



DevNet Coffee Break NXAPI

Cisco Connect Croatia 2017

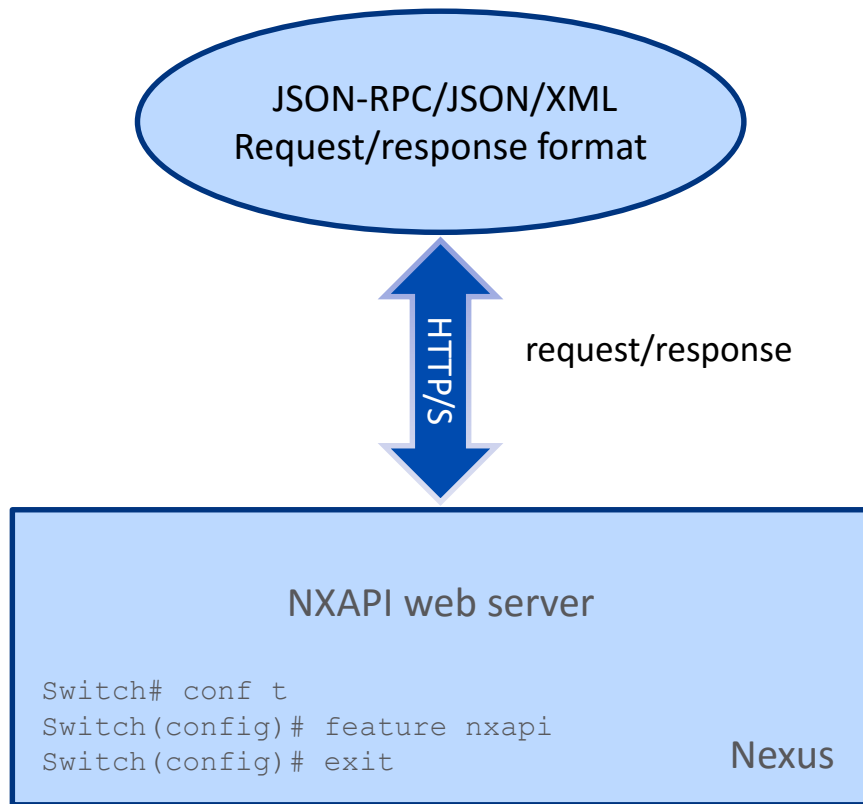
Djordje Vulovic

Consulting Systems Engineer

dvulovic@cisco.com

https://github.com/djordjevulovic/Cisco_Connect_HR_2017

Introducing NX-API



DevNet Sandbox Labs

The screenshot displays the DevNet Sandbox Labs interface. The top navigation bar includes the DevNet logo, a 'LAB MANAGEMENT' tab, and links for DVULOVIC, DEVNET, HELP, and TUTORIALS. The main content area features a grid of lab cards. The first row includes IOx (Version 1.3), Jabber Guest, Mantl (Version 10.0), MediaSense (Version 10.0), and Meraki. The second row includes NX-API (highlighted with a red box), Open NX-OS (Version 7.0(3)S1), OpenPlatform NFV CentOS, OpenPlatform NFV Ubuntu, and OpenPnP. Each card shows a title, a brief description, and a button to either 'RESERVE' or 'ALWAYS ON'. The URL in the browser address bar is <https://devnetsandbox.cisco.com/RM/Topology>. The URL in the footer is <https://devnetsandbox.cisco.com/RM/Diagram/index/54657202-fa36-45d4-ac1c-f02ba6b313...>. A 'Back to top' link is located in the bottom right corner.

