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

JSON-RPC/JSON/XML
Request/response format

HTTP/S

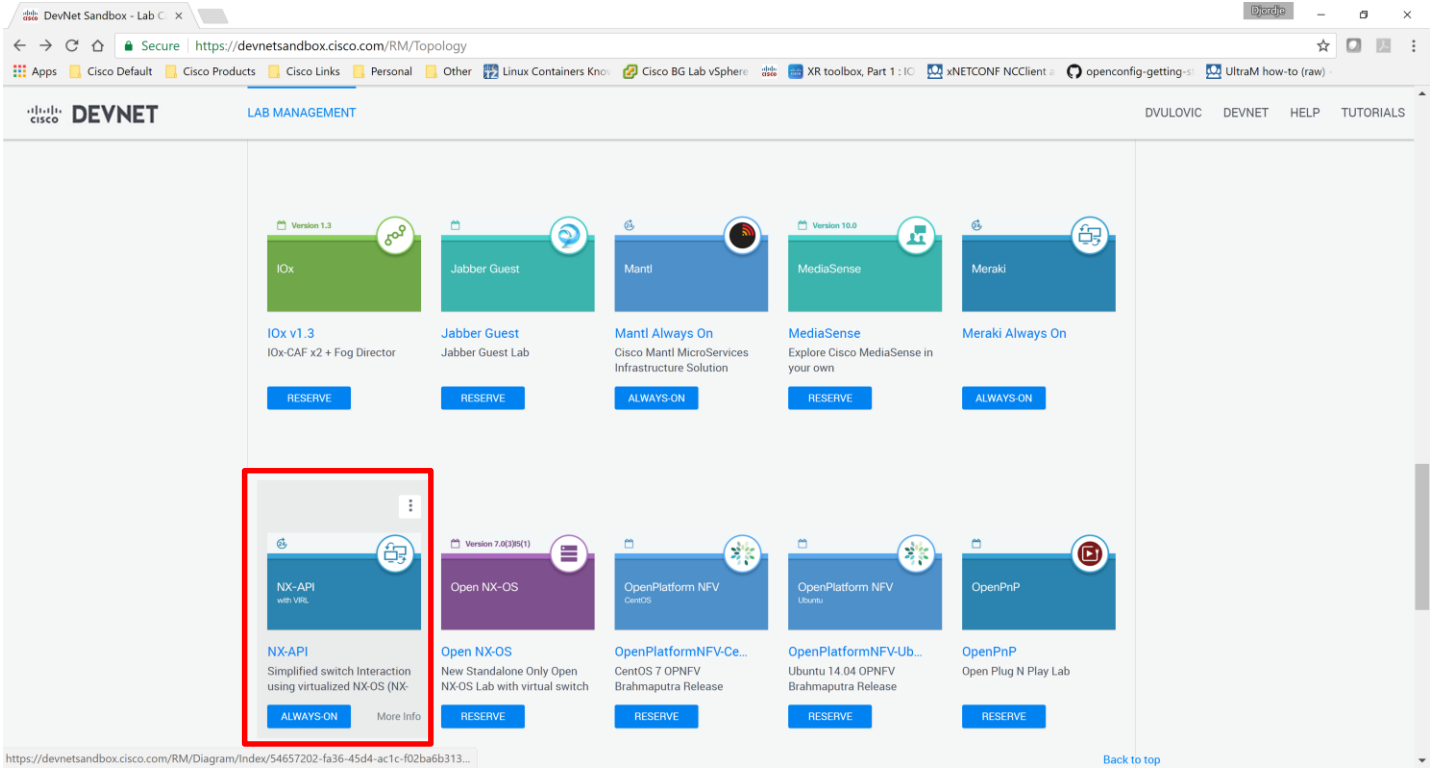
request/response

NXAPI web server

```
Switch# conf t
Switch(config)# feature nxapi
Switch(config)# exit
```

