Overview

This chapter provides an overview for using the Representational State Transfer (REST) application programming interface (API) with the Cisco Nexus 1000V Series Switches.

This chapter contains the following sections:

RESTful Web Services API

The Cisco Nexus 1000V Virtual Supervisor Module (VSM) supports the RESTful webservices API and provides limited functionality through this service.

You can create, read, update, and delete an object on the Cisco Nexus 1000V VSM using the RESTful web services API. REST is based on HTTP and, therefore, these four operations are in turn mapped to GET, POST, and DELETE HTTP operations. In order to call any REST function, you can use tools such as a web browser, the cURL tool, and Windows PowerShell.

The following is the basic construct of a REST URL:

```
http[s]://<IP_address>/api/<object locator>
```

The object locator consists of two parts:

- <object locator> := <name space>/<object name>
- <name space> indicates the broader class of functions and <object name> refers to the specific object.

For example, in the following URL:

```
http://10.10.10.2/api/n1k/license
```

n1k is the namespace and license is the object name.

If you are using a browser, type in the URL. For example, if you want to get the license information of your VSM that has an IP address of 10.10.10.2, you type the URL as follows:

```
https://10.10.10.2/api/n1k/license
```

The browser prompts you for a username and a password. After entering them, you get the following output:

```
<?xml version="1.0" encoding="UTF-8" ?>
<set name="license_set"
    <instance name="NEXUS_VSG_SERVICES_PKG" url="/api/n1k/license">
        <properties>
             <expires>Never</expires>
             <type>NEXUS_VSG_SERVICES_PKG</type>
             <available>16</available>
             <status>Unused</status>
             <used>0</used>
        </properties>
    </instance>
    <instance name="NEXUS_ASA1000V_SERVICES_PKG" url="/api/n1k/license">
        <properties>
            <expires>Never</expires>
             <type>NEXUS_ASA1000V_SERVICES_PKG</type>
             <available>16</available>
             <status>Unused</status>
             <used>0</used>
        </properties>
     </instance>
```

To access the same through cURL, you use the following format:

```
curl http://username:password@10.10.10.2/api/n1k/license
```

Finding Namespace Lists and Functions

Every REST API function is associated with a namespace. Functions that are not specific to Hyper-V are under the n1k namespace. To find the list of namespaces, construct the URL as follows:

```
https://10.10.10.2/api
```

You get the following output:

```
<instance url="/api">
    <children>
        <child name="n1k" url="/api/n1k"/>
    </children>
</instance>
```

The above output tells you that there are two namespaces—`hyper-v` and `n1k`.

List of Functions Available For N1K

To find all the functions under the `n1k` namespace, you enter the following in the web browser:

https://10.10.10.2/api/n1k

You get the following output:

```
<instance url="/api/n1k">
    <children>
        <child name="network-segment-pool" url="/api/n1k/network-segment-pool"/>
        <child name="uplink" url="/api/n1k/uplink"/>
        <child name="vnic" url="/api/n1k/vnic"/>
        <child name="network-segment" url="/api/n1k/network-segment"/>
        <child name="logical-network" url="/api/n1k/logical-network"/>
```

```
        <child name="ip-pool-template" url="/api/n1k/ip-pool-template"/>
        <child name="license" url="/api/n1k/license"/>
        <child name="summary" url="/api/n1k/summary"/>
        <child name="hyper-v" url="/api/n1k/hyper-v"/>
        <child name="port-profile" url="/api/n1k/port-profile"/>
        <child name="vem" url="/api/n1k/vem"/>
        <child name="uplink-port-profile" url="/api/n1k/uplink-port-profile"/>
        <child name="virtual-port-profile" url="/api/n1k/virtual-port-profile"/>
    </children>
</instance>
```

Response Description

Each child shows the available functions under the current `n1k` namespace.

Keyword	Description
`name`	Name of a function.
`uri`	Relative uniform resource identifier (URI).