Lab Name	Version	Description	Action
IOx	Version 1.3	IOx-CAF x2 + Fog Director	RESERVE
Jabber Guest		Jabber Guest Lab	RESERVE
Mantl	Version 10.0	Cisco Mantl MicroServices Infrastructure Solution	ALWAYS ON
MediaSense	Version 10.0	Explore Cisco MediaSense in your own	RESERVE
Meraki		Meraki Always On	ALWAYS ON
NX-API		Simplified switch Interaction using virtualized NX-OS (NX-API)	ALWAYS ON
Open NX-OS	Version 7.0(3)S1	New Standalone Only Open NX-OS Lab with virtual switch	RESERVE
OpenPlatform NFV CentOS		CentOS 7 OPNFV Brahma Putra Release	RESERVE
OpenPlatform NFV Ubuntu		Ubuntu 14.04 OPNFV Brahma Putra Release	RESERVE
OpenPnP		Open Plug N Play Lab	RESERVE



<https://devnetsandbox.cisco.com>

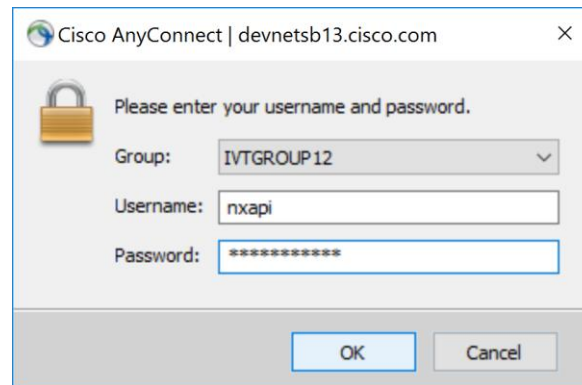
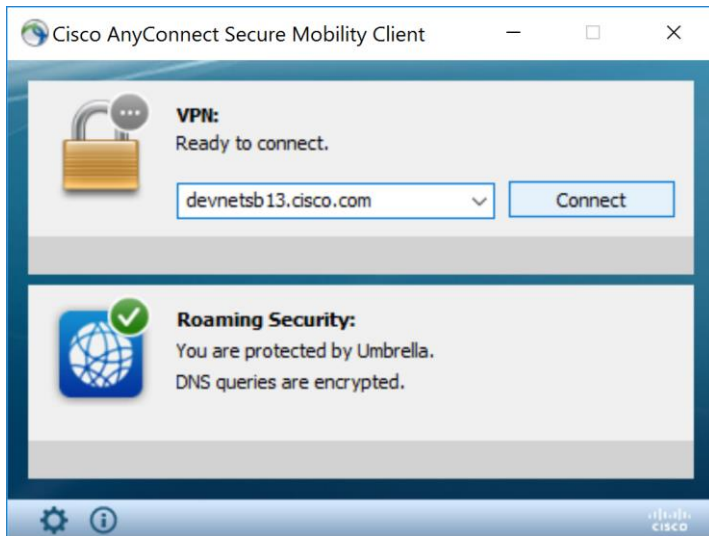
DevNet SandBox

The screenshot shows the DevNet SandBox web interface. The browser address bar displays the URL: <https://devnetsandbox.cisco.com/RM/Diagram/Index/54657202-fa36-45d4-ac1c-f02ba6b31349?diagramType=Topology>. The page header includes the DevNet logo and navigation links like 'LAB MANAGEMENT', 'DVULOVIC', 'DEVNET', 'HELP', and 'TUTORIALS'. The main content area is titled 'SANDBOX LAB (RESERVE TO ACTIVATE)' and features a 'Try It Now' button with the text 'No Reservation Required' and a link to 'See Instructions'. On the left sidebar, under 'INSTRUCTIONS', there is a section for 'NX-API Overview' with a red box highlighting the following details:

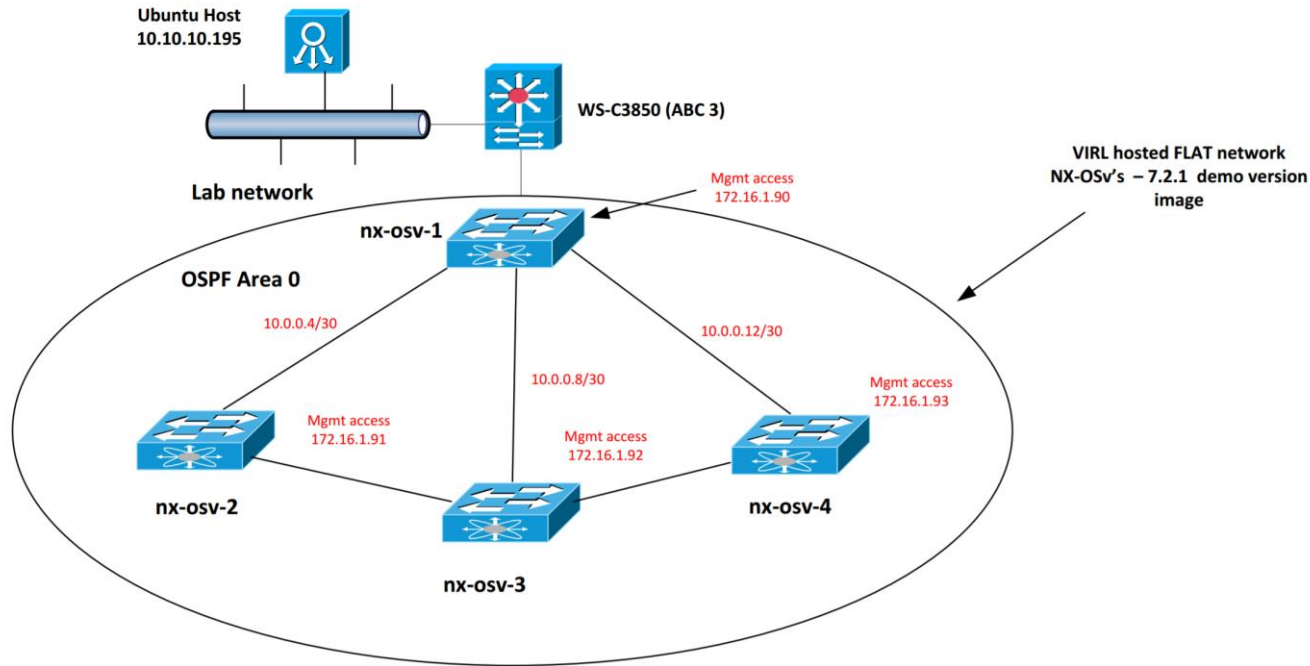
- ASA head end address: **devnetsb13.cisco.com**
- Credentials: **[nxapi/nxapi_1234!]**

Below this, the 'Hello World:' section states: 'The Hello World example application assists the developer with the basic knowledge to easily and quickly get up and running with developing using NX-OS programmability scripts.' and provides a link to <https://github.com/datacenter/nxos>. The 'Additional Information:' section lists links for 'NX-API on DevNet' and 'Sandbox Support'.

Connect to DevNet SandBox



NXAPI DevNet Sandbox Diagram



Notes

1. static route on WS3850 –
172.16.1.0/24 to VIRT host
10.10.10.190

“show version” Request

http://172.16.1.90/ins

POST /ins HTTP/1.1

Host: 172.16.1.90

Content-Type: application/json-rpc

Authorization: Basic Y2lzY286Y2lzY28=

```
{
  "jsonrpc": "2.0",
  "method": "cli",
  "params": {
    "cmd": "show version",
    "version": 1
  },
  "id": 1
}
```

Username: cisco
Password: cisco

Build Request in Postman Tool (1)

The screenshot displays the Postman interface for building a request. At the top, there is a tab labeled 'show version' with a close button (X) and a plus button (+). To the right, the environment is set to 'No Environment' with a dropdown arrow, an eye icon, and a settings gear icon.

The main request configuration area includes a dropdown menu set to 'POST' (highlighted with a green box), a text input field containing the URL 'http://172.16.1.90/ins' (highlighted with a red box), and buttons for 'Params', 'Send', and 'Save'.

Below the request configuration, there are tabs for 'Authorization' (selected), 'Headers (2)', 'Body', 'Pre-request Script', and 'Tests'. The 'Authorization' tab is highlighted with an orange underline.