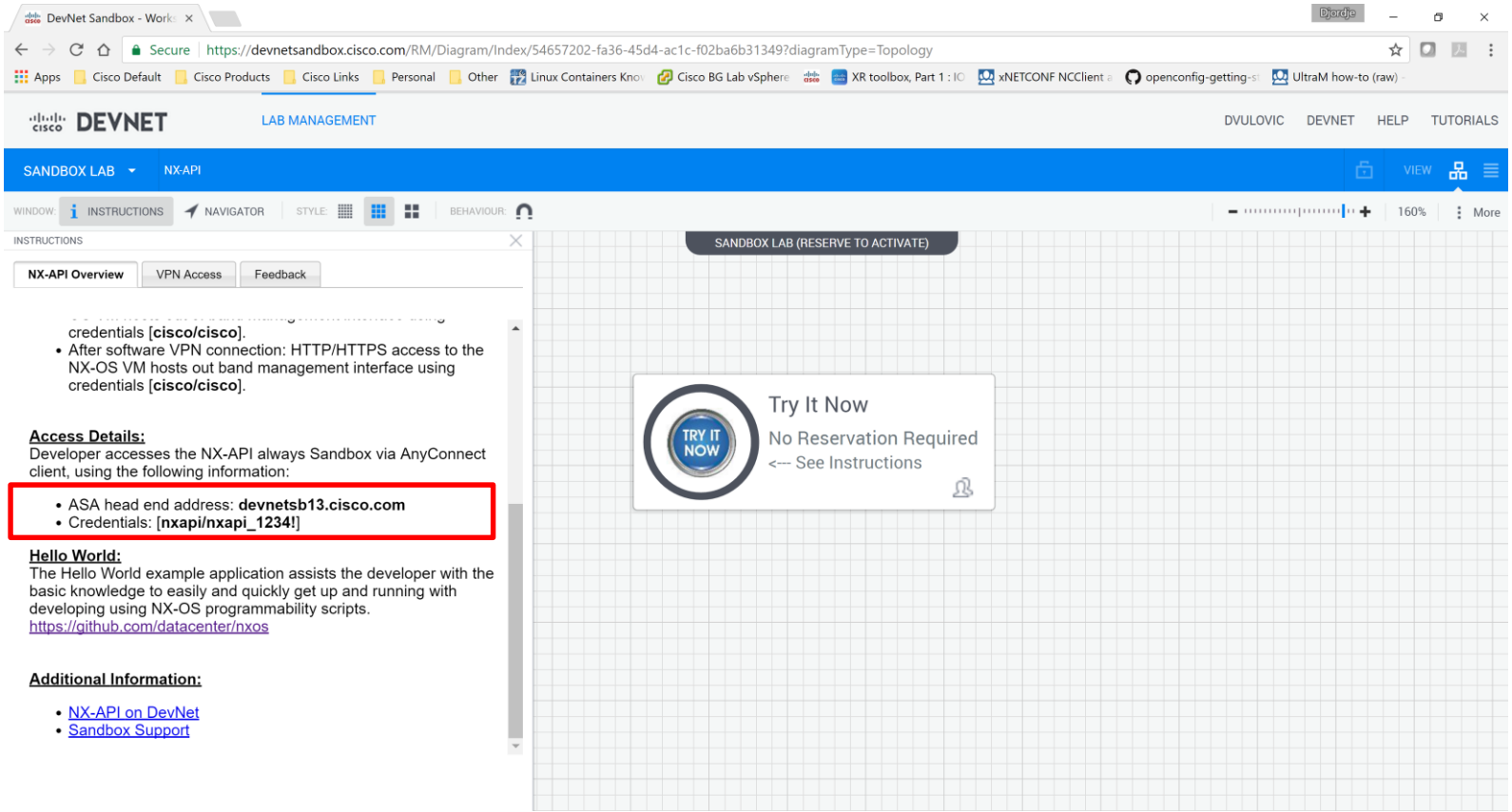
Nexus

DevNet Sandbox Labs

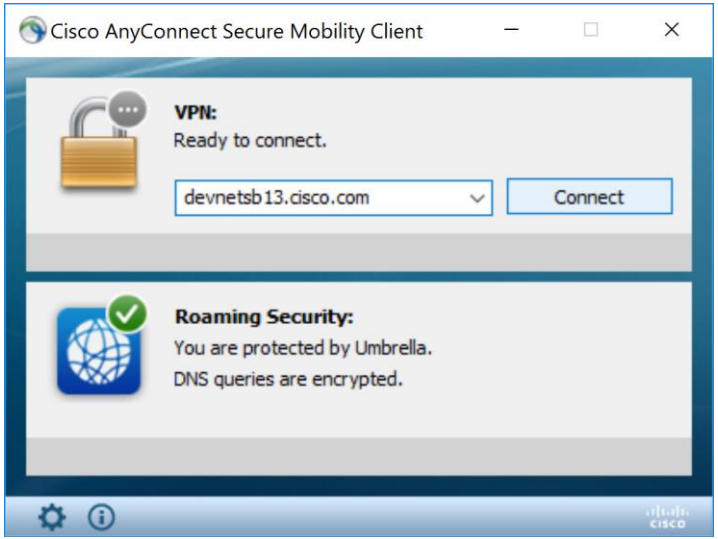


