CONTENTS

PART I

NFVIS Related APIs  11

CHAPTER 1

Introduction to Cisco Enterprise NFVIS REST APIs  1

  REST API Credentials  1

  API Request Methods  1

CHAPTER 2

System and IP Configuration APIs  3

  System Configuration APIs  3

    Example: PUT System Configuration API  5

    Example: GET System Details API  6

  System Routes APIs  9

    Example: POST System Route API  9

    Example: PUT System Route API  10

    Example: GET System Route API  11

    Example: DELETE System Route API  12

VLAN APIs  13

  Example: PUT VLAN API  14

  Example: GET VLAN API  15

  Example: DELETE VLAN API  18

User Management APIs  20

  Example: POST Add User API  22

  Example: POST Change Role API  22

  Example: POST Change Password API  23

  Example: GET Users API  23

  Example: Delete User API  24

  Example: POST Configure Minimum Password Length  25
Examples: POST Configure Password Lifetime 25
Examples: POST Configure Account Inactivity Period 26
Example: POST Activate an Inactive User Account 27
TACACS+ Server APIs 28
Example: POST TACACS Server API 28
Example: GET TACACS Server API 29
Example: PUT TACACS Server API 30
Example: DELETE TACACS Server API 31
Trusted IP Connection APIs 32
Example: PUT Trusted IP Connection API 33
Example: GET Trusted IP Connection API 34
Banner and Message APIs 35
Example: PUT Banner-MOTD API 36
Example: GET Banner-MOTD API 36
Example: DELETE Banner-MOTD API 38
Disk Space APIs 39
Example: GET Disk Space API 39
System Time APIs 41
Example: PUT System Time Manual Time API 42
Example: PUT System Time Preferred Server API 42
Example: PUT System Time Backup Server API 43
Example: PUT System Time Timezone API 44
Example: GET System Time API 45
Platform Details API 46
Port Details APIs 48
Portal Access APIs 50
Example: PUT Portal Access (Enable/Disable) 50
Example: GET Portal Access API 51
System Log APIs 52
Example: POST System Log API 53
Example: GET System Log API 54
DPDK Support APIs 55
Backup and Restore APIs 55
Route Distribution APIs 57
### Dynamic SR-IOV APIs 58

### CHAPTER 3  PnP APIs 61

- **Certificate Creation APIs** 61
  - Example: POST Signing Request API 63
  - Example: POST Install Certificate API 63
  - Example: POST Use Certificate API 64
- **PnP Action APIs** 65
  - Example: POST PnP Action API 65
- **PnP APIs** 66
- **PnP Server APIs** 66
  - Example: PUT PnP Server API 67
  - Example: GET PnP Server API 68
  - Example: DELETE PnP Server API 69

### CHAPTER 4  Resource APIs 71

- **CPU Allocation Summary API** 71
  - Example: GET CPU Allocation Summary API 71
- **Resources CPU APIs** 72
  - Example: GET Resources CPU API 72
- **Resource Precheck APIs** 73
  - Example: GET Resource Precheck API 74
- **Resources VM APIs** 75
  - Example: GET Resources VNF API 75

### CHAPTER 5  Networks and Bridges APIs 77

- **Bridge APIs** 77
  - Example: POST Bridge Creation API 78
  - Example: GET Bridge Configuration API 79
  - Example: DELETE Bridge API 80
- **Network Creation APIs** 81
  - Example: POST Network API 82

### CHAPTER 6  VM Lifecycle Management APIs 85
VM Image Registration APIs 85
  Example: POST Image Registration API 87
  Example: POST Image Registration to External Disk API 87
  Example: GET Image Configuration API 88
  Example: GET Image Status API 89
  Example: DELETE Image Registration API 90

Custom Flavor Creation APIs 90
  Example: POST Flavor API 92
  Example: GET Flavor Configuration API 92
  Example: GET Flavor Status API 93

VM Deployment APIs 94
  Example: POST VM Deployment API for Cisco ISRv 95
  Example: DELETE VM Deployment API 99
  Examples for VM Deployment Payload with Bootstrap Configuration Options 100
  Adding or Editing a vNIC Using the VM Deployment API 105
  Changing the Flavor Using the VM Deployment API 107

VM Action APIs 107
  Example: POST Start VM API 108
  Example: POST Stop VM API 109
  Example: POST Restart VM API 109
  Example: POST Enable VM Monitoring API 110
  Example: POST Disable VM Monitoring API 111

VM Network APIs 112
  Network File System APIs 112
  VNC Console Start API 114
  VM Multi Serial Port APIs 114

CHAPTER 7

System Monitoring APIs 117
  Host CPU Stats APIs 117
    Example: GET Host CPU Stats API 118
  Host CPU Table API 120
    Example: GET Host CPU Table API 120
  Host Disk Stats APIs 123
    Example: GET Host Disk Stats API 124
Host Memory Stats APIs 130
  Example: GET Host Memory Stats API 130
Host Memory Table APIs 131
  Example: GET Host Memory Table APIs 132
Host Port Stats APIs 133
  Example: GET Host Port Stats API 134
Host Port Table APIs 136
  Example: GET Host Port Table API 137
VNF CPU Stats APIs 139
  Example: GET VNF CPU Stats API 140
VNF Disk Stats APIs 142
  Example: GET VNF Disk Stats API 142
VNF Memory Stats API 144
  Example: GET VNF Memory Stats API 144
VNF Port Stats APIs 146
  Example: GET VNF Port Stats API 146

CHAPTER 8
System Operations APIs 151
  External Disks API 151
    Example: GET External Disks API 151
  File List APIs 152
    Example: GET File List APIs 152
  File Delete API 154
    Example: File Delete API 155
  USB Mount API 156
    Example: POST USB Mount API 156
    Example: POST USB Unmount API 157
    Example: GET USB Mount Point 158
  USB Copy API 159
    Example: POST USB Copy API 159
  Host Reboot API 160
  DHCP Renew API 160
    Example: POST WAN DHCP Renew API 160
    Example: Bridge DHCP Renew 161
CHAPTER 9

SPAN Session and Packet Capture APIs 163
Example: POST SPAN Session API 164
Example: GET SPAN Session APIs 165
Packet Capture APIs 168
Example: POST Packet Capture APIs 169

CHAPTER 10

Upgrade Package APIs 171
Upgrade Package Register API 171
Example: POST Upgrade Package Register API 172
Example: GET Upgrade Package Register API 173
Example: DELETE Upgrade Package Register API 174
Upgrade Apply-Image APIs 175
Example: POST Upgrade Apply-Image API 175
Example: GET Upgrade Apply-Image API 176
Example: DELETE Upgrade Apply-Image API 177

CHAPTER 11

Factory Default Reset APIs 179
Example: POST Factory Default Reset All 179
Example: POST Factory Default Reset All Except Images 180
Example: POST Factory Default Reset All Except Images Connectivity 181

CHAPTER 12

Syslog Support APIs 183
Example: POST Syslog Server 183
Example: PUT Remote Logging Host Configuration 184
Example: DELETE Remote Logging Host Configuration 185
Example: PUT Syslog Severity 186
Example: PUT Syslog Facility 187
Example: GET Remote Logging Host 188

CHAPTER 13

SNMP Support APIs 191
Example: POST Configuring SNMP Communities 192
Example: POST SNMP Traps 192
Example: POST SNMP Host 192
CHAPTER 14  

**TACACS and RADIUS Support APIs** 195  

TACACS Support APIs 195  
  Example: POST TACACS Server 195  
  Example: PUT TACACS Server 196  
  Example: GET TACACS Server API 197  
  Example: DELETE TACACS Server 198  

RADIUS Support APIs 199  
  Example: GET RADIUS Server 199  
  Example: POST RADIUS Server 200  
  Example: PUT RADIUS Server 200  
  Example: DELETE RADIUS Server 200  

CHAPTER 15  

**Port and Port Channel APIs** 201  

Example: GET Port and Port Channel Information API 203  
Example: POST Create a Port Channel API 207  
Example: PUT Add a Port to a Port Channel API 209  
Example: PUT Add a Port Channel to an Existing Bridge API 210  
Example: PUT Configure the LACP Mode of a Port Channel API 211  
Example: PUT Configure the Bond Mode of a Port Channel API 212  
Example: PUT Configure Trunks on a Port Channel API 213  
Example: DELETE Remove a Port from a Port Channel API 213  
Example: PUT Remove a Port Channel from a Bridge API 214  
Example: DELETE Delete a Port Channel API 215  
Example: GET LLDP Information API 216  
Example: PUT Enable LLDP Configuration API 219  
Example: GET Port Admin Status API 220  
Example: PUT Configure Port Admin Status API 221  
Speed, Autoneg and Duplex APIs 222
CHAPTER 16  Port Security APIs  223

CHAPTER 17  Secure Overlay APIs  225

Single IP Configuration APIs  227

PART II  Switch Related APIs  229

CHAPTER 18  DOT1x APIs  231

DOT1x guest-vlan Timeout Value APIs  231
DOT1x Default authentication APIs  233
DOT1x System Authentication Control APIs  233
RADIUS Source Interface Address APIs  234

CHAPTER 19  IP Gateway APIs  237

IP Route APIs  238

CHAPTER 20  Spanning-Tree All or Individual Elements APIs  239

Create Spanning-Tree APIs  241
Modify Spanning-Tree APIs  243
Delete Spanning-Tree APIs  244

CHAPTER 21  Interface Stat APIs  247

Interface Port APIs  249

CHAPTER 22  Interface GigabitEthernet Switchport APIs  251

CHAPTER 23  Interface GigabitEthernet Spanning-Tree APIs  255

CHAPTER 24  SPAN/RSPAN APIs  257

CHAPTER 25  VLAN and interface VLAN related APIs  261
PART I

NFVIS Related APIs

• Introduction to Cisco Enterprise NFVIS REST APIs, on page 1
• System and IP Configuration APIs, on page 3
• PnP APIs, on page 61
• Resource APIs, on page 71
• Networks and Bridges APIs, on page 77
• VM Lifecycle Management APIs, on page 85
• System Monitoring APIs, on page 117
• System Operations APIs, on page 151
• SPAN Session and Packet Capture APIs, on page 163
• Upgrade Package APIs, on page 171
• Factory Default Reset APIs, on page 179
• Syslog Support APIs, on page 183
• SNMP Support APIs, on page 191
• TACACS and RADIUS Support APIs, on page 195
• Port and Port Channel APIs, on page 201
• Port Security APIs, on page 223
• Secure Overlay APIs, on page 225
Chapter 1

Introduction to Cisco Enterprise NFVIS REST APIs

- REST API Credentials, on page 1
- API Request Methods, on page 1

REST API Credentials

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

- User name: admin
- Password: password for admin

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 1: Supported Headers

<table>
<thead>
<tr>
<th>Format</th>
<th>Content-Type</th>
<th>Accept</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML</td>
<td>application/vnd.yang.data+xml</td>
<td>application/vnd.yang.data+xml</td>
</tr>
<tr>
<td>JSON</td>
<td>application/vnd.yang.data+json</td>
<td>application/vnd.yang.data+json</td>
</tr>
</tbody>
</table>

API Request Methods

The following are the supported REST API request methods:

<table>
<thead>
<tr>
<th>HTTP Request Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Retrieves the specified resource or representation. GET is a read-only operation that does not change the engine state or have any side effects.</td>
</tr>
<tr>
<td>POST</td>
<td>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.</td>
</tr>
</tbody>
</table>

Note: The GET method supports "?deep" query to get more detailed information.
<table>
<thead>
<tr>
<th>HTTP Request Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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.
CHAPTER 2

System and IP Configuration APIs

- System Configuration APIs, on page 3
- System Routes APIs, on page 9
- VLAN APIs, on page 13
- User Management APIs, on page 20
- TACACS+ Server APIs, on page 28
- Trusted IP Connection APIs, on page 32
- Banner and Message APIs, on page 35
- Disk Space APIs, on page 39
- System Time APIs, on page 41
- Platform Details API, on page 46
- Portal Access APIs, on page 50
- System Log APIs, on page 52
- DPDK Support APIs, on page 55
- Backup and Restore APIs, on page 55
- Route Distribution APIs, on page 57
- Dynamic SR-IOV APIs, on page 58

System Configuration APIs

Table 2: System Configuration APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To retrieve complete information on system configuration</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/settings-native/ /api/config/system/settings</td>
</tr>
<tr>
<td>To configure the system by setting the default gateway, management IP address and/or WAN IP address</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/system/settings</td>
</tr>
</tbody>
</table>
Example for System Configuration Payload

```xml
<system>
  <settings>
    <hostname>MyNFVIS123</hostname>
    <mgmt>
      <ip>
        <address>192.168.1.2</address>
        <netmask>255.255.255.0</netmask>
      </ip>
    </mgmt>
    <wan>
      <dhcp/>
    </wan>
  </settings>
</system>
```

In the example, the management interface is configured with a static IP address and the WAN interface is set to DHCP. You can configure both the management and the WAN interface with static IP addresses; however, you can configure DHCP on only one of the interfaces.

### Table 3: Description for System Details Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostname</td>
<td>String</td>
<td>Hostname of the system. The hostname now follows RFC952 rules, allowing only alphabets, numbers and hyphen. The hostname can begin and end with either an alphabet or a digit. Host software must handle host names of up to 255 characters.</td>
<td>Yes</td>
</tr>
<tr>
<td>default-gw</td>
<td>String</td>
<td>IP address of the default gateway.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Note**

When using default gateway, DHCP configuration is not allowed on any interface, include WAN and MGMT interfaces.
Yes

Management IP address

When an interface is configured with a static IP address, DHCP is automatically disabled on that interface.

Note

String

mgmt ip address

Netmask for the IP address.

mgmt ip netmask

String

Set dhcp on the WAN interface.

Note

String

wan dhcp

You can configure DHCP either on the WAN interface or the management interface; you cannot configure DHCP on both the interfaces simultaneously.

Note

Example: PUT System Configuration API

curl -v -u admin:admin -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -k -X PUT https://209.165.201.1/api/config/system -d "<system>  
  <settings>  
    <hostname>Do3rdENCSE75SettingsNoGW</hostname>  
    <default-gw>172.19.183.1</default-gw>  
    <mgmt>  
      <ip>  
        <address>172.19.183.75</address>  
        <netmask>255.255.255.0</netmask>  
      </ip>  
    </mgmt>  
    <wan>  
      <ip>  
        <address>4.3.2.5</address>  
        <netmask>255.255.0.0</netmask>  
      </ip>  
    </ wan>  
  </settings>  
</system>"
Example: GET System Details API

curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X GET https://209.165.201.1/api/operational/system/settings-native

Note: Unnecessary use of --request, GET is already inferred.

* Trying 209.165.201.1...*

* Connected to 209.165.201.1 (209.165.201.1) port 443 (NO)

* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH

* successfully set certificate verify locations:

* CAfile: /etc/ssl/certs/ca-bundle.crt

* CPath: none

* TLSv1.2 (OUT), TLS handshake, Client hello (1):

* TLSv1.2 (IN), TLS handshake, Server hello (2):

* NPnP negotiated HTTP/1.1

* TLSv1.2 (IN), TLS handshake, Certificate (11):

* TLSv1.2 (IN), TLS handshake, Server key exchange (12):

* TLSv1.2 (IN), TLS handshake, Server finished (14):

* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):

* TLSv1.2 (OUT), TLS change cipher, Client hello (1):

* TLSv1.2 (OUT), TLS handshake, Unknown (67):

* TLSv1.2 (OUT), TLS handshake, Finished (20):

* TLSv1.2 (IN), TLS change cipher, Client hello (1):

* TLSv1.2 (IN), TLS handshake, Finished (20):

* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384

* Server certificate:

* subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate

* start date: Sep 2 17:03:09 2016 GMT

* expire date: Aug 31 17:03:09 2026 GMT

* issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate

* SSL certificate verify result: self signed certificate (18), continuing anyway.

* Server auth using Basic with user 'admin'
GET /api/operational/system/settings-native HTTP/1.1
Host: 172.19.183.75
Authorization: Basic YWRtaW46YWRtaW4=
User-Agent: curl/7.50.1
Accept: application/vnd.yang.data+xml
Content-Type: application/vnd.yang.data+xml

  <mgmt>
    <ip-info>
      <interface>MGMT</interface>
      <ipv4_address>192.168.1.2</ipv4_address>
      <netmask>255.255.255.0</netmask>
      <ipv6_address>fe80::2f2:8bff:fec3:4a54</ipv6_address>
      <prefixlen>64</prefixlen>
      <mac_address>00:f2:8b:c3:4a:54</mac_address>
      <mtu>1500</mtu>
      <txqueuelen>1000</txqueuelen>
    </ip-info>
    <stats>
      <rx_packets>12481280</rx_packets>
      <rx_bytes>14392431432</rx_bytes>
      <rx_errors>0</rx_errors>
      <rx_dropped>210</rx_dropped>
      <rx_overruns>0</rx_overruns>
      <rx_frame>0</rx_frame>
      <tx_packets>3080505</tx_packets>
      <tx_bytes>238975886</tx_bytes>
      <tx_errors>0</tx_errors>
      <tx_dropped>0</tx_dropped>
      <tx_overruns>0</tx_overruns>
      <tx_frame>0</tx_frame>
      <tx_carrier>0</tx_carrier>
      <tx_collisions>0</tx_collisions>
    </stats>
    <dhcp>
      <enabled>false</enabled>
      <offer>false</offer>
      <interface>NA</interface>
      <fixed_address>0.0.0.0</fixed_address>
      <subnet_mask>0.0.0.0</subnet_mask>
      <gateway>0.0.0.0</gateway>
      <lease_time>0</lease_time>
      <message_type>0</message_type>
      <name_servers>NA</name_servers>
      <server_identifier>0.0.0.0</server_identifier>
      <renewal_time>0</renewal_time>
      <rebinding_time>0</rebinding_time>
      <vendor_encapsulated_options>NA</vendor_encapsulated_options>
      <domain_name>NA</domain_name>
      <renew>0001-01-01T00:00:00+00:00</renew>
  </mgmt>
</settings-native>
<rebind>0001-01-01T00:00:00-00:00</rebind>
<expire>0001-01-01T00:00:00-00:00</expire>
</dhcp>
</mgmt>
</wan>
</ip-info>
<interface>wan-br</interface>
<ipv4_address>209.165.201.22</ipv4_address>
<netmask>255.255.255.0</netmask>
<ipv6_address>fe80::2f2:8bff:fec3:49e0</ipv6_address>
<prefixlen>64</prefixlen>
<mac_address>00:f2:8b:c3:49:e0</mac_address>
<mtu>1500</mtu>
</ip-info>
</wan>
</stats>
<rx_packets>2971387</rx_packets>
<rx_bytes>420208255</rx_bytes>
<rx_errors>0</rx_errors>
<rx_dropped>229</rx_dropped>
<rx_overruns>0</rx_overruns>
<rx_frame>0</rx_frame>
<tx_packets>155</tx_packets>
<tx_bytes>45522</tx_bytes>
<tx_errors>0</tx_errors>
<tx_dropped>0</tx_dropped>
<tx_overruns>0</tx_overruns>
<tx_frame>0</tx_frame>
</stats>
</dhcp>
</ip-info>
<stats>
<rx_packets>2971387</rx_packets>
<rx_bytes>420208255</rx_bytes>
<rx_errors>0</rx_errors>
<rx_dropped>229</rx_dropped>
<rx_overruns>0</rx_overruns>
<rx_frame>0</rx_frame>
<tx_packets>155</tx_packets>
<tx_bytes>45522</tx_bytes>
<tx_errors>0</tx_errors>
<tx_dropped>0</tx_dropped>
<tx_overruns>0</tx_overruns>
<tx_frame>0</tx_frame>
</stats>
</dhcp>
</ip-info>
<interface>NA</interface>
<fixed_address>0.0.0.0</fixed_address>
<subnet_mask>0.0.0.0</subnet_mask>
<gateway>0.0.0.0</gateway>
<lease_time>0</lease_time>
<message_type>0</message_type>
<server_identifier>0.0.0.0</server_identifier>
<renewal_time>0</renewal_time>
<rebinding_time>0</rebinding_time>
<domain_name>NA</domain_name>
<renew>0001-01-01T00:00:00-00:00</renew>
<rebind>0001-01-01T00:00:00-00:00</rebind>
<expire>0001-01-01T00:00:00-00:00</expire>
</dhcp>
</wan>
</stats>
</dhcp>
</ip-info>
<domain>NA</domain>
<dns>
<nameserver1>172.19.183.147</nameserver1>
<nameserver2>0.0.0.0</nameserver2>
<nameserver3>0.0.0.0</nameserver3>
</dns>
<hostname>Do3rdENCS75SettingsNoGW</hostname>
<gateway>
<ipv4_address>209.165.201.1</ipv4_address>
</gateway>
</settings-native>
* Connection #0 to host 209.165.201.1 left intact
System Routes APIs

Table 4: System Routes APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To create a new route</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/system/routes</td>
</tr>
<tr>
<td>To modify an existing route</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/system/routes/route/&lt;host destination,netmask&gt;</td>
</tr>
<tr>
<td>To retrieve the details of a route</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/routes/route/&lt;host destination,netmask&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/config/system/routes</td>
</tr>
<tr>
<td>To delete a route</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/system/routes</td>
</tr>
</tbody>
</table>

Example for System Routes Payload

```
<route>
  <destination>209.165.201.1</destination>
  <prefixlen>16</prefixlen>
  <dev>lan-br</dev>
</route>
```

Table 5: System Routes Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination</td>
<td>String</td>
<td>The route destination address.</td>
<td>Yes</td>
</tr>
<tr>
<td>prefixlen</td>
<td>Integer</td>
<td>The netmask for the destination address.</td>
<td>Yes</td>
</tr>
<tr>
<td>gateway</td>
<td>String</td>
<td>The gateway for the route.</td>
<td>No</td>
</tr>
<tr>
<td>dev</td>
<td>String</td>
<td>The device/interface that the route will use.</td>
<td>No</td>
</tr>
</tbody>
</table>

Note

Though only the destination and prefixlen are mandatory parameters for creating a route, a valid route requires that you specify the gateway or the interface or both.

Example: POST System Route API

To create a new route:

```
curl -k -v -u "admin:admin" -H "Accept:application/vnd.yang.data+xml" -H
```
"Content-Type:application/vnd.yang.data+xml" -X POST 
https://209.165.201.1/api/config/system/routes -d
"<route><destination>209.165.201.5</destination><prefixlen>16</prefixlen></route>"

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (t0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAsfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* MPN, negotiated HTTP1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Aug 27 06:20:53 2016 GMT
  * expire date: Aug 25 06:20:53 2026 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'

> POST /api/config/system/routes HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.50.1
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 75
>
> upload completely sent off: 75 out of 75 bytes
< HTTP/1.1 201 Created
< Server: nginx/1.6.3
< Date: Sat, 27 Aug 2016 08:54:50 GMT
< Content-Type: text/html
< Content-Length: 0
< Location: https://209.165.201.1/api/config/system/routes/route/21.1.0.0,16
< Connection: keep-alive
< Last-Modified: Sat, 27 Aug 2016 08:54:49 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1472-288089-901692
< Pragma: no-cache
<
<

The above example does not create a valid route because the gateway or device is not specified.

Example: PUT System Route API

curl -k -v -u "admin:admin" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X PUT
Example: GET System Route API

To get route details and operational status for all routes:

curl -k -v -u "admin:admin" -X GET "https://209.165.201.1/api/operational/system/routes?deep"

Note: Unnecessary use of -X or --request, GET is already inferred.

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
  * TLSv1.2 (IN), TLS handshake, Server hello (2):
  * NPN, negotiated HTTP/1.1
  * TLSv1.2 (IN), TLS handshake, Certificate (11):
  * TLSv1.2 (OUT), TLS handshake, Server key exchange (12):
  * TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
  * TLSv1.2 (OUT), TLS change cipher, Client hello (1):
  * TLSv1.2 (OUT), TLS handshake, Unknown (67):
  * TLSv1.2 (OUT), TLS handshake, Finished (20):
  * TLSv1.2 (IN), TLS change cipher, Client hello (1):
  * TLSv1.2 (IN), TLS handshake, Finished (20):
  * SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
  * Server certificate:
    * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
    * start date: Aug 27 06:20:53 2016 GMT
    * expire date: Aug 25 06:20:53 2026 GMT
    * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'

Example: GET System Route API
Example: DELETE System Route API

```
curl -k -v -u "admin:admin" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X DELETE https://209.165.201.1/api/config/system/routes -d "<route><destination>21.1.0.0</destination><prefixlen>16</prefixlen></route>"
```
VLAN APIs

The management VLAN is configured on the WAN interface.

Table 6: VLAN APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To configure a new VLAN or modify an existing VLAN</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/bridges/bridge/wan-br/vlan</td>
</tr>
</tbody>
</table>
To get the configured VLAN info

<table>
<thead>
<tr>
<th>Method</th>
<th>Payload</th>
<th>Description</th>
</tr>
</thead>
</table>
| GET    | No      | /api/config/bridges/bridge/wan2-br/vlan  
|        |         | /api/config/bridges/bridge/user-br/vlan |

To view the operational VLAN (the VLAN that is configured for the NFVIS management traffic on the wan-br).

<table>
<thead>
<tr>
<th>Method</th>
<th>Payload</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>No</td>
<td>/api/operational/bridge-settings/bridge/wan-br/vlan</td>
</tr>
</tbody>
</table>

To delete a VLAN

<table>
<thead>
<tr>
<th>Method</th>
<th>Payload</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>No</td>
<td>/api/config/bridges/bridge/wan-br/vlan</td>
</tr>
</tbody>
</table>

Example for VLAN Payload

```xml
<vlan> <vlan-id> </vlan>
```

The valid range for VLAN is from 1 to 4094.

**Example: PUT VLAN API**

Use the PUT VLAN API to create a new VLAN or modify an existing VLAN. When you modify a VLAN, the existing VLAN ID is replaced with the modified VLAN ID.

```
curl -k -v -u admin:Cisco#123 -H Content-Type:application/vnd.yang.data+xml -k -X PUT https://192.0.2.2/api/config/bridges/bridge/wan-br/vlan -d "<vlan>120</vlan>"
```

Example: PUT VLAN API
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Feb 15 23:33:39 2017 GMT
  * expire date: Feb 13 23:33:39 2027 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'

> PUT /api/config/system/settings/wan/vlan HTTP/1.1
> Host: 192.0.2.2
> Authorization: Basic YWRtaW46Q2lzY28jMTIz
> User-Agent: curl/7.49.1
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 16
>
* upload completely sent off: 16 out of 16 bytes
< HTTP/1.1 204 No Content
< Server: nginx/1.10.1
< Date: Thu, 16 Feb 2017 22:24:44 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Thu, 16 Feb 2017 22:24:36 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1487-283876-32584
< Pragma: no-cache

**Example: GET VLAN API**

Use this GET API to view the configured VLAN information.

```
curl -k -v -u admin:Cisco#123 -H Accept:application/vnd.yang.data+xml -H Content-Type:application/xml -k -X
```
Example: GET VLAN API

**GET** https://192.0.2.2/api/config/bridges/bridge/wan-br/vlan
* Trying 192.0.2.2...
* Connected to 192.0.2.2 (192.0.2.2) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
* CAfile: /etc/pki/tls/certs/ca-bundle.crt
  CPath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
  * Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Feb 15 23:33:39 2017 GMT
  * expire date: Feb 15 23:33:39 2027 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
> GET /api/config/system/settings/wan/vlan HTTP/1.1
> Host: 192.0.2.2
> Authorization: Basic YWRtaW46Q2lzY28jMTIz
> User-Agent: curl/7.49.1
> Accept: application/vnd.yang.data+xml
> Content-Type: application/xml
> HTTP/1.1 200 OK
< Server: nginx/1.10.1
Use this GET API to view the operational VLAN (the VLAN that is configured for the NFVIS management traffic on the wan-br).

curl -k -v -u admin:Cisco#123 -H Accept:application/vnd.yang.data+xml -H Content-Type:application/xml -k -X GET https://192.0.2.2/api/operational/bridge-settings/wan-br/vlan

* Trying 192.0.2.2...

* Connected to 192.0.2.2 (192.0.2.2) port 443 (#0)

* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH

* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CPath: none

* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA

* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Feb 15 23:33:39 2017 GMT
* expire date: Feb 13 23:33:39 2027 GMT
* issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'

> GET /api/operational/system/settings-native/wan/vlan HTTP/1.1
> Host: 192.0.2.2
> Authorization: Basic YWRtaW46YWRtaW46MTIz
> User-Agent: curl/7.49.1
> Accept:application/vnd.yang.data+xml
> Content-Type:application/xml
>
< HTTP/1.1 200 OK
< Server: nginx/1.10.1
< Date: Thu, 16 Feb 2017 22:44:37 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
<
xmllns:system="http://www.cisco.com/nfv">
  <tag>120</tag>
</vlan>

Example: DELETE VLAN API

curl -k -v -u admin:Cisco#123 -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -k -X DELETE https://192.0.2.2/api/config/bridges/bridge/wan-br/vlan
  * Trying 192.0.2.2... (192.0.2.2) port 443 (#0)
  * Connected to 192.0.2.2 (192.0.2.2) port 443 (#0)
  * Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
  * successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  CApath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Feb 15 23:33:39 2017 GMT
  * expire date: Feb 13 23:33:39 2027 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'

> DELETE /api/config/system/settings/wan/vlan HTTP/1.1
> Host: 192.0.2.2
> Authorization: Basic YWRtaW46Q2lzY28jMTIz
> User-Agent: curl/7.49.1
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
>
> HTTP/1.1 204 No Content
> Server: nginx/1.10.1
> Date: Thu, 16 Feb 2017 22:48:59 GMT
> Content-Type: text/html
> Content-Length: 0
> Connection: keep-alive
> Last-Modified: Thu, 16 Feb 2017 22:48:50 GMT
### User Management APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a user</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/rbac/authentication/users</td>
</tr>
<tr>
<td>Modify a user</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/rbac/authentication/users/</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/user/&lt;user-name&gt;/change-password</td>
</tr>
<tr>
<td>Change the user role</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/rbac/authentication/users/</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/user/oper/change-role</td>
</tr>
<tr>
<td>Get all users</td>
<td>GET</td>
<td>No</td>
<td>/api/config/rbac/authentication/users/user?deep</td>
</tr>
<tr>
<td>Delete a user</td>
<td>Delete</td>
<td>Yes</td>
<td>/api/config/rbac/authentication/users/user</td>
</tr>
<tr>
<td>Configure the minimum password length</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/rbac/authentication/</td>
</tr>
<tr>
<td>Configure the password lifetime</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/rbac/authentication/password-lifetime/</td>
</tr>
<tr>
<td>Configure the account inactivity period</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/rbac/authentication/account-inactivity/</td>
</tr>
<tr>
<td>Activate an inactive user account</td>
<td>POST</td>
<td>No</td>
<td>/api/operations/rbac/authentication/users/user/username/activate</td>
</tr>
</tbody>
</table>

#### Example for Add User Payload

```xml
<user>
  <name>testuser</name>
  <role>administrator</role>
  <password>Test123#</password>
</user>
```

#### Example for Change Role Payload

```xml
<input>
  <old-role>auditors</old-role>
  <new-role>operators</new-role>
</input>
```
Example for Change Password Payload

```html
<input>
<old-password>Hello123#</old-password>
<new-password>Hello123$</new-password>
<confirm-password>Hello123$</confirm-password>
</input>
```

Example for Minimum Password Length Payload

```html
<min-pwd-length>9</min-pwd-length>
```

Example for Password Lifetime Payload

```html
<enforce>true</enforce>
<min-days>7</min-days>
<max-days>30</max-days>
```

Example for Account Inactivity Period Payload

```html
<enforce>true</enforce>
<inactivity-days>50</inactivity-days>
```

Table 7: User Management API Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the user</td>
<td>No</td>
</tr>
<tr>
<td>role</td>
<td>String</td>
<td>Role of the user</td>
<td>Yes</td>
</tr>
<tr>
<td>password</td>
<td>String</td>
<td>Password of the user</td>
<td>Yes</td>
</tr>
<tr>
<td>old-role</td>
<td>String</td>
<td>Existing role of the user</td>
<td>Yes</td>
</tr>
<tr>
<td>new-role</td>
<td>String</td>
<td>New role of the user</td>
<td>Yes</td>
</tr>
<tr>
<td>old-password</td>
<td>String</td>
<td>Existing password</td>
<td>Yes</td>
</tr>
<tr>
<td>new-password</td>
<td>String</td>
<td>New password for the user</td>
<td>Yes</td>
</tr>
<tr>
<td>confirm-password</td>
<td>String</td>
<td>Confirms the new password</td>
<td>Yes</td>
</tr>
<tr>
<td>min-pwd-length</td>
<td>Number</td>
<td>Minimum length required for passwords of all users. The minimum length must be between 7 to 128 characters.</td>
<td>Yes</td>
</tr>
<tr>
<td>enforce</td>
<td>String</td>
<td>Enforces or removes the rule. Valid values for this parameter are true and false.</td>
<td>Yes</td>
</tr>
<tr>
<td>min-days</td>
<td>Number</td>
<td>Number of days after which the users can change the password.</td>
<td>Yes</td>
</tr>
<tr>
<td>max-days</td>
<td>Number</td>
<td>Number of days before which the users must change the password.</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Example: POST Add User API

curl -X POST -v -k -u admin:Admin123$
https://209.165.201.1/api/config/rbac/authentication/users/ -H
Content-Type:application/vnd.yang.data+xml
-d
"<user><name>testuser12</name><role>administrators</role><password>Hello123#</password></user>"

Example: POST Change Role API

curl -X POST -v -k -u admin:Cisco123#
https://209.165.201.1/api/operations/rbac/authentication/users/user/oper/change-role
-H Content-Type:application/vnd.yang.data+xml
-d
"<input><old-role>auditors</old-role><new-role>operators</new-role></input>"
Example: POST Change Password API

curl -X POST -v -k -u admin:Admin123# 
https://209.165.201.1/api/operations/rbac/authentication/users/user/testuser12/change-password

Example: GET Users API

curl -X GET -v -k -u "admin:Admin123#" -H "Content-Type: application/vnd.yang.collection+xml"
-H "Accept: application/vnd.yang.collection+xml"
"https://209.165.201.1/api/config/rbac/authentication/users/user?deep"

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
Example: Delete User API

curl -X DELETE -v -k -u admin:Admin123# https://209.165.201.1/api/config/rbac/authentication/users/user/testuser12 -H Content-Type:application/vnd.yang.data+xml

* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> DELETE /api/config/rbac/authentication/users/user/testuser12 HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjM=
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type:application/vnd.yang.data+xml
>
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Thu, 22 Dec 2016 19:07:04 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Thu, 22 Dec 2016 19:07:04 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1482-433624-331310
<Pragma: no-cache

Example: Delete User API

Example: Delete User API
Example: POST Configure Minimum Password Length

curl -X POST -v -k -u admin:Admin123# https://209.165.201.1/api/config/rbac/authentication/
-H Content-Type:application/vnd.yang.data+xml -d "<min-pwd-length>9</min-pwd-length>"

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: nfvis
* Server auth using Basic with user 'admin'
> POST /api/config/rbac/authentication/ HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46QWRtaW4jMTIz
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 34
>
* upload completely sent off: 34 out of 34 bytes
< HTTP/1.1 204 No Content
< Server: nginx
< Date: Tue, 31 Oct 2017 11:56:36 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 209.165.201.1 left intact

Examples: POST Configure Password Lifetime

curl -X POST -v -k -u admin:Admin#123 https://209.165.201.1/api/config/rbac/authentication/password-lifetime/ -H
Content-Type:application/vnd.yang.data+xml -d "<enforce>true</enforce>"

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: nfvis
* Server auth using Basic with user 'admin'
> POST /api/config/rbac/authentication/password-lifetime/ HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46QWRtaW4jMTIz
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 23
>
* upload completely sent off: 23 out of 23 bytes
< HTTP/1.1 204 No Content
< Server: nginx
< Date: Tue, 31 Oct 2017 11:59:48 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 209.165.201.1 left intact
Examples: POST Configure Account Inactivity Period

```
curl -X POST -v -k -u admin:Admin#123
https://209.165.201.1/api/config/rbac/authentication/account-inactivity/ -H
Content-Type:application/vnd.yang.data+xml -d "<enforce>true</enforce>"
```

```
  * Trying 209.165.201.1...
  * Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
  * TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  * Server certificate: nfvis
  * Server auth using Basic with user 'admin'
  > POST /api/config/rbac/authentication/account-inactivity/ HTTP/1.1
  > Host: 209.165.201.1
  > Authorization: Basic YWRtaW46QWRtaW4jMTIz
  > User-Agent: curl/7.43.0
  > Accept: */*
  > Content-Type:application/vnd.yang.data+xml
  > Content-Length: 23
  >
  * upload completely sent off: 23 out of 23 bytes
  < HTTP/1.1 204 No Content
  < Server: nginx
  < Date: Tue, 31 Oct 2017 11:59:48 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 209.165.201.1 left intact
```
* Server certificate: nfvis
* Server auth using Basic with user 'admin'
> POST /api/config/rbac/authentication/account-inactivity/ HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46QWRtaW4jMTIz
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 23
>
* upload completely sent off: 23 out of 23 bytes
< HTTP/1.1 204 No Content
< Server: nginx
< Date: Tue, 31 Oct 2017 12:00:52 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 209.165.201.1 left intact

curl -X POST -v -k -u admin:Admin#123 https://209.165.201.1/api/config/rbac/authentication/account-inactivity/ -H Content-Type:application/vnd.yang.data+xml -d "<inactivity-days>50</inactivity-days>"
* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: nfvis
* Server auth using Basic with user 'admin'
> POST /api/config/rbac/authentication/account-inactivity/ HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46QWRtaW4jMTIz
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 23
>
* upload completely sent off: 23 out of 23 bytes
< HTTP/1.1 204 No Content
< Server: nginx
< Date: Tue, 31 Oct 2017 12:00:52 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 209.165.201.1 left intact

Example: POST Activate an Inactive User Account

curl -X POST -v -k -u admin:Admin#123 https://209.165.201.1/api/operations/rbac/authentication/users/user/guest_user/activate -H Content-Type:application/vnd.yang.data+xml
* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: nfvis
* Server auth using Basic with user 'admin'
> POST /api/operations/rbac/authentication/users/user/guest_user/activate HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46QWRtaW4jMTIz

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
TACACS+ Server APIs

Table 8: TACACS+ Server APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To configure a TACACS+ server</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/security_servers/tacacs-server</td>
</tr>
<tr>
<td>To modify a TACACS+ server</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/security_servers/tacacs-server</td>
</tr>
<tr>
<td>To get the TACACS+ server</td>
<td>GET</td>
<td>No</td>
<td>/api/config/security_servers/tacacs-server?deep</td>
</tr>
<tr>
<td>To delete a TACACS+ server</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/security_servers/tacacs-server/host/&lt;ip-address/domain-name&gt;</td>
</tr>
</tbody>
</table>

Example for TACACS+ Server Payload

Table 9: TACACS+ Server Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
</table>

Example: POST TACACS Server API


* Hostname was NOT found in DNS cache
* Trying 172.19.181.173...
* Connected to 172.19.181.173 (172.19.181.173) port 443 (#0)
* successfully set certificate verify locations:
*  CAfile: none
Example: GET TACACS Server API

curl -k -v -u "admin:cisco123" -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+json -X GET https://209.165.201.1/api/config/security_servers/tacacs-server?deep

* Hostname was NOT found in DNS cache
  * Trying 209.165.201.1...
  * Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
  * successfully set certificate verify locations:
  * CAfile: none
  * CACert: /etc/ssl/certs
* SSLv3, TLS handshake, Client hello (1):
* SSLv3, TLS handshake, Server hello (2):
* SSLv3, TLS handshake, CERT (11):
  * SSLv3, TLS handshake, Server key exchange (12):
  * SSLv3, TLS handshake, Server finished (14):
* SSLv3, TLS handshake, Client key exchange (16):
* SSLv3, TLS change cipher, Client hello (1):
  * SSLv3, TLS handshake, Finished (20):
  * SSLv3, TLS change cipher, Client hello (1):
  * SSLv3, TLS handshake, Finished (20):
  * SSLv3, TLS change cipher, Client hello (1):
  * SSLv3, TLS handshake, Finished (20):
  * SSLv3, TLS change cipher, Client hello (1):
  * SSLv3, TLS handshake, Finished (20):
  * SSLv3, TLS change cipher, Client hello (1):
  * SSLv3, TLS handshake, Finished (20):
Example: PUT TACACS Server API

* Hostname was NOT found in DNS cache
* Trying 172.19.181.173...
* Connected to 172.19.181.173 (172.19.181.173) port 443 (0)
* successfully set certificate verify locations:
  * CAfile: none
Example: DELETE TACACS Server API

```
curl -k -v -u "admin:cisco123" -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+json -X DELETE https://192.0.2.2/api/config/security_servers/tacacs-server/host/5.5.5.5
```

**CApath:** /etc/ssl/certs

* SSLv3, TLS handshake, Client hello (1):  
* SSLv3, TLS handshake, Server hello (2):  
* SSLv3, TLS handshake, CERT (11):  
* SSLv3, TLS handshake, Server key exchange (12):  
* SSLv3, TLS handshake, Server finished (14):  
* SSLv3, TLS handshake, Client key exchange (16):  
* SSLv3, TLS change cipher, Client hello (1):  
* SSLv3, TLS handshake, Finished (20):  
* SSLv3, TLS change cipher, Client hello (1):  
* SSLv3, TLS handshake, Finished (20):  
* SSL connection using ECDHE-RSA-AES256-GCM-SHA384

**Server certificate:**  
* subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate  
* start date: 2017-01-13 23:47:41 GMT  
* expire date: 2027-01-11 23:47:41 GMT  
* issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate  
* SSL certificate verify result: self signed certificate (18), continuing anyway.

