "uniqueid": 10}]
```

In the json file above, the highlighted numbers are the disktemplate ids. In the example above, the disktemplate id is: 1,4,7,10.

Example2:- Get specified Template (for example template id is 7)

```
stack@hcg-cp1-controller-m1-clm:/var/hcg40/crm/restapiserver$ curl -i -X  GET  -H
"Content-Type: application/json"  -H "X-Auth-Token: $OS_TOKEN"
https://10.200.79.117:5001/templates/disk/7
{"volume_groups": [{"logical_volume": [{"name": "root", "mount": "/", "fstype":
"ext4", "mkfs_opts": "", "uniqueid": 25, "size": "35%"}, {"name": "log", "mount":
"/var/log", "fstype": "ext4", "mkfs_opts": "-O large_file", "uniqueid": 28, "size":
"50%"}, {"name": "crash", "mount": "/var/crash", "fstype": "ext4", "mkfs_opts": "-O
large_file", "uniqueid": 31, "size": "10%"}], "name": "hlm-vg", "uniqueid": 13,
"physical_volume": [{"uniqueid": 13, "disk_name": "/dev/sda_root"}]},
{"logical_volume": [{"name": "compute", "mount": "/var/lib/nova", "fstype": "ext4",
"mkfs_opts": "-O large_file", "uniqueid": 34, "size": "92%"}], "name": "vg-comp",
"uniqueid": 16, "physical_volume": [{"uniqueid": 16, "disk_name": "/dev/sdb"}]}],
"templatetype": "DISK", "name": "COMPUTE-DISKS-songbo", "uniqueid": 7}
```

Example3:- Creating disk Template

```
curl -i -X POST  -H "Content-Type: application/json"  -H "X-Auth-Token: $OS_TOKEN"
-d '{
                    "name": "COMPUTE-DISKS-testing32",
                    "volume-groups": [
                       {
                         "name": "hlm-vg",
                         "physical-volumes": [
                           "/dev/sda_root"
                         ],
                         "logical-volumes": [
                           {
                             "name": "root",
                             "size": "35%",
                             "fstype": "ext4",
                             "mount": "/"
                           },
                           {
                             "name": "log",
                             "size": "50%",
                             "mount": "/var/log",
                             "fstype": "ext4",
                             "mkfs-opts": "-O large_file"
                           },
                           {
                             "name": "crash",
                             "size": "10%",
                             "mount": "/var/crash",
                             "fstype": "ext4",
                             "mkfs-opts": "-O large_file"
                           }
                         ]
                       },
                       {
                         "name": "vg-comp",
                         "physical-volumes": [
                           "/dev/sdb"
                         ],
                         "logical-volumes": [
                           {
                             "name": "compute",
                             "size": "92%",
                             "mount": "/var/lib/nova",
                             "fstype": "ext4",
                             "mkfs-opts": "-O large_file"
                           }
                         ]
                       }
                     ]
                   }' https://10.200.79.117:5001/templates/disk
```

Example4:- Use json file to create one disk temaplate

Store the json data in one file such as disk_post.json, then use the curl command shown below:

```
curl -i -X POST  -H "Content-Type: application/json"  -H "X-Auth-Token: $OS_TOKEN"
-d '@disk_post.json'  https://10.200.79.118:5001/templates/disk

HTTP/1.1 100 Continue
HTTP/1.1 201 CREATED
Date: Sat, 08 Oct 2016 18:55:30 GMT
Server: Apache/2.4.10 (Debian)
Content-Length: 59
Content-Type: application/json
{"message": "Disk Template created successfully", "id": 10}
```

Example5:- Updating the disk template

In this example the modified disk template name is COMPUTE-DISKS-new, modified size:45% . The updated template id is: 10

Note: When working with disk templates, the disk template id is used instead of the template id. In the disk structure, there will be only one template in the template table, and there are multiple disk template entries in disk template.

```
curl -i -X PUT  -H "Content-Type: application/json"  -H "X-Auth-Token: $OS_TOKEN" -d
'{
                    "name": "COMPUTE-DISKS-testing32",
                    "volume-groups": [
                       {
                         "name": "hlm-vg",
                         "physical-volumes": [
                           "/dev/sda_root"
                         ],
                         "logical-volumes": [
                           {
                             "name": "root",
                             "size": "45%",
                             "fstype": "ext4",
                             "mount": "/"
                           },
                           {
                             "name": "log",
                             "size": "50%",
                             "mount": "/var/log",
                             "fstype": "ext4",
                             "mkfs-opts": "-O large_file"
                           },
                           {
                             "name": "crash",
                             "size": "10%",
                             "mount": "/var/crash",
                             "fstype": "ext4",
                             "mkfs-opts": "-O large_file"
                           }
                         ]
                       },
                       {
                         "name": "vg-comp",
                         "physical-volumes": [
                           "/dev/sdb"
                         ],
                         "logical-volumes": [
                           {
                             "name": "compute",
                             "size": "92%",
                             "mount": "/var/lib/nova",
                             "fstype": "ext4",
                             "mkfs-opts": "-O large_file"
                           }
                         ]
                       }
                     ]
                   }' https://10.200.79.117:5001/templates/disk/10

