Cisco Enterprise NFVIS User Guide

First Published:

Last Modified:

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

Text Part Number:

CONTENTS

Preface

Preface vii

Audience vii

Related Documentation vii

Obtaining Documentation and Submitting a Service Request vii

CHAPTER 1

About Cisco Enterprise NFVIS 1

Supported Hardware Platforms 1

Key Tasks You can Perform Using Cisco Enterprise NFVIS 2

CHAPTER 2

Installing Cisco Enterprise NFVIS Using the KVM Console 3

Installation Prerequisites 3

Installing Cisco Enterprise NFVIS on the Cisco UCS C220 M4 Rack Server 4

Logging Into the CIMC GUI 4

Configuring a Hardware RAID Controller (Optional) 4

Activating a Virtual Device 5

Mapping the Cisco Enterprise NFVIS Image 6

Booting the Device 6

Installing Cisco Enterprise NFVIS on Cisco UCS E-Series Servers 7

Sample Configuration on the Cisco ISR Router to Bring Up a Cisco UCS E Server 8

Performing Initial System Configuration 10

Default System Configuration on the Cisco UCS C220 M4 Server 11

Default System Configuration on the Cisco UCS E-Series Servers 12

Set up Initial Configuration Using the KVM Console 13

Set up Initial Configuration Using Cisco Enterprise NFV Portal 13

Cisco Network Plug-n-Play Support 14

CHAPTER 3

VM Life Cycle Management 17

Workflow of VM Life Cycle Management 18

CHAPTER 4 VM Deployment Scenarios 19

Registering VM Images 19

Single Image Deployment 20

Steps for Deploying a VM 20

Service Chaining with two VM Images 22

Steps for Service Chaining with Two VM Images 23

Service Chaining of Multiple VMs with Windows or Linux Servers 23

Steps for Service Chaining of Multiple VMs with Windows or Linux Servers 23

VM Image Packaging 24

Generating a VM OVA Package 24

Package Manifest File 25

VM Image Properties File 26

Profile Properties File 28

Bootstrap Configuration File 29

Example for VM Packaging 29

Example: Package.mf 29

Example: Image Properties 30

Example: Bootstrap Configuration File 31

CHAPTER 5 REST APIs for Cisco Enterprise NFVIS 33

REST API Credentials 33

API Request Methods 34

VM Lifecycle Management APIs 34

VM Image Registration APIs 34

Example: Image Registration API 35

Example: Get Image Configuration API 36

Example: Get Image Status API 36

Example: Image Deletion API 37

Bridge Creation APIs 37

Example: Bridge Creation API 38

Example: Get Bridge Configuration API 39

Example: Bridge Deletion API 40

Network Creation APIs 40

Example: Network Creation API 41
Example: Get Network Configuration API 42
Example: Network Deletion API 43
Custom Flavor Creation APIs 43
Example: Flavor Creation API 44
Example: Get Flavor Configuration API 45
Example: Flavor Status API 46
Example: Flavor Deletion API 47
VM Deployment APIs 47
Example: VM Deployment API for Cisco CSR 1000V 51
Example: VM Deployment Deletion API 53
Example: VM Deployment API for Cisco vWAAS 53
Example: VM Deployment API for Cisco vWLC 54
Adding or Deleting a vNIC Using the VM Deployment API 55
Changing the Flavor Using the VM Deployment API 56
Service Chaining of VMs 57
VM Operations APIs 57
Example: Start VM API 58
Example: Stop VM API 59
Example: Restart VM API 59
VM Operational Status APIs 60
System and IP Configuration APIs 60
System Details APIs 60
Example: System Details API 62
Example: Get System Details API 63
Default IP Configuration APIs 63
System Utilization APIs 64
Platform Details API 64
Port Details APIs 65
Host Port Statistics APIs 65
Example: Host Port Statistics API 66
Host Port Statistics Table APIs 68
Example: Host Port Statistics Table API (Single Port) 68
Host CPU State APIs 69
Example: Host CPU State API 70

Host CPU Statistics Table API 71
Example: Host CPU Statistics Table API 72
Host Memory Statistics APIs 75
Example: Host Memory Statistics API 75
Host Memory Statistics Table APIs 76
Example: Host Memory Statistics Table APIs 77
Host Disk Statistics APIs 78
Example: Host Disk Statistics API 78
Log API 80
Example: Log API 81
Host Reboot API 81
PnP Server APIs 82
Using Self-Signed and CA-Signed Certificates 82
Certificate Creation APIs 83
Example for the Signing Request API 84
Example for the Install Certificate API 85
Example for the Use Certificate API 86
User Management APIs 86
Example: Get Users API 87
Example: Get Specific User 87
Example: Add User API 88
Example: Modify User API 88
Example: Delete User API 88

Preface

This user guide provides information about how to install and configure Cisco Enterprise Network Function Virtualization Infrastructure Software (Cisco Enterprise NFVIS) on a supported Cisco hardware device. The guide also provides details on virtual machine deployments and life cycle management using Representation State Transfer (REST) application programming interface.

- Audience, page vii
- Related Documentation, page vii
- Obtaining Documentation and Submitting a Service Request, page vii

Audience

This guide is intended for network administrators and operators who are familiar with Cisco Unified Computing System (UCS) servers, Cisco Integrated Services Routers (ISRs), and basic Linux installation requirements.

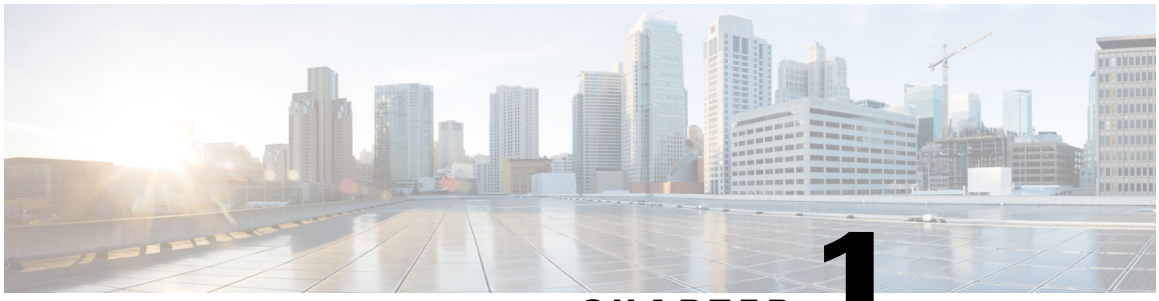
Related Documentation

- Getting Started Guide
- Cisco UCS C220 M4 Server Installation and Service Guide
- Configuration Guide for Cisco Network Plug and Play on Cisco APIC-EM

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.



CHAPTER 1

About Cisco Enterprise NFVIS

Cisco Enterprise Network Function Virtualization Infrastructure Software (Cisco Enterprise NFVIS) is a Linux-based infrastructure software designed to help service providers and enterprises dynamically deploy virtualized network functions (hereinafter called VMs), such as firewall, virtual router, and WAN acceleration, on a supported Cisco device. There is no need to add hardware for every network function, and you can use automated, centralized provisioning, and management to eliminate costly truck rolls.

This solution enables service providers to offer services on demand by provisioning and chaining VMs. For example, a customer in need of a firewall functionality can use the service provider's portal to choose among a list of VMs, such as Cisco Adaptive Security Appliance (ASA), which can be deployed dynamically on the hardware platform across all locations.

Benefits of Cisco Enterprise NFVIS

- Cost effective solution to consolidate multiple physical network appliances into a single server running multiple virtual network functions.
- Flexibility in deploying services quickly and in a timely manner.
- Cloud based VM life cycle management and provisioning.
- In-box life cycle management software to deploy and chain VMs dynamically on the platform.
- Programmable APIs.

- Supported Hardware Platforms, page 1
- Key Tasks You can Perform Using Cisco Enterprise NFVIS, page 2

Supported Hardware Platforms

Depending on your requirement, you can install Cisco Enterprise NFVIS on the Cisco UCS C220 M4 Rack Server or a few flavors of Cisco UCS E-Series Server Modules housed within the Cisco Integrated Services Routers (ISR G2). The following is the list of supported hardware platforms:

- Cisco UCS C220 M4 Rack Server
- Cisco ISR4451-X with UCS-E180D-M2/K9
- Cisco ISR4331 with UCS-E140S-M2/K9

- Cisco ISR4351 with UCS-E160D-M2/K9
- Cisco ISR2911 with UCS-E140S-M2/K9

Cisco UCS C220 M4 Rack Server

The Cisco UCS C220 M4 Rack Server is the high-density, general-purpose enterprise infrastructure and application server that delivers world-record performance for a wide range of enterprise workloads, including virtualization, collaboration, and bare-metal applications.

Cisco UCS E-Series Server Modules

The Cisco UCS E-Series Servers (E-Series Servers) are the next generation of Cisco UCS Express servers. E-Series Servers are a family of size, weight, and power efficient blade servers that are housed within the Generation 2 Cisco Integrated Services Routers (ISR G2) and the Cisco 4400 and Cisco 4300 Series Integrated Services Router. These servers provide a general-purpose compute platform for branch office applications deployed either as bare metal on operating systems, such as Microsoft Windows or Linux; or as virtual machines on hypervisors.

Supported VMs

Currently, the following Cisco supplied VMs and third party VMs are supported:

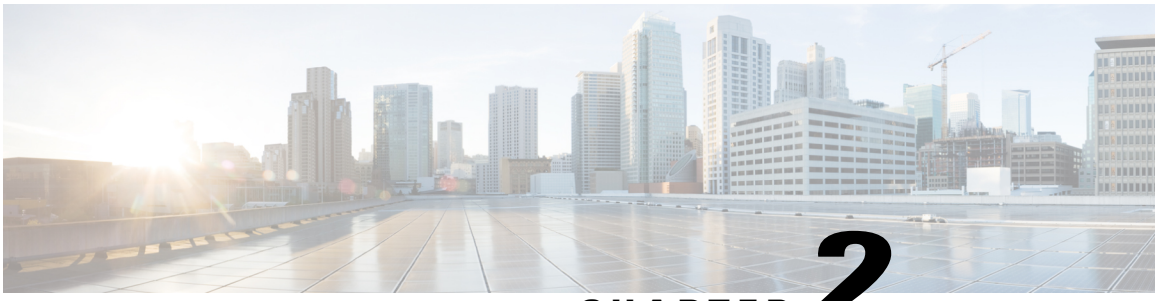
- Cisco CSR 1000V Series Cloud Services Router
- Cisco Virtual Wide Area Application Services (vWAAS)
- Cisco Adaptive Security Virtual Appliance (ASAv)
- Virtual Wireless LAN Controller (vWLC)
- Linux Server VM
- Windows Server 2012 VM

Key Tasks You can Perform Using Cisco Enterprise NFVIS

The following is a high-level list of tasks you can perform using Cisco Enterprise NFVIS:

- Perform VM image registration and deployment
- Create new networks and bridges, and assign ports to bridges
- Create custom flavors
- Perform service chaining of VMs
- Perform VM operations
- Verify system information including CPU, port, memory, and disk statistics

The APIs for performing these tasks are explained in chapter 5, REST APIs for Cisco Enterprise NFVIS.



CHAPTER 2

Installing Cisco Enterprise NFVIS Using the KVM Console

This chapter describes how to install Cisco Enterprise Network Function Virtualization Infrastructure Software (Cisco Enterprise NFVIS) on a common purpose virtualization ready Cisco x86 hardware platform.

- Installation Prerequisites , page 3
- Installing Cisco Enterprise NFVIS on the Cisco UCS C220 M4 Rack Server, page 4
- Installing Cisco Enterprise NFVIS on Cisco UCS E-Series Servers, page 7
- Performing Initial System Configuration, page 10

Installation Prerequisites

- Ensure that you have the IP address configured for Cisco Integrated Management Controller (CIMC) GUI, as well as a login account with administrative privileges.
- Ensure that you have the installation media for the Cisco Enterprise NFVIS, either on a DVD or as an ISO image.
- The IP address of the system (required for remote access) is available.
- Ensure that hyper-threading is disabled in the BIOS.

Note The installation steps are slightly different between the Cisco UCS C220 M4 Rack Server and the Cisco UCS E-Series servers housed in Cisco ISR routers. See the following sections for details:

Installing Cisco Enterprise NFVIS on the Cisco UCS C220 M4 Rack Server, on page 4

Installing Cisco Enterprise NFVIS on Cisco UCS E-Series Servers, on page 7

Assumptions

- The user is familiar with the supported hardware device, CIMC, Cisco Network Plug and Play, and Cisco Application Policy Infrastructure Controller Enterprise Module (APIC-EM).

- The initial setup of the hardware device is complete, and the device is ready for loading the Cisco Enterprise NFVIS.
- The user is familiar with general Linux installation.

For more details on the supported hardware devices, see respective documentation available on Cisco.com.

Installing Cisco Enterprise NFVIS on the Cisco UCS C220 M4 Rack Server

This section provides information about a series of tasks you need to perform in order to install Cisco Enterprise NFVIS on a Cisco UCS C220 M4 Rack Server.

Logging Into the CIMC GUI

Before You Begin

- Make sure that you have configured the IP address to access CIMC.
- If not installed, install Adobe Flash Player 10 or later on your local machine.

For details on how to configure an IP address for CIMC, see the Set up CIMC for UCS C-Series Server guide on cisco.com.

Step 1 In your web browser, enter the IP address that you configured to access CIMC during initial setup.

Step 2 If a security dialog box displays, do the following:

a) **Optional**: Select the check box to accept all content from Cisco.

b) Click **Yes** to accept the certificate and continue.

Step 3 In the log in window, enter your username and password.
When logging in for the first time to an unconfigured system, use **admin** as the username and **password** as the password.

Step 4 Click **Log In**.
The **Change Password** dialog box only appears the first time you log into CIMC.

Step 5 Change the password as appropriate and save.
The CIMC home page is displayed.

Configuring a Hardware RAID Controller (Optional)

You can choose to store the server data files on local Redundant Array of Inexpensive Disks (RAID).

Note This task is applicable only to the devices that support hardware RAID controllers. RAID controllers are of two types: hardware and software. You cannot install Cisco Enterprise NFVIS with software RAID controllers.

Step 1 After logging into CIMC, ensure that the status of each of the physical drives that you want to configure as RAID is **unconfigured good**. To view the status, click the **Storage** tab, and then click the **Physical Drive Info** tab.
If the status of the physical drive is **unconfigured good** , proceed to **Step 3**. If the physical drive is already configured, and the status is not **unconfigured good** , perform **Step 2**.

Step 2 Open the **Virtual Drive Info** tab, and click **Delete Virtual Drive** to delete the drive, and change the status to **unconfigured good**.

Step 3 In the **Navigation** pane, click the **Storage** tab.

Step 4 On the **Storage** tab, select the Raid controller of your choice.

Step 5 On the **Controller Info** tab, click the **Create Virtual Drive from Unused Physical Drives** option.
The **Create Virtual Drive from Unused Physical Drives** window appears.

Step 6 From the **RAID Level** drop-down list, select **1**.

Step 7 From the **Physical Drives** table, select two drives of your choice, and add them to the **Drive Groups** table.
A drive group name is automatically created and listed in the **Drive Groups** table. A virtual drive name is automatically created in the **Virtual Drive Name** field based on the RAID level and the physical drives selected. You can change the name, if required.

Step 8 Click **Create Virtual Drive**.
The virtual drive is created. You can view the virtual drive from the **Virtual Drive Info** tab.

Step 9 From the **Virtual Drive Info** tab, click **Initialize**.

Step 10 Select **Fast Initialize** as the initialize type, and click **Initialize VD**.

Step 11 From the **Virtual Drive Info** tab, click **Set as Boot Drive** to set the virtual drive as the boot drive.
The RAID controller configuration is complete now. For more details on CIMC screens and dialog boxes, open the CIMC online help.

Activating a Virtual Device

You will have to launch the KVM Console to activate virtual devices.

Before You Begin

Ensure that you have the Java 1.6.0_14 or a higher version installed on your system.

Step 1 Download the Cisco Enterprise NFVIS ISO image file from a prescribed location to your local system.

Step 2 From the **Server** tab, click **Launch KVM Console**.

Note A JNLP file will be downloaded to your system with unwanted extension details. You must rename the file with the *.jnlp* extension immediately after it is downloaded to avoid the session timeout. For example, you can rename the file as *viewer.jnlp*.

Step 3 Open the renamed *.jnlp* file. When it prompts you to download Cisco Virtual KVM Console, click **Yes**. Ignore all security warnings and continue with the launch.
The KVM Console is displayed.

Step 4 From the **Virtual Media** menu on the KVM Console, select **Activate Virtual Devices**.
If prompted with an unencrypted virtual media session message, select **Accept this session** and click **Apply**. The virtual devices are activated now.

Mapping the Cisco Enterprise NFVIS Image

Step 1 From the **Virtual Media** menu on the KVM Console, select **Map CD/DVD...**.

Step 2 Browse for the installation file (ISO) on your system, and select it .

Step 3 Click **Map Device**.
The ISO image file is now mapped to the CD/DVD.

Booting the Device

After mapping the image, you will have to boot the device using the KVM console. First, you have to boot from the DVD source, and then reboot from the hard drive.

Step 1 From the **Server** tab, select **BIOS**.

Step 2 From the **BIOS Actions** area, select **Configure Boot Order**.
If any boot order is listed in the **Configure Boot Order** dialog box, proceed to **Step 3** to delete the boot order. If no boot order is listed, skip **Step 3**, and proceed to **Step 4**.

Step 3 Select the boot order listed, and click **Delete**.

Step 4 Select **Add Virtual Media** from the **Add Boot Device** section in the **Configure Boot Order** dialog box.
The **Add Virtual Media** dialog box appears.

Step 5 In the **Name** field, enter the boot device name (for example, KVM-DVD).

Step 6 In the **Sub Type** field, select **KVM Mapped DVD**.

Step 7 The values are automatically displayed in the **State** field and the **Order** filed.

Step 8 Click **Add Device**.
The boot order is now configured.

Step 9 From the **Power** menu on the KVM Console, select **Power Cycle System (cold boot)**.

The KVM console will automatically install Cisco Enterprise NFVIS. The entire installation might take 30 minutes to complete.

Note Do not hit or enter any key after the installation process starts. Hitting a key can stop the installation process. If the installation is not in progress, and the system is rebooted, it means the boot order was not set properly, or the ISO image was not mapped properly.

Step 10 When the installation is complete, change the boot order to hard drive by deleting **KVM Mapped DVD** from the KVM console.

Step 11 Click **Reboot**.
You have now successfully installed Cisco Enterprise NFVIS. You can log into the system when the command prompt changes from "local host" to "nfvis." Use **admin** as the login name and **admin** as the default password.

Step 12 You can verify the installation using the System API or by viewing the system information from the Cisco Enterprise NFV portal.

Installing Cisco Enterprise NFVIS on Cisco UCS E-Series Servers

Before You Begin

- Configure the UCS E interface on the Cisco ISR router.
- Configure the Gigabit Ethernet interface on the Cisco ISR router.
- Ensure that you have the IP address configured for CIMC GUI, as well as a login account with administrative privileges.

For more details on how to perform the basic configuration on the Cisco ISR routers, see the following topics and guides:

- Sample Configuration on the Cisco ISR Router to Bring Up a Cisco UCS E Server, on page 8
- Getting Started Guide for Cisco UCS E-Series Servers, Release 1.0(2) Installed in the Cisco ISR 4451-X
- Getting Started Guide for Cisco UCS E-Series Servers, Release 1.0

Step 1 Log into the CIMC GUI.
For details, see Logging Into the CIMC GUI , on page 4

Step 2 Configure a Raid controller, if required.
For details, see Configuring a Hardware RAID Controller (Optional), on page 4

Step 3 From the **Server** tab, click **Launch KVM Console**.
The KVM Console opens in a separate window.

Step 4 From the KVM console, click the **Virtual Media** tab.

Step 5 In the **Virtual Media** tab, map the virtual media using either of the following methods:

a) Select the **Mapped**check box for the CD/DVD drive containing the operating system.

b) Click **Add Image**, browse and select the Cisco Enterprise NFVIS ISO image , click **Open** to mount the image, and then select the **Mapped** check box for the mounted image.