* Server auth using Basic with user 'admin'

> PUT /api/config/security_servers/tacacs-server/host/5.5.5.5 HTTP/1.1  
> Authorization: Basic YWRtaW46Y2lzY28xMjM=  
> User-Agent: curl/7.35.0  
> Host: 172.19.181.173  
> Accept:application/vnd.yang.data+xml  
> Content-Type:application/vnd.yang.data+json  
> Content-Length: 92  
>  
> * upload completely sent off: 92 out of 92 bytes  
< HTTP/1.1 204 No Content  
* Server nginx/1.10.1 is not blacklisted  
< Server: nginx/1.10.1  
< Date: Mon, 27 Feb 2017 18:20:13 GMT  
< Content-Type: text/html  
< Content-Length: 0  
< Connection: keep-alive  
< Last-Modified: Mon, 27 Feb 2017 18:20:13 GMT  
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate  
< Etag: 1488-219613-571277  
< Pragma: no-cache
Trusted IP Connection APIs

### Table 10: Trusted IP Connection APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To add, modify, or remove the trusted source IP connection</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/system/settings</td>
</tr>
<tr>
<td>To verify the configuration of the trusted source IP addresses</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/settings-native/trusted-source</td>
</tr>
<tr>
<td>To verify the system settings</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/settings-native?deep</td>
</tr>
<tr>
<td>To verify the trusted source or system settings</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/settings?deep</td>
</tr>
</tbody>
</table>

#### Example for the Trusted IP Connection Payload

```xml
<settings>
  <hostname>nfvis</hostname>
  <trusted-source>192.0.2.0/24</trusted-source>
  <mgmt>
    <ip>
      <address>198.51.100.1</address>
      <netmask>255.255.255.0</netmask>
    </ip>
  </mgmt>
  <wan>
    <ip>
```
Table 11: Trusted IP Connection Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostname</td>
<td>String</td>
<td>Hostname of the system</td>
<td>Yes</td>
</tr>
<tr>
<td>trusted-source</td>
<td>String</td>
<td>Source IP address You can specify a single IP address or a range of IP addresses.</td>
<td>No</td>
</tr>
<tr>
<td>mgmt ip address</td>
<td>String</td>
<td>Specifies the management IP address and netmask.</td>
<td>Yes</td>
</tr>
<tr>
<td>wan ip address</td>
<td>String</td>
<td>Specifies the WAN IP address and netmask.</td>
<td>Yes</td>
</tr>
<tr>
<td>default-gw</td>
<td>String</td>
<td>IP address of the default gateway</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example: PUT Trusted IP Connection API

Use this API to add, modify, or remove the trusted source IP address or addresses.

To delete all trusted source IP addresses, you need to remove the trusted source element (trusted-source) from the payload. You can modify a trusted source IP address by replacing it with a new IP address.

curl -k -v -u "admin:Cisco123#" -H "Content-Type:application/vnd.yang.data+xml" -X PUT https://198.51.100.1/api/config/system/settings -d ""<settings><hostname>nfvis</hostname><trusted-source>192.0.2.0/24</trusted-source><mgmt><ip><address>198.51.100.1</address><netmask>255.255.255.0</netmask></ip></mgmt><wan><ip><address>198.51.100.2</address><netmask>255.255.255.0</netmask></ip></wan><default-gw>198.51.100.3</default-gw></settings>"

* Trying 198.51.100.1...
* Connected to 198.51.100.1 (198.51.100.1) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP/1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
Example: GET Trusted IP Connection API

curl -v -k -u admin:Cisco123# -X GET
'https://198.51.100.1/api/operational/system/settings-native/trusted-source'

Note: Unnecessary use of -X or --request, GET is already inferred.
* Trying 198.51.100.1...
* Connected to 198.51.100.1 (198.51.100.1) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* Connection #0 to host 198.51.100.1 left intact
* SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'
> GET /api/operational/system/settings-native/trusted-source HTTP/1.1
  > Host: 198.51.100.1
  > Authorization: Basic YWRtaW46Q2lzY28xMjMj
  > User-Agent: curl/7.50.1
  > Accept: */*
  >
  < HTTP/1.1 200 OK
  < Server: nginx/1.10.1
  < Date: Tue, 14 Mar 2017 21:08:49 GMT
  < Content-Type: application/vnd.yang.collection+xml
  < Transfer-Encoding: chunked
  < Connection: keep-alive
  < Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
  < Pragma: no-cache
  <
  <collection xmlns:y="http://tail-f.com/ns/rest">
    <trusted-source xmlns="http://www.cisco.com/nfv">192.0.2.0/24</trusted-source>
  </collection>
* Connection #0 to host 198.51.100.1 left intact

## Banner and Message APIs

### Table 12: Banner and Message APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To configure or update a banner or message of the day or both</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/banner-motd</td>
</tr>
<tr>
<td>To get system banner details and user-defined banner and message of the day</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/banner-motd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/operational/banner-motd/system-banner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/operational/banner-motd/banner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/operational/banner-motd/motd</td>
</tr>
<tr>
<td>To get user-defined banner and message of the day details</td>
<td>GET</td>
<td>No</td>
<td>/api/config/banner-motd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/config/banner-motd/banner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/config/banner-motd/motd</td>
</tr>
<tr>
<td>To delete the user-defined banner or message of the day</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/banner-motd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/config/banner-motd/banner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/config/banner-motd/motd</td>
</tr>
</tbody>
</table>

### Example for Banner and Message Payload

```xml
<banner-motd>
  <banner> my banner </banner>
</banner-motd>`
Example: PUT Banner-MOTD API

curl -k -u "admin:Cisco123*" -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X PUT https://209.165.201.1/api/config/banner-motd -d '<banner-motd><banner>my banner</banner><motd>my motd</motd></banner-motd>'

Table 13: Banner and Message Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>banner</td>
<td>String</td>
<td>Specifies the user-defined banner.</td>
<td>No</td>
</tr>
<tr>
<td>motd</td>
<td>String</td>
<td>Message of the day</td>
<td>No</td>
</tr>
</tbody>
</table>

Example: GET Banner-MOTD API

Use this operational API to get information about the system-defined banner.

curl -k -u "admin:Cisco123*" -X GET "https://209.165.201.1/api/operational/banner-motd/system-banner"

Example: PUT Banner-MOTD API
Example: GET Banner-MOTD API

Use this GET API to get information about the user-defined banner and message of the day.

curl -k -v -u "admin:Cisco123*" -X GET "https://209.165.201.1/api/config/banner-motd"

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
Example: DELETE Banner-MOTD API

Use this DELETE API to delete the user-defined banner.

curl -k -v -u "admin:Cisco123*" -X DELETE 
"https://209.165.201.1/api/config/banner-motd/banner"
* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> DELETE /api/config/banner-motd/banner HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMq
> User-Agent: curl/7.43.0
> Accept: */*
>
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Wed, 08 Feb 2017 20:27:29 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Wed, 08 Feb 2017 20:27:29 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1486-585649-542089
< Pragma: no-cache

Use this DELETE API to delete the user-defined message of the day.

curl -k -v -u "admin:Cisco123*" -X DELETE 
"https://209.165.201.1/api/config/banner-motd/motd"
* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> DELETE /api/config/banner-motd/motd HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMq
> User-Agent: curl/7.43.0
> Accept: */*
>
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Wed, 08 Feb 2017 20:33:52 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Wed, 08 Feb 2017 20:33:52 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1486-586032-109043
< Pragma: no-cache
<
After deleting the banner or message of the day, you can run the GET operational API to confirm the deletion. If you use the parameter "banner" or "motd" along with the GET API, you get a 404 error if the deletion is successful. If you run the GET API without the parameter (/api/operational/banner-motd), you get the output with empty "banner-motd" tag, if the deletion is successful.

Disk Space APIs

Table 14: Disk Space API

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get the information on disk space</td>
<td>GET</td>
<td>Yes</td>
<td>/api/operational/system/disk-space</td>
</tr>
</tbody>
</table>

Example: GET Disk Space API

```bash
curl -k -v -u "admin:admin" -X GET "https://209.165.201.1/api/operational/system/disk-space?deep"
```

Note: Unnecessary use of -X or --request, GET is already inferred.

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CAPath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP/1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
  * TLSv1.2 (IN), TLS change cipher, Client hello (1):
  * TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* Server certificate:
  * subject: CN=nfv
  * start date: Oct 23 17:25:04 2018 GMT
  * expire date: Oct 22 17:25:04 2023 GMT
  * issuer: CN=nfv
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
> GET /api/operational/system/disk-space?deep HTTP/1.1
> Host: 172.25.221.106
Example: GET Disk Space API

  <disk-info>
    <disk-name>lv_data</disk-name>
    <associated-physical-disk>sde2</associated-physical-disk>
    <total-size>41G</total-size>
    <size-used>8.6G</size-used>
    <size-available>32G</size-available>
    <use-percent>22%</use-percent>
  </disk-info>
  <disk-info>
    <disk-name>lv_var</disk-name>
    <associated-physical-disk>sde2</associated-physical-disk>
    <total-size>2.0G</total-size>
    <size-used>118M</size-used>
    <size-available>1.7G</size-available>
    <use-percent>7%</use-percent>
  </disk-info>
  <disk-info>
    <disk-name>lv_root</disk-name>
    <associated-physical-disk>sde2</associated-physical-disk>
    <total-size>7.8G</total-size>
    <size-used>1.8G</size-used>
    <size-available>5.7G</size-available>
    <use-percent>24%</use-percent>
  </disk-info>
  <disk-info>
    <disk-name>extdatastore2</disk-name>
    <associated-physical-disk>sdd</associated-physical-disk>
    <total-size>1.8T</total-size>
    <size-used>77M</size-used>
    <size-available>1.7T</size-available>
    <use-percent>1%</use-percent>
  </disk-info>
</disk-space>
System Time APIs

Table 15: System Time APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To set the manual time</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/system/time/set-manual-time</td>
</tr>
<tr>
<td>To configure the preferred and backup servers</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/system/time/ntp/preferred_server, /api/config/system/time/ntp/backup_server</td>
</tr>
<tr>
<td>To set the timezone</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/system/time/timezone</td>
</tr>
<tr>
<td>To get the system time information</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/time</td>
</tr>
<tr>
<td>To add NTP IPv6 server</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/system/time/</td>
</tr>
<tr>
<td>To delete NTP IPv6 server</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/system/time/ntp-ipv6/</td>
</tr>
<tr>
<td>To get time status</td>
<td>GET</td>
<td>NO</td>
<td>/api/operational/system/time</td>
</tr>
</tbody>
</table>

Example for System Time API Payload

```xml
<time>2017-01-01T00:00:00</time>
<preferred_server><ip-address></preferred_server>
<backup_server><ip-address></backup_server>
<timezone><zone/subzone></timezone>
```

Table 16: System Time API Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>preferred_server</td>
<td>String</td>
<td>Preferred server IP address or domain name.</td>
<td>Yes</td>
</tr>
<tr>
<td>backup_server</td>
<td>String</td>
<td>Backup server IP address or domain name.</td>
<td>No</td>
</tr>
<tr>
<td>timezone</td>
<td>String</td>
<td>Specifies the timezone.</td>
<td>No</td>
</tr>
<tr>
<td>ntp-server</td>
<td>String</td>
<td>Specifies the IPv6 address or domain name.</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Example: PUT System Time Manual Time API

curl -v -k -u admin:Cisco123* -H "Content-Type: application/vnd.yang.data+xml" -X
PUT https://209.165.201.1/api/config/system/time/set-manual-time -d
  '<input><time>2017-01-01T00:00:00</time></input>'

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> PUT /api/config/system/time/set-manual-time HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMq
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 46
> * upload completely sent off: 46 out of 46 bytes
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Wed, 01 Jan 2020 11:11:51 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Wed, 30 Nov 2016 04:10:28 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1480-479028-836845
< Pragma: no-cache
<

Example: PUT System Time Preferred Server API

curl -v -k -u admin:Cisco123* -H "Content-Type: application/vnd.yang.data+xml" -X
PUT https://209.165.201.1/api/config/system/time/ntp/preferred_server -d
  '<preferred_server>209.165.201.2</preferred_server>'
Example: PUT System Time Backup Server API

curl -v -k -u admin:Cisco123* -H "Content-Type: application/vnd.yang.data+xml" -X PUT https://209.165.201.1/api/config/system/time/ntp/backup_server -d "<backup_server>209.165.201.4</backup_server>"
Example: PUT System Time Timezone API

curl -v -k -u admin:Cisco123 -H "Content-Type: application/vnd.yang.data+xml" -X PUT https://209.165.201.1/api/config/system/time/timezone -d '<timezone>America/New_York</timezone>'

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> PUT /api/config/system/time/timezone HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMq
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/vnd.yang.data+xml
> Content-Length: 43
>
* upload completely sent off: 43 out of 43 bytes
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Wed, 01 Jan 2020 11:16:47 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Wed, 01 Jan 2020 11:16:47 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1480-479368-378871
< Pragma: no-cache

Example: PUT System Time Timezone API
Example: GET System Time API

curl -v -k -u admin:Cisco123* -H "Content-Type: application/vnd.yang.data+xml" -X GET https://209.165.201.1/api/operational/system/time?deep

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> GET /api/operational/system/host_time HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMq
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/vnd.yang.data+xml
> 
< HTTP/1.1 200 OK
< Server: nginx/1.6.3
< Date: Wed, 01 Jan 2020 11:21:13 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

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
Platform Details API

Table 17: Platform Details APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get information about the hardware</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/platform-detail</td>
</tr>
</tbody>
</table>

Sample Output for the Platform Details API

curl -k -v -u admin:Cisco123# -X GET 'https://172.19.162.209/api/operational/platform-detail'

Note: Unnecessary use of -X or --request, GET is already inferred.

* Trying 172.19.162.209...
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* Successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP/1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* Server certificate:
  * subject: CN=nfv
  * start date: Aug 17 11:21:43 2017 GMT
  * expire date: Aug 15 11:21:43 2027 GMT
* issue: CN=nfv
* SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'

GET /api/operational/platform-detail HTTP/1.1
Host: 172.19.162.209
Authorization: Basic YWRtaW46Q2lzY28xMjMj
User-Agent: curl/7.50.1
Accept: */*

HTTP/1.1 200 OK
Server: nginx
Date: Fri, 18 Aug 2017 13:21:47 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

<platform-detail
xmlns="http://www.cisco.com/nfvos/platform-info"
xmns:y="http://tail-f.com/ns/rest"
xmns:platform_info="http://www.cisco.com/nfvos/platform-info">
<hardware_info>
<Manufacturer>Cisco Systems Inc</Manufacturer>
<PID>UCSC-C220-M4S</PID>
<SN>FCH1924V2AH</SN>
<hardware-version>74-12419-01</hardware-version>
<UUID>663F3347-5499-0D49-A76E-533A4AA9C755</UUID>
<Version>3.6.0-916</Version>
<Compile_Time>Monday, August 07, 2017 [01:30:11 PDT]</Compile_Time>
<CPU_Information>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz 8 cores</CPU_Information>
<Memory_Information>65701956 kB</Memory_Information>
<Disk_Size>1000.2 GB</Disk_Size>

<software_packages>
<Kernel_Version>3.10.0-514.10.2.el7.x86_64</Kernel_Version>
<QEMU_Version>1.5.3</QEMU_Version>
<LibVirt_Version>2.0.0</LibVirt_Version>
<OVS_Version>2.3.2</OVS_Version>
</software_packages>

<port_detail>
<Name>eth0</Name>
</port_detail>
<port_detail>
<Name>eth1</Name>
</port_detail>
<port_detail>
<Name>eth2</Name>
</port_detail>
<port_detail>
<Name>eth3</Name>
</port_detail>
<port_detail>
<Name>eth4</Name>
</port_detail>
<port_detail>
<Name>eth5</Name>
</port_detail>
</switch_detail>
<UUID>NA</UUID>
>Type>NA</Type>
<Name>NA</Name>
Port Details APIs

Table 18: Port Details APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get information about the physical port</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/platform-detail/port_detail</td>
</tr>
</tbody>
</table>

Sample Output for the Port Details API

curl -k -v -u admin:Cisco123# -X GET 'https://172.19.162.209/api/operational/platform-detail/port_detail'

Note: Unnecessary use of -X or --request, GET is already inferred.

* Trying 172.19.162.209...
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP/1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* Server certificate:
  * subject: CN=nfv
  * start date: Aug 17 11:21:43 2017 GMT
  * expire date: Aug 15 11:21:43 2027 GMT
  * issuer: CN=nfv
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'

> GET /api/operational/platform-detail/port_detail HTTP/1.1
> Host: 172.19.162.209
> Authorization: Basic YWRtaW46QzE5Q2JhMjQzNDk2MjI=
> User-Agent: curl/7.50.1
> Accept: */*
> < HTTP/1.1 200 OK
> Server: nginx
> < Date: Fri, 18 Aug 2017 13:24:32 GMT
> < Content-Type: application/vnd.yang.collection+xml
> < Transfer-Encoding: chunked
<Connection: keep-alive
<Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Pragma: no-cache

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software

NFVIS Related APIs

Port Details APIs

<collection xmlns="http://tail-f.com/ns/rest">
<port_detail xmlns="http://www.cisco.com/nfvos/platform-info">
<Name>eth0</Name>
<Type>physical</Type>
<Media>Twisted Pair</Media>
<Link>up</Link>
<Speed>1000</Speed>
<MTU>1500</MTU>
<MAC>80:e0:1d:4a:8c:56</MAC>
<PCI_detail>01:00.0</PCI_detail>
</port_detail>
<port_detail xmlns="http://www.cisco.com/nfvos/platform-info">
<Name>eth1</Name>
<Type>physical</Type>
<Media>Twisted Pair</Media>
<Link>up</Link>
<Speed>1000</Speed>
<MTU>1500</MTU>
<MAC>80:e0:1d:4a:8c:57</MAC>
<PCI_detail>01:00.1</PCI_detail>
</port_detail>
<port_detail xmlns="http://www.cisco.com/nfvos/platform-info">
<Name>eth2</Name>
<Type>physical</Type>
<Media>Twisted Pair</Media>
<Link>down</Link>
<Speed>0</Speed>
<MTU>1500</MTU>
<MAC>80:e0:1d:37:0f:28</MAC>
<PCI_detail>04:00.0</PCI_detail>
</port_detail>
<port_detail xmlns="http://www.cisco.com/nfvos/platform-info">
<Name>eth3</Name>
<Type>physical</Type>
<Media>Twisted Pair</Media>
<Link>down</Link>
<Speed>0</Speed>
<MTU>1500</MTU>
<MAC>80:e0:1d:37:0f:29</MAC>
<PCI_detail>04:00.1</PCI_detail>
</port_detail>
<port_detail xmlns="http://www.cisco.com/nfvos/platform-info">
<Name>eth4</Name>
<Type>physical</Type>
<Media>Twisted Pair</Media>
<Link>down</Link>
<Speed>0</Speed>
<MTU>1500</MTU>
<MAC>80:e0:1d:37:0f:2a</MAC>
<PCI_detail>04:00.2</PCI_detail>
</port_detail>
Portal Access APIs

Table 19: Portal Access APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To enable or disable the portal access</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/system/portal</td>
</tr>
<tr>
<td>To get the portal access status</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/portal/status</td>
</tr>
</tbody>
</table>

Example for a Portal Access Payload

```xml
<portal>
  <access>enabled</access>
</portal>
```

Table 20: Portal Access Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>access</td>
<td>String</td>
<td>Specify the portal access as &quot;enabled&quot; or &quot;disabled&quot;.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example: PUT Portal Access (Enable/Disable)

curl -v -k -u "admin:Cisco123#" -H "Content-Type:application/vnd.yang.data+xml" -X PUT https://209.165.201.1/api/config/system/portal -d "<portal><access>enabled</access></portal>"

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (v0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!a NULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/ssl/certs/ca-bundle.crt
  * Cpath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
Example: GET Portal Access API

curl -v -k -u admin:Cisco123# -X GET
'https://209.165.201.1/api/operational/system/portal/status'

Note: Unnecessary use of -X or --request, GET is already inferred.
* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Mar 14 06:53:22 2017 GMT
  * expire date: Mar 12 06:53:22 2027 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'
> PUT /api/config/system/portal HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.50.1
> Accept: */*
> Content-Type: application/vnd.yang.data+xml
> Content-Length: 41
>
> upload completely sent off: 41 out of 41 bytes
< HTTP/1.1 204 No Content
< Server: nginx/1.10.1
< Date: Tue, 14 Mar 2017 19:34:42 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Tue, 14 Mar 2017 19:34:42 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1489-520082-470197
< Pragma: no-cache
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Mar 14 06:53:22 2017 GMT
  * expire date: Mar 12 06:53:22 2027 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'

> GET /api/operational/system/portal/status HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.50.1
> Accept: */*
>
HTTP/1.1 200 OK
Server: nginx/1.10.1
Date: Tue, 14 Mar 2017 19:35:05 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

System Log APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To set system logs</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/system/set-log</td>
</tr>
<tr>
<td>To get the system log configuration details</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/logging-level</td>
</tr>
</tbody>
</table>

Example for System Log Payload

<input>
   
   <logtype>all</logtype>
   <level>warning</level>

</input>

Table 21: Payload Description for Setting Log Level

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>logtype</td>
<td>String</td>
<td>Type of the log. There are two types: configuration and operational. You can specify one of the following: configuration, operational, all (includes both configuarion and opeartional logs)</td>
<td>Yes</td>
</tr>
</tbody>
</table>
level | String | Indicates the log level. The supported log levels are: debug, info, warning, error, and critical. 
| --- | --- | --- |
| Yes | Note | The info and warning log levels are set by default respectively for the configuration and operational log types. You can change them as required. However, the change to the log level is not persisted across a reboot. After a reboot, the default log levels are used. 

**Example: POST System Log API**

```
curl -k -v -u admin:Cisco123# -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X POST https://209.165.201.1/api/operations/system/set-log -d '<input><logtype>all</logtype><level>warning</level></input>'
```

* Trying 209.165.201.1... 
* Connected to 209.165.201.1 (209.165.201.1) port 443 (0) 
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH 
* successfully set certificate verify locations: 
* CAfile: /etc/pki/tls/certs/ca-bundle.crt 
* CApth: none 
* TLSv1.0 (OUT), TLS handshake, Client hello (1): 
* TLSv1.0 (IN), TLS handshake, Server hello (2): 
* TLSv1.0 (IN), TLS handshake, Certificate (11): 
* TLSv1.0 (IN), TLS handshake, Server key exchange (12): 
* TLSv1.0 (IN), TLS handshake, Server finished (14): 
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16): 
* TLSv1.0 (OUT), TLS change cipher, Client hello (1): 
* TLSv1.0 (OUT), TLS handshake, Finished (20): 
* TLSv1.0 (IN), TLS change cipher, Client hello (1): 
* TLSv1.0 (IN), TLS handshake, Finished (20): 
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA 
* Server certificate: 
* subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate 
* start date: Dec 8 07:50:20 2016 GMT 
* expire date: Dec 6 07:50:20 2026 GMT 
* issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate 
* SSL certificate verify result: self signed certificate (18), continuing anyway. 
* Server auth using Basic with user 'admin' 
* POST /api/operations/system/set-log HTTP/1.1 
* Host: 209.165.201.1 
* Authorization: Basic YWRtaW46Q2lzY28xMjMj 
* User-Agent: curl/7.49.1 
* Accept:application/vnd.yang.data+xml 
* Content-Type:application/vnd.yang.data+xml 
* Content-Length: 59 
> 
> * upload completely sent off: 59 out of 59 bytes 
< HTTP/1.1 204 No Content 
< Server: nginx/1.6.3 
< Date: Thu, 05 Jan 2017 03:49:32 GMT
curl -k -v -u admin:Cisco123# -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X GET https://209.165.201.1/api/operational/system/logging-level

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Dec 8 07:50:20 2016 GMT
  * expire date: Dec 6 07:50:20 2026 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
> GET /api/operational/system/logging-level HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.49.1
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
>
< HTTP/1.1 200 OK
< Server: nginx/1.6.3
< Date: Thu, 05 Jan 2017 03:45:53 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
<
**DPDK Support APIs**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To enable DPDK and VM migration</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/system/settings/</td>
</tr>
<tr>
<td>To Disable DPDK (in error state)</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/system/settings/dpdk</td>
</tr>
<tr>
<td>To get the status of DPDK</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/settings-native/dpdk-status</td>
</tr>
</tbody>
</table>

*Table 22: Payload Description for DPDK Support*

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpdk</td>
<td>String</td>
<td>Specify enabling DPDK</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Example : POST to enable DPDK**

curl -k -v -u admin/admin -H "Accept: application/vnd.yang.data+json" -H "Content-Type: application/vnd.yang.data+json" -X POST https://localhost/api/config/system/settings/ --data '{"dpdk": "enable"}'

**Example: DELETE to disable DPDK**

curl -k -v -u admin/admin -X DELETE https://localhost/api/config/system/settings/dpdk

**Example: GET to get the status of DPDK:**

curl -k -v -u admin/admin -X GET https://localhost/api/operational/system/settings-native/dpdk-status

**Backup and Restore APIs**

**Backup APIs**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To start configuration-only backup</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/hostaction/backup/configuration-only/</td>
</tr>
<tr>
<td>To start configuration-and-vms backup</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/hostaction/backup/configuration-and-vms/</td>
</tr>
</tbody>
</table>
Table 23: Payload Description for Setting Log Level

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>file-path</td>
<td>String</td>
<td>Path representing location to the file</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example: POST to start a configuration-only backup

curl -k -v -u admin:admin -H "Accept:application/vnd.yang.data+json" -H "Content-Type:application/vnd.yang.data+json" -X POST https://localhost/api/operations/hostaction/backup/configuration-only/ --data '{"input": {"file-path": "intdatastore:sample.bkup"}}'

Example: POST to start configuration-and-vms backup:


Restore APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To start restore from a backup package</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/hostaction/restore/</td>
</tr>
</tbody>
</table>

Table 24: Payload Description for Setting Log Level

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>restore-option</td>
<td>String</td>
<td>Option to restore without connectivity settings. Accepted values: except-connectivity</td>
<td>No</td>
</tr>
</tbody>
</table>

Example: To start a restore


Example: To start a restore while preserving connectivity settings:

## Route Distribution APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To configure route distribution</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/route-distributions</td>
</tr>
<tr>
<td>To update route distribution</td>
<td>GET</td>
<td>No</td>
<td>/api/config/route-distributions?deep</td>
</tr>
<tr>
<td>configuration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To delete route distribution</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/route-distributions</td>
</tr>
<tr>
<td>configuration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To get route distribution state</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/route-distributions</td>
</tr>
<tr>
<td>data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Example for route distribution payload

```xml
<route-distribute>
  <neighbor-address>172.25.221.106</neighbor-address>
  <local-bridge>wan-br</local-bridge>
  <remote-as>65000</remote-as>
  <network-subnet>
    <subnet>10.20.0.0/24</subnet>
  </network-subnet>
</route-distribute>
```

### Table 25: Payload Description for Route Distribution

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>neighbor-address</td>
<td>String</td>
<td>Neighbor IPv4 address secure overlay connection.</td>
<td>Yes</td>
</tr>
<tr>
<td>local-address</td>
<td>String</td>
<td>Local IPv4 address</td>
<td>No</td>
</tr>
<tr>
<td>local-bridge</td>
<td>String</td>
<td>Local bridge name for overlay (default wan-br)</td>
<td>No</td>
</tr>
<tr>
<td>local-as</td>
<td>String</td>
<td>Local autonomous system number</td>
<td>Yes</td>
</tr>
<tr>
<td>remote-as</td>
<td>String</td>
<td>Remote autonomous system number</td>
<td>Yes</td>
</tr>
<tr>
<td>router-id</td>
<td>String</td>
<td>Local router id IP address</td>
<td>No</td>
</tr>
<tr>
<td>network-subnet</td>
<td>String</td>
<td>List of subnets to be announced. H.H.H.H/N (atleast one subnet needs to be announced)</td>
<td>Yes</td>
</tr>
<tr>
<td>next-hop</td>
<td>String</td>
<td>IPv4 address of any local interface</td>
<td>No</td>
</tr>
</tbody>
</table>
Example: POST create route distribution

```bash
curl -k -v -u "admin:admin" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X POST https://209.165.201.1/api/config/route-distributions -d '<route-distribute><neighbor-address>172.25.221.106</neighbor-address><local-bridge>wan-br</local-bridge><local-as>65000</local-as><remote-as>65000</remote-as><network-subnet><subnet>10.20.0.0/24</subnet></network-subnet></route-distribute>'
```

Example: GET update route distribution

```bash
curl -k -v -u "admin:admin" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X PUT https://209.165.201.1/api/config/rout-distributions/route-distribute/172.25.221.106 -d '<route-distribute><neighbor-address>172.25.221.106</neighbor-address><local-bridge>wan-br</local-bridge><local-as>65000</local-as><remote-as>65000</remote-as><network-subnet><subnet>10.20.0.0/24</subnet></network-subnet></route-distribute>'
```

Example: GET route distributions state information

```bash
curl -k -v -u "admin:admin" -X GET "https://209.165.201.1/api/operational/route-distributions?deep"
```

Example: DELETE all route distributions

```bash
curl -k -v -u "admin:admin" -X DELETE "https://209.165.201.1/api/config/route-distributions"
```

### Dynamic SR-IOV APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To enable SR-IOV</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/pnics/pnic/eth0-1/sriov/numvfs</td>
</tr>
<tr>
<td>To set switchmode</td>
<td>PUT</td>
<td>No</td>
<td>/api/config/pnics/pnic/eth0-1/sriov/switchmode</td>
</tr>
<tr>
<td>To disable SR-IOV</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/pnics/pnic/eth0-1/sriov</td>
</tr>
<tr>
<td>To get SR-IOV operational data</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/pnics/pnic/eth0-1/sriov</td>
</tr>
<tr>
<td>To create SR-IOV network with trunk mode</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/networks</td>
</tr>
<tr>
<td>To create SR-IOV network with access mode</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/networks</td>
</tr>
<tr>
<td>To delete SR-IOV network</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/networks/network/eth0-1-SRIOV-1</td>
</tr>
</tbody>
</table>

**Example: PUT enable SR-IOV**

```bash
curl -k -v -u admin:admin -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X PUT https://209.165.201.1/api/config/pnics/pnic/eth0-1/sriov/numvfs -d '<numvfs>1</numvfs>'
```

**Example: DELETE disable SR-IOV**
curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X DELETE 
https://209.165.201.1/api/config/pnics/pnic/eth0-1/sriov

**Example:** GET SR-IOV operational data

curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X GET 
https://209.165.201.1/api/operational/pnics/pnic/eth0-1/sriov

**Example:** POST create SR-IOV network with trunk mode

curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X POST https://209.165.201.1/api/config/networks --data '<network><name>eth0-1-SRIOV-1</name><sriov>true</sriov><trunk>true</trunk><vlan>30</vlan></network>'

**Example:** POST create SR-IOV network with access mode

curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X POST https://209.165.201.1/api/config/networks --data '<network><name>eth0-1-SRIOV-1</name><sriov>true</sriov><trunk>false</trunk><vlan>30</vlan></network>'

**Example:** DELETE SR-IOV network

curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X DELETE 
https://209.165.201.1/api/config/networks/network/eth0-1-SRIOV-1
PnP APIs

- Certificate Creation APIs, on page 61
- PnP Action APIs, on page 65
- PnP APIs, on page 66
- PnP Server APIs, on page 66

Certificate Creation APIs

Table 26: Certificate Creation APIs

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

Example for Signing Request Payload