HTTP/1.1 100 Continue
HTTP/1.1 200 OK
Date: Sat, 08 Oct 2016 19:50:31 GMT
Server: Apache/2.4.10 (Debian)
Content-Length: 59
Content-Type: application/json
{"message": "Disk Template updated successfully", "id": 10}
```

Example6:- Update the disk template from json file:

In this example, edit the above json file and store in the file name "disk_put.json":

```
curl -i -X PUT  -H "Content-Type: application/json"  -H "X-Auth-Token: $OS_TOKEN" -d
'@disk_put.json' https://10.200.79.118:5001/templates/disk/10
```

disk_put.json can be download from: https://wiki.hpcloud.net/pages/viewpageattachments.action?pageId=64917457&highlight=disk_put.json#CRM+Runbook-attachment-disk_put.json

Example7:- DELETE the disk template:

In the disk template input model design, the command does not delete the whole disk input model because there are multiple disk template model in the template table. So in order to delete the disk template table, it will be necessary to specify the template id.

This example shows how to delete template id=10:

```
curl -i -X DELETE -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN"
https://10.10.32.119:5001/templates/disk/10
HTTP/1.1 200 OK
Date: Sat, 08 Oct 2016 18:48:40 GMT
Server: Apache/2.4.10 (Debian)
Content-Length: 49
Content-Type: application/json
{"message": "Disk Template deleted successfully"}
```

Compute Template APIs

URI:
/compute/template

REST Method:
POST

Command:

```
curl -H "X-Auth-Token: $OS_TOKEN" -H "Content-Type: application/json" -X POST
https://<clm-vip>:5001/compute/template -d
'{
    "name": "HPC-1",
    "cputemplateid": 7,
    "memorytemplateid": 1,
    "nictemplateid": 19,
    "disktemplateid": 4
}'
```

Purpose:
Create compute template

Output:
Json containing success or failure message and Id of the newly created compute template
E.g. {"message": "Compute Template added/updated successfully", "id": 7}

Error Code:
200 - Success
422 - Failure

Input:
JSON containing Compute TemplateName, CPU TemplateId, Memory TemplateId, NIC TemplateId, Disk TemplateId.

URI:
/compute/template/<Compute TemplateId>

REST Method:
GET

Command:

```
curl -i -H "Content-Type: application/json" -H "X-Auth-Token:  $OS_TOKEN" -X GET
https://<clm-vip>:5001/compute/template/<Compute TemplateId>
```

Purpose:
Get's specified compute template along with SubTemplate details

Output:
Json containing details for CPU Template, NIC Template, Memory Template and Disk Template associated with Compute Template
E.g.

```
{
```

```
"disktemplate": {
 "volume_groups": [{
  "logical_volume": [{
   "name": "root",
   "mount": "/",
   "fstype": "ext4",
   "mkfs_opts": "",
   "id": 1,
   "size": "34%"
  }, {
   "name": "log",
   "mount": "/var/log",
   "fstype": "ext4",
   "mkfs_opts": "-O large_file",
   "id": 4,
   "size": "49%"
  }, {
   "name": "crash",
   "mount": "/var/crash",
   "fstype": "ext4",
   "mkfs_opts": "-O large_file",
   "id": 7,
   "size": "12%"
  }],
  "id": 1,
  "name": "hlm-vg",
  "physical_volume": [{
   "id": 1,
   "disk_name": "/dev/sda_root"
  }]
 }, {
  "logical_volume": [{
   "name": "compute",
   "mount": "/var/lib/nova",
   "fstype": "ext4",
   "mkfs_opts": "-O large_file",
   "id": 10,
   "size": "85%"
  }],
  "id": 4,
  "name": "vg-comp",
  "physical_volume": [{
   "id": 4,
   "disk_name": "/dev/sdb"
  }]
 }],
 "templatetype": "DISK",
 "id": 1,
 "name": "disk2"
},
"nictemplate": {
 "name": "ComputeInterfaces_tb1c1",
 "porttonetworks": [{
  "name": "BONDED_INTERFACE",
  "networktypelist": ["PXE", "CLM"],
  "forcednetworkgroups": ["BLS"],
  "propertylist": {},
  "externalnetlist": [""],
  "id": 1,
```