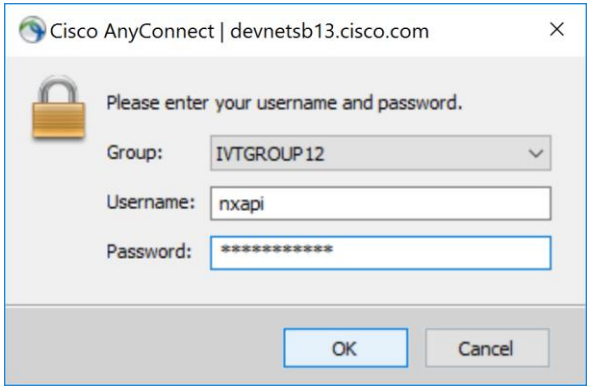
https://devnetsandbox.cisco.com

DevNet SandBox

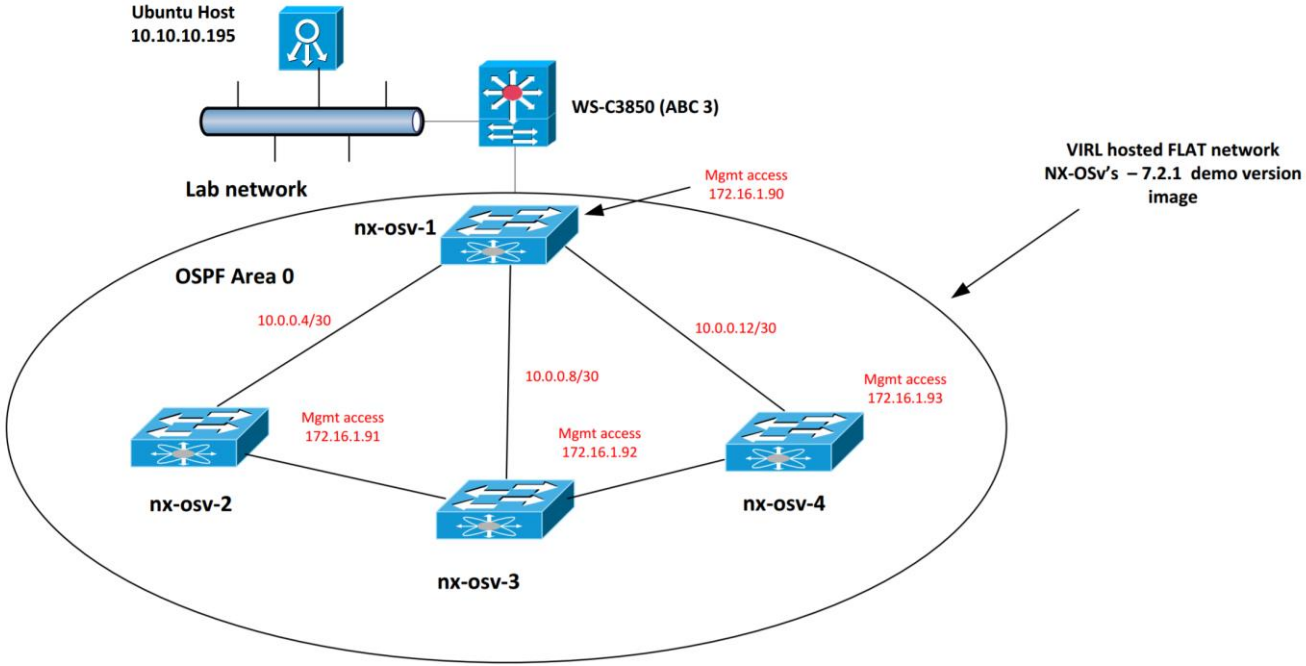