You must keep the Virtual Media tab open during the installation process. Closing the tab unmaps all virtual media.

Step 6 From the **Server** tab, select **BIOS**.

Step 7 From the **BIOS Actions** area, select **Configure Boot Order**.
The **Configure Boot Order** dialog box appears.

Step 8 From the **Device Types** area, select **CD/DVD Linux Virtual CD/DVD**, and then click **Add**.

Step 9 Select **HDD PCI RAID Adapter**, and then click **Add**.

Step 10 Set the boot order sequence using the **Up** and **Down** options. The **CD/DVD Linux Virtual CD/DVD** boot order option must be the first choice.

Step 11 Click **Apply** to complete the boot order setup.

Step 12 Reboot the server by selecting the **Power Off Server** option from the **Server Summary** page in CIMC.

Step 13 After the server is down, select the **Power On Server** option in CIMC.
When the server reboots, the KVM console will automatically install Cisco Enterprise NFVIS from the virtual CD/DVD drive. The entire installation might take 30 minutes to complete.

Note Do not hit or enter any key after the installation process starts. Hitting a key can stop the installation process. . If the installation is not in progress, and the system is rebooted, it means the boot order was not set properly, or the ISO image was not mapped properly.

Step 14 When the installation is complete, change the boot order to hard drive by deleting **KVM Mapped DVD** from the KVM console.

Step 15 Click **Reboot**.
You have now successfully installed Cisco Enterprise NFVIS. You can log into the system when the command prompt changes from "local host" to "nfvis." Use **admin** as the login name and **admin** as the default password.

Step 16 You can verify the installation using the System API or by viewing the system information from the Cisco Enterprise NFV portal.

Sample Configuration on the Cisco ISR Router to Bring Up a Cisco UCS E Server

The following sample configuration shows the basic configuration performed on the Cisco ISR 4451 router with DHCP enabled.

```
Last configuration change at 02:36:37 UTC Thu Feb 18 2016
!
version 15.5
service timestamps debug datetime msec
service timestamps log datetime msec
no platform punt-keepalive disable-kernel-core
!
hostname NFVIS-ISR4451
!
boot-start-marker
boot system bootflash:isr4300-universalk9.03.16.01a.S.155-3.S1a-ext.SPA.bin
boot-end-marker
!
!
vrf definition Mgmt-intf
```

```
 !
 address-family ipv4
 exit-address-family
 !
 address-family ipv6
 exit-address-family
!
!
no aaa new-model
!
!
!
ip domain name cisco.com
!
!
!
subscriber templating
!
multilink bundle-name authenticated
!
!
!
license udi pid ISR4331/K9 sn FDO192207MN
!
!
ucse subslot 1/0
 imc access-port shared-lom console
 imc ip address 172.19.183.172 255.255.255.0 default-gateway 172.19.183.1
!
spanning-tree extend system-id
!
!
redundancy
 mode none
!
!
!
vlan internal allocation policy ascending
!
!
!
interface GigabitEthernet0/0/0
 ip address 172.19.183.171 255.255.255.0
 media-type rj45
 negotiation auto
!
interface GigabitEthernet0/0/1
 no ip address
 shutdown
 negotiation auto
!
interface GigabitEthernet0/0/2
 no ip address
 shutdown
 negotiation auto
!
interface ucse1/0/0
 ip unnumbered GigabitEthernet0/0/0
 negotiation auto
 switchport mode trunk
 no mop enabled
 no mop sysid
!
interface ucse1/0/1
 no ip address
 no negotiation auto
 switchport mode trunk
 no mop enabled
 no mop sysid
!
interface GigabitEthernet0
 vrf forwarding Mgmt-intf
```

```
 no ip address
 shutdown
 negotiation auto
!
interface Vlan1
 no ip address
 shutdown
!
ip default-gateway 172.19.183.1
ip forward-protocol nd
no ip http server
no ip http secure-server
ip tftp source-interface GigabitEthernet0
ip route 0.0.0.0 0.0.0.0 172.19.183.1
ip route 172.19.183.172 255.255.255.255 ucse1/0/0
ip ssh version 2
!
!
!

control-plane
!
!
line con 0
 stopbits 1
line aux 0
 stopbits 1
line vty 0 4
 password lab
 login local
 transport input all
 transport output all
!
!
end
```

Performing Initial System Configuration

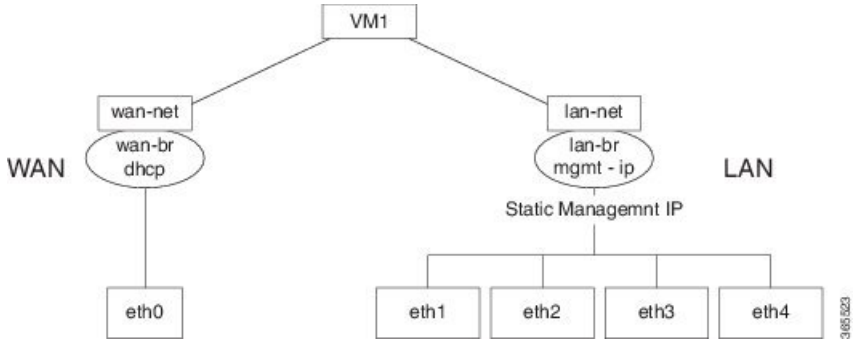
You can set up initial configuration using the KVM console or Cisco Enterprise NFV portal.

- Set up Initial Configuration Using the KVM Console, on page 13
- Set up Initial Configuration Using Cisco Enterprise NFV Portal, on page 13

Default System Configuration on the Cisco UCS C220 M4 Server

Configuring the networks in Cisco Enterprise NFVIS allows inbound and outbound traffic and VMs to be service chained. The following diagram illustrates the default network configuration:

Figure 1: Default Network Configuration of Cisco NFVIS with Cisco UCS C220 M4



The following networks and bridges are created by default and cannot be deleted. You can configure more as required.

- A LAN network (lan-net) and a LAN bridge (lan-br)—The default static management IP address (192.168.1.1) for the NFVIS host is configured on the LAN bridge. All other ports for inbound and outbound traffic are associated with the LAN bridge. Any LAN port can be used to access the default static IP address. By default, the hostname is set to "nfvis."
- A WAN network (wan-net) and a WAN bridge (wan-br)—This is created with the "eth0" port, and is configured to enable the DHCP connection.
- An internal management network (int-mgmt-net) and a bridge (int-mgmt-br)—This network is internally used for system monitoring.

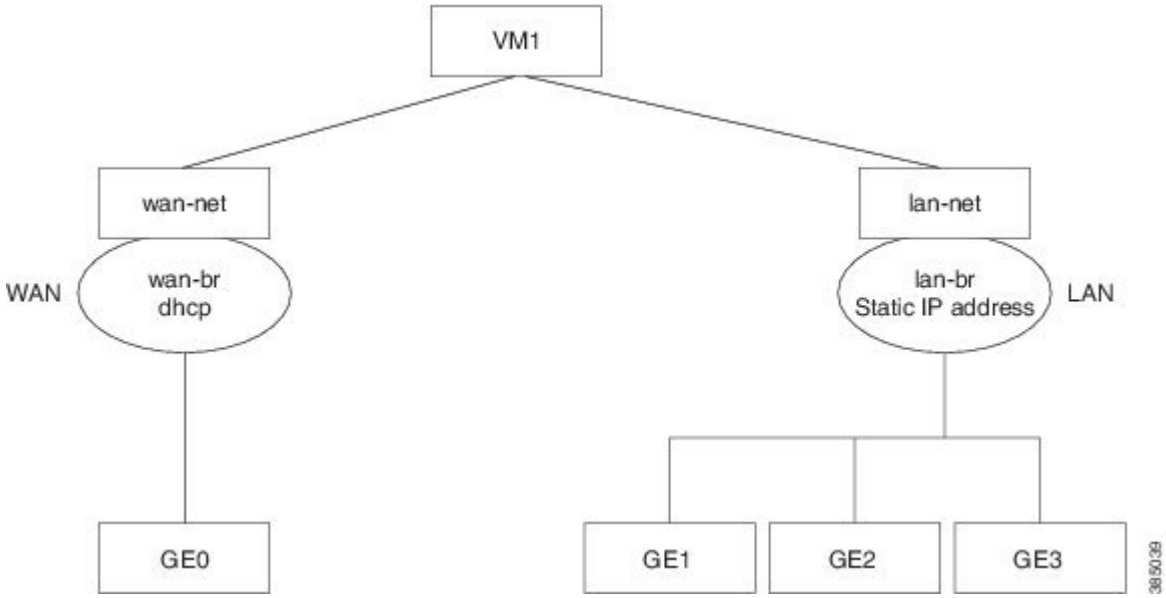
By default, the first port on the device is associated with the WAN bridge. All the other ports on the device are associated with the LAN bridge.

For more details on the initial setup, see the Installing the Server chapter in the Cisco UCS C220 M4 Server Installation and Service Guide.

Default System Configuration on the Cisco UCS E-Series Servers

The diagram below illustrates the default networking topology for Cisco NFVIS with the Cisco UCS-E Server.

Figure 2:



The following networks and bridges are created by default and cannot be deleted. You can configure more as required.

- A LAN network (lan-net) and a LAN bridge (lan-br)—The default static management IP address (192.168.1.1) for the NFVIS host is configured on the LAN bridge. All other ports for inbound and outbound traffic are associated with the LAN bridge. By default, the hostname is set to "nfvis."
- A WAN network (wan-net) and a WAN bridge (wan-br)— The physical WAN ports are on the Cisco ISR module. They are not externally available on the Cisco UCS E server. The WAN traffic comes from the ISR WAN ports, and goes through the backplane to the Cisco UCS-E server. The backplane has one internal WAN interface (GE0) to establish connection with the Cisco UCS-E server. By default, the "GE0" interface is enabled for the DHCP connection.
- An internal management network (int-mgmt-net) and a bridge (int-mgmt-br)—This network is internally used for system monitoring.

For more details on the initial setup, see the Getting Started Guide for Cisco UCS E-Series Servers and the Cisco UCS E-Series Network Compute Engine.

Set up Initial Configuration Using the KVM Console

You must set the default gateway and management IP address for enabling remote access to the system using the Cisco Enterprise NFV portal.

Step 1 On the KVM console, enter **admin** as the login name and password when the command prompt changes to "nfvis" after installing Cisco Enterprise NFVIS.

Step 2 Enter **Configure**.

Step 3 Enter the following command to set the default gateway:

```
set system settings default-gw ip-address
```

Step 4 Enter the following command to configure the management IP address and net mask:

```
set system settings mgmt-ip ip-address ip-address netmask ip-address
```

Step 5 Enter the following command to disable DHCP:

```
set system settings wan dhcp disable
```

Note DHCP might supersede the default gateway, if it is not disabled.

Step 6 Enter **commit** to save the configuration, and exit the configuration mode.

Step 7 Enter the following command to verify the initial configuration.

```
show system settings-native
```

Now, you can use the management IP address to access the Cisco Enterprise NFV portal. In your browser, type "https://<ip-address>" to open the portal.

Set up Initial Configuration Using Cisco Enterprise NFV Portal

Step 1 Using an Ethernet cable, connect your laptop to the first LAN port of the device. For example, use "eth1," if the device is Cisco UCS C220 M4.

Note Ensure that Wi-Fi is disabled on your laptop.

Step 2 Perform the following steps on a Mac system:

a) Go to **Preferences** > **Network** >**Thunderbolt** .

b) Choose **Manually** from the **Configure IPv4** selection box.

c) In the **IP Address** field, enter an IP address. For example, 192.168.1.5.

d) In the **Subnet Mask** field, enter 255.255.255.0.

Step 3 In your browser, type **https://192.168.1.1** to open Cisco Enterprise NFV portal.

Step 4 Use **admin** as login name and password.

Step 5 From the sidebar, select **Settings**.

Step 6 Choose the **IP Settings** tab, and click **Edit** to modify the settings as required.

Step 7 Click **Save**.

Cisco Network Plug-n-Play Support

The Cisco Network Plug and Play (Cisco Network PnP) solution provides a simple, secure, unified, and integrated offering for enterprise network customers to ease new branch or campus device rollouts, or for provisioning updates to an existing network. The solution provides a unified approach to provision enterprise networks comprised of Cisco routers, switches, and wireless devices with a near zero touch deployment experience. This solution uses Cisco Application Policy Infrastructure Controller Enterprise Module (APIC-EM) to centrally manage remote device deployments.

When a device is powered on for the first time, the Cisco Network PnP agent discovery process, which is embedded in the device, wakes up in the absence of the startup configuration file, and automatically discovers the IP address of the Cisco Network PnP server located in the Cisco APIC-EM. The Cisco Network PnP agent uses DHCP or Domain Name System (DNS) lookup to auto discover the server.

Currently, you can use the Cisco Network Plug and Play client to:

- Auto discover the server
- Provide device information to the server
- Bulk provisioning of user credentials

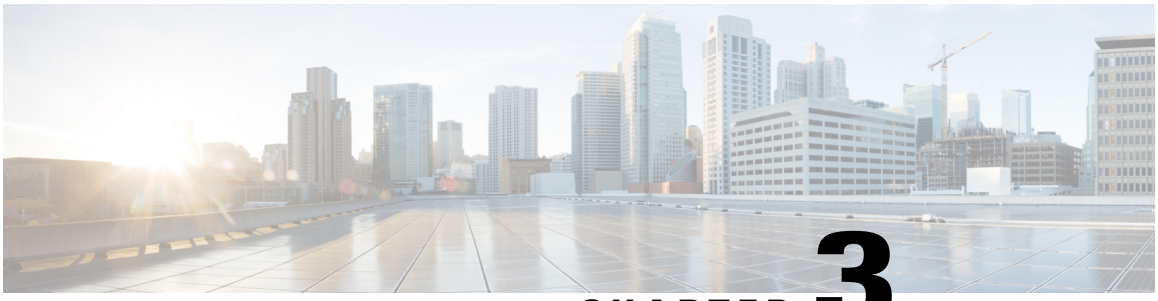
Bulk Provisioning of User Credentials

You can change the default user name and password of the devices using the Cisco Network PnP client. The Cisco Network PnP server sends the configuration file to Cisco Network PnP clients residing on multiple devices in the network, and the new configuration is automatically applied to all the devices.

Note For bulk provisioning of user credentials, ensure that you have the necessary configuration file uploaded to the Cisco APIC-EM. The following is a sample configuration file:

```
<aaa xmlns="http://tail-f.com/ns/aaa/1.1">
    <authentication>
      <users>
        <user>
          <name>admin</name>
          <password>user123</password>
        </user>
      </users>
    </authentication>
</aaa>
```

For more details on the Cisco Network PnP solution and how to upload a configuartion file, see the Configuration Guide for Cisco Network Plug and Play on Cisco APIC-EM.



CHAPTER 3

VM Life Cycle Management

VM life cycle management refers to the entire process of obtaining the VM images, registering and deploying the VMs, and getting them service chained as per your requirements. You can perform these tasks and more using a set of Representation State Transfer (REST) APIs or the Cisco Enterprise NFVIS portal.

VM Packaging Format

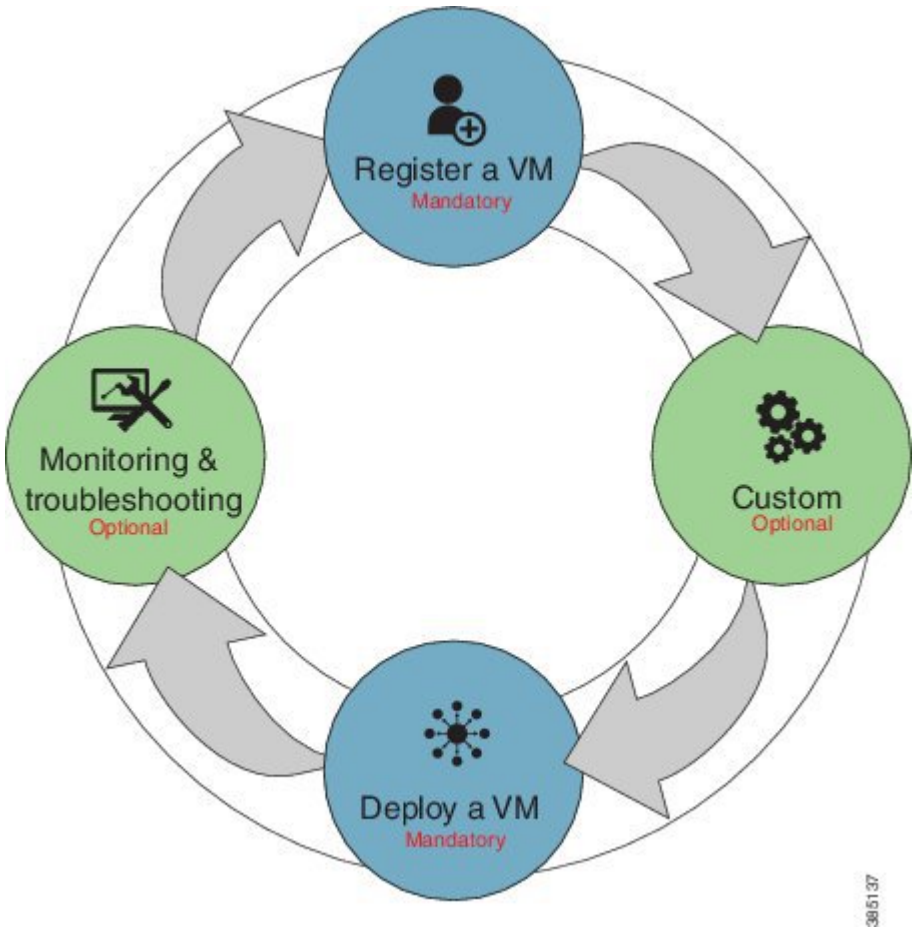
The VM source file must be available in the *.ova* or *.tar.gz* format. All Cisco supplied VMs are available in the prescribed format. Vendors are responsible for packaging all third party VMs in the prescribed format.

- Workflow of VM Life Cycle Management, page 18

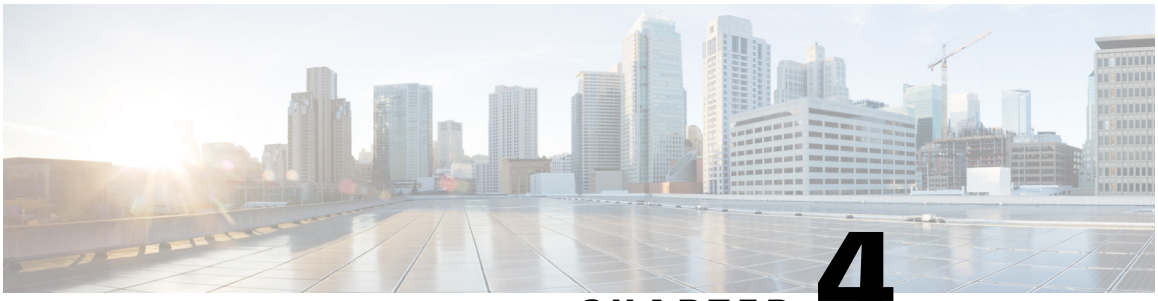
Workflow of VM Life Cycle Management

The following diagram depicts the basic workflow of the VM life cycle management using REST APIs:

Figure 3: VM Life Cycle Management



1. **Register a VM:** To register a VM, you must first download the relevant OVA file to your system. Once you have downloaded the file you can register the image. Registering the image is a one-time activity. Once an image is registered, you can perform multiple VM deployments using the registered image.
2. **Customize a VM:** After registering a VM, you can profile your VM by creating custom flavors. The flavor creation option lets you provide specific profiling details for the VM, such as the virtual CPU on which the VM will run, and the amount of virtual CPU memory the VM will consume.
3. **Deploy a VM:** A VM is configured and customized during deployment. You can configure a VM by providing values to the parameters or variables that are passed to the system during deployment. Depending on the VM that you are deploying, some parameters are mandatory and others optional.
4. **Monitor a VM:** There are a set of APIs and commands that allow you to monitor a VM. For example, you can start, stop or reboot a VM.