```
  "type": "",
  "tullist": [""],
  "port": {
   "devicelist": [{
    "busaddress": "0000:06:00.0",
    "devicename": "hed1"
   }, {
    "busaddress": "0000:06:00.1",
    "devicename": "hed2"
   }],
   "bondpropertylist": {
    "options": "{'mode': 'active-backup', 'miimon': 200, 'primary': 'hed1'}",
    "provider": "linux"
   },
   "name": "bond0"
  },
  "fcoeinterfaces": [""]
 }, {
  "name": "BONDED_DPDK_INTERFACE",
  "networktypelist": [""],
  "forcednetworkgroups": [""],
  "propertylist": {},
  "externalnetlist": [""],
  "id": 4,
  "type": "",
  "tullist": ["VxLAN-VLAN1-TUL"],
  "port": {
   "devicelist": [{
    "busaddress": "0000:09:00.0",
    "devicename": "dpdk0"
   }, {
    "busaddress": "0000:09:00.1",
    "devicename": "dpdk1"
   }],
   "bondpropertylist": {
    "options": "{'mode': 'active-backup', 'miimon': 200}",
    "provider": "openvswitch"
   },
   "name": "bond1"
  },
  "fcoeinterfaces": [""]
 }, {
  "name": "SRIOV_INTERFACE",
  "networktypelist": [""],
  "forcednetworkgroups": [""],
  "propertylist": {
   "sriov-only": true,
   "vf-count": 10
  },
  "externalnetlist": [""],
  "id": 7,
  "type": "",
  "tullist": ["VLAN2-TUL"],
  "port": {
   "busaddress": "0000:87:00.0",
   "devicename": "hed4"
  },
  "fcoeinterfaces": [""]
 }],
```

```
 "templatetype": "NIC",
 "nicmappingid": 10,
 "dpdkdevices": {
  "component-options": [{
   "name": "n-dpdk-rxqs",
   "value": 64
  }],
  "eal-options": [{
   "name": "socket-mem",
   "value": "1024,1024"
  }, {
   "name": "n",
   "value": 2
  }],
  "components": ["openvswitch"],
  "devices": [{
   "driver": "igb_uio",
   "name": "dpdk0"
  }, {
   "driver": "igb_uio",
   "name": "dpdk1"
  }]
 },
 "id": 19
},
"memorytemplate": {
 "name": "mem1",
 "templatetype": "MEM",
 "id": 1,
 "platformmemory": "2M",
 "defaultpagesize": "2M",
 "pages": [{
  "numanode": 1,
  "pagesize": "2M",
  "pagecount": 2600
 }, {
  "numanode": 1,
  "pagesize": "1G",
  "pagecount": 50
 }, {
  "numanode": 0,
  "pagesize": "2M",
  "pagecount": 4000
 }, {
  "numanode": 0,
  "pagesize": "1G",
  "pagecount": 50
 }, {
  "numanode": -1,
  "pagesize": "2M",
  "pagecount": 1000
 }, {
  "numanode": -1,
  "pagesize": "1G",
  "pagecount": 25
 }]
},
"name": "HPC-1",
"cputemplate": {
```

```
  "templatetype": "CPU",
  "coreassignment": [{
   "platform": 5,
   "numaid": 0,
   "vm": 8,
   "vswitch": 2
  }, {
   "platform": 1,
   "numaid": 1,
   "vm": 13,
   "vswitch": 1
  }],
  "id": 7,
  "name": "COMPUTE-CPU-15-CORES"
 }
}
```


Error Code:
200 - Success
422 - Failure

Input:
JSON containing Compute TemplateId

URI:
/compute/template

REST Method:
PUT

Command:

```
curl -H "X-Auth-Token: $OS_TOKEN" -H "Content-Type: application/json" -X PUT
https://<clm-vip>:5001/compute/template -d
'{
    "name": "HPC-1",
    "cputemplateid": 7,
    "memorytemplateid": 1,
    "nictemplateid": 19,
    "disktemplateid": 4,
    "computetemplateid": 16
}'
```

Purpose:
Modifies existing compute template

Output:
Json containing success or failure message and Id of the modified Compute Template
E.g. {"message": "Compute Template added/updated successfully", "id": 31}

Error Code:
200 - Success
422 - Failure

Input:
JSON containing updated values for Compute TemplateName, CPU TemplateId, Memory TemplateId, NIC TemplateId, Disk TemplateId and Compute TemplateId to be modified

URI:
/compute/template

REST Method:
DELETE

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X DELETE https://10.10.2.119:5001/compute/template/map/<Compute TemplateId>

Purpose:
Deletes existing compute template

Output:
Json containing success or failure message for the deleted Compute Template
E.g. {"message": "Compute Template has been deleted successfully"}

Error Code:
200 - Success
422 - Failure

Input:
JSON containing Compute TemplateId to be deleted

URI:
/compute/template/validate

REST Method:
POST

Command:
curl -H "X-Auth-Token: $OS_TOKEN" -H "Content-Type: application/json" -X POST https://<crm-vip>:5001/compute/template/validate -d <compute_template_validate.json>
Purpose:
Performs validations across sub/device templates

Output:
Returns success if compute template validation was successful.
{"message": "Compute Template validation successful"}

Returns following error if Numa node pci bus address in input json does not match with NIC inventory:
{"message": "No numa nodes found for pci bus address: ['0000:06:00.01', '0000:87:00.11']"}

Returns following error if socket memory (specified in nictemplate section)does not match with [no. of cores (specified in cputemplate section) * 2G].
{"message": "No. of 1G huge pages for vswitch (soc-mem): 1024 does not match no. of cores: 2) for numa node: 0"}(

Returns following error if 1G pages are node defined for numa node in cputemplate section
{"message": "1G page is not defined for numa node: 1 "}

Returns following error if no of numa nodes in socket-mem list (specified in nictemplate section)) does not match with numa nodes specified in nic inventory
{"message": "Numa nodes count do not match. socket-mem:4096. Number of numa nodes specifed in nic inventory: 2"

Returns following error if memory specified in memorytemplate section (vswitch element) is less than specified socket-mem for particular numa node
{"message": "No. of 1G huge pages for vswitch (soc-mem): 4096 does not match no. of NIC ports assigned to vswitch (available memory: 2048)for numa node: 0 "}

Returns following error if numa node count under socket-mem does not match with numa nodes specified in the nic inventory
{"message": "Numa nodes count do not match. socket-mem: (4096, 2048) Number of numa nodes specified in nic inventory: 1"}

Returns following errors if no. of cores specified for numa nodes is less than nic ports:
{"message": "No. of cores: 1 does not match no. of nic ports assigned to vswitch: 2 for numa node: 0"}
Error Code:
200 - Success
422 - Failure

Input:
JSON containing compute hostname, nictemplate details (returned by get nic template api), memorytemplate details (returned by get memory template api) and cputemplate details (returned by get cpu template api)
Refer to attached validate_compute_template.json

Create the payload (input json) manually as follows:
'{
"computename": "<Compute Host Name>",
"memorytemplate": {output of GET /templates/memory/<memorytempid>},
"cputemplate": {output of GET /templates/cpu/<cputempid>}
"nictemplate": {output of GET /templates/nic/<nictempid>}

}'

URI:
/compute/template/map/<MapId>

REST Method:
POST

Command:
curl -H "X-Auth-Token: $OS_TOKEN" -H "Content-Type: application/json" -X POST https://<clm-ip>:5001/compute/template/map -d '{ "nodeid": <nodeid>, "computetemplateid": <computetemplateid> }'

Purpose:
Maps Compute Node with ComputeTemplate, CPUTemplate, MemoryTemplate, DiskTemplate and NICTemplate
If the mapping already exists, this api would update existing mapping with provided computetemplateid (and associated component templateid's)

Output:
Json containing list of component templateid's
E.g. {"disktemplateid": 10, "nictemplateid": 7, "memorytemplateid": 4, "cputemplateid": 1}

Error Code:
200 - Success
422 - Failure

Input:
JSON containing Compute NodeId (Compute host) and Compute TemplateId to be associated

URI:
/compute/template/map/<nodeid>

REST Method:
GET

Command:
curl -H "X-Auth-Token: $OS_TOKEN" -H "Content-Type: application/json" -X GET https://<clm-ip>:5001/compute/template/map/<nodeid>

Purpose:
Gets the compute template details associated with specified nodeid.
(Output is similar to that of GET /compute/template/<computetemplateid>)

Output:
Json containing details for CPU Template, NIC Template, Memory Template and Disk Template associated with Compute Template

Error Code:
200 - Success
422 - Failure

Input:
Nodeid

```
URI:
/compute/template/map/

REST Method:
DELETE

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token:  $OS_TOKEN" -X DELETE
https://<clm-ip>:5001/compute/template/map/1

