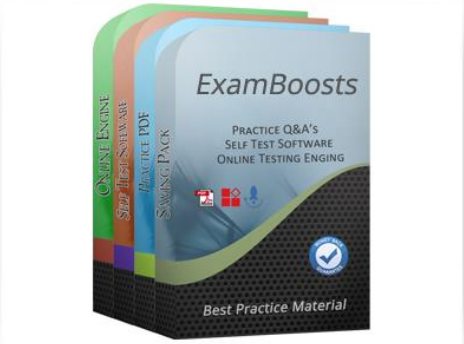
ExamBoosts

TRY BEFORE YOU BUY

Download a free sample of any of our exam questions and answers

- 24/7 customer support, Secure shopping site
- Free One year updates to match real exam scenarios
- If you failed your exam after buying our products we will refund the full amount back to you.



365 Days Free Updates

Free update is available within 365 days after your purchase. After 365 days, you will get 50% discounts for updating.

Security & Privacy

We respect customer privacy. We use McAfee's security service to provide you with utmost security for your personal information & peace of mind.

Instant Download

After Payment, our system will send you the products you purchase in mailbox in a minute after payment. If not received within 2 hours, please contact us.

Money Back Guarantee

Full refund if you fail the corresponding exam in 60 days after purchasing. And Free get any another product.

http://www.examboosts.com/

Reliable & Efficient Test Practice Questions to Satisfy All Candidates

Exam : **300-435**

Title : Automating and Programming Cisco Enterprise Solutions

Vendor : Cisco

Version : DEMO

NO.1 Refer to the exhibit. The configuration commands are entered in CLI config mode to configure a static telemetry subscription on a Cisco IOS XE device. The commands are accepted by the device, but the consumer receives no telemetry data. Which change must be made to ensure that the consumer receives the telemetry data?

```
telemetry ietf subscription 154
 encoding encode-tdl
 filter xpath /memory-ios-xe-oper:memory-statistics/memory-statistic
 source-vrf Mgmt-intf
 stream yang-push
 update-policy periodic 6000
```

A. The IP address of the receiver must be set.

B. The stream type must be set to YANG.

C. The update policy period must be shortened.

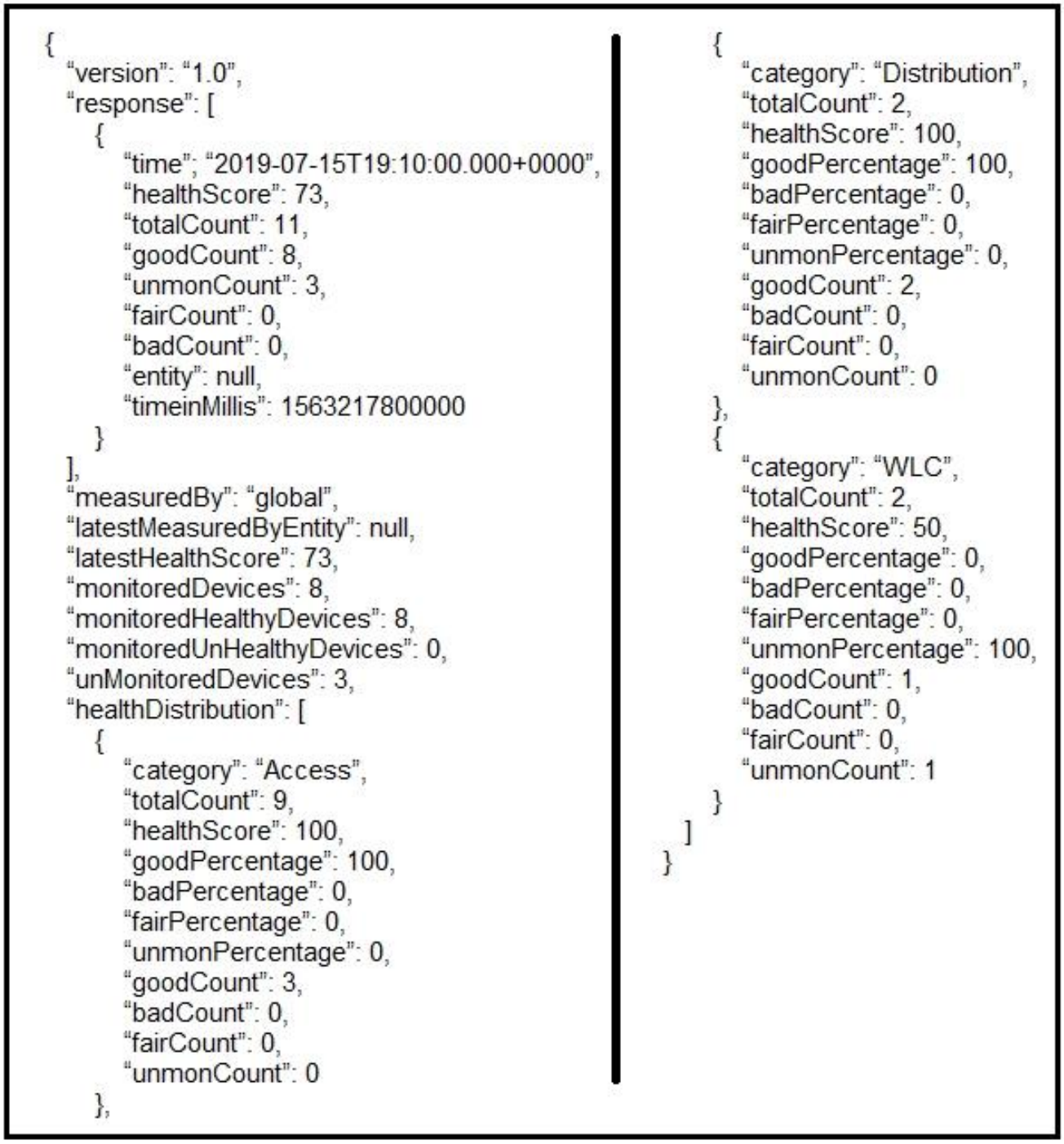
D. The sender IP address must be set.