CHAPTER 4

VM Deployment Scenarios

This section provides details on the following deployment scenarios using REST APIs:

- Single Image Deployment
- Service Chaining with two VM Images
- Service Chaining of multiple VMs with Windows or Linux Servers

The following VM images are used to explain the deployment scenarios:

- Cisco CSR 1000V Series Cloud Services Router —csr1000v.03.16.01a
- Cisco Adaptive Security Virtual Appliance (ASAv)— asav941-203
- Linux server—ubuntu-14.04.3-server-amd64-disk1.ova

- Registering VM Images, page 19
- VM Image Packaging, page 24
- Example for VM Packaging, page 29

Registering VM Images

You will have to register all three VM images before deploying them.

Note Register all the VM images required for the VM deployment depending on the topology. A VM image registration is done only once per VM image. You can perform multiple VM deployments using the registered VM image.

To register a Cisco CSR 1000V image:

1. Copy the *csr1000v.ova* file to the Cisco NFVIS server.
2. Register the Cisco CSR 1000V image using the following API method:

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
```

```
Content-Type:application/vnd.yang.data+xml -X
POST https://NFVIS_IP/api/config/esc_datamodel/images -d
'<image><name>csr1000v.03.16.01a</name><src>file://filename_with_full-path-of
the-ova-file/csr1000v.ova</src></image>'
```

3 Verify the image status using the following API method:

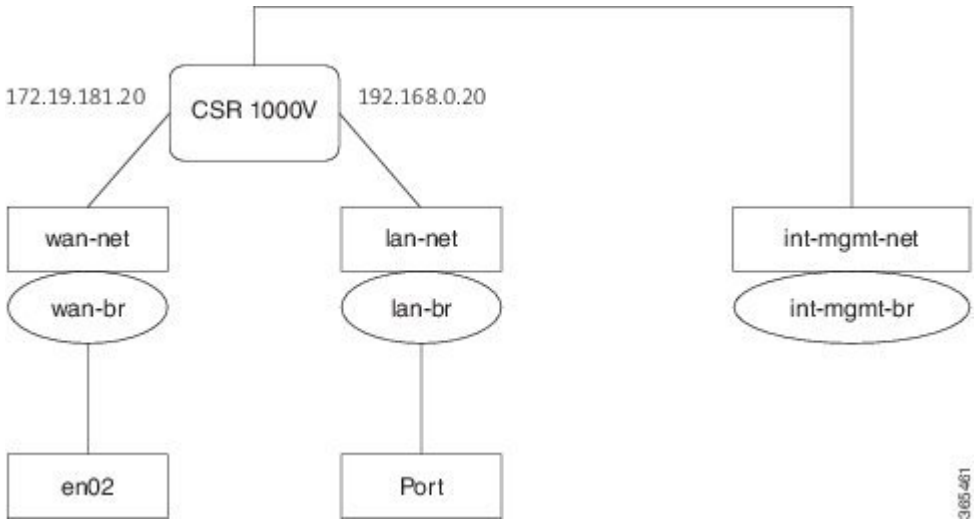
```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X
GET https://NFVIS_IP/api/operational/esc_datamodel/opdata/images/image/ csr1000v.03.16.01a
 ?deep
```

4 Now, repeat Steps 1 to 3 to register the Cisco ASAv and Windows Server images. Ensure that you provide the exact image name and source file location when running the API commands.

Single Image Deployment

In this example, a Cisco CSR 1000V image with three network interface cards is deployed. The following diagram illustrates the deployment topology:

Figure 4: Single VM Deployment



Steps for Deploying a VM

By default a LAN network (lan-net), WAN network (wan-net), internal management network (int-mgmt-net), LAN bridge (lan-br), and WAN bridge (wan-br) are created in the system. Also, the WAN bridge is attached to the eno2 port. The IP address of the int-mgmt-net interface is assigned automatically for each VM during deployment. However, you will have to attach one port to the LAN bridge using APIs.

When deploying a VM, you can attach SR-IOV networks to the VM.

To deploy a Cisco CSR 1000V image:

1 Assign a port to the LAN bridge.

```
admin@nfvis> curl -k -v -u admin:admin -H content-type:application/vnd.yang.data+xml -X
 PUT https:// NFVIS_IP /api/config/bridges/bridge/lan-br --data
```

```
'<bridge><name>lan-br</name><port><name>enp4s0f0</name></port></bridge>'
```

2 Verify that all bridges are configured.

```
admin@nfvis> curl -k -v -u admin:admin -H content-type:application/vnd.yang.data+xml -X
GET https:// NFVIS_IP /api/config/bridges?deep
```

3 Verify that all networks are configured.

```
admin@nfvis> curl -k -v -u
    admin:admin -H content-type:application/vnd.yang.data+xml -X GET https:// NFVIS_IP
/api/config/networks?deep
```

4 Deploy the Cisco CSR 1000V VM.

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X
POST https:// NFVIS_IP /api/config/esc_datamodel/tenants/tenant/admin/deployments --data
<deployment>
    <name>CSR</name>
    <vm_group>
        <name>CSR</name>
        <image>csr1000v.03.16.01a</image>
        <bootup_time>600</bootup_time>
        <recovery_wait_time>0</recovery_wait_time>
        <recovery_policy>
            <action_on_recovery>REBOOT_ONLY</action_on_recovery>
        </recovery_policy>
        <interfaces>
            <interface>
                <nicid>0</nicid>
                <network>int-mgmt-net</network>
                <port_forwarding>
                    <port>
                        <type>ssh</type>
                        <protocol>tcp</protocol>
                        <vnf_port>22</vnf_port>
                        <external_port_range>
                            <start>20022</start><end>20022</end>
                        </external_port_range>
                    </port>
                </port_forwarding>
            </interface>
            <interface>
                <nicid>1</nicid>
                <network>lan-net</network>
                <ip_address>192.168.0.20</ip_address>
            </interface>
            <interface>
                <nicid>2</nicid>
                <network>wan-net</network>
                <ip_address>172.19.181.20</ip_address>
            </interface>
        </interfaces>
        <scaling>
            <min_active>1</min_active>
            <max_active>1</max_active>
        </scaling>
        <kpi_data>
            <kpi>
                <event_name>VM_ALIVE</event_name>
                <metric_value>1</metric_value>
                <metric_cond>GT</metric_cond>
                <metric_type>UINT32</metric_type>
                <metric_collector>
                    <type>ICMPPing</type>
                    <nicid>0</nicid>
                    <poll_frequency>3</poll_frequency>
```

```
                    <polling_unit>seconds</polling_unit>
                    <continuous_alarm>false
                    </continuous_alarm>
                </metric_collector>
            </kpi>
        </kpi_data>
        <rules>
            <admin_rules>
                <rule>
                    <event_name>VM_ALIVE</event_name>
                    <action>ALWAYS log</action>
                    <action>TRUE servicebooted.sh</action>
                    <action>FALSE recover autohealing</action>
                </rule>
            </admin_rules>
        </rules>
        <config_data>
            <configuration>
                <dst>bootstrap_config</dst>
                <variable>
                    <name>TECH_PACKAGE</name>
                    <val>security</val>
                </variable>
            </configuration>
        </config_data>
    </vm_group>
</deployment>
```

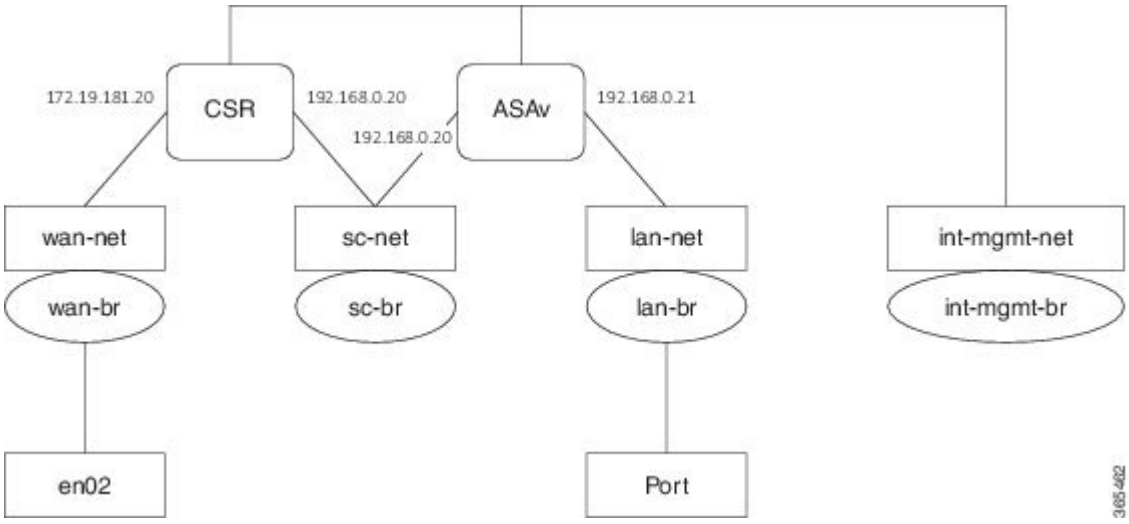
5 Verify the deployment status.

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X GET
https://NFVIS_IP/api/operational/esc_datamodel/opdata/tenants/tenant/admin/deployments/CSR,-,-?deep
```

Service Chaining with two VM Images

In this example, a Cisco CSR1000v VM and a Cisco ASAv VM are service chained. For that, you will have to deploy both VMs.

Figure 5: Service Chaining with two VM Images



Steps for Service Chaining with Two VM Images

1. Create a new bridge for service chaining.

```
admin@nfvis> curl -k -v -u admin:admin -H content-type:application/vnd.yang.data+xml -X
 POST https:// NFVIS_IP /api/config/bridges --data
'<bridge><name>sc-br</name></bridge>'
```

2. Create a new network for service chaining, and attach the bridge to the network.

```
admin@nfvis> curl -k -v -u admin:admin -H content-type:application/vnd.yang.data+xml -X
 POST https:// NFVIS_IP /api/config/networks --data
'<network><name>sc-net</name><bridge>sc-br</bridge> </network>'
```

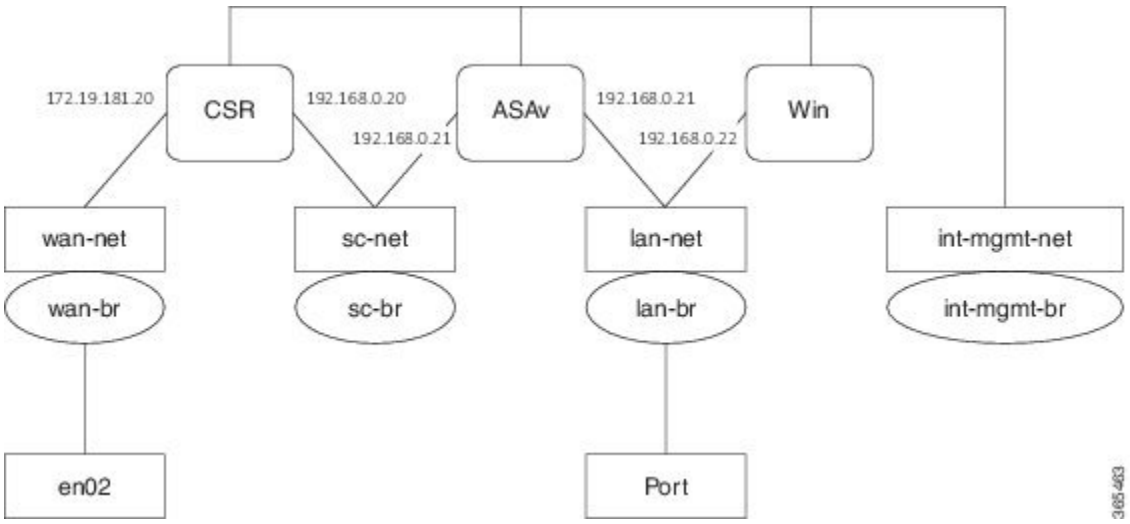
3. Verify that all bridges and networks are configured.
4. Deploy the Cisco CSR 1000V VM and verify the deployment status.
5. Deploy the cisco ASAv VM, and verify the deployment status.

   See Steps for Deploying a VM , on page 20 for API command details for Steps 3 to 5.

Service Chaining of Multiple VMs with Windows or Linux Servers

In this example, multiple VMs will be service chained. Cisco CSR 1000V and Cisco ASAv VMs are already deployed. This section covers Linux server deployment (Windows 2012 server can also be deployed using the same steps.)

Figure 6: Service Chaining of Multiple VMs with Windows or Linux Servers



Steps for Service Chaining of Multiple VMs with Windows or Linux Servers

1. Create networks and bridges as required, and attach the port to the LAN bridge.

See Steps 1 and 2 in Steps for Service Chaining with Two VM Images, on page 23 for details on creating networks and bridges.

2 Deploy Cisco CSR 1000V and Cisco ASAv, and verify their deployment status.

3 Deploy the Linux server VM.

4 Verify the server deployment status.

See the Steps for Deploying a VM , on page 20 for API command details for Steps 2 to 4.

VM Image Packaging

VM packaging is based on the Open Virtualization Format (OVF) packaging standard, in which a single file is distributed in open virtualization appliance (OVA) format. The *.ova* image is shared using a TAR archive file with the root disk image and descriptor files.

Note Cisco Enterprise NFVIS supports VM packaging in *.ova* or *.tar.gz* (compressed form of OVA) format. Ensure that all supported third party VM images are available in the supported format.

The below table lists the configuration files available with the VM package. These files are used during deployment.

Table 1: Package Content Details

File	Description	Mandatory/Optional
Package Manifest (package.mf)	Lists the files in the package and the expected checksum for the files.	Mandatory
VM image properties (vmname_properties.xml)	XML file with resources and features supported by the VM	Mandatory
VM image (vmname_ovf_env.xml)	Image file of the VM. Multiple images are supported. One root_disk image file is mandatory.	Mandatory
VM bootstrap file-1	The bootstrap file 1 of the VM	Optional
VM bootstrap file-2	The bootstrap file 2 of the VM	Optional

Generating a VM OVA Package

OVA files are provided for Cisco CSR 1000V, Cisco ASAv, and tiny Linux and Windows server 2000. Vendors are responsible for packaging all third party VMs in the OVA format.

1 Create a VM qcow2 image.

2 Create an *image_properties.xml* file with the VM properties. Ensure that you add all mandatory fields. Include the profiles supported for the VM in this file, and select one default profile. If you do not want to monitor the VM bootup, make the bootup time as -1.

3 Create *vmname_ova_env.xml*, if any bootstrap configuration is required for the VM. If the bootstrap configuration requires inputs from the user, use the tokens in the xml file. These tokens are populated during the VM deployment with the provided data.

4 Create a *package.mf* file, which lists all the files to be bundled into the OVA file along with checksums.

5 Generate the OVA file using "tar -cvf ova_file_name list_of_files_to_be_bundled".

For example, *tar -cvf csr1000v.ova csr1000v-universalk9.03.16.01a.S.155-3.S1a-ext-serial.qcow2 image_properties.xml csr_ovf_env.xml package.mf*.

Package Manifest File

The package manifest XML file provides a list of the files in the package with their names and their expected checksum. SHA1 algorithm (sha1sum) is used to calculate the checksum. This is a mandatory file to be bundled in the VM package. The manifest file must be named as *package.mf*.

Table 2: Package Manifest File Details

Property Name	Description	Property Tag	Mandatory/Optional
File information	XML tree with details of file name, file type, and expected checksum. The root_image and image_properties files are required.	<file_info>	Mandatory
File name	Name of the file	<name>	Mandatory
File type	Describes the file type. Supported types: • root_image • image_properties • bootstrap_config_file • ephemeral_disk1_image • ephemeral_disk2_image	<type>	Mandatory
Expected checksum	The calculated SHA1 checksum to be validated.	<sha1_checksum>	Mandatory

VM Image Properties File

This XML file provides information about the resources supported or required for the VM operation. All mandatory parameters have to be defined. It also supports custom attributes. This is a mandatory file to be bundled in the VM package. The OVA package supports up to 10 disks to be bundled into the package.

Table 3: VM Image Properties File Details

Property Name	Description	Property Tag	Possible Values	Mandatory/Optional
VNF Type	VM functionality provided. Router and firewall are predefined types.	<vnf_type>	Router, firewall, Windows, Linux, and custom_type	Mandatory
Name	Name associated with the VM packaging. This name is referenced for VM deployment.	<name>	Any	Mandatory
Version	Version of the package	<version>	Any	Mandatory
Boot-up time	Boot-up time (in seconds) of the VNF before it can be reachable via ping.	<bootup_time>	Any in seconds, (-1) to not monitor boot-up	Mandatory
Root Disk Image Bus	Root image disk bus	<root_file_disk_bus>	virtio, scsi, and ide	Mandatory
Disk-1 bus type	Additional disk1 image disk bus	<disk_1_file_disk_bus>	virtio, scsi, and ide	Optional
Disk-2 bus type	Disk2 image disk bus	<disk_2_file_disk_bus>	virtio, scsi, and ide	Optional
Disk-10 bus type	Disk10 image disk bus	<disk_10_file_disk_bus>	virtio, scsi, and ide	Optional
Root Disk Image format	Root image disk format	<root_image_disk_format>	qcow2 and raw	Mandatory
Disk-1 Image format	Additional disk 1 image format	<disk_1_image_format>	qcow2 and raw	Optional
Disk-2 Image format	Disk 2 image format	<disk_2_image_format>	qcow2 and raw	Optional

Disk-10 Image format	Disk 10 image format	<disk_10_image_format>	qcow2 and raw	Optional
Serial Console	Serial console supported	<console_type_serial>	true, false	Optional
Minimum vCPU	Minimum vCPUs required for a VM operation	<vcpu_min>		Mandatory
Maximum vCPU	Maximum vCPUs supported by a VM	<vcpu_max>		Mandatory
Minimum memory	Minimum memory in MB required for VM operation	<memory_mb_min>		Mandatory
Maximum memory	Maximum memory in MB supported by a VM	<memory_mb_max>		Mandatory
Minimum root disk size	Minimum disk size in GB required for VM operation	<root_disk_gb_min>		Optional
Maximum root disk size	Maximum disk size in GB supported by a VM	<root_disk_gb_max>		Optional
Maximum vNICs	Maximum number of vNICs supported by a VM	<vnic_max>		Mandatory
SRIOV support	SRIOV supported by VM interfaces. This should have a list of supported NIC device drivers.	<sriov_supported>	true, false	Optional
SRIOV driver list	List of drivers to enable SRIOV support	< sriov_driver_list>		Optional
PCI passthru support	PCI passthru support by VM interfaces	<pcie_supported>	true, false	Optional
PCIE driver list	List of VNICS to enable PCI passthru support	< pcie _driver_list>		Optional
Bootstrap-file-1	The bootstrap (bootstrap-1) file for the VM	< bootstrap_file_1>	File name of the bootstrap file 1	Optional
Bootstrap-file-2	The bootstrap file (bootstrap-2) for the VM	<bootstrap_file-2>	File name of the bootstrap file 2	Optional

Custom properties	List of properties can be defined within the custom_property tree. (Example: For CSR, the technology packages are listed in this block.)	<custom_property>		Optional
Profiles for VM deployment	List of VM deployment profiles. Minimum one profile is required	<profiles>		Mandatory
Default profile	The default profile is used when no profile is specified during deployment.	<default_profile>		Mandatory
Monitoring Support	A VM supports monitoring to detect failures.	<monitoring_supported>	true, false	Mandatory
Monitoring Method	A method to monitor a VM. Currently, only ICMP ping is supported.	<monitoring_methods>	ICMPPing	Mandatory if monitoring is true
Low latency	If a VM's low latency (for example, router and firewall) gets dedicated resource (CPU) allocation. Otherwise, shared resources are used.	<low_latency>	true, false	Mandatory
Privileged-VM	Allows special features like promiscuous mode and snooping . By default, it is false.	<privileged_vm>	true, false	Optional

Profile Properties File

Table 4: Profile Properties Details

Property Name	Description	Tag	Possible Values	Mandatory/Optional
Profile for VM deployment	A profile defines the resources required for VM deployment. This profile is referenced during VM deployment.	<profile>		Mandatory
Name	Profile name	<name>	Any	Mandatory

Description	Description of the profile	<description>	Any	Mandatory
vCPU	vCPU number in a profile	<vcpus>		Mandatory
Memory	Memory - MB in profile	<memory_mb>		Mandatory
Root Disk Size	Disk size - GB in profile .	<root_disk_mb>		Mandatory
Custom properties	Custom properties can be added to a profile. For example, technology package for Cisco CSR1000v)	<custom_property>		Optional

Note A virtual console is supported by default. Specify the root disk size as zero for multiple disks (for example, vWaas deployment) as the system does not support populating multiple disk sizes. Actual disk sizes are calculated from the root_disk files.

Bootstrap Configuration File

The bootstrap configuration file is an **XML** file, and contains properties specific to a VM and the environment. Properties can have tokens, which can be populated during deployment time from the deployment payload.

Example for VM Packaging

The following examples illustrate the templates for VM packaging using the Cisco CSR 1000V image:

- Example: Package.mf, on page 29
- Example: Image Properties, on page 30
- Example: Bootstrap Configuration File, on page 31

Example: Package.mf