```xml
<signing-request>
  <country-code>US</country-code>
</signing-request>
```
Table 27: Description for Signing Request Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;country-code&gt;</td>
<td>String</td>
<td>Two-letter ISO abbreviation for your country.</td>
<td>No</td>
</tr>
<tr>
<td>&lt;state&gt;</td>
<td>String</td>
<td>Name of the state where your organization's head office is located.</td>
<td>No</td>
</tr>
<tr>
<td>&lt;locality&gt;</td>
<td>Boolean</td>
<td>Name of the city where your organization's head office is located.</td>
<td>No</td>
</tr>
<tr>
<td>&lt;organization&gt;</td>
<td>Boolean</td>
<td>Name of the organization</td>
<td>No</td>
</tr>
<tr>
<td>&lt;organization-unit-name&gt;</td>
<td>String</td>
<td>Name of the department or group that will use the certificate.</td>
<td>No</td>
</tr>
<tr>
<td>&lt;common-name&gt;</td>
<td>URL</td>
<td>Fully qualified domain name that you want to secure.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example for Install Certificate Payload

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

Table 28: Description for Install Certificate Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;install-cert&gt;</td>
<td>URL</td>
<td>Full path of the certificate.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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 29: Description for Use Certificate Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
</table>
Example: POST Signing Request API

```
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://209.165.201.1/api/operations/system/certificate/signing-request
```

Example: POST Install Certificate API

```
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://209.165.201.1/api/operations/system/certificate/install-cert
```
Example: POST Use Certificate API

```
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://209.165.201.1/api/operations/system/certificate/use-cert
```

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* Connected to 209.165.201.1 (209.165.201.1) 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: 209.165.201.1
  > 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 209.165.201.1 left intact
  > Closing connection #0

Example: POST Use Certificate API
PnP Action APIs

Table 30: PnP Action API

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To start, stop, and restart a PnP action</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/pnp/action</td>
</tr>
</tbody>
</table>

Example for PnP action Payload

<input>
<command><start><stop><restart>

Example: POST PnP Action API

curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X POST https://209.165.201.1/api/operations/pnp/action -d '<input><command><start><stop><restart>'

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> POST /api/operations/pnp/action HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.43.0
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 39
>
* upload completely sent off: 39 out of 39 bytes
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Fri, 12 Aug 2016 14:38:13 GMT
< Content-Type: text/html
PnP APIs

PnP Server APIs

Table 31: PnP Server APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get the PnP IP address and port number</td>
<td>GET</td>
<td>No</td>
<td>/api/config/pnp?deep</td>
</tr>
<tr>
<td>To get the PnP operational status</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/pnp/status</td>
</tr>
<tr>
<td>To modify the PnP IP address and port number</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/pnp</td>
</tr>
<tr>
<td>To delete the PnP IP address and port number</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/pnp</td>
</tr>
<tr>
<td>To add PnP static IPv6 address</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/pnp</td>
</tr>
</tbody>
</table>

Example for PnP Server Payload (Static Mode)

```xml
<pnp>
  <static>
    <ip-address>192.0.2.1</ip-address>
    <port>80</port>
  </static>
  <automatic>
    <dhcp>disable</dhcp>
    <dns>disable</dns>
    <cco>disable</cco>
  </automatic>
</pnp>
```

Example for PnP Server Payload (Automatic Mode)

```xml
<pnp>
  <automatic>
    <dhcp>enable</dhcp>
    <dns>enable</dns>
    <cco>enable</cco>
    <timeout>100</timeout>
  </automatic>
</pnp>
```
Table 32: PnP Server Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;static&gt;</td>
<td>number</td>
<td>Static IP address</td>
<td>Yes (if you disable the automatic option)</td>
</tr>
<tr>
<td>&lt;ip-address&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;port&gt;</td>
<td>number</td>
<td>Port number</td>
<td>Yes (in static mode)</td>
</tr>
<tr>
<td>&lt;dhcp&gt;&lt;disable/&gt;&lt;/dhcp&gt;</td>
<td>text</td>
<td>Enable or disable DHCP</td>
<td>Yes (one of the options is mandatory)</td>
</tr>
<tr>
<td>&lt;dhcp&gt;&lt;enable/&gt;&lt;/dhcp&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;dns&gt;&lt;disable/&gt;&lt;/dns&gt;</td>
<td>text</td>
<td>Enable or disable DNS</td>
<td>Yes (one of the options is mandatory)</td>
</tr>
<tr>
<td>&lt;dns&gt;&lt;enable/&gt;&lt;/dns&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;cco&gt;&lt;disable/&gt;&lt;/cco&gt;</td>
<td>text</td>
<td>Enable or disable CCO</td>
<td>Yes (one of the options is mandatory)</td>
</tr>
<tr>
<td>&lt;cco&gt;&lt;enable/&gt;&lt;/cco&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;timeout&gt;</td>
<td>number</td>
<td>Timeout in seconds. Default is 60 seconds.</td>
<td>No</td>
</tr>
</tbody>
</table>

Example: PUT PnP Server API

Use this API to enable static mode for PnP discovery.

curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X PUT https://209.165.201.1/api/config/pnp -d '<pnp><static><ip-address>209.165.201.2</ip-address><port>50</port></static><automatic><dhcp>disable</dhcp><dns>disable</dns><cco>disable</cco></automatic></pnp>'

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
Use this API to enable automatic mode for PnP discovery.

curl -k -v -u admin:admin -H 'Accept:application/vnd.yang.data+xml' -H 'Content-Type:application/vnd.yang.data+xml' -X PUT https://209.165.201.1/api/config/pnp -d '<pnp><automatic><timeout>100</timeout><dhcp>enable</dhcp><dns>enable</dns><cco>enable</cco></automatic></pnp>'

Example: GET PnP Server API

Use this API to get the PnP IP address and port number.

curl -X GET -v -k -u admin:admin https://192.0.2.2/api/config/pnp -H 'Content-type:application/vnd.yang.data+xml'

Example: GET PnP Server API
Example: DELETE PnP Server API

```
curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X DELETE https://209.165.201.1/api/config/pnp
```

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software

NFVIS Related APIs

Example: DELETE PnP Server API
Example: DELETE PnP Server API
Resource APIs

- CPU Allocation Summary API, on page 71
- Resources CPU APIs, on page 72
- Resource Precheck APIs, on page 73
- Resources VM APIs, on page 75

CPU Allocation Summary API

This API provides the total number of CPUs available for use, and the total number of CPUs that are already used by VMs.

Table 33: CPU Allocation Summary API

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get information on the number of CPUs allocated to VMs, and the CPUs that are already used by VMs.</td>
<td>GET</td>
<td>No</td>
<td>api/operational/resources/cpu-info/allocation</td>
</tr>
</tbody>
</table>

Example: GET CPU Allocation Summary API

```
curl -k -v -u "admin:admin" -X GET
"https://209.165.201.1/api/operational/resources/cpu-info/allocation?deep"
```

* About to connect() to 209.165.201.1 port 443 (#0)
* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* SSL connection using TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Aug 26 07:41:22 2016 GMT
  * expire date: Aug 24 07:41:22 2026 GMT
Resources CPU APIs

These APIs return CPU information for each CPU or the user specified CPU (cpu-id). These APIs also display a list of VMs (VNF name, VCPU number, VCPU ID) pinned to the CPU or CPUs.

Table 34: Resources CPU APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get the VMs running in each physical CPU in the system.</td>
<td>GET</td>
<td>No</td>
<td>api/operational/resources/cpu-info/cpus</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/operational/resources/cpu-info/cpus/cpu</td>
</tr>
<tr>
<td>To get the VMs running in a specific physical CPU in the system.</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/resources/cpu-info/cpus/cpu/&lt;cpu-id&gt;</td>
</tr>
</tbody>
</table>

Example: GET Resources CPU API

curl -k -v -u "admin:admin" -X GET "https://209.165.201.1/api/operational/resources/cpu-info/cpus/cpu/?deep"
* About to connect() to 209.165.201.1 port 443 (#0)
* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) 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: Aug 26 07:41:22 2016 GMT
    * expire date: Aug 24 07:41:22 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/operational/resources/cpu-info/cpus/cpu/7?deep HTTP/1.1
  > Authorization: Basic YWRtaW46YWRtaW4=
  > User-Agent: curl/7.29.0
  > Host: 209.165.201.1
  > Accept: */*
  > HTTP/1.1 200 OK
  < Server: nginx/1.6.3
  < Date: Sat, 27 Aug 2016 06:32:52 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
   xmlns:resource-info="http://www.cisco.com/nfvis/resources">
    <cpu-id>7</cpu-id>
    <socket-id>0</socket-id>
    <core-id>7</core-id>
    <system-use>false</system-use>
    <vnf>
      <name>1472148428.ROUTER</name>
      <vcpus>4</vcpus>
      <low-latency>true</low-latency>
      <vcpu-id>0</vcpu-id>
    </vnf>
  </cpu>
* Connection #0 to host 209.165.201.1 left intact

**Resource Precheck APIs**

Use the resource precheck APIs in the following scenarios to check if sufficient resources are available:

- Right before deploying a new VM. Do not proceed to deploy the VM if no sufficient resources are available.
- Right before updating a flavor of a deployed VM. Do not modify the VM if no sufficient resources are available.

**Table 35: Resource Precheck APIs**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
</table>

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
**Example: GET Resource Precheck API**

```
curl -k -v -u "admin:admin" -X GET "https://209.165.201.1/api/operational/resources/precheck/vnf/newvnf,csr1kv-large,true?deep"
```

---

When the low-latency property of a VM is true, the VM will require one or more dedicated CPUs.

For a new VM, the `<vnf_name>` can be any string (for example, "new-vnf"). For updating a deployed VM, the `<vnf_name>` must be the `<deployment_name>.<vm_group_name>`.

---

<table>
<thead>
<tr>
<th>Check if there are sufficient resources for the deployment of a VM.</th>
<th>GET</th>
<th>No</th>
<th>/api/operational/resources/precheck/vnf/&lt;vnf_name&gt;,&lt;flavor_name&gt;,&lt;true or false for low-latency&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check if there are sufficient resources for updating a deployed VM.</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/resources/precheck/vnf/&lt;deployment_name&gt;.&lt;vm_group_name&gt;</td>
</tr>
</tbody>
</table>

---

**Note**

* API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software*
<flavor-name>csr1kv-large</flavor-name>
<low-latency>true</low-latency>
<sufficient-resources>false</sufficient-resources>
<cause>No enough CPU resources</cause>
</vnf>
* Connection #0 to host 209.165.201.1 left intact

Resources VM APIs

These APIs return CPU information for each VM or the user specified VM. These APIs also display a list CPUs pinned by the VM.

Table 36: Resources VM APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get the CPUs and VCPUs allocated to each of the VMs in the system.</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/resources/cpu-info/vnfs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/operational/resources/cpu-info/vnfs/vnf</td>
</tr>
<tr>
<td>To get the CPUs and VCPUs allocated to a specific VM in the system.</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/resources/cpu-info/vnfs/vnf/ &lt;deployment_name&gt;.&lt;vm_group_name&gt;</td>
</tr>
</tbody>
</table>

Example: GET Resources VNF API

curl -k -v -u "admin:admin" -X GET "https://209.165.201.1/api/operational/resources/cpu-info/vnfs/vnf/1472148662.ROUTER2?deep"*
* About to connect() to 209.165.201.1 port 443 (#0)
* Trying 209.165.201.1...*
* Connected to 209.165.201.1 (209.165.201.1) 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: Aug 26 07:41:22 2016 GMT
  * expire date: Aug 24 07:41:22 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/operational/resources/cpu-info/vnfs/vnf/1472148662.ROUTER2?deep HTTP/1.1
  > Authorization: Basic YWRtaW46YWRtaW4=
  > User-Agent: curl/7.29.0
Example: GET Resources VNF API

```xml
     xmlns:resource-info="http://www.cisco.com/nfvis/resources">
  <name>1472148662.ROUTER2</name>
  <vcpus>2</vcpus>
  <low-latency>true</low-latency>
  <cpu>
    <vcpu-id>0</vcpu-id>
    <socket-id>0</socket-id>
    <core-id>3</core-id>
    <cpu-id>3</cpu-id>
  </cpu>
  <cpu>
    <vcpu-id>1</vcpu-id>
    <socket-id>0</socket-id>
    <core-id>2</core-id>
    <cpu-id>2</cpu-id>
  </cpu>
</vnf>
```

* Connection #0 to host 209.165.201.1 left intact
Networks and Bridges APIs

• Bridge APIs, on page 77
• Network Creation APIs, on page 81

Bridge APIs

By default, a LAN bridge (lan-br), a WAN bridge (wan-br) and wan2-br for ENCS 5000 series are created in the system.

Table 37: Bridge APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To create a bridge</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/bridges</td>
</tr>
<tr>
<td>To verify a bridge configuration</td>
<td>GET</td>
<td>No</td>
<td>/api/config/bridges?deep</td>
</tr>
<tr>
<td>To get specific IP/DHCP info for all bridges</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/bridge-settings/ip dhcp_configuration</td>
</tr>
<tr>
<td>To get specific IP/DHCP info for specific bridge</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/bridge-settings/ip dhcp_configuration/br_name</td>
</tr>
<tr>
<td>To modify a bridge, and attach a port to the bridge</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/bridges/bridge/&lt;bridge name&gt;</td>
</tr>
<tr>
<td>To delete a bridge</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/bridges/bridge/&lt;bridge name&gt;</td>
</tr>
</tbody>
</table>

Example for Bridge Payload

```xml
<bridge>
  <name>sc-br</name>
  <port>
    <name>eth3</name>
  </port>
</bridge>
```
Table 38: Bridge Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bridge name</td>
<td>String</td>
<td>Name of the bridge.</td>
<td>Yes</td>
</tr>
<tr>
<td>port name</td>
<td>String</td>
<td>Name of the port the bridge is attached to.</td>
<td>Yes</td>
</tr>
<tr>
<td>dhcp</td>
<td></td>
<td>Flag to specify DHCP configuration</td>
<td>No</td>
</tr>
<tr>
<td>ip address</td>
<td>String</td>
<td>IP address</td>
<td>No</td>
</tr>
<tr>
<td>ip netmask</td>
<td>String</td>
<td>Netmask</td>
<td>No</td>
</tr>
<tr>
<td>dhcp-ipv6</td>
<td></td>
<td>Flag to specify DHCP IPv6 configuration</td>
<td>No</td>
</tr>
<tr>
<td>slaac-ipv6</td>
<td></td>
<td>Flag to specify SLAAC IPv6 configuration</td>
<td>No</td>
</tr>
<tr>
<td>ipv6 address</td>
<td>String</td>
<td>IPv6 address and prefix length</td>
<td>No</td>
</tr>
<tr>
<td>vlan</td>
<td>Integer</td>
<td>VLAN tag</td>
<td>No</td>
</tr>
</tbody>
</table>

Example: POST Bridge Creation API

```bash
curl -k -v -u admin:admin -H Content-Type:application/vnd.yang.data+xml -X POST https://209.165.201.1/api/config/bridges -d "<bridge><name>sc-br</name><port><name>eth3</name></port><dhcp/></bridge>".
```

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
Example: GET Bridge Configuration API

curl -k -v -u admin:admin -H Content-Type:application/vnd.yang.data+xml -X GET "https://209.165.201.1/api/config/bridges?deep"
* About to connect() to 209.165.201.1 port 443 (#0)
* Trying 209.165.201.1... connected
* Connected to 209.165.201.1 (209.165.201.1) 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: 209.165.201.1
> 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

  <bridge>
    <name>lan-br</name>
    <port>
      <name>eth0</name>
    </port>
  </bridge>
  <bridge>
    <name>wan-br</name>
    <port>
      <name>eth1</name>
    </port>
  </bridge>
</bridges>
Example: DELETE Bridge API

curl -k -v -u admin:admin -X DELETE https://209.165.201.1/api/config/bridges/bridge/sc-br

* About to connect() to 209.165.201.1 port 443 (#0)
* Trying 209.165.201.1... connected
* Connected to 209.165.201.1 (209.165.201.1) 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: 209.165.201.1
> Accept: */*
>
HTTP/1.1 204 No Content
< Server: nginx/1.6.3

Example: GET IPv4 address for all bridges

curl -k -v -u admin:admin -H "Accept:application/vnd.yang.data+json" -H "Content-Type:application/vnd.yang.data+json" -X GET https://localhost/api/operational/bridge-settings/ip-info/ipv4_address

Example: GET dhcp enabled under wan-br


Example: DELETE Bridge API

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
Network Creation APIs

By default a LAN network (lan-net), a WAN network (wan-net) and wan2-net for ENCS 5000 series are created in the system.

### Table 39: Network Creation APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To create a network</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/networks</td>
</tr>
<tr>
<td>To verify network configuration details</td>
<td>GET</td>
<td>No</td>
<td>/api/config/networks?deep</td>
</tr>
<tr>
<td>To modify a network</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/networks/network/&lt;network name&gt;</td>
</tr>
<tr>
<td>To delete a network</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/networks/network/&lt;network name&gt;</td>
</tr>
</tbody>
</table>

### Example for Network Creation Payload

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

### Table 40: Network Creation Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>network name</td>
<td>String</td>
<td>Name of the network.</td>
<td>Yes</td>
</tr>
<tr>
<td>bridge</td>
<td>String</td>
<td>Name of the bridge the network is attached to.</td>
<td>Yes</td>
</tr>
<tr>
<td>trunk</td>
<td>Boolean</td>
<td>Network set to trunk mode.</td>
<td>No/true</td>
</tr>
<tr>
<td>sriov</td>
<td>Boolean</td>
<td>SR-IOV supported on the network.</td>
<td>No/false</td>
</tr>
<tr>
<td>native-tagged</td>
<td>Boolean</td>
<td>Specifies if the network is tagged or not.</td>
<td>No</td>
</tr>
</tbody>
</table>
Example: POST Network API

curl -k -v -u admin:admin -H Content-Type:application/vnd.yang.data+xml -X
   POST  https://209.165.201.1/api/config/networks -d
   "<network><name>sc-net</name><bridge>sc-bridge</bridge></network>"

* About to connect() to 209.165.201.1 port 443 (#0)
* Trying 209.165.201.1... connected
* Connected to 209.165.201.1 (209.165.201.1) 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: 209.165.201.1
> 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://209.165.201.1/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 209.165.201.1 left intact

---

<table>
<thead>
<tr>
<th>native-vlan</th>
<th>Integer</th>
<th>Specifies a native VLAN. It sets the native characteristics when the interface is in <strong>trunk</strong> mode. If you do not configure a native VLAN, the default VLAN 1 is used as the native VLAN.</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>vlan</td>
<td>Integer</td>
<td>Specifies the VLAN number. If the <strong>trunk</strong> parameter is configured as true, this parameter specifies a set of VLAN numbers and ranges. If <strong>trunk</strong> parameter is false, access mode is true, then this parameter can have only one VLAN number.</td>
<td>No</td>
</tr>
</tbody>
</table>
VM Lifecycle Management APIs

- VM Image Registration APIs, on page 85
- Custom Flavor Creation APIs, on page 90
- VM Deployment APIs, on page 94
- VM Action APIs, on page 107
- VM Network APIs, on page 112
- Network File System APIs, on page 112
- VNC Console Start API, on page 114
- VM Multi Serial Port APIs, on page 114

VM Image Registration APIs

Table 41: VM Registration APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image registration</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/vm_lifecycle/images</td>
</tr>
<tr>
<td>Get image configuration</td>
<td>GET</td>
<td>No</td>
<td>/api/config/vm_lifecycle/images?deep</td>
</tr>
<tr>
<td>Get image status</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/vm_lifecycle/opdata/images/image/&lt;image_name&gt;?deep</td>
</tr>
<tr>
<td>Image Unregistration</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/vm_lifecycle/images/image/&lt;image_name&gt;</td>
</tr>
</tbody>
</table>

Example for Image Registration Payload

```xml
<image>
  <name>isrv9.16.03.01</name>
  <src>http://<filename_with_full-path-of-the-file>/isrv-universalk9.16.03.01.tar.gz</src>
</image>
```
Added in NFVIS 3.12.x release:

Table 42: Image Registration Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the VM image</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Example: POST Image Registration API


Example: POST Image Registration to External Disk API

curl -k -v -u admin:Cisco123# -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X POST https://209.165.201.1/api/config/vm_lifecycle/images -d '<image><name>Linuxnew</name><src>file:///mnt/extdatastore2/uploads/TinyLinux.tar.gz</src><properties><property><name>placement</name><value>datastore3</value></property></properties></image>'

Note: Unnecessary use of -X or --request, POST is already inferred.

* Trying 209.165.201.1...
* TCP_NODELAY set
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1

<table>
<thead>
<tr>
<th>src</th>
<th>URL</th>
<th>Full path of the VM image</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added in NFVIS 3.12.x release</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>certificate_validation</td>
<td>True/false</td>
<td>Enable certificate validation by setting this tag to &quot;true&quot;</td>
<td>Yes</td>
</tr>
<tr>
<td>certificate_string</td>
<td>String</td>
<td>Validate Web-server with the raw contents of a certificate file</td>
<td>Yes</td>
</tr>
<tr>
<td>certificate_file</td>
<td>URL</td>
<td>Validate Web-server with a certificate file</td>
<td>Yes</td>
</tr>
</tbody>
</table>
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/ssl/cert.pem
  * CAPath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* ALPN, server did not agree to a protocol
* Server certificate:
  * subject: CN=nfvis
  * start date: Jun 12 19:40:33 2018 GMT
  * expire date: Jun 11 19:40:33 2023 GMT
  * issuer: CN=nfvis
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
* POST /api/config/vm_lifecycle/images HTTP/1.1
  > Host: 209.165.201.1
  > Authorization: Basic YWRtaW46Q2lzY28xMjMj
  > User-Agent: curl/7.54.0
  > Accept:application/vnd.yang.data+xml
  > Content-Type:application/vnd.yang.data+xml
  > Content-Length: 190
  >
  > upload completely sent off: 190 out of 190 bytes
  < HTTP/1.1 201 Created
  < Server: nginx
  < Date: Tue, 12 Jun 2018 22:59:05 GMT
  < Content-Type: text/html
  < Content-Length: 0
  < Location: https://172.25.221.106/api/config/vm_lifecycle/images/image/Linuxnew
  < Connection: keep-alive
  < Last-Modified: Tue, 12 Jun 2018 22:59:04 GMT
  < Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
  < Etag: 1528-844344-814906
  < Pragma: no-cache
  <
  > Connection #0 to host 209.165.201.1 left intact

Example: GET Image Configuration API

  /* About to connect() to 209.165.201.1 port 80 (#0)
  * Trying 209.165.201.1...
  * Connected to 209.165.201.1 (209.165.201.1) port 80 (#0)
  * Server auth using Basic with user 'admin'
  > GET /api/config/vm_lifecycle/images?deep HTTP/1.1
  > Authorization: Basic YWRtaW46YWRtaW4=
  > User-Agent: curl/7.29.0
  > Host: 209.165.201.1
  > Accept:application/vnd.yang.data+xml
Example: GET Image Status API

/* About to connect() to 209.165.201.1 port 80 (#0)
/* Trying 209.165.201.1...
/* Connected to 209.165.201.1 (209.165.201.1) port 80 (#0)
/* Server auth using Basic with user 'admin'
> GET /api/operational/vm_lifecycle/opdata/images/image/isrv-image?deep HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 209.165.201.1
> 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
> <name>isrv.03.16.02</name>
> <image_id>585a1792-145c-4946-9929-e040d3002a59</image_id>
> <public>true</public>
> </image>
/* Connection #0 to host 209.165.201.1 left intact
The supported image states are:

- IMAGE_UNDEF_STATE
- IMAGE_CREATING_STATE
- IMAGE_ACTIVE_STATE
- IMAGE_DELETING_STATE
- IMAGE_DELETED_STATE
- IMAGE_ERROR_STATE

Example: DELETE Image Registration API

curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X DELETE https://209.165.201.1/api/config/vm_lifecycle/images/image/isr-3.16.0.1a

/*About to connect() to 209.165.201.1 port 80 (#0)
/* Trying 209.165.201.1...
/* Connected to 209.165.201.1 (209.165.201.1) port 80 (#0)
/* Server auth using Basic with user 'admin'
> DELETE /api/config/vm_lifecycle/images/image/isr-image HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 209.165.201.1
> 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 209.165.201.1 left intact

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>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>APIs</th>
</tr>
</thead>
</table>

Table 43: Flavor Creation APIs
To create a flavor | POST | Yes | /api/config/vm_lifecycle/flavors

To get configuration details of a flavor | GET | No | • /api/config/vm_lifecycle/flavors
• /api/config/vm_lifecycle/flavors?deep
• /api/config/vm_lifecycle/flavors/flavor/<flavor_name>?deep

To view the operational status of a flavor | GET | No | /api/operational/vm_lifecycle/opdata/flavors/flavor/<flavor_name>?deep

To delete a flavor | DELETE | No | /api/config/vm_lifecycle/flavors/flavor/<flavor-name>

Example for Flavor Creation Payload

```xml
<flavor>
  <name>ISR_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 44: Description for Flavor Creation Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the flavor.</td>
<td>Yes</td>
</tr>
<tr>
<td>vcpus</td>
<td>Number</td>
<td>Number of virtual CPUs.</td>
<td>Yes</td>
</tr>
<tr>
<td>memory_mb</td>
<td>Number</td>
<td>Amount of memory in Mega Bytes.</td>
<td>Yes</td>
</tr>
<tr>
<td>root_disk_mb</td>
<td>Number</td>
<td>Virtual root disk size in gigabytes.</td>
<td>Yes</td>
</tr>
<tr>
<td>ephemeral_disk_mb</td>
<td>Number</td>
<td>A temporary storage that is added to your instance.</td>
<td>No</td>
</tr>
<tr>
<td>swap_disk_mb</td>
<td>Number</td>
<td>The space used on a hard disk as RAM</td>
<td>No</td>
</tr>
</tbody>
</table>

Note: Added support in 3.7.1.
Example: POST Flavor API

curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X POST https://209.165.201.1/api/config/vm_lifecycle/flavors -d '<flavor><name>windows</name><ephemeral_disk_mb>0</ephemeral_disk_mb><memory_mb>4096</memory_mb><root_disk_mb>12288</root_disk_mb><swap_disk_mb>0</swap_disk_mb><vcpus>2</vcpus></flavor>'

Example: GET Flavor Configuration API


API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
Example: GET Flavor Status API

curl -k -v -u admin:admin -X
GET https://209.165.201.1/api/operational/vm_lifecycle/flavors?deep
* About to connect() to 209.165.201.1 port 80 (#0)
* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 80 (#0)
* Server auth using Basic with user 'admin'
> GET /api/operational/vm_lifecycle/flavors?deep HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 209.165.201.1
> 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
<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>
<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>
<properties>
<property>
<name>source_image</name>
<value>ASAv_IMAGE</value>
</property>
</properties>
</flavor>
* Connection #0 to host 209.165.201.1 left intact
VM Deployment APIs

Table 45: VM Deployment APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy a VM</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/vm_lifecycle/tenants/tenant/admin/deployments</td>
</tr>
<tr>
<td>Get deployment</td>
<td>GET</td>
<td>No</td>
<td>/api/config/vm_lifecycle/tenants/tenant/admin/deployments?deep</td>
</tr>
<tr>
<td>configuration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Get deployment</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/vm_lifecycle/tenants/tenant/admin/deployments?deep</td>
</tr>
<tr>
<td>status and details</td>
<td></td>
<td></td>
<td>/api/operational/vm_lifecycle/opdata/tenants/tenant/admin/deployments((deployment_name)),,-,?deep</td>
</tr>
<tr>
<td>Undeploy a VM</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/vm_lifecycle/tenants/tenant/admin/deployments/deployment/&lt;deployment_name&gt;</td>
</tr>
</tbody>
</table>

* Connection #0 to host 209.165.201.1 left intact
Example: POST VM Deployment API for Cisco ISRv

curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X POST https://209.165.201.1/api/config/vm_lifecycle/tenants/tenant/admin/deployments --data '
  <deployment>
    <name>WINIsodep</name>
    <vm_group>
      <name>WINIsovmgrp</name>
      <image>WinServer2012R2.iso</image>
      <flavor>windows</flavor>
      <bootup_time>-1</bootup_time>
      <recovery_wait_time>0</recovery_wait_time>
      <kpi_data>
        <enabled>true</enabled>
      </kpi_data>
      <scaling>
        <min_active>1</min_active>
        <max_active>1</max_active>
        <elastic>true</elastic>
      </scaling>
      <placement>
        <type>zone_host</type>
        <enforcement(strict)</enforcement>
        <host>datastore1</host>
      </placement>
      <recovery_policy>
        <recovery_type>AUTO</recovery_type>
        <action_on_recovery>REBOOT_ONLY</action_on_recovery>
      </recovery_policy>
    </vm_group>
  </deployment>'

/* About to connect() to 209.165.201.1 port 80 (#0)
/* Trying 209.165.201.1...
/* Connected to 209.165.201.1 (209.165.201.1) port 80 (#0)
/* Server auth using Basic with user 'admin'
> POST /api/config/vm_lifecycle/tenants/tenant/admin/deployments HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 209.165.201.1
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 1313
> Expect: 100-continue
> * Done waiting for 100-continue
> * Done waiting for 100-continue
> * Done waiting for 100-continue
> < HTTP/1.1 201 Created
> < Server:
> < Location: http://209.165.201.1/api/config/vm_lifecycle/tenants/tenant/admin/deployments/deployment/WinServer2012R2
> < 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 209.165.201.1 left intact

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
To enable NIM support on a Cisco ISRv running on Cisco ENCS, you must use the following variable in the ISRv deployment payload:

```xml
<variable>
  <name>ngio</name>
  <val>enable</val>
</variable>
```

### Table 46: Description for VM Deployment Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>deployment name</td>
<td>string</td>
<td>Name of the deployment</td>
<td>Yes</td>
</tr>
<tr>
<td>vm_group name</td>
<td>string</td>
<td>Name of the VM group.</td>
<td>Yes</td>
</tr>
<tr>
<td>vim_vm_name</td>
<td>string</td>
<td>Image name that was used to register.</td>
<td>Yes</td>
</tr>
<tr>
<td>bootup_time</td>
<td>integer</td>
<td>Bootup time could vary depending on the VM image that you have chosen.</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, bootup time is 600 seconds for a Cisco ISRv image. If no</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>monitoring is required for the VM, set the bootup time as -1.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note A monitored VM must have a valid bootup time. The corresponding KPI</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fields are mandatory for the monitored VM. In the case of an unmonitored</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VM, KPI fields are optional.</td>
<td></td>
</tr>
<tr>
<td>placement type</td>
<td>string</td>
<td>Set VM deployment placement. For example deploying the VM on external</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>datastore if the system has external datastore. Must set value to &quot;zone_host&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>if deploying the VM on external datastore or NFS.</td>
<td></td>
</tr>
<tr>
<td>placement host</td>
<td>string</td>
<td>Specify placement datastore. For example ENCS system has external</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>datastore. Specify placement host. Allowed values are: datastore2,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>datastore3, nfs_storage</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Description</td>
<td>Required for</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>recovery_wait_time</td>
<td>integer</td>
<td>Time in seconds that this VM takes to perform a normal warm reboot. This will be used to avoid premature VM recovery in case VM becomes unresponsive due to operator reboot. This is important as VM recovery will result in loss of data that is stored on root disk. If speedy recovery is more important than the data on the root disk, this value can be optionally set to 0.</td>
<td></td>
</tr>
<tr>
<td>recovery_policy</td>
<td>string</td>
<td>The action performed during recovery. Possible values: REBOOT_ONLY; REDEPLOY_ONLY; REBOOT_THEN_REDEPLOY</td>
<td>Yes (for monitored VMs)</td>
</tr>
<tr>
<td>action_on_recovery</td>
<td>string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>interface nicid</td>
<td>integer</td>
<td>The network interface card ID. At least one NIC ID is mandatory for monitored VMs. It is optional for unmonitored VMs.</td>
<td>Yes (for monitored VMs)</td>
</tr>
<tr>
<td>network</td>
<td>string</td>
<td>Name of the network attached to the NIC ID. All networks (such as LAN and WAN) except the internal management network require an IP address. The vNIC attachment to the internal management network is only required for VMs, which require monitoring. If this interface is for monitoring, network must be set to &quot;int-mgmt-net&quot;</td>
<td>Yes (for monitored VMs)</td>
</tr>
<tr>
<td>ip_address</td>
<td>string</td>
<td>IPv4 address</td>
<td>Yes</td>
</tr>
<tr>
<td>port_forwarding</td>
<td>-</td>
<td>Note If port forwarding is included, all elements under it are mandatory.</td>
<td>No</td>
</tr>
<tr>
<td>port type</td>
<td>enum</td>
<td>SSH, HTTPS, TCP, and Telnet</td>
<td>No</td>
</tr>
<tr>
<td>protocol</td>
<td>string</td>
<td>TCP</td>
<td>No</td>
</tr>
<tr>
<td>vnf_port</td>
<td>integer</td>
<td>Port number corresponding to the protocol used.</td>
<td>No</td>
</tr>
<tr>
<td>external_port_range</td>
<td>integer</td>
<td>Unique port number to specify the start and end range.</td>
<td>No</td>
</tr>
<tr>
<td>scaling</td>
<td>container</td>
<td>Specifies how many instances of a particular type of VM need to be instantiated, and whether elastic scale-in and scale-out are required.</td>
<td>Yes</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Description</td>
<td>Required for Monitored VMs</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>min_active</td>
<td>integer</td>
<td>Describes the minimum number of VMs to be activated.</td>
<td>Yes</td>
</tr>
<tr>
<td>max_active</td>
<td>integer</td>
<td>Describes the maximum number of VMs to be activated.</td>
<td>Yes</td>
</tr>
<tr>
<td>kpi_data</td>
<td>-</td>
<td>Key performance indicators data.</td>
<td>Yes (for monitored VMs)</td>
</tr>
<tr>
<td>event_name</td>
<td>string</td>
<td>Name of the event.</td>
<td>Yes (for monitored VMs)</td>
</tr>
<tr>
<td>metric_value</td>
<td>string</td>
<td>The metric threshold value of the KPI.</td>
<td>Yes (for monitored VMs)</td>
</tr>
<tr>
<td>metric_cond</td>
<td>enum</td>
<td>Specifies the direction of the metric value change for this KPI.</td>
<td>Yes (for monitored VMs)</td>
</tr>
<tr>
<td>metric_type</td>
<td>integer</td>
<td>Supported metric types are INT8, UINT8, INT16, UINT16, INT32, UINT32, FLOAT, DOUBLE, and STRING.</td>
<td>Yes (for monitored VMs)</td>
</tr>
<tr>
<td>metric_collector_type</td>
<td>String</td>
<td>If the image boot-up time is provided, monitoring must be set to ICMPIng. This field type can be empty if boot-up time is -1.</td>
<td>Yes (for monitored VMs)</td>
</tr>
<tr>
<td>nicid&gt;</td>
<td>Integer</td>
<td>The card ID of the interface through which this VM is monitored. It should be the ID specified in one of the interfaces section in the payload.</td>
<td>Yes (for monitored VMs)</td>
</tr>
<tr>
<td>poll_frequency</td>
<td>Integer</td>
<td></td>
<td>Yes (for monitored VMs)</td>
</tr>
<tr>
<td>polling_unit</td>
<td>string</td>
<td></td>
<td>Yes (for monitored VMs)</td>
</tr>
<tr>
<td>continuous_alarm</td>
<td>boolean</td>
<td>Continuous events needs to be generated. Value supported: false, true</td>
<td>Yes (for monitored VMs)</td>
</tr>
<tr>
<td>rule event_name</td>
<td>string</td>
<td>Name of the event.</td>
<td>No</td>
</tr>
</tbody>
</table>
**Example: DELETE VM Deployment API**

```bash
curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X DELETE https://209.165.201.1/api/config/vm_lifecycle/tenants/tenant/admin/deployments/deployment/ISRdep1 /* About to connect() to 209.165.201.1 port 80 (#0) /* Trying 209.165.201.1... /* Connected to 209.165.201.1 (209.165.201.1) port 80 (#0) /* Server auth using Basic with user 'admin'
> DELETE /api/config/vm_lifecycle/tenants/tenant/admin/deployments/deployment/ISRdep1 HTTP/1.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.29.0
> Host: 209.165.201.1
> 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 209.165.201.1 left intact
```
Examples for VM Deployment Payload with Bootstrap Configuration Options

**Note**
You need to specify the exact name of the VM bootstrap configuration file under the `<dst>` element in the deployment payload. This name can vary with each VM. For example, the Cisco ASAv bootstrap configuration file is "day0-config".

**Option 1 Example: Deployment Payload for Bundling Bootstrap Configuration Files into the VM Package**

In this method, the bootstrap configuration variables can be tokenized. You have to provide token values at the time of deployment using the deployment payload.