Answer: A

Explanation:

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1610/b_1610_programmability_cg/model_driven_telemetry.html

NO.2 Refer to the exhibit. Which device type is functioning in a degraded state?



```
{
  "version": "1.0",
  "response": [
    {
      "time": "2019-07-15T19:10:00.000+0000",
      "healthScore": 73,
      "totalCount": 11,
      "goodCount": 8,
      "unmonCount": 3,
      "fairCount": 0,
      "badCount": 0,
      "entity": null,
      "timeinMillis": 1563217800000
    }
  ],
  "measuredBy": "global",
  "latestMeasuredByEntity": null,
  "latestHealthScore": 73,
  "monitoredDevices": 8,
  "monitoredHealthyDevices": 8,
  "monitoredUnHealthyDevices": 0,
  "unMonitoredDevices": 3,
  "healthDistribution": [
    {
      "category": "Access",
      "totalCount": 9,
      "healthScore": 100,
      "goodPercentage": 100,
      "badPercentage": 0,
      "fairPercentage": 0,
      "unmonPercentage": 0,
      "goodCount": 3,
      "badCount": 0,
      "fairCount": 0,
      "unmonCount": 0
    },
    {
      "category": "Distribution",
      "totalCount": 2,
      "healthScore": 100,
      "goodPercentage": 100,
      "badPercentage": 0,
      "fairPercentage": 0,
      "unmonPercentage": 0,
      "goodCount": 2,
      "badCount": 0,
      "fairCount": 0,
      "unmonCount": 0
    },
    {
      "category": "WLC",
      "totalCount": 2,
      "healthScore": 50,
      "goodPercentage": 0,
      "badPercentage": 0,
      "fairPercentage": 0,
      "unmonPercentage": 100,
      "goodCount": 1,
      "badCount": 0,
      "fairCount": 0,
      "unmonCount": 1
    }
  ]
}
```


A. access point

B. distribution switch

C. access switch

D. wireless LAN controller

Answer: D

Explanation:

The WLC (Wireless LAN Controller) category shows a healthScore of 50, with only 1 out of 2 devices marked as "good" and 1 as "unmonitored," indicating degraded health compared to other device categories with a healthScore of 100.

NO.3 Refer to the exhibit. A network engineer must create a script that provides an alert every time

a switch power supply fails in the network. To perform this task, the network engineer is using Cisco Catalyst Center (formerly DNA Center) event webhooks in a Python script. Which code snippet must be added to the box in the code to subscribe to the event?

```
payload = [
    {
        "name": "Webhook for power supply failure",
        "description": "Power supply failure on switch",
        "subscriptionEndpoints": [
            {
                "instanceId": INSTANCEID,
                "subscriptionDetails": {
                    "connectorType": "REST",
                    [                    ]
                }
            }
        ],
        "filter": {
            "eventIds": [
                "NETWORK-DEVICES-2-201"
            ]
        }
    }
]
```

A. "connectorMethod": "POST"

B. "method": "POST"

C. "subscribeTo": "POST"

D. "connector": "POST"

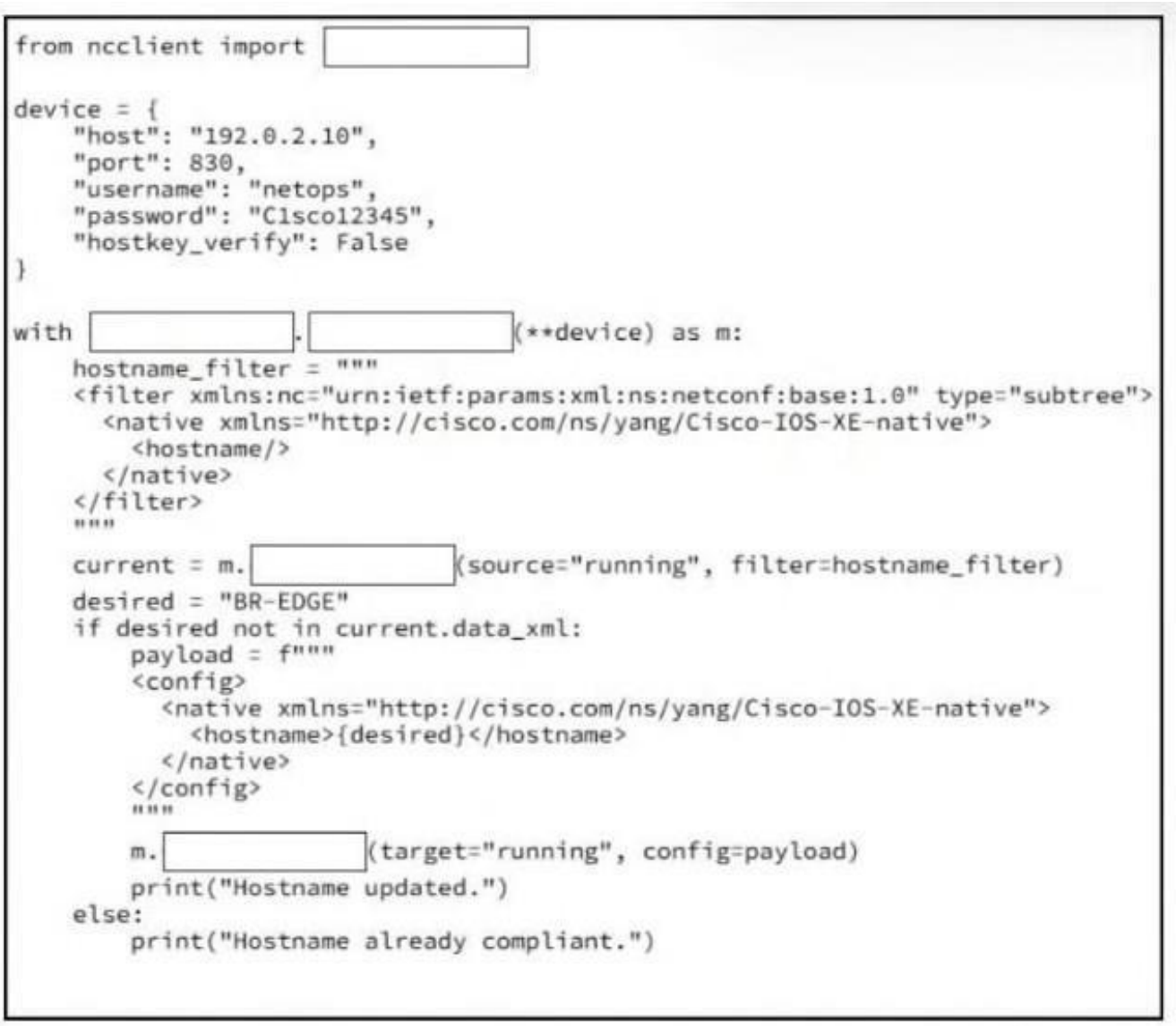
Answer: B

Explanation:

When configuring event subscription endpoints in Cisco Catalyst Center (formerly DNA Center), the correct field to specify the HTTP method for REST webhooks is: "method": "POST" This ensures that events such as power supply failures are pushed to the designated endpoint using the POST method.