```
** sha1sum - for calculating checksum
<PackageContents>
  <File_Info>
    < name>csr_serial_3.16.01a.S.qcow2</file_name>
    < type>root_image</file_type>
    <sha1_checksum>93de73ee3531f74fddf99377972357a8a0eac7b</sha1_checksum>
  </File_Info>
  <File_Info>
    < name>image_properties.xml</file_name>
    < type>image_properties</file_type>
    <sha1_checksum>c5bb6a9c5e8455b8698f49a489af3082c1d9e0a9</sha1_checksum>
</File_Info>
```

```
  <File_Info>
    < name>csr_ovf_env.xml</file_name>
    < type> bootstrap_file_1</file_type>
    <sha1_checksum>c5bb6a9c5e8455b8698f49a489af3082c1d9e0a9</sha1_checksum>
   </File_Info>
   <File_Info>
     < name>csr_disk1_image.qcow2</file_name>
     < type>ephemeral_disk1_image</file_type>
     <sha1_checksum>aac24513098ec6c2f0be5d595cd585f6a3bd9868</sha1_checksum>
   </File_Info>
</PackageContents>
```

Example: Image Properties

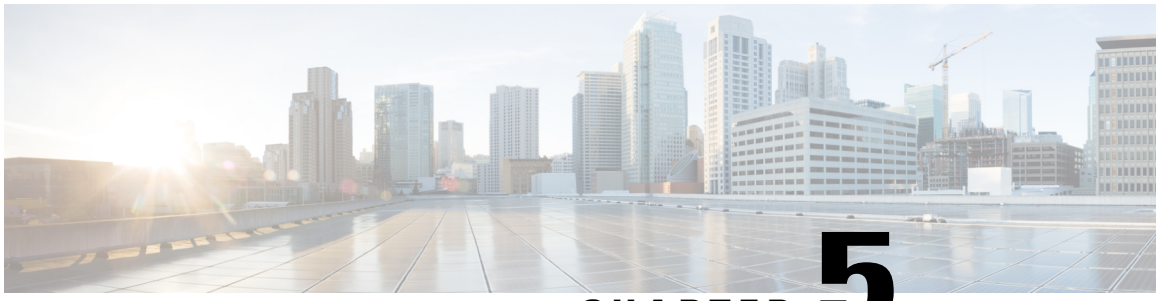
```
<?xml version="1.0" encoding="UTF-8"?>
<image_properties>
    <vnf_type>ROUTER</vnf_type>
    <name>csr1000v-universalk9</name>
    <version>03.16.01.S</version>
    <bootup_time>600</ bootup_time >
    <root_file_disk_bus>virtio</root_file_disk_bus>
    <root_image_disk_format>qcow2</root_image_disk_format>
    <vcpu_min>1</vcpu_min>
    <vcpu_max>8</vcpu_max>
    <memory_mb_min>4096</memory_mb_min>
    <memory_mb_max>8192</memory_mb_max>
    <vnic_max>8</vnic_max>
    <root_disk_gb_min>8</root_disk_gb_min>
    <root_disk_gb_max>8</root_disk_gb_max>
    <console_type_serial>true</console_type_serial>
    <sriov_supported>true</sriov_supported>
    <sriov_driver_list>igb</sriov_driver_list>
    <sriov_driver_list>igbvf</sriov_driver_list>
    <sriov_driver_list>i40evf</sriov_driver_list>
    <pcie_supported>true</pcie_supported>
    <pcie _driver_list> igb </pcie_driver_list>
    <pcie _driver_list> igbvf</pcie_driver_list>
    <pcie _driver_list> i40evf</pcie_driver_list>
    <bootstrap_file_1> ovf-env.xml </bootstrap_file_1>
    <monitoring_supported>true</monitoring_supported>
    <monitoring_methods>ICMPPing</monitoring_methods>
    <low_latency>true</low_latency>
    <privileged_vm>true</privileged_vm>
    <cdrom>true</cdrom>
    <custom_property>
        <tech_package>ax</tech_package>
        <tech_package>sec</tech_package>
        <tech_package>ipbase</tech_package>
        <tech_package>appx</tech_package>
    </custom_property>
    <profiles>
     <profile>
      <name>csr1kv-small</name>
       <description>CSR upto 50MBPS performance</description>
      <vcpus>1</vcpus>
       <memory_mb>4096</memory_mb>
       <root_disk_mb>8</root_disk_mb>
     </profile>
     <profile>
      <name>csr1kv-medium</name>
       <description>CSR upto 250MBPS performance</description>
       <vcpus>2</vcpus>
        <memory_mb>4096</memory_mb>
        <root_disk_mb>8</root_disk_mb>
     </profile>
```

```
    <profile>
     <name>csr1kv-large</name>
      <description>CSR upto 500MBPS performance</description>
      <vcpus>1</vcpus>
       <memory_mb>4096</memory_mb>
       <root_disk_mb>8</root_disk_mb>
    </profile>
  </profiles>
   <default_profile>small</default_profile>
</image_properties>
```

Example: Bootstrap Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>
<Environment
xmlns:oe="http://schemas.dmtf.org/ovf/environment/1">
  <PropertySection>
   <Property oe:key="com.cisco.csr1000v.config-version.1" oe:value="1.0"/>
    <Property oe:key="com.cisco.csr1000v.enable-ssh-server.1" oe:value="True"/>
    <Property oe:key="com.cisco.csr1000v.login-password.1" oe:value="admin"/>
    <Property oe:key="com.cisco.csr1000v.login-username.1" oe:value="lab"/>
    <Property oe:key="com.cisco.csr1000v.mgmt-interface.1" oe:value="GigabitEthernet1"/>
   <Property oe:key="com.cisco.csr1000v.mgmt-ipv4-gateway.1" oe:value="${NICID_0_GATEWAY}"/>

    <Property oe:key="com.cisco.csr1000v.mgmt-ipv4-network.1" oe:value=""/>
    <Property oe:key="com.cisco.csr1000v.license" oe:value="${TECH_PACKAGE}"/>
   <Property oe:key="com.cisco.csr1000v.ios-config-0002" oe:value="ip route 0.0.0.0 0.0.0.0
 ${NICID_0_GATEWAY}"/>
  </PropertySection>
</Environment>
```



CHAPTER 5

REST APIs for Cisco Enterprise NFVIS

- REST API Credentials, page 33
- API Request Methods, page 34
- VM Lifecycle Management APIs, page 34
- VM Operations APIs, page 57
- System and IP Configuration APIs, page 60
- System Utilization APIs, page 64
- Using Self-Signed and CA-Signed Certificates, page 82
- User Management APIs, page 86

REST API Credentials

Ensure you include the following credential information in REST API requisition:

- User name: admin
- Password: admin (default password)

The payload in request can be in XML or JSON format. The headers (content-type and accept) must be set accordingly.

The following two groups of headers are supported:

Table 5: Supported Headers

xml	Content-Type:application/vnd.yang.data+xml Accept:application/vnd.yang.data+xml
JSON	Content-Type:application/vnd.yang.data+json Accept:application/vnd.yang.data+json

API Request Methods

The following are the supported REST API request methods:

HTTP Request Method	Description
GET	Retrieves the specified resource or representation. GET is a read-only operation that does not change the engine state or have any side effects. **Note** The GET method supports "*?deep*" query to get more detailed information.
POST	Submits data to be processed to the specified resource. The data to be processed is included in the request body. A POST operation can create a new resource.
PUT	Updates the specified resource with new information. The data that is included in the PUT operation replaces the previous data. • The PUT operation is used to replace or modify an existing resource. The PUT operation cannot be used to create a new resource. • The request body of a PUT operation must contain the complete representation of the mandatory attributes of the resource.
DELETE	Deletes a resource. If you delete a resource that has already been deleted, a 404 Not Found response is returned.

Note You can use any command line tool, such as curl, that supports transferring of data using the HTTPS protocol. All REST API commands must be preceded by *https://<host_server_ip>* . This is the Cisco Enterprise NFVIS host IP address.

VM Lifecycle Management APIs

VM Image Registration APIs

Table 6: VM Registration API

Action	Method	Payload Required	API
Image registration	POST	Yes	/api/config/esc_datamodel/images

Get image configuration	GET	No	/api/config/esc_datamodel/images?deep
Get image status	GET	No	/api/operational/esc_datamodel/opdata/images/image /<image_name>?deep
Image deletion	DELETE	No	/api/config/esc_datamodel/images/image/<image_name>

Example for Image Registration Payload

```
<image><name>csr1000v.03.16.01a</name><src>file://
<filename_with_full-path-of-the-ova-file>/csr1000v.ova</src></image>
```

Table 7: Image Registration Payload Description

Property	Type	Description	Mandatory/Default Value
<image> <name>	String	Name of the VM image	Yes
<src>	URL	Full path of the VM file	Yes

Example: Image Registration API

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml
 -X POST https://<NFVIS_IP>/api/config/esc_datamodel/images -d
'<image><name>csr1000v.03.16.01a</name><src>file://<filename_with_full-path-of
the-ova-file>/csr1000v.ova</src></image>'
\* About to connect() to 172.19.183.144 port 80 (#0)
\* Trying 172.19.183.144...
\* Connected to 172.19.183.144 (172.19.183.144) port 80 (#0)
\* Server auth using Basic with user 'admin'
> POST /api/config/esc_datamodel/images HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 172.19.183.144
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 87
>
\* upload completely sent off: 87 out of 87 bytes
< HTTP/1.1 201 Created
< Server:
< Location: [http://172.19.183.144/api/config/esc_datamodel/images/image/csr-image]
< Date: Thu, 10 Dec 2015 11:15:50 GMT
< Last-Modified: Thu, 10 Dec 2015 11:15:50 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1449-746150-710421
< Content-Length: 0
< Content-Type: text/html
< Pragma: no-cache
<
\* Connection #0 to host 172.19.183.144 left intact
```

Example: Get Image Configuration API

```
admin@nfvis> curl -k \-v \-u admin:admin \-H Accept:application/vnd.yang.data+xml \-H
Content-Type:application/vnd.yang.data+xml \-X
GET https://172.19.183.144/api/config/esc_datamodel/images?deep
\* About to connect() to 172.19.183.144 port 80 (#0)
\* Trying 172.19.183.144...
\* Connected to 172.19.183.144 (172.19.183.144) port 80 (#0)
\* Server auth using Basic with user 'admin'
> GET /api/config/esc_datamodel/images?deep HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 172.19.183.144
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
>
< HTTP/1.1 200 OK
< Server:
< Date: Thu, 10 Dec 2015 11:16:10 GMT
< Last-Modified: Thu, 10 Dec 2015 11:15:50 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1449-746150-710421
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Pragma: no-cache
<
<images xmlns="[http://www.cisco.com/esc/esc|http://www.cisco.com/esc/esc]"
xmlns:y="[http://tail-f.com/ns/rest|http://tail-f.com/ns/rest]"&amp;nbsp;
xmlns:esc="[http://www.cisco.com/esc/esc|http://www.cisco.com/esc/esc]">
 <image>
 <name>csr1000v.03.16.01a</name>
 <src>file:///data/nfvos-pkg/csr/csr1000v.ova</src>
 </image>
</images>
\* Connection #0 to host 172.19.183.144 left intact
\[root@localhost csr\]#
```

Example: Get Image Status API

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X
GET https://172.19.183.144/api/operational/esc_datamodel/opdata/images/image/csr-image?deep
/* About to connect() to 172.19.183.144 port 80 (#0)
/*   Trying 172.19.183.144...
/* Connected to 172.19.183.144 (172.19.183.144) port 80 (#0)
/* Server auth using Basic with user 'admin'
> GET /api/operational/esc_datamodel/opdata/images/image/csr-image?deep HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 172.19.183.144
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
> < HTTP/1.1 200 OK
< Server:
< Date: Thu, 10 Dec 2015 11:16:22 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Type: application/vnd.yang.data+xml< Transfer-Encoding: chunked
< Pragma: no-cache<
<image xmlns="http://www.cisco.com/esc/esc" xmlns:y="http://tail-f.com/ns/rest"
xmlns:esc="http://www.cisco.com/esc/esc">
<name>csr1000v.03.16.01a</name>
<image_id>585a1792-145c-4946-9929-e040d3002a59</image_id>
<public>true</public>
```

```
<state>IMAGE_ACTIVE_STATE</state></image>
/* Connection #0 to host 172.19.183.144 left intact
[root@localhost csr]#
```

Note The supported image states are: Undefined, Deploying, Inactive, Disabled, Stopping, Shut off, Starting, Rebooting, Inert, Active, Undeploying, Image Creating State, and Error.

Example: Image Deletion API

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X
DELETE https://172.19.183.144/api/config/esc_datamodel/images/image/csr-image


/*About to connect() to 172.19.183.144 port 80 (#0)
/*   Trying 172.19.183.144...
/* Connected to 172.19.183.144 (172.19.183.144) port 80 (#0)
/* Server auth using Basic with user 'admin'
> DELETE /api/config/esc_datamodel/images/image/csr-image HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 172.19.183.144
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml>
< HTTP/1.1 204 No Content
< Server:
< Date: Thu, 10 Dec 2015 12:44:28 GMT
< Last-Modified: Thu, 10 Dec 2015 12:44:28 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1449-751468-864441
< Content-Length: 0
< Content-Type: text/html
< Pragma: no-cache
</* Connection #0 to host 172.19.183.144 left intact
[root@localhost ~]#
```

Bridge Creation APIs

By default, a LAN bridge (lan-br) and a WAN bridge (wan-br) are created in the system. However, you will have to attach one port to the LAN bridge.

Table 8: Bridge Creation APIs

Action	Method	Payload Required	API
To create a bridge	POST	Yes	/api/config/bridges
To verify a bridge configuration	GET	No	/api/config/bridges?deep

To modify a bridge, and attach a port to the bridge	PUT	Yes	/api/config/bridges/bridge/<bridge name>
To delete a bridge	DELETE	No	/api/config/bridges/bridge/<bridge name>

Example for Bridge Creation Payload

```
<bridge><name>sc-br</name><port><name>eth3</name></port></bridge>
```

Table 9: Bridge Creation Payload Description

Property	Type	Description	Mandatory/Default Value
<bridge> <name>	String	Name of the bridge.	Yes
<port><name>	String	Name of the port the bridge is attached to.	Yes

Example: Bridge Creation API

```
admin@nfvis> curl -k -v -u admin:admin -H Content-Type:application/vnd.yang.data+xml -X
POST https://172.19.181.175/api/config/bridges -d
"<bridge><name>sc-br</name><port><name>eth3</name></port></bridge>"
* About to connect() to 172.19.181.175 port 443 (#0)
*   Trying 172.19.181.175... connected
* Connected to 172.19.181.175 (172.19.181.175) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* warning: ignoring value of ssl.verifyhost
* skipping SSL peer certificate verification
* SSL connection using TLS_DHE_RSA_WITH_AES_128_CBC_SHA
* Server certificate:
*  subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*  start date: Mar 21 20:02:15 2016 GMT
*  expire date: Mar 19 20:02:15 2026 GMT
*  common name: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*  issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> POST /api/config/bridges HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.16.2.3 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: 172.19.181.175
> Accept: */*
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 66
>
< HTTP/1.1 201 Created
< Server: nginx/1.6.3
< Date: Sat, 02 Apr 2016 00:21:25 GMT
< Content-Type: text/html
< Content-Length: 0
< Location: https://172.19.181.175/api/config/bridges/bridge/sc-br
< Connection: keep-alive
< Last-Modified: Sat, 02 Apr 2016 00:21:24 GMT
```

```
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1459-556484-952070
< Pragma: no-cache
<
* Connection #0 to host 172.19.181.175 left intact
* Closing connection #0
```

Example: Get Bridge Configuration API

```
admin@nfvis> curl -k -v -u admin:admin -H Content-Type:application/vnd.yang.data+xml -X
GET "https://172.19.181.175/api/config/bridges?deep"
* About to connect() to 172.19.181.175 port 443 (#0)
*   Trying 172.19.181.175... connected
* Connected to 172.19.181.175 (172.19.181.175) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* warning: ignoring value of ssl.verifyhost
* skipping SSL peer certificate verification
* SSL connection using TLS_DHE_RSA_WITH_AES_128_CBC_SHA
* Server certificate:
*  subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*  start date: Mar 21 20:02:15 2016 GMT
*  expire date: Mar 19 20:02:15 2026 GMT
*  common name: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*  issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> GET /api/config/bridges?deep HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.16.2.3 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: 172.19.181.175
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.6.3
< Date: Sat, 02 Apr 2016 00:18:44 GMT
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Connection: keep-alive
< Last-Modified: Sat, 02 Apr 2016 00:16:51 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1459-556211-275675
< Pragma: no-cache
<

<bridges xmlns="http://www.cisco.com/nfv/network" xmlns:y="http://tail-f.com/ns/rest"
xmlns:network="http://www.cisco.com/nfv/network">
  <bridge>
    <name>lan-br</name>
    <port>
      <name>eth0</name>
    </port>
  </bridge>
  <bridge>
    <name>wan-br</name>
    <port>
      <name>eth1</name>
    </port>
  </bridge>
<bridge>
    <name>sc-br</name>
    <port>
      <name>eth3</name>
    </port>
  </bridge>
</bridges>
```

```
* Connection #0 to host 172.19.181.175 left intact
* Closing connection #0
```

Example: Bridge Deletion API

```
admin@nfvis> curl -k -v -u admin:admin -X
DELETE https://172.19.181.175/api/config/bridges/bridge/sc-br
* About to connect() to 172.19.181.175 port 443 (#0)
*   Trying 172.19.181.175... connected
* Connected to 172.19.181.175 (172.19.181.175) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* warning: ignoring value of ssl.verifyhost
* skipping SSL peer certificate verification
* SSL connection using TLS_DHE_RSA_WITH_AES_128_CBC_SHA
* Server certificate:
*  subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*  start date: Mar 21 20:02:15 2016 GMT
*  expire date: Mar 19 20:02:15 2026 GMT
*  common name: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*  issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> DELETE /api/config/bridges/bridge/sc-br HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.16.2.3 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: 172.19.181.175
> Accept: */*
>
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Sat, 02 Apr 2016 00:19:30 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Sat, 02 Apr 2016 00:19:30 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1459-556370-37827
< Pragma: no-cache
<
* Connection #0 to host 172.19.181.175 left intact
* Closing connection #0
```

Network Creation APIs

By default a LAN network (lan-net) and a WAN network (wan-net) are created in the system. You will have to attach at least one network to a bridge.

Table 10: Network Creation APIs

Action	Method	Payload Required	API
To create a network	POST	Yes	/api/config/networks

To verify network configuration details	GET	No	/api/config/networks?deep
To modify a network	PUT	Yes	/api/config/networks/network/<network name>
To delete a network	DELETE	No	/api/config/networks/network/<network name>

Example for Network Creation Payload

```
<network><name>sc-net</name><bridge>sc-bridge</bridge></network>
```

Table 11: Network Creation Payload Description

Property	Type	Description	Mandatory/Default Value
<network> <name>	String	Name of the network.	Yes
<bridge>	String	Name of the bridge the network is attached to.	Yes
trunk	Boolean	Network set to trunk mode.	No/true
sriov	Boolean	SR-IOV supported on the network.	No/false

Example: Network Creation API