List of Functions Available for Hyper-V

```
<instance url="/api/n1k/hyper-v">
    <children>
        <child name="vsem-system-info" url="/api/n1k/hyper-v/vsem-system-info"/>
        <child name="vm-network" url="/api/n1k/hyper-v/vm-network"/>
        <child name="switch-extension-info" url="/api/n1k/hyper-v/switch-extension-info"/>
    </children>
</instance>
```

Create, Read, Update, and Delete Operations

Creating an Object

To create an object, you must construct an HTTP POST request:

```
https://<IP address>/api/<name space>/<object locator>
```

The request must have a payload that contains JavaScript Object Notation- (JSON)-formatted fields that are part of the newly created object:

```
\{"<property>":"<value>", "<property>":"<value>", .....\}
```

For example, to create an IP address pool with a pool name of pool1 on a VSM with an IP address of 10.10.10.2, send a POST request by entering the following:

Use cURL to perform the following:

```
curl -u admin:Sfish123 10.10.10.2/api/n1k/ip-pool-template -d ' {"name":"pool1",
"addressRangeStart":"192.168.0.2" , "addressRangeEnd":"192.168.0.16"} '
```

The same can be obtained through a browser using addons such as the REST console.

Use PowerShell to perform the following:

```
 #Basic parameters required for accessing REST-APIs
 $User = "admin"
```

```
 $Password = ConvertTo-SecureString -String "Secret123" -AsPlainText -Force
 $VSMIPaddress = "10.10.10.2"
 $URI = "http://"+ $VSMIPaddress
 $Credential = New-Object -TypeName System.Management.Automation.PSCredential
-ArgumentList  $User, $Password
 #Create-
 $args = ' {"name" : "pool1" , "addressRangeStart":"192.168.0.2" ,
"addressRangeEnd":"192.168.0.16"}
 ConvertFrom-Json -InputObject $args
 Invoke-RestMethod -Uri http://10.10.10.2/api/n1k/ip-pool-template  -Credential
$Credential -Method Post -Body $args
```

To find out the valid property names for a given function, see the Response Sample in each function definition later in this document.

Reading an Object

To read an object, you must construct an HTTP GET request:

```
https://<IP address>/api/<name space>/<object locator>/<instance name>
```

For example, to read switch extension manager information from a VSM with an IP address of 10.10.10.2, send a GET request by entering the following:

Use cURL to perform the following:

```
curl -u admin:Sfish123 10.10.10.2/api/n1k/hyper-v/vsem-system-info
```

Use PowerShell to perform the following:

```
#Read the VSEM info - HTTP GET
$VersionURI = $URI + "/api/hyper-v/vsem-system-info"
Invoke-RestMethod -Uri $VersionURI -Credential $Credential -Method Get  -Outfile
testout.xml
```

Sends the results to the specified output file. In this case, "testout.xml" . Enter \ a path and file name. If you omit the path, the default is the current location.

Updating an Object

To update an object, you must construct an HTTP POST request:

```
https://<IP address>/api/<name space>/<object locator>/<instance name>
```

The request must have a payload that contains JSON-formatted fields that are updated on the object:

```
\{"<property>":"<value>", "<property>":"<value>", .....\}
```

For example, to modify the address range for the IP pool named pool1 on a VSM with an IP address of 10.10.10.2, send a POST request by entering the following IP pool:

Use cURL to perform the following:

```
curl -u admin:Sfish123 10.10.10.2/api/n1k/ip-pool-template/pool1 -d ' {"name":"pool1",
"addressRangeStart":"192.168.0.5"  ,"addressRangeEnd":"192.168.0.20"} '
```

Use PowerShell to perform the following:

```
#Update the IP-address Pool Information - HTTP POST
$IPPURI = $URI + "/api/nik/ip-pool-template/pool1"
$IPPArg = '{"addressRangeStart":"192.168.0.5", "addressRangeEnd":"192.168.0.20"}'
```

```
ConvertFrom-Json -InputObject $IPPArg
Invoke-RestMethod -Uri $IPPURI -Credential $Credential -Method Post -Body $IPPArg
```

To find the valid property names for a given function, see the Response Sample in each function definition later in this document.

Deleting an Object

To delete an object, you must construct an HTTP DELETE request:

```
https://<IP address>/api/<name space>/<object locator>/<instance name>
```

To delete a network segment named VMN4 from a VSM with an IP address of 10.10.10.2, send a DELETE request by entering the following:

Use cURL to perform the following:

```
curl -u admin:Sfish123 10.10.10.2/api/n1k/network-segment/VMN4 -X DELETE
```

Use PowerShell to perform the following:

```
#Delete a network segment - HTTP Delete
$VMNURI = $URI + "/api/n1k/network-segment/VMN4"
Invoke-RestMethod -Uri $VMNURI -Credential $Credential -Method Delete
```