NO.4 Drag and Drop Question

An engineer must enforce hostname compliance on a Cisco IOS XE router using Python and NETCONF. The engineer must read the current hostname via NETCONF and, if mismatched, push the desired hostname to the running datastore. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used. Options may be used more than once.


from ncclient import
device = {
"host": "192.0.2.10",
"port": 830,
"username": "netops",
"password": "C1sco12345",
"hostkey_verify": False
}
with . (**device) as m:
hostname_filter = """
<filter xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" type="subtree">
<native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
<hostname/>
</native>
</filter>
"""
current = m. (source="running", filter=hostname_filter)
desired = "BR-EDGE"
if desired not in current.data_xml:
payload = f"""
<config>
<native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
<hostname>{desired}</hostname>
</native>
</config>
"""
m. (target="running", config=payload)
print("Hostname updated.")
else:
print("Hostname already compliant.")
connect_ssh
get_config
merge_config
edit_config
connect
manager
get

Answer:

Explanation:
NETCONF sessions with nccclient are established through the manager module, and the connection is opened with manager.connect. To read the current hostname from the running datastore, the script uses get_config with the subtree filter. If the hostname is not compliant, the change is pushed to the running datastore with edit_config, which applies the NETCONF configuration payload.

NO.5 Refer to exhibit. A network engineer creates an Ansible playbook execution task that automates the removal of unnecessary IP addresses from the loopback interfaces of Cisco IOS XE devices.
Which code snippet must be added to the box in the code?

```
---
- name: DELETE LOOPBACK INTERFACES
  hosts: CSR1kv
  gather_facts: false
  connection: network_cli

  tasks:
  - name: DELETE UNNECESSARY IP ADDRESSES
    [          ]
      config:
        - name: Loopback10
        - name: Loopback20
        - name: Loopback30
      state: deleted
```

A. ios_l3_interfaces
B. ios_command
C. ios_config
D. ios_vrf
Answer: A
Explanation:

The correct module to use for managing Layer 3 interfaces (such as loopbacks) on Cisco IOS XE devices in an Ansible playbook is . This module allows configuration or deletion of L3 interfaces using structured data, as shown in the playbook. The state deleted ios_l3_interfaces along with a list of interface names is a valid use case for ios_l3_interfaces.

NO.6 Drag and Drop Question

An engineer must apply an initial CLI configuration to a router connected to a terminal server for Day-0 setup. The engineer must automate login and configuration over Telnet to the console port using pexpect and save the changes. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used.

```
import pexpect

child = pexpect.[          ]("telnet 192.0.2.5 2001", [          ]=20)
child.[          ]("Username:")
child.[          ]("netops")
child.expect("Password:")
child.sendline("C1sco12345")
child.expect(">")
child.sendline("enable")
child.expect("Password:")
child.sendline("C1scoEnable")
child.expect("#")
child.sendline("configure terminal")
child.expect("(config)#")
child.sendline("hostname BR1-EDGE")
child.sendline("interface GigabitEthernet1")
child.sendline("ip address dhcp")
child.sendline("no shutdown")
child.sendline("end")
child.sendline("write memory")
child.expect("#")
child.close()
```

spawnpty	spawn	wait	sendline
readline	expect	timeout	

Answer:

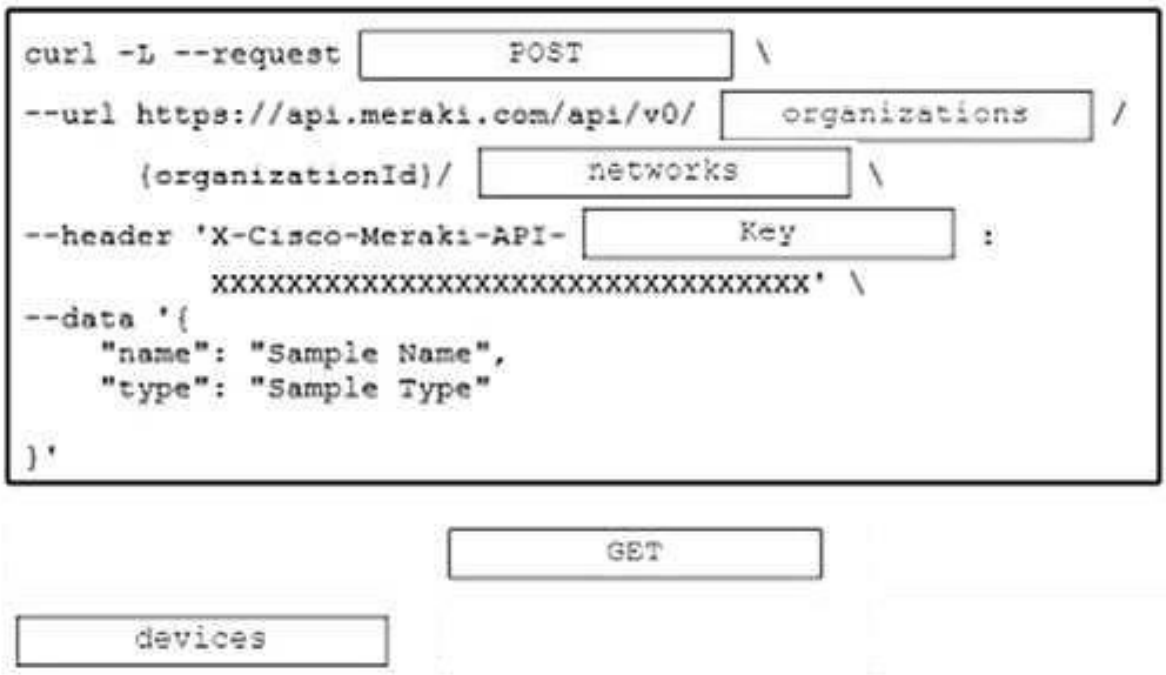
Explanation:
In pexpect, the Telnet session is started with spawn, and the session timeout is set with the timeout argument. The script then waits for the Username prompt with expect and sends the username with sendline. This enables the automated console login flow before applying the initial router configuration and saving it.

NO.7 Refer to the exhibit. A Python script has been created that calls the Cisco SD-WAN vManage Device Inventory API to get the list of vEdges. The JSON data that returns to a Python dictionary has been converted and assigned to a variable named "d". A portion of the JSON is shown in the exhibit. Which code will complete the expression hostname= to access the hostname?