```
admin@nfvis> curl -k -v -u admin:admin -H Content-Type:application/vnd.yang.data+xml -X
POST  https://172.19.181.175/api/config/networks -d
"<network><name>sc-net</name><bridge>sc-bridge</bridge></network>"

* About to connect() to 172.19.181.175 port 443 (#0)
*   Trying 172.19.181.175... connected
* Connected to 172.19.181.175 (172.19.181.175) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* warning: ignoring value of ssl.verifyhost
* skipping SSL peer certificate verification
* SSL connection using TLS_DHE_RSA_WITH_AES_128_CBC_SHA
* Server certificate:
*  subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*  start date: Mar 21 20:02:15 2016 GMT
*  expire date: Mar 19 20:02:15 2026 GMT
*  common name: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*  issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> POST /api/config/networks HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.16.2.3 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: 172.19.181.175
> Accept: */*
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 62
```

```
>
< HTTP/1.1 201 Created
< Server: nginx/1.6.3
< Date: Sat, 02 Apr 2016 00:14:37 GMT
< Content-Type: text/html
< Content-Length: 0
< Location: https://172.19.181.175/api/config/networks/network/sc-net
< Connection: keep-alive
< Last-Modified: Sat, 02 Apr 2016 00:14:37 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1459-556077-695828
< Pragma: no-cache
<
* Connection #0 to host 172.19.181.175 left intact
* Closing connection #0
```

Example: Get Network Configuration API

```
admin@nfvis> curl -k -v -u  admin:admin -H Content-Type:application/vnd.yang.data+xml -X
GET "https://172.19.181.175/api/config/networks?deep"
* About to connect() to 172.19.181.175 port 443 (#0)
*   Trying 172.19.181.175... connected
* Connected to 172.19.181.175 (172.19.181.175) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* warning: ignoring value of ssl.verifyhost
* skipping SSL peer certificate verification
* SSL connection using TLS_DHE_RSA_WITH_AES_128_CBC_SHA
* Server certificate:
*  subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*  start date: Mar 21 20:02:15 2016 GMT
*  expire date: Mar 19 20:02:15 2026 GMT
*  common name: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*  issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> GET /api/config/networks?deep HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.16.2.3 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: 172.19.181.175
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.6.3
< Date: Sat, 02 Apr 2016 00:09:25 GMT
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Connection: keep-alive
< Last-Modified: Sat, 02 Apr 2016 00:09:11 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1459-555751-599025
< Pragma: no-cache
<

<networks xmlns="http://www.cisco.com/nfv/network" xmlns:y="http://tail-f.com/ns/rest"
xmlns:network="http://www.cisco.com/nfv/network">
  <network>
    <name>lan-net</name>
    <bridge>lan-br</bridge>
  </network>
  <network>
    <name>wan-net</name>
    <bridge>wan-br</bridge>
  </network>
  <network>
    <name>sc-net</name>
    <bridge>sc-bridge</bridge>
  </network>
</networks>
```

```
* Connection #0 to host 172.19.181.175 left intact
* Closing connection #0
```

Example: Network Deletion API

```
admin@nfvis> curl -k -v -u  admin:admin -X DELETE
https://172.19.181.175/api/config/networks/network/sc-net
* About to connect() to 172.19.181.175 port 443 (#0)
*   Trying 172.19.181.175... connected
* Connected to 172.19.181.175 (172.19.181.175) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* warning: ignoring value of ssl.verifyhost
* skipping SSL peer certificate verification
* SSL connection using TLS_DHE_RSA_WITH_AES_128_CBC_SHA
* Server certificate:
*  subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*  start date: Mar 21 20:02:15 2016 GMT
*  expire date: Mar 19 20:02:15 2026 GMT
*  common name: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*  issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> DELETE /api/config/networks/network/sc-net HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.16.2.3 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: 172.19.181.175
> Accept: */*
>
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Sat, 02 Apr 2016 00:10:04 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Sat, 02 Apr 2016 00:10:04 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1459-555804-514130
< Pragma: no-cache
<
* Connection #0 to host 172.19.181.175 left intact
* Closing connection #0
```

Custom Flavor Creation APIs

After registering a VM, you can define custom flavors of the VM based on your requirements. These flavors are also known as profiles.

Table 12: Flavor Creation APIs

Action	Method	Payload Required	APIs
To create a flavor	POST	Yes	/api/config/esc_datamodel/flavors

To get configuration details of a flavor	GET	No	• /api/config/esc_datamodel/flavors • /api/config/esc_datamodel/flavors?deep • /api/config/esc_datamodel /flavors/flavor/<flavor_name>?deep
To view the operational status of a flavor	GET	No	/api/operational/esc_datamodel /opdata/flavors/flavor/<flavor_name>?deep
To delete a flavor	DELETE	No	/api/config/esc_datamodel/flavors/flavor/<flavor-name>

Example for Flavor Creation Payload

```
<flavor>
 <name>CSR_FLAVOR</name>
 <vcpus>2</vcpus>
 <memory_mb>4096</memory_mb>
 <root_disk_mb>0</root_disk_mb>
 <ephemeral_disk_mb>0</ephemeral_disk_mb>
 <swap_disk_mb>0</swap_disk_mb>
</flavor>
```

Table 13: Description for Flavor Creation Payload

Property	Type	Description	Mandatory/Default Value
<flavor> <name>	String	Name of the flavor.	Yes
<vcpus>	Number	The virtual CPU identifier.	Yes
<memory_mb>	Number	Memory of the virtual CPU.	Yes
<root_disk_mb>	Number	Memory of the root disk.	No
<ephemeral_disk_mb>	Number	A temporary storage that is added to your instance.	No
<swap_disk_mb>	Number	The space used on a hard disk as RAM	No

Example: Flavor Creation API

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X
POST https://172.19.183.144/api/config/esc_datamodel/flavors -d
'<flavor><name>ASAv10</name><vcpus>2</vcpus><memory_mb>4096</memory_mb>
<root_disk_mb>0</root_disk_mb><ephemeral_disk_mb>0
</ephemeral_disk_mb><swap_disk_mb>0</swap_disk_mb></flavor>'
* About to connect() to 172.19.183.144 port 80 (#0)
* Trying 172.19.183.144...
* Connected to 172.19.183.144 (172.19.183.144) port 80 (#0)
* Server auth using Basic with user 'admin'
> POST /api/config/esc_datamodel/flavors HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 172.19.183.144
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 188
>
* upload completely sent off: 188 out of 188 bytes
< HTTP/1.1 201 Created< Server:
< Location: http://172.19.183.144/api/config/esc_datamodel/flavors/flavor/CSR_FLAVOR_demo
< Date: Fri, 11 Dec 2015 11:15:23 GMT
< Last-Modified: Fri, 11 Dec 2015 11:15:23 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1449-832523-873124
< Content-Length: 0
< Content-Type: text/html
< Pragma: no-cache
<
* Connection #0 to host 172.19.183.144 left intact
[root@localhost ~]#
```

Example: Get Flavor Configuration API

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X
GET https://172.19.183.144/api/config/esc_datamodel/flavors?deep
* About to connect() to 172.19.183.144 port 80 (#0)
* Trying 172.19.183.144...
* Connected to 172.19.183.144 (172.19.183.144) port 80 (#0)
* Server auth using Basic with user 'admin'
> GET /api/config/esc_datamodel/flavors?deep HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 172.19.183.144
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
>
< HTTP/1.1 200 OK
< Server:
< Date: Fri, 11 Dec 2015 11:11:31 GMT
< Last-Modified: Fri, 11 Dec 2015 01:32:26 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1449-797546-701321
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Pragma: no-cache
<
<flavors xmlns="http://www.cisco.com/esc/esc" xmlns:y="http://tail-f.com/ns/rest"
xmlns:esc="http://www.cisco.com/esc/esc">
 <flavor>
  <name>ASAv10</name>
  <description>ASAv10 profile</description>
  <vcpus>1</vcpus>
  <memory_mb>2048</memory_mb>
  <root_disk_mb>8192</root_disk_mb>
  <ephemeral_disk_mb>0</ephemeral_disk_mb>
  <swap_disk_mb>0</swap_disk_mb>
  <properties>
```

```
    <property>
    <name>source_image</name>
    <value>ASAv_IMAGE</value>
    </property>
   </properties>
  </flavor>
  <flavor>
   <name>ASAv30</name>
   <description>ASAv30 profile</description>
   <vcpus>4</vcpus>
   <memory_mb>8192</memory_mb>
   <root_disk_mb>16384</root_disk_mb>
   <ephemeral_disk_mb>0</ephemeral_disk_mb>
   <swap_disk_mb>0</swap_disk_mb>
   <properties>
     <property>
     <name>source_image</name>
     <value>ASAv_IMAGE</value>
     </property>
   </properties>
  </flavor>
</flavors>
* Connection #0 to host 172.19.183.144 left intact
[root@localhost ~]#
```

Example: Flavor Status API

```
admin@nfvis> curl -k -v -u admin:admin -X
GET https://172.19.183.144/api/operational/esc_datamodel/flavors?deep
* About to connect() to 172.19.183.144 port 80 (#0)
* Trying 172.19.183.144...
* Connected to 172.19.183.144 (172.19.183.144) port 80 (#0)
* Server auth using Basic with user 'admin'
> GET /api/operational/esc_datamodel/flavors?deep HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 172.19.183.144
> Accept: */*>
< HTTP/1.1 200 OK< Server:
< Date: Fri, 11 Dec 2015 10:58:48 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Pragma: no-cache
<flavors xmlns="http://www.cisco.com/esc/esc" xmlns:y="http://tail-f.com/ns/rest"
xmlns:esc="http://www.cisco.com/esc/esc">
 <flavor>
   <name>ASAv10</name>
   <description>ASAv10 profile</description>
   <vcpus>1</vcpus>
   <memory_mb>2048</memory_mb>
   <root_disk_mb>8192</root_disk_mb>
   <ephemeral_disk_mb>0</ephemeral_disk_mb>
   <swap_disk_mb>0</swap_disk_mb>
    <properties>
     <property>
     <name>source_image</name>
     <value>ASAv_IMAGE</value>
     </property>
    </properties>
 </flavor>
 <flavor>
   <name>ASAv30</name>
   <description>ASAv30 profile</description>
   <vcpus>4</vcpus>
   <memory_mb>8192</memory_mb>
   <root_disk_mb>16384</root_disk_mb>
   <ephemeral_disk_mb>0</ephemeral_disk_mb>
```

```
   <swap_disk_mb>0</swap_disk_mb>
    <properties>
     <property>
     <name>source_image</name>
     <value>ASAv_IMAGE</value>
     </property>
    </properties>
 </flavor>
</flavors>
* Connection #0 to host 172.19.183.144 left intact
```

Example: Flavor Deletion API

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X
DELETE https://172.19.183.144/api/config/esc_datamodel/flavors/flavor/CSR_FLAVOR_demo
* About to connect() to 172.19.183.144 port 80 (#0)
* Trying 172.19.183.144...
* Connected to 172.19.183.144 (172.19.183.144) port 80 (#0)
* Server auth using Basic with user 'admin'
> DELETE /api/config/esc_datamodel/flavors/flavor/CSR_FLAVOR_demo HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0> Host: 172.19.183.144
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
>
< HTTP/1.1 204 No Content
< Server:
< Date: Fri, 11 Dec 2015 11:16:26 GMT
< Last-Modified: Fri, 11 Dec 2015 11:16:26 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1449-832586-685717
< Content-Length: 0
< Content-Type: text/html
< Pragma: no-cache
<
* Connection #0 to host 172.19.183.144 left intact
[root@localhost ~]#
```

VM Deployment APIs

Table 14: VM Deployment APIs

Action	Method	Payload Required	API
Deploy a VM	POST	Yes	/api/config/esc_datamodel/tenants/tenant/admin/deployments
Verify deployment configuration	GET	No	• /api/config/esc_datamodel/tenants/tenant/admin/deployments?deep • /api/config/esc_datamodel/tenants/tenant/admin /deployments /<deployment_name>,-,-?deep

Delete a VM deployment	DELETE	No	/api/config/esc_datamodel/tenants/tenant/admin/deployments /deployment/<deployment_name>

Example for VM Deployment Payload

This section provides an example of deployment payload using the Cisco CSR 1000V image, and description for its parameters.

```
<deployment>
        <name>CSR</name>
    <vm_group>
        <name>CSR_VM</name>
        <image>csr-image</image>
        <bootup_time>600</bootup_time>
        <recovery_wait_time>0</recovery_wait_time>
        <recovery_policy>
            <action_on_recovery>REBOOT_ONLY</action_on_recovery>
        </recovery_policy>
        <interfaces>
            <interface>
                <nicid>0</nicid>
                <network>wan-net</network>
                <port_forwarding>
                    <port>
                        <type>ssh</type>
                        <protocol>tcp</protocol>
                        <vnf_port>22</vnf_port>
                        <external_port_range>
                            <start>20022</start>
                            <end>20022</end>
                        </external_port_range>
                    </port>
                </port_forwarding>
            </interface>
            <interface>
                <nicid>1</nicid>
                <network>lan-net</network>
                <ip_address>192.168.0.20</ip_address>
            </interface>
        </interfaces>
        <scaling>
            <min_active>1</min_active>
            <max_active>1</max_active>
        </scaling>
        <kpi_data>
            <kpi>
                <event_name>VM_ALIVE</event_name>
                <metric_value>1</metric_value>
                <metric_cond>GT</metric_cond>
                <metric_type>UINT32</metric_type>
                <metric_collector>
                    <type>ICMPPing</type>
                    <nicid>0</nicid>
                    <poll_frequency>3</poll_frequency>
                    <polling_unit>seconds</polling_unit>
                    <continuous_alarm>false</continuous_alarm>
                </metric_collector>
            </kpi>
        </kpi_data>
        <rules>
            <admin_rules>
                <rule>
                    <event_name>VM_ALIVE</event_name>
                    <action>ALWAYS log</action>
                    <action>TRUE servicebooted.sh</action>
```

```
                    <action>FALSE recover autohealing</action>
                </rule>
            </admin_rules>
        </rules>
        <config_data>
            <configuration>
                <dst>bootstrap_config</dst>
                <variable>
                    <name>TECH_PACKAGE</name>
                    <val>security</val>
                </variable>
            </configuration>
        </config_data>
    </vm_group>
</deployment>
```

Table 15: Description for VM Deployment Payload

Property	Type	Description	Mandatory/Default Value
<deployment> <name>	string	Name of the deployment	Yes
<vm_group> <name>	string	Name of the VM group.	Yes
<vm_group> <image>	string	Image name that was used to register.	Yes
bootup_time	integer	Bootup time could vary depending on the VM image that you have chosen. For example, bootup time is 600 seconds for a Cisco CSR 1000V image. If no monitoring is required for the VM, set the bootup time as -1. **Note** A monitored VM must have a valid bootup time. The corresponding KPI fields are mandatory for the monitored VM. In the case of an unmonitored VM, KPI fields are optional.	Yes (for monitored VMs)
<recovery_wait_time>	integer		
<recovery_policy> <action_on_recovery>	string		
<interface> <nicid>	integer	The network interface card ID. **Note** At least one NIC ID is mandatory for monitored VMs. It is optional for unmonitored VMs.	Yes (for monitored VMs)
<network>	string	Name of the network attached to the NIC ID.	Yes (for monitored VMs)
<ip_address>	string	IPv4 address	Yes (for monitored VMs)

<port_forwarding>	-	**Note** If port forwarding is included, all elements under it are mandatory.	No
<port><type>	enum	SSH, HTTPS, TCP, and Telnet	No
<protocol>	string	TCP	No
<vnf_port>	integer	Port number corresponding to the protocol used.	No
<external_port_range> <start> <end>	integer	Unique port number to specify the start and end range.	No
<scaling>	container	Specifies how many instances of a particular type of VM needs to be instantiated and whether elastic scale in and scale out is required.	Yes
<min_active>	integer	Describes the minimum number of VMs in the deployment. Irrespective of what the load is on these VMs, Cisco ESC ensures at least the minimum number of service VMs will always be running.	Yes
<max_active>	integer	Describes the maximum number of VMs to be activated by Cisco ESC. New VMs are activated when the load increases.	Yes
<kpi_data>	-	Key performance indicators data.	Yes (for monitored VMs)
<event_name>	string	Name of the event.	Yes (for monitored VMs)
<metric_value>	string	The metric threshold value of the KPI.	Yes (for monitored VMs)
<metric_cond>	enum	Specifies the direction of the metric value change for this KPI. There are four valid values: • GE & GT—An alarm is sent when the metric value increases from a lower position to equal or exceed the specified value. • LE & LT—An alarm is sent when the metric value decreases from a higher position to equal or go down the specified value.	Yes (for monitored VMs)
<metric_type>	integer	Supported metric types are INT8, UINT8, INT16, UINT16, INT32, UINT32, FLOAT, DOUBLE, and STRING .	Yes (for monitored VMs)

<metric_collector> <type>		If the image boot-up time is provided, monitoring must be set to ICMPPing. This field type can be empty if boot-up time is -1.	Yes (for monitored VMs)
<poll_frequency>	Integer		Yes (for monitored VMs)
<polling_unit>	String		Yes (for monitored VMs)
<continuous alarm>	Boolean		No/false
<rule> <event_name>			No
<action>	String	• Always log—Whether the event is pingable or not, the details are always logged. • TRUE servicebooted.sh—The action identified by this keyword in the dynamic mapping file is triggered when the VM moves from a non-pingable to a pingable state. • FALSE recover autohealing—The action identified by this keyword is triggered, and the VM is recovered without the administrator's intervention.	No
<configuration> <dst>		If the VM supports bootstrap configuration file in the OVA package, a token must be filled in the bootstrap template file during the VM deployment.	
<variable> <name>		TECH_PACKAGE is the token for a Cisco CSR 1000V image. This needs to be specified in the variable name. This varies with each VM.	
<val>			

Example: VM Deployment API for Cisco CSR 1000V

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X POST
https://172.19.183.144/api/config/esc_datamodel/tenants
/tenant/admin/deployments --data
'<deployment>
    <name>CSR</name>
```

```
    <vm_group>
        <name>CSR_VM</name>
        <image>csr-image</image>
        <bootup_time>600</bootup_time>
        <recovery_wait_time>0</recovery_wait_time>
        <recovery_policy>
            <action_on_recovery>REBOOT_ONLY</action_on_recovery>
        </recovery_policy>
        <interfaces>
            <interface>
                <nicid>0</nicid>
                <network>int-mgmt-net</network>
                <port_forwarding>
                    <port>
                        <type>ssh</type>
                        <protocol>tcp</protocol>
                        <vnf_port>22</vnf_port>
                        <external_port_range>
                            <start>20022</start>
                            <end>20022</end>
                        </external_port_range>
                    </port>
                </port_forwarding>
            </interface>
            <interface>
                <nicid>1</nicid>
                <network>lan-net</network>
                <ip_address>192.168.0.20</ip_address>
            </interface>
        </interfaces>
        <scaling>
            <min_active>1</min_active>
            <max_active>1</max_active>
        </scaling>
        <kpi_data>
            <kpi>
                <event_name>VM_ALIVE</event_name>
                <metric_value>1</metric_value>
                <metric_cond>GT</metric_cond>
                <metric_type>UINT32</metric_type>
                <metric_collector>
                    <type>ICMPPing</type>
                    <nicid>0</nicid>
                    <poll_frequency>3</poll_frequency>
                    <polling_unit>seconds</polling_unit>
                    <continuous_alarm>false</continuous_alarm>
                </metric_collector>
            </kpi>
        </kpi_data>
        <rules>
            <admin_rules>
                <rule>
                    <event_name>VM_ALIVE</event_name>
                    <action>ALWAYS log</action>
                    <action>TRUE servicebooted.sh</action>
                    <action>FALSE recover autohealing</action>
                </rule>
            </admin_rules>
        </rules>
        <config_data>
            <configuration>
                <dst>bootstrap_config</dst>
                <variable>
                    <name>TECH_PACKAGE</name>
                    <val>security</val>
                </variable>
            </configuration>
        </config_data>
    </vm_group>
</deployment>'
/* About to connect() to 172.19.183.144 port 80 (#0)
/* Trying 172.19.183.144...
/* Connected to 172.19.183.144 (172.19.183.144) port 80 (#0)
```