The following is the extract from the Cisco ASAv bootstrap configuration with tokenized variables. Tokenized variables are highlighted in this example.

ASAv Version 9.4.1
firewall transparent
ssh version 2
!
interface management0/0
description vnf-mgmt-net
nameif vnf-mgmt
security-level 100
ip address ${VNF_MGMT_IP} ${VNF_MGMT_NETMASK}
no shutdown
!
interface GigabitEthernet0/0
description service-net
nameif outside
security-level 0
bridge-group 10
no shutdown
!
interface GigabitEthernet0/1
description lan-net
nameif inside
bridge-group 10
security-level 100
no shutdown
!
interface BVI10
ip address ${BRIDGE_IP} ${BRIDGE_MASK}
!
snmp-server enable
snmp-server community public
http server enable
http 0.0.0.0 0.0.0.0 management
crypto key generate rsa modulus 2048
username test password test123
ssh 0.0.0.0 0.0.0.0 management
aaa authentication ssh console LOCAL
route vnf-mgmt 0.0.0.0 0.0.0.0 ${VNF_MGMT_GW} 1
route outside 0.0.0.0 0.0.0.0 ${BRIDGE_GW} 1

The following is an example for the Cisco ASAv deployment payload with the tokenized variables.

```xml
<deployment>
  <name>ASAv</name>
</deployment>
```
<vm_group>
  <name>FirwallGroup</name>
  <image>asavImage</image>
  <flavor>IASAv51</flavor>
  <bootup_time>600</bootup_time>
  <recovery_wait_time>0</recovery_wait_time>
  <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>20024</start>
            <end>20024</end>
          </external_port_range>
        </port>
      </port_forwarding>
    </interface>
    <interface>
      <nicid>1</nicid>
      <network>sc-net</network>
    </interface>
    <interface>
      <nicid>2</nicid>
      <network>lan-net</network>
    </interface>
  </interfaces>
  <kpi_data>
    <enabled>true</enabled>
    <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>FALSE recover autohealing</action>
        <action>TRUE servicebooted.sh</action>
      </rule>
    </admin_rules>
    <user_rules/>
  </rules>
  <scaling>
    <min_active>1</min_active>
    <max_active>1</max_active>
  </scaling>
  <config_data>
    <configuration>
Option 2 Example: Bootstrap Configuration without Tokens in the Deployment Payload

In this example, the entire Cisco ASAv bootstrap configuration is copied under the <data> element.

```xml
<deployment>
  <name>ASAv</name>
  <vm_group>
    <name>ASAvGroup</name>
    <bootup_time>-1</bootup_time>
    <config_data>
      <configuration>
        <dst>day0-config</dst>
        <data>
          ASA Version 9.4.1
          firewall transparent
          ssh version 2
          interface management0/0
          description vnf-mgmt-net
          nameif vnf-mgmt
          security-level 100
          ip address 11.20.0.3 255.255.255.0
          no shutdown
          interface GigabitEthernet0/0
          description service-net
          nameif outside
          security-level 0
          bridge-group 10
          no shutdown
          
          interface GigabitEthernet0/1
          description lan-net
          nameif inside
        </data>
      </configuration>
    </config_data>
  </vm_group>
</deployment>
```
bridge-group 10
security-level 100
no shutdown
interface BVI10
ip address 12.20.0.3 255.255.255.0

snmp-server enable
snmp-server community public
http server enable
http 0.0.0.0 0.0.0.0 management
crypto key generate rsa modulus 2048
username test password test123
ssh 0.0.0.0 0.0.0.0 management
aaa authentication ssh console LOCAL
route vnf-mgmt 0.0.0.0 0.0.0.0 11.20.0.1 1
route outside 0.0.0.0 0.0.0.0 12.20.0.1 1

</data>
</image>
</interfaces>
</nicid>0</nicid>
</network>vnf-mgmt-net</network>
</interface>
</nicid>1</nicid>
</ip_address>12.20.0.68</ip_address>
</network>sc-net</network>
</interface>
</nicid>
</interfaces>
</kpi_data>
<kpi>
<event_name>VM_ALIVE</event_name>
<metric_collector>
<continuous_alarm>false</continuous_alarm>
<nicid>0</nicid>
<poll_frequency>3</poll_frequency>
<polling_unit>seconds</polling_unit>
<type>ICMPPing</type>
</metric_collector>
<metric Cond>GT</metric Cond>
<metric_type>UINT32</metric_type>
<metric_value>1</metric_value>
</kpi>
</kpi_data>
<recovery_wait_time>0</recovery_wait_time>
</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>
</scaling>
<max active>1</max_active>
<min active>1</min_active>
</scaling>
</vm_group>
</deployment>
Option 3 Example: Deployment Payload with Local Bootstrap Configuration File

In this example, a reference to the Cisco ASAv local bootstrap configuration file is provided from the payload under the `<configuration>` element. If the bootstrap configuration file has tokens, you have to provide token values in the deployment payload under the configuration section.

```xml
<deployment>
  <name>asaV</name>
  <vm_group>
    <name>firewall_Group</name>
    <image>ASAvImage</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>wan-net</network>
        <ip_address>172.19.181.42</ip_address>
      </interface>
      <interface>
        <nicid>2</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>
  </vm_group>
</deployment>
```
Adding or Editing a vNIC Using the VM Deployment API

Using the VM deployment API, you can add, edit, or delete as many vNICs as you want. For these actions, you will have to use the PUT method of the VM deployment API. VM's vNIC can be updated when VM is active or stopped.

Editing vNIC (add / delete / changing network) will reboot the VM is the VM does not support vNIC hot-add / hot-delete / hot-modify.

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/vm_lifecycle/tenants/tenant/admin/deployments—Provides the names of all VMs that are deployed.
- **GET** https://<nfvis_ip>/api/config/vm_lifecycle/tenants/tenant/admin/deployments/deployment/ISR1—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, ISR1.

curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X PUT https://<nfvis_ip>/api/config/vm_lifecycle/tenants/tenant/admin/deployments/deployment/ISR1/vm_group/ISR-VM/interfaces --data "'<interfaces>
  <interface>
    <nicid>0</nicid>
    <network>int-mgmt-net</network>
  </interface>
  <interface>
    <nicid>1</nicid>
    <network>sc-net</network>
  </interface>
</interfaces>"
Example: Editing a vNIC

You can edit the attributes of an existing vNIC. In this example, the network is changed from sc-net to wan-net for NIC ID 1.

curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X PUT https://<nfvis_ip>/api/config/vm_lifecycle/tenants/tenant/admin/deployments/deployment/ISR1/vm_group/ISR-VM/interfaces --data '

Example: Deleting a vNIC

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 (ISR1 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://<nfvis_ip>/api/config/vm_lifecycle/tenants/tenant/admin/deployments/deployment/ISR1/vm_group/ISR-VM/interfaces --data '

See the Example: POST VM Deployment API for Cisco ISRv, on page 95 for details on the API command.
Changing the Flavor Using the VM Deployment API

Using this deployment API, you can change or update the flavor. Before changing an existing flavor to a new one, ensure that you have the new flavor created using the flavor creation API. VM’s flavor change be updated when VM is active or stopped.

Example: Changing the Flavor

In this example, the existing flavor ID is changed to `isr-flavor` for the VM deployed as ISR1.

```
curl -k -v -u admin:admin -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X PUT https://nfvis_ip/api/config/vm_lifecycle/tenants/tenant/admin/deployments/deployment/ISR1/vm_group/ISR-VM/flavor --data '<flavor>isr-flavor</flavor>'
```

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

See the Example: POST VM Deployment API for Cisco ISRv, on page 95 for details on the API command.

VM Action APIs

You may want to get the VM name before running the VM operations API. To get the VM name, use the following operational status API:

```
/api/operational/vm_lifecycle/opdata/tenants/tenant/admin/deployments/<deploy name>,.-,?deep
```

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>To start a VM</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/vmAction</td>
</tr>
<tr>
<td>To stop a VM</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/vmAction</td>
</tr>
<tr>
<td>To reboot a VM</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/vmAction</td>
</tr>
<tr>
<td>To enable VM monitoring</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/vmAction</td>
</tr>
<tr>
<td>To disable VM monitoring</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/vmAction</td>
</tr>
<tr>
<td>To backup a VM</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/vmBackupAction</td>
</tr>
</tbody>
</table>
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, REBOOT, ENABLE_MONITOR or DISABLE_MONITOR as required.

```xml
<vmAction>
    <actionType>START</actionType>
    <vmName>ISR</vmName>
</vmAction>
```

### Table 48: Description for VM Operations Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmAction actionType</td>
<td>String</td>
<td>Type of VM action. Value supported: STOP, START, REBOOT, ENABLE_MONITOR, DISABLE_MONITOR</td>
<td>Yes</td>
</tr>
<tr>
<td>vmName</td>
<td>String</td>
<td>Name of the VM instance.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example: POST Start VM API

```bash
curl -k -v -u "admin:admin" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X POST https://209.165.201.1/api/operations/vmAction --data '<vmAction><actionType>START</actionType><vmName><vm-instance name></vmName></vmAction>'
```

* About to connect() to 209.165.201.1 port 80 (#0)
* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) 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: 209.165.201.1
  > 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 209.165.201.1 left intact
```
Example: POST Stop VM API

curl -k -v -u "admin:admin" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X POST [https://209.165.201.1/api/operations/vmAction|http://209.165.201.1/api/operations/vmAction] --data '<vmAction><actionType>STOP</actionType><vmName><vm-instance name></vmName></vmAction>'

Example: POST Restart VM API

curl -k -v -u "admin:admin" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X POST https://209.165.201.1/api/operations/vmAction --data '<vmAction><actionType>REBOOT</actionType><vmName><vm-instance name></vmName></vmAction>'
Example: POST Enable VM Monitoring API

curl -k -v -u "admin:password" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X POST https://209.165.201.1/api/operations/vmAction --data "<vmAction><actionType>ENABLE_MONITOR</actionType><vmName><vm-instance name></vmName></vmAction>"

* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
  * ALPN, offering h2
  * ALPN, offering http/1.1
  * Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
  * successfully set certificate verify locations:
    * CAfile: /etc/ssl/cert.pem
    * CApath: none
  * TLSv1.2 (OUT), TLS handshake, Client hello (1):
  * TLSv1.2 (IN), TLS handshake, Server hello (2):
    * NPN, negotiated HTTP1.1
  * TLSv1.2 (IN), TLS handshake, Certificate (11):
  * TLSv1.2 (IN), TLS handshake, Server key exchange (12):
  * TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
  * TLSv1.2 (OUT), TLS change cipher, Client hello (1):
  * TLSv1.2 (OUT), TLS handshake, Unknown (67):
  * TLSv1.2 (OUT), TLS handshake, Finished (20):
    * TLSv1.2 (IN), TLS change cipher, Client hello (1):
    * TLSv1.2 (IN), TLS handshake, Finished (20):
  * SSL connection using TLSv1.2 / DHE-RSA-AES256-SHA
  * ALPN, server did not agree to a protocol
  * Server certificate:
    * subject: CN=nfvis
    * start date: Apr 18 18:54:43 2018 GMT
    * expire date: Apr 15 18:54:43 2028 GMT
    * issuer: CN=nfvis
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
  > POST /api/operations/vmAction HTTP/1.1
  > Host: 209.165.201.1
  > Authorization: Basic YWRtaW46Q2lzY28xMjMj
  > User-Agent: curl/7.54.0
  > Accept:application/vnd.yang.data+xml
  > Content-Type:application/vnd.yang.data+xml
  > Content-Length: 133
  >
  * upload completely sent off: 133 out of 133 bytes
  < HTTP/1.1 204 No Content
  < Server: nginx
  < Date: Wed, 25 Apr 2018 21:57:32 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 209.165.201.1 left intact
Example: POST Disable VM Monitoring API

curl -k -v -u "admin:password" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X POST https://209.165.201.1/api/operations/vmAction --data '<vmAction><actionType>DISABLE_MONITOR</actionType><vmName><vm-instance name></vmName></vmAction>'

* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/ssl/cert.pem
  * CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP/1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
  * TLSv1.2 (IN), TLS handshake, Server key exchange (12):
  * TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / DHE-RSA-AES256-SHA
* ALPN, server did not agree to a protocol
* Server certificate:
  * subject: CN=nfvis
  * start date: Apr 18 18:54:43 2018 GMT
  * expire date: Apr 15 18:54:43 2028 GMT
  * issuer: CN=nfvis
* SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'
  > POST /api/operations/vmAction HTTP/1.1
  > Host: 209.165.201.1
  > Authorization: Basic YWRtaW46Q2lzY28xMjMj
  > User-Agent: curl/7.54.0
  > Accept:application/vnd.yang.data+xml
  > Content-Type:application/vnd.yang.data+xml
  > Content-Length: 133
  >
  *
  * upload completely sent off: 133 out of 133 bytes
< HTTP/1.1 204 No Content
< Server: nginx
< Date: Wed, 25 Apr 2018 21:57:32 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 209.165.201.1 left intact
## VM Network APIs

**Table 49: VM Network APIs**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>To delete an existing subnet</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/vm_lifecycle/networks/network/int-mgmt-net/subnet/int-mgmt-net-subnet</td>
</tr>
<tr>
<td>To create a new subnet</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/vm_lifecycle/networks/network/int-mgmt-net</td>
</tr>
</tbody>
</table>

### Example for VM Networks Payload

This section provides an example of networks payload.

```xml
<subnet>
  <name>int-mgmt-net-subnet</name>
  <dhcp>false</dhcp>
  <address>105.20.0.0</address>
  <netmask>255.255.255.0</netmask>
  <gateway>105.20.0.1</gateway>
</subnet>
```

**Table 50: Description for VM Networks Payload**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Management subnet name</td>
<td>Yes - Should be set to int-mgmt-net-subnet</td>
</tr>
<tr>
<td>address</td>
<td>String</td>
<td>Subnet address for this network</td>
<td>Yes</td>
</tr>
<tr>
<td>netmask</td>
<td>String</td>
<td>Netmask for the network</td>
<td>No</td>
</tr>
<tr>
<td>gateway</td>
<td>String</td>
<td>The gateway IP</td>
<td>No</td>
</tr>
</tbody>
</table>

## Network File System APIs

**Table 51: Network File System APIs**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>To mount NFS</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/system</td>
</tr>
<tr>
<td>To unmount NFS</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/system/storage</td>
</tr>
<tr>
<td>To register images on NFS</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/vm_lifecycle/images</td>
</tr>
</tbody>
</table>
NFVIS Related APIs
Network File System APIs

To unregister
images on NFS

DELETE

To deploy VM on POST
NFS using an
image

No

/api/config/vm_lifecycle/images/image

Yes

/api/config/vm_lifecycle/tenants/tenant/admin/deployments

Example for Network File System Payload
This section provides an example of NFS payload.
<image>
<name> myas10</name>
<src>file:///data/mount/nfs_storage/repository/asav961.tar.gz</src>
<properties>
<property>
<name>placement</name>
<value>nfs_storage</value>
</property>
</properties>
</image>
{"deployment":
{"name":"15065483181",
"vm_group":
{"name":"myasav1",
"image":"asav961",
"flavor":"ASAv5",
"bootup_time":"-1",
"recovery_wait_time":"0",
"placement":{"type":"zone_host",
"host":"nfs_storage"},
"recovery_policy":{"action_on_recovery":"REBOOT_ONLY"},
"interfaces":{"interface":[{"nicid":0,"network":"lan-net","model":"virtio"}]},
"scaling":{"min_active":"1","max_active":"1"}}}}

Added in NFVIS 3.12.x release:

<image>
<name>ubuntu</name>
<src>file:///data/intdatastore/uploads/ubuntu_raw.tar.gz</src>
<properties>
<property>
<name>placement</name>
<value>iscsi:test</value>
</property>
</properties>
</image>

<tenant>
<name>admin</name>
<deployments>
<deployment>
<name>ubuntu</name>
<vm_group>
<name>ubgrp</name>

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
113


VNC Console Start API

**Table 52: VNC Console Start API**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To start a VNC console</td>
<td>POST</td>
<td>No</td>
<td>/api/operations/vncconsole/start</td>
</tr>
</tbody>
</table>

VM Multi Serial Port APIs

**Table 53: VM Multi Serial Port API**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To deploy attaching serial port to VNF</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/vm_lifecycle/tenants/tenant/admin/deployments</td>
</tr>
</tbody>
</table>

**Example for VM Multi Serial Port API**

```
{
    "deployment": {
        "name": "15065483181",
        "vm_group": {
            "name": "myasav1",
            "network": "int-mgmt-net",
            "nicid": "0"
        }
    }
}
```
Table 54: Description for VM Multi Serial Port Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>serial</td>
<td>String</td>
<td>Serial port number</td>
<td>Yes</td>
</tr>
<tr>
<td>serial_type</td>
<td>String</td>
<td>Serial type, telnet or console</td>
<td>Yes</td>
</tr>
<tr>
<td>service_port</td>
<td>String</td>
<td>Service port number</td>
<td>Yes</td>
</tr>
</tbody>
</table>
VM Multi Serial Port APIs
CHAPTER 7

System Monitoring APIs

The system monitoring APIs are used to get statistics on the host and VNFs running on the host. These statistics are used by the portal for pictorial representation. These statistics are collected over a specified duration. For large durations, average values are returned. The default duration for all host and VNF queries is set to five minutes. If data is not available for a particular interval during the specified duration, the API returns "na" (not available) for that interval.

- Host CPU Stats APIs, on page 117
- Host CPU Table API, on page 120
- Host Disk Stats APIs, on page 123
- Host Memory Stats APIs, on page 130
- Host Memory Table APIs, on page 131
- Host Port Stats APIs, on page 133
- Host Port Table APIs, on page 136
- VNF CPU Stats APIs, on page 139
- VNF Disk Stats APIs, on page 142
- VNF Memory Stats API, on page 144
- VNF Port Stats APIs, on page 146

### Host CPU Stats APIs

**Table 55: Host CPU Stats APIs**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get the host CPU utilization of a CPU state</td>
<td>GET</td>
<td>No</td>
<td>• /api/operational/system-monitoring/host/cpu/stats&lt;br&gt;• /api/operational/system-monitoring/host/cpu/stats/cpu-usage?deep&lt;br&gt;• /api/operational/system-monitoring/host/cpu/stats/cpu-usage/&lt;duration&gt;,&lt;cpu-state&gt;?deep</td>
</tr>
</tbody>
</table>

Valid duration: 1min, 5min, 15min, 30min, 1h, 1H, 6h, 6H, 1d, 1D, 5d, 5D, 30d, and 30D
curl -k -u "admin:admin" -X GET
https://192.0.2.2/api/operational/system-monitoring/host/cpu/stats/cpu-usage/5min,non-idle?deep
Note: Unnecessary use of -X or --request, GET is already inferred.
* Trying 192.0.2.2...
* Connected to 192.0.2.2 (192.0.2.2) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
  * TLSv1.0 (IN), TLS handshake, Server hello (2):
  * TLSv1.0 (IN), TLS handshake, Certificate (11):
  * TLSv1.0 (IN), TLS handshake, Server key exchange (12):
  * TLSv1.0 (IN), TLS handshake, Server finished (14):
  * TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
  * TLSv1.0 (OUT), TLS change cipher, Client hello (1):
  * TLSv1.0 (OUT), TLS handshake, Finished (20):
  * TLSv1.0 (IN), TLS change cipher, Client hello (1):
  * TLSv1.0 (IN), TLS handshake, Finished (20):
  * SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
  * Server certificate:
    * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
    * start date: Feb 3 05:02:29 2017 GMT
    * expire date: Feb 1 05:02:29 2027 GMT
    * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
    * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
> GET /api/operational/system-monitoring/host/cpu/stats/cpu-usage/5min,non-idle?deep HTTP/1.1
> Host: 192.0.2.2
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.49.1
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
>
< HTTP/1.1 200 OK
< Server: nginx/1.10.1
< Date: Tue, 07 Feb 2017 03:44:43 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
<
<cpu-usage xmlns="http://www.cisco.com/nfvos/system-monitoring"
xmlns:y="http://tail-f.com/ns/rest"
xmlns:system_monitoring="http://www.cisco.com/nfvos/system-monitoring">
  <duration>5min</duration>
  <collect-start-date-time>2017-02-07T03:39:40-00:00</collect-start-date-time>
  <collect-interval-seconds>10</collect-interval-seconds>
  <cpu>
    <id>0</id>
    <usage-percentage>[1.62, 1.16, 1.22, 1.44, 1.41, 1.46, 1.63, 1.82, 3.77, 2.61, 0.94, 1.32, 1.36, 1.14, 1.34, 1.38, 2.75, 2.33, 1.4, 1.28, 1.2, 1.26, 1.42, 1.44, 1.76, 1.22, 1.0, 1.32, 1.16]</usage-percentage>
  </cpu>
  <cpu>
    <id>1</id>
    <usage-percentage>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</usage-percentage>
  </cpu>
</cpu-usage>
Table 56: Field Description for Host CPU Stats API Response

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration</td>
<td>Required duration</td>
</tr>
<tr>
<td>usage-percentage</td>
<td>It is the percentage of CPU used for the requested state.</td>
</tr>
<tr>
<td>state</td>
<td>CPU state</td>
</tr>
<tr>
<td></td>
<td>The allowed CPU states are: non-idle, interrupt, nice, system, user, and wait.</td>
</tr>
</tbody>
</table>

Host CPU Table API

Table 57: Host CPU Table APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get the host CPU utilization statistics table (minimum, maximum, and average) of all CPU states on each of the CPUs</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system-monitoring/host/cpu/table</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/operational/system-monitoring/host/cpu/table?deep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/operational/system-monitoring/host/cpu/table/cpu-usage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/operational/system-monitoring/host/cpu/table/cpu-usage?deep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/operational/system-monitoring/host/cpu/table/cpu-usage/&lt;duration&gt;?deep</td>
</tr>
</tbody>
</table>

Valid duration: 1min, 5min, 15min, 30min, 1h, 1H, 6h, 6H, 1d, 1D, 5d, 5D, 30d, and 30D

Example: GET Host CPU Table API

```
curl -k -v -u admin:Cisco123# -X GET
```

Note: Unnecessary use of -X or --request, GET is already inferred.
* Trying 172.19.162.209...
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT50:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
* CAfile: /etc/pki/tls/certs/ca-bundle.crt
  CApath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Feb 3 05:02:29 2017 GMT
  * expire date: Feb 1 05:02:29 2027 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'
  > GET /api/operational/system-monitoring/host/cpu/table/cpu-usage/1h?deep HTTP/1.1
  > Host: 172.19.162.209
  > Authorization: Basic YWRtaW46Q2lzY28xMjMj
  > User-Agent: curl/7.49.1
  > Accept: */*
  > < HTTP/1.1 200 OK
  < Server: nginx/1.10.1
  < Date: Tue, 07 Feb 2017 04:10:56 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
  <
  <cpu-usage
  xmlns="http://www.cisco.com/nfvos/system-monitoring"
  xmlns:y="http://tail-f.com/ns/rest"
  xmlns:system_monitoring="http://www.cisco.com/nfvos/system-monitoring">
  <!--duration-->1h<!--duration-->
  <cpu>
  <id>0</id>
  <states>
    <state>non-idle</state>
    <min-percentage>0.9</min-percentage>
    <max-percentage>13.56</max-percentage>
    <average-percentage>1.72</average-percentage>
  </states>
  <states>
    <state>interrupt</state>
    <min-percentage>0.0</min-percentage>
    <max-percentage>0.0</max-percentage>
    <average-percentage>0.0</average-percentage>
  </states>
  <states>
    <state>nice</state>
    <min-percentage>0.0</min-percentage>
    <max-percentage>0.06</max-percentage>
    <average-percentage>0.0</average-percentage>
  </states>
  <states>
    <state>softirq</state>
    <min-percentage>0.0</min-percentage>
  </cpu>
</cpu-usage>
Example: GET Host CPU Table API

```
<cpu id="15">
  <states>
    <state>non-idle</state>
    <min-percentage>0.0</min-percentage>
    <max-percentage>0.0</max-percentage>
    <average-percentage>0.0</average-percentage>
  </states>
  <states>
    <state>interrupt</state>
    <min-percentage>0.0</min-percentage>
    <max-percentage>0.0</max-percentage>
    <average-percentage>0.0</average-percentage>
  </states>
  <states>
    <state>nice</state>
    <min-percentage>0.0</min-percentage>
    <max-percentage>0.0</max-percentage>
    <average-percentage>0.0</average-percentage>
  </states>
  <states>
    <state>softirq</state>
    <min-percentage>0.0</min-percentage>
    <max-percentage>0.0</max-percentage>
    <average-percentage>0.0</average-percentage>
  </states>
  <states>
    <state>steal</state>
    <min-percentage>0.0</min-percentage>
    <max-percentage>0.0</max-percentage>
    <average-percentage>0.0</average-percentage>
  </states>
</cpu>
...
Table 58: Field Description for Host CPU Table API Response

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration</td>
<td>Duration of this collection</td>
</tr>
<tr>
<td>cpu states</td>
<td>Indicates the CPU state. This can be non-idle, interrupt, nice, soft interrupt request line (IRQ), steal, system, user and wait.</td>
</tr>
<tr>
<td>cpu states min-percentage</td>
<td>Minimum percentage of CPU usage</td>
</tr>
<tr>
<td>cpu states max-percentage</td>
<td>Maximum percentage of CPU usage</td>
</tr>
<tr>
<td>cpu states average-percentage</td>
<td>Average percentage of CPU usage</td>
</tr>
</tbody>
</table>

Host Disk Stats APIs

Table 59: Host Disk Stats APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
</table>
### Example: GET Host Disk Stats API

**Example 1: disk-operations**

```bash
curl -k -v -u "admin:admin" -X GET "https://209.165.201.2/api/operational/system-monitoring/host/disk/stats/disk-operations/5min?deep
<
```

The valid duration can be: 1min, 5min, 15min, 30min, 1h, 1H, 6h, 6H, 1d, 1D, 5d, 5D, 30d, and 30D
Example: GET Host Disk Stats API
Example: GET Host Disk Stats API

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
### Table 60: Field Description for Host Disk Stats API Response

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>io-time-ms</td>
<td>Time spent doing I/Os in milliseconds</td>
</tr>
<tr>
<td>io-time-weighted-ms</td>
<td>Measure of both I/O completion time and the backlog that may be accumulating.</td>
</tr>
<tr>
<td>merged-reads-per-sec</td>
<td>The number of read operations that could be merged into already queued operations, that is one physical disk access served two or more logical operations.</td>
</tr>
<tr>
<td>merged-writes-per-sec</td>
<td>The number of write operations that could be merged into other already queued operations, that is one physical disk access served two or more logical operations.</td>
</tr>
<tr>
<td>bytes-read-per-sec</td>
<td>Bytes read per second</td>
</tr>
<tr>
<td>bytes-written-per-sec</td>
<td>Bytes written per second</td>
</tr>
<tr>
<td>reads-per-sec</td>
<td>Number of read operations per second</td>
</tr>
<tr>
<td>writes-per-sec</td>
<td>Number of write operations per second</td>
</tr>
<tr>
<td>time-per-read-ms</td>
<td>The average time a read operation took to complete</td>
</tr>
</tbody>
</table>
Example: GET Host Disk Stats API

time-per-write-ms | The average time a write operation took to complete

Example 2: disk-space

curl -k -v -u admin:Cisco123# -X GET
https://209.165.201.2/api/operational/system-monitoring/host/disk/stats/disk-space/5min?deep

Note: Unnecessary use of -X or --request, GET is already inferred.

* Trying 209.165.201.2...
* Connected to 209.165.201.2 (209.165.201.2) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS change cipher, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Feb 18 12:04:07 2017 GMT
  * expire date: Feb 16 12:04:07 2027 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'

> GET /api/operational/system-monitoring/host/disk/stats/disk-space/5min?deep HTTP/1.1
> Host: 209.165.201.2
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.49.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.10.1
< Date: Wed, 22 Feb 2017 05:59:38 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
<
<disk-space xmlns="http://www.cisco.com/nfvos/system-monitoring"
xmlns:y="http://tail-f.com/ns/rest"
xmlns:system_monitoring="http://www.cisco.com/nfvos/system-monitoring">
<collect-start-date-time>2017-02-22T05:54:30-00:00</collect-start-date-time>
<collect-interval-seconds>10</collect-interval-seconds>
<mount-point>
<name>/boot</name>
<free-GB>0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33</free-GB>
<used-GB>0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1</used-GB>
This API response provides information about the disk name and data for various disk usage types.

Table 61: Field Description for Host Disk Stats API Response

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>free-GB</td>
<td>Gigabytes available</td>
</tr>
<tr>
<td>used-GB</td>
<td>Gigabytes in use</td>
</tr>
<tr>
<td>reserved-GB</td>
<td>Gigabytes reserved for the root user</td>
</tr>
</tbody>
</table>
Host Memory Stats APIs

Table 62: Host Memory Statistics APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
</table>
| To get the host memory utilization | GET    | No               | • /api/operational/system-monitoring/host/memory  
• /api/operational/system-monitoring/host/memory?deep  
• /api/operational/system-monitoring/host/memory/stats/mem-usage  
• /api/operational/system-monitoring/host/memory/stats/mem-usage?deep  
• /api/operational/system-monitoring/host/mem-usage/<duration>?deep |

The valid duration can be: 1min, 5min, 15min, 30min, 1h, 1H, 6h, 6H, 1d, 1D, 5d, 5D, 30d, and 30D

Example: GET Host Memory Stats API

curl -k -v -u admin:Cisco123# -X GET
Note: Unnecessary use of -X or --request, GET is already inferred.
* Trying 172.19.162.209...  
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)  
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH  
* successfully set certificate verify locations:  
* CAfile: /etc/pki/tls/certs/ca-bundle.crt  
* CApath: none  
* TLSv1.0 (OUT), TLS handshake, Client hello (1):  
* TLSv1.0 (IN), TLS handshake, Server hello (2):  
* TLSv1.0 (IN), TLS handshake, Certificate (11):  
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):  
* TLSv1.0 (IN), TLS handshake, Server finished (14):  
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16):  
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):  
* TLSv1.0 (OUT), TLS handshake, Finished (20):  
* TLSv1.0 (IN), TLS change cipher, Client hello (1):  
* TLSv1.0 (IN), TLS handshake, Finished (20):  
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA  
* Server certificate:  
* subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate  
* start date: Feb 3 05:02:29 2017 GMT  
* expire date: Feb 1 05:02:29 2027 GMT  
* issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate  
* SSL certificate verify result: self signed certificate (18), continuing anyway.  
* Server auth using Basic with user 'admin'  
> GET /api/operational/system-monitoring/host/memory/stats/mem-usage/5min?deep HTTP/1.1  
> Host: 172.19.162.209  
> Authorization: Basic YWRtaW46Q2l1YXN0cmlz  
> User-Agent: curl/7.49.1  
> Accept: */*
>
HTTP/1.1 200 OK
Server: nginx/1.10.1
Date: Tue, 07 Feb 2017 04:24:45 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

<mem-usage xmlns="http://www.cisco.com/nfvos/system-monitoring"
xmлиns:="http://tail-f.com/ns/rest"
xmлиns:="system_monitoring= http://www.cisco.com/nfvos/system-monitoring">
<duration>5min</duration>
<collect-start-date-time>2017-02-07T04:19:40-00:00</collect-start-date-time>
<collect-interval-seconds>10</collect-interval-seconds>
16.56, 16.57, 16.58, 16.6, 16.61, 16.62]</buffered-MB>
<cached-MB>[3730.54, 3730.55, 3730.56, 3730.56, 3730.56, 3730.57, 3730.58, 3730.58, 3730.58,
3730.59, 3730.59, 3730.6, 3730.6, 3730.61, 3730.62, 3730.62, 3730.63, 3730.63,
3730.64, 3730.66, 3730.81, 3730.94, 3731.07, 3731.18, 3731.24, 3731.3, 3731.36,
3731.38]</cached-MB>
<free-MB>[54090.05, 54089.9, 54089.84, 54089.93, 54089.81, 54089.7, 54089.67, 54089.67,
54089.67, 54089.66, 54089.66, 54089.63, 54089.63, 54089.51, 54089.44, 54089.36, 54089.46,
54089.57, 54089.14, 54088.85, 54088.94, 54088.17, 54076.76, 54080.71, 54088.02,
54087.82, 54087.59, 54087.54, 54087.69]</free-MB>
<used-MB>[6086.81, 6086.9, 6086.98, 6086.8, 6086.76, 6086.8, 6086.78, 6086.85, 6086.86,
6086.83, 6086.67, 6086.55, 6086.68, 6086.83, 6086.86, 6086.84, 6086.75, 6086.67, 6087.09,
6087.36, 6087.83, 6088.08, 6087.79, 6099.19, 6095.12, 6087.67, 6087.74, 6087.86, 6087.84,
6087.66]</used-MB>
<slab-recl-MB>[186.79, 186.79, 186.79, 186.79, 186.79, 186.79, 186.79, 186.79, 186.79, 186.79,
186.79, 186.79, 186.79, 186.79, 186.79, 186.79, 186.79, 186.79, 186.79, 186.79,
186.79, 186.79, 186.79, 186.79, 186.79, 186.79, 186.79, 186.79, 186.79, 186.79]</slab-recl-MB>
<slab-unrecl-MB>[52.04, 52.08, 52.05, 52.11, 52.24, 52.3, 52.33, 52.24, 52.17, 52.26,
52.37, 52.41, 52.35, 52.32, 52.4, 52.37, 52.3, 52.31, 52.31, 52.35, 52.28, 52.22,
52.2, 52.18, 52.2, 52.25, 52.29, 52.29]</slab-unrecl-MB>
</mem-usage>

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

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

### Host Memory Table APIs

**Table 63: Host Memory Table APIs**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
</table>

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
To get the host memory utilization in tabular format (minimum, maximum, and average) for each memory type:

<table>
<thead>
<tr>
<th>GET</th>
<th>No</th>
</tr>
</thead>
</table>
| /api/operational/system-monitoring/host/memory/table | *
| /api/operational/system-monitoring/host/memory/table?deep | *
| /api/operational/system-monitoring/host/memory/table/mem-usage | *
| /api/operational/system-monitoring/host/memory/table/mem-usage?deep | *
| /api/operational/system-monitoring/host/memory/table/mem-usage/<duration>?deep | *

The valid duration can be: 1min, 5min, 15min, 30min, 1h, 1H, 6h, 6H, 1d, 1D, 5d, 5D, 30d, and 30D.

---

**Example: GET Host Memory Table APIs**

```bash
curl -k -v -u admin:Cisco123# -X GET 'https://172.19.162.209/api/operational/system-monitoring/host/memory/table/mem-usage/1h?deep'
```

Note: Unnecessary use of `-X` or `--request`, GET is already inferred.