A. d["data"][0]["host-name"]
B. d[data][0][host-name]
C. d("data")[0]("host-name")
D. d["host-name"]["data"]{"0"}
Answer: A
Explanation:
The double-quotations are a necessary syntax of Python. And for the json portion doesnt use parentheses. It always uses brackets. d["data"][0]["host-name"] is the only logical answer.

NO.8 Drag and Drop Question
Drag and drop the code snippets from the bottom onto the blanks in the code to create a network for a new Cisco Meraki organization. Not all option are used?

Answer:



NO.9 How do traditional versus software-defined networks compare?

A. Software-defined networks support the use of Intent APIs, and traditional networks are typically hardware-based.

B. Software-defined networks enable the use of distributed network configurations, and traditional networks are centralized.

C. Traditional networks use dynamic access lists, and software-defined networks need individual hardware to be programmed.

D. Traditional networks provide centralized network management, and software-defined networks rely on devices.

Answer: A

Explanation:

Software-defined networks (SDN) support the use of Intent APIs to enable programmability and automation, while traditional networks are typically hardware-based and managed manually.

NO.10 Drag and Drop Question

Drag and drop the code snippets from the bottom onto the boxes in the code to enable operational data, and then retrieve interface data. Not all options are used.

```
from ncclient import manager

def get_interface_state(host, port, user, passwd, filename):
    with manager.connect(host, port, user, passwd) as m:
        with open [          ] as f:
            rpc_reply = m.[          ](target='running', config=f.read())
        rpc_reply = m.[          ](filter=('subtree', "<interfaces-state/>"))
    return(rpc_reply)
```

(filen)	edit_setup	get
put	(filename)	edit_config

Answer:

```
from ncclient import manager

def get_interface_state(host, port, user, passwd, filename):
    with manager.connect(host, port, user, passwd) as m:
        with open (filename) as f:
            rpc_reply = m.edit_config (target='running', config=f.read())
        rpc_reply = m.get (filter=('subtree', "<interfaces-state/>"))
    return(rpc_reply)
```

(filen)	edit_setup
put	

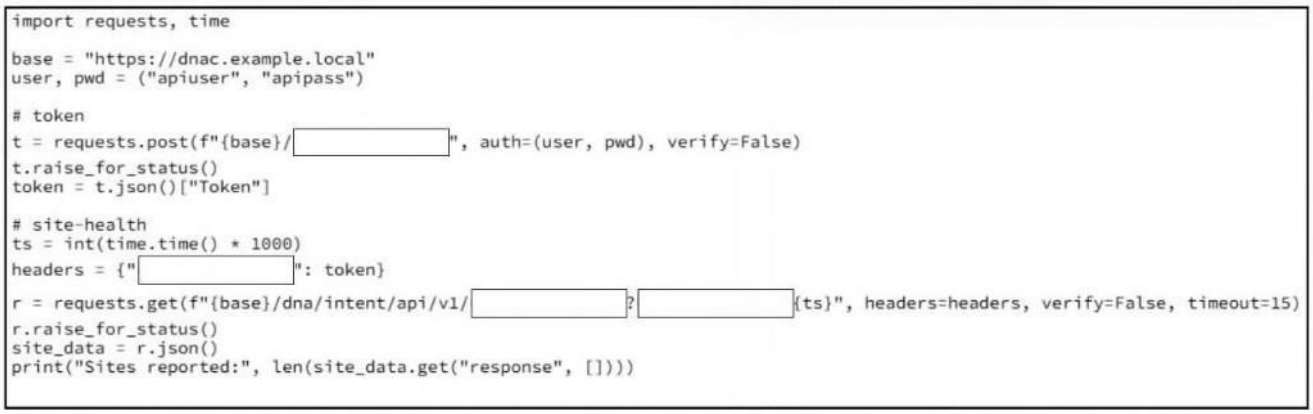
Explanation:

```
with open(filename) as f:
    rpc_reply = m.edit_config(target='running', config=f.read())
rpc_reply = m.get(filter=('subtree', "<interfaces-state/>"))
```

(filename) is used to open the file specified by the function parameter.
edit_config is the NETCONF operation for editing the running config.
get retrieves operational state data such as interface state.

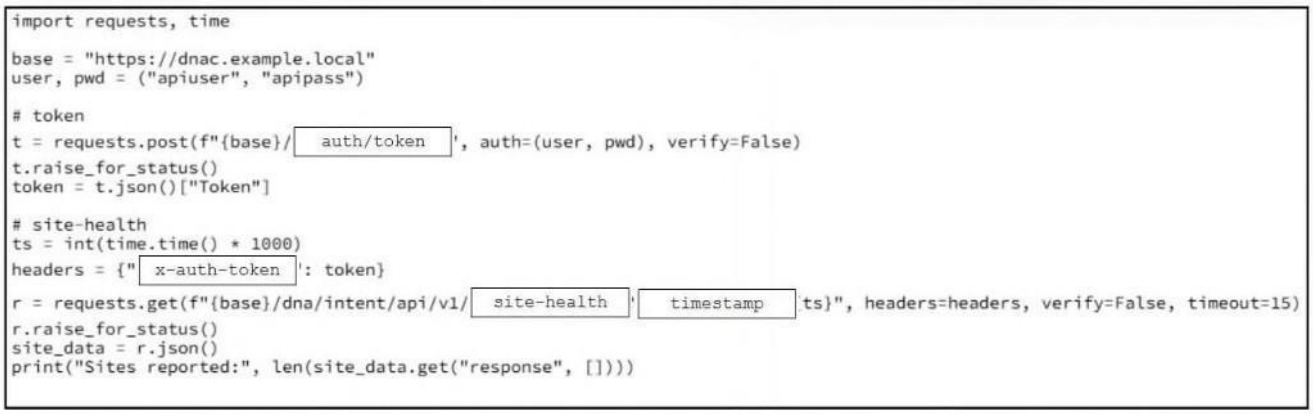
NO.11 Drag and Drop Question
An engineer must produce a site-level health snapshot from Cisco Catalyst Center for an executive dashboard. The engineer must authenticate and query the Site Health resource using the token

header and a millisecond timestamp query parameter. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used.



x-auth-token | site-health | startTime | network-health | timestamp | auth/token | Authorization

Answer:



startTime | network-health | Authorization

Explanation:
Cisco Catalyst Center authentication is performed against the auth/token endpoint to obtain the API token. Subsequent requests must pass that token in the x-auth-token header. The site-level health snapshot is retrieved from the site-health resource, and the query uses the timestamp parameter with the required millisecond value to return the health view for that point in time.

