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.

REST tunneling is a special resource api/cli to which CLI commands may be posted to HTTP (HTTP POST) and CLI responses are returned. For example, to create a port-profile you type the following:

```
curl -u admin:Secret123 10.193.196.201/api/cli -d '{"cmd": "config ; port-profile pp1 ;
description pp1 ; copy r s "}'
```

The following is the basic construct of a REST URL:

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

The resource locator consists of two parts:

- <resource 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, n1k is the namespace and license is the object name.

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

If you are using curl or a browser extension such as REST Console to issue REST calls, type 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
```

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

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

OR

```
curl -u username:password 10.10.10.2/api/n1k/license
```

Caution When using the reload command through the REST tunnel, you need to make sure that the configuration is saved before using it. Otherwise the changes are lost.

Note You can view detailed information pertaining to all the REST resources. To get this information, you must append ?meta to the URI.

```
http://<IP_address>/api/<resource_name>?meta
```

Finding Namespace Lists and Functions

Every REST API function is associated with a namespace. Functions that are specific to Nexus 1000V on Hyper-V are under the n1k namespace. The n1k namespace is found when the following URL is entered:

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

You get the following output:

```
<?xml version="1.0" encoding="utf-8"?>
<instance url="/api">
<children>
<child name="span" url="/api/span"/>
<child name="vpath" url="/api/vpath"/>
<child name="user" url="/api/user"/>
<child name="port-profile" url="/api/port-profile"/>
<child name="n1k" url="/api/n1k"/>
<child name="vlan" url="/api/vlan"/>
<child name="vnode" url="/api/vnode"/>
</children>
</instance>
```

In the above output, span, vpath, user, port-profile, vlan, and vnode are the functions that are a part of the global namespace and n1k is the namespace that includes the Hyper-V specific functions.

List of Functions Available For Nexus 1000V on Hyper-V

To find all the functions under the n1k namespace, enter the following URL:

```
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.
`url`	Relative uniform resource locator (URL).

To find the functions under the Hyper-V namesapce, enter the following URL:

```
https://10.10.10.2/api/n1k/hyper-v
```

You get the following output:

```
<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>
```

Supported Response Formats

JSON Format

The REST API supports the JavaScript Object Notation (JSON) format for a response. For JSON response, specify `Accept: application/json` in the HTTP header as shown in the following example:

```
GET /api/vc/summary HTTP/1.1
Host: 10.10.10.2
Accept: application/json
Authorization: Basic YWRtaW46U2Zpc2gxMjM=
```

To specify the JSON response format through cURL, use the following:

```
curl -u <user>:<password> <vsm_ip>/api/n1k/license -H "Accept: application/json"
```

The following example shows the response received in the JSON format:

```
{
"NEXUS1000V_LAN_SERVICES_PKG": {
"url": "/api/n1k/license/NEXUS1000V_LAN_SERVICES_PKG",
"properties": {
"expires": "01 Feb 2014",
"type": "NEXUS1000V_LAN_SERVICES_PKG",
"available": 1022,
"status": "In use",
"used": 0
}
}
}
```


XML Format

The REST API supports the XML format for a response. For XML response, specify `Accept: application/xml` in the HTTP header as shown in the following example:

```
GET /api/vc/summary HTTP/1.1
Host: 10.10.10.2
Accept: application/xml
Authorization: Basic YWRtaW46U2Zpc2gxMjM=
```

To specify the XML response format through cURL, use the following:

```
curl -u <user>:<password> <vsm_ip>/api/n1k/summary -H "Accept: application/xml"
```

The following example shows the response received in the XML format:

```
<?xml version="1.0" encoding="utf-8"?>
<instance url="/api/n1k/summary">
<properties>
<haStatus>false</haStatus>
<name>N1k-SITE-MANAGER</name>
<switchMode>Advanced</switchMode>
<mode>L3</mode>
<version>version 5.2(1)SM1(5.2)</version>
<ip>10.106.196.249</ip>
</properties>
</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>/<resource 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:Secret123 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-
 $args1 = ' {"name" : "pool1" , "addressRangeStart":"192.168.0.2" ,
"addressRangeEnd":"192.168.0.16"}
 Invoke-RestMethod -Uri http://10.10.10.2/api/n1k/ip-pool-template-Credential
$Credential -Method Post-Body $args1
```

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>/<resource 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:Secret123 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/n1k/hyper-v/vsem-system-info"
Invoke-RestMethod -Uri $VersionURI -Credential $Credential -Method Get  -Outfile
testout.xml
```

The above PowerShell command sends the results to the specified output file, which in this case,is "testout.xml" . The default location where the outfile gets saved is the current folder location.

Updating an Object

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

```
https://<IP address>/api/<name space>/<resource 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:Secret123 10.10.10.2/api/n1k/ip-pool-template/pool1 -d ' {"name"
"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/n1k/ip-pool-template/pool1"
$IPPArg = '{"addressRangeStart":"192.168.0.5", "addressRangeEnd":"192.168.0.20"}'
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>/<resource 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:Secret123 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
```