```
/* Server auth using Basic with user 'admin'
> POST /api/config/esc_datamodel/tenants/tenant/admin/deployments HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 172.19.183.144
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 1313
> Expect: 100-continue
> * Done waiting for 100-continue
< HTTP/1.1 201 Created
< Server:
< Location:
http://172.19.183.144/api/config/esc_datamodel/tenants/tenant/admin/deployments/deployment/CSRdep1
< Date: Thu, 10 Dec 2015 11:17:53 GMT
< Last-Modified: Thu, 10 Dec 2015 11:17:53 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1449-746273-842306
< Content-Length: 0
< Content-Type: text/html
< Pragma: no-cache
<
/* Connection #0 to host 172.19.183.144 left intact
[root@localhost csr]#
[root@localhost csr]# virsh list --all
Id    Name                           State---
2     CSRdep1.CSR_VMG                running
[root@localhost csr]#
```

Example: VM Deployment Deletion API

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X
DELETE
https://172.19.183.144/api/config/esc_datamodel/tenants/tenant/admin/deployments/deployment/CSRdep1
/* About to connect() to 172.19.183.144 port 80 (#0)
/* Trying 172.19.183.144...
/* Connected to 172.19.183.144 (172.19.183.144) port 80 (#0)
/* Server auth using Basic with user 'admin'
> DELETE /api/config/esc_datamodel/tenants/tenant/admin/deployments/deployment/CSRdep1
HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 172.19.183.144
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
>
< HTTP/1.1 204 No Content
< Server:
< Date: Thu, 10 Dec 2015 12:43:31 GMT
< Last-Modified: Thu, 10 Dec 2015 12:43:31 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1449-751411-880440
< Content-Length: 0
< Content-Type: text/html
< Pragma: no-cache
<
/* Connection #0 to host 172.19.183.144 left intact
[root@localhost ~]#
```

Example: VM Deployment API for Cisco vWAAS

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
```

```
Content-Type:application/vnd.yang.data+xml -X POST
https://172.19.181.204/api/config/esc_datamodel/tenants
/tenant/admin/deployments --data
'<deployment>
   <name>vWAAS</name>
   <vm_group>
        <name>vWAAS</name>
        <image>vWAAS</image>
        <bootup_time>-1</bootup_time>
        <recovery_wait_time>0</recovery_wait_time>
        <recovery_policy>
            <action_on_recovery>REBOOT_ONLY</action_on_recovery>
        </recovery_policy>
        <interfaces>
            <interface>
                <nicid>0</nicid>
                <network>lan-net</network>
                <ip_address>12.20.0.96</ip_address>
            </interface>
        </interfaces>
        <scaling>
            <min_active>1</min_active>
            <max_active>1</max_active>
        </scaling>
        <config_data>
            <configuration>
                <dst>bootstrap_config</dst>
                <variable>
                    <name>IP_ADDRESS</name>
                    <val>12.20.0.96</val>
                </variable>
                <variable>
                    <name>NETMASK</name>
                    <val>255.255.255.0</val>
                </variable>
                <variable>
                    <name>GATEWAY</name>
                    <val>12.20.0.30</val>
                </variable>
            </configuration>
        </config_data>
    </vm_group>
</deployment>'
```

Note Notice that the bootup time is -1. A '-1' value for bootup time indicates that the VM is not monitored.

Example: VM Deployment API for Cisco vWLC

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X
POST https://172.19.181.204/api/config/esc_datamodel/tenants/tenant/admin/deployments --data
'<deployment>
    <name>vWLC</name>
   <vm_group>
        <name>vWLC</name>
        <image>vWLC</image>
        <bootup_time>-1</bootup_time>
        <recovery_wait_time>0</recovery_wait_time>
        <interfaces>
            <interface>
                <nicid>0</nicid>
                <network>lan-net</network>
                <ip_address>12.20.0.24</ip_address>
            </interface>
        </interfaces>
        <scaling>
            <min_active>1</min_active>
```

```
            <max_active>1</max_active>
        </scaling>
        <kpi_data>
            <kpi>
                <event_name>VM_ALIVE</event_name>
                <metric_value>1</metric_value>
                <metric_cond>GT</metric_cond>
                <metric_type>UINT32</metric_type>
                <metric_collector>
                    <type>ICMPPing</type>
                    <nicid>0</nicid>
                    <poll_frequency>3</poll_frequency>
                    <polling_unit>seconds</polling_unit>
                    <continuous_alarm>false</continuous_alarm>
                </metric_collector>
            </kpi>
        </kpi_data>
        <rules>
            <admin_rules>
                <rule>
                    <event_name>VM_ALIVE</event_name>
                    <action>ALWAYS log</action>
                    <action>TRUE servicebooted.sh</action>
                    <action>FALSE recover autohealing</action>
                </rule>
            </admin_rules>
        </rules>
    </vm_group>
</deployment>'
```

Note If the user provides boostrap or day-0 config information, the system will ignore it because configuration is not supported for vWLC currently.

Adding or Deleting a vNIC Using the VM Deployment API

There is no new API to add or delete Virtualized Network Interface Cards (vNICs). Using the VM deployment API, you can add or delete as many vNICs as you want. For both actions, you will have to use the PUT method of the VM deployment API.

Example: Adding more than one vNIC

You should know the deployment name and the VM group name to use the PUT form of the VM deployment API. To get them, use the following commands before running the PUT form of the VM deployment API :

- GET https://<server_ip>/api/config/esc_datamodel/tenants/tenant/admin/deployments—Provides the names of all VMs that are deployed.
- GET https://<server_ip>/api/config/esc_datamodel/tenants/tenant/admin/deployments/deployment/CSR1—Provides the VM group name for a particular deployment.

Additional interfaces are passed into the same deployment URL as shown in this example. A new vNIC (NIC ID 2) is added to the deployed VM, CSR1.

```
curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X
PUT
https://<server_ip>/api/config/esc_datamodel/tenants/tenant/admin/deployments/deployment/CSR1/vm_group/CSR-VM/interfaces
 --data
'<interfaces>
    <interface>
```

```
        <nicid>0</nicid>
        <network>int-mgmt-net</network>
    </interface>
    <interface>
        <nicid>1</nicid>
        <network>sc-net</network>
    </interface>
    <interface>
        <nicid>2</nicid>
        <network>lan-net</network>
    </interface>
</interfaces>'
```

Example: Deleting a vNIC

Note Ensure that the VM is in shut down mode before deleting a vNIC. After deleting the vNIC, restart the VM.

To delete a vNIC that is part of the VM deployed, remove the vNIC ID from the payload, and then run the PUT form of the VM deployment API. For example, assume that you want to remove vNIC 2 from the above configuration (CSR1 deployment), use the PUT form of the VM deployment API as shown in the example:

```
curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X
PUT
https://<server_ip>/api/config/esc_datamodel/tenants/tenant/admin/deployments/deployment/CSR1/vm_group/CSR-VM/interfaces
 --data
'<interfaces>
    <interface>
        <nicid>0</nicid>
        <network>int-mgmt-net</network>
    </interface>
    <interface>
        <nicid>1</nicid>
        <network>sc-net</network>
    </interface>
  </interfaces>'
```

See the Example: VM Deployment API for Cisco CSR 1000V, on page 51 for details on the API command.

Changing the Flavor Using the VM Deployment API

There is no new API to change the existing flavor of a VM. Using the VM deployment API, you can change or update the flavor. For this, you will have to use the PUT method of the VM deployment API. Before changing an existing flavor to a new one, ensure that you have the new flavor created using the flavor creation API.

Example: Changing the Flavor

In this example, the existing flavor ID is changed to **csr-flavor** for the VM deployed as CSR1.

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml
-X PUT
https://<server_ip-server_port>/api/config/esc_datamodel/tenants/tenant/admin/deployments/deployment/CSR1/vm_group/CSR-VM/flavor
 --data
'<flavor>csr-flavor</flavor>'
```

Note A VM is automatically power cycled when a flavor of the VM is changed.

See the Example: VM Deployment API for Cisco CSR 1000V, on page 51 for details on the API command.

Service Chaining of VMs

Service chaining here refers to a set of network services in the form of VMs using an intermediate network. Cisco Enterprise NFVIS supports service chaining of two or more VMs eliminating the need of dedicated hardware devices for different types of network services.

To service chain traffic between two or more VMs, you will have to create the following:

- Bridge—For example, you can create a new bridge called sc-br.
- Network—For example, you can create a new network called sc-net.
- Launch VM1 and VM2 with an interface from each VM to the service chain network (sc-net).

For more details on how to configure service chaining using APIs, see the following topics:

- Service Chaining with two VM Images, on page 22
- Service Chaining of Multiple VMs with Windows or Linux Servers, on page 23

SR-IOV

When a network card that supports single root I/O virtualization (SR-IOV) is used, virtual functions for the corresponding ports are created automatically. These virtual functions are exposed as networks in the system. When deploying a VM, the user can attach these networks to the VM. By default, two virtual functions are created for each port of a SR-IOV supported network card.

Currently, the Intel X520 SFP+ network card is supported.

VM Operations APIs

Note To get the VM name, use the following operational status API:

/api/operational/esc_datamodel/opdata/tenants/tenant/admin/services/service_definition?deep

Table 16: VM Operations APIs

Action	Method	Payload Required	APIs
To start a VM	POST	Yes	/api/operations/vmAction
To stop a VM	POST	Yes	/api/operations/vmAction
To reboot a VM	POST	yes	/api/operations/vmAction

Example for VM Operations Payload

This section provides an example of operations payload for starting a VM. You can change the action type value to STOP or REBOOT as required.

```
<vmAction>

    <actionType>START</actionType>
    <vmName>csr1000v.03.16.01a</vmName>

 </vmAction>
```

Table 17: Description for VM Operations Payload

Property	Type	Description	Mandatory/Default Value
<vmAction><actionType>	String	Type of VM action.	Yes
<vmName>	String	Name of the VM instance.	Yes

Example: Start VM API

```
admin@nfvis> curl -k -v -u "admin:admin" -H "Accept:application/vnd.yang.data+xml" -H
"Content-Type:application/vnd.yang.data+xml" -X
POST https://172.19.183.144/api/operations/vmAction --data
'<vmAction><actionType>START</actionType><vmName>
 <vm-instance name></vmName></vmAction>'
* About to connect() to 172.19.183.144 port 80 (#0)
* Trying 172.19.183.144...
* Connected to 172.19.183.144 (172.19.183.144) port 80 (#0)
* Server auth using Basic with user 'admin'
> POST /api/operations/vmAction HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 172.19.183.144
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 130
>
* upload completely sent off: 130 out of 130 bytes
< HTTP/1.1 204 No Content
< Server:
< Date: Fri, 11 Dec 2015 11:36:33 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Length: 0
< Content-Type: text/html< Pragma: no-cache
<
* Connection #0 to host 172.19.183.144 left intact
[root@localhost ~]#
```

Example: Stop VM API

```
admin@nfvis> curl -k -v -u "admin:admin" -H "Accept:application/vnd.yang.data+xml" -H
"Content-Type:application/vnd.yang.data+xml" -X
POST
[https://172.19.183.144/api/operations/vmAction|http://172.19.183.144/api/operations/vmAction]
 \--data
'<vmAction><actionType>STOP</actionType><vmName><vm-instance name></vmName></vmAction>'
\* About to connect() to 172.19.183.144 port 80 (#0)
\*Trying 172.19.183.144...
\* Connected to 172.19.183.144 (172.19.183.144) port 80 (#0)
\* Server auth using Basic with user 'admin'
> POST /api/operations/vmAction HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 172.19.183.144
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 129
>
\* upload completely sent off: 129 out of 129 bytes
< HTTP/1.1 204 No Content
< Server:
< Date: Fri, 11 Dec 2015 11:34:36 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Length: 0
< Content-Type: text/html
< Pragma: no-cache
<
\* Connection #0 to host 172.19.183.144 left intact
\[root@localhost \~\]#
```

Example: Restart VM API

```
admin@nfvis> curl -k -v -u "admin:admin" -H "Accept:application/vnd.yang.data+xml" -H
"Content-Type:application/vnd.yang.data+xml" -X
POST https://172.19.183.144/api/operations/vmAction --data
'<vmAction><actionType>REBOOT</actionType><vmName>
 <vm-instance name></vmName></vmAction>'
* About to connect() to 172.19.183.144 port 80 (#0)
* Trying 172.19.183.144...
* Connected to 172.19.183.144 (172.19.183.144) port 80 (#0)
* Server auth using Basic with user 'admin'
> POST /api/operations/vmAction HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0> Host: 172.19.183.144
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 131
>
* upload completely sent off: 131 out of 131 bytes
< HTTP/1.1 204 No Content
< Server:
< Date: Fri, 11 Dec 2015 11:30:28 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Length: 0
< Content-Type: text/html
< Pragma: no-cache
<
* Connection #0 to host 172.19.183.144 left
intact[root@localhost ~]#
```

VM Operational Status APIs

Table 18: VM Operational Status APIs

Action	Method	Payload Required	API
To get all operational status of a VM	GET	No	• /api/operational/esc_datamodel/opdata/tenants?deep • /api/operational/esc_datamodel/opdata/tenants/tenant/admin?deep • /api/operational/esc_datamodel/opdata/tenants/tenant/admin/deployments?deep • /api/operational/esc_datamodel/opdata/tenants/tenant/admin/deployments/<deployment_name><service_name>,<service_version>?deep

System and IP Configuration APIs

System Details APIs

Action	Method	Payload Required	API
Set NFVIS host name	PUT	Yes	/api/config/system/setting/hostname
Set NFVIS default gateway	PUT	Yes	/api/config/system/setting/default-gw
Set IP address and netmask on the lan-br interface.	PUT	Yes	/api/config/system/setting/mgmt-ip

Action	Method	Payload Required	API
Set IP address on the wan-br interface. When using the static IP address, the <dhcp> tag needs to be set to be set to "disable". When the <dhcp> tag is set to "enable", the wan-br interface will get the IP address from DHCP and the static IP address will be ignored	PUT	Yes	/api/config/system/setting/wan-ip
Get the following system details: • NFVIS host name. • NFVIS default gateway. • IP address and netmask on the lan-br interface. • IP address and netmask on the wan-br interface.	GET	No	/api/config/system?deep

Example for System Details Payload

```
<system>
  <settings>
    <hostname>nfvis-33</hostname>
    <default-gw>192.168.1.2</default-gw>
    <mgmt>
     <ip-address>10.1.1.35</ip-address>
     <netmask>255.255.255.0</netmask>
    </mgmt>
    <wan>
     <dhcp>disable</dhcp>
     <ip-address>172.19.181.173</ip-address>
     <netmask>255.255.255.0</netmask>
    </wan>
  /settings>
</system>
```

Table 19: Description for System Details Payload

Property	Type	Description	Mandatory/Default Value

<settings><hostname>	String	Hostname of the system.	Yes
<default-gw>	Integer	IP address of the default gateway.	Yes
<mgmt> <ip-address>	Integer	Management IP address	Yes
<netmask>	Integer	Netmask for the IP address.	Yes
<wan> <dhcp>	String	Enable or disable DHCP.	Yes
<ip-address>	Integer	DHCP IP address.	Yes
<netmask>	Integer	Netmask for the DHCP IP address.	Yes

Example: System Details API

```
admin@nfvis> curl -k -v -u /dev/stderr -u admin:admin -H Accept:application/vnd.yang.data+xml
 -H
Content-Type:application/vnd.yang.data+xml -X
PUT https://172.19.181.173/api/config/system -d

<system>
    <settings>
        <hostname>nfvis-33</hostname>
        <default-gw>192.168.1.2</default-gw>
        <mgmt>
            <ip-address>10.1.1.35</ip-address>
            <netmask>255.255.255.0</netmask>
        </mgmt>
        <wan>
            <dhcp>disable</dhcp>
            <ip-address>172.19.181.173</ip-address>
            <netmask>255.255.255.0</netmask>
        </wan>
    </settings>
</system>

* About to connect() to 127.0.0.1 port 443 (#0)
*   Trying 127.0.0.1...
* Connected to 127.0.0.1 (127.0.0.1) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* SSL connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate:
*       subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*       start date: Apr 22 07:13:23 2016 GMT
*       expire date: Apr 20 07:13:23 2026 GMT
*       common name: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
*       issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> PUT /api/config/system HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 127.0.0.1
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 289
>
} [data not shown]
* upload completely sent off: 289 out of 289 bytes
```

```
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Sat, 23 Apr 2016 02:13:32 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Sat, 23 Apr 2016 02:11:38 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1461-377498-922360
< Pragma: no-cache
<
* Connection #0 to host 127.0.0.1 left intact
```

Example: Get System Details API

```
admin@nfvis> curl  -k -v -u admin:admin -X Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X GET
https://172.19.181.173/api/config/system?deep

<system xmlns="http://www.cisco.com/nfv" xmlns:y="http://tail-f.com/ns/rest"
xmlns:system="http://www.cisco.com/nfv">
    <settings>
        <hostname>nfvis-demo</hostname>
        <default-gw>192.168.1.2</default-gw>
        <mgmt>
            <ip-address>192.168.1.10</ip-address>
            <netmask>255.255.255.0</netmask>
        </mgmt>
        <wan>
            <dhcp>disable</dhcp>
            <ip-address>172.19.162.211</ip-address>
            <netmask>255.255.255.0</netmask>
        </wan>
    </settings>
<certificate>
    <y:operations>

<signing-request>/api/config/system/certificate/_operations/signing-request</signing-request>

      <install-cert>/api/config/system/certificate/_operations/install-cert</install-cert>
      <use-cert>/api/config/system/certificate/_operations/use-cert</use-cert>
    </y:operations>
  </certificate>
</system>
```

Default IP Configuration APIs

Table 20: Default IP Configuration APIs

Action	Method	Payload Required	API
To retrieve the basic IP configuration information	GET	No	/api/config/system?deep
To modify the basic IP configuration information	POST	No	/api/config/system

Example: Get Default IP Configuration

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X
GET https://172.19.183.144/api/config/system?deep

<system xmlns="http://www.cisco.com/nfv" xmlns:y="http://tail-f.com/ns/rest"
xmlns:system="http://www.cisco.com/nfv">
  <hostname>nfvos-2</hostname>
  <ip-address>192.168.1.1</ip-address>
  <netmask>255.255.255.0</netmask>
  <default-gw>172.19.181.1</default-gw>
</system>
```

Example: Modify Default IP Configuration

```
admin@nfvis> curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H
Content-Type:application/vnd.yang.data+xml -X
PUT http://172.19.183.144/api/config/system
<system>
 <hostname>nfvos-3</hostname>
 <ip-address>192.168.1.2</ip-address>
 <netmask>255.255.255.0</netmask>
 <default-gw>192.168.1.1</default-gw>
</system>
<system>
```

System Utilization APIs

Platform Details API

Table 21: Platform Details APIs

Action	Method	Payload Required	API
To get information about the hardware	GET	No	/api/operational/platform-detail

Response Example for the Platform Details API

```
Cisco Systems Inc
UCSC-C220-M4S
FCH1906V15L
09EA2FD6-B863-D64D-BE1E-F12414A26FF9
cisco nfv os ver 1.0.0
```

```
Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz 16 cores
16158876
111.8G

3.10.0-229.el7.x86_64

1.5.3
1.2.8
2.3.2

eno1
```

Port Details APIs

Table 22: Port Details APIs

Action	Method	Payload Required	API
To get information about the physical port	GET	No	/api/operational/platform-detail/port_detail

Response Example for the Port Details API

```
eno1
physical
up
1000
1500
a8:9d:21:ce:de:50
01:00.0
```

Host Port Statistics APIs

Table 23: Host Port Statistics APIs

Action	Method	Payload Required	API
To get the packet counts information (error-rx, error-tx, error-total, packets-rx, packets-tx, and packets-total) on all host interfaces	GET	No	• /api/operational/hostport • /api/operational/hostport/port-stats • /api/operational/hostport/port-stats?deep • /api/operational/hostport/port-stats/stats-usage/<endDateTime>,<duration>?deep