Connect to DevNet SandBox



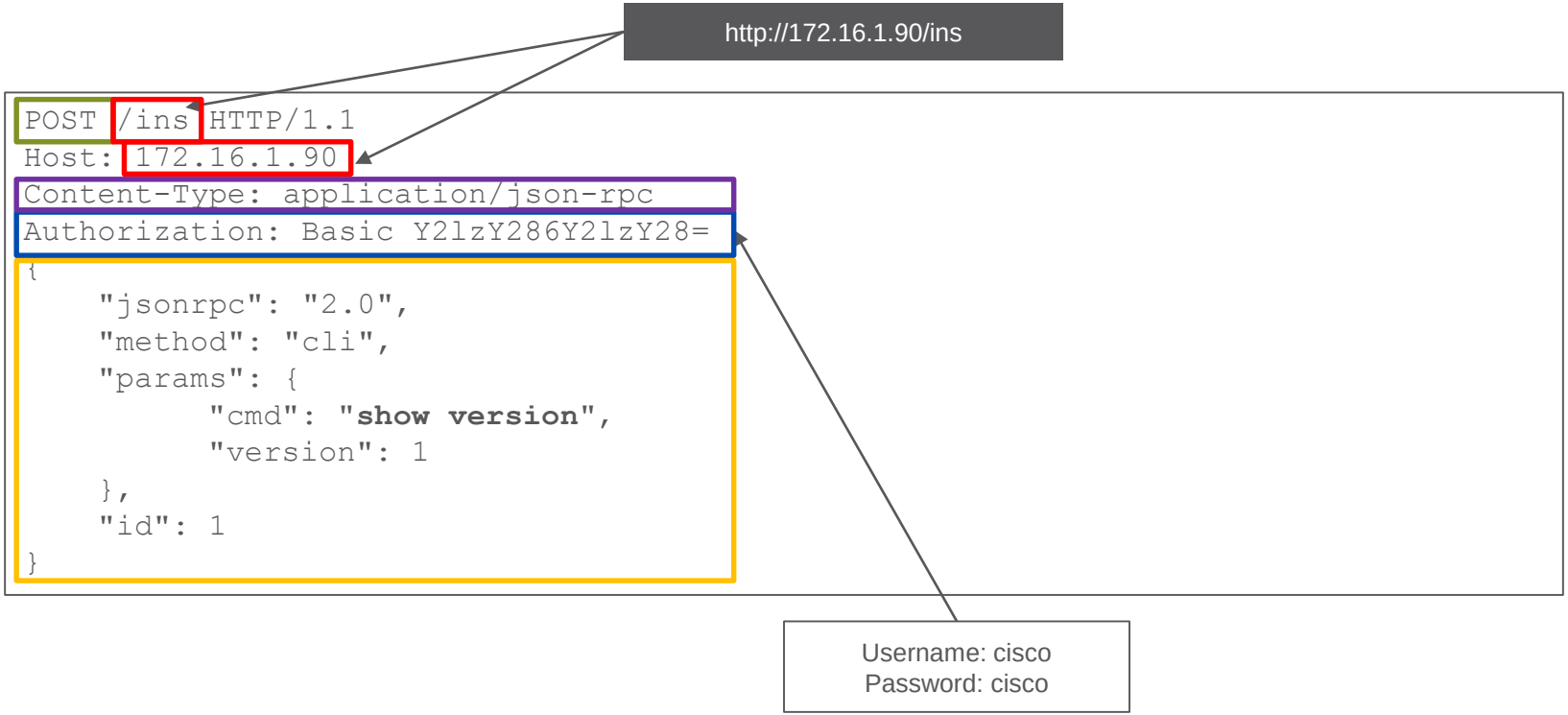


NXAPI DevNet Sandbox Diagram