In the 'Authorization' section, the 'Type' dropdown is set to 'Basic Auth' (highlighted with a blue box). To the right of this section are 'Clear' and 'Update Request' buttons (the latter is highlighted with an orange box). Below the 'Type' dropdown, there are input fields for 'Username' and 'Password', both containing the text 'cisco'. A checkbox labeled 'Show Password' is checked.

On the right side of the 'Authorization' section, there is a note: 'The authorization header will be generated and added as a custom header'. Below this note is a checkbox labeled 'Save helper data to request' which is unchecked.

Build Request in Postman Tool (2)

show version × +

No Environment ▼

▶ show version

POST ▼ http://172.16.1.90/ins Params Send ▼ Save ▼

Authorization ● Headers (2) Body ● Pre-request Script Tests Code

Key	Value	Bulk Edit	Presets ▼
<input checked="" type="checkbox"/> Content-Type	application/json-rpc		
<input checked="" type="checkbox"/> Authorization	Basic Y2lzY286Y2lzY28=		
New key	value		

Build Request in Postman Tool (3)

The screenshot displays the Postman REST client interface. At the top, a tab labeled 'show version' is active. The main area shows a POST request to the URL 'http://172.16.1.90/ins'. The 'Body' tab is selected, and the 'raw' radio button is chosen, which is highlighted with an orange box. The body content is a JSON object:

```
{
  "jsonrpc": "2.0",
  "method": "cli",
  "params": {
    "cmd": "show version",
    "version": 1
  },
  "id": 1
}
```

 This JSON code is also highlighted with an orange box. The interface includes tabs for Authorization, Headers (2), Body, Pre-request Script, and Tests. On the right, there are buttons for 'Send', 'Save', and 'Code', along with a 'No Environment' dropdown and icons for view and settings.

Execute Request in Postman Tool

The screenshot shows the Postman interface for a POST request. The request URL is `http://172.16.1.90/ins`. The headers tab is active, showing two headers: `Content-Type` with value `application/json-rpc` and `Authorization` with value `Basic Y2lzY286Y2lzY28=`. The response body is displayed in JSON format, showing a successful response with a `jsonrpc` field set to `"2.0"` and a `result` field containing a `body` object with various system information.

Request Details:

- Method: POST
- URL: `http://172.16.1.90/ins`
- Headers (2):
 - `Content-Type`: `application/json-rpc`
 - `Authorization`: `Basic Y2lzY286Y2lzY28=`

Response Details:

- Status: 200 OK
- Time: 274 ms
- Body (JSON):

```
1 {
2   "jsonrpc": "2.0",
3   "result": {
4     "body": {
5       "header_str": "Cisco Nexus Operating System (NX-OS) Software\nTAC support: http://www.cisco.com/tac\nDocuments: http://www.cisco.com/en/US/products/ps9372
/tsd_products_support_series_home.html\nCopyright (c) 2002-2015, Cisco Systems, Inc. All rights reserved.\nThe copyrights to certain works contained herein are owned
by\nother third parties and are used and distributed under license.\nSome parts of this software are covered under the GNU Public\nLicense. A copy of the license is
available at\nhttp://www.gnu.org/licenses/gpl.html.\nNX-OSv is a demo version of the Nexus Operating System\n",
6       "loader_ver_str": "N/A",
7       "kickstart_ver_str": "7.2(0)D1(1) [build 7.2(0)ZD(0.120)]",
8       "sys_ver_str": "7.2(0)D1(1) [build 7.2(0)ZD(0.120)]",
9       "kick_file_name": "bootflash:///titanium-d1-kickstart.7.2.0.ZD.0.120.bin",
10      "kick_cmpl_time": " 3/8/2015 1:00:00",
11      "kick_tmstamp": "03/08/2015 11:04:12",
12      "isan_file_name": "bootflash:///titanium-d1.7.2.0.ZD.0.120.bin",
13      "isan_cmpl_time": " 3/8/2015 1:00:00",
14      "isan_tmstamp": "03/08/2015 15:34:48",
15      "chassis_id": "NX-OSv Chassis",
```