* Trying 172.19.162.209...
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
  * Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Feb 3 05:02:29 2017 GMT
  * expire date: Feb 1 05:02:29 2027 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
> GET /api/operational/system-monitoring/host/memory/table/mem-usage/1h?deep HTTP/1.1
> Host: 172.19.162.209
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.49.1
> Accept: */*
> < HTTP/1.1 200 OK
< Server: nginx/1.10.1
< Date: Tue, 07 Feb 2017 04:27:22 GMT
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
```
Host Port Stats APIs

Table 64: Host Port Stats APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
</table>

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
To get the packet counts information (error-rx, error-tx, error-total, packets-rx, packets-tx, and packets-total) on all host interfaces

<table>
<thead>
<tr>
<th>GET</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>/api/operational/system-monitoring/host/port</td>
<td></td>
</tr>
<tr>
<td>/api/operational/system-monitoring/host/port/stats</td>
<td></td>
</tr>
<tr>
<td>/api/operational/system-monitoring/host/port/stats?deep</td>
<td></td>
</tr>
<tr>
<td>/api/operational/system-monitoring/host/port/stats/port-usage/&lt;duration&gt;?deep</td>
<td></td>
</tr>
</tbody>
</table>

The valid duration can be: 1min, 5min, 15min, 30min, 1h, 1H, 6h, 6H, 1d, 1D, 5d, 5D, 30d, and 30D

**Example: GET Host Port Stats API**

curl -k -v -u admin:Cisco123# -X GET 'https://172.19.162.209/api/operational/system-monitoring/host/port/stats/port-usage/5min?deep'

* Trying 172.19.162.209...
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Feb 18 12:04:07 2017 GMT
  * expire date: Feb 16 12:04:07 2027 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
> GET /api/operational/system-monitoring/host/port/stats/port-usage/5min?deep HTTP/1.1
> Host: 172.19.162.209
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.49.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.10.1
< Date: Wed, 22 Feb 2017 05:43:42 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
<
<port-usage xmlns="http://www.cisco.com/nfvos/system-monitoring"
xmlns:y="http://tail-f.com/ns/rest"
xmns:system_monitoring="http://www.cisco.com/nfvos/system-monitoring">
<duration>5min</duration>
<collect-start-date-time>2017-02-22T05:38:40-00:00</collect-start-date-time>
<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Total Packets per sec</th>
<th>RX Packets per sec</th>
<th>TX Packets per sec</th>
<th>Total Errors per sec</th>
<th>RX Errors per sec</th>
<th>TX Errors per sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>eth0</td>
<td></td>
<td>[38.8, 24.38, 34.9, 37.94, 21.64, 20.84, 31.72, 36.22, 22.44]</td>
<td>[36.66, 22.02, 32.72]</td>
<td>[2.14, 2.36, 2.18]</td>
<td>[0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0]</td>
</tr>
<tr>
<td>eth1</td>
<td></td>
<td>[34.58, 19.66, 30.5, 32.92, 18.3, 17.08, 26.88, 32.52]</td>
<td>[34.44, 19.54, 30.46]</td>
<td>[0.14, 0.12, 0.04]</td>
<td>[0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0]</td>
</tr>
<tr>
<td>eth2</td>
<td></td>
<td>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0]</td>
</tr>
<tr>
<td>eth3</td>
<td></td>
<td>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0]</td>
</tr>
<tr>
<td>eth4</td>
<td></td>
<td>[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0, 0.0]</td>
<td>[0.0, 0.0]</td>
</tr>
</tbody>
</table>
Table 65: Field Description for Host Port Statistics API Response

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Interface name</td>
</tr>
<tr>
<td>total-packets-per-sec</td>
<td>Total (rx + tx) packet rate</td>
</tr>
<tr>
<td>rx-packets-per-sec</td>
<td>Packets received per second</td>
</tr>
<tr>
<td>tx-packets-per-sec</td>
<td>Packets transmitted per second</td>
</tr>
<tr>
<td>total-errors-per-sec</td>
<td>Total (rx + tx) error rate</td>
</tr>
<tr>
<td>rx-errors-per-sec</td>
<td>Error rate for received packets</td>
</tr>
<tr>
<td>tx-errors-per-sec</td>
<td>Error rate for transmitted packets</td>
</tr>
</tbody>
</table>

Host Port Table APIs

Table 66: Host Port Table APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
To get statistics information about all ports

<table>
<thead>
<tr>
<th>GET</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>/api/operational/system-monitoring/host/port</td>
<td></td>
</tr>
<tr>
<td>/api/operational/system-monitoring/host/port/table</td>
<td></td>
</tr>
<tr>
<td>/api/operational/system-monitoring/host/port/table?deep</td>
<td></td>
</tr>
<tr>
<td>/api/operational/system-monitoring/host/port/table/port-usage/&lt;duration&gt;,&lt;name&gt;?deep</td>
<td></td>
</tr>
</tbody>
</table>

The valid duration can be: 1min, 5min, 15min, 30min, 1h, 1H, 6h, 6H, 1d, 1D, 5d, 5D, 30d, and 30D

Example: GET Host Port Table API

curl -k -v -u admin:Cisco123# -X GET

Note: Unnecessary use of -X or --request, GET is already inferred.

* Trying 172.19.162.209...
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CPath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):

* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA

* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Feb 18 12:04:07 2017 GMT
  * expire date: Feb 16 12:04:07 2027 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'

> GET /api/operational/system-monitoring/host/port/table?deep HTTP/1.1
> Host: 172.19.162.209
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.49.1
> Accept: */*
> HTTP/1.1 200 OK
> Server: nginx/1.10.1
> Date: Wed, 22 Feb 2017 05:50:53 GMT
> Content-Type: application/vnd.yang.data+xml
> Content-Length: 1470
> Transfer-Encoding: chunked
> Connection: keep-alive
> Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
> Pragma: no-cache

<table xmlns="http://www.cisco.com/nfvos/system-monitoring"
xmllns:y="http://tail-f.com/ns/rest"
xmllns:system_monitoring="http://www.cisco.com/nfvos/system-monitoring">
<port-usage>
Example: GET Host Port Table API
<status>down</status>
<ip-address>NA</ip-address>
.rx-packets>0</rx-packets>
.tx-packets>0</tx-packets>
.rx-packets-per-sec>0.0</rx-packets-per-sec>
.tx-packets-per-sec>0.0</tx-packets-per-sec>
</port-usage>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the host interface or port</td>
</tr>
<tr>
<td>collect-start-date-time</td>
<td>The actual start date and time of this collection</td>
</tr>
<tr>
<td>duration</td>
<td>The duration of this collection</td>
</tr>
<tr>
<td>Status</td>
<td>Port status</td>
</tr>
<tr>
<td>IP_Address</td>
<td>IP address of this interface</td>
</tr>
<tr>
<td>collect-interval-seconds</td>
<td>Time interval of the collection</td>
</tr>
<tr>
<td>rx-packets</td>
<td>Received packets</td>
</tr>
<tr>
<td>tx-packets</td>
<td>Transmitted packets</td>
</tr>
<tr>
<td>rx-packets-per-sec</td>
<td>Received packet rate (packets/second)</td>
</tr>
<tr>
<td>tx-packets-per-sec</td>
<td>Transmitted packet rate (packets/second)</td>
</tr>
</tbody>
</table>

**VNF CPU Stats APIs**

Table 68: VNF CPU Stats APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
</table>
To get CPU statistics information of VMs

- GET
  - /api/operational/system-monitoring/vnf/vcpu
  - /api/operational/system-monitoring/vnf/vcpu/stats
  - /api/operational/system-monitoring/vnf/vcpu/stats?deep
  - /api/operational/system-monitoring/vnf/vcpu/stats/vcpu-usage
  - /api/operational/system-monitoring/vnf/vcpu/stats/vcpu-usage?deep

- GET No
  - /api/operational/system-monitoring/vnf
    - /vcpu/stats/vcpu-usage/<duration>?deep
  - /api/operational/system-monitoring/vnf
    - /vcpu/stats/vcpu-usage/<duration>/vnf/<vnf-name>?deep

The valid duration can be: 1min, 5min, 15min, 30min, 1h, 6h, 1d, 1D, 5d, 5D, 30d, and 30D

Example: GET VNF CPU Stats API

This example is for all VNFs.

curl -k -v -u admin:Cisco123# -X GET
https://209.165.201.2/api/operational/system-monitoring/vnf/vcpu/stats/vcpu-usage/5min?deep

Note: Unnecessary use of -X or --request, GET is already inferred.
- Trying 209.165.201.2...
- Connected to 209.165.201.2 (209.165.201.2) port 443 (10)
- Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
- successfully set certificate verify locations:
  - CAfile: /etc/pki/tls/certs/ca-bundle.crt
  - CApath: none
- TLSv1.0 (OUT), TLS handshake, Client hello (1):
- TLSv1.0 (IN), TLS handshake, Server hello (2):
- TLSv1.0 (IN), TLS handshake, Certificate (11):
- TLSv1.0 (IN), TLS handshake, Server key exchange (12):
- TLSv1.0 (IN), TLS handshake, Server finished (14):
- TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
- TLSv1.0 (OUT), TLS change cipher, Client hello (1):
- TLSv1.0 (OUT), TLS handshake, Finished (20):
- TLSv1.0 (IN), TLS change cipher, Client hello (1):
- TLSv1.0 (IN), TLS handshake, Finished (20):
- SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
- Server certificate:
  - subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  - start date: Mar 8 19:19:56 2017 GMT
  - expire date: Mar 6 19:19:56 2027 GMT
- issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
- SSL certificate verify result: self signed certificate (18), continuing anyway.
- Server auth using Basic with user 'admin'
- GET /api/operational/system-monitoring/vnf/vcpu/stats/vcpu-usage/5min?deep HTTP/1.1
- Host: 209.165.201.2
- Authorization: Basic YWRtaW46Q2lzY28xMjl4
- User-Agent: curl/7.49.1
- Accept: */*
-
HTTP/1.1 200 OK
Server: nginx/1.10.1
Date: Thu, 09 Mar 2017 20:37:13 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

<vcpu-usage xmlns="http://www.cisco.com/nfvos/system-monitoring"
xmlns:y="http://tail-f.com/ns/rest"
xmlns:system_monitoring="http://www.cisco.com/nfvos/system-monitoring">
<duration>5min</duration>
<vnf>
<name>1489003560.ROUTER</name>
<collect-start-date-time>2017-03-09T20:32:10-00:00</collect-start-date-time>
<collect-interval-seconds>10</collect-interval-seconds>
<total-percentage>
</total-percentage>
<vcpu>
$id>0</id>
<vcpu-percentage>
[7.06, 9.82, 10.22, 6.5, 6.5, 6.54, 6.6, 6.48, 6.3, 6.3, 6.38, 6.46, 7.76, 8.44, 6.4, 6.4, 6.44, 6.5, 6.42, 6.38, 6.46, 6.48, 6.6, 6.64, 6.66, 6.44, 6.36, 6.52]
</vcpu-percentage>
</vcpu>
<vcpu>
$id>1</id>
<vcpu-percentage>
</vcpu-percentage>
</vcpu>
<vcpu>
$id>2</id>
<vcpu-percentage>
[10.6, 10.68, 10.72, 10.6, 10.6, 10.64, 10.66, 10.6, 10.6, 10.64, 10.7, 10.66, 10.64, 10.7, 10.7, 10.7, 10.74, 10.76, 10.76, 10.7, 10.74, 10.8, 10.76]
</vcpu-percentage>
</vcpu>
<vcpu>
$id>3</id>
<vcpu-percentage>
[30.78, 30.36, 30.12, 30.14, 29.9, 29.82, 29.74, 29.76, 29.54, 29.18, 28.96, 28.9, 28.7, 28.32, 28.08, 27.9, 27.82, 27.46, 27.06, 26.96, 26.78, 26.56, 26.38, 26.12, 25.92, 26.2, 26.52, 25.1]
</vcpu-percentage>
</vcpu>
</vnf>
<vnf>
<name>1489002218.OTHER</name>
<collect-start-date-time>2017-03-09T20:32:10-00:00</collect-start-date-time>
<collect-interval-seconds>10</collect-interval-seconds>
<total-percentage>
[0.36, 0.3, 0.3, 0.18, 0.16, 0.32, 0.2, 0.2, 0.2, 0.2, 0.2, 0.16, 0.18, 0.26, 0.2, 0.24, 0.22, 0.18, 0.3, 0.26, 0.2, 0.2, 0.24, 0.3, 0.26, 0.24, 0.26, 0.2]
</total-percentage>
<vcpu>
$id>0</id>
<vcpu-percentage>
[0.36, 0.26, 0.24, 0.18, 0.16, 0.32, 0.2, 0.2, 0.2, 0.2, 0.16, 0.18, 0.22, 0.18, 0.3, 0.22, 0.14, 0.24, 0.26, 0.2, 0.2, 0.24, 0.26, 0.2, 0.28, 0.28, 0.14, 0.2]
</vcpu-percentage>
</vcpu>
</vnf>
</vcpu-usage>
# VNF Disk Stats APIs

## Table 69: VNF Disk Stats APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get the VNF disk statistics</td>
<td>GET</td>
<td>No</td>
<td>• /api/operational/system-monitoring/vnf/disk&lt;br&gt;• /api/operational/system-monitoring/vnf/disk/stats&lt;br&gt;• /api/operational/system-monitoring/vnf/disk/stats?deep&lt;br&gt;• /api/operational/system-monitoring/vnf/disk/stats/disk-operations&lt;br&gt;• /api/operational/system-monitoring/vnf/disk/stats/disk-operations?deep&lt;br&gt;• /api/operational/system-monitoring/vnf/disk/stats/disk-operations/&lt;duration&gt;?deep&lt;br&gt;• /api/operational/system-monitoring/vnf/disk/stats/disk-operations/&lt;duration&gt;/vnf?deep&lt;br&gt;• /api/operational/system-monitoring/vnf/disk/stats/disk-operations/&lt;duration&gt;/vnf/&lt;vnf-name&gt;?deep</td>
</tr>
</tbody>
</table>

The valid duration can be: 1min, 5min, 15min, 30min, 1h, 1H, 6h, 6H, 1d, 1D, 5d, 5D, 30d, and 30D

## Example: GET VNF Disk Stats API

This example is for all VMs.

```bash
curl -k -v -u admin:Cisco123# -X GET https://209.165.201.2/api/operational/system-monitoring/vnf/disk/stats/disk-operations/5min?deep
```

Note: Unnecessary use of -X or --request, GET is already inferred.

- Trying 209.165.201.2...
- Connected to 209.165.201.2 (209.165.201.2) port 443 (#0)
- Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
- successfully set certificate verify locations:
  * CAs: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
- TLSv1.0 (OUT), TLS handshake, Client hello (1):
- TLSv1.0 (IN), TLS handshake, Server hello (2):
- TLSv1.0 (IN), TLS handshake, Certificate (11):
- TLSv1.0 (IN), TLS handshake, Server key exchange (12):
- TLSv1.0 (IN), TLS handshake, Server finished (14):
- TLSv1.0 (OUT), TLS handshake, Client key exchange (16):

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
* Server certificate:
* subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* start date: Feb 18 12:04:07 2017 GMT
* expire date: Feb 16 12:04:07 2027 GMT
* issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'
> GET /api/operational/system-monitoring/vnf/disk/stats/disk-operations/5min?deep HTTP/1.1
> Host: 209.165.201.2
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.49.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.10.1
< Date: Wed, 22 Feb 2017 06:17:48 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
<
<disk-operations xmlns="http://www.cisco.com/nfvos/system-monitoring"
/nfvos/system-monitoring">
<duration>5min</duration>
</disk-operations>
VNF Memory Stats API

Table 70: VNF Memory Stats APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get the memory statistics of VMs</td>
<td>GET</td>
<td>No</td>
<td>• /api/operational/system-monitoring/vnf/memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• /api/operational/system-monitoring/vnf/memory/stats</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• /api/operational/system-monitoring/vnf/memory/stats?deep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• /api/operational/system-monitoring/vnf/memory/stats/mem-usage/&lt;duration&gt;?deep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• /api/operational/system-monitoring/vnf</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/memory/stats/mem-usage/&lt;duration&gt;/vnf/&lt;vnf-name&gt;?deep</td>
</tr>
</tbody>
</table>

The valid duration can be: 1min, 5min, 15min, 30min, 1h, 1H, 6h, 6H, 1d, 1D, 5d, 5D, 30d, and 30D

Example: GET VNF Memory Stats API

This example is for all VMs.

curl -k -v -u "admin:admin" -X GET https://209.165.201.2/api/operational/system-monitoring/vnf/memory/stats/mem-usage/5min?deep

Note: Unnecessary use of -X or --request, GET is already inferred.
* Trying 209.165.201.2...
* Connected to 209.165.201.2 (209.165.201.2) port 443 (ssl) (1/1)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
  * TLSv1.0 (IN), TLS handshake, Server hello (2):
  * TLSv1.0 (IN), TLS handshake, Certificate (11):
  * TLSv1.0 (IN), TLS handshake, Server key exchange (12):
  * TLSv1.0 (IN), TLS handshake, Server finished (14):
  * TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
  * TLSv1.0 (OUT), TLS change cipher, Client hello (1):
  * TLSv1.0 (OUT), TLS handshake, Finished (20):
Example: GET VNF Memory Stats API

Table 71: Field Description for VNF Memory Stats API Response

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>total-MB</td>
<td>Total memory of the VNF in MB</td>
</tr>
<tr>
<td>rss-MB</td>
<td>Resident Set Size of the VNF in MB</td>
</tr>
</tbody>
</table>
# VNF Port Stats APIs

## Table 72: VNF Port Stats APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get the VNF port statistics</td>
<td>GET</td>
<td>No</td>
<td>• /api/operational/system-monitoring/vnf/port</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• /api/operational/system-monitoring/vnf/port/stats</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• /api/operational/system-monitoring/vnf/port/stats?deep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• /api/operational/system-monitoring/vnf/port/stats/port-usage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• /api/operational/system-monitoring/vnf/port/stats/port-usage?deep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• /api/operational/system-monitoring/vnf/port/stats/port-usage/&lt;duration&gt;?deep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• /api/operational/system-monitoring/vnf/port/stats/port-usage/&lt;duration&gt;/vnf?deep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• /api/operational/system-monitoring/vnf/port/stats/port-usage/&lt;duration&gt;/vnf/&lt;vnf-name&gt;?deep</td>
</tr>
</tbody>
</table>

The valid duration can be: 1min, 5min, 15min, 30min, 1h, 1H, 6h, 6H, 1d, 1D, 5d, 5D, 30d, and 30D

## Example: GET VNF Port Stats API

This example is for all VMs.

```
curl -k -v -u admin:Cisco123# -X GET https://209.165.201.2/api/operational/system-monitoring/vnf/port/stats/port-usage/5min?deep
```

Note: Unnecessary use of -X or --request, GET is already inferred.

* Trying 209.165.201.2...
* Connected to 209.165.201.2 (209.165.201.2) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
Example: GET VNF Port Stats API

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software

* start date: Feb 18 12:04:07 2017 GMT
* expire date: Feb 16 12:04:07 2027 GMT
* issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'

> GET /api/operational/system-monitoring/vnf/port/stats/port-usage/5min?deep HTTP/1.1
> Host: 209.165.201.2
> Authorization: Basic YWRtaW46Q2lzY28xMjM=
> User-Agent: curl/7.49.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.10.1
< Date: Wed, 22 Feb 2017 06:14:09 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
<
<port-usage xmlns="http://www.cisco.com/nfvos/system-monitoring"
 xmlns:y="http://tail-f.com/ns/rest"
 xmlns:system_monitoring="http://www.cisco.com/nfvos/system-monitoring">
<duration>5min</duration>
<vnf>
<name>1487397034.OTHER</name>
<collect-start-date-time>2017-02-22T06:09:00-00:00</collect-start-date-time>
<collect-interval-seconds>10</collect-interval-seconds>
<port>
<port-name>vnic0</port-name>
<tx-packets-per-sec>[0.08, 0.16, 0.06, 0.08, 0.16, 0.06, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12, 0.18, 0.0, 0.12]"n" automated forces any way.
Example: GET VNF Port Stats API

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
Table 73: Field Description for VNF Port Stats API Response

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>total-packets-per-sec</td>
<td>Total packets received and sent per second</td>
</tr>
<tr>
<td>rx-packets-per-sec</td>
<td>Packets received per second</td>
</tr>
<tr>
<td>tx-packets-per-sec</td>
<td>Packets sent per second</td>
</tr>
<tr>
<td>total-errors-per-sec</td>
<td>Total error rate (for packet reception and transmission)</td>
</tr>
<tr>
<td>rx-errors-per-sec</td>
<td>Error rate for receiving packets</td>
</tr>
<tr>
<td>tx-errors-per-sec</td>
<td>Error rate for sending packets</td>
</tr>
</tbody>
</table>
Example: GET VNF Port Stats API
External Disks API

Table 74: External Disks API

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get a list of external disks</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/ext-disks</td>
</tr>
</tbody>
</table>

Example: GET External Disks API

curl -X GET -v -k -u admin:admin https://1.2.3.4/api/operational/system/ext-disks
  * Trying 172.19.147.237...
  * Connected to 172.19.147.237 (172.19.147.237) port 443 (#0)
  * TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  * Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * Server auth using Basic with user 'admin'
  > GET /api/operational/system/ext-disks HTTP/1.1
  > Host: 172.19.147.237
  > Authorization: Basic YWRtaW46YWRtaW4=
  > User-Agent: curl/7.43.0
  > Accept: */*
  >
  < HTTP/1.1 204 No Content
  < Server: nginx/1.6.3
  < Date: Fri, 26 Aug 2016 23:03:50 GMT
  < Content-Type: application/vnd.yang.collection+xml
  < Content-Length: 0
  < Connection: keep-alive
File List APIs

Use the File List APIs to get information about all files under the "/mnt-usb" (USB) and "/data/upload1" (local) folders.

Table 75: File List APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get a list of VM images available for registration on the USB</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/file-list/disk/usb</td>
</tr>
<tr>
<td>To get a list of VM images available for registration on the local system</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/file-list/disk/local</td>
</tr>
</tbody>
</table>

Example: GET File List APIs


* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (40)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> GET /api/operational/system/file-list/disk/usb HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28jMTIz
> User-Agent: curl/7.43.0
> Accept:application/vnd.yang.collection+json
> Content-Type:application/vnd.yang.collection+json
>
< HTTP/1.1 200 OK
< Server: nginx/1.10.1
< Date: Wed, 22 Feb 2017 12:12:11 GMT
< Content-Type: application/vnd.yang.collection+json
< Transfer-Encoding: chunked
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Pragma: no-cache
<
{
  "collection": {
    "system:usb": [
      {
        "name": "win2k.tar.gz",
        "path": "/mnt-usb/usb1/Win2k",
      }
    ]
  }
}
NFVIS Related APIs

```
"size": "5.1G",
"type": "VM Package",
"date-modified": "2016-04-06 12:07:52"
},
{
"name": "CentOS-7-x86_64-Everything-1511.tar.gz",
"path": "/mnt-usb/usb1/OtherLin",
"size": "439M",
"type": "VM Package",
"date-modified": "2016-01-19 12:47:38"
},
{
"name": "ubuntu-14.04.3-server-amd64-disk1.tar.gz",
"path": "/mnt-usb/usb1/OtherLin",
"size": "527M",
"type": "VM Package",
"date-modified": "2016-01-19 12:46:30"
},
{
"name": "Cisco_NFVIS-3.4.0-454-20160927_022810.iso",
"path": "/mnt-usb/usb1",
"size": "1.8G",
"type": "Other",
"date-modified": "2016-09-27 02:06:48"
},
{
"name": "asav961.tar.gz",
"path": "/mnt-usb/usb1",
"size": "164M",
"type": "VM Package",
"date-modified": "2016-10-07 14:20:52"
},
{
"name": "Cisco-KVM-vWAAS-2500-6.2.1-b-11.tar.gz",
"path": "/mnt-usb/usb1",
"size": "919M",
"type": "VM Package",
"date-modified": "2016-10-07 14:19:24"
},
{
"name": "TinyLinux.tar.gz",
"path": "/mnt-usb/usb1",
"size": "17M",
"type": "VM Package",
"date-modified": "2016-01-19 11:23:14"
},
{
"name": "Cisco-KVM-vWAAS-2500-6.3.0-b98.tar.gz",
"path": "/mnt-usb/usb1",
"size": "979M",
"type": "VM Package",
"date-modified": "2016-12-05 10:29:52"
},
{
"name": "IndexerVolumeGuid",
"path": "/mnt-usb/usb1/System Volume Information",
"size": "76",
"type": "Other",
"date-modified": "2017-02-06 11:05:38"
},
{
"name": "isrv-universalk9.16.03.01.tar.gz",
"path": "/mnt-usb/usb2",
"size": "1.1G",
```

Example: GET File List APIs

File Delete API

Table 76: File Delete API

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
</table>

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
Delete one or more files from the host server (/data/upload1/)

| POST | Yes | /api/operations/system/file-delete/file |

Example for File Delete Payload

<input><name><xyz.txt></name></input>

Table 77: File Delete Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the file that you want to delete.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example: File Delete API

curl -k -v -u "admin:admin" -H content-type:application/vnd.yang.data+json -X POST https://209.165.201.1/api/operations/system/file-delete/file -d "<input><name>xyz.txt</name></input>"

Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
  * Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
  * successfully set certificate verify locations:
    * CAfile: /etc/pki/tls/certs/ca-bundle.crt
    * CApath: none
  * TLSv1.0 (OUT), TLS handshake, Client hello (1):
  * TLSv1.0 (IN), TLS handshake, Server hello (2):
  * TLSv1.0 (IN), TLS handshake, Certificate (11):
  * TLSv1.0 (IN), TLS handshake, Server key exchange (12):
  * TLSv1.0 (IN), TLS handshake, Server finished (14):
  * TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
  * TLSv1.0 (OUT), TLS change cipher, Client hello (1):
  * TLSv1.0 (OUT), TLS handshake, Finished (20):
  * TLSv1.0 (IN), TLS change cipher, Client hello (1):
  * TLSv1.0 (IN), TLS handshake, Finished (20):
  * SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
  * Server certificate:
    * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
    * start date: Oct 21 07:43:27 2016 GMT
    * expire date: Oct 19 07:43:27 2026 GMT
    * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
    * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
> POST /api/operations/system/file-delete/file HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.49.1
> Accept: */*
> content-type:application/vnd.yang.data+json
> Content-Length: 34
> *
upload completely sent off: 34 out of 34 bytes
< HTTP/1.1 204 No Content
USB Mount API

The supported USB formats are FAT32 and exFAT.

Table 78: USB Mount API

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To mount a USB drive on a server that supports Cisco Enterprise NFVIS</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/system/usb/mount</td>
</tr>
<tr>
<td>To unmount a USB drive from an NFVIS server</td>
<td>POST</td>
<td>No</td>
<td>/api/operations/system/usb/unmount</td>
</tr>
<tr>
<td>To view list of mount points</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/usb/mnt-info</td>
</tr>
</tbody>
</table>

Example for USB Mount Payload

<mount>ACTIVE</mount>

Table 79: USB Mount Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mount</td>
<td>string</td>
<td>Mounts the USB drive. You can copy files from the USB drive only after mounting the USB drive.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example: POST USB Mount API

Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
Example: POST USB Unmount API

Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'
> POST /api/operations/system/usb/unmount HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.49.1
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
>
< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Tue, 31 Jan 2017 22:25:38 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 209.165.201.1 left intact
Example: GET USB Mount Point

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'

> GET /api/operational/system/usb/mnt-info HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.49.1
> Accept: application/vnd.yang.collection+xml
> Content-Type: application/vnd.yang.data+xml

>

< HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Tue, 31 Jan 2017 23:53:41 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 209.165.201.1 left intact
USB Copy API

Table 80: USB Copy API

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy a single file from a mounted USB drive to the local folder of the server (/data/upload1/)</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/system/file-copy/usb/file</td>
</tr>
</tbody>
</table>

Example for USB Copy Payload

$input><name><path_of_file_relative_to_usb/example_file.txt></name></input>

Table 81: USB Copy Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the file with complete path relative to USB Path of the file within the USB drive. For example, if the file in the USB drive is like the following: images/isrv.tar.gz—The name parameter in payload must be &quot;images/isrv.tar.gz&quot;. asav.tar.gz—The name parameter in payload must be &quot;asav.tar.gz&quot;.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example: POST USB Copy API

```
curl -k -v -u admin:admin -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
```
Host Reboot API

Table 82: Host Reboot API

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To reboot the host server</td>
<td>POST</td>
<td>No</td>
<td>/api/operations/hostaction/reboot</td>
</tr>
</tbody>
</table>

DHCP Renew API

Table 83: WAN DHCP Renew API

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To renew the DHCP IP address on the WAN bridge</td>
<td>POST</td>
<td>No</td>
<td>/api/operations/hostaction/wan-dhcp-renew</td>
</tr>
<tr>
<td>To renew DHCP on bridge</td>
<td>POST</td>
<td>Yes</td>
<td>/api/operations/hostaction/bridge-dhcp-renew/bridge/br_name</td>
</tr>
</tbody>
</table>

Example: POST WAN DHCP Renew API

```
```

- Hostname was NOT found in DNS cache
- Trying 209.165.201.1...
- Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
- Successfully set certificate verify locations:
  - CAfile: none
  - CApath: /etc/ssl/certs
- SSLv3, TLS handshake, Client hello (1):
- SSLv3, TLS handshake, Server hello (2):
- SSLv3, TLS handshake, CERT (11):
Example: Bridge DHCP Renew

```bash
curl -k -v -u admin:admin -H "Accept:application/vnd.yang.data+json" -H "Content-Type:application/vnd.yang.data+json" -X POST https://localhost/api/operations/hostaction/bridge-dhcp-renew/bridge/test-br
```
Example: Bridge DHCP Renew
CHAPTER 9

SPAN Session and Packet Capture APIs

Table 84: SPAN Session APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To create a SPAN session</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/monitor</td>
</tr>
<tr>
<td>To get the SPAN monitor session status</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/monitor?deep</td>
</tr>
<tr>
<td>To get the SPAN session configuration details</td>
<td>GET</td>
<td>No</td>
<td>/api/config/monitor?deep</td>
</tr>
<tr>
<td>to get the SPAN session operational status</td>
<td>GET</td>
<td>No</td>
<td>• /api/operational/system/monitor/session</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• /api/operational/system/monitor/session?deep</td>
</tr>
</tbody>
</table>

Example for a SPAN Session Payload

```xml
<session>
  <number>20</number>
  <destination>
    <vm-vnic>
      <vm-name>Linux2</vm-name>
      <vnic-id>0</vnic-id>
    </vm-vnic>
  </destination>
  <source>
    <interfaces>
      <vm-vnic>
        <vm-name>Linux1</vm-name>
        <vnic-id>0</vnic-id>
        <direction>both</direction>
      </vm-vnic>
      <interface>
        <name>GEO-0</name>
        <direction>both</direction>
      </interface>
    </interfaces>
  </source>
</session>
```
Table 85: SPAN Session Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Integer</td>
<td>SPAN session number</td>
<td>Yes</td>
</tr>
<tr>
<td>destination</td>
<td>String</td>
<td>Destination for the mirrored traffic</td>
<td>Yes</td>
</tr>
<tr>
<td>vm-name</td>
<td>String</td>
<td>Name of the VM</td>
<td>Yes</td>
</tr>
<tr>
<td>vnic-id</td>
<td>String</td>
<td>Virtual network interface controller ID</td>
<td>Yes</td>
</tr>
<tr>
<td>source</td>
<td>String</td>
<td>Source the mirrored traffic</td>
<td>Yes</td>
</tr>
<tr>
<td>direction</td>
<td>String</td>
<td>Direction of the traffic</td>
<td>Yes</td>
</tr>
<tr>
<td>interface</td>
<td>String</td>
<td>Source or destination interface.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Example: POST SPAN Session API, on page 164
- Example: GET SPAN Session APIs, on page 165
- Packet Capture APIs, on page 168

Example: POST SPAN Session API

curl -v -u admin:XXXX -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -k -X POST https://209.165.201.1/api/config/monitor -d '<session><number>20</number><destination><vm-vnic><vm-name>Linux2</vm-name><vnic-id>0</vnic-id></vm-vnic></destination><source><interfaces><vm-vnic><vm-name>Linux1</vm-name><vnic-id>0</vnic-id><direction>both</direction></vm-vnic><interface><name>GE0-0</name><direction>both</direction></interface></interfaces></source></session>'

Note: Unnecessary use of -X or --request, POST is already inferred.
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAtfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApth: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
Example: GET SPAN Session APIs

Use this operational API to get the SPAN monitor session status.


---

NFVIS Related APIs

Example: GET SPAN Session APIs

- SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
- Server certificate:
  - subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  - start date: Mar 13 23:55:53 2017 GMT
  - expire date: Mar 11 23:55:53 2027 GMT
- issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
- SSL certificate verify result: self signed certificate (18), continuing anyway.
- Server auth using Basic with user 'admin'
- POST /api/config/monitor HTTP/1.1
- Authorization: Basic YWRtaW46TXlUZXN0MTIzIw==
- User-Agent: curl/7.50.1
- Accept: application/vnd.yang.data+xml
- Content-Type: application/vnd.yang.data+xml
- Content-Length: 330

- upload completely sent off: 330 out of 330 bytes
- HTTP/1.1 201 Created
- Server: nginx/1.10.1
- Date: Wed, 15 Mar 2017 02:42:25 GMT
- Content-Type: text/html
- Content-Length: 0
- Connection: keep-alive
- Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
- Etag: 1489-545745-460682
- Pragma: no-cache
- sj22lab-as2:145

---

Example: GET SPAN Session APIs

Use this operational API to get the SPAN monitor session status.


---

NFVIS Related APIs

Example: GET SPAN Session APIs

- SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
- Server certificate:
  - subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  - start date: Mar 13 23:55:53 2017 GMT
  - expire date: Mar 11 23:55:53 2027 GMT
- issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
- SSL certificate verify result: self signed certificate (18), continuing anyway.
- Server auth using Basic with user 'admin'
- POST /api/config/monitor HTTP/1.1
- Authorization: Basic YWRtaW46TXlUZXN0MTIzIw==
- User-Agent: curl/7.50.1
- Accept: application/vnd.yang.data+xml
- Content-Type: application/vnd.yang.data+xml
- Content-Length: 330

- upload completely sent off: 330 out of 330 bytes
- HTTP/1.1 201 Created
- Server: nginx/1.10.1
- Date: Wed, 15 Mar 2017 02:42:25 GMT
- Content-Type: text/html
- Content-Length: 0
- Connection: keep-alive
- Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
- Etag: 1489-545745-460682
- Pragma: no-cache
- sj22lab-as2:145

---
Example: GET SPAN Session APIs

Use this GET API to get the SPAN session configuration details.