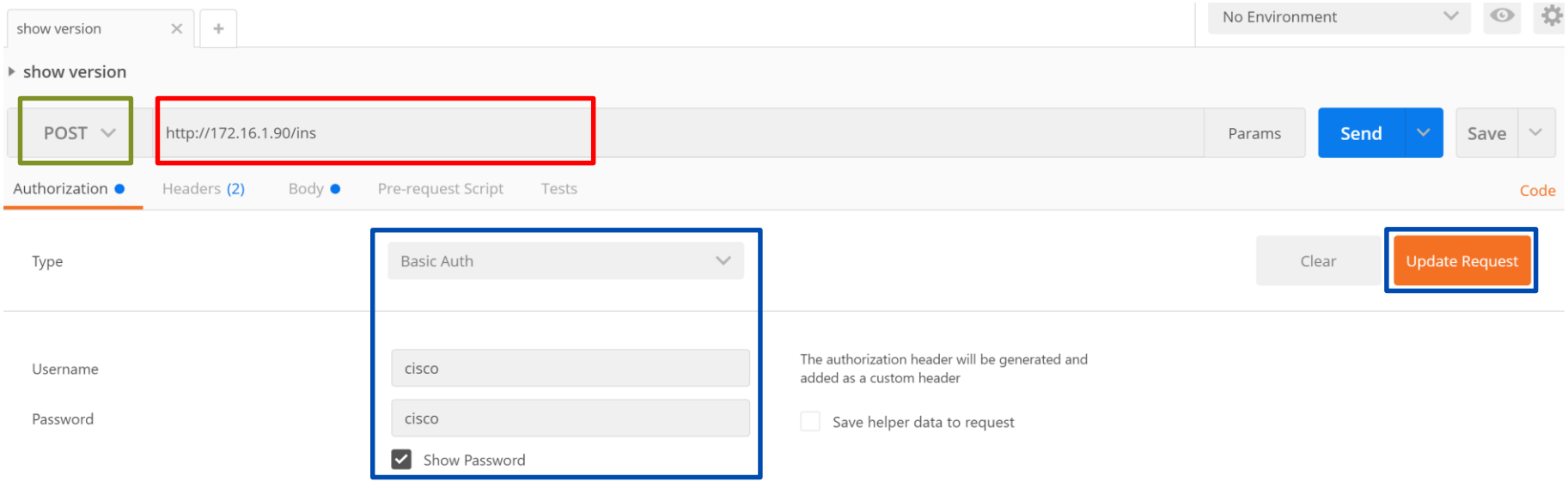


Notes
1. static route on WS3850 – 172.16.1.0/24 to VIRL host 10.10.10.190

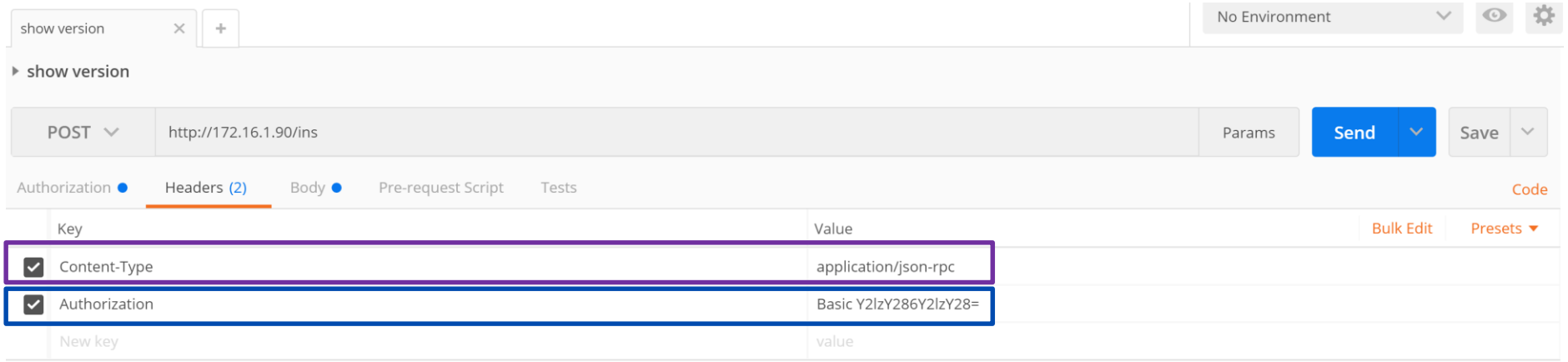
