CPS Installation Guide for OpenStack, Release 18.4.0 (Restricted Release) (1)

First Published: 2018-09-14

Last Modified: 2018-10-05

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Preface

About This Guide

This document is a part of the Cisco Policy Suite documentation set.

For information about available documentation, see the *CPS Documentation Map* for this release at Cisco.com.

Audience

This guide is best used by these readers:

- Network administrators
- Network engineers
- Network operators
- System administrators

This document assumes a general understanding of network architecture, configuration, and operations.

Additional Support

For further documentation and support:

- Contact your Cisco Systems, Inc. technical representative.
- Call the Cisco Systems, Inc. technical support number.
- Write to Cisco Systems, Inc. at support@cisco.com.

- Refer to support matrix at https://www.cisco.com/c/en/us/support/index.html and to other documents related to Cisco Policy Suite.

Conventions (all documentation)

This document uses the following conventions.

Conventions	Indication
bold font	Commands and keywords and user-entered text appear in **bold** font.
italic font	Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font.
[ ]	Elements in square brackets are optional.
{x | y | z }	Required alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
courier font	Terminal sessions and information the system displays appear in courier font.
< >	Nonprinting characters such as passwords are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

Note Means reader take note. Notes contain helpful suggestions or references to material not covered in the manual.

Caution Means reader be careful. In this situation, you might perform an action that could result in equipment damage or loss of data.

Warning IMPORTANT SAFETY INSTRUCTIONS.

Means danger. You are in a situation that could cause bodily injury. Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents. Use the statement number provided at the end of each warning to locate its translation in the translated safety warnings that accompanied this device.

SAVE THESE INSTRUCTIONS

Note Regulatory: Provided for additional information and to comply with regulatory and customer requirements.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see What's New in Cisco Product Documentation.

To receive new and revised Cisco technical content directly to your desktop, you can subscribe to the What's New in Cisco Product Documentation RSS feed. RSS feeds are a free service.

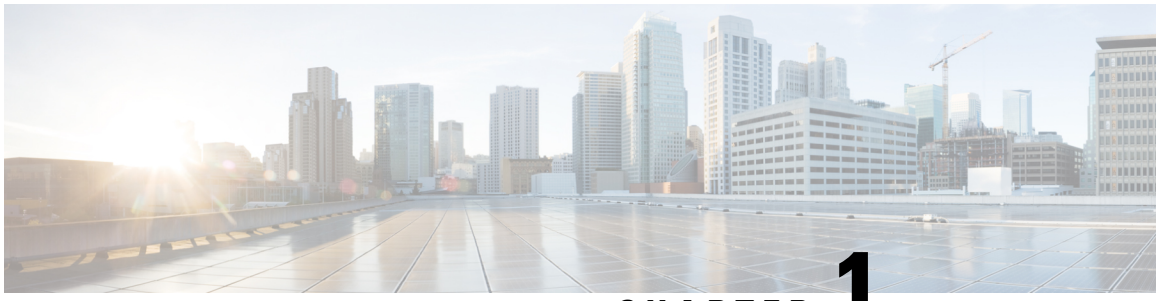
Important Notes

Important Any feature or GUI functionality that is not documented may not be supported in this release or may be customer specific, and must not be used without consulting your Cisco Account representative.

RESTRICTED RELEASE

Important This is a Short Term Support (STS) release with availability and use restrictions. Contact your Cisco Account or Support representatives for more information.

RESTRICTED RELEASE



CHAPTER 1

Preinstallation Tasks

Overview

Cisco Policy Suite offers a carrier-grade, high capacity, high performance, virtualized software solution, capable of running on VMware, OpenStack/KVM hypervisors or cloud infrastructures. To meet the stringent performance, capacity, and availability demands, the Cisco software requires that all allocated hardware system resources be 100% available when needed, and not oversubscribed or shared across separate VM's.

The following steps outline the basic process for a new installation of CPS:

Chapter 1:

1. Review virtual machine requirements
2. Orchestration Requirements
3. Install OpenStack
4. CPU Pinning
5. Configure OpenStack Users and Networks
6. Define Availability Zones
7. Download the required CPS images
8. Import images to Glance

9. Create Cinder Volumes
10. Verify or updated Default Quotas
11. Create Flavors
12. Set up Access and Security

Chapter 2:

1. Create CPS VMs using Nova Boot Commands
2. Create CPS VMs using Heat
3. Deploy CPS
4. Validate CPS Deployment
5. SR-IOV Support
6. Enable Custom Puppet to Configure Deployment
7. HTTPS Support for Orchestration API

Chapter 3:

1. Installation APIs
2. Upgrade APIs
    1. Unmount ISO
    2. Mount ISO
    3. Upgrade CPS
    4. Upgrade Status
3. System Configuration APIs

Virtual Machine Requirements

For customers operating a cloud infrastructure, the infrastructure must be configured to guarantee CPU, memory, network, and I/O availability for each CPS VM. Oversubscription of system resources will reduce the performance and capacity of the platform, and may compromise availability and response times. CPU core requirements are listed as pCPUs (physical cores) not vCPU's (hyper-threaded virtual cores).

In addition, the CPS carrier-grade platform requires:

- RAM reservation is enabled for all memory allocated to the CPS VM.
- CPU Hyperthreading must be ENABLED. To prevent over-subscription of CPU cores, CPU pinning should be ENABLED.
- CPU benchmark of at least 13,000 rating per chip and 1,365 rating per thread.
- The total number of VM CPU cores allocated should be 2 less than the total number of CPU cores per blade.

- Monitor the CPU STEAL statistic. This statistic should not cross 2% for more than 1 minute.

Note A high CPU STEAL value indicates the application is waiting for CPU, and is usually the result of CPU over allocation or no CPU pinning. CPS performance cannot be guaranteed in an environment with high CPU STEAL.

- CPU must be a high performance Intel x86 64-bit chipset.

Note BIOS settings should be set to high-performance values, rather than energy saving, hibernating, or speed stepping (contact hardware vendor for specific values).

- For deployments which cannot scale by adding more VM's, Cisco will support the allocation of additional CPU's above the recommendation to a single VM, but does not guarantee a linear performance increase.
- Cisco will not support performance SLA's for CPS implementations with less than the recommended CPU allocation.
- Cisco will not support performance SLA's for CPS implementations with CPU over-allocation (assigning more vCPU than are available on the blade, or sharing CPU's).
- Scaling and higher performance can be achieved by adding more VM's, not by adding more system resources to VM's.
- RAM latency should be lower than 15 nanosecond.
- RAM should be error-correcting ECC memory.
- Disk storage performance should be less than 2 millisecond average latency.
- Disk storage performance needs to support greater than 5000 input/output operations per second (IOPS) per CPS VM.
- Disk storage must provide redundancy and speed, such as RAID 0+1.
- Hardware and hardware design must be configured for better than 99.999% availability.
- For HA deployments, Cisco requires the customer designs comply with the Cisco CPS HA design guidelines.
  - At least two of each CPS VM type must be deployed: Policy Server (qns), Policy Director (lb), OAM (pcrfclient), Session Manager (sessionmgr).
  - Each CPS VM type must not share common HW zone with the same CPS VM type.
- The number of CPU cores, memory, NICs, and storage allocated per CPS VM must meet or exceed the requirements.

The following table provides information related to vCPU requirements based on:

- Hyper-threading: Enabled (Default)
- CPU Pinning: Enabled

- CPU Reservation: Yes (if allowed by hypervisor)
- Memory Reservation: Yes (if allowed)
- Hard Disk (in GB): 100

Table 1: HA Virtual Machine Requirements - Chassis Architecture

Physical Cores / Blade	VM Type	Memory (in GB)	Hard Disk (in GB)	vCPU	Configuration
Blade with 16 CPUs	Policy Server VMs (QNS)	16	100	12	Threading = 200 Mongo per host = 10 Criss-cross Mongo for Session Cache = 2 on each VM
Blade with 16 CPUs	Session Manager VMs	128	100	6	
Blade with 16 CPUs	Control Center (OAM) VMs	16	100	6	
Blade with 16 CPUs	Policy Director VMs (LB)	32	100	12	
Blade with 24 CPUs	Policy Server VMs (QNS)	16	100	10	Threading = 100 Mongo per host = 10 Criss-cross Mongo for Session Cache = 2 on each VM Hyper-threading = Default (Enable)
Blade with 24 CPUs	Session Manager VMs	80	100	8	
Blade with 24 CPUs	Control Center (OAM) VMs	16	100	8	
Blade with 24 CPUs	Policy Director VMs (LB)	32	100	12	

Table 2: HA Virtual Machine Requirements - Cloud Architecture

Physical Cores / Blade	VM Type	Memory (in GB)	Hard Disk (in GB)	vCPU	Configuration
Blade with 16 CPUs	Policy Server VMs (QNS)	16	100	12+	Threading = 200 Mongo per host = 10 Criss-cross Mongo for Session Cache = 2 on each VM
Blade with 16 CPUs	Session Manager VMs	128	100	6+	
Blade with 16 CPUs	Control Center (OAM) VMs	16	100	6+	
Blade with 16 CPUs	Policy Director VMs (LB)	32	100	8+	

Physical Cores / Blade	VM Type	Memory (in GB)	Hard Disk (in GB)	vCPU	Configuration
Blade with 24 CPUs	Policy Server VMs (QNS)	16	100	10+	Threading = 100 Mongo per host = 10 Criss-cross Mongo for Session Cache = 2 on each VM
Blade with 24 CPUs	Session Manager VMs	80	100	8+	
Blade with 24 CPUs	Control Center (OAM) VMs	16	100	8+	
Blade with 24 CPUs	Policy Director VMs (LB)	32	100	12+	

Note For large scale deployments having Policy Server (qns) VMs more than 35, Session Manager (sessionmgr) VMs more than 20, Policy Director (lb) VMs more than 2, recommended RAM for OAM (pcrfclient) VMs is 64GB.

Orchestration Requirements

The following orchestration capabilities are required to administer a CPS deployment in OpenStack:

- Ability to independently create/delete/re-create the Cluster Manager VM.
- Ability to snapshot Cluster Manager VM and restore the Cluster Manager VM from snapshot.
- Ability to attach and detach the ISO cinder volume to/from the Cluster Manager VM.
- CPS recommends that the CPS software ISO be mapped to a cinder volume. In deployments where this recommendation is used, prior to installation or upgrade or migration, the ISO cinder volume must be attached to the Cluster Manager so that the ISO can be mounted inside the Cluster Manager. The sample HEAT template provided in this document demonstrates how to automate mounting the ISO inside Cluster Manager. In deployments where this recommendation is not used, the CPS software ISO must be made available inside Cluster Manager VM and mounted using the method implemented by the customer.
- The Config drive must be used to pass in files such as userdata and the Config drive must be mounted to CPS VM in the 'iso9660' format.
- Any cinder volume required by the product code must be attached first to the VM and any customer environment specific cinder volumes should be attached after. One exception is the ISO cinder volume attached to Cluster Manager VM. In cases where ISO cinder volume is attached in a different order, the API to mount the ISO needs to be supplied with the right device name in the API payload.
- eth0 needs to be on the 'internal' network for inter-VM communication.
- On all CPS VMs, the Cluster Manager IP needs to be injected in /etc/hosts to ensure connectivity between each host and the Cluster Manager.
- CPS VM's role needs to be injected in `/etc/broadhop/.profile`, for example:

  ```
  NODE_TYPE=pcrfclient01
  ```

- For upgrades/migration and rollbacks, the orchestrator must have the ability to independently create/delete/re-create half/all of the following CPS VMs:

- Policy Server (qns)
- Policy Director (lb and iomanager)
- OAM (pcrfclient)
- Session Manager (sessionmgr)

  During a rollback, half of SM VMs must be deleted during Rollback procedure. As a result, the replica sets must be configured such that not all members of the replica set are in one half or the other. Replica set members must be distributed across the two upgrade sets so that the replica set is not entirely deleted. Refer to the *CPS Migration and Upgrade Guide* for more details.

- For scaling, the orchestrator must have the ability to independently create/delete/re-create half/all of the following CPS VMs in each scaling unit:
  - Policy Server (qns)
  - Session Manager (sessionmgr)

Install OpenStack

CPS is supported on the following OpenStack versions:

- OpenStack Liberty
- Newton
- Queens

CPS can also be installed on Cisco distributed platforms: Ultra B1.0 or Mercury 2.2.8

For more information about installing OpenStack and Cisco distributed platforms, refer to:

- OpenStack Liberty: http://docs.openstack.org/liberty/
- OpenStack Newton: https://docs.openstack.org/newton/
- OpenStack Queens: https://docs.openstack.org/queens/
- Ultra B1.0: https://www.cisco.com/c/en/us/solutions/service-provider/virtualized-packet-core/index.html
- Mercury 2.2.8:

  https://www.cisco.com/c/en/us/support/cloud-systems-management/nfv-infrastructure/tsd-products-support-series-home.html

Before you install OpenStack, you must perform some prerequisite tasks. The following sections describe these prerequisite tasks.

Note The example commands in the following sections are related to OpenStack Liberty. For commands related to other supported platforms, refer to the corresponding OpenStack documentation.

CPU Pinning

CPU pinning is supported and recommended in OpenStack deployments where hyperthreading is enabled. This enables CPS VMs to be pinned to dedicated physical CPU cores.

Refer to the following link for general instructions to enable CPU pinning for guest VMs: http://redhatstackblog.redhat.com/2015/05/05/cpu-pinning-and-numa-topology-awareness-in-openstack-compute/

Prerequisites

Make sure you have installed the following:

- OpenStack Liberty (OSP 7.2) or OpenStack Newton or OpenStack Queens or Ultra B1.0 or Mercury 2.2.8

Install numactl

The numactl package provides a command to examine the NUMA layout of the blades. Install this package on compute nodes to help determine which CPUs to set aside for pinning.

Run the following command to install numactl:

```
yum install numactl
```

Identify the Physical CPUs to Use for Pinning

Step 1 Run the following command on the compute nodes where you want to set aside physical CPUs for pinning.

```
numactl --hardware
```

```
[root@os6-compute-1 ~]# numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 8 9 10 11
node 0 size: 98245 MB
node 0 free: 87252 MB
node 1 cpus: 4 5 6 7 12 13 14 15
node 1 size: 98304 MB
node 1 free: 95850 MB
node distances:
node   0   1
  0:  10  20
  1:  20  10
```

Step 2 Determine the pool of CPUs you want to set aside for pinning.

At least 2 CPUs should be set aside for the Hypervisor in each node if it is a compute only blade. If the blade is operating as both a control and compute node, set aside more CPUs for OpenStack services.

Select the remaining CPUs for pinning.

In the above example, the following CPUs could be selected for pinning: **2, 3, 6, 7, 8-11, 12-15**.

Prevent Hypervisor from Using CPUs Set Aside for Pinning

To configure the hypervisor so that it will not use the CPUs identified for CPU Pinning:

Step 1 Open the KVM console for the Compute node.

Step 2 Execute the following command to update the kernal boot parameters:

```
grubby --update-kernel=ALL --args="isolcpus=2,3,6,7,8,9,10,11,12,13,14,15"
```

Step 3 Edit the `/etc/nova/nova.conf` file on that blade and set the `vcpu_pin_set` value to a list or range of physical CPU cores to reserve for virtual machine processes. For example:

```
vcpu_pin_set=2,3,6,7,8,9,10,11,12,13,14,15
```

Step 4 Reset the blade.

Step 5 After Linux has finished the boot process, enter the following command to verify the above Kernel boot options:

```
cat /proc/cmdline
```

The iso1cpus options you defined will be displayed, for example:

```
BOOT_IMAGE=/vmlinuz-3.10.0-327.el7.x86_64 root=/dev/mapper/cinder--volumes-slash ro rd.lvm.lv=rhel/swap
 crashkernel=auto rd.lvm.lv=cinder-volumes/slash rhgb quiet LANG=en_US.UTF-8
isolcpus=2,3,6,7,8,9,10,11,12,13,14,15
```

Configure Host Aggregates and Update Flavors

Step 1 Follow the instructions in the post (refer to Prerequisites, on page 7) to create host-aggregate, add compute hosts to the aggregate and set the CPU pinning metadata.

Step 2 Update Flavors which are NOT used for CPU pinning (non-CPS VM flavors) with the following command:

```
nova flavor-key <id> set "aggregate_instance_extra_specs:pinned"="false"
```

Step 3 Update Flavors which are CPS VM flavors (or a sub-set, if you are planning to bring up only certain VMs in CPS with CPU pinning) with the following command:

```
nova flavor-key <id> set hw:cpu_policy=dedicated
nova flavor-key <id> set aggregate_instance_extra_specs:pinned=true
```

Step 4 Launch a CPS VM with the performance enhanced Flavor. Note the host on which the instance is created and the instance name.

`nova show <id>` will show the host on which the VM is created in the field: `OS-EXT-SRV-ATTR:host`

`nova show <id>` will show the virsh Instance name in the field: `OS-EXT-SRV-ATTR:instance_name`

Step 5 To verify that vCPUs were pinned to the reserved physical CPUs, log in to the Compute node on which the VM is created and run the following command:

```
virsh dumpxml <instance_name>
```

The following section will show the physical CPUs in the field `cpuset` from the list of CPUs that were set aside earlier. For example:

```
<vcpu placement='static'>4</vcpu>
  <cputune>
    <shares>4096</shares>
    <vcpupin vcpu='0' cpuset='11'/>
    <vcpupin vcpu='1' cpuset='3'/>
    <vcpupin vcpu='2' cpuset='2'/>
    <vcpupin vcpu='3' cpuset='10'/>
    <emulatorpin cpuset='2-3,10-11'/>
  </cputune>
```

Configure OpenStack Users and Networks

You use keystone commands to create users. For more information about keystone commands, refer to the keystone command reference at: http://docs.openstack.org/cli-reference/index.html

Step 1 Create an OpenStack tenant with the name **`core`** (under which you can install the VMs) as shown in the following command:

```
source /root/keystonerc_admin

openstack project create --description "PCRF Admin" core
```

Step 2 For the above tenant, create an OpenStack user with the name **`core`** as shown in the following command:

```
source /root/keystonerc_admin

openstack user create --password "Core123" --email "core@cisco.com" --project core core
```

The tenant must have access to the following three provider VLANs. In this guide, the names of the VLANs are:

- Internal - Used for inter-VM communication of CPS VMs.
- Gx - Used by CPS to access the PCRF
- Management - Used to access the management functions of the CPS

Note Hosts in the CPS cluster can be configured to have IPv4 or IPv6 addresses. Currently, IPv6 is supported only for external interfaces. All alphabet characters used in virtual IPv6 addresses configured in csv files must be in small case letters.

You can specify any names.

Step 3 Set up the VLANs as shown in the following table:

Table 3: VLANs

Name	Subnet	VLAN-ID	Allocation Pool	Purpose
Internal	Specific to environment, for example:172.xx.xx.0/24	Specific to environment, for example: 20xx	Specific to environment, for example:172.xx.xx.16 to 172.xx.xx.220	The neutron network used by Cisco Policy Suite VMs for internal / private communication.
Management	Specific to environment, for example: 10.xx.xx.0/24	Specific to environment, for example: 20xx	Specific to environment, for example: 10.xx.xx.100 to 10.xx.xx.120	The neutron network used by Cisco Policy Suite VMs to connect on Management network.
Gx	Specific to environment, for example:192.xx.xx.0/24	Specific to environment, for example: 20xx	Specific to environment, for example: 192.xx.xx.16 to 192.xx.xx.220	The neutron network used by Cisco Policy Suite VMs to connect on Gx network.

The following example illustrates how to create networks and subnets:

```
source /root/keystonerc_admin
# $1 network name
# $2 vlan_id
# $3 gw ip
# $4 pool start
# $5 pool end
# $6 subnet x.x.x.x./y
# $OSTACKTENANT  core
#### Networks
echo "openstack network create --share --project $OSTACKTENANT --provider-network-type "vlan"
--provider-physical-network "physnet1" --provider-segment $2 "$1""
openstack network create --share --project $OSTACKTENANT --provider-network-type "vlan"
--provider-physical-network "physnet1" --provider-segment $2  "$1"
echo "openstack subnet create --no-dhcp --project $OSTACKTENANT --gateway $3 --subnet-range $6
--network $1 --allocation-pool start=$4,end=$5 $2""
openstack subnet create --no-dhcp --project $OSTACKTENANT --gateway $3 --subnet-range $6 --network
$1 --allocation-pool start=$4,end=$5 $2"
```

All of these networks must be either shared or accessible by Cisco Policy Suite OpenStack network.

Define Availability Zones

Step 1 A core user must have administrator role to launch VMs on specific hosts in OpenStack. Add the administrator role to the core user in the core tenant as shown in the following command:

```
openstack role add --project core --user core admin
```

Note The administrator role for the core user is not required if you do not intend to launch the VM on a specific host and if you prefer to allow nova to select the host for the VM.

Step 2 You must define at least one availability zone in OpenStack. Nova hypervisors list will show list of available hypervisors. For example:

```
[root@os24-control]# nova hypervisor-list
+----+---------------------------+
| ID | Hypervisor hostname       |
+----+---------------------------+
| 1  | os24-compute-2.cisco.com |
| 2  | os24-compute-1.cisco.com |
+----+---------------------------+
```

Step 3 Create availability zones specific to your deployment. The following commands provide an example of how to create the availability zones:

```
nova aggregate-create osXX-compute-1 az-1

nova aggregate-add-host osXX-compute-1 osXX-compute-1.cisco.com

nova aggregate-create osXX-compute-2 az-2

nova aggregate-add-host osXX-compute-2 osXX-compute-2.cisco.com
```

Note The above command creates two availability zones az-1 and az-2. You need to specify the zones az-1 or az-2 using Nova boot commands (see Create CPS VMs using Nova Boot Commands, on page 17), or in the Heat environment files (see Create CPS VMs using Heat, on page 22). You can also put more than one compute node in an availability zone. You could create az-1 with both blades, or in a 6-blade system, put three blades in each and then use `az-1:osXX-compute-2.cisco.com` to lock that VM onto that blade.

Availability zone for svn01 volume should be the same as that of pcrfclient01, svn02 volume as that of pcrfclient02, similarly for mongo01 and sessionmgr01, mongo02 and sessionmgr02. The same concept is applicable to cluman – the ISO volume and Cluster Manager (cluman) should be in same zone.

Step 4 Configure the compute nodes to create volumes on availability zones: Edit the `/etc/cinder/cinder.conf` file to add the `storage_availability_zone` parameter below the `[DEFAULT]` line. For example:

```
ssh root@os24-compute-1.cisco.com

 [DEFAULT]
storage_availability_zone=az-1:os24-compute-1.cisco.com
```

After adding the storage availability zone lines in `cinder.conf` file, restart the cinder volume with following command:

```
systemctl restart openstack-cinder-volume
```

Repeat Step 4, on page 11 for other compute nodes.