Example: Host Port Statistics API

```
admin@nfvis> curl -k -v -u "admin:admin" -X GET
"https://172.19.181.152/api/operational/hostport/port-stats/stats-usage/2015-12-03T14:54:00,5min?deep"
....
>
< HTTP/1.1 200 OK
< Server:
< Date: Wed, 24 Feb 2016 06:20:11 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Pragma: no-cache
<

<stats-usage xmlns="http://www.cisco.com/nfvos/hostport" xmlns:y="http://tail-f.com/ns/rest"
 xmlns:hostport="http://www.cisco.com/nfvos/hostport">
 <endDateTime>2015-12-03T14:54:00-00:00</endDateTime>
 <duration>5min</duration>
 <collectStartDateTime>2016-02-24T06:15:10-00:00</collectStartDateTime>
 <collectEndDateTime>2016-02-24T06:20:00-00:00</collectEndDateTime>
 <collectInterval>10</collectInterval>
 <port>
 <name>enp1s0f0</name>    <<== interface name, enp1s0f0
 <packets-total>[48.1, 48.68, 47.88, 48.34, 48.74, 48.0, 49.36, 54.68, 47.82, 48.12, 48.82,
 48.3, 46.28, 36.8, 44.78, 48.1,
 48.62, 47.92, 49.14, 47.12, 46.72, 47.8, 47.18, 48.28, 49.64, 48.96, 48.52, 47.62, 48.4]
 </packets-total>
 <packets-rx>[48.08, 48.6, 47.88, 48.32, 48.66, 48.0, 48.3, 50.44, 47.8, 48.04, 48.82,
48.28, 46.2, 36.8, 44.76,
 48.02, 48.62, 47.9, 49.06, 47.12, 46.7, 47.72, 47.18, 48.26, 49.56, 48.96, 48.5, 47.54,
48.4]
 </packets-rx>
 <packets-tx>[0.02, 0.08, 0.0, 0.02, 0.08, 0.0, 1.06, 4.24, 0.02, 0.08, 0.0, 0.02, 0.08,
0.0, 0.02, 0.08, 0.0, 0.02,
 0.08, 0.0, 0.02, 0.08, 0.0, 0.02, 0.08, 0.0, 0.02, 0.08, 0.0]
 </packets-tx>
 <errors-total>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </errors-total>
 <errors-rx>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </errors-rx>
 <errors-tx>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </errors-tx>
 </port>
 <port>
 <name>enp1s0f1</name>
 <packets-total>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </packets-total>
 <packets-rx>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </packets-rx>
 <packets-tx>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </packets-tx>
 <errors-total>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

```
 </errors-total>
 <errors-rx>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </errors-rx>
 <errors-tx>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </errors-tx>
 </port>
 <port>
 <name>enp4s0f0</name>
 <packets-total>[0.1, 0.08, 0.02, 0.08, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </packets-total>
 <packets-rx>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </packets-rx>
 <packets-tx>[0.1, 0.08, 0.02, 0.08, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </packets-tx>
 <errors-total>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </errors-total>
 <errors-rx>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </errors-rx>
 <errors-tx>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
 </errors-tx>
 </port>

!
!
!
```

Table 24: Field Description for Host Port Statistics API Response

Field	Description
<name>enp1s0f0</name>	Interface name
<packets-total></packets-total>	Total packets counts
<packets-rx></packets-rx>	Received packets counts
<packets-tx></packets-tx>	Transmitted packets counts
<errors-total></errors-total>	Total error packets counts
<errors-rx></errors-rx>	Received error packets counts
<errors-tx></errors-tx>	Transmitted error packets counts

Host Port Statistics Table APIs

Table 25: Host Port Statistics Table APIs

Action	Method	Payload Required	API
To get statistics information about all ports	GET	No	/api/operational/hostport_stats_tbl/port_stats_tbl
To get statistics information about a single port	GET	No	/api/operational/hostport_stats_tbl/port_stats_tbl/ <port_name>,<endDateTime>,<duration>? deep

Note

- The default duration for the query to return statistics is set to five minutes.
- The date stamp returned by the API is universal date and time (GMT, not local time zone based). Use the **date -u** command to manually obtain the universal date and time on the host server.
- The "end date time" value is not processed yet. It could be the local time zone of the user.

Example: Host Port Statistics Table API (Single Port)

```
admin@nfvis> curl -k -v -u "admin:admin" -X GET
https://172.19.181.152/api/operational/hostport_stats_tbl/port_stats_tbl/enp1s0f1,2015-11-10T11:52:00-00:00,1D?deep

...

< HTTP/1.1 200 OK
< Server:
< Date: Wed, 24 Feb 2016 05:47:16 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Pragma: no-cache
<

<port_stats_tbl xmlns="http://www.cisco.com/nfvos/port_stat"
xmlns:y="http://tail-f.com/ns/rest" xmlns:nfvos_port_stat="http://www.cisc
o.com/nfvos/port_stat">
 <Name>enp1s0f1</Name>
 <startDateTime>2015-11-10T11:52:00-00:00</startDateTime>
 <duration>1D</duration>
 <Status>down</Status>
 <IP_Address>NA</IP_Address>
 <collectStartDateTime>2016-02-23T05:48:20-00:00</collectStartDateTime>
```

```
 <collectEndDateTime>2016-02-24T05:46:50-00:00</collectEndDateTime>
 <collectInterval>70</collectInterval>
 <rxpck>0</rxpck>
 <txpck>0</txpck>
 <rxpck_rate>0.0000000000e+00</rxpck_rate>
 <txpck_rate>0.0000000000e+00</txpck_rate>
</port_stats_tbl>
* Connection #0 to host 172.19.181.152 left intact
* Closing connection #0
* SSLv3, TLS alert, Client hello (1):
```

Table 26: Field Description for Port Statistics Table API Response

<Name></Name>	Name of the host interface or port
<startDateTime></startDateTime>	The required start date and time of the collection
<duration></duration>	The duration of this collection
<Status></Status>	Port status
<IP_Address></IP_Address>	IP address of this interface
<collectStartDateTime></collectStartDateTime>	The actual start date and time of this collection
<collectEndDateTime></collectEndDateTime>	The actual end date and time of this collection
<collectInterval></collectInterval>	Time interval of the collection
<rxpck></rxpck>	Received packets
<txpck></txpck>	Transmitted packets
<rxpck_rate></rxpck_rate>	Received packet rate (packets/second)
<txpck_rate></txpck_rate>	Transmitted packet rate (packets/second)

Host CPU State APIs

Table 27: Host CPU State APIs

Action	Method	Payload Required	API

To get the host CPU utilization of a CPU state on each of the CPU cores.	GET	No	• api/operational/hostcpu • /api/operational/hostcpu/cpu-state • /api/operational/hostcpu/cpu-state?deep • /api/operational/hostcpu/cpu-state/states • /api/operational/hostcpu/cpu-state/states?deep • /api/operational/hostcpu/cpu-state/states/<cpu-state>,<endDateTime>,<duration>?deep

Example: Host CPU State API

```
admin@nfvis> curl -k -v -u "admin:admin" -X GET
https://172.19.181.152/api/operational/hostcpu/cpu-state/states/non-idle,2015-12-03T14:54:00,5min?deep

...

< HTTP/1.1 200 OK
< Server:
< Date: Wed, 24 Feb 2016 05:59:29 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Pragma: no-cache
<

<states xmlns="http://www.cisco.com/nfvos/hostcpu" xmlns:y="http://tail-f.com/ns/rest"
xmlns:hostcpu="http://www.cisco.com/nfvos/hostcpu">
 <state>non-idle</state>
 <endDateTime>2015-12-03T14:54:00-00:00</endDateTime>
 <duration>5min</duration>
 <collectStartDateTime>2016-02-24T05:54:20-00:00</collectStartDateTime>
 <collectEndDateTime>2016-02-24T05:59:20-00:00</collectEndDateTime>
 <collectInterval>10</collectInterval>
 <cores>
 <id>0</id>
 <values>[2.84, 2.5, 2.04, 1.88, 2.08, 1.84, 2.3, 0.94, 3.18, 2.32, 3.58, 1.58, 0.98, 3.94,
 4.4, 3.1, 5.0, 2.18, 4.7, 2.38, 2.62, 2.46, 3.24, 3.42, 1.94, 1.96, 1.92, 4.02, 3.92,
3.62]</values>
/cores>
 <cores>
 <id>1</id>
 <values>[0.84, 2.1, 3.38, 3.12, 1.24, 2.7, 1.52, 1.44, 0.78, 0.6, 0.22, 0.3, 0.38, 1.0,
2.18, 1.72, 0.36, 1.34, 2.7, 2.36, 1.02, 1.44, 2.46, 1.38, 2.12, 0.84, 1.9, 1.9, 0.3,
0.82]</values>
 </cores>
 <cores>
 <id>2</id>
 <values>[3.82, 3.78, 1.9, 2.44, 1.66, 2.64, 2.4, 2.9, 4.94, 4.36, 1.94, 3.86, 2.62, 1.5,
1.84, 2.74, 1.46, 3.32, 2.44, 4.12, 0.74, 1.74, 3.44, 3.08, 2.36, 1.24, 1.0, 2.12, 0.32,
1.08]</values>
 </cores>
 <cores>
 <id>3</id>
 <values>[2.04, 1.96, 1.48, 3.14, 2.08, 1.9, 1.4, 1.42, 2.92, 2.38, 2.66, 0.8, 0.66, 1.58,
 1.6, 3.12, 1.7, 3.8, 4.4, 4.64, 2.12, 1.02, 1.64, 3.44, 2.08, 2.58, 3.26, 2.68, 1.16,
1.86]</values>
 </cores>
 <cores>
 <id>4</id>
 <values>[1.68, 2.68, 4.92, 3.0, 2.94, 3.54, 5.82, 1.52, 2.88, 7.56, 5.08, 2.54, 3.24, 1.38,
```

```
 4.22, 4.04, 6.62, 2.3, 3.98, 4.36, 4.58, 3.84, 1.58, 2.66, 1.98, 5.58, 3.22, 5.62, 7.44,
8.0]</values>
 </cores>
 <cores>
 <id>5</id>
 <values>[3.52, 1.54, 1.18, 0.68, 0.68, 1.32, 2.84, 3.6, 2.34, 1.34, 4.44, 3.16, 2.1, 2.1,
 3.22, 1.24, 1.0, 1.36, 2.52, 1.94, 3.64, 2.02, 3.04, 3.48, 2.88, 2.68, 3.68, 2.88, 1.94,
3.34]</values>
 </cores>
 <cores>
 <id>6</id>
 <values>[3.68, 5.38, 4.26, 4.04, 6.4, 3.7, 2.96, 4.7, 4.02, 2.44, 1.44, 5.0, 6.16, 7.02,
4.6, 4.22, 5.06, 7.42, 4.14, 2.84, 5.98, 3.24, 2.86, 5.18, 3.3, 2.16, 7.0, 6.54, 6.76,
2.2]</values>
 </cores>
 <cores>
 <id>7</id>
 <values>[3.52, 2.64, 3.58, 3.86, 5.3, 4.62, 2.86, 5.72, 1.68, 1.42, 2.78, 5.06, 6.24, 5.12,
 4.26, 1.88, 1.12, 2.12, 3.58, 1.42, 3.58, 6.76, 5.4, 5.44, 5.74, 5.4, 1.98, 2.22, 0.56,
1.74]</values>
 </cores>
</states>
* Connection #0 to host 172.19.181.152 left intact
* Closing connection #0
* SSLv3, TLS alert, Client hello (1):
```

Table 28: Field Description for Host CPU State API Response

<state></state>	CPU state
<endDateTime></endDateTime>	Required end date time.
<duration></duration>	Required duration
<collectStartDateTime></collectStartDateTime>	Actual start date time of this collection.
<collectEndDateTime></collectEndDateTime>	Actual end date time of this collection.
<collectInterval></collectInterval>	Time interval for data collection.
<cores> <id></id> <values></values> </cores>	CPU core ID CPU usage data in the specified time duration. (The time interval between each data is 10 seconds in this example.)

Host CPU Statistics Table API

Table 29: Host CPU Statistics Table APIs

Action	Method	Payload Required	API

To get the host CPU utilization statistics table (minimum, maximum, and average) of all CPU states on each of the CPU cores .	GET	No	• /api/operational/hostcpu • /api/operational/hostcpu/cpu-table • /api/operational/hostcpu/cpu-table?deep • /api/operational/hostcpu/cpu-table/cpu-usage • /api/operational/hostcpu/cpu-table/cpu-usage?deep • /api/operational/hostcpu/cpu-table/cpu-usage/<endDateTime>,<duration>?deep

Example: Host CPU Statistics Table API

```
admin@nfvis> curl -k -v -u "admin:admin" -X GET
"https://172.19.181.152/api/operational/hostcpu/cpu-table/cpu-usage/2015-12-03T14:54:00,1h?deep"

....

< HTTP/1.1 200 OK
< Server:
< Date: Wed, 24 Feb 2016 06:05:56 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Pragma: no-cache
<

<cpu-usage xmlns="http://www.cisco.com/nfvos/hostcpu" xmlns:y="http://tail-f.com/ns/rest"
xmlns:hostcpu="http://www.cisco.com/nfvos/hostcpu">
 <endDateTime>2015-12-03T14:54:00-00:00</endDateTime>
 <duration>1h</duration>
 <cores>
 <id>0</id>
 <states>
 <state>non-idle</state>
 <min>0.52</min>
 <max>9.22</max>
 <average>3.46</average>
 </states>
 <states>
 <state>interrupt</state>
 <min>0.0</min>
 <max>0.0</max>
 <average>0.0</average>
 </states>
 <states>
 <state>nice</state>
 <min>0.0</min>
 <max>0.0</max>
 <average>0.0</average>
 </states>
 <states>
 <state>softirq</state>
 <min>0.0</min>
 <max>0.08</max>
 <average>0.01</average>
 </states>
 <states>
 <state>steal</state>
 <min>0.0</min>
```

```
<max>0.0</max>
<average>0.0</average>
</states>
<states>
<state>system</state>
<min>0.0</min>
<max>1.24</max>
<average>0.17</average>
</states>
<states>
<state>user</state>
<min>0.0</min>
<max>2.42</max>
<average>0.37</average>
</states>
<states>
<state>wait</state>
<min>0.0</min>
<max>0.32</max>
<average>0.04</average>
</states>
</cores>
<cores>
<id>1</id>    <<== cpu core, 1
<states>
<state>non-idle</state>
<min>0.22</min>
<max>5.16</max>
<average>1.93</average>
</states>
<states>
<state>interrupt</state>
<min>0.0</min>
<max>0.0</max>
<average>0.0</average>
</states>
<states>
<state>nice</state>
<min>0.0</min>
<max>0.0</max>
<average>0.0</average>
</states>
<states>
<state>softirq</state>
<min>0.0</min>
<max>0.0</max>
<average>0.0</average>
</states>
<states>
<state>steal</state>
<min>0.0</min>
<max>0.0</max>
<average>0.0</average>
</states>
<states>
<state>system</state>
<min>0.0</min>
<max>1.14</max>
<average>0.15</average>
</states>
<states>
<state>user</state>
<min>0.0</min>
<max>1.68</max>
<average>0.22</average>
</states>
<states>
<state>wait</state>
<min>0.0</min>
<max>0.16</max>
<average>0.0</average>
</states>
</cores>
```

```
 <cores>
 <id>2</id>
 <states>
 <state>non-idle</state>
 <min>0.32</min>
 <max>14.42</max>
 <average>3.26</average>
 </states>
 <states>
 <state>interrupt</state>
 <min>0.0</min>
 <max>0.0</max>
 <average>0.0</average>
 </states>
 <states>
 <state>nice</state>
 <min>0.0</min>
 <max>0.0</max>
 <average>0.0</average>
 </states>
 <states>
 <state>softirq</state>
 <min>0.0</min>
 <max>0.08</max>
 <average>0.0</average>
 </states>
 <states>
 <state>steal</state>
 <min>0.0</min>
 <max>0.0</max>
 <average>0.0</average>
 </states>
 <states>
 <state>system</state>
 <min>0.0</min>
 <max>0.82</max>
 <average>0.14</average>
 </states>
 <states>
 <state>user</state>
 <min>0.0</min>
 <max>8.52</max>
 <average>0.36</average>
 </states>
 <states>
 <state>wait</state>
 <min>0.0</min>
 <max>0.08</max>
 <average>0.0</average>
 </states>
 </cores>

</cpu-usage>
* Connection #0 to host 172.19.181.152 left intact
* Closing connection #0
* SSLv3, TLS alert, Client hello (1):
```

Table 30: Field Description for Host CPU Statistics Table API Response

Field	Description
<duration></duration>	Duration of this collection

<cores> <id></id> <states> <state></state> <min></min> <max></max> <average></average> </states>	CPU core id Indicates the CPU state. This could be non-idle, interrupt, wait, nice, soft interrupt request line (IRQ), and user. CPU usage, minimum value CPU usage, maximum value CPU usage, average value

Host Memory Statistics APIs

Table 31: Host Memory Statistics APIs

Action	Method	Payload Required	API
To get the host memory utilization for all memory types	GET	No	• /api/operational/hostmemory • /api/operational/hostmemory?deep • /api/operational/hostmemory/mem-usage • /api/operational/hostmemory/mem-usage?deep • /api/operational/hostmemory/mem-usage/<endDateTime>,<duration>?deep

Example: Host Memory Statistics API

```
admin@nfvis> curl -k -v -u "admin:admin" -X GET
"https://172.19.181.152/api/operational/hostmemory/mem-usage/2015-12-03T14:54:00,5min?deep"

....