“show version” Response

```
{
  "jsonrpc": "2.0",
  "result": {
    "body": {
      "loader_ver_str": "N/A",
      "kickstart_ver_str": "7.2(0)D1(1) [build 7.2(0)ZD(0.120)]",
      "sys_ver_str": "7.2(0)D1(1) [build 7.2(0)ZD(0.120)]",
      "kick_file_name": "bootflash:///titanium-d1-kickstart.7.2.0.ZD.0.120.bin",
      "kick_cmpl_time": " 3/8/2015 1:00:00",
      "kick_tmstamp": "03/08/2015 11:04:12",
      "isan_file_name": "bootflash:///titanium-d1.7.2.0.ZD.0.120.bin",
      "isan_cmpl_time": " 3/8/2015 1:00:00",
      "isan_tmstamp": "03/08/2015 15:34:48",
      "chassis_id": "NX-OSv Chassis",
      "module_id": "NX-OSv Supervisor Module",
      "cpu_name": "QEMU Virtual CPU version 2.0",
      "memory": 3064940,
      "mem_type": "kB",
      "proc_board_id": "TM3EE4392FB",
      "host_name": "nx-osv-1",
      "bootflash_size": 1582402,
      "kern_uptm_days": 27,
      "kern_uptm_hrs": 23,
      "kern_uptm_mins": 23,
      "kern_uptm_secs": 44,
      "manufacturer": "Cisco Systems, Inc."
    }
  },
  "id": 1
}
```

“show vlan” Request

```
POST /ins HTTP/1.1
Host: 172.16.1.90
Content-Type: application/json-rpc
Authorization: Basic Y2lzY286Y2lzY28=
{
  "jsonrpc": "2.0",
  "method": "cli",
  "params": {
    "cmd": "show vlan",
    "version": 1
  },
  "id": 1
}
```

“show vlan” Response

```
{
  "jsonrpc": "2.0",
  "result": {
    "body": {
      "TABLE_vlanbrief": {
        "ROW_vlanbrief": [
          ...
            {
              "vlanshowbr-vlanid": 167772160,
              "vlanshowbr-vlanid-utf": 10,
              "vlanshowbr-vlanname": "VLAN0010",
              "vlanshowbr-vlanstate": "active",
              "vlanshowbr-shutstate": "noshutdown",
              "vlanshowplist-ifidx": "Ethernet2/3,Ethernet3/2"
            },
          ...
        ]
      },
      ...
    }
  },
  "id": 1
}
```

Python: Class Wrapper for NXAPI show

```
class NXAPI_device:

    def __init__(self, ip, username, password):
        self.ip = ip
        self.url = "http://" + ip + "/ins"
        self.username = username
        self.password = password

    def NXAPI_cli_show(self, command):

        my_headers = {'content-type': 'application/json-rpc'}

        payload = [{"jsonrpc": "2.0",
                     "method": "cli",
                     "params": {"cmd": command,
                                "version": 1},
                     "id": 1}
                    ]

        response = requests.post(self.url, data=json.dumps(payload), headers=my_headers,
                                auth=(self.username, self.password))

        jsonObject = json.loads(response.text)

        return jsonObject
```

Python: Function to Show Net VLAN Usage

```
def show_vlan_usage(device_list, vlanid):

    print('{:10s} {:20s} {:30s}\n'.format("VLAN ID", "Device IP ", "Interface"))

    for device in device_list:
        jsonObject = device.NXAPI_cli_show('show vlan id ' +vlanid)

        if "error" not in jsonObject and jsonObject["result"] != None:

#            print(json.dumps(jsonObject, sort_keys=True, indent=4))

            if ("vlanshowplist-ifidx" in jsonObject["result"]["body"]["TABLE_vlanbriefid"]["ROW_vlanbriefid"]):
                portlist = jsonObject["result"]["body"]["TABLE_vlanbriefid"]["ROW_vlanbriefid"]["vlanshowplist-ifidx"]

                ports = portlist.split(",")

                for port in ports:

                    print('{:10s} {:20s} {:30s}'.format(str(vlanid), str(device.ip), str(port)))
```