Download the ISO Image

Download the CPS ISO image file (CPS_x.x.x.release.iso) for the release from software.cisco.com and load it on the OpenStack control node.

Download the Base Image

CPS supports the QCOW2 image format for OpenStack installations. The QCOW2 base image is available to download as a separate file, and is not packaged inside the ISO.

Download the CPS QCOW2 base image file and extract it as shown in the following command:

```
tar -zxvf CPS_x.x.x_Base.qcow2.release.tar.gz
```

Locate the base image that is the root disk used by Cisco Policy Suite VM.

Import Images to Glance

Note The commands mentioned in this section are specific to OpenStack Liberty. For other OpenStack release specific commands, refer to https://releases.openstack.org/.

Import the Cisco Policy Suite base QCOW2 or VMDK image into the OpenStack glance repository.

To import the QCOW2 image, enter the following:

```
source /root/keystonerc_admin
```

```
glance image-create --name "<base vm name>" --visibility "<visibility>" --disk-format "qcow2"
--container "bare" --file <path of base qcow2>
```

To import the VMDK image, enter the following:

```
source /root/keystonerc_admin
```

```
glance image-create --name " <base vm name> " --visibility "<visibility>" --disk-format
"vmdk" --container "bare" --file <path of base vmdk>
```

Import the ISO image by running the following command:

```
source /root/keystonerc_admin
```

```
glance image-create --name "CPS_x.x.x.release.iso" --visibility "public" --disk-format "iso"
--container "bare" --file <path to iso file>
```

For more information on glance commands, refer to http://docs.openstack.org/cli-reference/glance.html.

Create Cinder Volumes

Create a cinder volume to map the glance image to the volume. This ensures that the cinder volume (and also the ISO that you imported to glance) can be automatically attached to the Cluster Manager VM when it is launched.

In the core tenant, create and format the following cinder volumes to be attached to various VMs:

- svn01
- svn02

- mongo01
- mongo02
- CPS_*x.x.x*.release.iso

It is recommended you work with Cisco AS to determine the size of each volume.

Note For mongo01 and mongo02, the minimum recommended size is 60 GB.

The following commands illustrate how to create the cinder volumes:

```
source /root/keystonerc_user
cinder create --metadata fstype=ext4 fslabel=newfs dio=yes --display-name svn01
--availability-zone az-1:os24-compute-1.cisco.com 2
cinder create --metadata fstype=ext4 fslabel=newfs dio=yes --display-name svn02
--availability-zone az-2:os24-compute-2.cisco.com 2
cinder create --metadata fstype=ext4 fslabel=newfs dio=yes --display-name mongo01
--availability-zone az-1:os24-compute-1.cisco.com 60
cinder create --metadata fstype=ext4 fslabel=newfs dio=yes --display-name mongo02
--availability-zone az-2:os24-compute-2.cisco.com 60
cps_iso_id=$(glance image-list | grep $cps_iso_name | awk ' {print $2}')
cinder create --display-name $cps_iso_name --image-id $cps_iso_id --availability-zone
az-1:os24-compute-1.cisco.com 3
```

Note

- Replace `$cps_iso_name` with the ISO filename. For example: `CPS_9.0.0.release.iso`
- If any host in the availability zone may be used, then only the zone needs to be specified. Currently, the recommendation only specifies `zone:host`

Verify or Update Default Quotas

OpenStack must have enough Default Quotas (that is, size of RAM, number of vCPUs, number of instances) to spin up all the VMs.

Update the Default Quotas in the following page of the OpenStack dashboard: **Admin** > **Defaults** > **Update Defaults**.

For example:

- instances: 20
- ram: 1024000
- cores: 100

Create Flavors

OpenStack flavors define the virtual hardware templates defining sizes for RAM, disk, vCPUs, and so on.

To create the flavors for your CPS deployment, run the following commands, replacing the appropriate values for your CPS deployment.

```
source /root/keystonerc_admin
   nova flavor-create --ephemeral 0 pcrfclient01 auto 16384 0 2
   nova flavor-create --ephemeral 0 pcrfclient02 auto 16384 0 2
   nova flavor-create --ephemeral 0 cluman auto 8192 0 4
   nova flavor-create --ephemeral 0 qps auto 10240 0 4
   nova flavor-create --ephemeral 0 sm auto 16384 0 4
   nova flavor-create --ephemeral 0 lb01 auto 8192 0 6
   nova flavor-create --ephemeral 0 lb02 auto 8192 0 6
```

Set up Access and Security

You must configure access to TCP and UDP ports.

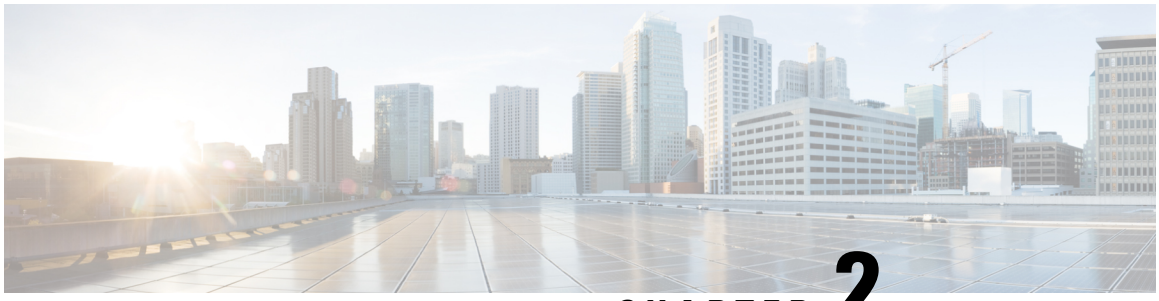
You can configure the access of the ports from the OpenStack dashboard (**Project** > **Access & Security** > **default / Manage Rules**) or using the CLI.

The following example illustrates configuration of the ports using the CLI:

```
source /root/keystonerc_user
openstack security group rule create default --protocol icmp --remote-ip 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 22:22 --remote-ip
0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 53:53 --remote-ip
0.0.0.0/0
openstack security group rule create default --protocol udp --dst-port 53:53 --remote-ip
0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 80:80 --remote-ip
0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 443:443 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 7443:7443 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 8443:8443 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 9443:9443 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 5540:5540 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 1553:1553 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 3868:3868 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 9160:9160 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 27717:27720 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 5432:5432 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 61616:61616 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 9443:9450 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 8280:8290 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 7070:7070 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 8080:8080 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 8090:8090 --remote-ip
```

```
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 7611:7611 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 7711:7711 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol udp --dst-port 694:694 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 10080:10080 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 11211:11211 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 111:111 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 27717:27720 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 2049:2049 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol udp --dst-port 2049:2049 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 32767:32767 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol udp --dst-port 32767:32767 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 2049:2049 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 2049:2049 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 8161:8161 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 12712:12712 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 9200:9200 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 2049:2049 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol udp --dst-port 5060:5060 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol tcp --dst-port 8548:8548 --remote-ip
 0.0.0.0/0
openstack security group rule create default --protocol udp --dst-port 8548:8548 --remote-ip
 0.0.0.0/0
```

where: default is the name of the security group.



CHAPTER 2

Installation

Installation Overview

Cisco Policy Suite VMs is deployed using either Nova boot commands or Heat templates.

Create CPS VMs using Nova Boot Commands

Step 1 Create cloud configuration files for each VM to be deployed (xxx-cloud.cfg). These configurations are used to define the OpenStack parameters for each CPS VM.

Refer to Sample Cloud Config Files, on page 20 to create these files.

Step 2 Run the following command on the control node:

```
source ~/keystonerc_core
```

Step 3 Deploy each CPS VM with the following nova boot command:

```
nova boot --config-drive true --user-data=<node>-cloud.cfg
 --image "base_vm" --flavor "<cluman|pcrfclient0x|sm|lb0x|qns0x>"
--nic net-id="<Internal n/w id>,v4-fixed-ip=
<Internal network private IP>"
--nic net-id="<Management network id>,v4-fixed-ip=
<Management n/w public ip>" --block-device-mapping
"/dev/vdb=<Volume id of iso>:::0"
--availability-zone "<availability zone:Host info>"
"cluman"
```

Note Configure the networks, internal IPs, management IPs and availability zones based on the requirements of your environment.

The following example shows the nova boot commands to deploy a Cluster Manager (cluman), two OAMs (pcrfclients), two sessionmgrs, two Policy Directors (load balancers), and four Policy Server (qns) VMs.

In the following example:

- 172.16.2.200 is the Internal VIP address.
- 172.18.11.156 is the management VIP address.
- 192.168.2.200 is the Gx VIP address

```
nova boot --config-drive true --user-data=cluman-cloud.cfg
--image "CPS_xx_x_x_Base" --flavor "cluman" --nic net-id=
"8c74819c-f3cb-46ad-b69a-d0d521b336d5,v4-fixed-ip=172.16.2.19"
--nic net-id="27a07da0-116f-4453-94b6-457bad9154b0,v4-fixed-ip=172.18.11.101"
--block-device-mapping "/dev/vdb=edf0113a-2ea0-4286-97f0-ee149f35b0d2:::0"
--availability-zone Zone1 "cluman"

nova boot --config-drive true --user-data=pcrfclient01-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "pcrfclient01" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.20" --nic
net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.152"
--block-device-mapping "/dev/vdb=139f2b90-eb74-4d5e-9e20-2af3876a7572:::0"
 --availability-zone "az-1:os8-compute-1.cisco.com" "pcrfclient01"

nova boot --config-drive true --user-data=pcrfclient02-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "pcrfclient02"  --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.21" --nic net-id=
"24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.153"
--block-device-mapping "/dev/vdb=27815c35-c5e8-463b-8ce4-fb1ec67d9446:::0"
--availability-zone "az-2:os8-compute-2.cisco.com" "pcrfclient02"

nova boot --config-drive true --user-data=sessionmgr01-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "sm"  --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.22"
--nic net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.157"
--block-device-mapping "/dev/vdb=8c3577d2-74f2-4370-9a37-7370381670e4:::0"
--availability-zone "az-1:os8-compute-1.cisco.com" "sessionmgr01"

nova boot --config-drive true --user-data=sessionmgr02-cloud.cfg
--image "base_vmCPS_xx_x_x_Base" --flavor "sm"
--nic net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.23"
--nic net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.158"
--block-device-mapping "/dev/vdb=67aa5cbd-02dd-497e-a8ee-797ac04b85f0:::0"
--availability-zone "az-2:os8-compute-2.cisco.com" "sessionmgr02"

nova boot --config-drive true --user-data=lb01-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "lb01" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.201"
--nic net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.154"
--nic net-id="d0a69b7f-5d51-424a-afbe-5f6486c6e90d,v4-fixed-ip=192.168.2.201"
--availability-zone "az-1:os8-compute-1.cisco.com" "lb01"

nova boot --config-drive true --user-data=lb02-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "lb02" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.202"
--nic net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.155"
--nic net-id="d0a69b7f-5d51-424a-afbe-5f6486c6e90d,v4-fixed-ip=192.168.2.202"
--availability-zone "az-2:os8-compute-2.cisco.com" "lb02"
```

```
nova boot --config-drive true --user-data=qns01-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "qps" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.24"
--availability-zone "az-1:os8-compute-1.cisco.com" "qns01"

nova boot --config-drive true --user-data=qns02-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "qps" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.25"
--availability-zone "az-1:os8-compute-1.cisco.com" "qns02"

nova boot --config-drive true --user-data=qns03-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "qps" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.26"
--availability-zone "az-2:os8-compute-2.cisco.com" "qns03"

nova boot --config-drive true --user-data=qns04-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "qps" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.27"
--availability-zone "az-2:os8-compute-2.cisco.com" "qns04"
```

Note Use the `cinder list` command to query OpenStack for the block-device-mapping IDs for the above nova boot commands.

Step 4 Update the ports to allow address pairing on the Neutron ports:

a) Use the following command to find the Neutron port ID for the lb01 internal IP address:

```
openstack port list | grep "<lb01_internal_IP>"
```

b) Use the following command to find the Neutron port ID for the lb02 internal IP address:

```
openstack port list | grep "<lb02_internal_IP>"
```

c) Update the above two Neutron ports to allow Internal VIP address by running the following command for each of the above ports:

```
openstack port set --allowed-address-pair ip_address=IP_ADDR|CIDR[,mac_address=MAC_ADDR]
```

For example:

```
[root@os8-control cloud(keystone_core)]# openstack port list | grep "172.16.2.201"
| db8944f3-407d-41ef-b063-eabbab43c039 || fa:16:3e:b1:f3:ab |
ip_address='172.16.2.201',subnet_id='6cfd1d1b-0931-44ad-bdc9-5015dc69f9d0' | ACTIVE |

[root@os8-control cloud(keystone_core)]# openstack port set --allowed-address-pairs
ip-address=172.16.2.200 db8944f3-407d-41ef-b063-eabbab43c039
```

d) Repeat Step c for External VIP addresses using neutron ports for the lb01/lb02 Management IP address and also Gx VIP address using neutron ports for lb01/lb02 Gx IP addresses.

Step 5 Wait approximately 10 minutes for the Cluster Manager VM to be deployed, then check the readiness status of the Cluster Manager VM using the following API:

GET http://<*Cluster Manager IP*>:8458/api/system/status/cluman

Refer to /api/system/status/cluman, on page 58 for more information.

When this API response indicates that the Cluster Manager VM is in a ready state (`"status": "ready"`), continue with Deploy CPS, on page 41.

Refer also to the `/var/log/cloud-init-output.log` on the Cluster Manager VM for deployment details.

Sample Cloud Config Files

For nova boot installation of CPS, you must create a cloud configuration file for each CPS VM to be deployed.

The following sections show an example Cluster Manager cloud configuration (`cluman-cloud.cfg`), and a pcrfclient01 cloud configuration (`pcrfclient01-cloud.cfg`).

These files must be placed in the directory in which you execute the nova launch commands, typically `/root/cps-install/`.

Cluster Manager Configuration File (for install type mobile)

```
#cloud-config
write_files:
 - path: /etc/sysconfig/network-scripts/ifcfg-eth0
   encoding: ascii
   content: |
     DEVICE=eth0
     BOOTPROTO=none
     NM_CONTROLLED=none
     IPADDR=172.16.2.19    <---- Internal IP to access via private IP
     NETMASK=255.255.255.0
     NETWORK=172.16.2.0    <------ Internal network
   owner: root:root
   permissions: '0644'
 - path: /etc/sysconfig/network-scripts/ifcfg-eth1
   encoding: ascii
   content: |
     DEVICE=eth1
     BOOTPROTO=none
     NM_CONTROLLED=none
     IPADDR=172.18.11.101   <---- Management IP to access via public IP
     NETMASK=255.255.255.0
     GATEWAY=172.18.11.1
     NETWORK=172.18.11.0
   owner: root:root
   permissions: '0644'
 - path: /var/lib/cloud/instance/payload/launch-params
   encoding: ascii
   owner: root:root
   permissions: '0644'
 - path: /root/.autoinstall.sh
   encoding: ascii
   content: |
     #!/bin/bash
     if [[ -d /mnt/iso ]] && [[ -f /mnt/iso/install.sh ]]; then
       /mnt/iso/install.sh << EOF
     mobile
     y
     1
     EOF
     fi
   permissions: '0755'
mounts:
 - [ /dev/vdb, /mnt/iso, iso9660, "auto,ro", 0, 0 ]
runcmd:
 - ifdown eth0
 - ifdown eth1
 - echo 172.16.2.19 installer >> /etc/hosts   <---- Internal/private IP of cluman
 - ifup eth0
 - ifup eth1
 - /root/.autoinstall.sh
```

Note If actual hostname for Cluster Manager VM is other than 'installer', then modify installer/cluman entry in `/etc/hosts` accordingly.

Example:

```
echo 172.16.2.19 installer <actual-hostname> >> /etc/hosts
```

Non-Cluster Manager Configuration File

- The following example configuration file is for pcrfclient01. You must create separate configuration files for each CPS VM to be deployed.

  For each file, modify the `NODE_TYPE`, and network settings (`IPADDR`, `GATEWAY`, `NETWORK`) accordingly.

  A typical CPS deployment would require the following files:

  - • pcrfclient01-cloud.cfg
    • pcrfclient02-cloud.cfg
    • lb01-cloud.cfg
    • lb02-cloud.cfg
    • sessionmgr01-cloud.cfg
    • sessionmgr02-cloud.cfg
    • qns01-cloud.cfg
    • qns02-cloud.cfg
    • qns03-cloud.cfg
    • qns04-cloud.cfg

    • pcrfclient01-cloud.cfg
    • pcrfclient02-cloud.cfg
    • lb01-cloud.cfg
    • lb02-cloud.cfg
    • sessionmgr01-cloud.cfg
    • sessionmgr02-cloud.cfg
    • qns01-cloud.cfg
    • qns02-cloud.cfg
    • qns03-cloud.cfg
    • qns04-cloud.cfg

- Modify `IPADDR` to the IP address used in nova boot command for that interface.

- Set `NETMASK`, `GATEWAY`, and `NETWORK` according to your environment.

```
#cloud-config
#hostname: pcrfclient01
fqdn: pcrfclient01
write_files:
 - path: /etc/sysconfig/network-scripts/ifcfg-eth0
   encoding: ascii
   content: |
     DEVICE=eth0
     BOOTPROTO=none
     NM_CONTROLLED=none
     IPADDR=172.16.2.20
     NETMASK=255.255.255.0
     NETWORK=172.16.2.0
   owner: root:root
   permissions: '0644'
 - path: /etc/sysconfig/network-scripts/ifcfg-eth1
   encoding: ascii
   content: |
     DEVICE=eth1
     BOOTPROTO=none
     NM_CONTROLLED=none
     IPADDR=172.18.11.152
     NETMASK=255.255.255.0
     GATEWAY=172.18.11.1
     NETWORK=172.18.11.0
   owner: root:root
   permissions: '0644'
 - path: /var/lib/cloud/instance/payload/launch-params
   encoding: ascii
   owner: root:root
   permissions: '0644'
 - path: /etc/broadhop.profile
   encoding: ascii
   content: "NODE_TYPE=pcrfclient01\n"
   owner: root:root
   permissions: '0644'
runcmd:
 - ifdown eth0
 - ifdown eth1
 - echo 172.16.2.19 installer >> /etc/hosts
 - ifup eth0
 - ifup eth1
 - sed -i '/^HOSTNAME=/d' /etc/sysconfig/network && echo HOSTNAME=pcrfclient01 >>
/etc/sysconfig/network
 - echo pcrfclient01 > /etc/hostname
- hostname pcrfclient01
```

Create CPS VMs using Heat

To create the CPS VMs using OpenStack Heat, you must first create an environment file and a Heat template containing information for your deployment.

These files include information about the ISO, base image, availability zones, management IPs, and volumes. Modify the sample files provided below with information for your deployment.

- Sample Heat Environment File, on page 23
- Sample Heat Template File, on page 24

After populating these files, continue with Create Heat Stack, on page 40.

Sample Heat Environment File

Note Update the network/vlan names, internal and management IPs, VIPs, and volumes for your environment.

`az-1`, `az-2` shown in the following sample are for example purposes only. Update these for your environment accordingly.