NO.12 Drag and Drop Question
Drag and drop the code snippets from the bottom onto the boxes in the code to implement a HTTP receiver in Python to handle incoming events from Cisco Meraki webhooks. Not all options are used.

```
from flask import [          ] , request, jsonify

app = Flask(__name__)

@app.route("/", methods=[" [          ] "])
def webhook():
    data = [          ].json
    print(f"{data['alertLevel'].upper()}: Alarm Type {data['alertType']}")
    return jsonify({"status": "success"})

if __name__ == "__main__":
    [          ].run(host="0.0.0.0", port=5000, threaded=True,
ssl_context=("serv.crt", "serv.key"))
```

http	request	Flask
GET	app	POST

Answer:

```
from flask import [ Flask ] , request, jsonify

app = Flask(__name__)

@app.route("/", methods=[" [ POST ] "])
def webhook():
    data = [ request ].json
    print(f"{data['alertLevel'].upper()}: Alarm Type {data['alertType']}")
    return jsonify({"status": "success"})

if __name__ == "__main__":
    [ app ].run(host="0.0.0.0", port=5000, threaded=True,
ssl_context=("serv.crt", "serv.key"))
```

http
GET

Explanation:

```python
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route("/", methods=["POST"])
def webhook():
    data = request.json
    print(f"{data['alertLevel'].upper()}: Alarm Type {data['alertType']}")
    return jsonify({"status": "success"})

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, threaded=True, ssl_context=("serv.crt", "serv.key"))
```

Flask is the class to create the app.
"POST" is the method for receiving webhook events.
request.json retrieves the incoming JSON data.
app.run(...) starts the Flask app.

NO.13 What is the role of traffic analytics in controller-based platforms?

A. packet filtering for isolation

B. archive retrieval for redundancy

C. policy enforcement for restrictions

D. flow measurement for optimization

Answer: D

Explanation:

Traffic analytics in controller-based platforms is used to measure and analyze network flows so the controller can understand usage patterns, congestion, and performance trends. That visibility supports optimization decisions such as improving path selection, balancing traffic, and tuning network behavior.

NO.14 An engineer needs to create a new network using the Meraki API. Which HTTP action to the URL
https://api.meraki.com/api/v0/organizations/<new_org_id>/networks will result in a 201 response code?

A. GET

B. POST

C. PUT

D. ADD

Answer: B

Explanation:

https://httpstatuses.com/201

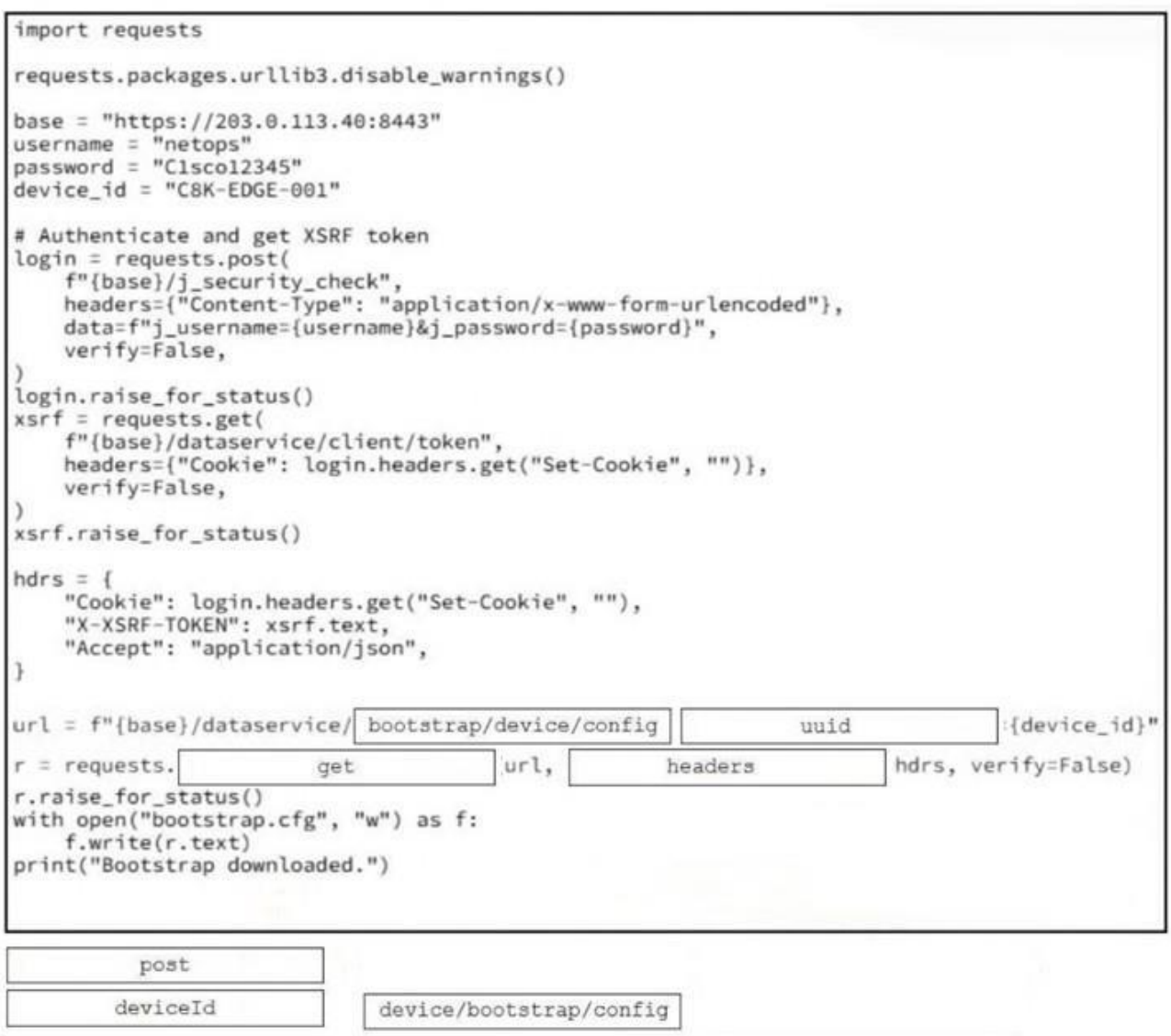
NO.15 Drag and Drop Question

An engineer must prepare SD-WAN Day 0 by exporting a WAN edge bootstrap configuration from vManage. The engineer must authenticate to vManage, obtain the XSRF token, and GET the bootstrap file for a specific device ID. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used.

```
import requests

requests.packages.urllib3.disable_warnings()

base = "https://203.0.113.40:8443"
username = "netops"
password = "C1sco12345"
device_id = "C8K-EDGE-001"

# Authenticate and get XSRF token
login = requests.post(
    f"{base}/j_security_check",
    headers={"Content-Type": "application/x-www-form-urlencoded"},
    data=f"j_username={username}&j_password={password}",
    verify=False,
)
login.raise_for_status()
xsrf = requests.get(
    f"{base}/dataservice/client/token",
    headers={"Cookie": login.headers.get("Set-Cookie", "")},
    verify=False,
)
xsrf.raise_for_status()

hdrs = {
    "Cookie": login.headers.get("Set-Cookie", ""),
    "X-XSRF-TOKEN": xsrf.text,
    "Accept": "application/json",
}

url = f"{base}/dataservice/[ ]?[ ]={device_id}"
r = requests.[ ](url, [ ]=hdrs, verify=False)
r.raise_for_status()
with open("bootstrap.cfg", "w") as f:
    f.write(r.text)
print("Bootstrap downloaded.")
```

post	bootstrap/device/config	uuid
deviceId	device/bootstrap/config	headers
get		

Answer:



```
import requests

requests.packages.urllib3.disable_warnings()

base = "https://203.0.113.40:8443"
username = "netops"
password = "C1sco12345"
device_id = "C8K-EDGE-001"

# Authenticate and get XSRF token
login = requests.post(
    f"{base}/j_security_check",
    headers={"Content-Type": "application/x-www-form-urlencoded"},
    data=f"j_username={username}&j_password={password}",
    verify=False,
)
login.raise_for_status()
xsrf = requests.get(
    f"{base}/dataservice/client/token",
    headers={"Cookie": login.headers.get("Set-Cookie", "")},
    verify=False,
)
xsrf.raise_for_status()

hdrs = {
    "Cookie": login.headers.get("Set-Cookie", ""),
    "X-XSRF-TOKEN": xsrf.text,
    "Accept": "application/json",
}

url = f"{base}/dataservice/ bootstrap/device/config   uuid   {device_id}"
r = requests.   get   url,   headers   hdrs, verify=False)
r.raise_for_status()
with open("bootstrap.cfg", "w") as f:
    f.write(r.text)
print("Bootstrap downloaded.")
```

post

deviceId

device/bootstrap/config

Explanation:
After authenticating to vManage and retrieving the XSRF token, the bootstrap configuration must be downloaded with an HTTP GET request. The request must include the session cookie and XSRF token in the headers parameter, and the target bootstrap export endpoint is built with the bootstrap configuration path plus the device identifier passed as the uuid query value.

NO.16 Refer to the exhibit. What is the expected output from the Python code?

```
# Simple Application to run a few commands on a Cisco Device
ipaddresses = ['192.168.0.1', "192.168.0.5", "10.10.10.10"]
username = "admin"
password = "cisco123"
commands_to_run=["show ver", "show ip interface brief"]
Debug = True


for device in ipaddresses:
      print ("Logging into "+device+", using "+username+"/"+password)

      # We want to execute commands on our device only if Debug=True

      for commands in commands_to_run:
            print ("    Executing "+commands+" on device: "+device)
```

A.

```
Logging into 192.168.0.1, using admin/cisco123
Logging into 192.168.0.5, using admin/cisco123
Logging into 10.10.10.10, using admin/cisco123
      Executing show ver on device: 192.168.0.1
      Executing show ip interface brief on device: 192.168.0.1
      Executing show ver on device: 192.168.0.5
      Executing show ip interface brief on device: 192.168.0.5
      Executing show ver on device: 10.10.10.10
      Executing show ip interface brief on device: 10.10.10.10
```

B.

```
Logging into 192.168.0.1, using admin/cisco123
Logging into 192.168.0.5, using admin/cisco123
Logging into 10.10.10.10, using admin/cisco123
```

C.

```
Simple Application to run a few commands on a Cisco Device
Loggig into 192.168.0.1, using admin/cisco123
We want to execute commands on our device only if Debug=True
      Executing show ver on device: 192.168.0.1
      Executing show ip interface brief on device: 192.168.0.1
Logging into 192.168.0.5, using admin/cisco123
We want to execute commands on our device only if Debug=True
      Executing show ver on device: 192.168.0.5
      Executing show ip interface brief on device: 192.168.0.5
Logging into 10.10.10.10, using admin/cisco123
We want to execute commands on our device only if Debug=True
      Executing show ver on device: 10.10.10.10
      Executing show ip interface brief on device: 10.10.10.10
```

D.

```
Logging into 192.168.0.1, using admin/cisco123
      Executing show ver on device: 192.168.0.1
      Executing show ip interface brief on device: 192.168.0.1
Logging into 192.168.0.5, using admin/cisco123
      Executing show ver on device: 192.168.0.5
      Executing show ip interface brief on device: 192.168.0.5
Logging into 10.10.10.10, using admin/cisco123
      Executing show ver on device: 10.10.10.10
      Executing show ip interface brief on device: 10.10.10.10
```

Answer: D

NO.17 FILL BLANK

Fill in the blank to complete the URL for the Cisco SD-WAN API that retrieves a list of users that are logged into a device.

http://<vmanage-ip-address>/dataservice/device/ [ ] deviceid=<deviceid>>

Answer: users?

Explanation:

https://developer.cisco.com/docs/sdwan/#!device-realtime-monitoring/users API call for real-time monitoring of users logged in to the device.

Device Users

Display the users currently logged in to the device.