“show version” Request



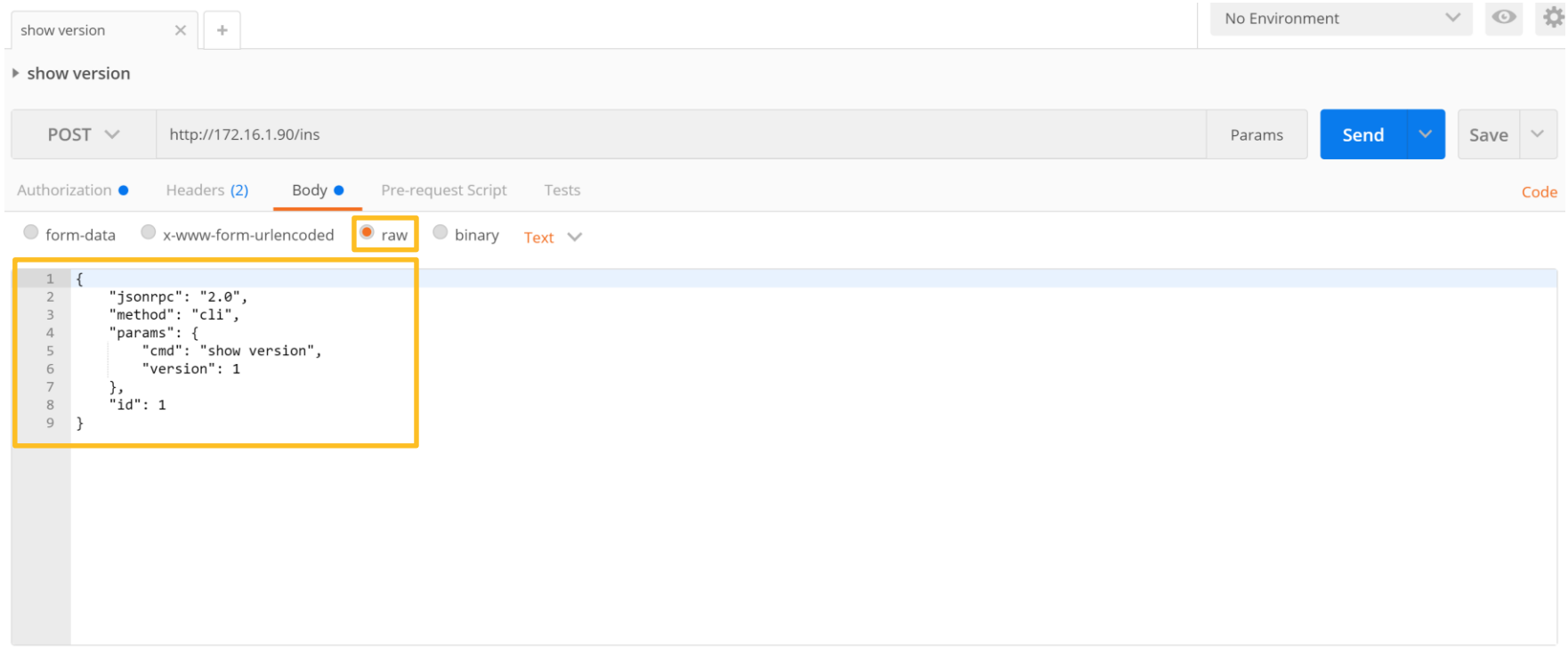
Build Request in Postman Tool (1)