Also update the heat template (hot-cps.yaml) with your availability zone variables (for example: `cps_az_1`, `cps_az_2`) after updating this heat environment file.

```
# cat hot-cps.env
# This is an example environment file parameters:
  cps_iso_image_name: CPS_9.0.0.release.iso
  base_vm_image_name: CPS_9.0.0_Base.release
  cps_az_1: az-1
  cps_az_2: az-2

  internal_net_name: internal
  internal_net_cidr: 172.16.2.0/24

  management_net_name: management
  management_net_cidr: 172.18.11.0/24
  management_net_gateway: 172.18.11.1

  gx_net_name: gx
  gx_net_cidr: 192.168.2.0/24

  cluman_flavor_name: cluman
  cluman_internal_ip: 172.16.2.19
  cluman_management_ip: 172.18.11.151

  lb_internal_vip: 172.16.2.200
  lb_management_vip: 172.18.11.156
  lb_gx_vip: 192.168.2.200
  lb01_flavor_name: lb01
  lb01_internal_ip: 172.16.2.201
  lb01_management_ip: 172.18.11.154
  lb01_gx_ip: 192.168.2.201
  lb02_flavor_name: lb02
  lb02_internal_ip: 172.16.2.202
  lb02_management_ip: 172.18.11.155
  lb02_gx_ip: 192.168.2.202

  pcrfclient01_flavor_name: pcrfclient01
  pcrfclient01_internal_ip: 172.16.2.20
  pcrfclient01_management_ip: 172.18.11.152
  pcrfclient02_flavor_name: pcrfclient02
  pcrfclient02_internal_ip: 172.16.2.21
  pcrfclient02_management_ip: 172.18.11.153

  qns01_internal_ip: 172.16.2.24
  qns02_internal_ip: 172.16.2.25
  qns03_internal_ip: 172.16.2.26
  qns04_internal_ip: 172.16.2.27

  sessionmgr01_internal_ip: 172.16.2.22
  sessionmgr01_management_ip: 172.18.11.157
```

```
  sessionmgr02_internal_ip: 172.16.2.23
  sessionmgr02_management_ip: 172.18.11.158

  mongo01_volume_id: "54789405-f683-401b-8194-c354d8937ecb"
  mongo02_volume_id: "9694ab92-8ddd-407e-8520-8b0280f5db03"
  svn01_volume_id: "5b6d7263-40d1-4748-b45c-d1af698d71f7"
  svn02_volume_id: "b501f834-eff9-4044-90c3-a24378f3734d"
  cps_iso_volume_id: "ef52f944-411b-42b1-b86a-500950f5b398"
```

Sample Heat Template File

Note

- Update the following sample heat template according to your environment, such as to add more VMs, networks to the VMs, and so on.
- For more information on MOG/PATS, contact your Cisco Technical Representative.
- Currently, eSCEF is an EFT product and is for Lab Use Only. This means it is not supported by Cisco TAC and cannot be used in a production network. The features in the EFT are subject to change at the sole discretion of Cisco.

```
#cat hot-cps.yaml
heat_template_version: 2014-10-16

description: A minimal CPS deployment for big bang deployment

parameters:
#=========================
# Global Parameters
#=========================
  base_vm_image_name:
    type: string
    label: base vm image name
    description: name of the base vm as imported into glance
  cps_iso_image_name:
    type: string
    label: cps iso image name
    description: name of the cps iso as imported into glance
  cps_install_type:
    type: string
    label: cps installation type (mobile|mog|pats|arbiter|andsf|escef)
    description: cps installation type (mobile|mog|pats|arbiter|andsf|escef)
    default: mobile
  cps_az_1:
    type: string
    label: first availability zone
    description: az for "first half" of cluster
    default: nova
  cps_az_2:
    type: string
    label: second availability zone
    description: az for "second half" of cluster
    default: nova

#=========================
# Network Parameters
#=========================
  internal_net_name:
    type: string
```

```
    label: internal network name
    description: name of the internal network
  internal_net_cidr:
    type: string
    label: cps internal cidr
    description: cidr of internal subnet

  management_net_name:
    type: string
    label: management network name
    description: name of the management network
  management_net_cidr:
    type: string
    label: cps management cidr
    description: cidr of management subnet
  management_net_gateway:
    type: string
    label: management network gateway
    description: gateway on management network
    default: ""

  gx_net_name:
    type: string
    label: gx network name
    description: name of the gx network
  gx_net_cidr:
    type: string
    label: cps gx cidr
    description: cidr of gx subnet
  gx_net_gateway:
    type: string
    label: gx network gateway
    description: gateway on gx network
    default: ""

  cps_secgroup_name:
    type: string
    label: cps secgroup name
    description: name of cps security group
    default: cps_secgroup

#=========================
# Volume Parameters
#=========================
  mongo01_volume_id:
    type: string
    label: mongo01 volume id
    description: uuid of the mongo01 volume

  mongo02_volume_id:
    type: string
    label: mongo02 volume id
    description: uuid of the mongo02 volume

  svn01_volume_id:
    type: string
    label: svn01 volume id
    description: uuid of the svn01 volume

  svn02_volume_id:
    type: string
    label: svn02 volume id
    description: uuid of the svn02 volume
```

```
  cps_iso_volume_id:
    type: string
    label: cps iso volume id
    description: uuid of the cps iso volume

#=========================
# Instance Parameters
#=========================
  cluman_flavor_name:
    type: string
    label: cluman flavor name
    description: flavor cluman vm will use
    default: cluman
  cluman_internal_ip:
    type: string
    label: internal ip of cluster manager
    description: internal ip of cluster manager
  cluman_management_ip:
    type: string
    label: management ip of cluster manager
    description: management ip of cluster manager

  lb_internal_vip:
    type: string
    label: internal vip of load balancer
    description: internal vip of load balancer
  lb_management_vip:
    type: string
    label: management vip of load balancer
    description: management vip of load balancer
  lb_gx_vip:
    type: string
    label: gx ip of load balancer
    description: gx vip of load balancer
  lb01_flavor_name:
    type: string
    label: lb01 flavor name
    description: flavor lb01 vms will use
    default: lb01
  lb01_internal_ip:
    type: string
    label: internal ip of load balancer
    description: internal ip of load balancer
  lb01_management_ip:
    type: string
    label: management ip of load balancer
    description: management ip of load balancer
  lb01_gx_ip:
    type: string
    label: gx ip of load balancer
    description: gx ip of load balancer
  lb02_flavor_name:
    type: string
    label: lb02 flavor name
    description: flavor lb02 vms will use
    default: lb02
  lb02_internal_ip:
    type: string
    label: internal ip of load balancer
    description: internal ip of load balancer
  lb02_management_ip:
    type: string
    label: management ip of load balancer
    description: management ip of load balancer
```

```
lb02_gx_ip:
  type: string
  label: gx ip of load balancer
  description: gx ip of load balancer

pcrfclient01_flavor_name:
  type: string
  label: pcrfclient01 flavor name
  description: flavor pcrfclient01 vm will use
  default: pcrfclient01
pcrfclient01_internal_ip:
  type: string
  label: internal ip of pcrfclient01
  description: internal ip of pcrfclient01
pcrfclient01_management_ip:
  type: string
  label: management ip of pcrfclient01
  description: management ip of pcrfclient01
pcrfclient02_flavor_name:
  type: string
  label: pcrfclient02 flavor name
  description: flavor pcrfclient02 vm will use
  default: pcrfclient02
pcrfclient02_internal_ip:
  type: string
  label: internal ip of pcrfclient02
  description: internal ip of pcrfclient02
pcrfclient02_management_ip:
  type: string
  label: management ip of pcrfclient02
  description: management ip of pcrfclient02

qns_flavor_name:
  type: string
  label: qns flavor name
  description: flavor qns vms will use
  default: qps
qns01_internal_ip:
  type: string
  label: internal ip of qns01
  description: internal ip of qns01
qns02_internal_ip:
  type: string
  label: internal ip of qns02
  description: internal ip of qns02
qns03_internal_ip:
  type: string
  label: internal ip of qns03
  description: internal ip of qns03
qns04_internal_ip:
  type: string
  label: internal ip of qns04
  description: internal ip of qns04


sessionmgr_flavor_name:
  type: string
  label: sessionmgr flavor name
  description: flavor sessionmgr vms will use
  default: sm
sessionmgr01_internal_ip:
  type: string
  label: internal ip of sessionmgr01
  description: internal ip of sessionmgr01
```

```
  sessionmgr01_management_ip:
    type: string
    label: management ip of sessionmgr01
    description: management ip of sessionmgr01
  sessionmgr02_internal_ip:
    type: string
    label: internal ip of sessionmgr02
    description: internal ip of sessionmgr02
  sessionmgr02_management_ip:
    type: string
    label: management ip of sessionmgr02
    description: management ip of sessionmgr02

resources:
#=========================
# Instances
#=========================

  cluman:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_1 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: cluman_flavor_name }
      networks:
        - port: { get_resource: cluman_internal_port }
        - port: { get_resource: cluman_management_port }
      block_device_mapping:
        - device_name: vdb
          volume_id: { get_param: cps_iso_volume_id }
      user_data_format: RAW
      user_data: { get_resource: cluman_config }
  cluman_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
      fixed_ips: [{ ip_address: { get_param: cluman_internal_ip }}]
  cluman_management_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: management_net_name }
      fixed_ips: [{ ip_address: { get_param: cluman_management_ip }}]
  cluman_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
          - path: /var/lib/cloud/instance/payload/launch-params
            permissions: "0644"
          - path: /etc/sysconfig/network-scripts/ifcfg-eth0
            permissions: "0644"
            content:
              str_replace:
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: cluman_internal_ip }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth1
            permissions: "0644"
            content:
```

```
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: cluman_management_ip }
                $gateway: { get_param: management_net_gateway }
          - path: /root/.autoinstall.sh
            permissions: "0755"
            content:
              str_replace:
                template: |
                  #!/bin/bash
                  if [[ -d /mnt/iso ]] && [[ -f /mnt/iso/install.sh ]]; then
                  /mnt/iso/install.sh << EOF
                  $install_type
                  y
                  1
                  EOF
                  fi
                params:
                  $install_type: { get_param: cps_install_type }
        mounts:
          - [ /dev/vdb, /mnt/iso, iso9660, "auto,ro", 0, 0 ]
        runcmd:
          - str_replace:
              template: echo $ip installer >> /etc/hosts
              params:
                $ip: { get_param: cluman_internal_ip }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
              params:
                $cidr: { get_param: internal_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
              params:
                $cidr: { get_param: management_net_cidr }
          - ifdown eth0 && ifup eth0
          - ifdown eth1 && ifup eth1
          - echo HOSTNAME=cluman >> /etc/sysconfig/network
          - echo cluman > /etc/hostname
          - hostname cluman
          - /root/.autoinstall.sh

  lb01:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_1 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: lb01_flavor_name }
      networks:
        - port: { get_resource: lb01_internal_port }
        - port: { get_resource: lb01_management_port }
        - port: { get_resource: lb01_gx_port }
      user_data_format: RAW
      user_data: { get_resource: lb01_config }
  lb01_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
```

```
    fixed_ips: [{ ip_address: { get_param: lb01_internal_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_internal_vip }
lb01_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: lb01_management_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_management_vip }
lb01_gx_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: gx_net_name }
    fixed_ips: [{ ip_address: { get_param: lb01_gx_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_gx_vip }
lb01_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=lb01\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: lb01_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: lb01_management_ip }
                $gateway: { get_param: management_net_gateway }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth2
          content:
            str_replace:
              template: |
                DEVICE=eth2
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: lb01_gx_ip }
                $gateway: { get_param: gx_net_gateway }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
```

```
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: management_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
            params:
              $cidr: { get_param: gx_net_cidr }
        - ifdown eth0 && ifup eth0
        - ifdown eth1 && ifup eth1
        - ifdown eth2 && ifup eth2
        - echo HOSTNAME=lb01 >> /etc/sysconfig/network
        - echo lb01 > /etc/hostname
        - hostname lb01

lb02:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: lb02_flavor_name }
    networks:
      - port: { get_resource: lb02_internal_port }
      - port: { get_resource: lb02_management_port }
      - port: { get_resource: lb02_gx_port }
    user_data_format: RAW
    user_data: { get_resource: lb02_config }
lb02_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: lb02_internal_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_internal_vip }
lb02_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: lb02_management_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_management_vip }
lb02_gx_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: gx_net_name }
    fixed_ips: [{ ip_address: { get_param: lb02_gx_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_gx_vip }
lb02_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=lb02\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
```

```
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: lb02_internal_ip }
              - path: /etc/sysconfig/network-scripts/ifcfg-eth1
                content:
                  str_replace:
                    template: |
                      DEVICE=eth1
                      BOOTPROTO=none
                      NM_CONTROLLED=no
                      IPADDR=$ip
                      GATEWAY=$gateway
                    params:
                      $ip: { get_param: lb02_management_ip }
                      $gateway: { get_param: management_net_gateway }
              - path: /etc/sysconfig/network-scripts/ifcfg-eth2
                content:
                  str_replace:
                    template: |
                      DEVICE=eth2
                      BOOTPROTO=none
                      NM_CONTROLLED=no
                      IPADDR=$ip
                      GATEWAY=$gateway
                    params:
                      $ip: { get_param: lb02_gx_ip }
                      $gateway: { get_param: gx_net_gateway }
            runcmd:
              - str_replace:
                  template: echo $ip installer >> /etc/hosts
                  params:
                    $ip: { get_param: cluman_internal_ip }
              - str_replace:
                  template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
                  params:
                    $cidr: { get_param: internal_net_cidr }
              - str_replace:
                  template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
                  params:
                    $cidr: { get_param: management_net_cidr }
              - str_replace:
                  template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
                  params:
                    $cidr: { get_param: gx_net_cidr }
              - ifdown eth0 && ifup eth0
              - ifdown eth1 && ifup eth1
              - ifdown eth2 && ifup eth2
              - echo HOSTNAME=lb02 >> /etc/sysconfig/network
              - echo lb02 > /etc/hostname
              - hostname lb02

      pcrfclient01:
        type: OS::Nova::Server
        properties:
          availability_zone: { get_param: cps_az_1 }
          config_drive: "True"
          image: { get_param: base_vm_image_name }
          flavor: { get_param: pcrfclient01_flavor_name }
          networks:
            - port: { get_resource: pcrfclient01_internal_port }
```

```
      - port: { get_resource: pcrfclient01_management_port }
    block_device_mapping:
      - device_name: vdb
        volume_id: { get_param: svn01_volume_id }
    user_data_format: RAW
    user_data: { get_resource: pcrfclient01_config }
pcrfclient01_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: pcrfclient01_internal_ip }}]
pcrfclient01_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: pcrfclient01_management_ip }}]
pcrfclient01_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=pcrfclient01\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: pcrfclient01_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: pcrfclient01_management_ip }
                $gateway: { get_param: management_net_gateway }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: management_net_cidr }
        - ifdown eth0 && ifup eth0
        - ifdown eth1 && ifup eth1
        - echo HOSTNAME=pcrfclient01 >> /etc/sysconfig/network
        - echo pcrfclient01 > /etc/hostname
        - hostname pcrfclient01
```

```
pcrfclient02:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: pcrfclient02_flavor_name }
    networks:
      - port: { get_resource: pcrfclient02_internal_port }
      - port: { get_resource: pcrfclient02_management_port }
    block_device_mapping:
      - device_name: vdb
        volume_id: { get_param: svn02_volume_id }
    user_data_format: RAW
    user_data: { get_resource: pcrfclient02_config }
pcrfclient02_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: pcrfclient02_internal_ip }}]
pcrfclient02_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: pcrfclient02_management_ip }}]
pcrfclient02_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=pcrfclient02\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: pcrfclient02_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: pcrfclient02_management_ip }
                $gateway: { get_param: management_net_gateway }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
```

```
              $cidr: { get_param: internal_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
              params:
                $cidr: { get_param: management_net_cidr }
          - ifdown eth0 && ifup eth0
          - ifdown eth1 && ifup eth1
          - echo HOSTNAME=pcrfclient02 >> /etc/sysconfig/network
          - echo pcrfclient01 > /etc/hostname
          - hostname pcrfclient02

qns01:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_1 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns01_internal_port }
    user_data_format: RAW
    user_data: { get_resource: qns01_config }
qns01_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns01_internal_ip }}]
qns01_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns01\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns01_internal_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - ifdown eth0 && ifup eth0
        - echo HOSTNAME=qns01 >> /etc/sysconfig/network
        - echo qns01 > /etc/hostname
        - hostname qns01

qns02:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_1 }
    config_drive: "True"
```

```
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns02_internal_port }
    user_data_format: RAW
    user_data: { get_resource: qns02_config }
qns02_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns02_internal_ip }}]
qns02_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns02\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns02_internal_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - ifdown eth0 && ifup eth0
        - echo HOSTNAME=qns02 >> /etc/sysconfig/network
        - echo qns02 > /etc/hostname
        - hostname qns02

qns03:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns03_internal_port }
    user_data_format: RAW
    user_data: { get_resource: qns03_config }
qns03_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns03_internal_ip }}]
qns03_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
```

```
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns03\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns03_internal_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - ifdown eth0 && ifup eth0
        - echo HOSTNAME=qns03 >> /etc/sysconfig/network
        - echo qns03 > /etc/hostname
        - hostname qns03

qns04:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns04_internal_port }
    user_data_format: RAW
    user_data: { get_resource: qns04_config }
qns04_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns04_internal_ip }}]
qns04_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns04\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns04_internal_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
```

```
            params:
              $ip: { get_param: cluman_internal_ip }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
              params:
                $cidr: { get_param: internal_net_cidr }
          - ifdown eth0 && ifup eth0
          - echo HOSTNAME=qns04 >> /etc/sysconfig/network
          - echo qns04 > /etc/hostname
          - hostname qns04

  sessionmgr01:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_1 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: sessionmgr_flavor_name }
      networks:
        - port: { get_resource: sessionmgr01_internal_port }
        - port: { get_resource: sessionmgr01_management_port }
      block_device_mapping:
        - device_name: vdb
          volume_id: { get_param: mongo01_volume_id }
      user_data_format: RAW
      user_data: { get_resource: sessionmgr01_config }
  sessionmgr01_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
      fixed_ips: [{ ip_address: { get_param: sessionmgr01_internal_ip }}]
  sessionmgr01_management_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: management_net_name }
      fixed_ips: [{ ip_address: { get_param: sessionmgr01_management_ip }}]
  sessionmgr01_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
          - path: /var/lib/cloud/instance/payload/launch-params
          - path: /etc/broadhop.profile
            content: "NODE_TYPE=sessionmgr01\n"
          - path: /etc/sysconfig/network-scripts/ifcfg-eth0
            content:
              str_replace:
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: sessionmgr01_internal_ip }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth1
            content:
              str_replace:
                template: |
                  DEVICE=eth1
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                  GATEWAY=$gateway
                params:
```

```
                $ip: { get_param: sessionmgr01_management_ip }
                $gateway: { get_param: management_net_gateway }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: management_net_cidr }
        - ifdown eth0 && ifup eth0
        - ifdown eth1 && ifup eth1
        - echo HOSTNAME=sessionmgr01 >> /etc/sysconfig/network
        - echo sessionmgr01 > /etc/hostname
        - hostname sessionmgr01

sessionmgr02:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: sessionmgr_flavor_name }
    networks:
      - port: { get_resource: sessionmgr02_internal_port }
      - port: { get_resource: sessionmgr02_management_port }
    block_device_mapping:
      - device_name: vdb
        volume_id: { get_param: mongo02_volume_id }
    user_data_format: RAW
    user_data: { get_resource: sessionmgr02_config }
sessionmgr02_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: sessionmgr02_internal_ip }}]
sessionmgr02_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: sessionmgr02_management_ip }}]
sessionmgr02_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=sessionmgr02\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: sessionmgr02_internal_ip }
```

```
  - path: /etc/sysconfig/network-scripts/ifcfg-eth1
    content:
      str_replace:
        template: |
          DEVICE=eth1
          BOOTPROTO=none
          NM_CONTROLLED=no
          IPADDR=$ip
          GATEWAY=$gateway
        params:
          $ip: { get_param: sessionmgr02_management_ip }
          $gateway: { get_param: management_net_gateway }
runcmd:
  - str_replace:
      template: echo $ip installer >> /etc/hosts
      params:
        $ip: { get_param: cluman_internal_ip }
  - str_replace:
      template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
      params:
        $cidr: { get_param: internal_net_cidr }
  - str_replace:
      template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
      params:
        $cidr: { get_param: management_net_cidr }
  - ifdown eth0 && ifup eth0
  - ifdown eth1 && ifup eth1
  - echo HOSTNAME=sessionmgr02 >> /etc/sysconfig/network
  - echo sessionmgr02 > /etc/hostname
  - hostname sessionmgr02
```

Create Heat Stack

Before beginning, verify you have populated your information in the environment (.env) file and heat template (.yaml) file and loaded both files on the control node.

Step 1 Run the following command on control node at the location where your environment and heat template files are located:

```
source ~/keystonerc_core
```

Step 2 Add/assign the heat stack owner to core tenant user:

```
openstack role add --project core --user core admin
```

Step 3 Verify that no existing CPS stack is present:

```
[root@os8-control ~(keystone_core)]# heat stack-list

+--------------------------------------+------------+-----------------+----------------------+
| id                                   | stack_name | stack_status    | creation_time        |
+--------------------------------------+------------+-----------------+----------------------+
+--------------------------------------+------------+-----------------+----------------------+
```

Step 4 Create the stack using the heat template (hot-cps.yaml) and environment file (hot-cps.env) you populated earlier.

```
[root@os8-control mbuild(keystone_core)]# heat stack-create --environment-file hot-cps.env
--template-file hot-cps.yaml cps
+--------------------------------------+------------+--------------------+----------------------+
| id                                   | stack_name | stack_status       | creation_time        |
+--------------------------------------+------------+--------------------+----------------------+
```

```
| 3f1ab6c2-673d-47b3-ae01-8946cac9e9e9 | cps        | CREATE_IN_PROGRESS | 2016-03-03T16:58:53Z |
+--------------------------------------+------------+--------------------+----------------------+
```

Step 5 Check the status using the `heat stack-list` command:

```
[root@os8-control mbuild(keystone_core)]# heat stack-list
+--------------------------------------+------------+-----------------+----------------------+
| id                                   | stack_name | stack_status    | creation_time        |
+--------------------------------------+------------+-----------------+----------------------+
| 3f1ab6c2-673d-47b3-ae01-8946cac9e9e9 | cps        | CREATE_COMPLETE | 2016-01-19T16:58:53Z |
+--------------------------------------+------------+-----------------+----------------------+
```

`CREATE_COMPLETE` will be reported when the heat stack is finished.

Step 6 Wait approximately 10 minutes for the Cluster Manager VM to be deployed, then check the readiness status of the Cluster Manager VM using the following API:

GET http://<*Cluster Manager IP*>:8458/api/system/status/cluman

Refer to /api/system/status/cluman, on page 58 for more information.

When this API responds that the Cluster Manager VM is in a ready state (`"status": "ready"`), continue with Deploy CPS, on page 41.

Refer also to the `/var/log/cloud-init-output.log` on the Cluster Manager VM for deployment details.

Deploy CPS

The following steps outline how to create a consolidated CPS configuration file and use the CPS platform orchestration APIs to deploy the CPS VMs on OpenStack:

Step 1 Create a consolidated CPS configuration file. This file contains all the information necessary to deploy VMs in the CPS cluster, including a valid CPS license key. Contact your Cisco representative to receive the CPS license key for your deployment.

Note Cisco Smart Licensing is supported for CPS 10.0.0 and later releases. For information about what Smart Licensing is and how to enable it for CPS, refer to *CPS Operations Guide*.

a) Refer to Sample YAML Configuration File - HA Setup, on page 80 for a sample CPS configuration to use as a template.
b) Refer to Configuration Parameters - HA System, on page 62 for a description of all parameters within this file.

Important Verify that all VM IP addresses and host names are configured properly in the YAML and Heat template files. You cannot modify the IP addresses or host names manually on the VMs (excluding Cluster Manager) after deploying the VMs, and CPS does not support modification of IP addresses or host names of deployed VMs.

Step 2 Load the consolidated configuration file you created in Step 1, on page 41 using the following API:

POST http://<*Cluster Manager IP*>:8458/api/system/config/

For example:

```
curl -v -X POST --data-binary @CPS_config_yaml.txt -H "Content-type: application/yaml"
http://x.x.x.x:8458/api/system/config/
```

Refer to /api/system/config/, on page 58 for more information.

Step 3 (Optional) To confirm the configuration was loaded properly onto the Cluster Manager VM, perform a GET with the same API:

GET http://<*Cluster Manager IP*>:8458/api/system/config/

Step 4 Apply the configuration using the following API:

POST http://<*Cluster Manager IP*>:8458/api/system/config/action/apply

For example:

```
curl -v -X POST -H  "Content-type: application/json" http://x.x.x.x:8458/api/system/config/action/apply
```

Refer to /api/system/config/, on page 58 for more information.

This API applies the CPS configuration file, triggers the Cluster Manager VM to deploy and bring up all CPS VMs, and performs all post-installation steps.

Important The VMs are rebooted in rescue mode for the first time for CentOS to adjust disk/hardware to the new version. Subsequent reboots if necessary is a normal operation.

Step 5 Run `change_passwd.sh` script on Cluster Manager to change the password of root user across the system.

For more information, refer to *Update Default Credentials* in *CPS Installation Guide for VMware*.

What to do next

To enable the feature **Disable Root SSH Login**, check whether there exists a user with uid 1000 on Cluster Manager.