```
curl -v -u admin:XXXXX -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -k -X GET https://209.165.201.1/api/config/monitor?deep
```

Note: Unnecessary use of -X or --request, GET is already inferred.

* * *
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Mar 13 23:55:53 2017 GMT
  * expire date: Mar 11 23:55:53 2027 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'
> GET /api/config/monitor?deep HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46TXlUZXN0MTIzIw==
> User-Agent: curl/7.50.1
> Accept: application/vnd.yang.data+xml
> Content-Type: application/vnd.yang.data+xml
>
< HTTP/1.1 200 OK
< Server: nginx/1.10.1
< Date: Wed, 15 Mar 2017 04:39:29 GMT
< Content-Type: application/vnd.yang.data+xml
< Transfer-Encoding: chunked
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1489-545745-460682
< Pragma: no-cache
<
xmns:span-session="http://www.cisco.com/nfv/span_session">
    <session>
      <number>20</number>
      <source>
        <interfaces>
          <vm-vnic>
            <vm-name>Linux1</vm-name>
            <vnic-id>0</vnic-id>
            <direction>both</direction>
          </vm-vnic>
          <interface>
            <name>GE0-0</name>
            <direction>both</direction>
          </interface>
        </interfaces>
      </source>
      <destination>
        <vm-vnic>
          <vm-name>Linux2</vm-name>
          <vnic-id>0</vnic-id>
        </vm-vnic>
      </destination>
    </session>
</monitor>
Packet Capture APIs

Table 86: Packet Capture APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To configure packet capture on a physical or virtual network interface controller</td>
<td>POST</td>
<td>Yes</td>
<td>api/operations/packet-capture/tcpdump</td>
</tr>
</tbody>
</table>

Example for the Packet Capture Payload for a Physical Port

```
<input>
  <port>eth0</port>
  <time>10</time>
</input>
```

Example for the Packet Capture Payload for a vNIC

```
<input>
  <vnic>
    <tenant-name>admin</tenant-name>
    <deployment-name>1489084431</deployment-name>
    <vm-name>ROUTER</vm-name>
    <vnic-id>0</vnic-id>
  </vnic>
  <time>10</time>
</input>
```

Table 87: Packet Capture Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>String</td>
<td>Physical or virtual network interface controller</td>
<td>Yes</td>
</tr>
<tr>
<td>time</td>
<td>String</td>
<td>Time period over which packets are captured. The default value is 60 seconds.</td>
<td>Yes</td>
</tr>
<tr>
<td>tenant-name</td>
<td>String</td>
<td>Name of the tenant</td>
<td>Yes</td>
</tr>
<tr>
<td>deployment-name</td>
<td>String</td>
<td>Name of the VM deployment</td>
<td>Yes</td>
</tr>
<tr>
<td>vm-name</td>
<td>String</td>
<td>Name of the VM</td>
<td>Yes</td>
</tr>
<tr>
<td>vnic-id</td>
<td>Integer</td>
<td>Virtual network interface controller ID</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Example: POST Packet Capture APIs

Use this POST API to configure packet capture on a physical port.

curl -v -k -u admin:Cisco123# -H "Content-Type: application/vnd.yang.data+xml" -H "Accept: application/vnd.yang.data+xml" -X POST https://209.165.201.1/api/operations/packet-capture/tcpdump -d '<input><port>eth0</port><time>10</time></input>'

* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'

> POST /api/operations/packet-capture/tcpdump HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.43.0
> Content-Type: application/vnd.yang.data+xml
> Accept: application/vnd.yang.data+xml
> Content-Length: 47
>
* upload completely sent off: 47 out of 47 bytes
< HTTP/1.1 200 OK
< Server: nginx/1.10.1
< Date: Wed, 29 Mar 2017 20:35:50 GMT
< Content-Type: application/vnd.yang.operation+xml
< Content-Length: 151
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Vary: Accept-Encoding
< Pragma: no-cache
<
<output xmlns='http://www.cisco.com/nfvos/packet_capture'>
<pcap-location>/data/intdatastore/pktcaptures/tcpdump_eth0.pcap</pcap-location>
</output>

* Connection #0 to host 209.165.201.1 left intact
Use this POST API to configure packet capture on a vNIC.

curl -v -k -u admin:Cisco123# -H "Content-Type: application/vnd.yang.data+xml" -H "Accept: application/vnd.yang.data+xml" -X POST https://209.165.201.1/api/operations/packet-capture/tcpdump -d ' <input> <vnic> <tenant-name>admin</tenant-name> <deployment-name>1489084431</deployment-name> <vm-name>ROUTER</vm-name> <vnic-id>0</vnic-id> </vnic> <time>10</time> </input>

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* Server auth using Basic with user 'admin'

> POST /api/operations/packet-capture/tcpdump HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.43.0
> Content-Type: application/vnd.yang.data+xml
> Accept: application/vnd.yang.data+xml
> Content-Length: 47

> * upload completely sent off: 47 out of 47 bytes
< HTTP/1.1 200 OK
< Server: nginx/1.10.1
< Date: Wed, 29 Mar 2017 20:35:50 GMT
< Content-Type: application/vnd.yang.operation+xml
< Content-Length: 151
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Vary: Accept-Encoding
< Pragma: no-cache
<
<output xmlns="/data/intdatastore/pktcaptures/1489084431_ROUTER_vnic0.pcaphttp://www.cisco.com/nfvoa/packet_capture">
<pcap-location>/data/intdatastore/pktcaptures/1489084431_ROUTER_vnic0.pcap</pcap-location>
</output>

* Connection #0 to host 209.165.201.1 left intact
CHAPTER 10

Upgrade Package APIs

- Upgrade Package Register API, on page 171
- Upgrade Apply-Image APIs, on page 175

Upgrade Package Register API

Table 88: Upgrade Package Register API

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To register a package for upgrade</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/system/upgrade</td>
</tr>
<tr>
<td>To view registered packages</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/upgrade/reg-info</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/operational/system/upgrade/reg-info?deep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/config/system/upgrade</td>
</tr>
<tr>
<td>To delete a registered package</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/system/upgrade/image-name</td>
</tr>
</tbody>
</table>

Example for Upgrade Package Register Payload

```xml
<image-name>
  <name>test3</name>
  <location>/data/intdatastore/uploads/package/upgradepackage
    filename(.nfvispkg)</location>
</image-name>
```

Table 89: Upgrade Package Register Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>image-name</td>
<td>String</td>
<td>Name of the image</td>
<td>Yes</td>
</tr>
</tbody>
</table>
If only one upgrade-package (.nfvispkg) exists on NFVIS/data/intdatastore/uploads directory, it's not necessary to specify the upgrade package name, after path/data/intdatastore/uploads.

When multiple upgrade-packages (.nfvispkg) exist on NFVIS/data/intdatastore/uploads, users need to specify specific upgrade-package to be registered, after path/data/intdatastore/uploads.

Example: POST Upgrade Package Register API

```
```

* Trying 209.165.201.1...
  * Connected to 209.165.201.1 (209.165.201.1) port 443 (40)
  * Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
  * successfully set certificate verify locations:
    * CAlfile: /etc/pki/tls/certs/ca-bundle.crt
    * CApath: none
  * TLSv1.0 (OUT), TLS handshake, Client hello (1):
  * TLSv1.0 (IN), TLS handshake, Server hello (2):
  * TLSv1.0 (IN), TLS handshake, Certificate (11):
    * TLSv1.0 (IN), TLS handshake, Server key exchange (12):
    * TLSv1.0 (IN), TLS handshake, Server finished (14):
    * TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
    * TLSv1.0 (OUT), TLS change cipher, Client hello (1):
    * TLSv1.0 (OUT), TLS handshake, Finished (20):
    * TLSv1.0 (IN), TLS change cipher, Client hello (1):
    * TLSv1.0 (IN), TLS handshake, Finished (20):
    * SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
  * Server certificate:
Example: GET Upgrade Package Register API

curl -k -v -u admin:admin -H content-type:application/vnd.yang.data+json -X GET https://209.165.201.1/api/operational/system/upgrade/reg-info

Note: Unnecessary use of -X or --request, GET is already inferred.

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApith: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP/1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client hello (2):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Mar 31 02:47:22 2017 GMT
  * expire date: Mar 29 02:47:22 2027 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'

Example: GET Upgrade Package Register API
Example: DELETE Upgrade Package Register API

curl -k -v -u admin:admin -X DELETE
https://209.165.201.1/api/config/system/upgrade/image-name/nfvis-3.3.1

> * Trying 209.165.201.1...
> * Connected to 209.165.201.1 (209.165.201.1) port 443 (no timeout)
> * Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
> * successfully set certificate verify locations:
> * CAfile: /etc/pki/tls/certs/ca-bundle.crt  CPath: none
> * TLSv1.0 (OUT), TLS handshake, Client hello (1):
> * TLSv1.0 (IN), TLS handshake, Client hello (1):
> * TLSv1.0 (IN), TLS handshake, Certificate (11):
> * TLSv1.0 (IN), TLS handshake, Server key exchange (12):
> * TLSv1.0 (IN), TLS handshake, Finished (14):
> * TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
> * TLSv1.0 (OUT), TLS change cipher, Client hello (1):
> * TLSv1.0 (OUT), TLS change cipher, Finished (20):
> * TLSv1.0 (IN), TLS change cipher, Client hello (1):
> * TLSv1.0 (IN), TLS change cipher, Finished (20):
> * SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
> * Server certificate:
> * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
> * start date: Aug 5 15:38:14 2016 GMT
> * expire date: Aug 3 15:38:14 2026 GMT
> * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
> * SSL certificate verify result: self signed certificate (18), continuing anyway.
> * Server auth using Basic with user 'admin'
> DELETE /api/config/system/upgrade/image-name/nfvis-3.3.1 HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.49.1
> Accept: */*
> HTTP/1.1 204 No Content
< Server: nginx/1.6.3
< Date: Fri, 05 Aug 2016 19:36:57 GMT
< Content-Type: text/html
< Content-Length: 0

Example: DELETE Upgrade Package Register API
Upgrade Apply-Image APIs

Table 90: Upgrade Apply-Image API

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To upgrade the existing image to a newly registered image</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/system/upgrade</td>
</tr>
<tr>
<td>To verify the upgrade status</td>
<td>GET</td>
<td>No</td>
<td>• /api/operational/system/upgrade/apply-image</td>
</tr>
<tr>
<td>To delete the upgraded image</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/system/upgrade/apply-image/&lt;image-name&gt;</td>
</tr>
</tbody>
</table>

Example for Upgrade Apply-Image Payload

```xml
<apply-image>
  <name>nfvis-3.3.1</name>
  <scheduled-time>24</scheduled-time>
</apply-image>
```

Table 91: Upgrade Apply-Image Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the image for the upgrade</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example: POST Upgrade Apply-Image API

```bash
curl -k -v -u admin:admin -H content-type:application/vnd.yang.data+json -X POST https://209.165.201.1/api/config/system/upgrade --data ´<apply-image> <name>nfvis-3.3.1</name> <scheduled-time>24</scheduled-time> </apply-image>´
```

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
Example: GET Upgrade Apply-Image API

curl -k -v -u admin:admin -H content-type:application/vnd.yang.data+json -X GET https://209.165.201.1/api/operational/system/upgrade/apply-image

Note: Unnecessary use of -X or --request, GET is already inferred.

Example: GET Upgrade Apply-Image API

curl -k -v -u admin:admin -H content-type:application/vnd.yang.data+json -X GET https://209.165.201.1/api/config/system/upgrade HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.49.1
> Accept: */*
> content-type:application/vnd.yang.data+json
> Content-Length: 53
>
> * upload completely sent off: 53 out of 53 bytes
< HTTP/1.1 201 Created
< Server: nginx/1.6.3
< Date: Fri, 05 Aug 2016 18:41:02 GMT
< Content-Type: text/html
< Content-Length: 0
< Location: https://209.165.201.1/api/config/system/upgrade/apply-image/nfvis=3.3.1
< Connection: keep-alive
< Last-Modified: Fri, 05 Aug 2016 18:41:02 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: "1470-422462-89670"
< Pragma: no-cache
<
> * Connection #0 to host 209.165.201.1 left intact
**Example: DELETE Upgrade Apply-Image API**

curl -k -v -u admin:admin -X DELETE
https://209.165.201.1/api/config/system/upgrade/apply-image/nfvis-3.3.1

* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt Cpath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Aug 5 15:38:14 2016 GMT
  * expire date: Aug 3 15:38:14 2026 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* SSL certificate verify result: self signed certificate (18), continuing anyway.

---

**NFVIS Related APIs**

* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: Mar 31 02:47:22 2017 GMT
  * expire date: Mar 29 02:47:22 2027 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
* SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'

> GET /api/operational/system/upgrade/apply-image HTTP/1.1
> Host: 209.165.201.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.50.1
> Accept: */*
> content-type:application/vnd.yang.data+json
>
< HTTP/1.1 200 OK
< Server: nginx/1.10.1
< Date: Fri, 31 Mar 2017 22:34:49 GMT
< Content-Type: application/vnd.yang.collection+xml
< Transfer-Encoding: chunked
< Connection: keep-alive
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Pragma: no-cache
<
<collection xmlns:y="http://tail-f.com/ns/rest">
  <apply-image xmlns="http://www.cisco.com/nfv">
    <name>Cisco_NFVIS_Upgrade-3.6.1-693-20170329_022604.nfvispkg</name>
    <scheduled-time>24</scheduled-time>
    <status>SCHEDULED</status>
  </apply-image>
</collection>

* Connection #0 to host 209.165.201.1 left intact
Example: DELETE Upgrade Apply-Image API

* Server auth using Basic with user 'admin'
  > DELETE /api/config/system/upgrade/apply-image/nfvis-3.3.1 HTTP/1.1
  > Host: 209.165.201.1
  > Authorization: Basic YWRtaW46YWRtaW4=
  > User-Agent: curl/7.49.1
  > Accept: */*
  >
  < HTTP/1.1 204 No Content
  < Server: nginx/1.6.3
  < Date: Fri, 05 Aug 2016 19:57:32 GMT
  < Content-Type: text/html
  < Content-Length: 0
  < Connection: keep-alive
  < Last-Modified: Fri, 05 Aug 2016 19:57:32 GMT
  < Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
  < Etag: 1470-427052-771331
  < Pragma: no-cache
  <
  * Connection #0 to host 209.165.201.1 left intact
CHAPTER 11

Factory Default Reset APIs

Table 92: Factory Default Reset APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
</table>
| To reset to factory default| POST   | No               | • /api/operations/factory-default-reset/all
|                            |        |                  | • /api/operations/factory-default-reset/all-except-images
|                            |        |                  | • /api/operations/factory-default-reset/all-except-images-connectivity |

• Example: POST Factory Default Reset All, on page 179
• Example: POST Factory Default Reset All Except Images, on page 180
• Example: POST Factory Default Reset All Except Images Connectivity, on page 181

Example: POST Factory Default Reset All

```bash
curl -v -u 'admin:Admin123$' -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X POST https://209.165.201.1/api/operations/factory-default-reset/all
```

* About to connect() to 209.165.201.1:443

* Connected to 209.165.201.1 (209.165.201.1) port 443

* SSL connection using EDH-RSA-DES-CBC3-SHA

* Server certificate:

* subject: /CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate

* start date: 2017-02-21 20:10:51 GMT

* expire date: 2027-02-19 20:10:51 GMT

* common name: Cisco-Enterprise-NFVIS-Self-Signed-Certificate (does not match '209.165.201.1')
Example: POST Factory Default Reset All Except Images

* About to connect() to 209.165.201.1:443
* Connected to 209.165.201.1 (209.165.201.1) port 443
* SSL connection using EDH-RSA-DES-CBC3-SHA

* Server certificate:
  * subject: /CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: 2017-02-21 20:10:51 GMT
  * expire date: 2027-02-19 20:10:51 GMT
  * common name: Cisco-Enterprise-NFVIS-Self-Signed-Certificate (does not match '209.165.201.1')

  * issuer: /CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate

> POST /api/operations/factory-default-reset/all HTTP/1.1
Authorization: Basic YWRtaW46QWRtaW4xMjMk
User-Agent: curl/7.9.6 (i686-pc-linux-gnu) libcurl 7.9.6 (OpenSSL 0.9.6)
Host: 209.165.201.1
Pragma: no-cache
Accept:application/vnd.yang.data+xml
Content-Type:application/vnd.yang.data+xml
Example: POST Factory Default Reset All Except Images Connectivity


* About to connect() to 209.165.201.1:443

* Connected to 209.165.201.1 (209.165.201.1) port 443

* SSL connection using EDH-RSA-DES-CBC3-SHA

* Server certificate:
  * subject: /CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: 2017-02-21 20:10:51 GMT
  * expire date: 2027-02-19 20:10:51 GMT
  * common name: Cisco-Enterprise-NFVIS-Self-Signed-Certificate (does not match '209.165.201.1')
  * issuer: /CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate

> POST /api/operations/factory-default-reset/all-except-images-connectivity HTTP/1.1

Authorization: Basic YWRtaW46QWRtaW4xMjMk
User-Agent: curl/7.9.6 (i686-pc-linux-gnu) libcurl 7.9.6 (OpenSSL 0.9.6)
Host: 209.165.201.1
Pragma: no-cache
Accept:application/vnd.yang.data+xml
Content-Type:application/vnd.yang.data+xml
Example: POST Factory Default Reset All Except Images Connectivity
Syslog Support APIs

Table 93: Syslog Support APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To configure syslog server</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/system/settings/logging</td>
</tr>
<tr>
<td>To update the syslog server</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/system/settings/logging/host/&lt;host-address&gt;</td>
</tr>
<tr>
<td>information</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To remove the syslog server</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/system/settings/logging/host/&lt;host-address&gt;</td>
</tr>
<tr>
<td>information</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To configure syslog severity</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/system/settings/logging/severity</td>
</tr>
<tr>
<td>To configure syslog facility</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/system/settings/logging/facility</td>
</tr>
<tr>
<td>To view syslog configuration</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/system/settings/logging</td>
</tr>
</tbody>
</table>

- Example: POST Syslog Server, on page 183
- Example: PUT Remote Logging Host Configuration, on page 184
- Example: DELETE Remote Logging Host Configuration, on page 185
- Example: PUT Syslog Severity, on page 186
- Example: PUT Syslog Facility, on page 187
- Example: GET Remote Logging Host, on page 188

Example: POST Syslog Server


Note: Unnecessary use of -X or --request, POST is already inferred.
* Trying 172.19.162.209...
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
* CAfile: /etc/pki/tls/certs/ca-bundle.crt
  CAnull: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
  * Server certificate:
    * subject: CN=nfvis
    * start date: Jun 2 18:39:33 2017 GMT
    * expire date: May 31 18:39:33 2027 GMT
    * issuer: CN=nfvis
    * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
  > POST /api/config/system/settings/logging HTTP/1.1
  > Host: 172.19.162.209
  > Authorization: Basic YWRtaW46Q2lzY28xMjMj
  > User-Agent: curl/7.49.1
  > Accept:application/vnd.yang.data+xml
  > Content-Type:application/vnd.yang.data+xml
  > Content-Length: 85
  > upload completely sent off: 85 out of 85 bytes
  < HTTP/1.1 201 Created
  < Server: nginx/1.10.1
  < Date: Thu, 08 Jun 2017 09:01:40 GMT
  < Content-Type: text/html
  < Content-Length: 0
  < Location: https://172.19.162.209/api/config/system/settings/logging/host/172.19.162.143
  < Connection: keep-alive
  < Last-Modified: Thu, 08 Jun 2017 09:01:40 GMT
  < Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
  < Etag: 1496-912500-289655
  < Pragma: no-cache
  <

Example: PUT Remote Logging Host Configuration

```
```

* Trying 172.19.162.209...
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
* CAfile: /etc/pki/tls/certs/ca-bundle.crt
  CAnull: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
Example: DELETE Remote Logging Host Configuration

```
```

* Trying 172.19.162.209...
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApth: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS handshake, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=nfvis
  * start date: Jun 2 18:39:33 2017 GMT
  * expire date: May 31 18:39:33 2027 GMT
  * issuer: CN=nfvis
* SSL certificate verify result: self signed certificate (18), continuing anyway.

Example: DELETE Remote Logging Host Configuration
Example: PUT Syslog Severity

curl -k -v -u admin:Cisco123# -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X PUT https://172.19.162.209/api/config/system/settings/logging/severity -d '<severity>error</severity>'

* Trying 172.19.162.209...
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
* CAfile: /etc/pki/tls/certs/ca-bundle.crt
* CApath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS change cipher, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=nfvis
  * start date: Jun 2 18:39:33 2017 GMT
  * expire date: May 31 18:39:33 2027 GMT
  * issuer: CN=nfvis
* SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'

HTTP/1.1 204 No Content
}

Example: PUT Syslog Severity

curl -k -v -u admin:Cisco123# -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X PUT https://172.19.162.209/api/config/system/settings/logging/severity -d '<severity>error</severity>'

* Trying 172.19.162.209...
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
* CAfile: /etc/pki/tls/certs/ca-bundle.crt
* CApath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Server hello (2):
* TLSv1.0 (IN), TLS handshake, Certificate (11):
* TLSv1.0 (IN), TLS handshake, Server key exchange (12):
* TLSv1.0 (IN), TLS handshake, Server finished (14):
* TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.0 (OUT), TLS change cipher, Client hello (1):
* TLSv1.0 (OUT), TLS change cipher, Finished (20):
* TLSv1.0 (IN), TLS change cipher, Client hello (1):
* TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=nfvis
  * start date: Jun 2 18:39:33 2017 GMT
  * expire date: May 31 18:39:33 2027 GMT
  * issuer: CN=nfvis
* SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'

HTTP/1.1 204 No Content

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
Example: PUT Syslog Facility

curl -k -v -u admin:Cisco123# -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X PUT https://172.19.162.209/api/config/system/settings/logging/facility -d 'facility=local5'</facility>

* Trying 172.19.162.209...
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
* CAfile: /etc/pki/tls/certs/ca-bundle.crt
* CApath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
  * TLSv1.0 (IN), TLS handshake, Server hello (2):
  * TLSv1.0 (IN), TLS handshake, Certificate (11):
  * TLSv1.0 (IN), TLS handshake, Server key exchange (12):
  * TLSv1.0 (IN), TLS handshake, Server finished (14):
  * TLSv1.0 (OUT), TLS handshake, Client key exchange (16):
  * TLSv1.0 (OUT), TLS change cipher, Client hello (1):
  * TLSv1.0 (OUT), TLS handshake, Finished (20):
  * TLSv1.0 (IN), TLS change cipher, Client hello (1):
  * TLSv1.0 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=nfvis
  * start date: Jun 2 18:39:33 2017 GMT
  * expire date: May 31 18:39:33 2027 GMT
* issuer: CN=nfvis
* SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'
> PUT /api/config/system/settings/logging/facility HTTP/1.1
> Host: 172.19.162.209
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.49.1
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml
> Content-Length: 27
>
> * upload completely sent off: 27 out of 27 bytes
< HTTP/1.1 204 No Content
< Server: nginx/1.10.1
< Date: Thu, 08 Jun 2017 09:20:21 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Thu, 08 Jun 2017 09:20:21 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1496-913621-135440
< Pragma: no-cache
<
Example: GET Remote Logging Host


Note: Unnecessary use of -X or --request, GET is already inferred.

* Trying 172.19.162.209...
* Connected to 172.19.162.209 (172.19.162.209) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CAdpath: none
* TLSv1.0 (OUT), TLS handshake, Client hello (1):
  * TLSv1.0 (IN), TLS handshake, Server hello (2):
  * TLSv1.0 (IN), TLS handshake, Certificate (11):
  * TLSv1.0 (IN), TLS handshake, Server key exchange (12):
  * TLSv1.0 (IN), TLS handshake, Server finished (14):
  * TLSv1.0 (OUT), TLS handshake, Client hello (1):
  * TLSv1.0 (OUT), TLS handshake, Finished (20):
  * TLSv1.0 (IN), TLS change cipher, Client hello (1):
  * TLSv1.0 (IN), TLS handshake, Finished (20):
  * SSL connection using TLSv1.0 / DHE-RSA-AES256-SHA
  * Server certificate:
    * subject: CN=nfvis
    * start date: Jun 2 18:39:33 2017 GMT
    * expire date: May 31 18:39:33 2027 GMT
    * issuer: CN=nfvis
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'

> GET /api/operational/system/settings/logging?deep HTTP/1.1
> Host: 172.19.162.209
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.49.1
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+xml

< HTTP/1.1 200 OK
< Server: nginx/1.10.1
< Date: Thu, 08 Jun 2017 09:03:37 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

  <host>
    <host>172.19.162.117</host>
    <transport>
      <tcp/>
    </transport>
    <port>1635</port>
  </host>
</logging>
</transport>
<port>163</port>
</host>
<host>
<host>172.19.162.112</host>
<port>1523</port>
</host>
</host>
<host>
<host>172.19.162.143</host>
<transport>
<udp/>
</transport>
<port>525</port>
</host>
</logging>
Example: GET Remote Logging Host
SNMP Support APIs

Table 94: SNMP Support APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To configure communities</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/snmp/communities</td>
</tr>
<tr>
<td>To enable SNMP traps</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/snmp/enable/traps</td>
</tr>
<tr>
<td>To configure SNMP hosts</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/snmp/hosts</td>
</tr>
<tr>
<td>To configure SNMP users</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/snmp/users</td>
</tr>
<tr>
<td>To configure SNMP groups</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/snmp/groups</td>
</tr>
<tr>
<td>To view SNMP configuration</td>
<td>GET</td>
<td>No</td>
<td>/api/config/snmp/agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/config/snmp/communities</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/config/snmp/enable/traps</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/config/snmp/hosts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/config/snmp/users</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/config/snmp/groups</td>
</tr>
</tbody>
</table>

SNMP Agent is enabled by default and therefore always enabled.

- Example: POST Configuring SNMP Communities, on page 192
- Example: POST SNMP Traps, on page 192
- Example: POST SNMP Host, on page 192
- Example: POST SNMP Users, on page 193
- Example: POST SNMP Groups, on page 193
- Example: GET SNMP Configurations, on page 193
Example: POST Configuring SNMP Communities


Table 95: Field Descriptions for SNMP Communities

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>community-name</td>
<td>Upto 32 char alphanumeric string (including _ and -)</td>
</tr>
<tr>
<td>community-access</td>
<td>Read-only</td>
</tr>
</tbody>
</table>

Example: POST SNMP Traps

curl -k -v -u "admin:XXX" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X POST https://172.19.162.235/api/config/snmp/enable/traps -d ' <trap-type>linkDown</trap-type>'

Table 96: Field Description for SNMP trap API

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trap-type</td>
<td>linkUp or linkDown</td>
</tr>
</tbody>
</table>

Example: POST SNMP Host

curl -k -v -u "admin:XXX" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X POST https://172.19.162.235/api/config/snmp/hosts -d ' <host> <host-name>listen_host7</host-name> <host-port>162</host-port> <host-ip-address>10.32.172.190</host-ip-address> <host-version>2</host-version> <host-security-level>noAuthNoPriv</host-security-level> <host-user-name>user1</host-user-name> </host>'

curl -k -v -u "admin:XXX" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X POST
Example: POST SNMP Users

```
<user>
  <user-name>test_user1</user-name>
  <user-version>2</user-version>
  <user-group>public</user-group>
</user>
```

Table 97: Field Description for SNMP User API

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>passphrase</td>
<td>Alphanumeric string with 8 character minimum length.</td>
</tr>
<tr>
<td>auth-protocol</td>
<td>md5 or sha</td>
</tr>
<tr>
<td>priv-protocol</td>
<td>aes or des</td>
</tr>
</tbody>
</table>

Example: POST SNMP Groups

```
curl -k -v -u "admin:XXX" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X POST https://172.19.162.235/api/config/snmp/groups -d ' 
<group>
  <group-name>testgroup2</group-name>
  <group-context-prefix>snmp</group-context-prefix>
  <group-version>2</group-version>
  <security-level>noAuthNoPriv</security-level>
  <read>read-access</read>
  <write>write-access</write>
  <notify>notify-access</notify>
</group>'
```

Example: GET SNMP Configurations

```
```
Example: GET SNMP Configurations
CHAPTER 14

TACACS and RADIUS Support APIs

- TACACS Support APIs, on page 195
- RADIUS Support APIs, on page 199

TACACS Support APIs

Table 98: TACACS Support APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To configure TACACS server</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/security_servers/tacacs-server/</td>
</tr>
<tr>
<td>To configure TACACS server</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/security_servers/tacacs-server/</td>
</tr>
<tr>
<td>To configure TACACS server</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/security_servers/tacacs-server/</td>
</tr>
<tr>
<td>To view TACACS server</td>
<td>GET</td>
<td>No</td>
<td>/api/config/security_servers/tacacs-server/</td>
</tr>
</tbody>
</table>

Example: POST TACACS Server

```
curl -k -v -u "admin:cisco123" -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+json -X POST
https://209.165.201.1/api/config/security_servers/tacacs-server -d '{"host":
"server":5.5.5.5", "secret": {"key": "0", "shared-secret": "heyworld", "admin-priv": "14", "oper-priv": "10"}}'}
```

* Hostname was NOT found in DNS cache
* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* successfully set certificate verify locations:
  * CAfile: none
  * CPath: /etc/ssl/certs
* SSLv3, TLS handshake, Client hello (1):
* SSLv3, TLS handshake, Server hello (2):
Example: PUT TACACS Server

```
curl -k -v -u "admin:cisco123" -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+json -X PUT https://209.165.201.1/api/config/security_servers/tacacs-server/host/5.5.5.5 -d '"host":{"server":"5.5.5.5", "secret": {"shared-secret":"helloworld", "admin-priv": "15"}}'
```
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSL connection using ECDHE-RSA-AES256-GCM-SHA384
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: 2017-01-13 23:47:41 GMT
  * expire date: 2027-01-11 23:47:41 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'

> PUT /api/config/security_servers/tacacs-server/host/5.5.5.5 HTTP/1.1
> Authorization: Basic YWRtaW46Y2lzY28xMjM=
> User-Agent: curl/7.35.0
> Host: 209.165.201.1
> Accept:application/vnd.yang.data+xml
> Content-Type:application/vnd.yang.data+json
> Content-Length: 92
>
* upload completely sent off: 92 out of 92 bytes

< HTTP/1.1 204 No Content
* Server nginx/1.10.1 is not blacklisted
* Server nginx/1.10.1
* Date: Mon, 27 Feb 2017 18:20:13 GMT
* Content-Type: text/html
* Content-Length: 0
* Connection: keep-alive
* Last-Modified: Mon, 27 Feb 2017 18:20:13 GMT
* Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
* Etag: 1488-219613-571277
*Pragma: no-cache
<

Example: GET TACACS Server API

```
curl -k -v -u "admin:cisco123" -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+json -X GET https://209.165.201.1/api/config/security_servers/tacacs-server?deep
```

Hostname was NOT found in DNS cache
* Trying 209.165.201.1...
* Connected to 209.165.201.1 (209.165.201.1) port 443 (#0)
* successfully set certificate verify locations:
  * CAfile: none
  * CAPath: /etc/ssl/certs
* SSLv3, TLS handshake, Client hello (1):
* SSLv3, TLS handshake, Server hello (2):
* SSLv3, TLS handshake, CERT (11):
* SSLv3, TLS handshake, Server key exchange (12):
* SSLv3, TLS handshake, Server finished (14):
* SSLv3, TLS handshake, Client key exchange (16):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSL connection using ECDHE-RSA-AES256-GCM-SHA384
* Server certificate:
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * start date: 2017-01-13 23:47:41 GMT
  * expire date: 2027-01-11 23:47:41 GMT
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'

> GET /api/config/security_servers/tacacs-server?deep HTTP/1.1
Example: DELETE TACACS Server

curl -k -v -u "admin:cisco123" -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+json -X DELETE

https://209.165.201.1/api/config/security_servers/tacacs-server/host/5.5.5.5
* SSLv3, TLS handshake, Server key exchange (12):  
* SSLv3, TLS handshake, Server finished (14):  
* SSLv3, TLS handshake, Client key exchange (16):  
* SSLv3, TLS change cipher, Client hello (1):  
* SSLv3, TLS handshake, Finished (20):  
* SSL connection using ECDHE-RSA-AES256-GCM-SHA384  

* Server certificate:  
  * subject: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate  
  * start date: 2017-01-13 23:47:41 GMT  
  * expire date: 2027-01-11 23:47:41 GMT  
  * issuer: CN=Cisco-Enterprise-NFVIS-Self-Signed-Certificate  
* SSL certificate verify result: self signed certificate (18), continuing anyway.  
* Server auth using Basic with user 'admin'  
> DELETE /api/config/security_servers/tacacs-server/host/5.5.5.5 HTTP/1.1  
> Authorization: Basic YWRtaW46Y2lzY28xMjM=  
> User-Agent: curl/7.35.0  
> Host: 209.165.201.1  
> Accept:application/vnd.yang.data+xml  
> Content-Type:application/vnd.yang.data+json  
>  
< HTTP/1.1 204 No Content  
* Server nginx/1.10.1 is not blacklisted  
< Server: nginx/1.10.1  
< Date: Mon, 27 Feb 2017 18:21:30 GMT  
< Content-Type: text/html  
< Content-Length: 0  
< Connection: keep-alive  
< Last-Modified: Mon, 27 Feb 2017 18:21:30 GMT  
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate  
< Etag: 1488-219690-404414  
< Pragma: no-cache  

---

RADIUS Support APIs

### Table 99: RADIUS Support APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To configure RADIUS server</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/security_servers/radius-server/</td>
</tr>
<tr>
<td>To update configurations on RADIUS server</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/security_servers/radius-server/</td>
</tr>
<tr>
<td>To delete configurations on RADIUS server</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/security_servers/radius-server/</td>
</tr>
<tr>
<td>To view RADIUS server configuration</td>
<td>GET</td>
<td>No</td>
<td>/api/config/security_servers/radius-server/</td>
</tr>
</tbody>
</table>

---

Example: GET RADIUS Server

curl -k -v -u "admin:admin" -H Accept:application/vnd.yang.data+xml -H Content-
Example: POST RADIUS Server

curl -k -v -u "admin:admin" -H Accept:application/vnd.yang.data+json -H Content-Type:application/vnd.yang.data+xml -X POST
https://209.165.201.1/api/config/security_servers/radius-server -d '{"host":
{"server":"5.5.5.5", "secret": {"key": "0", "shared-secret": "heyworld", "admin-priv": "14", "oper-priv": "10"}}}'

Example: PUT RADIUS Server

curl -k -v -u "admin:cisco123" -H Accept:application/vnd.yang.data+json -H Content-Type:application/vnd.yang.data+xml -X PUT
https://209.165.201.1/api/config/security_servers/radius-server/host/5.5.5.5 -d '{"host":
{"server":"5.5.5.5", "secret": {"shared-secret":"helloworld", "admin-priv": "15"}}}'

Example: DELETE RADIUS Server

curl -k -v -u "admin:cisco123" -H Accept:application/vnd.yang.data+json -H Content-Type:application/vnd.yang.data+xml -X DELETE
https://209.165.201.1/api/config/security_servers/radius-server/host/5.5.5.5
# Port and Port Channel APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To show information about all ports including port channels</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/pnics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>/api/operational/pnics?deep</td>
</tr>
<tr>
<td>To create a port channel</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/pnics</td>
</tr>
<tr>
<td>To add a port to a port channel</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/pnics/pnic/name/member_of</td>
</tr>
<tr>
<td>To add a port channel to a new bridge</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/bridges/</td>
</tr>
<tr>
<td>To add a port channel to an existing bridge</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/bridges/bridge/bridgenname</td>
</tr>
<tr>
<td>To configure the LACP mode of a port channel</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/pnics/pnic/portchannel_name/lACP_type</td>
</tr>
<tr>
<td>To configure the bond mode of a port channel</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/pnics/pnic/portchannel_name/bond_mode</td>
</tr>
<tr>
<td>To configure trunks on a port channel</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/pnics/pnic/portchannelname/trunks</td>
</tr>
<tr>
<td>To remove a port from a port channel</td>
<td>DELETE</td>
<td>Yes</td>
<td>/api/config/pnics/pnic/portname/member_of</td>
</tr>
<tr>
<td>To remove a port channel from a bridge</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/bridges/bridge/bridgenname</td>
</tr>
<tr>
<td>Action</td>
<td>Method</td>
<td>Payload Required</td>
<td>API</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>--------</td>
<td>------------------</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>To delete a port channel</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/pnics/pnic/portchannelname</td>
</tr>
<tr>
<td><strong>Note</strong> Before deleting a port channel, you must remove all members assigned to the port channel. If the port channel is configured on the bridge, you must remove the port channel from the bridge.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| To enable or disable LLDP                   | PUT    | Yes              | /api/config/pnics/pnic/portname/lldp              |
| To show LLDP neighbors and stats           | GET    | No               | /api/operational/lldp                           |
|                                             |        |                  | /api/operational/lldp?deep                       |
| To configure the port admin status         | PUT    | Yes              | /api/config/pnics/pnic/portname/adminstatus       |
| To get information about the admin status of a port | GET    | No               | /api/config/pnics/pnic/portname/adminstatus       |

**Table 100: Ports and Port Channels APIs Payload Description**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>pnic name</td>
<td>String</td>
<td>Name of the port or port channel.</td>
<td>Yes</td>
<td>&lt;pnic&gt;&lt;name&gt;pc&lt;/name&gt;</td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>Type of the port. Valid values are ethernet and port_channel. To create a port channel, you must specify the value as port_channel.</td>
<td>Yes</td>
<td>&lt;type&gt;port_channel&lt;/type&gt;</td>
</tr>
<tr>
<td>lACP_type</td>
<td>String</td>
<td>The LACP type for a port channel. Valid values are off, active, and passive. Default is off.</td>
<td>No</td>
<td>&lt;lACP_type&gt;active&lt;/lACP_type&gt;</td>
</tr>
<tr>
<td>bond_mode</td>
<td>String</td>
<td>The bond mode for a port channel. Valid values are active-backup, balance-slb, and balance-tcp. Default is balance-tcp.</td>
<td>No</td>
<td>&lt;bond_mode&gt;balance-tcp&lt;/bond_mode&gt;</td>
</tr>
</tbody>
</table>
### Property Information

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>trunks</td>
<td>Integer</td>
<td>VLAN IDs. Valid range is from 1 to 4096. Default is VLAN 1. Enter VLANs separated by commas, VLAN ranges separated by dashes, or a combination of both.</td>
<td>No</td>
<td><code>&lt;trunks&gt;10,20&lt;/trunks&gt;</code></td>
</tr>
<tr>
<td>member_of</td>
<td>String</td>
<td>The name of the port channel to which you want to add a port or from which you want to remove a port.</td>
<td>Yes</td>
<td><code>&lt;member_of&gt;pc&lt;/member_of&gt;</code></td>
</tr>
<tr>
<td>port name</td>
<td>String</td>
<td>The name of the port channel that you want to add to the bridge or remove from the bridge.</td>
<td>Yes</td>
<td><code>&lt;port&gt;&lt;name&gt;pc&lt;/name&gt;&lt;/port&gt;</code></td>
</tr>
<tr>
<td>bridge name</td>
<td>String</td>
<td>The name of the bridge from which you want to remove the port channel or to which you want to add the port channel.</td>
<td>Yes</td>
<td><code>&lt;bridge&gt;&lt;name&gt;test-br&lt;/name&gt;&lt;/bridge&gt;</code></td>
</tr>
<tr>
<td>lldp</td>
<td>String</td>
<td>Enables or disables LLDP on a port. Valid values are enable and disable. Default is disable.</td>
<td>No</td>
<td><code>&lt;lldp&gt;enabled&lt;/lldp&gt;</code></td>
</tr>
<tr>
<td>adminstatus</td>
<td>String</td>
<td>Shuts down or brings up a port administratively. Valid values are up and down.</td>
<td>No</td>
<td><code>&lt;adminstatus&gt;up&lt;/adminstatus&gt;</code></td>
</tr>
</tbody>
</table>

- Example: GET Port and Port Channel Information API, on page 203
- Example: POST Create a Port Channel API, on page 207
- Example: PUT Add a Port to a Port Channel API, on page 209
- Example: PUT Add a Port Channel to an Existing Bridge API, on page 210
- Example: PUT Configure the LACP Mode of a Port Channel API, on page 211
- Example: PUT Configure the Bond Mode of a Port Channel API, on page 212
- Example: PUT Configure Trunks on a Port Channel API, on page 213
- Example: DELETE Remove a Port from a Port Channel API, on page 213
- Example: PUT Remove a Port Channel from a Bridge API, on page 214
- Example: DELETE Delete a Port Channel API, on page 215
- Example: GET LLDP Information API, on page 216
- Example: PUT Enable LLDP Configuration API, on page 219
- Example: GET Port Admin Status API, on page 220
- Example: PUT Configure Port Admin Status API, on page 221
- Speed, Autoneg and Duplex APIs, on page 222

### Example: GET Port and Port Channel Information API

```
curl -v -k -u admin:Admin#123 -X GET https://198.51.100.11/api/operational/pnics
```
* About to connect() to 198.51.100.11 port 443 (198.51.100.11)
* Trying 198.51.100.11...
Example: GET Port and Port Channel Information API

```xml
  <pnic>
    <name>eth0</name>
  </pnic>
  <pnic>
    <name>eth1</name>
  </pnic>
  <pnic>
    <name>eth2</name>
  </pnic>
  <pnic>
    <name>eth3</name>
  </pnic>
  <pnic>
    <name>eth4</name>
  </pnic>
  <pnic>
    <name>eth5</name>
  </pnic>
</pnics>
```

* Connected to 198.51.100.11 (198.51.100.11) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* SSL connection using TLS_DHE_RSA_WITH_AES_256_CBC_SHA
* Server certificate:
  * subject: CN=nfvis
  * start date: Dec 05 15:26:32 2017 GMT
  * expire date: Dec 03 15:26:32 2027 GMT
  * common name: nfvis
  * issuer: CN=nfvis
* Server auth using Basic with user 'admin'

```
curl -v -k -u admin:Admin#123 -X GET https://198.51.100.11/api/operational/pnics?deep
```
Example: GET Port and Port Channel Information API

> GET /api/operational/pnics?deep HTTP/1.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.29.0
> Host: 198.51.100.11
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx
< Date: Wed, 06 Dec 2017 17:47:14 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
>
<pnic>
  <name>eth0</name>
  <speed>1G</speed>
  <operational-speed>1000</operational-speed>
  <link_state>up</link_state>
  <mac_address>58:ac:78:59:ca:66</mac_address>
  <mtu>9216</mtu>
  <refcnt>0</refcnt>
  <stats>
    <receive>
      <bytes>62837229</bytes>
      <packets>496647</packets>
      <errors>0</errors>
      <dropped>0</dropped>
      <broadcast>60009</broadcast>
      <multicast>408172</multicast>
    </receive>
    <transmit>
      <bytes>517791</bytes>
      <packets>1565</packets>
      <errors>0</errors>
      <dropped>0</dropped>
      <broadcast>1462</broadcast>
      <multicast>12</multicast>
    </transmit>
  </stats>
</pnic>
<pnic>
  <name>eth1</name>
  <speed>1G</speed>
  <operational-speed>0</operational-speed>
  <link_state>down</link_state>
  <mac_address>58:ac:78:59:ca:67</mac_address>
  <mtu>9216</mtu>
  <refcnt>1</refcnt>
  <stats>
    <receive>
      <bytes>0</bytes>
      <packets>0</packets>
      <errors>0</errors>
      <dropped>0</dropped>
      <broadcast>0</broadcast>
      <multicast>0</multicast>
    </receive>
    <transmit>
      <bytes>0</bytes>
    </transmit>
  </stats>
</pnic>
</pnics>
<packets>0</packets>
<errors>0</errors>
<dropped>0</dropped>
<broadcast>0</broadcast>
<multicast>0</multicast>
</transmit>
</stats>
</pnic>
</pnic>

 Ethernet Interface Statistics:
- eth2:
  - Speed: 1Gbps
  - Operational Speed: 0bps
  - Link State: Down
  - MAC Address: a0:36:9f:7b:87:9c
  - MTU: 9216
  - Reference Count: 2
  - Receive:
    - Packets: 0
    - Errors: 0
    - Dropped: 0
    - Broadcast: 0
    - Multicast: 0
  - Transmit:
    - Packets: 0
    - Errors: 0
    - Dropped: 0
    - Broadcast: 0
    - Multicast: 0

- eth3:
  - Speed: 1Gbps
  - Operational Speed: 0bps
  - Link State: Down
  - MAC Address: a0:36:9f:7b:87:9d
  - MTU: 9216
  - Reference Count: 3
  - Receive:
    - Packets: 0
    - Errors: 0
    - Dropped: 0
    - Broadcast: 0
    - Multicast: 0
  - Transmit:
    - Packets: 0
    - Errors: 0
    - Dropped: 0
    - Broadcast: 0
    - Multicast: 0

- eth4:
Example: POST Create a Port Channel API

```
curl -k -v -u admin:Admin#123 -X POST -H Content-type:application/vnd.yang.data+xml
https://198.51.100.11/api/config/pnics
--data '<pnic><name>pc</name><type>port_channel</type></pnic>'
```
Example: POST Create a Port Channel API

```
curl -k -v -u admin:Admin#123 -X POST -H Content-type:application/vnd.yang.data+xml https://198.51.100.11/api/config/pnics --data
  <pnic><name>pc</name><type>port_channel</type><lacp_type>active</lacp_type><bond_mode>balance-tcp</bond_mode>
    <trunks>10,20</trunks></pnic>
```

Note: Unnecessary use of -X or --request, POST is already inferred.