< HTTP/1.1 200 OK
< Server:
< Date: Wed, 24 Feb 2016 06:12:40 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Pragma: no-cache
<

<mem-usage xmlns="http://www.cisco.com/nfvos/hostmem" xmlns:y="http://tail-f.com/ns/rest"
xmlns:hostmem="http://www.cisco.com/nfvos/hostmem">
 <endDateTime>2015-12-03T14:54:00-00:00</endDateTime>
 <duration>5min</duration>
 <collectStartDateTime>2016-02-24T06:07:40-00:00</collectStartDateTime>
 <collectEndDateTime>2016-02-24T06:12:30-00:00</collectEndDateTime>
```

```
 <collectInterval>10</collectInterval>
 <buffered>[26.12, 26.12, 26.12, 26.12, 26.13, 26.13, 26.13, 26.13, 26.13, 26.13, 26.13,
26.13, 26.13,
 26.13, 26.13, 26.13, 26.13, 26.13, 26.13, 26.13, 26.13, 26.13, 26.13, 26.13, 26.13,
 26.13, 26.13, 26.13]
 </buffered>
 <cached>[755.73, 755.73, 755.73, 755.73, 755.73, 755.73, 755.73, 755.73, 755.73, 755.74,
755.76, 755.76, 755.76,
 755.76, 755.76, 755.76, 755.77, 755.77, 755.77, 755.77, 755.77, 755.77, 755.77, 755.77,
755.77, 755.77, 755.77, 755.77, 755.77]
 </cached>
 <free>[481.26, 481.29, 481.29, 481.21, 481.18, 481.19, 481.19, 481.17, 481.18, 481.2,
481.19, 481.16, 481.15, 481.14,
 481.16, 481.14, 481.14, 481.14, 481.14, 481.14, 481.07, 481.08, 481.11, 481.08, 481.0,
481.01, 481.01, 481.02, 480.98]
 </free>
 <used> [681.85, 681.81, 681.82, 681.9, 681.92, 681.91, 681.92, 681.94, 681.92, 681.89,
681.89, 681.91, 681.92, 681.92,
 681.9, 681.91, 681.91, 681.91, 681.91, 681.9, 681.96, 681.95, 681.92, 681.95, 682.03,
682.02, 682.02, 682.01, 682.05]
 </used>
 <slab_recl>[29.01, 29.01, 29.01, 29.01, 29.01, 29.01, 29.01, 29.01, 29.01, 29.01, 29.01,
29.01, 29.02, 29.02, 29.02,
 29.02, 29.02, 29.02, 29.02, 29.02, 29.02, 29.02, 29.02, 29.02, 29.02, 29.02, 29.02,
 29.02]
 </slab_recl>
 <slab_unrecl>[6.65, 6.66, 6.66, 6.66, 6.65, 6.65, 6.65, 6.65, 6.66, 6.65, 6.65, 6.65, 6.66,
 6.66, 6.66, 6.67,
 6.67, 6.67, 6.67, 6.67, 6.69, 6.67, 6.67, 6.67, 6.68, 6.67, 6.67, 6.67, 6.68]
 </slab_unrecl>
</mem-usage>
* Connection #0 to host 172.19.181.152 left intact
* Closing connection #0
* SSLv3, TLS alert, Client hello (1):
```

This API response provides usage information for the following memory types:

- Buffered
- Cached
- Free
- Used
- Slab recl
- Slab unrecl

Host Memory Statistics Table APIs

Table 32: Host Memory Statistics Table APIs

Action	Method	Payload Required	API

To get the host memory utilization in tabular format (minimum, maximum, and average) for each memory type	GET	No	• /api/operational/hostmem-table • /api/operational/hostmem-table?deep • /api/operational/hostmem-table/mem-usage • /api/operational/hostmem-table/mem-usage?deep • /api/operational/hostmem-table/mem-usage/<endDateTime>,<duration>?deep

Example: Host Memory Statistics Table APIs

```
admin@nfvis> curl -k -v -u "admin:admin" -X GET
"https://172.19.181.152/api/operational/hostmem-table/mem-usage/2015-12-03T14:54:00,5min?deep"
....
< HTTP/1.1 200 OK
< Server:
< Date: Wed, 24 Feb 2016 06:16:24 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Pragma: no-cache
<

<mem-usage xmlns="http://www.cisco.com/nfvos/hostmem" xmlns:y="http://tail-f.com/ns/rest"
xmlns:hostmem="http://www.cisco.com/nfvos/hostmem">
 <endDateTime>2015-12-03T14:54:00-00:00</endDateTime>
 <duration>5min</duration>
 <memory>
 <type>buffered</type>
 <min>26.13</min>
 <max>26.15</max>
 <average>26.14</average>
 </memory>
 <memory>
 <type>cached</type>
 <min>755.77</min>
 <max>755.81</max>
 <average>755.79</average>
 </memory>
 <memory>
 <type>free</type>
 <min>474.81</min>
 <max>481.17</max>
 <average>477.24</average>
 </memory>
 <memory>
 <type>slab_recl</type>
 <min>29.02</min>
 <max>29.02</max>
 <average>29.02</average>
 </memory>
 <memory>
 <type>slab_unrecl</type>
 <min>6.66</min>
 <max>6.72</max>
 <average>6.68</average>
 </memory>
 <memory>
 <type>used</type>
 <min>681.82</min>
 <max>688.16</max>
```

```
 <average>685.76</average>
 </memory>
</mem-usage>
* Connection #0 to host 172.19.181.152 left intact
* Closing connection #0
* SSLv3, TLS alert, Client hello (1):
```

Host Disk Statistics APIs

Table 33: Host Disk Statistics APIs

Action	Method	Payload Required	API
To get arrays of utilizations (per type) for the list of disks or disk partitions on the host server	GET	No	• /api/operational/hostdisk • /api/operational/hostdisk?deep • /api/operational/hostdisk/disk-stats • /api/operational/hostdisk/disk-stats?deep • /api/operational/hostdisk/disk-stats/usage/<endDateTime>,<duration>?deep

Example: Host Disk Statistics API

```
admin@nfvis> curl -k -v -u "admin:admin" -X GET
"https://172.19.181.152/api/operational/hostdisk/disk-stats/usage/2015-12-03T14:54:00,5min?deep"
....
>
< HTTP/1.1 200 OK
< Server:
< Date: Wed, 24 Feb 2016 06:24:52 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Pragma: no-cache
<

<usage xmlns="http://www.cisco.com/nfvos/hostdisk" xmlns:y="http://tail-f.com/ns/rest"
xmlns:hostdisk="http://www.cisco.com/nfvos/hostdisk">
 <endDateTime>2015-12-03T14:54:00-00:00</endDateTime>
 <duration>5min</duration>
 <collectStartDateTime>2016-02-24T06:19:50-00:00</collectStartDateTime>
 <collectEndDateTime>2016-02-24T06:24:40-00:00</collectEndDateTime>
 <collectInterval>10</collectInterval>
 <disk>
 <name>disk-dm-0</name>
 <io-time>[2.58, 0.12, 0.34, 1.32, 2.52, 0.48, 0.24, 1.8, 3.4, 0.18, 0.24, 1.24, 2.42, 0.16,
 0.54, 1.58,
 2.82, 0.2, 0.58, 1.1, 2.8, 0.02, 0.4, 1.9, 2.5, 0.1, 0.26, 1.34, 2.52]</io-time>
 <io-time-weighted>[330.5, 0.12, 22.14, 154.44, 266.68, 2.4, 29.04, 222.0, 423.48, 0.5,
12.64, 147.58,
 389.38, 0.4, 16.9, 152.82, 361.46, 0.78, 3.9, 97.54, 352.08, 0.04, 34.32, 206.9, 281.06,
0.1, 19.96, 151.5, 288.08]
```

```
 </io-time-weighted>
 <merged-read>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</merged-read>
 <merged-written>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</merged-written>
 <octets-read>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 81.92, 327.68, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</octets-read>
 <octets-written>[446054.4, 28262.4, 63897.6, 281559.04, 478986.24, 57671.68, 58163.2,
284917.76, 494796.8,
 57344.0, 47431.68, 188416.0, 314818.56, 37601.28, 105840.64, 283525.12, 449003.52, 47022.08,
 103628.8, 220282.88,
 454492.16, 25722.88, 93470.72, 365117.44, 446791.68, 46202.88, 66682.88, 277053.44,
452116.48]</octets-written>
 <ops-read>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0,
 0.0, 0.02, 0.08, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</ops-read>
 <ops-written>[95.38, 6.9, 14.78, 62.22, 103.7, 12.96, 12.94, 61.12, 107.02, 13.28, 10.96,
 40.66, 65.42, 8.98,
 23.68, 60.32, 95.6, 10.56, 23.96, 48.04, 96.64, 6.28, 20.54, 76.84, 96.36, 11.22, 14.74,
59.02, 96.52]</ops-written>
 <time-read>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</time-read>
 <time-written>[2.8, 0.0, 0.48, 2.44, 2.12, 0.16, 0.66, 3.46, 3.28, 0.0, 0.4, 2.82, 4.88,
0.02, 0.42, 2.12, 3.06,
 0.1, 0.16, 1.06, 2.96, 0.0, 0.52, 2.68, 2.4, 0.0, 0.44, 2.38, 2.5]</time-written>
 <pending-ops>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</pending-ops>
 </disk>
 <disk>
 <name>disk-dm-1</name>
 <io-time>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</io-time>
 <io-time-weighted>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</io-time-weighted>
 <merged-read>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</merged-read>
 <merged-written>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</merged-written>
 <octets-read>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</octets-read>
 <octets-written>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</octets-written>
 <ops-read>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</ops-read>
 <ops-written>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</ops-written>
 <time-read>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</time-read>
 <time-written>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</time-written>
 <pending-ops>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</pending-ops>
 </disk>
 <disk>
 <name>disk-dm-2</name>
 <io-time>[0.0, 0.0, 0.02, 0.08, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0,
```

```
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</io-time>
 <io-time-weighted>[0.0, 0.0, 0.02, 0.08, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</io-time-weighted>
 <merged-read>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</merged-read>
 <merged-written>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</merged-written>
 <octets-read>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</octets-read>
 <octets-written>[327.68, 655.36, 2949.12, 1310.72, 0.0, 901.12, 3604.48, 0.0, 327.68,
1310.72, 819.2,
 3522.56, 983.04, 0.0, 327.68, 1556.48, 2785.28, 7454.72, 1310.72, 1556.48, 2949.12, 7864.32,
 81.92, 4341.76,
 16138.24, 573.44, 983.04, 0.0, 1228.8]</octets-written>
 <ops-read>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</ops-read>
 <ops-written>[0.08, 0.08, 0.4, 0.32, 0.0, 0.14, 0.56, 0.0, 0.08, 0.32, 0.1, 0.46, 0.24,
0.0, 0.08, 0.38,
 0.46, 0.94, 0.32, 0.38, 0.54, 1.2, 0.02, 0.3, 0.9, 0.14, 0.24, 0.0, 0.18]</ops-written>
 <time-read>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</time-read>
 <time-written>[0.0, 0.0, 0.06, 0.24, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</time-written>
 <pending-ops>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</pending-ops>
 </disk>

!
!
!
```


This API response provides information about the disk name and data for various disk usage types.

Log API

The log API returns logs based on the following log types:

- nfvis-log
- detailed-log
- overview-log

Table 34: Log API

Action	Method	Payload Required	API

To get logging information	GET	No	• api/operational/logs/logs/nfvis-log • api/operational/logs/logs/detailed-log • api/operational/logs/logs/overview-log

Example: Log API

```
admin@nfvis> curl -k -v -u "admin:admin" -X GET
https://172.19.181.152/api/operational/logs/logs/nfvis-log
....
>
< HTTP/1.1 200 OK
< Server:
< Date: Wed, 24 Feb 2016 06:37:06 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Pragma: no-cache
<

<log xmlns="http://www.cisco.com/nfv/log" xmlns:y="http://tail-f.com/ns/rest"
xmlns:log="http://www.cisco.com/nfv/log">
 <name>nfvis-log</name>
 <logs>Subscribed to /networkFile descriptor 3 (/dev/null) leaked on lvs invocation. Parent
 PID 5168: /bin/sh
....
</logs>
</log>
* Connection #0 to host 172.19.181.152 left intact
* Closing connection #0
* SSLv3, TLS alert, Client hello (1):
```

Host Reboot API

Table 35: Host Reboot API

Action	Method	Payload Required	API
To reboot the host server	POST	No	/api/operations/hostaction/reboot

PnP Server APIs

Table 36: PnP Server APIs

Action	Method	Payload Required	API
To obtain the PnP IP address and port number	GET	No	/api/config/pnp?deep
To modify the PnP IP address and port number	PUT	No	/api/config/pnp
To delete the PnP IP address and port number	DELETE	No	/api/config/pnp

Response Example for the PnP Server API

```
172.19.181.11
(PnP server IP address)

80
(port number)
```

Using Self-Signed and CA-Signed Certificates

Cisco Enterprise NFVIS uses the following types of certificates to provide secure network access:

- Self-signed certificates—This is the default certificate that the system uses for secure communication.
- CA-signed certificates —Can be generated and signed by a certificate authority (CA). Cisco Enterprise NFVIS can import third party certificates.

A CA-signed certificate implementation includes the following tasks. You can perform these tasks using APIs.

1. Generate a certificate signing request (CSR) using the "signing-request" API.
2. Download the certificate signing request file to your system, and send to the certificate authority for issuing a digital certificate. The CA will return the digital certificate to you.
3. Upload the digital certificate to Cisco Enterprise NFVIS.
4. Install the certificate using the "install certificate" API.
5. Specify which certificate is to be used by the system. For this, use the "use certificate" API.

Certificate Creation APIs

Table 37: Certificate Creation APIs

Action	Method	Payload Required	API
To create a certificate signing request	POST	Yes	/api/operations/system/certificate/signing-request
To install a certificate, which will be used by the local portal and REST API	POST	Yes	/api/operations/system/certificate/install-cert
To switch between self-signed and CA signed certificates	POST	Yes	/api/operations/system/certificate/use-cert

Example for Signing Request Payload

```
<signing-request>
 <country-code>US</country-code>
 <state>California</state>
 <locality>San Jose</locality>
 <organization>Cisco</organization>
 <organization-unit-name>Cisco</organization-unit-name>
 <common-name>nfvis.cisco.com</common-name>
</signing-request>
```

Table 38: Description for Signing Request Payload

Property	Type	Description	Mandatory/Default Value
<country-code>	String	Two-letter ISO abbreviation for your country.	No
<state>	String	Name of the state where your organization's head office is located.	No

<locality>	Boolean	Name of the city where your organization's head office is located.	No
<organization>	Boolean	Name of the organization	No
<organization-unit-name>	String	Name of the department or group that will use the certificate.	No
<common-name>	URL	Fully qualified domain name that you want to secure.	Yes

Example for Install Certificate Payload

```
<install-cert>
  <path>file:///data/upload1/servercert.pem</path>
</install-cert>
```

Table 39: Description for Install Certificate Payload

Property	Type	Description	Mandatory/Default Value
<install-cert> <path>	URL	Full path of the certificate.	Yes

Example for Use Certificate Payload

```
<use-cert>
  <cert-type>ca-signed</cert-type>
</use-cert>
```

The <cert-type> parameter is mandatory in the use certificate payload. You can .

Table 40: Description for Use Certificate Payload

Property	Type	Description	Mandatory/Default Value
<use-cert> <cert-type>	String	The <self-signed> or <ca-signed> certificate type.	Yes

Example for the Signing Request API

```
admin@nfvis> curl -k -v -u admin:admin -H Content-Type:application/vnd.yang.data+xml -X
POST -d <signing-request><country-code>US</country-code><state>California</state><locality>San
 Jose</locality><organization>Cisco</organization>
<organization-unit-name>Cisco</organization-unit-name><common-name>nfvis.cisco.com</common-name></signing-request>

https://172.19.181.175/api/operations/system/certificate/signing-request
* About to connect() to 172.19.181.175 port 443 (#0)
* Trying 172.19.181.175... connected
```

```
* Connected to 172.19.181.175 (172.19.181.175) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* warning: ignoring value of ssl.verifyhost
* skipping SSL peer certificate verification
* SSL connection using TLS_DHE_RSA_WITH_AES_128_CBC_SHA
* Server certificate:
* subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* start date: Apr 04 23:26:13 2016 GMT
* expire date: Apr 02 23:26:13 2026 GMT
* common name: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> POST /api/operations/system/certificate/signing-request HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.16.2.3 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: 172.19.181.175
> Accept: */*
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 250
>
< HTTP/1.1 200 OK
< Server: nginx/1.6.3
< Date: Wed, 06 Apr 2016 23:29:39 GMT
< Content-Type: application/vnd.yang.operation+xml
< Content-Length: 85
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Vary: Accept-Encoding
< Pragma: no-cache
<
<output xmlns='http://www.cisco.com/nfv'>
 <url>/download/nfvis.csr</url>
</output>
* Connection #0 to host 172.19.181.175 left intact
* Closing connection #0
```

Example for the Install Certificate API

```
admin@nfvis> curl -k -v -u admin:admin -H Content-Type:application/vnd.yang.data+xml -X
POST -d <install-cert><path>file:///data/upload1/servercert.pem</path></install-cert>
 https://172.19.181.175/api/operations/system/certificate/install-cert
* About to connect() to 172.19.181.175 port 443 (#0)
* Trying 172.19.181.175... connected
* Connected to 172.19.181.175 (172.19.181.175) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* warning: ignoring value of ssl.verifyhost
* skipping SSL peer certificate verification
* SSL connection using TLS_DHE_RSA_WITH_AES_128_CBC_SHA
* Server certificate:
* subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* start date: Apr 04 23:26:13 2016 GMT
* expire date: Apr 02 23:26:13 2026 GMT
* common name: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> POST /api/operations/system/certificate/install-cert HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.16.2.3 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: 172.19.181.175
> Accept: */*
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 81
>
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Wed, 06 Apr 2016 23:19:33 GMT
```

```
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Pragma: no-cache
<
* Connection #0 to host 172.19.181.175 left intact
* Closing connection #0
```

Example for the Use Certificate API

```
admin@nfvis> curl -k -v -u admin:admin -H Content-Type:application/vnd.yang.data+xml -X
POST -d <use-cert><cert-type>ca-signed</cert-type></use-cert>
https://172.19.181.175/api/operations/system/certificate/use-cert
* About to connect() to 172.19.181.175 port 443 (#0)
* Trying 172.19.181.175... connected
* Connected to 172.19.181.175 (172.19.181.175) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* warning: ignoring value of ssl.verifyhost
* skipping SSL peer certificate verification
* SSL connection using TLS_DHE_RSA_WITH_AES_128_CBC_SHA
* Server certificate:
* subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* start date: Apr 04 23:26:13 2016 GMT
* expire date: Apr 02 23:26:13 2026 GMT
* common name: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> POST /api/operations/system/certificate/use-cert HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.16.2.3 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: 172.19.181.175
> Accept: */*
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 57
>
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Wed, 06 Apr 2016 23:23:19 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Pragma: no-cache
<
* Connection #0 to host 172.19.181.175 left intact
* Closing connection #0
```

User Management APIs

Action	Method	Payload Required	API
Get all users	GET	No	api/operational/system/users
Get a specific user	GET	No	api/operational/system/users/user/<name>
Add a user	POST	Yes	api/operations/system/users/add-user

Action	Method	Payload Required	API
Modify a user	POST	Yes	api/operations/system/users/modify-user
Delete a user	POST	Yes	api/operations/system/users/delete-user

Example: Get Users API

```
admin@nfvis> curl -X GET -k -v -u admin:admin
https://172.19.162.247/api/operational/system/users?deep

< HTTP/1.1 200 OK
< Server: nginx/1.6.3
< Date: Sat, 16 Apr 2016 02:08:00 GMT
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Pragma: no-cache
<

<users xmlns="http://www.cisco.com/nfv" xmlns:y="http://tail-f.com/ns/rest"
xmlns:system="http://www.cisco.com/nfv">
  <user>
    <user>admin</user>
    <name>admin</name>
  </user>
  <y:operations>
    <add-user>/api/operational/system/users/_operations/add-user</add-user>
    <modify-user>/api/operational/system/users/_operations/modify-user</modify-user>
    <delete-user>/api/operational/system/users/_operations/delete-user</delete-user>
  </y:operations>
</users>
* Connection #0 to host 172.19.162.247 left intact
```

Example: Get Specific User

```
admin@nfvis> curl -X GET -k -v -u admin:admin
https://172.19.162.247/api/operational/system/users/user/admin
> GET /api/operational/system/users/user/admin HTTP/1.1
> Host: 172.19.162.247
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.42.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.6.3
< Date: Sat, 16 Apr 2016 02:27:37 GMT
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Pragma: no-cache
<
```

Example: Add User API

```
admin@nfvis> curl -X POST -k -v -u admin:admin
https://172.19.162.247/api/operations/system/users/add-user -H
"content-type:application/vnd.yang.data+json" -H
"Accept:application/vnd.yang.data+json" -d '{"input": {"user": "test", "password": "test",
 "name": "test user"}}' -v

* Server auth using Basic with user 'admin'
> POST /api/operations/system/users/add-user HTTP/1.1
> Host: 172.19.162.247
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.42.1
> content-type:application/vnd.yang.data+json
> Accept:application/vnd.yang.data+json
> Content-Length: 68
>
* upload completely sent off: 68 out of 68 bytes
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Sat, 16 Apr 2016 02:11:15 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Pragma: no-cache
<
* Connection #0 to host 172.19.162.247 left intact
```

Example: Modify User API

```
admin@nfvis> curl -X POST -k -v -u admin:admin
https://172.19.162.247/api/operations/system/users/modify-user -H
"content-type:application/vnd.yang.data+json" -H
"Accept:application/vnd.yang.data+json" -d '{"input": {"user": "test", "password": "new
test password", "name": "modified test user"}}' -v

> POST /api/operations/system/users/modify-user HTTP/1.1
> Host: 172.19.162.247
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.42.1
> content-type:application/vnd.yang.data+json
> Accept:application/vnd.yang.data+json
> Content-Length: 90
>
* upload completely sent off: 90 out of 90 bytes
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Sat, 16 Apr 2016 02:17:13 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Pragma: no-cache
<
* Connection #0 to host 172.19.162.247 left intact
```

Example: Delete User API

```
admin@nfvis> curl -X POST -k -u admin:admin
https://172.19.162.247/api/operations/system/users/delete-user -H
"content-type:application/vnd.yang.data+json" -H
"Accept:application/vnd.yang.data+json" -d '{"input": {"user": "test"}}' -v
```

```
> POST /api/operations/system/users/delete-user HTTP/1.1
> Host: 172.19.162.247
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.42.1
> content-type:application/vnd.yang.data+json
> Accept:application/vnd.yang.data+json
> Content-Length: 27
>
* upload completely sent off: 27 out of 27 bytes
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Sat, 16 Apr 2016 02:25:56 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Pragma: no-cache
<
```