Use the following command to check there exists a user with uid 1000:

```
cat /etc/passwd | grep x:1000
```

If a user with uid 1000 exists on the Cluster Manager, change the uid on the Cluster Manager by executing the following command:

```
usermod -u <new-uid> <user-name-with-uid-as-1000>
```

This is done because the feature **Disable Root SSH Login** creates a user with uid 1000.

Validate CPS Deployment

Step 1 To monitor the status of the deployment, use the following API:

GET http://<*Cluster Manager IP*>:8458/api/system/config/status

Refer to /api/system/config/status , on page 88 for more information.

Step 2 After the deployment has completed, verify the readiness of the entire CPS cluster using the following API:

GET http://<*Cluster Manager IP*>:8458/api/system/status/cps

Refer to /api/system/status/cps, on page 90 for more information.

Step 3 Connect to the Cluster Manager and issue the following command to run a set of diagnostics and display the current state of the system.

```
/var/qps/bin/diag/diagnostics.sh
```

What to do next

Important After the validation is complete, take a backup of the Cluster Manager configuration. For more information on taking the backup, refer to *CPS Backup and Restore Guide*. In case the Cluster Manager gets corrupted this backup can be used to recover the Cluster Manager.

Troubleshooting

- CPS clusters deployed using the orchestration APIs report the following licensing errors in `/var/log/broadhop/qns.log` on the OAM (pcrfclient) VMs:

  ```
  [LicenseManagerTimer] ERROR c.b.licensing.impl.LicenseManager - Unable to load the
  license file.  Server is not licensed!
  ```

  This error can be ignored.

SR-IOV Support

CPS supports single root I/O virtualization (SR-IOV) on Intel NIC adapters.

CPS also supports bonding of SR-IOV sub-interfaces for seamless traffic switchover.

The Intel SR-IOV implementation includes anti-spoofing support that will not allow MAC addresses other than the one configured in the VF to communicate. As a result, the active failover mac policy is used.

To support seamless failover of interfaces, the VLAN interfaces should be created directly on top of the VF interfaces (for example, eth0.123 and eth1.123) and then those interfaces are bonded. If VLAN interfaces are created on top of a bond, their MAC address will not follow the bonds when a failover occurs and the old MAC will be used for the new active interface.

The following sample configuration shows the bonding of two interfaces using a single IP address:

```
[root@qns0x ~]# cat /proc/net/bonding/bond0310
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup) (fail_over_mac active)
Primary Slave: None
Currently Active Slave: eth1.310
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1.310
MII Status: up
Speed: 10000 Mbps
Duplex: full
```

```
Link Failure Count: 1
Permanent HW addr: fa:16:3e:aa:a5:c8
Slave queue ID: 0

Slave Interface: eth2.310
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 1
Permanent HW addr: fa:16:3e:26:e3:9e
Slave queue ID: 0
[root@qns02 ~]# cat /proc/net/bonding/bond0736
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup) (fail_over_mac active)
Primary Slave: None
Currently Active Slave: eth1.736
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1.736
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 1
Permanent HW addr: fa:16:3e:aa:a5:c8
Slave queue ID: 0

Slave Interface: eth2.736
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 1
Permanent HW addr: fa:16:3e:26:e3:9e
Slave queue ID: 0

[root@qns0x ~]# more /etc/sysconfig/network-scripts/ifcfg-*
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-bond0310
::::::::::::::
DEVICE=bond0310
BONDING_OPTS="mode=active-backup miimon=100 fail_over_mac=1"
TYPE=Bond
BONDING_MASTER=yes
BOOTPROTO=none
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV6INIT=no
IPADDR=172.16.255.11
NETMASK=255.255.255.192
NETWORK=172.16.255.0
IPV4_FAILURE_FATAL=no
IPV6INIT=no
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
ONBOOT=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-bond0736
```

```
::::::::::::::
DEVICE=bond0736
BONDING_OPTS="mode=active-backup miimon=100 fail_over_mac=1"
TYPE=Bond
BONDING_MASTER=yes
BOOTPROTO=none
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV6INIT=yes
IPV6ADDR=fd00:4888:1000:30c2::23/64
IPV6_DEFAULTGW=fd00:4888:1000:30c2::1
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=no
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
ONBOOT=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth0
::::::::::::::
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
IPADDR=192.168.66.34
NETMASK=255.255.255.0
NETWORK=192.168.66.0
IPV6INIT=yes
IPV6ADDR=fd00:4888:1000:f000::aab1/64
IPV6_DEFAULTGW=fd00:4888:1000:f000::1
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth1
::::::::::::::
DEVICE=eth1
TYPE=Ethernet
ONBOOT=yes
BOOTPROTO=none
USRCTL=no
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth1.310
::::::::::::::
DEVICE=eth1.310
ONBOOT=yes
MASTER=bond0310
BOOTPROTO=none
USRCTL=no
SLAVE=yes
VLAN=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth1.736
::::::::::::::
DEVICE=eth1.736
ONBOOT=yes
MASTER=bond0736
BOOTPROTO=none
USRCTL=no
SLAVE=yes
VLAN=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth2
::::::::::::::
DEVICE=eth2
```

```
ONBOOT=yes
BOOTPROTO=none
USRCTL=no
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth2.310
::::::::::::::
DEVICE=eth2.310
ONBOOT=yes
MASTER=bond0310
BOOTPROTO=none
USRCTL=no
SLAVE=yes
VLAN=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth2.736
::::::::::::::
DEVICE=eth2.736
ONBOOT=yes
MASTER=bond0736
BOOTPROTO=none
USRCTL=no
SLAVE=yes
VLAN=yes
```

Consistent Network Device Naming

CPS instances require that network interfaces be assigned IP addresses statically. The names of network interfaces (eth0, eth1, and so on) are assumed to reflect network interfaces representing neutron ports passed to OpenStack nova-boot or heat template in that order. In this case, eth0 is assumed to reflect the first neutron port, eth1 the second, and so on.

For CPS deployments on OpenStack which use SR-IOV, often two or more network drivers are used. When more than one network driver is used, network interface names can become unpredictable and can change based on the order in which the network drivers are loaded into the kernel.

The following section describes how to map a network interface for a given network drivers type to its correct expected name in the guest OS.

Requirements:

- Correct IP address assignment requires that network names used in the network interfaces file must match the name of the network interface in the guest OS.
- The order of neutron ports of a given type (non-SR-IOV or SR-IOV) in nova-boot or heat template directly maps to the order of the PCI device slot of the associated network interfaces in the guest OS.
- The mapping between the network interface of a given network driver type and network driver name are passed during the creation of an instance through the cloud-init configuration.

The expected network interface name configuration is passed into CPS instance's guest OS using a YAML format configuration file located at: `/var/lib/cloud/instance/payload/ifrename.yaml`.

The file should have a section for each driver type and list the interfaces for that driver type with the following information:

- Rank order (0, 1, 2…) for the interface among other interfaces of the same driver type, as is specified in the nova boot command/heat template
- Expected name of the interface (eth0, eth1, eth2 etc.)

For example:

```
- path: /var/lib/cloud/instance/payload/ifrename.yaml
  encoding: ascii
  owner: root:root
  permissions: '0644'
  content:  |
    ---
      -  virtio_net
        0 : eth0
      -  ixgbevf:
        0 : eth1
        1 : eth2
```

Driver names for SR-IOV ports can be determined by checking the interface card vendor documentation. For regular virtio ports, the driver name is 'virtio_net'.

This `ifrename.yaml` file must be added in the existing write_files: section of cloud-init configurations for each CPS VM.

The configuration file above instructs cloud-init to create a file `ifrename.yaml` at `/var/lib/cloud/instance/payload`, owned by root, with permissions of 644 and contents as mentioned in "content:" section. In this example:

- the first SR-IOV neutron port (managed by 'ixgbevf' driver) is mapped to to eth1
- the second SR-IOV port (managed by 'ixgbevf' driver) is mapped to eth2
- the only non-SR-IOV port (managed by 'virtio-net' driver) to eth0.

Regardless of the order in which neutron ports are passed, or order in which network drivers are loaded, this configuration file specifies which network interface name should go to which network interface.

Enable Custom Puppet to Configure Deployment

Some customers may need to customize the configuration for their deployment. When customizing the CPS configuration, it is important to make the customization in a way that does not impact the normal behavior for VM deployment and redeployment, upgrades/migration, and rollbacks.

For this reason, customizations should be placed in the `/etc/puppet/env_config` directory. Files within this directory are given special treatment for VM deployment, upgrade, migrations, and rollback operations.

Note If system configurations are manually changed in the VM itself after the VM has been deployed, these configurations will be overridden if that VM is redeployed.

The following section describes the steps necessary to make changes to the puppet installer.

Customizations of the CPS deployment are dependent on the requirements of the change. Examples of customizations include:

- deploying a specific facility on a node (VM)
- overriding a default configuration.

To explain the process, let us consider that we modify all VMs built from an installer, so we use the Policy Server (QNS) node definition.

For the above mentioned example, add custom routes via the examples42-network Puppet module. (For more information on the module, refer to https://forge.puppetlabs.com/example42/network).

Step 1 Make sure that the proper paths are available:

```
mkdir -p /etc/puppet/env_config/nodes
```

Step 2 Install the necessary Puppet module. For example:

```
puppet module install \
--modulepath=/etc/puppet/env_config/modules:/etc/puppet/modules \
example42-network
Notice: Preparing to install into /etc/puppet/env_config/modules ...
Notice: Downloading from https://forge.puppetlabs.com ...
Notice: Installing -- do not interrupt ...
/etc/puppet/env_config/modules
example42-network (v3.1.13)
```

Note For more information on installing and updating Puppet modules, refer to https://docs.puppetlabs.com/puppet/latest/reference/modules_installing.html.

Step 3 Copy the existing node definition into the env_config nodes:

```
cp /etc/puppet/modules/qps/nodes/qps.yaml \

/etc/puppet/env_config/nodes
```

Step 4 Add a reference to your custom Puppet manifest:

```
echo ' custom::static_routes:' >> \

/etc/puppet/env_config/nodes/qps.yaml
```

Step 5 Create your new manifest for static routes:

```
cat
 >/etc/puppet/env_config/modules/custom/manifests/static_routes.pp <<EOF class custom::static_routes
 {
   network::route {'eth0':
     ipaddress => ['192.168.1.0',],
     netmask   => ['255.255.255.0',],
     gateway   => ['10.105.94.1',],
   }
}
EOF
```

Step 6 Validate the syntax of your newly created puppet script(s):

```
puppet parser validate

/etc/puppet/env_config/modules/custom/manifests/static_routes.pp
```

Step 7 Rebuild your Environment Configuration:

```
/var/qps/install/current/scripts/build/build_env_config.sh
```

Step 8 Reinitialize your environment:

```
/var/qps/install/current/scripts/upgrade/reinit.sh
```

At this point your new manifest is applied across the deployment. For more details, refer to the installer image in the `/etc/puppet/env_config/README`.

What to do next

It is recommended that version control is used to track changes to these Puppet customizations.

For example, to use 'git', perform the following steps:

1. Initialize the directory as a repository:

   ```
   # git init
   ```

   Initialized empty Git repository in `/var/qps/env_config/.git/`.

2. Add everything:

   ```
   # git add .
   ```

3. Commit your initial check-in:

   ```
   # git commit -m 'initial commit of env_config'
   ```

4. If you are making more changes and customizations, make sure you create new revisions for those:

   ```
   # git add .

   # git commit -m 'updated static routes'
   ```

HTTPS Support for Orchestration API

Installation

By default, the Orchestration API service starts with the HTTP mode on Cluster Manager.

You can change the mode to start with HTTPS self-signed certificate by setting the `api_https=one_way_ssl` facter value in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file in Cluster Manager. This ensures that the API server starts by using the pre-loaded self-signed SSL certificates.

Important You cannot upload certificates using the API.

To configure the Orchestration API server to start with the HTTPS self-signed certificate mode, make the following changes to the Heat template. These changes create the `/etc/facter/facts.d/cluman_facts.yaml` file and also set the puppet facter value to `api_https=one_way_ssl` in the configuration file in Cluster Manager.

```
cluman_api_name:
 type: string
 label: cluman orch api
 description: cluman orch
 default: one_way_ssl
# This will set the default value to one_way_ssl
```

```
- path: /etc/facter/facts.d/cluman_facts.yaml
 permissions: "0755"
 content:
 str_replace:
 template: |
   api_https: $kval
 params:
   $kval: { get_param: cluman_api_name }
```

Sample YAML configuration to run the Orchestration API server:

- Using self-signed certificates (one_way_ssl):

```
cat /etc/facter/facts.d/cluman_facts.yaml
 api_https: one_way_ssl
```

- Using trusted certificates (one_way_ssl):

```
cat /etc/facter/facts.d/cluman_facts.yaml
 api_https: one_way_ssl
 api_keystore_path: /var/certs/keystore.jks
 api_keystore_password: yoursecret
 api_keystore_type: JKS
 api_cert_alias: server-tls
 api_tls_version: TLSv1.2
 api_validate_certs: FALSE
 api_validate_peers: FALSE
```

- Using mutual authentication (two_way_ssl):

```
cat /etc/facter/facts.d/cluman_facts.yaml
 api_https: two_way_ssl
 api_keystore_path: /var/certs/keystore.jks
 api_keystore_password: yoursecret
 api_keystore_type: JKS
 api_cert_alias: server-tls
 api_tls_version: TLSv1.2
 api_truststore_path: /var/certs/truststore.jks
 api_truststore_password: yoursecret
 api_truststore_type: JKS
 api_validate_certs: TRUE
 api_validate_peers: TRUE
 api_enable_crldp: TRUE
```

Note

- For more information on how to add certificates to the keystore or truststore, see Adding Certificates to Keystore and Truststore, on page 52.
- Trusted certificates, keystores, or the truststore should not be located at `/opt/orchestration_api_server/`.
- For a list of the configuration parameters for HTTPS, see Configuration Parameters for HTTPS, on page 53.

After Cluster Manager is deployed, you can reconfigure the API server to run on HTTP (default) or HTTPS mode. The prerequisites to configure the HTTPS mode are as follows:

- For self-signed certificates, set `api_https=one_way_ssl` in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file.
- For trusted certificates:

1. Install the certificates on Cluster Manager.
2. Import the certificates into the keystore and the truststore.
3. Set `api_https` value to `one_way_ssl` or `two_way_ssl` (mutual authentication) in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file.

To apply the configuration run the following **puppet** commands on Cluster Manager. These commands reconfigure Cluster Manager only.

1. `cd /opt/cluman`
2. `CLUMAN_DIR="/opt/cluman";`
3. `puppet apply --logdest /var/log/cluman/puppet-run.log --modulepath=${CLUMAN_DIR}/puppet/modules --config ${CLUMAN_DIR}/puppet/puppet.conf ${CLUMAN_DIR}/puppet/nodes/node_repo.pp`

Note

1. For fresh installation, only HTTP or HTTPS with self-signed certificates mode is allowed.
2. For `one_way_ssl`, the `api_validate_peers` parameter should be set to *FALSE*.
3. In case some parameters are missing in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file:
   - For one way ssl, the Orchestration API server starts by using the self-signed certificates.
   - For two way ssl, the Orchestration API server rolls back to the default HTTP mode.

Upgrade

Upgrade CPS to run the Orchestration API server on HTTP or HTTPS. To change the behavior, configuration parameters must be configured before triggering the upgrade.

Follow the steps below to upgrade CPS:

- For self-signed certificates, set `api_https=one_way_ssl` in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file and then trigger the upgrade.
- For trusted certificates:
  1. Install the certificates on Cluster Manager.
  2. Import the certificates into the keystore and the truststore.
  3. Set `api_https` value to `one_way_ssl` or `two_way_ssl` (mutual authentication) in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file.
  4. Trigger the upgrade.

Note To roll back the configuration to default, that is HTTP mode, do the following:

1. Move the `/etc/facter/facts.d/cluman_facts.yaml` configuration file to the `/root/` folder.
2. Run the following **puppet** commands on Cluster Manager:
   1. `cd /opt/cluman`
   2. `CLUMAN_DIR="/opt/cluman";`
   3. `puppet apply --logdest /var/log/cluman/puppet-run.log --modulepath=${CLUMAN_DIR}/puppet/modules --config ${CLUMAN_DIR}/puppet/puppet.conf ${CLUMAN_DIR}/puppet/nodes/node_repo.pp`

Adding Certificates to Keystore and Truststore

A keystore contains private keys and certificates used by the TLS and SSL servers to authenticate themselves to TLS and SSL clients respectively. Such files are referred to as keystores. When used as a truststore, the file contains certificates of trusted TLS and SSL servers or of certificate authorities. There are no private keys in the truststore.

Note Your trusted certificates and keystores or truststores should not be located at `/opt/orchestration_api_server/`

Step 1 Create the PKCS12 file for key and certificate chains.

```
openssl pkcs12 -export-name <cert name> -n chain.crt -inkey <cert_private_key> - out server.p12
```

For example: `openssl pkcs12 -export -name server-tls -in chain.crt -inkey server.key -out server.p12`

Step 2 Create the Java KeyStore on the server.

```
keytool -importkeystore -destkeystore <keystore_name.jks> -srckeystore server.p12 -srcstoretype pkcs12 -alias server-tls
```

```
keytool -importkeystore -destkeystore keystore.jks -srckeystore server.p12 -srcstoretype pkcs12 -alias server-tls
```

Step 3 Import the root certificate or CA certificate in the truststore.

```
# Import your root certificate into a new trust store and follow the prompts
```

```
keytool -import -alias root -file root.crt -keystore truststore.jks
```

You must remember the keystore password and this needs to be updated in the `/etc/facter/facts.d/cluman_facts.yaml` file.

Configuration Parameters for HTTPS

The following parameters can be defined in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file. This file is loaded only onto the Cluster Manager VM. All parameters and values are case sensitive.

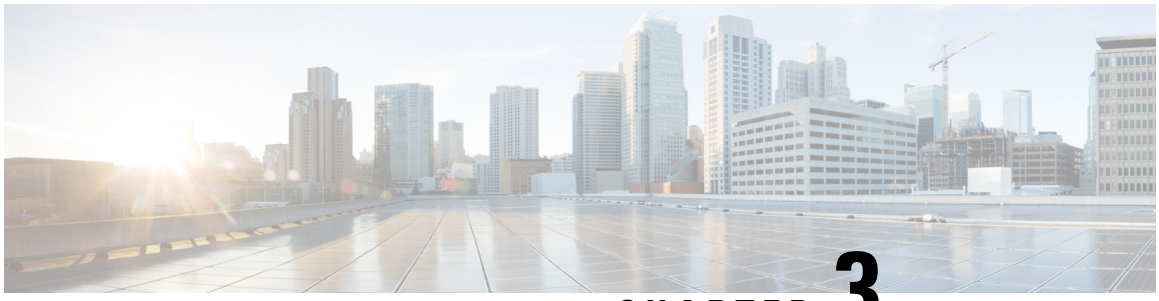
Note Before loading the configuration file to the Cluster Manager VM, verify that the YAML file uses the proper syntax. There are many publicly-available Websites that you can use to validate your YAML configuration file.

Table 4: HTTPS Configuration Parameters

Parameter	Description
`api_https`	Runs the application with or without HTTPS (one way or mutual authentication). Valid options: • disabled (default) • one_way_ssl • two_way_ssl
`api_tls_version`	List of protocols that are supported. Valid options: • TLSv1.1 • TLSv1.2
`api_keystore_path`	Path to the Java keystore which contains the host certificate and private key. Required for one_way_ssl and two_way_ssl.
`api_keystore_type`	Type of keystore. Valid options: • Java KeyStore (JKS) • PKCS12 • JCEKS`` • Windows-MY} • Windows-ROOT Required for one_way_ssl and two_way_ssl.
`api_keystore_password`	Password used to access the keystore. Required for one_way_ssl and two_way_ssl.

Parameter	Description
`api_cert_alias`	Alias of the certificate to use. Required for one_way_ssl and two_way_ssl.
`api_truststore_path`	Path to the Java keystore which contains the CA certificates used to establish trust. Required for two_way_ssl.
`api_truststore_type`	The type of keystore. Valid options: • Java KeyStore (JKS) • PKCS12 • JCEKS`` • Windows-MY} • Windows-ROOT Required for two_way_ssl.
`api_truststore_password`	Password used to access the truststore. Required for two_way_ssl.
`api_validate_certs`	Decides whether or not to validate TLS certificates before starting. If enabled, wizard refuses to start with expired or otherwise invalid certificates. Valid options: • true • false Required for one_way_ssl and two_way_ssl.
`api_validate_peers`	Decides whether or not to validate TLS peer certificates. Valid options: • true • false Required for one_way_ssl and two_way_ssl.
`api_need_client_auth`	Decides whether or not client authentication is required. Valid options: • true • false Required for one_way_ssl and two_way_ssl.

Parameter	Description
`api_enable_crldp`	Decides whether or not CRL Distribution Points (CRLDP) support is enabled. Valid options: • true • false Required for two_way_ssl.



CHAPTER 3

Orchestration API

Important After the configuration is complete, take a backup of the configuration to be used in case there is an issue with configuration at a later stage. For more information on taking the backup, refer to *CPS Backup and Restore Guide*.

Installation APIs

Input and Output Formats

The CPS Orchestration API supports both YAML and JSON formats for both inputs (request payload) and outputs (response payloads).

The input format is specified by the "Content-Type" attribute in the header. The input format is mandatory if the request includes a message body; it must be specified in the header for any API such request.

The output format is specified by the "Accept" attribute in the header. The output format is optional.

The following formats are supported for Content-Type and Accept attributes:

- application/json
- application/yaml
- text/yaml

The default output format (if the Accept attribute is not specified) for all APIs is always application/json except for following APIs, for which the default output format is text/yaml:

- /api/system/config
- /api/system/config/additional-hosts
- /api/system/config/hosts
- /api/system/config/replica-sets

- /api/system/mongo/config

/api/system/status/cluman

Purpose

This API returns the readiness status of the Cluster Manager VM.

Cluster Manager VM Readiness

If `/mnt/iso/install.sh` is executing, the status is returned as 'not ready'.

If `/mnt/iso/install.sh` has completed executing, status is returned as 'ready'.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/status/cluman

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** JSON
- **Response:** 200 OK: success

  The following example shows the status reported for a new CPS deployment:

```
{
    "status": "ready",
}
```

API logs are at written to: `/var/log/orchestration-api-server.log`

/api/system/config/

Purpose

This API is used to load an initial configuration or return (GET) the current CPS cluster configuration.

This API is also used to apply the loaded configuration to all VMs within the CPS cluster.

API logs are at written to: `/var/log/orchestration-api-server.log`

Retrieve the Current Configuration

To retrieve (GET) the current CPS cluster configuration that is loaded on the CPS Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200: OK.