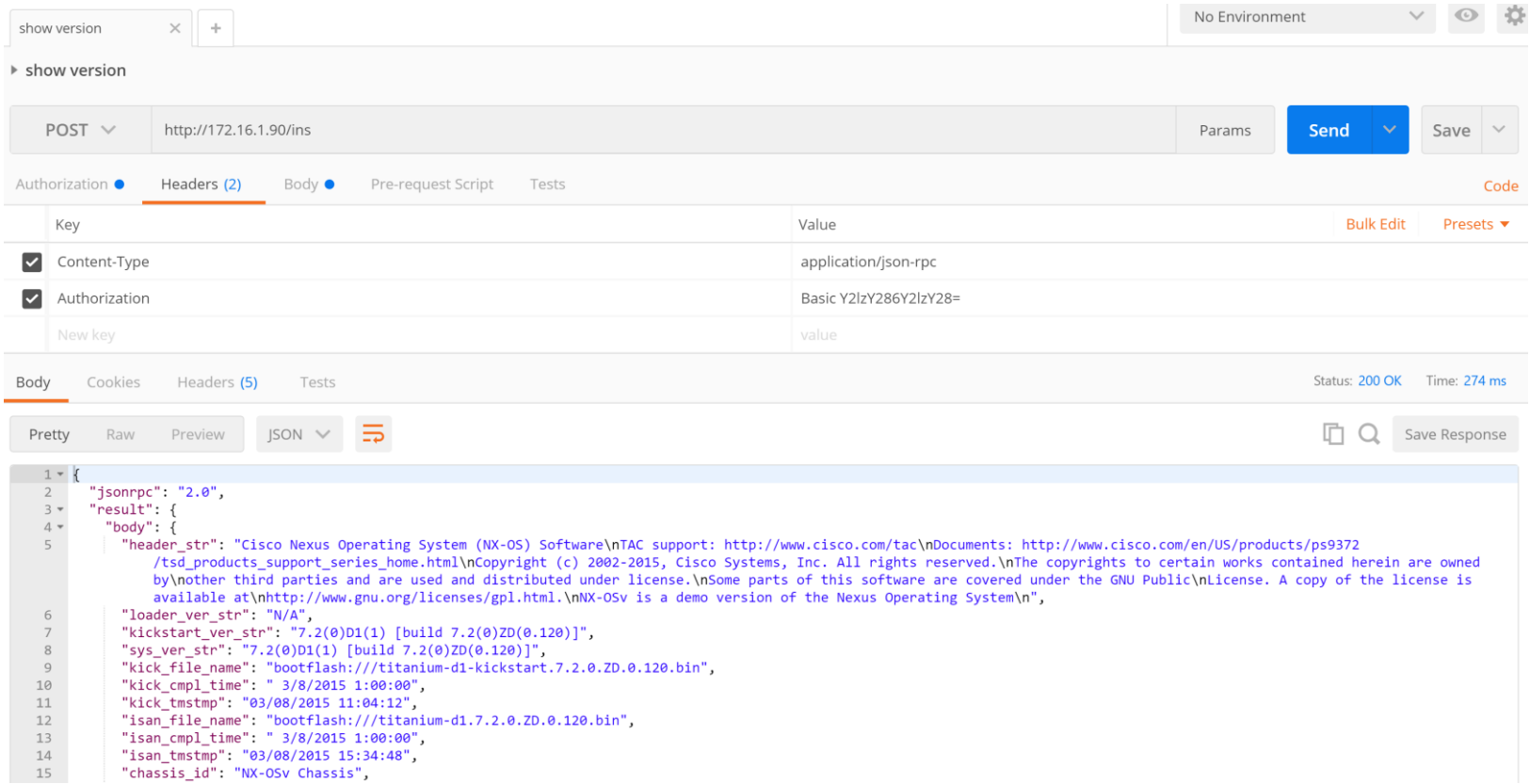
Build Request in Postman Tool (2)



Build Request in Postman Tool (3)



Execute Request in Postman Tool



“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_tmstmp": "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_tmstmp": "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")
```


CISCO