GET https://{vmanage-ip-address}/dataservice/device/users?deviceId=deviceId

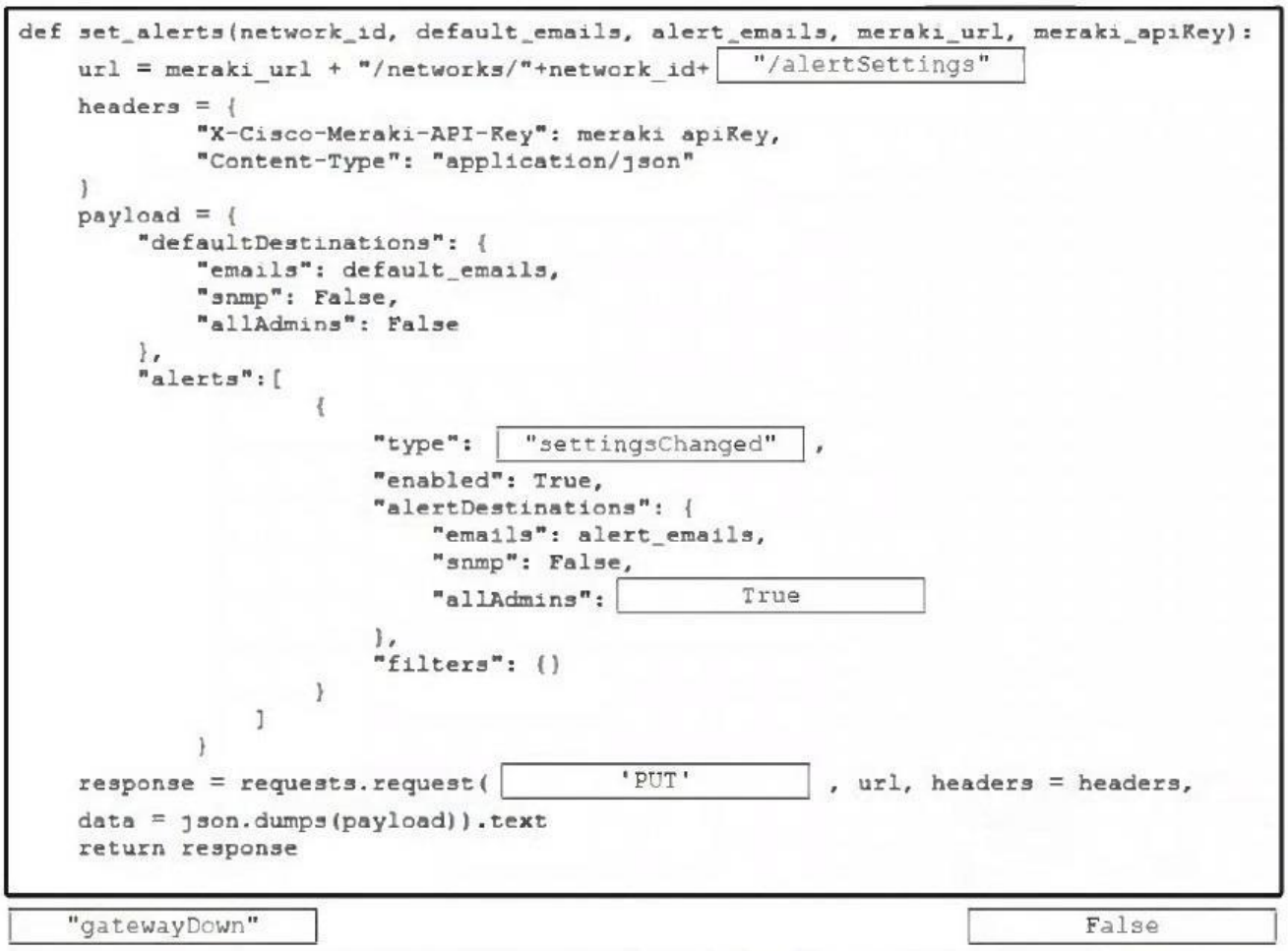
NO.18 Drag and Drop Question

Drag and drop the code snippets from the bottom onto the boxes in the code to configure the set_alerts function to send an email to all admins if a setting is modified by using the Cisco Meraki API. Not all options are used.

```
def set_alerts(network_id, default_emails, alert_emails, meraki_url, meraki_apiKey):
    url = meraki_url + "/networks/"+network_id+ [          ]
    headers = {
            "X-Cisco-Meraki-API-Key": meraki_apiKey,
            "Content-Type": "application/json"
    }
    payload = {
        "defaultDestinations": {
            "emails": default_emails,
            "snmp": False,
            "allAdmins": False
        },
        "alerts":[
                    {
                        "type": [          ],
                        "enabled": True,
                        "alertDestinations": {
                            "emails": alert_emails,
                            "snmp": False,
                            "allAdmins": [          ]
                        },
                        "filters": {}
                    }
                ]
            }
    response = requests.request( [          ] , url, headers = headers,
    data = json.dumps(payload)).text
    return response
```

"gatewayDown"	'PUT'	"/alertSettings"	False
"settingsChanged"	True		

Answer:



Explanation:
To correctly configure Meraki alert settings via the API:
The endpoint is "/alertSettings" appended to the network URL.
The alert type for configuration changes is "settingsChanged".
To notify all admins, the allAdmins field must be set to True.
The HTTP method used to modify alert settings is 'PUT'.

NO.19 Which two types of solution are built with the Meraki Location Scanning API? (Choose two.)
A. networking automation
B. mapping
C. guest Wi-Fi
D. Sense
E. wayfinder
Answer: BE
Explanation:
https://developer.cisco.com/meraki/build/wayfinding-mapwize/

NO.20 Refer to the exhibit. What is a valid XML instances of this YANG module?

```
module interfaces {

  typedef dotted-quad {
    type string {
      pattern
        '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}'
          + '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])';
    }
    description
      "Four octets written as decimal numbers and
       separated with the '.' (full stop) character.";
  }

  container interfaces {
    list interface {
      key "name";
      leaf name {
        type string;
        mandatory "true";
        description
          "Interface name.";
      }
      leaf address {
        type dotted-quad;
        mandatory "true";
        description
          "Interface IP address.";
      }
      leaf subnet-mask {
        type dotted-quad;
        mandatory "true";
        description
          "Interface subnet mask.";
      }
      leaf enabled {
        type boolean;
        default "false";
        description
          "Enable or disable the interface.";
      }
    }
  }
}
```

```
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="http://example.com/interfaces">
    <interface>
      <name>GigabitEthernet 0/0/0</name>
      <address>10.10.10.1</address>
      <subnet-mask>255.255.255.0</subnet-mask>
    </interface>
    <interface>
      <name>GigabitEthernet 0/0/1</name>
      <address>192.168.1.1</address>
      <subnet-mask>255.255.255.0</subnet-mask>
    </interface>
  </interfaces></data>
```