  Example Response (No Configuration Present) XML:

```
---
configVersion: null
hosts: null
vlans: null
additionalHosts: null
config: null
licenses: null
replicaSets: null
```

  For a response showing an example configuration file refer to Sample YAML Configuration File - HA Setup, on page 80.

Load a Configuration

Note This API can only be used once for initial deployment. Once a configuration has been applied (/system/config/action/apply) as described below, this API is no longer available.

Note Before loading the configuration file to your CPS cluster, verify that the YAML file uses the proper syntax. There are many publicly-available websites which you can use to validate your YAML configuration file.

Note When this API is issued, the following basic validations are performed on the consolidated configuration (YAML) file submitted in the payload:

- The replica set hosts are included in hosts or additionalHosts section
- Standard CPS aliases are present (lb01, lb02, and so on)
- Standard CPS vlan names are present (Internal, Management, and so on)
- Range checking (for example, IPv4/IPv6 IP address syntax validation)
- Cross-referencing of vlans with hosts

If a validation error is detected, an appropriate message is provided in the API response, and reported in `/var/log/orchestration-api-server.log`.

To load a new CPS cluster configuration on the CPS Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** POST
- **Payload:** Include the YAML configuration file in the POST. Refer to Sample YAML Configuration File - HA Setup, on page 80 for more information about this configuration file.
- **Response:** 200: success; 400: malformed or invalid; 403: Configuration may not be changed at this time (for example, after it has been applied).

To verify the configuration was properly loaded, perform another GET to http://<*Cluster Manager IP*>:8458/api/system/config/

Apply the Loaded Configuration

Note This API can only be used once for initial deployment. After a configuration has been applied, the API is no longer available.

Once a new configuration file has been uploaded to the Cluster Manager VM, you must apply the configuration. This triggers the Cluster Manager VM prepare and push out the new configurations to all VMs in the cluster, as well as perform any post-update steps.

During an initial deployment of a CPS cluster, the CPS VMs in the cluster will remain in an inactive/waiting state until this configuration file is applied.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/action/apply

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/json
- **Method:** POST
- **Payload:** There is no payload.
- **Response:** 200: success; 400: malformed or invalid; 403: Configuration may not be applied at this time; 500: System error. See logs.

To check the status of the CPS cluster after applying a configuration, refer to /api/system/config/status , on page 88.

Encrypt Administration Traffic Parameters

The administration traffic parameters (rysyslog, haproxy, SNMPv3) can be configured under the "config:" section which defines general global parameters used to deploy CPS.

Important For fresh installation, in case the parameters *rsyslog_tls* and *rsyslog_ca* are not set, they would be initialized to default values and feature would be enabled. If the user wants to disable the feature *rsyslog_tls* should be set to **`FALSE`**.

Similarly, for *haproxy_stats_tls*, if no value is set (TRUE or FALSE), the default value (TRUE) is used and the feature is enabled.

For SNMPv3, until the snmpv3 tag is not commented out, the feature would not be enabled.

Note For upgrade scenario, if parameters are not defined they are initialized to empty.

Table 5: Traffic Parameters

Parameter	Description
rsyslog_tls	This field is used to enable or disable encryption for rsyslog. Default: TRUE
rsyslog_cert	This field is used to define the path for trusted Certificate of server.

Parameter	Description
rsyslog_ca	This field is used to define the Path of certifying authority (CA). Default: `/etc/ssl/cert/quantum.pem`
rsyslog_key	This field is used to define the path of private key.
haproxy_stats_tls	This field is used to enable or disable the encryption for HAproxy statistics. Default: TRUE

Sample YAML format (for enabling SNMPv3):

```
config:
 # enable SNMP V3.
# If null, SNMP V3 is disabled.
# To enable add the following:
# v3User: The SNMP V3 user: REQUIRED
# engineId: hex value (ie, 0x0102030405060708): REQUIRED
# authProto: SHA or MD5: REQUIRED
# authPass: at least 8 characters: REQUIRED
# privProto: AES or DES: REQUIRED
# privPass: OPTIONAL
snmpv3:
v3User: "cisco_snmpv3" #---->Default value. You can change as per your deployment requirements
engineId: "0x0102030405060708"
authProto: "SHA"
authPass: "cisco_12345"
privProto: "AES"
privPass: ""rsyslogTls: "TRUE"
```

Sample YAML format (for rsyslog, haproxy):

```
config:
rsyslogCa: "/etc/ssl/certs/quantum.pem"
rsyslogCert: "/etc/ssl/cert/quantum.pem"
rsyslogKey: "/etc/ssl/cert/quantum.key"
haproxyStatsTls: "TRUE"
```

Configuration Parameters - HA System

The following parameters can be defined in the CPS configuration file. Refer also to: Sample YAML Configuration File - HA Setup, on page 80.

In this file, the Internal, Management and Gx networks must have an exact case match of "Internal", "Management" and " Gx" in the following sections:

- hosts: interfaces: value of "network"
- vlans: value of "name"

All parameters and values are case sensitive.

Note Before loading the configuration file to your CPS cluster, verify that the YAML file uses the proper syntax. There are many publicly-available websites which you can use to validate your YAML configuration file.

Table 6: Configuration Parameters - HA System

Parameter	Description
`configVersion`	The version of the configuration file. This must be set to `configVersion: 1.0`.
`hosts:`	This section defines the host entries for each of the CPS VMs in the deployment.
`- name:`	Defines the host name of the VM. This name must be resolvable in the enterprise DNS environment. **Note** CPS host names must conform to RFC 952 and RFC 1123; characters such as "_" are not allowed.
`alias:`	Defines the internal host name used by each CPS VMs for internal communication, such as lb0x, pcrfclient0x, sessionmgr0x, or qns0x.
`interfaces`	This section defines the network details for each VM.
`network:`	The network name which must match a VLAN name (see below).
`ipAddress:`	The IP interface address.
`vlans:`	This section defines the separate VLANs to be configured. The "Internal" and "Management" VLANs are always needed. For additional networks, add more as needed.
`- name:`	Defines the name for a particular VLAN. It is recommended to use a name representing the network for certain traffic. The VLAN names defined here must be used in the `network` field in the `hosts` section above. The "Internal" VLAN Name is always needed. Names must consist only of alphanumeric characters and underscores, and must not start with a number.
`vipAlias:`	The hostname associated with virtual interfaces on the Policy Directors (LBs), typically "Internal", "Management", and "Gx".
`vip:`	The Virtual IP address used on this VLAN. The virtual addresses are used to distribute the traffic between two Policy Directors. If using IPv6, the address must be specified in canonical form as described in RFC5929.
`guestNIC:`	The Name of the interface specified in the host cloud config or Heat definition.
`pcrfVipAlias:`	The OAM (pcrfclient) vip alias.

Parameter	Description
`additionalHosts`	This section defines any hosts not configured in the hosts section above. **Note** Policy Director (LB) VIPs are defined in this section as '`lbvip01`' and '`lbvip02`', as well as the '`arbitervip`' which defines the prcfclient01 internal IP. In a CPS cluster which is configured with more than 2 Policy Directors (LBs), HAproxy and the VIPs are hosted only on LB01 and LB02. The additional LBs serve only as diameter endpoints to route diameter traffic. Any other hosts which CPS must interact with, such as NTP or NMS servers, must be defined in this section. Any hosts defined here are added to each CPS VM `/etc/hosts` file. **Note** The host names defined here do **not** need to conform to RFC 952 and RFC 1123.
`- name:`	The hostname of the host.
`alias:`	The internal host name used by CPS nodes for internal communication, such as qns01.
`ipAddress:`	The IP address to use in the `/etc/hosts` file.
`config:`	This section defines general global parameters used to deploy CPS.
`qpsUser:`	Do not change.
`selinuxState:`	Do not change. Security Enhanced Linux (SELinux) support: disabled | enforcing. Default: disabled
`selinuxType:`	Do not change.
`broadhopVar:`	Do not change. Default: broadhop
`tacacsEnabled:`	Enter TRUE to enable TACACS+ authentication. For more information, refer to the *CPS Installation Guide for VMware*. Default: FALSE
`tacacsServer:`	Defines the IP address of the TACACS+ server.
`tacacsSecret:`	Defines the password/secret of the TACACS+ server.
`tacacsService`	A string value indicating which service to be used when authorizing and auditing against the TACACS+ servers. Default: `pcrflinuxlogin` if no value is specified.
`tacacsProtocol`	A string value indicating which protocol to be used when authorizing and auditing against the TACACS+ servers. Default: `ssh`

Parameter	Description
tacacsTimeout	An integer that represents how long the software needs to wait, in seconds, for the TACACS+ server to respond to the queries. Default: 5
tacacsDebug	An integer value indicating the debug level to run the software in. Currently, this is effectively boolean. Default: 0
redisAuthenticationEnabled	This field is used to enable or disable Redis authentication. Default: true (For fresh installations) To enable or disable Redis authentication for upgrade and migration setups, refer to Redis Authentication for Upgrading/Migrating Systems, on page 74.
redisAuthenticationPasswd	This field is used to add an encrypted password for Redis. For more information about generating encrypted password, refer to *Password Encryption* section under *Redis Authentication* in *CPS Installation Guide for VMware*.
redisServerCount	This value specifies the number of Redis server instances running on each policy director (lb) VM. Redis authentication is enabled with the number of instances as defined in *redisServerCount*. If the value for Redis server count is not provided, default value of 3 is used. To disable Redis explicitly, Redis server count should have value 0. Default: 3 Value range: 0 to 64

Parameter	Description
redisForLdapRequired	This parameter is used only when dedicated LDAP instance is required. Default: false Possible values: true, false If you configure LDAP instance explicitly, first Redis instance on policy director (lb) VMs running on port 6379 is used for LDAP and the remaining is used for diameter. **Note** If you configure redisForLdapRequired parameter, then the following changes are automatically added in configuration files. In /etc/broadhop/qns.conf file, an additional parameter -DldapRedisQPrefix=ldap is added. /etc/broadhop/redisTopology.ini file has the following content if redisForLdapRequired=true and redisServerCount=3: ldap.redis.qserver.1=lb01:6379 policy.redis.qserver.2=lb01:6380 policy.redis.qserver.3=lb01:6381 ldap.redis.qserver.4=lb02:6379 policy.redis.qserver.5=lb02:6380 policy.redis.qserver.6=lb02:6381 If a dedicated LDAP instance is required, you many also want to consider increasing the total Redis servers to accommodate the diameter traffic. For example, if redisForLdapRequired property was not configured, and redisServerCount=3 then after configuring redisForLdapRequired as true, you want to increase total redis server count to 4 by setting redisServerCount=4.
databaseNics	This parameter allows user to provide interface names on which the firewall is opened for replica-set on a VM. If databaseNics is not configured, firewall is opened only for internal interface for a replica-set. If databaseNics is configured, then firewall is opened for configured interfaces and internal interface as well (even if it is not mentioned in databaseNics). This field has comma (,) or semicolon (;) separated interface names for firewall ports to be opened for a replica-set on a VM. **Note** This field is effective only when the firewall is enabled.
freeMemPer:	By default, a low memory alert is generated when the available memory of any CPS VM drops below 10% of the total memory. To change the default threshold, enter a new value (0.0-1.0) for the alert threshold. The system generates an alert trap whenever the available memory falls below this percentage of total memory for any given VM. Default: 0.10 (10% free).

Parameter	Description
`syslogManagers:`	Entries are space separated tuples consisting of protocol:hostname:port. Only UDP is supported at this time. Default: 514. For example: udp:corporate_syslog_ip:514 udp:corporate_syslog_ip2:514
`syslogManagersPorts:`	A comma separated list of port values. This must match values in the syslog_managers_list.
`logbackSyslogDaemonPort:`	Port value for the rsyslog proxy server to listen for incoming connections, used in the rsyslog configuration on the Policy Director (lb) and in the logback.xml on the OAM (pcrfclient). Default: 6515
`logbackSyslogDaemonAddr:`	IP address value used in the `/etc/broadhop/controlcenter/logback.xml` on the OAM (pcrfclient). Default: lbvip02
`cpuUsageAlertThreshold:`	The following cpu_usage settings are related to the High CPU Usage Alert and High CPU Usage Clear traps that can be generated for CPS VMs. Refer to the *CPS SNMP, Alarms and Clearing Procedures Guide* for more details about these SNMP traps. Set the higher threshold value for CPU usage. The system generates an Alert trap whenever the CPU usage is higher than this value.
`cpuUsageClearThreshold:`	The lower threshold value for CPU usage. The system generates a Clear trap whenever the CPU usage is than this value and Alert trap is already generated.
`cpuUsageTrapIntervalCycle:`	The interval period to execute the CPU usage trap script. The interval value is calculated by multiplying five with the given value. For example, if set to one, then the script is executed every five seconds. The default value is 12, which means the script is executed every 60 seconds.
`snmpTrapCommunity:`	The SNMP trap community string. Default: broadhop
`snmpRoCommunity:`	This value is the SNMP read-only community string. Default: broadhop
`monQnsLb:`	Do not change.
`freeMemoryPerAlert:`	By default, a low memory alert is generated when the available memory of any CPS VM drops below 10% of the total memory. To change the default threshold, enter a new value (0.0-1.0) for the alert threshold. The system generates an alert trap whenever the available memory falls below this percentage of total memory for any given VM. Default: 0.10 (10% free)

Parameter	Description
`freeMemoryPerClear:`	Enter a value (0.0-1.0) for the clear threshold. The system generates a low memory clear trap whenever available memory for any given VM is more than 30% of total memory. Default: 0.3 (30% of the total memory)
`monitorReplicaTimeout:`	This value is used to configure the replica-set timeout value. The default value is 540 seconds considering four replica sets. The customer can set timeout value according to the number of replica sets in their network. To recover a single session replica-set, it takes approximately 120 sec and adding 20% buffer to it; we are using 540 sec for default (for four replica sets). Without any latency between sessionmgr VMs, one replica-set recovers in ~135 seconds. If latency (40 -100 ms) is present between sessionmgr VMs, add a 10% buffer to 135 seconds and set the timeout value for the required number of replica sets in the deployment.
`sctpEnabled:`	Enables (`TRUE`) or disables (`FALSE`) Stream Control Transmission Protocol (SCTP) support for Diameter interfaces. Default: `TRUE`
`firewallState:`	Enables or disables linux firewall (IPtables) on all VMs. Valid Options: enabled / disabled Default: enabled

Parameter	Description
`snmpv3:`	Enable SNMPv3 support within CPS by deleting `null` and uncommenting (removing #) the following snmpv3 object parameters: • `v3User:` Username to be used for SNMPv3 request/response and trap. This parameter is required. Default: cisco_snmpv3 • `engineId:` This value is used for SNMPv3 request/response and on which NMS manager can receive the trap. It must be a hex value. This parameter is required. Default: 0x0102030405060708 • `authProto:` SHA or MD5. This value specifies the authentication protocol to be used for SNMPv3. This parameter is required. Default: SHA • `authPass:` This value specifies the authentication password to be used for SNMPv3 requests. It should have minimum length as 8 characters. This parameter is required. Default: cisco_12345 • `privProto:` This value specifies Privacy/Encryption protocol to be used in SNMPv3 request/response and SNMP trap. User can use AES/DES protocol. This parameter is required. Default: AES • `privPass:` This value specifies Privacy/Encryption password to be used in SNMPv3. If it is blank then value specified in `authPass` is used as privPass. This parameter is optional. Default: *blank (no value)*
`snmpRouteLan:`	This field contains the value of a VLAN name which can be used to access the KPIs value provided by SNMP. Default: Management
`remoteClumanIp:`	This parameter is used for GR deployments to synchronize mongo configuration across sites. For more information, refer to /api/system/config/replica-sets/action/sync-mongo, on page 112.

Parameter	Description
`dbAuthenticationEnabled:`	This field is used to enable or disable MongoDB authentication. Possible value: true or false **Note** You must configure `dbAuthenticationEnabled` parameter. This parameter cannot be left empty. To disable the authentication, the parameter value must be set as false. To enable, the value should be true, and admin and readonly passwords must be set. This is applicable only for new installs and not for upgrades. For MongoDB authentication process, refer to MongoDB Authentication Process, on page 88.
`dbAuthenticationAdminPasswd:`	This parameter is the plain or encrypted password for admin user depending on the value set in `dbAuthenticationEncryption` parameter.
`dbAuthenticationReadonlyPasswd:`	This parameter is the plain or encrypted password for readonly user depending on the value set in `dbAuthenticationEncryption` parameter.
`dbAuthenticationEncryption:`	If this parameter is false, then the `dbAuthenticationAdminPasswd` and `dbAuthenticationReadonlyPasswd` are in plain text. **Note** Make sure to remove the `dbAuthenticationAdminPasswd` and `dbAuthenticationReadonlyPasswd` fields from your input YAML file after configuring API. If this parameter is true, then the encrypted password needs to be configured. For encrypted passwords, you need to SSH to a Cluster Manager and execute the following command: `/var/qps/bin/support/mongo/encrypt_passwd.sh <Password>` Default: false
`enableSshLoginSecurity:`	This parameter allows user to enable or disable SSH login security. Default: disabled Possible values: enabled, disabled
`cpsAdminUserCluman:`	This parameter is used to configure Cluster Manager administrator user.
`cpsAdminPasswordCluman:`	This parameter is the encrypted password for administrator user.
`whitelistedHostsForSsh:`	Valid values are an array of whitelisted hosts specified in string for which SSH access needs to be allowed. This configuration is effective only when the SSH login security is enabled. If the hostname is mentioned then it should be resolvable by CPS VM's. No validation on hostname/IP addresses is provided. You can specify both IPv4/IPv6 address. **Note** New whitelisted host list overwrites the old list. If the new whitelist host configuration is empty then all old additional whitelisted hosts (apart from standard local CPS VM's host) are deleted.
`sysUsers:`	This section defines CPS system users.

Parameter	Description
`- name:`	The username of this user.
`password:`	The password must be encrypted for this user. Refer to the section *Password Encryption* in *CPS Installation Guide for VMware* for instructions to generate an encrypted password.
`groups:`	This section defines the groups to which this user belongs. **Note** User group can be qns-svn, qns-ro, qns-su, qns-admin and pwauth. pwauth group is valid only for `qns` username and no other username.
`- <group>`	List each group on a separate line.
`hvUsers`	This section defines the hypervisor users.
`- name:`	The username of a user with root access to the host/blade. If installing CPS to multiple blade servers, it is assumed that the same username and password can be used for all blades.
`password:`	The password for this user. To pass special characters, they need to be replaced with the "% Hex ASCII" equivalent. For example, "$" would be "%24" or "hello$world" would be "hello%24world".
`additionalUsers:`	This section defines additional CPS system users, such as those given access to Control Center.
`- name:`	The username of this user.
`password:`	The clear text password for this user.
`groups:`	This section defines the groups to which this user belongs.
`- <group>`	List each group on a separate line.
`licenses:`	This section is used to enter the CPS license information. Contact your Cisco representative to receive your CPS license key(s).
`- feature:`	The name of the feature license, for example: "MOBILE_CORE".
`license:`	The license key for this feature.
`replicaSets:`	This section defines the CPS MongoDB replica sets.
`- title:`	The database for which the replica set is being created.
`setName:`	The name of the replica set.
`oplogSize:`	MongoDB operations log (oplog) size, in MB. Default: 5120
`arbiters:`	The hostnames and ports of the arbiter.
`arbiterDataPath:`	The data directory on the arbiter VM.

Parameter	Description
`members:`	The list of members for the replica set. Each list element is a session manager hostname:port. For example, sessionmgr01:27718.
`- <member>`	List each member hostname:port on a separate line.
`dataPath:`	The data directory path on the Session Manager VM.
`LDAP SSSD`	For more information, refer to LDAP SSSD, on page 72.
`enablePrometheus:`	This parameter is used to enable/disable Prometheus in CPS. Default: disabled Possible Values: enabled, disabled For more information, refer to *Prometheus and Grafana* chapter in *CPS Operations Guide*.
`statsGranularity:`	This parameter is used to configure statistics granularity in seconds. Default: 10 seconds Possible Values: Positive Number For more information, refer to *Prometheus and Grafana* chapter in *CPS Operations Guide*.
DSCP Configuration	For more information, refer to DSCP Configuration, on page 74.
Critical Files Configuration	For more information, refer to Critical File Monitoring Configuration, on page 77.

LDAP SSSD

Note For LDAP SSSD routable IP is required. LDAP server must be accessible from CPS VMs (LDAP client).

Table 7: LDAP SSSD

Parameter	Description
`ldapOnAll:`	When set to true, it installs the LDAP SSSD on all CPS VMs. When set to false, it install the LDAP SSSD only on pcrfclient/policy directors (lb) VMs. **Note** true or false must be in small case.
`ldapEnabled:`	When set to true, applies the SSSD configuration as per input provided by user. When set to false, use the default configuration. **Note** true or false must be in small case.

Parameter	Description
`ldapServer:`	Contains server IP:port to configure LDAP. Format: ldaps://<*serverip*>:<*port*>
`ldapSearchBase:`	This is required for SSSD configuration. The default base DN to use for performing LDAP user operations. Format: ou=users,dc=cisco,dc=com
`ldapDefaultBindDn:`	The default bind DN to use for performing LDAP operations. Format: uid=admin,ou=system
`ldapSecret:`	The authentication token for the default bind DN. Currently, only clear text passwords are supported. For example, secret
`ldapDefaultUser:`	The default LDAP user to be configured in LDAP server. For example, admin
`ldapOuUser:`	The default LDAP user OU. For example, users
`ldapOuGroup:`	The default LDAP group user OU. For example, groups
`ldapDefaultGroup:`	The LDAP attribute that corresponds to the group name. For example, Admin
`ldapDefaultGroupEditor:`	This is a user group which has the editor access to Grafana. For example, User
`ldapDcName:`	This is a single entity of all domains. Format: dc=cisco,dc=com

After migration from CPS 13.x.x or CPS 14.x.x to CPS 18.2.0 release, LDAP SSSD configuration is installed on default VM (pcrfclient/lb) and not on all VMs. You need to configure LDAP SSSD on all the other VMs.

Once LDAP SSSD configuration is complete, you need to authenticate the LDAP certificate. For more information, refer to *LDAP SSSD Configuration* section in *CPS Installation Guide for VMware*.

If you are migrating from a lower version such as CPS 13.x.x to CPS 18.x.x and you do not want the LDAP SSSD, modify the LDAP parameters as follows in YAML file:

```
ldapOnAll=false
ldapEnabled=false
```

After the modification, run `import_deploy.sh` so that LDAP SSSD is not installed by default

For more information about LDAP SSSD certificate authentication and troubleshooting, refer to *LDAP SSSD Configuration* section in *CPS Installation Guide for VMware*.