Example App: Show Vlan Usage on Network

```
C:\Users\dvulovic\PycharmProjects\NXAPI>python NXAPI_show_vlan_network_usage.py
usage: NXAPI_show_vlan_network_usage.py [-h] vlan
NXAPI_show_vlan_network_usage.py: error: the following arguments are required: vlan
```

```
C:\Users\dvulovic\PycharmProjects\NXAPI>python NXAPI_show_vlan_network_usage.py 20
```

VLAN ID	Device IP	Interface
20	172.16.1.90	Ethernet2/3
20	172.16.1.91	Ethernet2/4
20	172.16.1.92	Ethernet2/4

```
C:\Users\dvulovic\PycharmProjects\NXAPI>python NXAPI_show_vlan_network_usage.py 1
```

VLAN ID	Device IP	Interface
1	172.16.1.90	Ethernet2/3
1	172.16.1.91	port-channel10
1	172.16.1.92	Ethernet2/3

```
C:\Users\dvulovic\PycharmProjects\NXAPI>python NXAPI_show_vlan_network_usage.py 10
```

VLAN ID	Device IP	Interface
10	172.16.1.90	Ethernet2/3
10	172.16.1.90	Ethernet3/2

“create vlan” Request

```
POST /ins HTTP/1.1
Host: 172.16.1.91
Content-Type: application/json-rpc
Authorization: Basic Y2lzY286Y2lzY28=
Cache-Control: no-cache
Postman-Token: cf00a124-2071-2cd2-d3b9-e22ae7a8df47
```

```
[
  {"id": "1", "method": "cli", "params": {"version": 1, "cmd": "conf t"}, "jsonrpc": "2.0"},
  {"id": "2", "method": "cli", "params": {"version": 1, "cmd": "vlan 556"}, "jsonrpc": "2.0"},
  {"id": "3", "method": "cli", "params": {"version": 1, "cmd": "name new_vlan_556"}, "jsonrpc": "2.0"},
  {"id": "4", "method": "cli", "params": {"version": 1, "cmd": "exit"}, "jsonrpc": "2.0"}
]
```

“create vlan” Response

```
[
  {
    "jsonrpc": "2.0",
    "result": null,
    "id": "1"
  },
  {
    "jsonrpc": "2.0",
    "result": null,
    "id": "2"
  },
  {
    "jsonrpc": "2.0",
    "result": null,
    "id": "3"
  },
  {
    "jsonrpc": "2.0",
    "result": null,
    "id": "4"
  }
]
```

Python: Class Wrapper for NXAPI conf

```
def NXAPI_cli_config(self, command_list):

    # jsonrpc_template = Template("{'jsonrpc': '2.0', 'method': '$method', 'params': ['$params', 1], 'id': '$jrpc_id'}")
    jsonrpc_template = Template(
        "{ 'jsonrpc': '2.0', 'method': '$method', 'params': {'cmd': '$cmd', 'version': 1}, 'id': '$jrpc_id'}")

    batch_cmd = "[" + jsonrpc_template.substitute(cmd="conf t", jrpc_id=1, method='cli')

    id_counter = 2

    for command in command_list:

        batch_cmd = batch_cmd + "," + jsonrpc_template.substitute(cmd=command, jrpc_id=id_counter, method='cli')
        id_counter += 1

    batch_cmd = batch_cmd + "," + jsonrpc_template.substitute(cmd="exit", jrpc_id=id_counter, method='cli') + "]"

    my_headers = {'content-type': 'application/json-rpc'}

    response = requests.post(self.url, data=json.dumps(ast.literal_eval(batch_cmd)), headers=my_headers,
                             auth=(self.username, self.password))

    return response
```

Python: Function to Create Net VLAN

```
def create_vlan_on_network(device_list, vlanid, vlanname):  
  
    print('Creating VLAN {} ({}).format(vlanid, vlanname))  
  
    for device in device_list:  
  
        response = device.NXAPI_cli_config(['vlan ' +vlanid,'name ' + vlanname])  
  
        if response.status_code == 200:  
            print("Device " + device.ip + " OK")  
        else:  
            print("Device " + device.ip + " FAILED")
```