```
* Trying 198.51.100.11...
* Connected to 198.51.100.11 (198.51.100.11) port 443 (#0)
* Initialized NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* SSL connection using TLS_DHE_RSA_WITH_AES_256_CBC_SHA
  * Server certificate:
    * subject: CN=nfvis
    * start date: Dec 05 15:26:32 2017 GMT
    * expire date: Dec 03 15:26:32 2027 GMT
    * common name: nfvis
    * issuer: CN=nfvis
  * Server auth using Basic with user 'admin'
> POST /api/config/pnics HTTP/1.1
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.29.0
> Host: 198.51.100.11
> Accept: */*
> Content-type:application/vnd.yang.data+xml
> Content-Length: 53
>
* upload completely sent off: 53 out of 53 bytes
< HTTP/1.1 201 Created
< Server: nginx
< Date: Wed, 06 Dec 2017 17:48:44 GMT
< Content-Type: text/html
< Content-Length: 0
< Location: https://198.51.100.11/api/config/pnics/pnic/pc
< Connection: keep-alive
< Last-Modified: Wed, 06 Dec 2017 17:48:44 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Pragma: no-cache
<
* Connection #0 to host 198.51.100.11 left intact
```
Example: PUT Add a Port to a Port Channel API

```
curl -k -v -u admin:Admin#123 -X PUT -H Content-type:application/vnd.yang.data+xml
https://198.51.100.11/api/config/pnics/pnic/eth1/member_of --data 'member_of=pc'
```

NFVIS Related APIs

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software

Example: PUT Add a Port to a Port Channel API
Example: PUT Add a Port Channel to an Existing Bridge API

```
curl -k -v -u admin:Admin#123 -X PUT -H Content-type:application/vnd.yang.data+xml https://198.51.100.11/api/config/bridges/bridge/test-br --data '<bridge><name>test-br</name><port><name>pc</name></port></bridge>'
```

Example: PUT Add a Port Channel to an Existing Bridge API

```
curl -k -v -u admin:Admin#123 -X PUT -H Content-type:application/vnd.yang.data+xml https://198.51.100.11/api/config/bridges/bridge/test-br --data '<bridge><name>test-br</name><port><name>pc</name></port></bridge>'
```

API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software
Example: PUT Configure the LACP Mode of a Port Channel API

curl -k -v -u admin:Admin#123 -X PUT -H Content-type:application/vnd.yang.data+xml https://198.51.100.11/api/config/pnics/pnic/pc/lacp_type --data '<lacp_type>active</lacp_type>'

Example: PUT Configure the LACP Mode of a Port Channel API

curl -k -v -u admin:Admin#123 -X PUT -H Content-type:application/vnd.yang.data+xml https://198.51.100.11/api/config/pnics/pnic/pc/lacp_type --data '<lacp_type>active</lacp_type>'

Example: PUT Configure the LACP Mode of a Port Channel API

curl -k -v -u admin:Admin#123 -X PUT -H Content-type:application/vnd.yang.data+xml https://198.51.100.11/api/config/pnics/pnic/pc/lacp_type --data '<lacp_type>active</lacp_type>'

Example: PUT Configure the LACP Mode of a Port Channel API

curl -k -v -u admin:Admin#123 -X PUT -H Content-type:application/vnd.yang.data+xml https://198.51.100.11/api/config/pnics/pnic/pc/lacp_type --data '<lacp_type>active</lacp_type>'
Example: PUT Configure the Bond Mode of a Port Channel API

curl -k -v -u admin:Admin123 -X PUT -H Content-type:application/vnd.yang.data+xml
https://198.51.100.11/api/config/pnics/pnic/pc/bond_mode --data
'bond_mode=balance-tcp'</bond_mode>

* Trying 198.51.100.11...
* Connected to 198.51.100.11 (198.51.100.11) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* Successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CAprompt: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
  * TLSv1.2 (IN), TLS handshake, Server hello (2):
    * NPN, negotiated HTTP/1.1
  * TLSv1.2 (IN), TLS handshake, Certificate (11):
  * TLSv1.2 (IN), TLS handshake, Server key exchange (12):
  * TLSv1.2 (IN), TLS handshake, Server finished (14):
  * TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
  * TLSv1.2 (OUT), TLS change cipher, Client hello (1):
  * TLSv1.2 (OUT), TLS change cipher, Unknown (67):
  * TLSv1.2 (OUT), TLS handshake, Finished (20):
  * TLSv1.2 (IN), TLS change cipher, Client hello (1):
  * TLSv1.2 (IN), TLS handshake, Finished (20):
  * SSL connection using TLSv1.2 / DHE-RSA-AES256-SHA
* Server certificate:
  * Subject: CN=nfvis
  * Start date: Dec 5 15:26:32 2017 GMT
  * Expire date: Dec 3 15:26:32 2027 GMT
  * Issuer: CN=nfvis
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
* Server auth using Basic with user 'admin'
> PUT /api/config/pnics/pnic/pc/bond_mode HTTP/1.1
> Host: 198.51.100.11
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.50.1
> Accept: */*
> Content-type:application/vnd.yang.data+xml
> Content-Length: 34
>
> * upload completely sent off: 34 out of 34 bytes
< HTTP/1.1 204 No Content
< Server: nginx
< Date: Wed, 06 Dec 2017 18:41:11 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Wed, 06 Dec 2017 18:32:48 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1512-585168-972196
< Pragma: no-cache
<
* Connection #0 to host 198.51.100.11 left intact
Example: PUT Configure Trunks on a Port Channel API

```
curl -k -v -u admin:Admin#123 -X PUT -H Content-type:application/vnd.yang.data+xml https://198.51.100.11/api/config/pnics/pnic/pc/trunks --data '<trunks>10,20</trunks>'
```

Example: DELETE Remove a Port from a Port Channel API

```
curl -k -v -u admin:Admin#123 -X DELETE -H Content-type:application/vnd.yang.data+xml https://198.51.100.11/api/config/pnics/pnic/eth1/member_of --data '<member_of>pc</member_of>'
```
Example: PUT Remove a Port Channel from a Bridge API

```
curl -k -v -u admin:Admin#123 -X PUT -H Content-type:application/vnd.yang.data+xml
https://198.51.100.11/api/config/bridges/bridge/testbridge --data
'<bridge><name>test-br</name></bridge>'
```

Example: PUT Remove a Port Channel from a Bridge API

```
* Trying 198.51.100.11...
* Connected to 198.51.100.11 (198.51.100.11) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CPath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP/1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=nfvis
  * start date: Dec 5 15:26:32 2017 GMT
  * expire date: Dec 3 15:26:32 2027 GMT
  * issuer: CN=nfvis
* SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
* Host: 198.51.100.11
* Authorization: Basic YWRtaW46Q2lzY28xMjMj
* User-Agent: curl/7.50.1
  * upload completely sent off: 25 out of 25 bytes
* HTTP/1.1 204 No Content
* Server: nginx
* Date: Wed, 06 Dec 2017 18:55:32 GMT
* Content-Type: text/html
* Content-Length: 0
* Connection: keep-alive
* Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
* Etag: 1512-586532-509745
* Pragma: no-cache
* Connection #0 to host 198.51.100.11 left intact
Example: DELETE Delete a Port Channel API

curl -k -v -u admin:Admin#123 -X DELETE -H Content-type:application/vnd.yang.data+xml
https://198.51.100.11/api/config/pnics/pnic/pc

* Trying 198.51.100.11...
* Connected to 198.51.100.11 (198.51.100.11) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP/1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* SSL connection using TLSv1.2 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=nfvis
  * start date: Dec 5 15:26:32 2017 GMT
  * expire date: Dec 3 15:26:32 2027 GMT
  * issuer: CN=nfvis
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
> PUT /api/config/bridges/bridge/test-br HTTP/1.1
> Host: 198.51.100.11
> Authorization: Basic YWRtaW46Q2lzY28xMjM=
> User-Agent: curl/7.50.1
> Accept: */*
> Content-type:application/vnd.yang.data+xml
> Content-Length: 37
>
* upload completely sent off: 37 out of 37 bytes
< HTTP/1.1 204 No Content
< Server: nginx
< Date: Wed, 06 Dec 2017 19:09:06 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Wed, 06 Dec 2017 19:09:05 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1512-587345-710932
<Pragma: no-cache
<
* Connection #0 to host 198.51.100.11 left intact
Example: GET LLDP Information API

curl -k -v -u admin:Cisco123# -X GET -H Content-type:application/vnd.yang.data+xml 
'https://172.19.162.231/api/operational/lldp?deep'

Note: Unnecessary use of -X or --request, GET is already inferred.

* Trying 172.19.162.231...

* Connected to 172.19.162.231 (172.19.162.231) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH

* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * Cpath: none

* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / DHE-RSA-AES256-SHA

Server certificate:
  * subject: CN=nfvis
  * start date: Dec 5 15:26:32 2017 GMT
  * expire date: Dec 3 15:26:32 2027 GMT
  * issuer: CN=nfvis
  * SSL certificate verify result: self signed certificate (18), continuing anyway.

* Server auth using Basic with user 'admin'

> DELETE /api/config/pnics/pnic/pc HTTP/1.1
> Host: 198.51.100.11
> Authorization: Basic YWRtaW46Q2lzY28xMjMj
> User-Agent: curl/7.50.1
> Accept: */*
> Content-type:application/vnd.yang.data+xml
>
< HTTP/1.1 204 No Content
< Server: nginx
< Date: Wed, 06 Dec 2017 19:11:24 GMT
< Content-Type: text/html
< Content-Length: 0
< Connection: keep-alive
< Last-Modified: Wed, 06 Dec 2017 19:11:24 GMT
< Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
< Etag: 1512-587484-283024
< Pragma: no-cache
<
  * Connection #0 to host 198.51.100.11 left intact
API Reference for Cisco Enterprise Network Function Virtualization Infrastructure Software

**Example: GET LLDP Information API**

```xml
    <neighbors>
        <name>eth0</name>
        <device_id>Switch1623</device_id>
        <holdtime>120</holdtime>
        <caps>Bridge, Router</caps>
        <platform>Cisco IOS Software, Catalyst L3 Switch Software (CAT3K_CAA-UNIVERSALK9-M), Version 15.0(1)EX3, RELEASE SOFTWARE (fc2)</platform>
        <portid>0</portid>
        <description>GigabitEthernet1/0/4</description>
    </neighbors>
    <neighbors>
        <name>eth1</name>
        <device_id>None</device_id>
        <holdtime>0</holdtime>
        <caps>None</caps>
        <platform>None</platform>
        <portid>None</portid>
        <description>None</description>
    </neighbors>
    <neighbors>
        <name>eth2</name>
        <device_id>None</device_id>
        <holdtime>0</holdtime>
        <caps>None</caps>
        <platform>None</platform>
        <portid>None</portid>
        <description>None</description>
    </neighbors>
    <neighbors>
        <name>eth3</name>
        <device_id>None</device_id>
        <holdtime>0</holdtime>
        <caps>None</caps>
        <platform>None</platform>
        <portid>None</portid>
        <description>None</description>
    </neighbors>
    <neighbors>
        <name>eth4</name>
        <device_id>None</device_id>
        <holdtime>0</holdtime>
        <caps>None</caps>
        <platform>None</platform>
        <portid>None</portid>
        <description>None</description>
    </neighbors>
</lldp>
```
Example: PUT Enable LLDP Configuration API

curl -v -k -u admin:Admin#123 -X PUT -H Content-type:application/vnd.yang.data+xml 'https://198.51.100.1/api/config/pnics/pnic/eth0/lldp --data '<lldp>enabled</lldp>'

* Connection #0 to host 172.19.162.231 left intact
Example: GET Port Admin Status API

Example: GET Port Admin Status API

Example: GET Port Admin Status API

curl -k -v -u admin:Cisco123# -X GET
https://198.51.100.1/api/config/pnics/pnic/eth5/adminstatus

* Trying 198.51.100.11...
* Connected to 198.51.100.11 (198.51.100.11) port 443 (#0)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAfile: /etc/pki/tls/certs/ca-bundle.crt
  * CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP/1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* SSL connection using TLSv1.2 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=nfvis
  * start date: Dec 5 15:26:32 2017 GMT
  * expire date: Dec 3 15:26:32 2027 GMT
  * issuer: CN=nfvis
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
> GET /api/config/pnics/pnic/eth5/adminstatus HTTP/1.1
> Host: 198.51.100.11
> Authorization: Basic YM9tYWt4c3Qx2VzYzQzYzQzYzQz
> User-Agent: curl/7.50.1
> Accept: */*
> HTTP/1.1 200 OK
> Server: nginx
> Date: Wed, 06 Dec 2017 19:15:23 GMT
> Content-Type: application/vnd.yang.data+xml
> Transfer-Encoding: chunked
> Connection: keep-alive
> Last-Modified: Wed, 06 Dec 2017 19:14:09 GMT
> Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
> Etag: 1512-587649-439226
> Pragma: no-cache
>

* Connection #0 to host 198.51.100.11 left intact
Example: PUT Configure Port Admin Status API

curl -v -k -u admin:Admin#123 -X PUT -H Content-type:application/vnd.yang.data+xml
Content-type:application/vnd.yang.data+xml
'https://198.51.100.11/api/config/pnics/pnic/eth5/adminstatus --data
'<adminstatus>up</adminstatus>'

* Trying 198.51.100.11...
* Connected to 198.51.100.11 (198.51.100.11) port 443 (#1)
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
  * CAtime: /etc/pki/tls/certs/ca-bundle.crt
  CAbased: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* NPN, negotiated HTTP1.1
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Unknown (67):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / DHE-RSA-AES256-SHA
* Server certificate:
  * subject: CN=nfvis
  * start date: Dec 5 15:26:32 2017 GMT
  * expire date: Dec 3 15:26:32 2027 GMT
  * issuer: CN=nfvis
  * SSL certificate verify result: self signed certificate (18), continuing anyway.
  * Server auth using Basic with user 'admin'
  > PUT /api/config/pnics/pnic/eth5/adminstatus HTTP/1.1
  > Host: 198.51.100.11
  > Authorization: Basic YWRtaW46Q2lzY28xMjMj
  > User-Agent: curl/7.50.1
  > Accept: */*
  > Content-type:application/vnd.yang.data+xml
  > Content-Length: 29
  >
  * upload completely sent off: 29 out of 29 bytes
  < HTTP/1.1 204 No Content
  < Server: nginx
  < Date: Wed, 06 Dec 2017 19:14:09 GMT
  < Content-Type: text/html
  < Content-Length: 0
  < Connection: keep-alive
  < Last-Modified: Wed, 06 Dec 2017 19:14:09 GMT
  < Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
  < Etag: 1512-587649-439226
  < Pragma: no-cache
  <
  * Connection #1 to host 198.51.100.11 left intact
# Speed, Autoneg and Duplex APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To configure speed</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/pnics/pnic/GE0-0/speed</td>
</tr>
<tr>
<td>To configure duplex</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/pnics/pnic/GE0-0/duplex</td>
</tr>
<tr>
<td>To configure speed and duplex</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/pnics/pnic/GE0-0/</td>
</tr>
<tr>
<td>To get the perational speed</td>
<td>GET</td>
<td>No</td>
<td>/api/config/pnics/pnic/GE0-0/operational-speed</td>
</tr>
<tr>
<td>To get the operational duplex</td>
<td>GET</td>
<td>No</td>
<td>/api/config/pnics/pnic/GE0-0/operational-duplex</td>
</tr>
<tr>
<td>To get the operational autoneg</td>
<td>GET</td>
<td>No</td>
<td>/api/config/pnics/pnic/GE0-0/operational-duplex</td>
</tr>
</tbody>
</table>
Port Security APIs

Example: Max Number of MAC Addresses

https://209.165.201.1/api/running/switch/interface/gigabitEthernet

Payload:
<gigabitEthernet>
  <name>1/0</name>
  <port-security>
    <max>2</max>
  </port-security>
</gigabitEthernet>

Example: Violation Discard

https://209.165.201.1/api/running/switch/interface/gigabitEthernet

Payload:
<gigabitEthernet>
  <name>1/0</name>
  <port-security>
    <violation>discard</violation>
  </port-security>
</gigabitEthernet>

Example: Enable Port Security

https://209.165.201.1/api/running/switch/interface/gigabitEthernet

Payload:
<gigabitEthernet>
  <name>1/0</name>
  <port-security>
    <enable/>
  </port-security>
</gigabitEthernet>
Example: Secure Static MAC

https://209.165.201.1/api/running/switch/mac

Payload:

```
<mac>
  <address-table>
    <static>
      <mac-entries>
        <vlan>1</vlan>
        <interface>
          <gigabitEthernet>1/0</gigabitEthernet>
        </interface>
        <type>secure</type>
      </mac-entries>
    </static>
  </address-table>
</mac>
```

Example: show switch interface port-security

GET https://209.165.201.1/api/operational/switch/interface/port-security?deep

Example: no port security enable

DELETE

https://209.165.201.1/api/running/switch/interface/gigabitEthernet/1/0/port-security/enable

Example: no mac address-table static

DELETE https://209.165.201.1/api/running/switch/mac/address-table/static/mac-entries
CHAPTER 17

Secure Overlay APIs

Table 101: Secure Overlay APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To create secure overlay configuration</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/secure-overlays</td>
</tr>
<tr>
<td>To get secure overlay configuration</td>
<td>GET</td>
<td>No</td>
<td>/api/config/secure-overlays?deep</td>
</tr>
<tr>
<td>To delete secure overlay configuration</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/secure-overlays</td>
</tr>
<tr>
<td>To get secure overlay state data</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/secure-overlays</td>
</tr>
</tbody>
</table>

Example for secure overlay payload

```xml
<secure-overlay>
  <name>mgmthub</name>
  <local-bridge>wan-br</local-bridge>
  <local-system-ip-addr>34.34.34.4</local-system-ip-addr>
  <remote-interface-ip-addr>10.85.189.36</remote-interface-ip-addr>
  <remote-system-ip-addr>10.19.18.251</remote-system-ip-addr>
  <remote-id>mgmt-hub.cloudvpn.com</remote-id>
  <psk>
    <local-psk>Cisco1234Admin</local-psk>
    <remote-psk>Cisco1234Admin</remote-psk>
  </psk>
</secure-overlay>
```

Table 102: Description for Secure Overlay Payloads

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of secure overlay connection.</td>
<td>Yes</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of secure overlay connection</td>
<td>No</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
<td>Mandatory</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>local-bridge</td>
<td>String</td>
<td>Local bridge name for overlay (default wan-br)</td>
<td>No</td>
</tr>
<tr>
<td>local-system-ip-addr</td>
<td>String</td>
<td>Local overlay system IPv4 address.</td>
<td>Yes</td>
</tr>
<tr>
<td>local-system-ip-subnet</td>
<td>String</td>
<td>Local overlay subnet. H.H.H.H/N Default is /32</td>
<td>No</td>
</tr>
<tr>
<td>remote-interface-ip-addr</td>
<td>String</td>
<td>Remote interface IPv4 address</td>
<td>Yes</td>
</tr>
<tr>
<td>remote-system-ip-addr</td>
<td>String</td>
<td>Remote system IPv4 address</td>
<td>Yes</td>
</tr>
<tr>
<td>remote-id</td>
<td>String</td>
<td>Remote id for overlay - IP or FQDN (default remote-interface-ip-addr)</td>
<td>No</td>
</tr>
<tr>
<td>ike-cipher</td>
<td>String</td>
<td>IKE algorithms. Possible values: aes128-sha1-modp1536 Default: aes128-sha1-modp1536</td>
<td>No</td>
</tr>
<tr>
<td>esp-cipher</td>
<td>String</td>
<td>ESP algorithms. Possible values: aes128-sha1 Default: aes128-sha1</td>
<td>No</td>
</tr>
<tr>
<td>psk</td>
<td>String</td>
<td>Pre-shared-key for authentication</td>
<td>No</td>
</tr>
<tr>
<td>psk local-psk</td>
<td>String</td>
<td>Local pre-shared-key</td>
<td>Yes</td>
</tr>
<tr>
<td>psk remote-psk</td>
<td>String</td>
<td>Remote pre-shared-key</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Example: POST Secure Overlay APIs**

curl -k -v -u "admin:123#Admin" -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X POST https://209.165.201.1/api/config/secure-overlays -d '  
<secure-overlay>  
  <name>mgmthub</name>  
  <local-bridge>wan-br</local-bridge>  
  <local-system-ip-addr>10.0.0.1</local-system-ip-addr>  
  <remote-interface-ip-addr>10.0.0.1</remote-interface-ip-addr>  
  <remote-system-ip-addr>10.0.0.2</remote-system-ip-addr>  
'
Example: POST create secure overlay with int-mgmt-net as local system ip address

```
curl -k -v -u "admin:admin" -H Accept:application/vnd.yang.data+xml -H Content-Type:application/vnd.yang.data+xml -X POST https://209.165.201.1/api/config/secure-overlays -d '  
<secure-overlay>  
  <name>mgmthub</name>  
  <local-bridge>wan-br</local-bridge>  
  <local-system-ip-addr>10.0.0.4</local-system-ip-addr>  
  <local-system-ip-bridge>int-mgmt-net</local-system-ip-bridge>  
  <remote-interface-ip-addr>10.0.0.1</remote-interface-ip-addr>  
  <remote-system-ip-addr>10.0.0.2</remote-system-ip-addr>  
  <remote-id>mgmt-hub.cloudvpn.com</remote-id>  
  <psk>  
    <local-psk>Cisco1234Admin</local-psk>  
    <remote-psk>Cisco1234Admin</remote-psk>  
  </psk>  
</secure-overlay>
```

Example: GET Secure Overlay APIs

```
curl -k -v -u "admin:123#Admin" -X GET "https://209.165.201.1/api/config/secure-overlays?deep"
```

Example: DELETE Secure Overlay APIs

```
curl -k -v -u "admin:123#Admin" -X DELETE "https://209.165.201.1/api/config/secure-overlays"
```

---

**Single IP Configuration APIs**

*Single IP Configuration APIs, on page 227*
Example for single IP configuration payload

```xml
<single-ip-mode>
  <vm-name>ROUTER.ROUTER</vm-name>
</single-ip-mode>
```

### Table 104: Description for Single IP Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>vm-name</td>
<td>String</td>
<td>Name of VM taking the public IP.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example: POST Single IP configuration APIs

```bash
curl -k -v -u "admin:123#Admin" -H "Accept:application/vnd.yang.data+xml" -H "Content-Type:application/vnd.yang.data+xml" -X PUT https://209.165.201.1/api/config/single-ip-mode -d "
  <single-ip-mode>
    <vm-name>ROUTER.ROUTER</vm-name>
  </single-ip-mode>"
```

Example: GET Single IP configuration APIs

```bash
curl -k -v -u "admin:123#Admin" -X GET "https://209.165.201.1/api/config/single-ip-mode"
```

Example: DELETE Single IP configuration APIs

```bash
curl -k -v -u "admin:123#Admin" -X DELETE "https://209.165.201.1/api/config/single-ip-mode"
```
PART II

Switch Related APIs

• DOT1x APIs, on page 231
• IP Gateway APIs, on page 237
• Spanning-Tree All or Individual Elements APIs, on page 239
• Interface Stat APIs, on page 247
• Interface GigabitEthernet Switchport APIs, on page 251
• Interface GigabitEthernet Spanning-Tree APIs, on page 255
• SPAN/RSPAN APIs, on page 257
• VLAN and interface VLAN related APIs, on page 261
CHAPTER 18

DOT1x APIs

Table 105: DOT1x APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To view the dot1x summary</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/switch/dot1x/summary</td>
</tr>
<tr>
<td>To view the dot1x configuration</td>
<td>GET</td>
<td>No</td>
<td>/api/config/switch/dot1x</td>
</tr>
</tbody>
</table>

Example: GET DOT1x APIs

curl -k -u admin:admin -X GET https://209.165.201.1/api/operational/switch/dot1x/summary

curl -k -u admin:admin -X GET https://209.165.201.1/api/config/switch/dot1x

• DOT1x guest-vlan Timeout Value APIs, on page 231
• DOT1x Default authentication APIs, on page 233
• DOT1x System Authentication Control APIs, on page 233
• RADIUS Source Interface Address APIs, on page 234

DOT1x guest-vlan Timeout Value APIs

Table 106: DOT1x guest-vlan Timeout Value APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To enable unauthorized users on the access interface to the guest VLAN</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/switch/dot1x/guest-vlan</td>
</tr>
</tbody>
</table>
### DOT¹x guest-vlan Timeout Value APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To set the time delay between enabling Dot¹X and adding a port to the guest VLAN</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/switch/dot¹x/guest-vlan/timeout</td>
</tr>
<tr>
<td>To restore the default configuration</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/switch/dot¹x/guest-vlan/timeout</td>
</tr>
<tr>
<td>To get the VLAN timeout value</td>
<td>GET</td>
<td>No</td>
<td>/api/config/switch/dot¹x/guest-vlan/timeout</td>
</tr>
</tbody>
</table>

#### Example for DOT¹x guest-vlan Timeout Value APIs Payload

```xml
<timeout>30</timeout>
```

#### Table 107: Description for DOT¹x guest-vlan Timeout Value APIs Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeout</td>
<td>integer</td>
<td>Specifies the time delay in seconds between enabling dot¹X and adding the port to the guest VLAN. (Range: 30–180)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

#### Example: POST DOT¹x guest-vlan Timeout Value APIs

```bash
curl -k -u admin:admin -d "<timeout>30</timeout>" -X POST https://209.165.201.1/api/config/switch/dot¹x/guest-vlan 
```

#### Example: PUT DOT¹x guest-vlan Timeout Value APIs

```bash
curl -k -u admin:admin -d "<timeout>40</timeout>" -X PUT https://209.165.201.1/api/config/switch/dot¹x/guest-vlan/timeout 
```

#### Example: DELETE DOT¹x guest-vlan Timeout Value APIs

```bash
curl -k -u admin:admin -X DELETE https://209.165.201.1/api/config/switch/dot¹x/guest-vlan/timeout 
```

#### Example: GET DOT¹x guest-vlan Timeout Value APIs

```bash
curl -k -u admin:admin -X GET https://209.165.201.1/api/config/switch/dot¹x/guest-vlan/timeout 
```
**DOT1x Default authentication APIs**

*Table 108: DOT1x Default authentication APIs*

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To enable authentication methods on a port</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/switch/dot1x/authentication</td>
</tr>
<tr>
<td>To restore the default configuration</td>
<td>GET</td>
<td>No</td>
<td>/api/config/switch/dot1x/authentication/default</td>
</tr>
<tr>
<td>To delete the authentication configuration</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/switch/dot1x/authentication</td>
</tr>
</tbody>
</table>

**Example for DOT1x Default authentication APIs Payload**

<default>radius</default>

**Example: POST DOT1x Default authentication APIs**

curl -k -u admin:admin -d "<default>radius</default>" -X POST https://209.165.201.1/api/config/switch/dot1x/authentication -H "Content-Type: application/vnd.yang.data+xml"

**Example: GET DOT1x Default authentication APIs**

curl -k -u admin:admin -X GET https://209.165.201.1/api/config/switch/dot1x/authentication/default

**Example: DELETE DOT1x Default authentication APIs**

curl -k -u admin:admin -X DELETE https://209.165.201.1/api/config/switch/dot1x/authentication

**DOT1x System Authentication Control APIs**

*Table 109: DOT1x System Authentication Control APIs*

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To enable dot1x globally</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/switch/dot1x</td>
</tr>
<tr>
<td>To get the configuration for system authentication control</td>
<td>GET</td>
<td>No</td>
<td>/api/config/switch/dot1x/system-auth-control</td>
</tr>
<tr>
<td>To restore default configuration</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/switch/dot1x/system-auth-control</td>
</tr>
</tbody>
</table>
Example for DOT1x System Authentication Control APIs Payload

<system-auth-control></system-auth-control>

Example: POST DOT1x System Authentication Control APIs

```bash
curl -k -u admin:admin -d "<system-auth-control></system-auth-control>" -X POST https://209.165.201.1/api/config/switch/dot1x -H "Content-Type: application/vnd.yang.data+xml"
```

Example: GET DOT1x System Authentication Control APIs

```bash
curl -k -u admin:admin -X GET https://209.165.201.1/api/config/switch/dot1x/system-auth-control
```

Example: DELETE DOT1x System Authentication Control APIs

```bash
curl -k -u admin:admin -X DELETE https://209.165.201.1/api/config/switch/dot1x/system-auth-control
```

### RADIUS Source Interface Address APIs

Table 110: RADIUS Source Interface Address APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To enable RADIUS-based VLAN assignment</td>
<td>POST</td>
<td>Yes</td>
<td>/api/config/switch/ip/radius/source-interface</td>
</tr>
<tr>
<td>To get the RADIUS-based VLAN configurations</td>
<td>GET</td>
<td>No</td>
<td>/api/config/switch/ip/radius/source-interface/vlan</td>
</tr>
<tr>
<td>To replace RADIUS-based VLAN</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/switch/ip/radius/source-interface/vlan</td>
</tr>
<tr>
<td>To disable RADIUS-based VLAN assignment</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/switch/ip/radius/source-interface/vlan</td>
</tr>
</tbody>
</table>

Example for RADIUS Source Interface Address APIs Payload

```xml
<vlan>505</vlan>
```

Example: POST RADIUS Source Interface Address APIs

```bash
curl -k -u admin:admin -d "<vlan>505</vlan>" -X POST https://209.165.201.1/api/config/switch/ip/radius/source-interface -H "Content-Type: application/vnd.yang.data+xml"
```

Example: GET RADIUS Source Interface Address APIs

```bash
curl -k -u admin:admin -X GET https://209.165.201.1/api/config/switch/ip/radius/source-interface/vlan
```
Example: PUT RADIUS Source Interface Address APIs

curl -k -u admin:admin -d "<vlan>506</vlan>" -X PUT
https://209.165.201.1/api/config/switch/ip/radius/source-interface/vlan
-H "Content-Type: application/vnd.yang.data+xml"

Example: DELETE RADIUS Source Interface Address APIs

curl -k -u admin:admin -X DELETE
https://209.165.201.1/api/config/switch/ip/radius/source-interface/vlan
IP Gateway APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To define a default gateway</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/running/switch/ip/default-gateway</td>
</tr>
<tr>
<td>To restore the default</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/ip/default-gateway</td>
</tr>
<tr>
<td>configuration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To show the default gateway</td>
<td>GET</td>
<td>No</td>
<td>api/running/switch/ip/default-gateway</td>
</tr>
</tbody>
</table>

Example for IP Gateway APIs Payload

```xml
<default-gateway>
gateway>169.254.1.3</gateway></default-gateway>
```

Example for IP Gateway APIs Payload Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>gateway</td>
<td>String</td>
<td>Specifies the default gateway IP address</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example: PUT IP Gateway APIs

```bash
```

Example: DELETE IP Gateway APIs

```bash
curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/ip/default-gateway
```

Example: GET IP Gateway APIs

```bash
curl -k -u admin:admin -X GET "https://209.165.201.1/api/running/switch/ip/default-gateway"
```

• IP Route APIs, on page 238
Table 113: IP Route APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To enable static routes</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/running/switch/ip/routing</td>
</tr>
<tr>
<td>To add a static route</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/running/switch/ip/route</td>
</tr>
<tr>
<td>To remove a static route</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/ip/route-ip-route-forwarding-list</td>
</tr>
</tbody>
</table>

Example for IP Route APIs Payload

```xml
<routing/>
```

Example for IP Route APIs Payload

```xml
<route>
<ip-route-forwarding-list>
<prefix>2.2.2.2</prefix>
<mask>255.255.255.255</mask>
<forwarding-address>5.5.5.1</forwarding-address>
</ip-route-forwarding-list>
</route>
```