Redis Authentication for Upgrading/Migrating Systems

Caution Enabling or disabling Redis authentication for upgraded or migrated systems require application downtime.

Change Redis User Password

1. Modify password using config PATCH API.
2. Wait for the patch task to be completed.
3. Run `redis_auth_upgrade.sh` script to change the password and provide the old plain text password.

   ```
   /var/qps/bin/support/redis/redis_auth_upgrade.sh -c <old_plaintext_password>
   ```
4. Restart all the java processes.

Disable Redis Authentication

1. Modify redis authentication using config PATCH API.
2. Wait for the patch task to be completed.
3. Run `redis_auth_upgrade.sh` script to disable authentication and provide the plain text password.

   ```
   /var/qps/bin/support/redis/redis_auth_upgrade.sh -d <plaintext_password>
   ```
4. Restart all the java processes.

Enable Redis Authentication

1. Modify redis authentication using config PATCH API.
2. Wait for the patch task to be completed.
3. Run `redis_auth_upgrade.sh` script to enable the authentication and provide the old plain text password.

   ```
   /var/qps/bin/support/redis/redis_auth_upgrade.sh -e <plaintext_password>
   ```
4. Restart all the java processes.

DSCP Configuration

You can configure DSCP bits using DSCP class or DSCP value on the following for IPv4 and/or IPv6:

Table 8: DSCP Configuration

Parameter	Description
vmRole	This parameter is used to specify the VM type. Valid values are: lb, pcrfclient, qns, sessionmgr, udc.

Parameter	Description
ipFamily	This parameter is used to specify ipv4 or ipv6 address. If no parameter is configured, then the value ipv4 and ipv6 are used.
outInterface	This parameter is used to specify the interface name i.e., eth0/eth1. If no parameter is configured, then DSCP marking is applied to any interface.
protocol	This parameter is used to specify tcp/udp and so on. If no parameter is configured, then DSCP marking is applied to any protocol.
destIp	This parameter is used to specify destination IP.
destPort	This parameter is used to specify destination port.
sourcePort	This parameter is used to specify the source port.
dscpClass	This parameter is used to specify DSCP class. Supported values are: af11, af12, af13, af21, af22, af23, af31, af32, af33, af41, af42, af43, cs1, cs2, cs3, cs4, cs5, cs6, cs7, ef
dscpValue	This parameter is used to specify DSCP value.

Retrieve the Current Configuration Change for DSCP

To retrieve (GET) the current CPS cluster configuration that is loaded on the CPS Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- Example Response (YAML format) XML:

```
dscpconfig:
- vmRole: "qns"
  ipFamily: ""
  outInterface: "eth0"
  protocol: "tcp"
  sourcePort: ""
  destIp: ""
  destPort: "80"
  dscpClass: ""
  dscpValue: "0x12"
```

For a response showing an example configuration file refer to Sample YAML Configuration File - HA Setup, on page 80.

Retrieve the Current DSCP Configuration

To retrieve (GET) the current DSCP configuration:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/dscp-config

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

  Example Response (YAML format) XML: HA Setup

```
# curl -s http://installer:8458/api/system/config/dscp-config
---
- vmRole: "qns"
  ipFamily: ""
  outInterface: "eth0"
  protocol: "tcp"
  sourcePort: ""
  destIp: ""
  destPort: "80"
  dscpClass: ""
  dscpValue: "0x12"
- vmRole: ""
  ipFamily: ""
  outInterface: "eth0"
  protocol: "udp"
  sourcePort: ""
  destIp: ""
  destPort: "5405"
  dscpClass: "af21"
  dscpValue: ""
```

  For a response showing an example configuration file refer to Sample YAML Configuration File - HA Setup, on page 80.

Load Updated DSCP Configuration

This API is used to load an updated DSCP configuration on the CPS Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/dscp-config

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml

- **Method:** PUT
- **Payload:** Include the YAML configuration file in the PUT request. The entire contents of the DSCP configuration must be included. Refer to Sample YAML Configuration File - HA Setup, on page 80 for more information about this configuration file.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

  Example Response: The updated contents of `dhcp.pp`, `reinit` are returned in the response in YAML format.

  Example Response (YAML format) XML: HA Setup

```
---
- vmRole: "sessionmgr"
ipFamily: ""
protocol: "tcp"
sourcePort: ""
destIp: ""
destPort: ""
outInterface: "eth3"
dscpClass: "af11"
dscpValue: ""
```

Note If you pass empty payload then all DSCP rules are removed (that is, disable DSCP configuration).

Critical File Monitoring Configuration

You can configure the critical file names to be monitored for write, execute or any other attribute changes.

Important Critical Files configuration is specific to Cluster Manager. If you are using Geographic Redundancy configuration, then you need to do the configuration across all the Cluster Managers.

Table 9: Critical Files Configuration

Parameter	Description
fileToBeMonitored	File name with absolute path of the file that needs to be monitored.
actionToBeMonitored	Action for file that needs to be monitored. Supported options are: • w –write • x - execute • a – attribute changes

Important File monitoring for read operation is not supported.

Rules configured in `CriticalFilesMonConfig` section of YAML files are added in `#BEGIN_CPS_AUDIT_RULES` and `#END_CPS_AUDIT_RULES` block in `/etc/audit/rules.d/audit.rules` file on Cluster Manager VM.

Sample output of AUDIT block in audit.rules:

```
#BEGIN_CPS_AUDIT_RULES
-w /etc/hosts -p wxa -k watch_critical_files
-w /etc/broadhop.profile -p wxa -k watch_critical_files
#END_CPS_AUDIT_RULES
```

Important Do not modify the rules in `#BEGIN_CPS_AUDIT_RULES` and `#END_CPS_AUDIT_RULES` block manually. Any modification done in this block is overwritten every time you execute `/var/qps/install/current/scripts/bin/support/update_audit_conf.py` script.

You can add the custom rules in `/etc/audit/rules.d/audit.rules` file outside of the `#BEGIN_CPS_AUDIT_RULES` and `#END_CPS_AUDIT_RULE` block but notification (SNMP trap) is not sent for the rules.

Note SNMP alarm with version v2c or v3 is generated based on SNMP configuration done in YAML file. There is no clear alarm.

Audit daemon logs all the audit events occurred in `/var/log/audit/audit.log` file with no delay.

`/var/qps/install/current/scripts/bin/support/snmp-traps/vm-traps/gen-crit-file-mod-traps.py` script monitors audit.log file for any file modification event since last execution of script and send traps for all the events occurred during this time.

`gen-crit-file-mod-traps.py` scripts last execution time is stored in `/var/tmp/lastGenCritFileModExeTime`. If the file does not contain any entry for last execution or the file is not present, then trap for events occurred during last 60 seconds is sent.

These traps are available in `/var/log/snmp/trap` file on active Policy Director (lb) VM.

You can execute the following command on Cluster Manager VM to validate particular audit logs:

```
ausearch -i -k watch_critical_files
```

Sample Output:

```
type=PROCTITLE msg=audit(08/26/2018 18:53:56.834:250) : proctitle=vim /etc/hosts
type=PATH msg=audit(08/26/2018 18:53:56.834:250) : item=1 name=/etc/hosts inode=5245468
dev=08:02 mode=file,644 ouid=root ogid=root rdev=00:00 objtype=CREATE
type=PATH msg=audit(08/26/2018 18:53:56.834:250) : item=0 name=/etc/ inode=5242881 dev=08:02
mode=dir,755 ouid=root ogid=root rdev=00:00 objtype=PARENT
type=CWD msg=audit(08/26/2018 18:53:56.834:250) : cwd=/root/modified_iso
type=SYSCALL msg=audit(08/26/2018 18:53:56.834:250) : arch=x86_64 syscall=open success=yes
exit=3 a0=0x1c74390 a1=O_WRONLY|O_CREAT|O_TRUNC a2=0644 a3=0x0 items=2 ppid=18335 pid=13946
auid=root uid=root gid=root euid=root suid=root fsuid=root egid=root sgid=root fsgid=root
tty=pts0 ses=9 comm=vim exe=/usr/bin/vim key=watch_critical_files
```

Retrieve the Current CPS Cluster Configuration

To retrieve (GET) the current CPS cluster configuration that is loaded on the CPS Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- Example Response (YAML format) XML: In response the following section with configured files if any or Null if not configured is displayed:

```
CriticalFilesMonConfig:
```

For a response showing an example configuration file refer to Sample YAML Configuration File - HA Setup, on page 80.

Retrieve Critical File Monitoring Configuration

To retrieve (GET) the current configuration:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/critFileMon-config

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

Example Response (YAML format) XML: HA Setup

```
# curl -s http://installer:8458/api/system/config/critFileMon-config
---HTTP/1.1 200 OK
Date: Fri, 24 Aug 2018 11:08:57 GMT
Content-Type: text/yaml
Content-Length: 171

---
- fileToBeMonitored: "/etc/hosts"
  actionToBeMonitored: "wxa"
- fileToBeMonitored: "/etc/shadow"
  actionToBeMonitored: "xa"
- fileToBeMonitored: "/etc/passwd"
  actionToBeMonitored: "xa"
```

For a response showing an example configuration file refer to Sample YAML Configuration File - HA Setup, on page 80.

Load Updated Critical File Monitoring Configuration

This API is used to load an updated critical file monitoring configuration on the CPS Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/critFileMon-config

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** PUT
- **Payload:** Include the YAML configuration file in the PUT request. The entire contents of the critical file monitoring config must be included. Refer to Sample YAML Configuration File - HA Setup, on page 80 for more information about this configuration file.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

  Sample command: `curl -i -X PUT http://installer:8458/api/system/config/critFileMon-config -H "Content-Type: application/yaml" --data-binary "@<input json file>"`

Sample YAML Configuration File - HA Setup

Use the following file as a template to create the YAML configuration file for your CPS deployment. Refer to Configuration Parameters - HA System, on page 62 for a description of the available parameters.

Important GuestNic must be populated as per network VLAN defined on ethenet interfaces in VMs.

Note RADIUS-based policy control is no longer supported in CPS 14.0.0 and later releases as 3GPP Gx Diameter interface has become the industry-standard policy control interface.

```
#
#  CPS system configuration
#
#  CPS configuration is a YAML file with all the configuration required
#  to bring up a new installation of CPS.
#
#  This example file lists all possible configuration fields.
#  Fields that are not marked as required can be left out of
#  the configuration. Fields that are not provided will use
#  the default value. If not default is indicated the default
#  is an empty string.

# The version of the configuration file. The installation documentation
# for the version of the CPS you are installing will indicate which
# configuration version you must use.
# REQUIRED
configVersion: 1.0
```

```
# Configuration section for CPS hosts
# REQUIRED
hosts:
  # The host section must specify all hosts that are members of the CPS
  # deployment. Host entries consist of the following REQUIRED fields
  #  name: the string to be used as a hostname for the VM
  #  alias: the string to be used in hostname lookup for the VM
  #  interfaces: Network details consisting of the following REQUIRED fields
  #    network: The network name which must match a VLAN name (see below)
  #    ipAddress: The interface address
  - name: "lb01"
    alias: "lb01"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.201"
      - network: "Management"
        ipAddress: "172.18.11.154"
      - network: "Gx"
        ipAddress: "192.168.2.201"
  - name: "lb02"
    alias: "lb02"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.202"
      - network: "Management"
        ipAddress: "172.18.11.155"
      - network: "Gx"
        ipAddress: "192.168.2.202"
  - name: "sessionmgr01"
    alias: "sessionmgr01"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.22"
      - network: "Management"
        ipAddress: "172.18.11.157"
  - name: "sessionmgr02"
    alias: "sessionmgr02"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.23"
      - network: "Management"
        ipAddress: "172.18.11.158"
  - name: "qns01"
    alias: "qns01"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.24"
  - name: "qns02"
    alias: "qns02"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.25"
  - name: "qns03"
    alias: "qns03"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.26"
  - name: "qns04"
    alias: "qns04"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.27"
  - name: "pcrfclient01"
```

```
    alias: "pcrfclient01"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.20"
      - network: "Management"
        ipAddress: "172.18.11.152"
  - name: "pcrfclient02"
    alias: "pcrfclient02"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.21"
      - network: "Management"
        ipAddress: "172.18.11.153"

# Configuration section for CPS VLANs
# REQUIRED
vlans:
  # VLAN entries consist of the following REQUIRED fields
  #  name: The VLAN name. This name must be used in the "network" field
  #        host interfaces (see above)
  #  vipAlias: Hostname associated with the vip
  #  vip: Virtual IP used no this network, if any.
  #  guestNic: The name of the interface specified in the host cloud config
  #            or the Heat definition.
  #
  - name: "Internal"
    vipAlias: "lbvip02"
    vip: "172.16.2.200"
    guestNic: "eth0"
  - name: "Management"
    vipAlias: "lbvip01"
    vip: "172.18.11.156"

  - name: "Gx"
    vipAlias: "gxvip"
    vip: "192.168.2.200"

# Configuration section for hosts not configured in the hosts section above.
# REQUIRED
additionalHosts:
  # additionalHosts entries consist of the following REQUIRED fields
  #  name: The hostname
  #  alias: The string to be used in the etc/host file.
  #  ipAddress: The IP address to use in the etc/host file.
  #
  # the "arbitervip" to the pcrfclient01 internal ip is mandatory.
  #
  - name: "lbvip01"
    ipAddress: "172.18.11.156"
    alias: "lbvip01"
  - name: "lbvip02"
    ipAddress: "172.16.2.200"
    alias: "lbvip02"
  - name: "diam-int1-vip"
    ipAddress: "192.168.2.200"
    alias: "gxvip"
  - name: "arbitervip"
    ipAddress: "172.16.2.20"
    alias: "arbitervip"

# Configuration section for general configuration items.
# REQUIRED
config:
  # Do not change. See install documentation for details.
```

```
# default: sys_user_0
qpsUser: "sys_user_0"

# Do not change. See install documentation for details.
# default: disabled
selinuxState: "disabled"

# Do not change. See install documentation for details.
# default: targeted
selinuxType: "targeted"

# See install documentation for details.
# default: broadhop
broadhopVar: "broadhop"

# Set true to enable TACACS+ authentication.
# default: FALSE
tacacsEnabled: "FALSE"

# The IP Address of the TACACS+ server
tacacsServer: "127.0.0.1"

# The password/secret of the TACACS+ server.
tacacsSecret: "CPE1704TKS"

# A set of SNMP Network Management Stations.
# NMS can be specified as IP addresses or IP
# addresses. Entries are space separated.
# Hostnames must also be specified in Additional
# Host configuration.
# See install documentation for details.
nmsManagers:

# Low Memory alert threshold %.
# default: 0.1 (10% free)
freeMemPer: "0.1"

# A space separated set of protocol:hostname:port
# entries. UDP is the only supported protocol.
# Example:
# upd:corporate_syslog_ip:514 udp:corporate_syslog_ip2:514
syslogManagers:

# A comma separated set of port values.
# This must match values in the syslog_managers_list.
# default: 514
syslogManagersPorts: "514"

# Port value for the rsyslog proxy server to listen
# for incoming connections
# default: 6515
logbackSyslogDaemonPort: "6515"

# IP address value used in the
# /etc/broadhop/controlcenter/logback.xml
# on the pcrfclient.
# default: lbvip02
logbackSyslogDaemonAddr: "lbvip02"


# High CPU alert threshold.
# The system will alert whenever the usage is
# higher than this value.
# default: 80
```

```
 cpuUsageAlertThreshold: "80"

 # Clear High CPU Trap threshold.
 # The system will generate a clear trap when a
 # High CPU trap has been generated and the CPU
 # usage is lower than this value.
 # default: 40
 cpuUsageClearThreshold: "40"

 # The number of 5 sec intervals to wait between
 # checking the CPU usage.
 # default: 12 (60 seconds)
 cpuUsageTrapIntervalCycle: "12"

 # The SNMP trap community string.
 snmpTrapCommunity: "broadhop"

 #The SNMP read community string.
 snmpRoCommunity: "broadhop"

 #
 monQnsLb:

# The memory alert threshold (0.1 is 10%)
 freeMemoryPerAlert: "0.1"

 # The memory clear threshold (0.3 is 30%)
 freeMemoryPerClear: "0.3"

 #
 monitorReplicaTimeout: "540"

 # Enable SCTP
 # TRUE - feature enabled
 # FALSE - feature disabled
 sctpEnabled: "TRUE"


 # Enables or disables linux firewall on all VMs (IPtables).
 # default: disabled
 firewallState: "disabled"

 # enable SNMP V3.
 # If null, SNMP V3 is disabled.
 # To enabled add the following:
 #   v3User: The SNMP V3 user: REQUIRED
 #   engineId: hex value (ie, 0x0102030405060708):  REQUIRED
 #   authProto: SHA or MD5: REQUIRED
 #   authPass: at least 8 characters:  REQUIRED
 #   privProto: AES or DES: REQUIRED
 #   privPass: OPTIONAL
 snmpv3:
    null
 #  v3User: "cisco_snmpv3"
 #  engineId: "0x0102030405060708"
 #  authProto: "SHA"
 #  authPass: "cisco_12345"
 #  privProto: "AES"
 #  privPass: ""


 # Users
 # There are different categories of users specified for the CPS.
 # All users have the following fields:
```

```
  #
  #  name: The user name. REQUIRED
  #  password: The password for the user. REQUIRED
  #            The password will need to be either in cleartext or
  #            encrypted. Please refer to Install documentation for details.
  #  groups: The groups for the user. Groups are specified as a list
  #          of group names.

  # System Users
  # Note that there must be a system use named sys_user_0
  sysUsers:
    - name: "qns"
      password: "$6$HtEnOu7S$8kkHDFJtAZtJXnhRPrPFI8KAlHFch41OJ405OnCCqO0CFuRmexvCRTk"
      groups:
        - pwauth

    - name: "qns-svn"
      password: "$6$HtEnOu7S$8kkHDFJtAZtJXnhRPrPFI8KAlHFch41OJ405OnCCqO0CFuRmexvCRTk"
    - name: "qns-ro"
      password: "$6$HtEnOu7S$8kkHDFJtAZtJXnhRPrPFI8KAlHFch41OJ405OnCCqO0CFuRmexvCRTk"

  # Hypervisor Users
  hvUsers:
    - name: "root"
      password: "CpS!^246"

  # Other Users for the CPS
  # e.g. Control Center Users
  additionalUsers:
    - name: "admin"
      password: "qns123"
      groups:
        - qns

# Configuration section for feature licenses
# REQUIRED
licenses:
  # Licenses have the following required fields:
  # feature: The name of the feature license.
  # license: The license key for the feature.
  # - feature: "feature 1  Name"
  #   license: "license 1 key string"
   - feature: "MOBILE_CORE"
     license: "xxxxxxx"
   - feature: "RADIUS_AUTH"
     license: "xxxxxxx"

# Configuration section for mongo replica sets
# REQUIRED
replicaSets:
  #
  # Mongo replica sets have the following REQUIRED fields
  # <Mongo Set Identifier> : The database for which the replica
  #                          set is being created.
  #   setName: The name of the replica set
  #   oplogSize: Mongo Oplog size
  #   arbiters: The Arbiter hostnames and ports
  #   arbiterDataPath: The data directory on the arbiter VM
  #   members: List of members for the replica set. Each list element
  #            will be a session manager hostname:port
  #   dataPath: The data directory path on the session manager VMs
  - title: SESSION-SET1
    setName: set01
    oplogSize: 5120
```

```
    arbiters:
    - "pcrfclient01:27717"
    arbiterDataPath: "/var/data/sessions.1"
    members:
      - "sessionmgr01:27717"
      - "sessionmgr02:27717"
    dataPath: "/var/data/sessions.1/1"
- title: SESSION-SET2
    setName: set08
    oplogSize: 5120
    arbiters:
    - "pcrfclient01:37717"
    arbiterDataPath: "/var/data/sessions.1/2"
        members:
      - "sessionmgr01:37717"
      - "sessionmgr02:37717"
    dataPath: "/var/data/sessions.1/2"
    seeds: "sessionmgr01:sessionmgr02:37717"
  - title: BALANCE-SET1
    setName: set02
    oplogSize: 5120
    arbiters:
      - "pcrfclient01:27718"
    arbiterDataPath: "/var/data/sessions.2"
    members:
      - "sessionmgr01:27718"
      - "sessionmgr02:27718"
    dataPath: "/var/data/sessions.2"
  - title: REPORTING-SET1
    setName: set03
    oplogSize: 5120
    arbiters:
     - "pcrfclient01:27719"
    arbiterDataPath: "/var/data/sessions.3"
    members:
      - "sessionmgr01:27719"
      - "sessionmgr02:27719"
    dataPath: "/var/data/sessions.3"
  - title: SPR-SET1
    setName: set04
    oplogSize: 3072
    arbiters:
    - "pcrfclient01:27720"
    arbiterDataPath: "/var/data/sessions.4"
    members:
      - "sessionmgr01:27720"
      - "sessionmgr02:27720"
    dataPath: /var/data/sessions.4
  - title: AUDIT-SET1
    setName: set05
    oplogSize: 3072
    arbiters:
    - "pcrfclient01:27725
    arbiterDataPath: "/var/data/sessions.5"
    members:
      - "sessionmgr01:27725"
      - "sessionmgr02:27725"
    dataPath: "/var/data/sessions.5"
  - title: ADMIN-SET1
    setName: set06
    oplogSize: 3072
    arbiters:
    - "pcrfclient01:27721"
    arbiterDataPath: "/var/data/sessions.6"
```

```
    members:
      - "sessionmgr01:27721"
      - "sessionmgr02:27721"
    dataPath: "/var/data/sessions.6"
  - title: ADMIN-SET2
    setName: set07
    oplogSize: 3072
    arbiters:
    - "pcrfclient01:27731"
    arbiterDataPath: "/var/data/sessions.7"
    members:
      - "sessionmgr01:27731"
      - "sessionmgr02:27731"
    dataPath: "/var/data/sessions.7"

# Configuration section for LDAP/SSSD
   ldapEnabled: “true”
   ldapOnAll:true
   ldapServer: "ldaps://<serverip>:10648"
   ldapSearchBase: "ou=users,dc=cisco,dc=com"
   ldapDefaultBindDn: "uid=admin,ou=system"
   ldapSecret: “secret”
   ldapDefaultUser: “admin”
   ldapOuUser: “users”
   ldapOuGroup: “groups”
   ldapDefaultGroup: “Admin”
   ldapDefaultGroupEditor: “User”
   ldapDcName: "dc=cisco,dc=com"

# Configuration section for DSCP configuration
# OPTIONAL
dscpconfig:
  #
  # dscpconfig have the following fields
  # vmRole - VM type i.e lb/pcrfclient/qns/sessionmgr/udc
  # ipFamily - ipv4 or ipv6 and if empty then ipv4 & ipv6
  # outInterface - interface name i.e eth0/eth1, if empty then apply to any interfaces
  # protocol - tcp/udp/etc.., if empty then apply to any protocol
  # destIp - Specify Destination IP
  # destPort - Specify Destination Port
  # sourcePort - Specify Source Port
  # dscpClass  - Specify DSCP class or value
  # dscpValue - Specify DSCP class or value
   - vmRole: "lb"
     protocol: "tcp"
     outInterface: "eth1"
     destPort: "27717"
     dscpClass: "af11"
   - role: "lb"
     protocol: "udp"
     destIp: "1.1.1.1"
     destPort: "27717"
     dscpClass: "af12"

# Configuration section for Critical File Monitor configuration
#
# CriticalFilesConfig have the following fields
# FileToBeMonitored: Absolute path of file which needs to monitor.
# ActionToBeMonitored: Action for which file needs to monitor. Supported options are wxa (
 w –write, x - execute and a – attribute changes).
---
critFileMonConfig:
---
- fileToBeMonitored: "/etc/hosts"
```

```
  actionToBeMonitored: "wxa"
- fileToBeMonitored: "/etc/shadow"
  actionToBeMonitored: "xa"
- fileToBeMonitored: "/etc/passwd"
  actionToBeMonitored: "xa"
```

MongoDB Authentication Process

- Change mongo user password (Application downtime is involved):
  - Modify password using config PATCH API.
  - Wait for the process to complete.
  - Execute change password script (`/var/qps/install/current/scripts/modules/mongo_change_password.py`) and enter the old password.

