Resource APIs

- CPU Allocation Summary API, on page 1
- Resources CPU APIs, on page 2
- Resource Precheck APIs, on page 3
- Resources VM APIs, on page 5

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 1: 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_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
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 2: 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 3: Resource Precheck APIs

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

Resource APIs

---

3
Check if there are sufficient resources for the deployment of a VM.

<table>
<thead>
<tr>
<th>GET</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>/api/operational/resources/precheck/vnf/&lt;vnf_name&gt;,&lt;flavor_name&gt;,&lt;true or false for low-latency&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Check if there are sufficient resources for updating a deployed VM.

<table>
<thead>
<tr>
<th>GET</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>/api/operational/resources/precheck/vnf/&lt;deployment_name&gt;.&lt;vm_group_name&gt;</td>
<td></td>
</tr>
</tbody>
</table>

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>.

---

**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"

* 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/precheck/vnf/newvnf,csr1kv-large,true?deep HTTP/1.1
* 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 06:28:59 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

  <vnf-name>newvnf</vnf-name>
<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 4: Resources VM APIs

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

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
    <name>1472148662 ROUTER2</name>  
    <vcpu>2</vcpu>  
    <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>  
    <cpu>  
      <vcpu-id>11</vcpu-id>  
      <socket-id>0</socket-id>  
      <core-id>3</core-id>  
      <cpu-id>11</cpu-id>  
    </cpu>  
    <cpu>  
      <vcpu-id>10</vcpu-id>  
      <socket-id>0</socket-id>  
      <core-id>2</core-id>  
      <cpu-id>10</cpu-id>  
    </cpu>  
  </vns>  
</nfvis>
```

* Connection #0 to host 209.165.201.1 left intact