Example: PUT IP Route APIs

```bash
curl -k -u admin:admin -d "<routing/>
<routing>" -X PUT https://209.165.201.1/api/running/switch/ip/routing -H "Content-Type: application/vnd.yang.data+xml"
```

Example: PUT IP Route APIs

```bash
curl -k -u admin:admin -d "<route>
<ip-route-forwarding-list>
<prefix>2.2.2.2</prefix>
<mask>255.255.255.255</mask>
<forwarding-address>5.5.5.1</forwarding-address>
</ip-route-forwarding-list>
</route>"
-X PUT https://209.165.201.1/api/running/switch/ip/route -H "Content-Type: application/vnd.yang.data+xml"
```

Example: DELETE IP Route APIs

```bash
curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/ip/route-ip-route-forwarding-list/2.2.2.2,255.255.255.255,5.5.5.1
```
## Spanning-Tree All or Individual Elements APIs

### Table 114: Spanning-Tree Individual Elements APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To get the spanning tree configuration</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/spanning-tree?deep</td>
</tr>
<tr>
<td>To define Bridge Protocol Data Unit (BPDU) handling when the spanning tree is disabled globally or on a single interface</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/spanning-tree?select=bpdu</td>
</tr>
<tr>
<td>To enable the spanning-tree functionality</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/spanning-tree?select=enable</td>
</tr>
<tr>
<td>To configure the spanning-tree bridge forward time, which is the amount of time a port remains in the listening and learning states before entering the forwarding state.</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/spanning-tree?select=forward-time</td>
</tr>
<tr>
<td>To select the Spanning Tree Protocol (STP) protocol</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/spanning-tree?select=mode</td>
</tr>
<tr>
<td>To configure the number of times Hello messages of the device is broadcasted to other devices.</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/spanning-tree?select=hello-time</td>
</tr>
<tr>
<td>To configure the STP maximum age</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/spanning-tree?select=max-age</td>
</tr>
<tr>
<td>Action</td>
<td>Method</td>
<td>Payload Required</td>
<td>API</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>--------</td>
<td>------------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>To shutdown an interface if it receives a loopback BPDU</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/spanning-tree?select=loopback-guard</td>
</tr>
<tr>
<td>To configure the path cost for MST calculations.</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/spanning-tree/pathcost?deep</td>
</tr>
<tr>
<td>To set the default path cost method.</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/spanning-tree/pathcost?select=method</td>
</tr>
<tr>
<td>To configure the device STP priority, which is used to determine which bridge is selected as the root bridge.</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/spanning-tree?select=priority</td>
</tr>
</tbody>
</table>

**Example: GET Spanning-Tree APIs**

```bash
```

**Example: GET Spanning-Tree bpdu APIs**

```bash
```

**Example: GET Spanning-Tree enable APIs**

```bash
curl -k -u admin:admin -X GET https://209.165.201.1/api/running/switch/spanning-tree?select=enable
```

**Example: GET Spanning-Tree forward-time APIs**

```bash
```

**Example: GET Spanning-Tree mode APIs**

```bash
curl -k -u admin:admin -X GET https://209.165.201.1/api/running/switch/spanning-tree?select=mode
```

**Example: GET Spanning-Tree hello-time APIs**

```bash
```

**Example: GET Spanning-Tree max-age APIs**

```bash
curl -k -u admin:admin -X GET https://209.165.201.1/api/running/switch/spanning-tree?select=max-age
```

**Example: GET Spanning-Tree loopback-guard APIs**

```bash
```
Example: GET Spanning-Tree pathcost APIs

curl -k -u admin:admin -X GET
https://209.165.201.1/api/running/switch/spanning-tree/pathcost?deep

Example: GET Spanning-Tree pathcost method APIs

curl -k -u admin:admin -X GET
https://209.165.201.1/api/running/switch/spanning-tree/pathcost?select=method

Example: GET Spanning-Tree priority APIs

curl -k -u admin:admin -X GET
https://209.165.201.1/api/running/switch/spanning-tree?select=priority

Create Spanning-Tree APIs

Table 115: Create Spanning-Tree APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To create the spanning-tree elements</td>
<td>PATCH</td>
<td>Yes</td>
<td>/api/running/switch/spanning-tree</td>
</tr>
</tbody>
</table>

Example for Create Spanning-Tree APIs Payload

<spanning-tree><bpdu>filtering</bpdu></spanning-tree>

Example for Create Spanning-Tree APIs Payload

<spanning-tree><forward-time>18</forward-time></spanning-tree>

Table 116: Description for Create Spanning-Tree APIs Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>forward-time</td>
<td>Integer</td>
<td>Specifies the spanning-tree forward time in seconds. (Range: 4–30)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example for Create Spanning-Tree APIs Payload

<spanning-tree><mode>rstp</mode></spanning-tree>

Table 117: Description for Create Spanning-Tree APIs Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>String</td>
<td>Specifies the STP, RSTP or MSTP mode</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Example for Create Spanning-Tree APIs Payload

```xml
<spanning-tree><hello-time>6</hello-time></spanning-tree>
```

Table 118: Description for Create Spanning-Tree APIs Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello-time</td>
<td>Integer</td>
<td>Specifies the spanning-tree Hello time in seconds. (Range: 1–10)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example for Create Spanning-Tree APIs Payload

```xml
<spanning-tree><max-age>24</max-age></spanning-tree>
```

Table 119: Description for Create Spanning-Tree APIs Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max-age</td>
<td>Integer</td>
<td>Specifies the spanning-tree bridge maximum age in seconds. (Range: 6–40)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example for Create Spanning-Tree APIs Payload

```xml
<spanning-tree><loopback-guard></loopback-guard></spanning-tree>
```

Example for Create Spanning-Tree APIs Payload

```xml
<spanning-tree><method>short</method></spanning-tree>
```

Table 120: Description for Create Spanning-Tree APIs Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>String</td>
<td>Specifies the default port path costs</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example for Create Spanning-Tree APIs Payload

```xml
<spanning-tree><priority>8192</priority></spanning-tree>
```

Table 121: Description for Create Spanning-Tree APIs Payload

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory/Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority</td>
<td>String</td>
<td>Specifies the device priority for the specified spanning-tree instance.</td>
<td>yes</td>
</tr>
</tbody>
</table>

Example: PATCH Create Spanning-Tree bpdu APIs

```bash
curl -k -u admin:admin -d "<spanning-tree><bpdu>filtering</bpdu></spanning-tree>" -X PATCH https://209.165.201.1/api/running/switch/spanning-tree -H "Content-Type: application/vnd.yang.data+xml"
```
Example: PATCH Create Spanning-Tree forward-time APIs

curl -k -u admin:admin -d "<spanning-tree><forward-time>18</forward-time></spanning-tree>" -X PATCH https://209.165.201.1/api/running/switch/spanning-tree -H "Content-Type: application/vnd.yang.data+xml"

Example: PATCH Create Spanning-Tree mode APIs

curl -k -u admin:admin -d "<spanning-tree><mode>rstp</mode></spanning-tree>" -X PATCH https://209.165.201.1/api/running/switch/spanning-tree -H "Content-Type: application/vnd.yang.data+xml"

Example: PATCH Create Spanning-Tree hello-time APIs


Example: PATCH Create Spanning-Tree max-age APIs

curl -k -u admin:admin -d "<spanning-tree><max-age>24</max-age></spanning-tree>" -X PATCH https://209.165.201.1/api/running/switch/spanning-tree -H "Content-Type: application/vnd.yang.data+xml"

Example: PATCH Create Spanning-Tree loopback-guard APIs


Example: PATCH Create Spanning-Tree method APIs


Example: PATCH Create Spanning-Tree priority APIs

curl -k -u admin:admin -d "<spanning-tree><priority>8192</priority></spanning-tree>" -X PATCH https://209.165.201.1/api/running/switch/spanning-tree -H "Content-Type: application/vnd.yang.data+xml"

Modify Spanning-Tree APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To modify the spanning-tree elements</td>
<td>PATCH</td>
<td>Yes</td>
<td>/api/running/switch/spanning-tree</td>
</tr>
</tbody>
</table>

Example: PATCH Modify Spanning-Tree bpdu APIs

curl -k -u admin:admin -d "<spanning-tree><bpdu>filtering</bpdu></spanning-tree>" -X PATCH https://209.165.201.1/api/running/switch/spanning-tree -H "Content-Type: application/vnd.yang.data+xml"
Example: PATCH Modify Spanning-Tree forward-time APIs
curl -k -u admin:admin -d "<spanning-tree><forward-time>18</forward-time></spanning-tree>" -X PATCH https://209.165.201.1/api/running/switch/spanning-tree -H "Content-Type: application/vnd.yang.data+xml"

Example: PATCH Modify Spanning-Tree mode APIs
curl -k -u admin:admin -d "<spanning-tree><mode>rstp</mode></spanning-tree>" -X PATCH https://209.165.201.1/api/running/switch/spanning-tree -H "Content-Type: application/vnd.yang.data+xml"

Example: PATCH Modify Spanning-Tree hello-time APIs

Example: PATCH Modify Spanning-Tree max-age APIs
curl -k -u admin:admin -d "<spanning-tree><max-age>24</max-age></spanning-tree>" -X PATCH https://209.165.201.1/api/running/switch/spanning-tree -H "Content-Type: application/vnd.yang.data+xml"

Example: PATCH Modify Spanning-Tree loopback-guard APIs

Example: PATCH Modify Spanning-Tree method APIs

Example: PATCH Modify Spanning-Tree priority APIs
curl -k -u admin:admin -d "<spanning-tree><priority>8192</priority></spanning-tree>" -X PATCH https://209.165.201.1/api/running/switch/spanning-tree -H "Content-Type: application/vnd.yang.data+xml"

Delete Spanning-Tree APIs

Table 123: Delete Spanning-Tree APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To restore the default configuration</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/spanning-tree/bpdu</td>
</tr>
<tr>
<td>To disable the spanning-tree functionality</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/spanning-tree/enable</td>
</tr>
<tr>
<td>Action</td>
<td>Method</td>
<td>Payload Required</td>
<td>API</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------</td>
<td>------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>To restore the default configuration</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/spanning-tree/forward-time</td>
</tr>
<tr>
<td>To restore the default configuration</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/spanning-tree/mode</td>
</tr>
<tr>
<td>To restore the default configuration</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/spanning-tree/hello-time</td>
</tr>
<tr>
<td>To restore the default configuration</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/spanning-tree/max-age</td>
</tr>
<tr>
<td>To restore the default configuration</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/spanning-tree/loopback-guard</td>
</tr>
<tr>
<td>To restore the default configuration</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/spanning-tree/pathcost</td>
</tr>
<tr>
<td>To restore the default device spanning-tree priority.</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/spanning-tree/priority</td>
</tr>
</tbody>
</table>

Example: DELETE Delete Spanning-Tree bpdu APIs

curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/spanning-tree/bpdu

Example: DELETE Delete Spanning-Tree APIs

curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/spanning-tree/enable

Example: DELETE Delete Spanning-Tree forward-time APIs

curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/spanning-tree/forward-time

Example: DELETE Delete Spanning-Tree mode APIs

curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/spanning-tree/mode

Example: DELETE Delete Spanning-Tree hello-time APIs

curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/spanning-tree/hello-time

Example: DELETE Delete Spanning-Tree max-age APIs

curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/spanning-tree/max-age

Example: DELETE Delete Spanning-Tree loopback-guard APIs

curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/spanning-tree/loopback-guard
Delete Spanning-Tree APIs

Example: DELETE Delete Spanning-Tree pathcost APIs

curl -k -u admin:admin -X DELETE  
https://209.165.201.1/api/running/switch/spanning-tree/pathcost

Example: DELETE Delete Spanning-Tree priority APIs

curl -k -u admin:admin -X DELETE  
https://209.165.201.1/api/running/switch/spanning-tree/priority
# Interface Stat APIs

## Table 124: Interface Stat APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To display the status of all interfaces or of a specific interface</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/switch/interface/status/gigabitEthernet/&quot;1/0&quot;</td>
</tr>
<tr>
<td>To display traffic seen by all the physical interfaces or by a specific interface</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/switch/interface/counters/gigabitEthernet/&quot;1/0&quot;</td>
</tr>
<tr>
<td>To display RMON Ethernet statistics</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/switch/interface/rmon/gigabitEthernet/&quot;1/0&quot;</td>
</tr>
<tr>
<td>To display information about the inline power for all interfaces or for a specific interface</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/switch/interface/inline-status/gigabitEthernet/&quot;1/0&quot;</td>
</tr>
<tr>
<td>To display the administrative and operational status of all interfaces or a specific interface.</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/switch/interface/switchPort/gigabitEthernet/&quot;1/0&quot;</td>
</tr>
<tr>
<td>To display the configuration for all configured interfaces</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/interface/gigabitEthernet</td>
</tr>
<tr>
<td>To display the configuration for a specific interface</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/interface/gigabitEthernet/&quot;1/0&quot;</td>
</tr>
<tr>
<td>To configure the speed of a given Ethernet interface when not using auto-negotiation</td>
<td>PATCH</td>
<td>Yes</td>
<td>/api/running/switch/interface/gigabitEthernet</td>
</tr>
</tbody>
</table>
Example for Interface Stat APIs Payload

```xml
<gigabitEthernet><name>1/0</name><speed>1000</speed></gigabitEthernet>
```

Example: GET Interface Stat APIs

```bash
curl -k -u admin:admin -X GET https://172.25.212.178/api/operational/switch/interface/status/gigabitEthernet/"1/0"
```

Example: GET Interface Stat APIs

```bash
curl -k -u admin:admin -X GET https://172.25.212.178/api/operational/switch/interface/counters/gigabitEthernet/"1/0"
```

Example: GET Interface Stat APIs

```bash
curl -k -u admin:admin -X GET https://172.25.212.178/api/operational/switch/interface/rmon/gigabitEthernet/"1/0"
```

Example: GET Interface Stat APIs

```bash
curl -k -u admin:admin -X GET https://172.25.212.178/api/operational/switch/interface/inline-status/gigabitEthernet/"1/0"
```

Example: GET Interface Stat APIs

```bash
curl -k -u admin:admin -X GET https://172.25.212.178/api/operational/switch/interface/switchPort/gigabitEthernet/"1/0"
```

Example: GET Interface Stat APIs

```bash
```

Example: GET Interface Stat APIs

```bash
```

Example: GET Interface Stat APIs

```bash
curl -k -u admin:admin -X GET https://209.165.201.1/api/running/switch/interface/gigabitEthernet/"1/0"
```

Example: PATCH Interface Stat APIs

```bash
curl -k -u admin:admin -d "<gigabitEthernet><name>1/0</name><speed>1000</speed></gigabitEthernet>" -X PATCH https://209.165.201.1/api/running/switch/interface/gigabitEthernet -H "Content-Type: application/vnd.yang.data+xml"
```

- Interface Port APIs, on page 249
## Interface Port APIs

### Table 125: Interface Port APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To delete the interface speed configuration</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/interface/gigabitEthernet/speed</td>
</tr>
<tr>
<td>To disable an interface</td>
<td>PATCH</td>
<td>Yes</td>
<td>/api/running/switch/interface/gigabitEthernet</td>
</tr>
<tr>
<td>To restart a disabled interface</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/interface/gigabitEthernet/&quot;1/0&quot;/shutdown</td>
</tr>
<tr>
<td>To delete the interface description</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/interface/gigabitEthernet/&quot;1/0&quot;/description</td>
</tr>
</tbody>
</table>

### Example for Interface Port APIs Payload

```
<gigabitEthernet><name>1/0</name><shutdown/></gigabitEthernet>
```

### Example: DELETE Interface Port APIs

```
curl -k -u admin:admin -X DELETE
https://209.165.201.1/api/running/switch/interface/gigabitEthernet/speed
```

### Example: PATCH Interface Port APIs

```
curl -k -u admin:admin -d "<gigabitEthernet><name>1/0</name><shutdown/></gigabitEthernet>" -X PATCH https://209.165.201.1/api/running/switch/interface/gigabitEthernet -H "Content-Type: application/vnd.yang.data+xml"
```

### Example: DELETE Interface Port APIs

```
curl -k -u admin:admin -X DELETE
https://209.165.201.1/api/running/switch/interface/gigabitEthernet/"1/0"/shutdown
```

### Example: DELETE Interface Port APIs

```
curl -i -k -u admin:admin -X DELETE
https://209.165.201.1/api/running/switch/interface/gigabitEthernet/"1/0"/description
```
## Interface GigabitEthernet Switchport APIs

### Table 126: Interface GigabitEthernet Switchport APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To retrieve interface switchport configuration</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/interface/gigabitEthernet/&quot;1/0&quot;/switchport</td>
</tr>
<tr>
<td>To configure interface switchport mode</td>
<td>PATCH</td>
<td>Yes</td>
<td>/api/running/switch/interface/gigabitEthernet/&quot;1/0&quot;/switchport/mode</td>
</tr>
<tr>
<td>To replace interface switchport trunk allowed vlans for interface</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/switch/interface/gigabitEthernet/&quot;1/0&quot;/switchport/trunk/allowed/vlan</td>
</tr>
<tr>
<td>To delete interface switchport protected-port</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/interface/gigabitEthernet/&quot;1/0&quot;/switchport/protected-port</td>
</tr>
<tr>
<td>To delete interface switchport mode</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/interface/gigabitEthernet/&quot;1/0&quot;/switchport/mode</td>
</tr>
<tr>
<td>To delete interface switchport trunk allowed vlan</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/interface/gigabitEthernet/&quot;1/0&quot;/switchport/trunk/allowed/vlan</td>
</tr>
</tbody>
</table>

### Example for Interface GigabitEthernet Switchport APIs Payload

```xml
<gigabitEthernet><name>1/0</name><switchport><mode>trunk</mode></switchport></gigabitEthernet>
```

### Example for Interface GigabitEthernet Switchport APIs Payload

```xml
<gigabitEthernet><name>1/0</name><switchport><trunk><native><vlan>100</vlan></native></trunk></switchport></gigabitEthernet>
```

### Example for Interface GigabitEthernet Switchport APIs Payload

```xml
<gigabitEthernet><name>1/0</name><switchport><trunk><allowed><vlan><ids>502</ids><ids>503</ids></vlan></allowed></trunk></switchport></gigabitEthernet>
```

### Example for Interface GigabitEthernet Switchport APIs Payload

```xml
<gigabitEthernet><name>1/0</name><switchport><community>1</community></switchport></gigabitEthernet>
```
Example for Interface GigabitEthernet Switchport APIs Payload

```xml
gigabitEthernet
  name=1/0
  switchport
  dot1q-tunnel
  vlan=100
</gigabitEthernet>
```

Example for Interface GigabitEthernet Switchport APIs Payload

```xml
gigabitEthernet
  name=1/0
  switchport
  access
  vlan=2
</gigabitEthernet>
```

Example for Interface GigabitEthernet Switchport APIs Payload

```xml
gigabitEthernet
  name=1/0
  switchport
  protected-port
</gigabitEthernet>
```

Example: GET Interface GigabitEthernet Switchport APIs

```bash
curl -k -u admin:admin -X GET https://209.165.201.1/api/running/switch/interface/gigabitEthernet/"1/0"/switchport
```

Example: PATCH Interface GigabitEthernet Switchport APIs

```bash
curl -k -u admin:admin -d "<gigabitEthernet>
  name=1/0
  switchport
  mode=trunk
</gigabitEthernet>"
-X PATCH https://209.165.201.1/api/running/switch/interface/gigabitEthernet -H "Content-Type: application/vnd.yang.data+xml"
```

Example: PATCH Interface GigabitEthernet Switchport APIs

```bash
curl -k -u admin:admin -d "<gigabitEthernet>
  name=1/0
  switchport
  trunk
  native
  vlan=100
</gigabitEthernet>"
-X PATCH https://209.165.201.1/api/running/switch/interface/gigabitEthernet -H "Content-Type: application/vnd.yang.data+xml"
```

Example: PATCH Interface GigabitEthernet Switchport APIs

```bash
curl -k -u admin:admin -d "<gigabitEthernet>
  name=1/0
  switchport
  trunk
  allowed
  vlan
  ids=502
  ids=503
</gigabitEthernet>"
-X PATCH https://209.165.201.1/api/running/switch/interface/gigabitEthernet -H "Content-Type: application/vnd.yang.data+xml"
```

Example: PUT Interface GigabitEthernet Switchport APIs

```bash
curl -k -v -u admin:admin -d "<gigabitEthernet>
  name=1/0
  switchport
  trunk
  allowed
  vlan
  ids=52
  ids=507
</gigabitEthernet>"
-X PUT https://209.165.201.1/api/config/switch/interface/gigabitEthernet/"1/0" -H "Content-Type: application/vnd.yang.data+xml"
```

Example: PATCH Interface GigabitEthernet Switchport APIs

```bash
curl -k -u admin:admin -d "<gigabitEthernet>
  name=1/0
  switchport
  community=1
</gigabitEthernet>"
-X PATCH https://209.165.201.1/api/running/switch/interface/gigabitEthernet -H "Content-Type: application/vnd.yang.data+xml"
```

Example: PATCH Interface GigabitEthernet Switchport APIs

```bash
curl -k -u admin:admin -d "<gigabitEthernet>
  name=1/0
  switchport
  dot1q-tunnel
  vlan=100
</gigabitEthernet>"
-X PATCH https://209.165.201.1/api/running/switch/interface/gigabitEthernet -H "Content-Type: application/vnd.yang.data+xml"
```
Example: PATCH Interface GigabitEthernet Switchport APIs

curl -k -u admin:admin -d 
"<gigabitEthernet><name>1/0</name><switchport><access><vlan>2</vlan></access></switchport></gigabitEthernet>
-X PATCH https://209.165.201.1/api/running/switch/interface/gigabitEthernet -H "Content-Type: application/vnd.yang.data+xml"

Example: PATCH Interface GigabitEthernet Switchport APIs

curl -k -u admin:admin -d 
"<gigabitEthernet><name>1/0</name><switchport><protected-port></protected-port></switchport></gigabitEthernet>
-X PATCH https://209.165.201.1/api/running/switch/interface/gigabitEthernet -H "Content-Type: application/vnd.yang.data+xml"

Example: DELETE Interface GigabitEthernet Switchport APIs

curl -k -u admin:admin -X DELETE
https://209.165.201.1/api/running/switch/interface/gigabitEthernet/"1/0"/switchport/protected-port

Example: DELETE Interface GigabitEthernet Switchport APIs

curl -k -u admin:admin -X DELETE
https://209.165.201.1/api/running/switch/interface/gigabitEthernet/"1/0"/switchport/mode

Example: DELETE Interface GigabitEthernet Switchport APIs

curl -k -u admin:admin -X DELETE
https://209.165.201.1/api/running/switch/interface/gigabitEthernet/"1/0"/switchport/trunk/allowed/vlan/
# Interface GigabitEthernet Spanning-Tree APIs

## Table 127: Interface GigabitEthernet Spanning-Tree APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To retrieve interface spanning-tree configuration</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/interface/gigabitEthernet/1/0/ spanning-tree</td>
</tr>
<tr>
<td>To configure interface spanning-tree element</td>
<td>PATCH</td>
<td>Yes</td>
<td>/api/running/switch/interface/gigabitEthernet</td>
</tr>
<tr>
<td>To configure the interface spanning-tree to guard the interface from becoming a root port</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/running/switch/interface/gigabitEthernet/1/0/ spanning-tree/guard/root</td>
</tr>
<tr>
<td>To delete the spanning-tree root guard</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/interface/gigabitEthernet/1/0/ spanning-tree/guard/root</td>
</tr>
<tr>
<td>To delete interface spanning-tree element</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/interface/gigabitEthernet/1/0/ spanning-tree</td>
</tr>
</tbody>
</table>

**Example for Interface GigabitEthernet Spanning-Tree APIs Payload**

```
<gigabitEthernet><name>1/0</name><spanning-tree><cost>2000</cost></spanning-tree></gigabitEthernet>
```

**Example for Interface GigabitEthernet Spanning-Tree APIs Payload**

```
<gigabitEthernet><name>1/0</name><spanning-tree><guard><root></root></guard></spanning-tree></gigabitEthernet>
```

**Example for Interface GigabitEthernet Spanning-Tree APIs Payload**

```
<gigabitEthernet><name>1/0</name><description>GigabitEthernet_slot_1_port_0</description></gigabitEthernet>
```

**Example for Interface GigabitEthernet Spanning-Tree APIs Payload**

```
<gigabitEthernet><name>1/0</name><bridge><multicast><unregistered><filtering></unregistered></filtering></multicast></bridge></gigabitEthernet>
```
Example: GET Interface GigabitEthernet Spanning-Tree APIs

curl -k -u admin:admin -X GET https://209.165.201.1/api/running/switch/interface/gigabitEthernet/"1/0"/spanning-tree

Example: PATCH Interface GigabitEthernet Spanning-Tree APIs

curl -k -u admin:admin -d "<gigabitEthernet><name>1/0</name><spanning-tree><cost>2000</cost></spanning-tree></gigabitEthernet>" -X PATCH https://209.165.201.1/api/running/switch/interface/gigabitEthernet -H "Content-Type: application/vnd.yang.data+xml"

Example: PATCH Interface GigabitEthernet Spanning-Tree APIs

curl -k -u admin:admin -d "<gigabitEthernet><name>1/0</name><spanning-tree><guard><root></root></guard></spanning-tree></gigabitEthernet>" -X PATCH https://209.165.201.1/api/running/switch/interface/gigabitEthernet -H "Content-Type: application/vnd.yang.data+xml"

Example: PUT Interface GigabitEthernet Spanning-Tree APIs

curl -k -u admin:admin -d "<gigabitEthernet><name>1/0</name><description>GigabitEthernet_slot_1_port_0</description></gigabitEthernet>" -X PUT https://209.165.201.1/api/running/switch/interface/gigabitEthernet/"1/0" -H "Content-Type: application/vnd.yang.data+xml"

Example: PATCH Interface GigabitEthernet Spanning-Tree APIs

curl -k -u admin:admin -d "<gigabitEthernet><name>1/0</name><bridge><multicast><unregistered>filtering</unregistered></multicast></bridge></gigabitEthernet>" -X PATCH https://209.165.201.1/api/running/switch/interface/gigabitEthernet -H "Content-Type: application/vnd.yang.data+xml"

Example: DELETE Interface GigabitEthernet Spanning-Tree APIs

curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/interface/gigabitEthernet/"1/0"/spanning-tree/guard/root

Example: DELETE Interface GigabitEthernet Spanning-Tree APIs

curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/interface/gigabitEthernet/"1/0"/spanning-tree/cost
## SPAN/RSPAN APIs

Table 128: SPAN/RSPAN APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To add source vlan to SPAN session</td>
<td>PATCH</td>
<td>Yes</td>
<td>/api/config/switch/monitor</td>
</tr>
<tr>
<td>To replace existing vlan with current source vlan to SPAN session</td>
<td>PUT</td>
<td>Yes</td>
<td>/api/config/switch/monitor</td>
</tr>
<tr>
<td>To delete source vlan in a SPAN session</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/switch/monitor/session/&quot;1&quot;/source_vlan</td>
</tr>
<tr>
<td>To delete source interface in a SPAN session</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/switch/monitor/session/&quot;1&quot;/source_interfaces/gigabitEthernet/&quot;1/0&quot;</td>
</tr>
<tr>
<td>To delete destination interface in a SPAN session</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/monitor/session/&quot;2&quot;/destination</td>
</tr>
<tr>
<td>To delete source vlan to RSPAN session</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/switch/monitor/session/&quot;1&quot;/source_remote</td>
</tr>
<tr>
<td>To delete destination vlan and reflector-port to RSPAN session</td>
<td>DELETE</td>
<td>No</td>
<td>/api/config/switch/monitor</td>
</tr>
<tr>
<td>To show source session</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/monitor/session/&quot;1&quot;/source</td>
</tr>
<tr>
<td>To show destination session</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/monitor/session/&quot;4&quot;/destination</td>
</tr>
</tbody>
</table>

Example for SPAN/RSPAN APIs Payload

```
<monitor><session><session-id>1</session-id><source><vlan>5</vlan></source></session></monitor>
```

Example for SPAN/RSPAN APIs Payload

```
<monitor><session><session-id>1</session-id><source><interfaces><gigabitEthernet><name>1/0</name><direction>both</direction></gigabitEthernet></interfaces></source></session></monitor>
```
Example for SPAN/RSPAN APIs Payload

```xml
<monitor><session><session-id>1</session-id><source><remote><vlan>20</vlan><remote></source></session><session></monitor>
```

Example for SPAN/RSPAN APIs Payload

```xml
<monitor><session><session-id>4</session-id><destination><remote><vlan>20</vlan><reflector-port></reflector-port><gigabitEthernet>1/4</gigabitEthernet><network></network></remote></destination><session></monitor>
```

Example: PATCH SPAN/RSPAN APIs

```
curl -k -v -u admin:admin -d
"<monitor><session><session-id>1</session-id><source><vlan>5</vlan></source></session></monitor>"
-X PATCH https://209.165.201.1/api/config/switch/monitor -H "Content-Type:
application/vnd.yang.data+xml"
```

Example: PUT SPAN/RSPAN APIs

```
curl -k -v -u admin:admin -d
"<monitor><session><session-id>1</session-id><source><vlan>6</vlan></source></session></monitor>"
-X PUT https://209.165.201.1/api/config/switch/monitor -H "Content-Type:
application/vnd.yang.data+xml"
```

Example: PATCH SPAN/RSPAN APIs

```
curl -k -v -u admin:admin -d
"<monitor><session><session-id>1</session-id><source><interfaces><gigabitEthernet><name>1/0</name><direction>both</direction></gigabitEthernet></interfaces></source></session></monitor>"
-X PATCH https://209.165.201.1/api/config/switch/monitor -H "Content-Type:
application/vnd.yang.data+xml"
```

Example: PATCH SPAN/RSPAN APIs

```
curl -k -v -u admin:admin -d
"<monitor><session><session-id>1</session-id><source><interfaces><gigabitEthernet><name>1/1</name><direction>rx</direction></gigabitEthernet></interfaces></source></session></monitor>"
-X PATCH https://209.165.201.1/api/config/switch/monitor -H "Content-Type:
application/vnd.yang.data+xml"
```

Example: PATCH SPAN/RSPAN APIs

```
curl -k -v -u admin:admin -d
"<monitor><session><session-id>1</session-id><source><remote><vlan>20</vlan></remote></source></session></monitor>"
-X PATCH https://209.165.201.1/api/config/switch/monitor -H "Content-Type:
application/vnd.yang.data+xml"
```

Example: PATCH SPAN/RSPAN APIs

```
curl -k -v -u admin:admin -d
"<monitor><session><session-id>4</session-id><destination><remote><vlan>20</vlan><reflector-port></reflector-port><gigabitEthernet>1/4</gigabitEthernet><network></network></remote></destination></session></monitor>"
-X PATCH https://209.165.201.1/api/config/switch/monitor -H "Content-Type:
application/vnd.yang.data+xml"
```

Example: DELETE SPAN/RSPAN APIs

```
curl -k -v -u admin:admin -X DELETE
https://209.165.201.1/api/config/switch/monitor/session/"1"/source/vlan
-H "Content-Type:
application/vnd.yang.data+xml"
```

Example: DELETE SPAN/RSPAN APIs

```
curl -k -v -u admin:admin -X DELETE
https://209.165.201.1/api/config/switch/monitor/session/"1"/source/interfaces/gigabitEthernet/"1/0"
```
Example: DELETE SPAN/RSPAN APIs

curl -k -v -u admin:admin -X DELETE 
https://172.252.189/api/running/switch/monitor/session/"2"/destination

Example: DELETE SPAN/RSPAN APIs

curl -k -v -u admin:admin -X DELETE 
https://209.165.201.1/api/config/switch/monitor/session/"1"/source/remote -H "Content-Type: application/vnd.yang.data+xml"

Example: DELETE SPAN/RSPAN APIs

curl -k -v -u admin:admin -d 
"<session><session-id>1</session-id><destination><remote><vlan>20</vlan><reflector-port></reflector-port><gigabitEthernet>1/4</gigabitEthernet><network></network></remote></destination></session>"  
-X DELETE https://209.165.201.1/api/config/switch/monitor -H "Content-Type: application/vnd.yang.data+xml"

Example: GET SPAN/RSPAN APIs

curl -k -v -u admin:admin -X GET 
https://209.165.201.1/api/running/switch/monitor/session/"1"/source

Example: GET SPAN/RSPAN APIs

curl -k -v -u admin:admin -X GET 
https://209.165.201.1/api/running/switch/monitor/session/"4"/destination
# CHAPTER 25

## VLAN and interface VLAN related APIs

### Table 129: VLAN and interface VLAN related APIs

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>Payload Required</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>To create VLAN</td>
<td>POST</td>
<td>Yes</td>
<td>/api/running/switch</td>
</tr>
<tr>
<td>To delete VLAN</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/vlan/90</td>
</tr>
<tr>
<td>To create interface VLAN</td>
<td>POST</td>
<td>Yes</td>
<td>/api/running/switch/interface</td>
</tr>
<tr>
<td>To display the VLAN configuration</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/vlan?deep</td>
</tr>
<tr>
<td>To displace all the interface VLAN</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/interface/vlan?deep</td>
</tr>
<tr>
<td>configurations</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To display a specific interface VLAN</td>
<td>GET</td>
<td>No</td>
<td>/api/running/switch/interface/vlan/90</td>
</tr>
<tr>
<td>configuration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To delete the VLAN interface</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/interface/vlan/90</td>
</tr>
<tr>
<td>To configure an IP address for a VLAN</td>
<td>PATCH</td>
<td>Yes</td>
<td>/api/running/switch/interface/vlan</td>
</tr>
<tr>
<td>interface</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To delete an IP address for a VLAN interface</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/interface/vlan/20/remote-span</td>
</tr>
<tr>
<td>To show the IP interface</td>
<td>GET</td>
<td>No</td>
<td>/api/operational/switch/ip/interface</td>
</tr>
<tr>
<td>To delete remote SPAN setting for VLAN</td>
<td>DELETE</td>
<td>No</td>
<td>/api/running/switch/interface/vlan/20/remote-span</td>
</tr>
</tbody>
</table>

### Example for VLAN and interface VLAN related APIs Payload

```xml
<vlan><vlan-id>90</vlan-id></vlan>
```
Example for VLAN and interface VLAN related APIs Payload
<vlan><vlan-id>90</vlan-id><ip><address><primary><address>13.13.13.13</address><mask>255.255.255.0</mask></primary></address></ip></vlan>

Example for VLAN and interface VLAN related APIs Payload
<vlan><vlan-id>20</vlan-id><remote-span/></vlan>

Example: POST VLAN and interface VLAN related APIs
`curl -k -u admin:admin -d "<vlan><vlan-id>90</vlan-id></vlan>" -X POST https://209.165.201.1/api/running/switch -H "Content-Type: application/vnd.yang.data+xml"

Example: DELETE VLAN and interface VLAN related APIs
`curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/vlan/90`

Example: POST VLAN and interface VLAN related APIs
`curl -k -u admin:admin -d "<vlan><vlan-id>90</vlan-id></vlan>" -X POST https://209.165.201.1/api/running/switch/interface -H "Content-Type: application/vnd.yang.data+xml"

Example: GET VLAN and interface VLAN related APIs

Example: GET VLAN and interface VLAN related APIs

Example: GET VLAN and interface VLAN related APIs
`curl -k -u admin:admin -X GET https://209.165.201.1/api/running/switch/interface/vlan/90`

Example: DELETE VLAN and interface VLAN related APIs
`curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/interface/vlan/90`

Example: PATCH VLAN and interface VLAN related APIs
`curl -k -v -u admin:admin -d "<vlan><vlan-id>90</vlan-id><ip><address><primary><address>13.13.13.13</address><mask>255.255.255.0</mask></primary></address></ip><vlan></vlan>" -X PATCH https://209.165.201.1/api/running/switch/interface/vlan -H "Content-Type: application/vnd.yang.data+xml"

Example: DELETE VLAN and interface VLAN related APIs
`curl -k -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/ip/route/ip-route-forwarding-list/2.2.2.2,255.255.255.255,5.5.5.1`

Example: GET VLAN and interface VLAN related APIs
`curl -k -u admin:admin -X GET https://209.165.201.1/api/operational/switch/ip/interface
**Example: PATCH VLAN and interface VLAN related APIs**

```bash
curl -k -v -u admin:admin -d "<vlan><vlan-id>20</vlan-id><remote-span/></vlan>" -X PATCH https://209.165.201.1/api/running/switch/interface/vlan -H "Content-Type: application/vnd.yang.data+xml"
```

**Example: DELETE VLAN and interface VLAN related APIs**

```bash
curl -k -v -u admin:admin -X DELETE https://209.165.201.1/api/running/switch/interface/vlan/20/remote-span -H "Content-Type: application/vnd.yang.data+xml"
```