Purpose:
Deletes Compute Template-Compute Host mapping

Output:
Json containing success or failure message for the deleted Compute Template
E.g. {"message": "Compute Template has been deleted successfully"}

Error Code:
200 - Success
422 - Failure

Input:
JSON containing Map Id to be deleted
```

Enable Compute Node with Associated Templates

```
URI:
/computes/<compute_name>/enable

REST Method:
POST

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X POST
https://<clm-vip>:5001/computes/<compute-name>/enable

Purpose:
To enable any of or all the 4 [CPU, Memory, NIC and Disk] templates on a particular
Compute node.
Check hcgcrm database to find what templates are associated with the specific
compute node being enabled. This information can be found in DB's "NodetoTemplate"
table.

Output:
None

Verification:
At the start of Enable API execution:
 Admin State - Provisioned/User-Disabled
 Availability State - Online
```

Upon Successful execution of Enable API:
Admin State - Enabled
Availability State - Online

The moment the "Enable" operation is initiated, a specific "job ID" will be generated against every such job. Using the "job ID", the user can track the status of job using below command

GET---Specify the job ID"
URI:
/job/<job-id>

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET https://<clm-vip>:5001/job/<job-id>

Purpose:
To get the status of the job

Output:
JSON containing job status

Input:
Job ID

Example Commands :

For Enable API:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X POST https://10.90.2.30:5001/computes/hcglr4tb9b1053-cp1-comp0004-clm/enable

```
For querying Job Status:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET
https://10.90.2.30:5001/job/16
```

Disable Compute Node

```
URI:
/computes/<compute_name>/disable

REST Method:
POST

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X POST
https://<clm-vip>:5001/computes/<compute-name>/disable

Purpose:
To put a specific compute node into "user-disabled" mode for maintenance check or to
reconfigure the compute resources by associating to new set of template(s).

Output:
None

Verification:
At the start of Disable API execution:
 Admin State - Enabled
 Availability State - Online

Upon Successful execution of Disable API:
 Admin State - User-Disabled
 Availability State - Online

Once the "Disable" operation is initiated, a specific "job ID" will be generated
against every such job. Using the "job ID", the user can track the status of job
using below command

GET---Specify the job ID"
URI:
/job/<job-id>

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET
https://<clm-vip>:5001/job/<job-id>

Purpose:
To get the status of the job

Output:
JSON containing job status

Input:
```

Job ID

Output:
None

Example Command :

For Disable API:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X POST https://10.90.2.30:5001/computes/hcglr4tb9b1053-cp1-comp0004-clm/disable

For querying Job Status:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET https://10.90.2.30:5001/job/16

Please note that the SRIOV or PCIPT instance will end up in a ‘VERIFY_RESIZE’ state and not 'ACTIVE' state post cold migration.
The user will have to confirm the migration from UI or running CLI. This behavior will remain so in Alpha and will be fixed post Alpha.

As part of Disable API, all instances get migrated to a new HOST, if a single instance fails, the whole Disable operation is marked failed and the user has the choice for force disable then

Force Disable Compute Node

URI:
/computes/<compute_name>/forcedisable

REST Method:
POST

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X POST https://<clm-vip>:5001/computes/<compute-name>/forcedisable

Purpose:
To put a specific compute node into "user-disabled" mode for maintenance check or to reconfigure the compute resources by associating to new set of template(s).
The only difference between "Disable" and "Force disable" is that in case of "force disable", all the VM instances running on a specific compute node upon which the API would be exercised, will be "Shut Down" and then the node will be marked as "User-disabled"
Output:
None

Verification:
At the start of Force-Disable API execution:
 Admin State - Enabled
 Availability State - Online

Upon Successful execution of Force-Disable API:
 Admin State - User-Disabled
 Availability State - Online

Once the "forcedisable" operation is initiated, a specific "job ID" will be generated against the job. Using the "job ID", the user can track the status using below command

GET---Specify the job ID"
URI:
/job/<job-id>

REST Method:
GET

Command:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET https://<clm-vip>:5001/job/<job-id>

Purpose:
To get the status of the job

Output:

JSON containing job status

Input:
Job ID

Output:
None

Example Command :

For Force-Disable API:
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X POST https://10.90.2.30:5001/computes/hcglr4tb9b1053-cp1-comp0004-clm/forcedisable

For querying Job Status:

```
curl -i -H "Content-Type: application/json" -H "X-Auth-Token: $OS_TOKEN" -X GET
https://10.90.2.30:5001/job/19
```

4AA4-xxxxENW, Month 20XX