    Syntax:

    `/var/qps/install/current/scripts/modules/mongo_change_password.py <old password>`
  - Restart all the JAVA processes.
- Disable mongo authentication (No application downtime is involved):
  - Modify mongo authentication configuration using config PATCH API.
  - Wait for the process to complete.
  - Execute disable mongo authentication script: `/var/qps/install/current/scripts/modules/mongo_auth_upgrade.py`
  - Restart all the JAVA processes.
- Enable mongo authentication (Mongo and application downtime is involved).
  - Modify mongo authentication configuration using config PATCH API.
  - Wait for the process to complete.
  - Execute enable mongo authentication script: `/var/qps/install/current/scripts/modules/mongo_auth_upgrade.py`
  - Restart all the JAVA processes.

/api/system/config/status

Purpose

This API retrieves the status of individual install and deploy tasks run when a new or updated configuration is applied on the Cluster Manager VM.

This API can be called while the installation and deployment tasks are actively running.

The status reports:

- timestamp: timestamp in milliseconds.
- taskname: name of the individual task.
- status:
  - START: start of task.
  - INFO: general information about the task.
  - WARNING: error information about the task.
  - SUCCESS: task was successfully completed.
  - FAILURE: task failed and deployment failed.
- details: information about this task.

Retrieve Deployment Status

To retrieve the deployment status:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/status

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success.

  Example Response:

```
---
[
{"timestamp":"1454367943000","taskName":"CPS Installation","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"Cluman Setup","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"Cluman Setup","status":"SUCCESS","details":"Wait
 for Puppet to complete"},
{"timestamp":"1454367943000","taskName":"Post Install","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"SyncSvn","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"SyncSvn","status":"WARNING","details":"Failed
 to sync SVN."},
{"timestamp":"1454367943000","taskName":"SyncSvn","status":"SUCCESS","details":""},
{"timestamp":"1454367943000","taskName":"build_set","status":"START","details":"Building
 replica sets"},
{"timestamp":"1454367943000","taskName":"build_set","status":"INFO","details":"Wrote
mongo config"},
{"timestamp":"1454367943000","taskName":"build_set","status":"INFO","details":"Syncing
 mongo config to other hosts"},
{"timestamp":"1454367943000","taskName":"build_set","status":"SUCCESS","details":"Replica
 sets have been created successfully"},
{"timestamp":"1454367943000","taskName":"SetPriority","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"SetPriority","status":"SUCCESS","details":""},
```

```
{"timestamp":"1454367943000","taskName":"AddAdditionalUsers","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"AddAdditionalUsers","status":"SUCCESS","details":""},
{"timestamp":"1454367943000","taskName":"Licenses","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"Licenses","status":"SUCCESS","details":""},
{"timestamp":"1454367943000","taskName":"Post Install","status":"SUCCESS","details":""}
]
```


The deployment process is complete when the following response is received: `"Post Install","status":"SUCCESS"`

Note The amount of time needed to complete the entire deployment process depends on the number of VMs being deployed, as well as the hardware on which it is being deployed. A typical deployment can take 45 minutes or more.

Startup status logs are written to: `/var/log/startupStatus.log` on the Cluster Manager VM.

API logs are written to: `/var/log/orchestration-api-server.log`

Refer to the /api/system/config/status , on page 88 to determine the readiness status of the CPS cluster.

/api/system/status/cps

Purpose

This API returns the readiness status of CPS cluster.

Cluster Readiness

This API returns the "readiness" status of the CPS cluster.

The cluster is deemed "ready" when Puppet has run to completion on all VMs and the Replica set creation is complete on the Session Manager VMs. The Orchestrator can use this API to check when the cluster is ready so that it can then invoke the Service Creation APIs.

This API reports an aggregate status of MongoDB replica sets, qns processes, and the cluster (Puppet) for all VMs.

This API will timeout after 150 seconds.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/status/cps

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** JSON
- **Response:**

The following example shows the readiness status for a CPS cluster:

```
{
    "clusterStatus": "ready",
    "mongoStatus": "ready",
    "qnsStatus": "ready"
}
```

mongoStatus and clusterStatus can report "ready", "not ready", or "error". qnsStatus can report "ready" or "not ready". If mongoStatus reports an "error" status, the clusterStatus also reports an "error" status.

If any database replica-sets are reporting "ok", but members are "off-line", mongoStatus reports "not ready".

If any of the replica-sets are down or in an error state, mongoStatus reports "error".

- **Error Codes:**
  - 200 OK: success
  - 404: Unknown entity
  - 500: Script config not found
  - 500: CPS status job interrupted
  - 500: CPS status job timeout
  - 500: CPS status job termination interrupted
  - 500: Failed retrieval of CPS status job results

API logs are at written to: `/var/log/orchestration-api-server.log`

/api/system

Purpose

This API is to used to determine the current state of the CPS system, and if necessary, override it in the event the reported state does not match the actual system state.

Many CPS orchestration APIs are accepted only when the CPS system is in a particular state. This API provides a method of overriding the reported API system state. It does not rectify or correct the underlying issue. For example setting the state to pre_deploy does not un-deploy the CPS deployment.

API logs are at written to: `/var/log/orchestration-api-server.log`

Retrieve the Current API State

To determine the current system state:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200: OK.

  Example Response:

```
{
  "state": "pre_config"
}
```

  This API can be used at any time.

The following states can be reported:

- **pre_config:** no configuration has been loaded onto the system (/api/system/config).
- **pre_deploy:** a configuration has been loaded, but not applied (api/system/config/action/apply).
- **deploying:** the system is in the process of being deployed.
- **deployed:** the system has finished the installation/deployment.
- **upgrading:** the system is in the process of being upgraded.
- **busy:** the system is currently processing an operation.

Override the Current API State

Caution This API should only be used as directed by a Cisco representative. Improper use can cause irreparable harm to the CPS deployment.

To override the current system state:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/json
- **Method:** POST
- **Payload:** JSON payload with the new state specified as one of the following options: `pre_config`, `pre_deploy`, `deploying`, `deployed`, or `upgrading`.

  For example:

```
{
  "state": "pre_config"
}
```

- **Response Codes:** 400: Invalid state, please use: [pre_config, pre_deploy, deploying, deployed, upgrading]; 500: System error. See logs.

  Example Response:

```
{
  "state": "pre_config"
}
```

Upgrade APIs

Caution The Upgrade API can trigger kernel upgrade if kernel version is updated in new CPS version. So all the necessary precautions prior to kernel upgrade of CPS VMs must be taken before an upgrade is triggered through orchestration API. If kernel is upgraded then VMs will be rebooted.

Upgrade API Prerequisites

The following sequence of commands should be executed in OpenStack before running the CPS upgrade APIs.

Note These commands are for illustration purpose only and do not override any setup specific constraints. The specific commands may differ on your environment.

Step 1 Create a glance image of the new CPS ISO.

```
glance image-create --name <name of CPS ISO> --disk-format iso --container-format bare --is-public
True --file <Absolute path to new CPS ISO>
```

Step 2 Create a cinder volume based on the glance image of the new CPS ISO.

```
cinder create --image-id <glance image id of new CPS ISO> --display-name <name of new CPS ISO volume>
--availability-zone <optional zone> <size of ISO in GBs>
```

Step 3 Detach the existing CPS ISO volume from the Cluster Manager VM.

```
nova volume-detach <nova instance ID of cluman> <cinder volume ID of old CPS ISO volume>
```

Step 4 Attach the new CPS ISO volume to the Cluster Manager VM. This will require either the name of device at which volume is attached to the Cluster Manager, or "auto" to attach the volume as any available device name. In either case, the following command will output name of device to which new CPS ISO volume is attached.

```
nova volume-attach <nova instance ID of cluman> <cinder volume ID of new CPS ISO volume> <Name of
device, e.g. /dev/vdb or auto for autoassign>
```

/api/system/upgrade

Purpose

The following APIs are used to mount and unmount an ISO image to the Cluster Manager VM, trigger an out-of-service upgrade of a CPS deployment, and view the status of the upgrade.

Note Before invoking any of these APIs, refer to Upgrade API Prerequisites, on page 93.

Logs are at written to: `/var/log/orchestration-api-server.log` on the Cluster Manager VM.

Unmount ISO

To unmount an existing CPS ISO image from `/mnt/iso` directory on the Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/upgrade/action/unmount

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/json
- **Method:** POST
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 400: The mount parameters are invalid; 500: System Error. See logs.

Note After invoking this API, it is recommended to detach the ISO image from the Cluster Manager VM using relevant command in OpenStack.

Mount ISO

Note Before invoking this API:

- A new cinder volume must be created in OpenStack based on the CPS ISO, and then attached to the Cluster Manager VM using relevant command in OpenStack. Refer to Upgrade API Prerequisites, on page 93 for more details.
- Run the **lsblk** command on the Cluster Manager VM to check the device name before running mount API. This needs to be checked after the CPS ISO volume has been attached to the Cluster Manager VM.

To mount the CPS ISO image onto /mnt/iso directory on the Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/upgrade/action/mount

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/json
- **Method:** POST
- **Payload:**

```
{
"deviceName": "<filename of the block device at which the cinder volume is attached"
Ex: "/dev/vdb>"
}
```

`/dev/vdb` is for illustration only. Replace with the device name to which the CPS ISO volume is attached on your Cluster Manager VM.

Example:

```
{
 "deviceName": "/dev/vdb"
}
```

- **Response Codes:** 200 OK: success; 400: The mount parameters are invalid; 500: System Error. See logs.

Upgrade CPS

Caution This API must only be used during a planned maintenance window. This API does not perform an in-service software upgrade. CPS processes will be restarted during this process and traffic will be affected.

This API can only be used once the CPS has been deployed and is in a ready state. Prior to that time this API will not be available.

To upgrade CPS using the mounted ISO:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/upgrade/action/apply

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/json
- **Method:** POST
- **Payload:**

  type: Only `"OUT_OF_SERVICE"` is supported.

  config: The SVN/policy repository configuration to back up prior to upgrade. This repository will be backed up and restored during the upgrade.

installType: The type of CPS deployment. Only `mobile` is supported.

Example:

```
{
"config": "run",
"installType": "mobile",
"type": "OUT_OF_SERVICE"
}
```

- **Response Codes:** 200 OK: success; 400: The input parameters are malformed or invalid.

The upgrade logs are at written to: /var/log/install_console_<*date_time*>.log on the Cluster Manager VM.

Note If you want to upgrade from 18.2.0 release to 18.3.0 release using option 2 (offline) upgrade, you need to execute the following steps as option 2 upgrade fails for the first run:

1. `rm /var/tmp/upgrade_status` and start monit manually using `service monit start`
2. Re-run the upgrade API again.

For subsequent option 2 upgrades, you do not need to execute the above mentioned workaround.

If you do not want to use the above workaround, contact your Cisco Technical Representative.

Upgrade Status

To view the status of an upgrade:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/upgrade/status

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 500: Script config not found

  Example Response:

```
{
        "status": "In-Progress"
    }
```

  - Not-Started - No upgrade has been initiated
  - In-Progress - Upgrade is currently underway
  - Completed - Upgrade has completed
  - Error - There is a problem with the upgrade

This API is only valid after the operator has issued an upgrade.

System Configuration APIs

/api/system/mongo/config

Purpose

This API is used to retrieve the contents of `/etc/broadhop/mongoConfig.cfg`. This API is also used to add members to existing Mongo replica sets.

Important This API does **not** support modifications to any other parameters within the Mongo configuration. It only add members to existing Mongo replica sets.

Important While choosing mongo ports for replica-sets, consider the following:

- Port is not in use by any other application. To check it, login to VM on which replica-set is to be created and execute the following command:

  ```
  netstat -lnp | grep <port_no>
  ```

  If no process is using same port then port can be chosen for replica-set for binding.

- Port number used should be greater than 1024 and not in ephemeral port range i.e, not in between following range :

  net.ipv4.ip_local_port_range = 32768 to 61000

API logs are at written to: `/var/log/orchestration-api-server.log`

Workflow

1. Retrieve Current Mongo Configuration, on page 97
2. Manually edit the YAML file retrieved in step 1 to add members to the existing replica sets.
3. Load Updated Configuration, on page 99
4. Apply Loaded Configuration, on page 100

Retrieve Current Mongo Configuration

To retrieve (GET) the current configuration:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/mongo/config

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error
  - Example Response (YAML format): HA Setup

```
---
- title: "SESSION-SET1"
  setName: "set01"
  oplogSize: "5120"
  arbiters:
  - "pcrfclient01:27717"
  arbiterDataPath: "/var/data/sessions.1"
  members:
  - "sessionmgr01:27717"
  - "sessionmgr02:27717"
  dataPath: "/var/data/sessions.1"
- title: "BALANCE-SET1"
  setName: "set02"
  oplogSize: "5120"
  arbiters:
  - "pcrfclient01:27718"
  arbiterDataPath: "/var/data/sessions.2"
  members:
  - "sessionmgr01:27718"
  - "sessionmgr02:27718"
  dataPath: "/var/data/sessions.2"
- ...
```

  - Example Response (YAML format): GR Setup

```
- title: "SESSION-SET1"
  setName: "set01"
  oplogSize: "1024"
  arbiters:
  - "arbiter-site3:27717"
  arbiterDataPath: "/var/data/sessions.1"
  siteId: "SITE1"
  members:
    - sessionmgr02-site1:27717
    - sessionmgr01-site1:27717
  dataPath: /var/data/sessions.1/set1
  primaryMembersTag: "SITE1"
  secondaryMembersTag: "SITE2"
  shardCount: "4"
  hotStandBy: "false"
  seeds: "sessionmgr01:sessionmgr02:27717"
```

  OR

```
- title: "SESSION-SET1"
```

```
setName: "set01"
oplogSize: "1024"
arbiters:
- "arbiter-site3:27717"
arbiterDataPath: "/var/data/sessions.1"
primaryMembers:
  - "sessionmgr02-site1:27717"
  - "sessionmgr01-site1:27717"
secondaryMembers:
  - "sessionmgr02-site2:27717"
  - "sessionmgr01-site2:27717"
dataPath: "/var/data/sessions.1"
hotStandBy: "false"
shardCount: "4"
seeds: "sessionmgr01:sessionmgr02:27717"
primaryMembersTag: "SITE1"
secondaryMembersTag: "SITE2"
siteId: "SITE1"
```

Note The response includes the complete Mongo configuration in YAML format.

Load Updated Configuration

Note This API can only be used once CPS has been deployed and is in a ready state. Prior to that time this API is not available.

Use this API to load an updated Mongo configuration on the CPS Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/mongo/config/

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** PUT
- **Payload:** Include the YAML configuration file in the PATCH request. The entire contents of the Mongo config must be included.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

  Example Response:

  The updated contents of `/etc/broadhop/mongoConfig.cfg` is returned in the response in YAML format.

Note After using this API to load the updated mongo configuration, you must apply the configuration. Refer to Apply Loaded Configuration, on page 100.

Apply Loaded Configuration

Note This API can only be used once the CPS has been deployed and is in a ready state. Prior to that time this API is not available.

Use this API to apply the updated Mongo configuration that you loaded using Load Updated Configuration, on page 99:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/mongo/action/addMembers

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/json
- **Method:** POST
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

  This API returns immediately and does not wait for the members to be added. Refer to the log file to check the status.

  Example Response:

```
{
    "logfile": "/var/log/broadhop/scripts/orch_api_03122016_203220.log"
}
```

/api/system/config/hosts

Purpose

This API is used to retrieve the current list of deployed CPS hosts, and to add or remove Policy Server (QNS), SessionMgr, and Policy Director (Load Balancer) hosts to the CPS cluster. This enables an orchestrator to increase (scale up) or decrease (scale down) the session processing capacity of the CPS cluster.

Important To scale up, you must create VMs using heat or nova boot commands. However, already existing stacks cannot be used to scale up using heat.

Note Only Policy Server (QNS) and SessionMgr hosts can be scaled down. Policy Director (Load Balancer) hosts cannot be scaled down.

Retrieve Current List of Deployed Hosts

To retrieve (GET) the current list of hosts deployed in the CPS cluster:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/hosts

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

  Example Response (YAML format):

```
---
- name: "lb01"
  alias: "lb01"
  interfaces:
  - network: "Internal"
    ipAddress: "172.16.2.201"
  - network: "Management"
    ipAddress: "172.18.11.154"
  - network: "Gx"
    ipAddress: "192.168.2.201"
- ...
```

  Note The example response shown above is abbreviated. The response includes the complete list of configured hosts.

Add New Policy Server (QNS), Session Manager, and Policy Director (Load Balancer) Hosts

This API adds additional Policy Server (QNS), SessionMgr, and/or Policy Director (Load Balancer) hosts to an existing deployment. The API uses the PATCH method, which adds new hosts without affecting the existing configured hosts.

Policy Server (QNS), SessionMgr, and/or Policy Director (Load Balancer) VMs must be added in pairs (for example, qns05, qns06 and sessionmgr03, sessionmgr04). Attempts to add odd numbers of VMs are rejected.

Before issuing this API, you must create the additional VMs using Heat or Nova boot commands. For example, to create two additional Policy Server VMs (qns05, qns06):

```
nova boot --config-drive true --user-data=qns05-cloud.cfg --image "base_vm" --flavor "qps"
 --nic
net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.28" --availability-zone
"az-2:os8-compute-2.cisco.com" "qns05"

nova boot --config-drive true --user-data=qns06-cloud.cfg --image "base_vm" --flavor "qps"
 --nic
net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.29" --availability-zone
"az-2:os8-compute-2.cisco.com" "qns06"
```

Note To add SessionMgr VMs, refer to /api/system/config/replica-sets, on page 105 to configure additional replica sets on newly deployed Session Mgr VMs.

When this API call completes, the Cluster Manager configuration is updated and all new VMs are deployed asynchronously.

Note The amount of time needed to complete the process depends on the number of VMs being deployed.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/hosts

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** PATCH
- **Payload:** Include the YAML configuration file in the PATCH request. Use the `op: add` parameter to add a host. Only the new hosts should be defined in the YAML configuration file submitted in the API request.

  Sample Payload:

  ```
  ---
  - op: add
    name: "qns05"
    alias: "qns05"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.28"
  - op: add
    name: "qns06"
    alias: "qns06"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.29"
  ```

- **Response Codes:** 200 OK: success; 400: Invalid data; 500: System error

To verify the configuration was properly loaded, perform another GET to http://<*Cluster Manager IP*>:8458/api/system/config/hosts

After issuing this API, /api/system, on page 91 reports a "busy" state. Once the operation is complete, it reports a "deployed" state.

Additionally, the /api/system/config/status , on page 88 can be used to monitor the progress of individual steps of the operation.

Status logs are also written to: `/var/log/startupStatus.log` on the Cluster Manager VM.

API logs are written to: `/var/log/orchestration-api-server.log` on the Cluster Manager VM.

In case of any errors, check the API log file `/var/log/orchestration-api-server.log` and do the following:

- Verify if puppet on the new Policy Director (Load Balancer) VM is completed successfully.
- In case of diameter calls issue, verify if puppet on lb01/02 VMs is completed successfully and haproxy-diameter configuration is updated. Also, verify if Policy Builder configuration for the new LB VMs is properly updated.
- Verify if `diagnistics.sh` status is clean after Policy Builder update.

Remove Policy Server (QNS) and Session Manager Hosts

This API removes Policy Server (QNS) and/or SessionMgr hosts from an existing deployment.

Note Only Policy Server (QNS) and SessionMgr hosts can be removed from an existing deployment. Policy Director (Load Balancer) hosts cannot be removed.

Caution Before removing any SessionMgr hosts, you must remove the replica-sets configured on those hosts using the /api/system/config/replica-sets, on page 105

Policy Server (QNS) VMs and SessionMgr VMs must be removed in pairs (for example qns05, qns06 and sessionmgr03, sessionmgr04). Attempts to remove odd numbers of VMs are rejected.

This API removes the specified VMs from the Cluster Manager configuration only. After issuing this API, the orchestrator terminates the VMs in OpenStack.

-
- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/hosts

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** PATCH

- **Payload:** Include the YAML configuration file in the PATCH request. Use the `op: remove` parameter to remove a host. Only the hosts which are to be removed should be defined in the YAML configuration file submitted in the API request.

Important Using `op: remove` parameter, only the hosts configuration is removed and not the actual VMs. You need to use nova commands to remove the VMs. For more information on nova commands, refer to OpenStack commands.

Sample Payload:

```
---
- op: remove
  name: "qns05"
  alias: "qns05"
 - op: remove
  name: "qns06"
  alias: "qns06"
```

After issuing this API, /api/system, on page 91 reports a "busy" state. Once the operation is complete, it reports a "deployed" state.

Additionally, the /api/system/config/status , on page 88 can be used to monitor the progress of individual steps of the operation.

Status logs are also written to: `/var/log/startupStatus.log` on the Cluster Manager VM.

API logs are written to: `/var/log/orchestration-api-server.log` on the Cluster Manager VM.

Configuration Parameters - Hosts

The following parameters can be defined in the Hosts YAML configuration file:

Table 10: Configuration Parameters - Hosts

Parameter	Description
`- op:`	The operation to be performed for this host, either `add` or `remove`.
`name:`	Defines the hostname of the VM. This name must be resolvable in the enterprise DNS environment.
`alias:`	Defines the internal host name used by each CPS VMs for internal communication, such as sessionmgr03 or qns05.
`interfaces:`	This section defines the network interface details for the VM.
`- network:`	Defines the CPS VLAN network name for the VM. QNS VMs are typically assigned to the "Internal" VLAN, and SessionMgrs are typically assigned both to "Internal" and "Management" VLANs.
`ipAddress:`	Defines the IP address of the VM.

/api/system/config/replica-sets

Purpose

This API is used to retrieve the current list of replica-sets for the Session database, to add additional replica-sets, or remove replica-sets.

Retrieve Current Replica-sets

To retrieve (GET) the current list of replica-sets configured for the Session database:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/replica-sets

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

Example Payload (YAML format): HA Setup

```
---
- title: "SESSION-SET1"
  setName: "set01"
  oplogSize: "5120"
  arbiters:
  - "pcrfclient01:27717"
  arbiterDataPath: "/var/data/sessions.1"
  members:
  - "sessionmgr01:27717"
  - "sessionmgr02:27717"
  dataPath: "/var/data/sessions.1"
 - ...
```

Example Payload (YAML format): GR Setup

```
---
 - title: "SESSION-SET1"
    setName: "set01"
    oplogSize: "1024"
    arbiters:
    - "arbiter-site3:27717"
    arbiterDataPath: "/var/data/sessions.1"
    siteId: "SITE1"
    members:
      - "sessionmgr02-site1:27717"
      - "sessionmgr01-site1:27717"
    dataPath: "/var/data/sessions.1/set1"
    primaryMembersTag: "SITE1"
    secondaryMembersTag: "SITE2"
    shardCount: "4"
    hotStandBy: "false"
    seeds: "sessionmgr01:sessionmgr02:27717"
```

If the user has configured `primaryMembersTag:` and `secondaryMembersTag:` parameters, then only these parameters will be visible in case of API GET is called to fetch configuration details. There will be single tag specified for SPR/balance/session geo tagging. The value will be matched with any one of the parameters mentioned in `qns.conf` for geo site tagging.

Note The response includes the complete list of configured replica-sets.

Important While choosing mongo ports for replica-sets, consider the following:

- The port must not be in use by any other application. To check the port number, login to VM on which replica-set is to be created and execute the following command:

  ```
  netstat -lnp | grep <port_no>
  ```

  If no process is using the port, then the port number can be chosen for replica-set for binding.

- The port number used should be greater than 1024 and not be in the ephemeral port range i.e, not in between following range: 32768 to 61000.

- While configuring mongo ports in a GR environment, there should be a difference of 100 ports between two respective sites. For example, consider there are two sites: Site1 and Site2. For Site1, if the port number used is 27717, then you can configure 27817 as the port number for Site2. This is helpful to identify a mongo member's site. By looking at first three digits, one can decide where the mongo member belongs to. However, this is just a guideline. You should avoid having mongo ports of two different sites to close to each other (for exampl, 27717 on Site-1 and 27718 on Site2).

  Reason: The reason is that the `build_set.sh` script fails when you create shards on the site (for example, Site1). This is because the script calculates the highest port number in the `mongoConfig` on the site where you are creating shards. This creates clash between the replica-sets on both sites. Since the port number which it allocates might overlap with the port number of `mongoConfig` on other site (for example, Site2). This is the reason why there should be some gap in the port numbers allocated between both the sites.

Add Replica-sets

This API configures additional replica-sets on newly deployed SessionMgr VMs. This API uses the PATCH method, which adds replica-sets without affecting the existing configured replica-sets.

When this API call completes, the Cluster Manager configuration is updated and all new replica-sets are created asynchronously.

Note The amount of time needed complete the process depends on the number of replica-sets being deployed.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/replica-sets

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** PATCH
- **Payload:** Include the YAML configuration file in the PATCH request. Use the `op: add` parameter to add a replica-set. Only the new replica-sets should be defined in the YAML configuration file submitted in the API request.

  Sample Payload (YAML format): HA Setup

```
---
- op: add
   title: SESSION
   arbiters:
   - "pcrfclient01"
   instances: 2
   members:
    - "sessionmgr03"
    - "sessionmgr04"
```

  Sample Payload (YAML format): GR Setup

```
---
- op: add
   title: SESSION
   arbiters:
   - "pcrfclient01"
   instances: 2
   members:
    - "sessionmgr03"
    - "sessionmgr04"
       -primaryMembersTag: "sitename"
       -secondaryMembersTag: "sitename"
```

- **Response Codes:** 200 OK: success; 400: Invalid data; 500: System error

To verify the configuration was properly loaded, perform another GET to http://<*Cluster Manager IP*>:8458/api/system/config/replica-sets

The status of this API call is reported in http://:<*Cluster Manager IP*>8458/api/system/config/status

Status logs are also written to: `/var/log/startupStatus.log` on the Cluster Manager VM.

API logs are written to: `/var/log/orchestration-api-server.log` on the Cluster Manager VM.

Remove Replica-sets

This API removes replica-sets from deployed SessionMgr VMs. This API uses the PATCH method.

This API must be issued before removing any Session Manager VMs during a scale down of the CPS Cluster using the /api/system/config/hosts, on page 100 API.

After issuing this API, the /api/system/config/status , on page 88 API can be used to monitor the removal of the ring-sets and the replica-sets. After the operation has completed, this API will return a SUCCESS status for the operation.

While the operation is ongoing, performing a GET with the /api/system/config/, on page 58 API returns a BUSY status for the operation. No other API operations are allowed while the system is in this state.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/replica-sets

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** PATCH
- **Payload:** Include the YAML configuration file in the PATCH request. Only the replica-sets which are to be removed should be defined in the YAML configuration file submitted in the API request.

  Sample Payload (YAML format):

```
---
- op: remove
   title: SESSION
   setName: set01
   arbiters:
   - "pcrfclient01"
   instances: 2
   members:
    - "sessionmgr03"
    - "sessionmgr04"
```

- **Response Codes:** 200 OK: success; 400: Invalid data; 500: System error

To verify the configuration was properly loaded, perform another GET to http://<*Cluster Manager IP*>:8458/api/system/config/replica-sets

The status of this API call is reported in http://:<*Cluster Manager IP*>8458/api/system/config/status

Status logs are also written to: `/var/log/startupStatus.log` on the Cluster Manager VM.

API logs are written to: `/var/log/orchestration-api-server.log` on the Cluster Manager VM.

Adding/Updating Shard Count

Use this API to create shards. This API also supports existing scaling session replica-set and adding shards to existing session replica-sets.

Shards must be created during installation after the qns restart process (post install step).

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/replica-sets/

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** PATCH
- **Payload:**

  Sample Payload (YAML format) for scaling new replica-set:

  Note hotStandBy, shardCount and seeds are optional parameters.

```
---
- op: "add"
  title: "SESSION"
  instances: "1"
  arbiters:
  - "pcrfclient01"
  members:
  - "sessionmgr01"
  - "sessionmgr02"
 hotStandBy: “true”
 shardCount: “4”
seeds: “sessionmgr01:sessionmgr02”
```

  Sample Payload (YAML format) for modifying the replica-set configuration:

  Note hotStandBy, shardCount and seeds are required parameters.

```
---
- op: "modify-shards"
  setName: "set10"
  hotStandBy: "true"
  shardCount: "5"
  seeds: "sessionmgr01:sessionmgr02:27820"
```

- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

To verify the configuration was properly loaded, perform another GET to http://<*Cluster Manager IP*>:8458/api/system/config/replica-sets

Update Priority

Priorities can be set in descending order using PATCH request.

In HA environment, priorities can be set for all replica sets of a particular replica database like session, admin, and so on. Also, you can set a particular replica-set under specific replica database.

In GR environment, priorities can be set for particular site and all replica-sets of a particular replica database like session, SPR, and so on. Also, you can set a particular replica-set under specific replica database. siteId parameter is mandatory in GR scenario.

Note It is required that replica-set are created before priority can be set. During installation, priority is added for all replica sets. In case a member is added using `addMember` API. it is required to execute set-priority API to set priority for given replica-set.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/replica-sets

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** PATCH
- **Payload:** To change the priority of more than one database type you must include another block in the request.

  Example Payload (YAML format): HA Setup

```
---
- op: "set-priority"
  title: "SESSION"
  setName: “set01”

- op: "set-priority"
  title: "SPR"
```

  Example Payload (YAML format): GR Setup

```
---
 - op: "set-priority"
  title: "SESSION"
  siteId: "SITE1"
```

  Note For HA, *title* parameter is mandatory. For GR, *title* and *siteId* are mandatory parameters. *setName* is optional parameter for both HA and GR deployments.

- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

Configuration Parameters - Replica-set

The following parameters can be defined in the ReplicaSet YAML configuration file:

Table 11: Configuration Parameters - Replica-set

Parameter	Description
`- op:`	The operation to be performed for this replica set, either `add` or `remove`.
`title:`	The database for which the replica set is being created. The only option supported is SESSION.
`arbiters:`	The hostnames of the arbiters.
`instances:`	The number of replica set instances to create. For each replica set, the API will automatically generate the next available port, for example 27737, 27757 and so on.
`members:`	The list of members for the replica set. Each list element will be a session manager hostname, for example sessionmgr03.
`- <member>`	List each member hostname on a separate line, for example: sessionmgr03 sessionmgr04 The port for each Session Manager is automatically generated by the API.
`siteId:`	This parameter can be either local or remote site.
`title:`	This parameter is used to represent replica-set of a particular type. For example, session, SPR, and so on.
`hotStandBy:`	This parameter is used to defined whether the created shard is to be used for primary or backup database. If set to true, then created shard will be used for backup database. If the parameter is not configured, then the created shard will be used for non backup database. By default, this parameter is not configured.
`shardCount:`	This parameter is used to defined the number of shards to be created. In modify request, shards can only be increased.
`seeds:`	This parameter is used to defined sharding for multiple sessionmgr VMs. Enter the sessionmgr VM name with port separated by a colon (:) with each pair separated by a colon (:). Example: sessionmgr01:sessionmgr02:27717, sessionmgr03:sessionmgr04:27717
`primaryMembersTag:`	This parameter is used to define the sitename for primary members of a replica set for geo tagging. This is an optional parameter.

Parameter	Description
`secondaryMembersTag:`	This parameter is used to define the sitename for secondary members of a replica set for geo tagging. This is an optional parameter.
`siteId:`	In GR setup, this parameter is used to define the replica-set corresponding to the given site.

Note The ReplicaSet API automatically generates values for the following parameters: `setname`, `oplogSize`, and `dataPath`. The default oplogSize is 5120 MB.

/api/system/config/replica-sets/action/sync-mongo

Purpose

This API is used to copy the `/etc/broadhop/mongoConfig.cfg` file from one site to another. API can be called on local cluman which in turn calls the remote cluman and update its data. The parameter `remoteClumanIp` needs to be configured using /api/system/config/config, on page 112. This is required before syncing operation can be started.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/replica-sets/action/sync-mongo

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** POST
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

/api/system/config/config

Purpose

This API is used to retrieve or update the 'config' section of the CPS cluster configuration.

API logs are at written to: `/var/log/orchestration-api-server.log`

Retrieve Current Configuration

To retrieve (GET) the 'config' section of the configuration currently loaded on the CPS cluster:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/config

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200: OK.

  Example Response (YAML format):

```
---
config:
  qpsUser: "sys_user_0”
  selinuxState: "disabled”
  selinuxType: "targeted”
  ...
  sysUsers:
     ...
  hvUsers:
     ...
  additionalUsers:
     ...
```

Note The example response shown above is abbreviated. The response will include the complete list of parameters from the 'config' section of the consolidated configuration.

Update Configuration

This API modifies the parameters within the 'config' section of the consolidated configuration on an existing deployment. This API uses the PATCH method, which enables you to modify specific parameters without needing to submit the entire configuration.

Note Only new sysUsers and additionalUsers can be added.

Modifying existing sysUsers and additionalUsers is not supported.

Adding new or modifying existing hvUsers is not supported.

When this API call completes, the Cluster Manager configuration is updated and the new configuration is then pushed to all CPS VMs.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/config

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** PATCH
- **Payload:** Include the YAML configuration file in the PATCH request. Only the modified parameters should be defined in the YAML file.

  For a list of parameters which can be defined in this file, refer to the parameters defined in the config, sysUsers, hvUsers, and additionalUsers sections listed in Configuration Parameters - HA System, on page 62.

  Sample Payload (YAML format):

```
---
  selinuxType: "permissive"
  firewallState: "enabled"
  selinuxState: "enabled"
  snmpv3:
    v3User: "cps-snmp"
    engineId: "4321"
    authPass: "snmp123"
    privPass: "snmp321"
  tacacsEnabled: "TRUE"
  firewallState: "enabled”
  additionalUsers:
    - name: orchuser
      password: CpS!^246
      groups:
       - qns
```

Note `root` user group is not authorized group for Control Center.

To add new TACACS configuration to an existing CPS deployment, use the PATCH method:

Sample Payload (YAML format):

```
---
tacacsEnabled = "TRUE"
tacacsServer = "127.0.0.1"
tacacsSecret = "CPE1704TKS"
```

Note The PATCH method will re-run puppet on all the VMs.

'config' section also supports the following "extra" TACACS parameters:

Sample Payload (YAML format):

```
tacacsService = "pcrflinuxlogin"
tacacsProtocol = "ssh"
tacacsTimeout = "5"
tacacsDebug = "0"
```

- **Response Codes:** 200 OK: success; 400: Invalid data; 500: System error

To verify the configuration was properly loaded, perform another GET to http://<*Cluster Manager IP*>:8458/api/system/config/config

The status of this API call is reported in http://:<*Cluster Manager IP*>8458/api/system/config/status

Status logs are also written to: `/var/log/startupStatus.log` on the Cluster Manager VM.

API logs are written to: `/var/log/orchestration-api-server.log` on the Cluster Manager VM.

/api/system/config/additional-hosts

Purpose

This API enables you to configure new peer nodes such as PCEF, NTP, NMS, and so on, by modifying the `/etc/hosts` files on all CPS VMs.

The API logs are written in the `/var/log/orchestration-api-server.log` and `/var/log/startupStatus.log` files.

Note This API does not add a CPS VM to the CPS cluster.

Retrieve AdditionalHosts Configuration

To retrieve (GET) the AdditionalHosts configuration from the CPS Cluster Manager VM:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/additional-hosts

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** GET
- **Payload:** There is no payload
- **Response Codes:** 200 OK: success

  Example Response (YAML format):

```
---
 - name: "Host1 name"
   alias: "Host1 internal name"
   ipAddress: "Host1 IP address"
```

```
- name: "Host2 name"
  alias: "Host2 internal name"
  ipAddress: "Host2 IP address"
- name: "Host3 name"
  alias: "Host3 internal name"
  ipAddress: "Host3 IP address"
```

Add or Update AdditionalHosts Entry

This API adds or updates a new AdditionalHosts entry in the configuration file.

When this API call completes, the Cluster Manager is configured with the new `/etc/hosts` file. All the other deployed VMs are then updated asynchronously and the status is reported in http://:<*Cluster Manager IP*>8458/api/system/config/status.

To add or update an AdditionalHosts configuration:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/additional-hosts

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see HTTPS Support for Orchestration API, on page 49.

- **Header:** Content-Type: application/yaml
- **Method:** PUT
- **Payload:** Include the YAML configuration file in the PUT request.

  Sample Payload (YAML format):

```
---
 - name: "Host name"
   alias: "Host internal name"
   ipAddress: "Host IP address"
 - name: "NewHost name"
   alias: "NewHost internal name"
   ipAddress: "NewHost IP address"
```

Important

- To add or update AdditionalHosts, update new payload with existing additional hosts information along with new or updated additional hosts. This request replaces all the additional hosts with new additional hosts information.
- To modify or delete AdditionalHosts, update new payload with modified or deleted additional hosts and perform PUT request. This request replaces additional hosts information in the `/etc/hosts` file of both Cluster Manager and CPS VMs.
- To verify that the AdditionalHosts configuration is properly loaded, perform another GET request to http://<*Cluster ManagerIP*>:8458/api/system/config/additional-hosts.

- **Response Codes:** 200 OK: success; 400: malformed or invalid; 500: system error

Configuration Parameters - AdditionalHosts

The following parameters can be defined in the AdditionalHosts YAML configuration file:

Parameter	Description
`- name:`	Defines the hostname of the VM. This name must be resolvable in the enterprise DNS environment.
`alias:`	Defines the internal host name used by CPS nodes for internal communication, such as qns01.
`ipAddress:`	Defines the IP address to use in the `/etc/hosts` file.

Secondary Key Ring Configuration

You can create ring during installation for HA or GR systems. If the ring creation fails during installation, you can use APIs to recreate the ring.

The following APIs can be used to create ring configuration:

- During fresh install you can use `http://<cluman-ip>:8458/api/system/config` and `http://<cluman-ip>:8458/api/system/config/action/apply` to create replica-set configuration for all replica-sets and apply it.
- Updated replica-sets (for example, used in scale up of replica-sets) using PATCH method: `http://<cluman-ip>:8458/api/system/config/replica-sets`
- Load updated configuration using PUT method: `http://<cluman-ip>:8458/api/system/mongo/config`

Note You cannot disable ring configuraton.

- Example for HA: replicaSet YAML changes during add replica-set

```
- op: "add"
  title: "SESSION"
  instances: "1"
  arbiters:
  - "pcrfclient01"
  members:
  - "sessionmgr01"
  - "sessionmgr02"
  shardCount: "4"
```

Use PATCH API `http://<cluman-ip>:8458/api/system/config/replica-sets` to create ring and replica-set.

Verify ring configuration by executing the following command:

`echo "db.cache_config.find()" | mongo sessionmgr01:27721/sharding` <-- Change host name and port according to your deployment

- Example for GR:

```
- op: "add"
  title: "SESSION"
  instances: "1"
  arbiters:
  - "arbiter-site3"
  primaryMembers:
  - "sessionmgr01-site1"
  - "sessionmgr02-site1"
  secondaryMembers:
  - "sessionmgr01-site2"
  - "sessionmgr02-site2"
  seeds: "sessionmgr01:sessionmgr02"
  shardCount: "4"
```

Use PATCH API `http://<cluman-ip>:8458/api/system/config/replica-sets` to create ring and replica-set.

Verify ring configuration by executing the following command:

`echo "db.cache_config.find()" | mongo sessionmgr01:27721/sharding` <-- Change host name and port according to your deployment

- Configure ring in case creation of the replica-set fails:
  - Modify ring operation

```
---
- op: "modify-rings"
  setName: "set09"
```

Call PATCH API `http://<cluman-ip>:8458/api/system/config/replica-sets` to create ring and replica-set.

Note This operation re-creates ring if they are not configured before.

Active-Active Geo HA Support

As an Active-Active GR user you can use an API to configure OSGi commands for distributing traffic across different databases depending upon site-name or host-name.

For the manual steps to configure Active/Active Geo HA, refer to *CPS Geographic Redundancy Guide*.

By default, Geo HA feature is not installed and is not enabled. To install and enable the Geo HA, perform the following steps:

Step 1 Add *isGeoHAEnabled*, *geoHaSessionLookupType*, *enableReloadDict*, *geoSiteName*, *siteId*, and *remoteSiteId* lines in YAML file to install and enable Geo HA feature:

```
---
policyServerConfig:
  geoSiteName: "SITE1"
  siteId: "SITE1"
  remoteSiteId: "SITE2"
  heartBeatMonitorThreadSleepMS: "500"
```

```
mongodbupdaterConnectTimeoutMS: "1000"
mongodbupdaterSocketTimeoutMS: "1000"
dbConnectTimeout: "1200"
threadMaxWaitTime: "1200"
dbSocketTimeout: "600"
geoHaSessionLookupType: "realm"
isGeoHaEnabled: "true"
enableReloadDict: "true"
remoteGeoSiteName: "SITE2"
deploymentType: "GR"
sessionLocalGeoSiteTag: "SITE1"
```

- *isGeoHAEnabled* as `true` installs and enables the Geo HA feature.
- *geoHaSessionLookupType* as `realm` or `host` configures the lookup type.
- *enableReloadDict* is used to enable dictionary reload flag (Only for GR).
- *geoSiteName*, *siteId* and *remoteSiteId* are used to configure site information.

To verify whether Geo HA feature has been enabled or not, execute the following command:

```
list_installed_features.sh | grep geo
```

Output should be: `com.broadhop.policy.geoha.feature=XXXX`

```
grep geoha /etc/broadhop/pcrf/features
```

Output should be: `com.broadhop.policy.geoha.feature`

Step 2 Call PATCH API to load the updated configuration:

```
curl -i -X PATCH http://<clumanIP>:8458/api/system/config/application-config -H "Content-Type:
application/yaml" --data-binary @<yaml file name>
```

Step 3 Configure geo lookup information: geoLookupConfig changes can be done during new installation or at a later time.

Important Currently, deleting of lookup keys is not supported.

```
grConfig:
  geoLookupConfig:
    - siteId: "SITE1"
      lookupKey:
        - "site-gx-client1.com"
        - "site-gx-client2.com"
```

where,

- *siteId* is the ID of the site for which lookup keys need to be generated.
- *lookupKey* can be `realm` or `host`. This should have same value as configured for *geoHaSessionLookupType* in Step Step 1, on page 118.

a) In case, you update lookup key configuration, you can call the PATCH API:

```
curl -i -X PATCH http://<clumanIP>:8458/api/system/config/application-config -H "Content-Type:
application/yaml" --data-binary @<yaml file name>
```

b) To verify site lookup, use the following OSGi commands:

```
nc qns01 9091
```

```
listsitelookup <SITE-ID>
```

Step 4 Add replica-set to add each primary (active) site with its secondary (remote) site ID: For more information, refer to /api/system/config/replica-sets, on page 105.

After adding replica-sets, update using PATCH API: `curl -i -X PATCH http://<clumanIP>:8458/api/system/config/replica-sets -H "Content-Type: application/yaml" --data-binary @<yaml file name>`

Step 5 Add shards for each site: For more information, refer to /api/system/config/replica-sets, on page 105.

After adding shards, update using PATCH API: `curl -i -X PATCH http://<clumanIP>:8458/api/system/config/replica-sets -H "Content-Type: application/yaml" --data-binary @<yaml file name>`