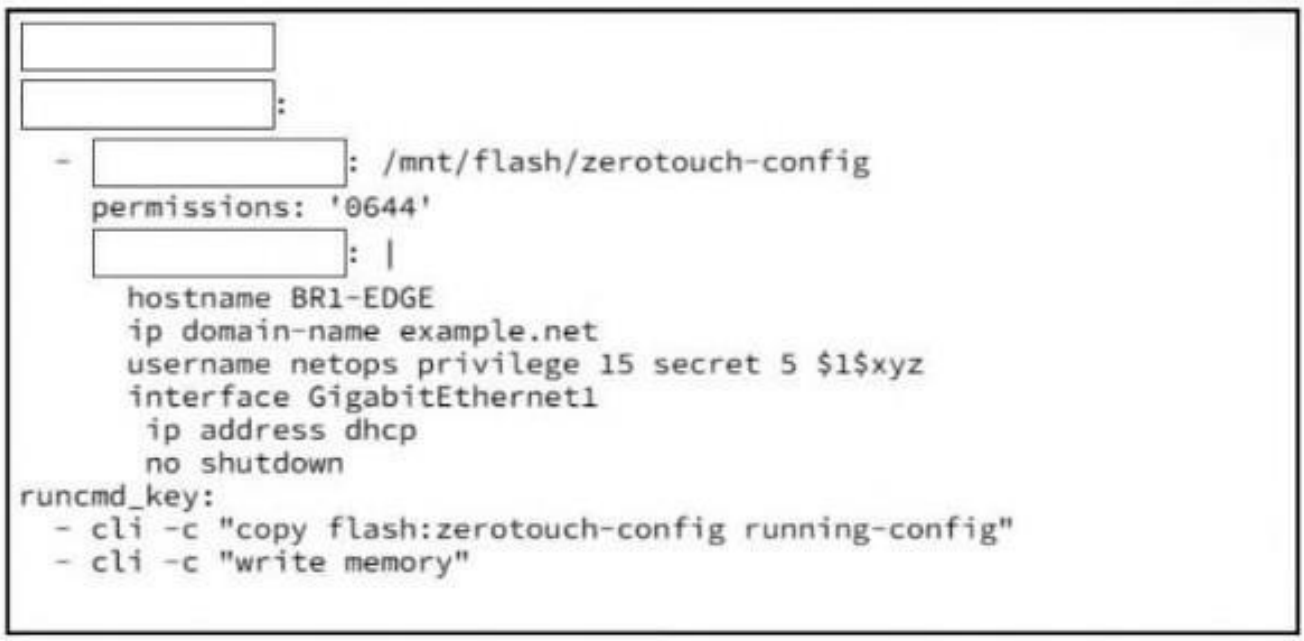
```
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="http://example.com/interfaces">
    <interface>
      <name>GigabitEthernet 0/0/0</name>
      <address>10.10.10.1</address>
      <enabled>true</enabled>
    </interface>
    <interface>
      <name>GigabitEthernet 0/0/1</name>
      <address>192.168.1.1</address>
      <subnet-mask>255.255.255.0</subnet-mask>
      <enabled>true</enabled>
    </interface></interfaces></data>
```

```
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="http://example.com/interfaces">
    <interface>
      <name>GigabitEthernet 0/0/0</name>
      <address> 2001:db8::2:1</address>
      <subnet-mask>255.255.255.0</subnet-mask>
    </interface>
    <interface>
      <name>GigabitEthernet 0/0/1</name>
      <address> 2001:db8::2:1</address>
      <subnet-mask>255.255.255.0</subnet-mask>
    </interface>
  </interfaces></data>
```

```
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="http://example.com/interfaces">
    <interface>
      <address>10.10.10.1</address>
      <subnet-mask>255.255.255.0</subnet-mask>
    </interface>
    <interface>
      <address>192.168.1.1</address>
      <subnet-mask>255.255.255.0</subnet-mask>
    </interface>
  </interfaces>
</data>
```

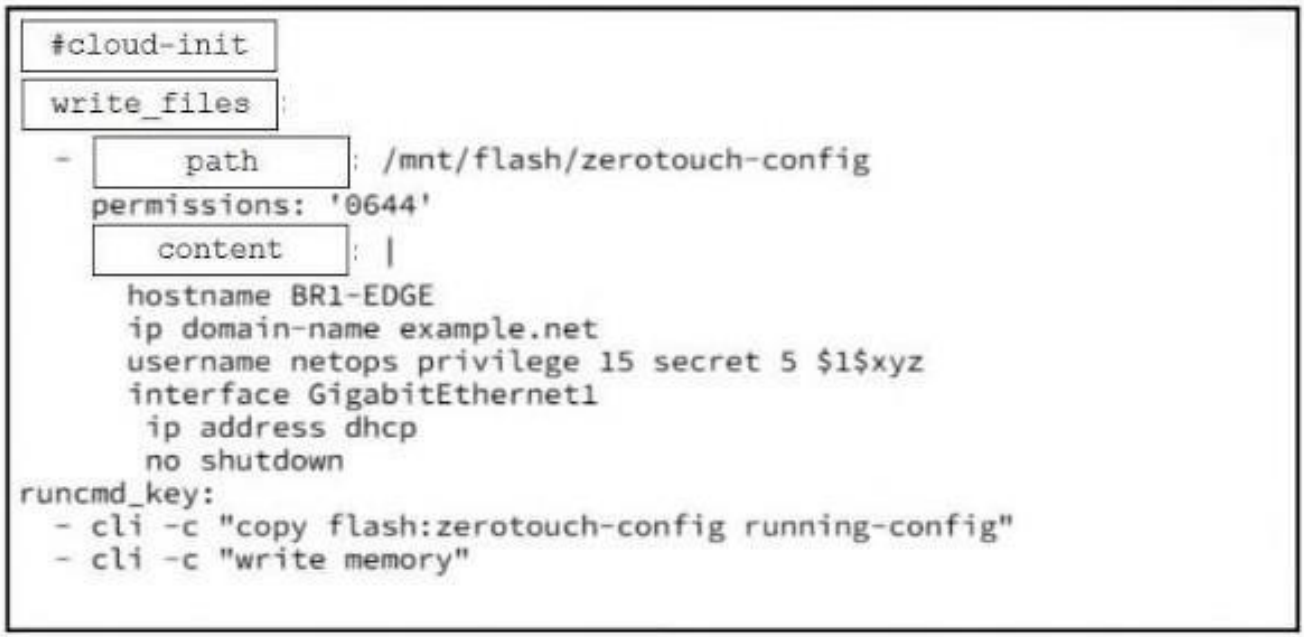
A. Option A
B. Option B
C. Option C
D. Option D
Answer: A

NO.21 Drag and Drop Question
An engineer must create a Day-0 cloud-init file for a virtual network device that supports zero- touch provisioning. The engineer must write the startup CLI to a file and execute commands that merge it into the running configuration on first boot. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used.



#cloud-init | path | #cloud-header | files_write
content | write_files | commands

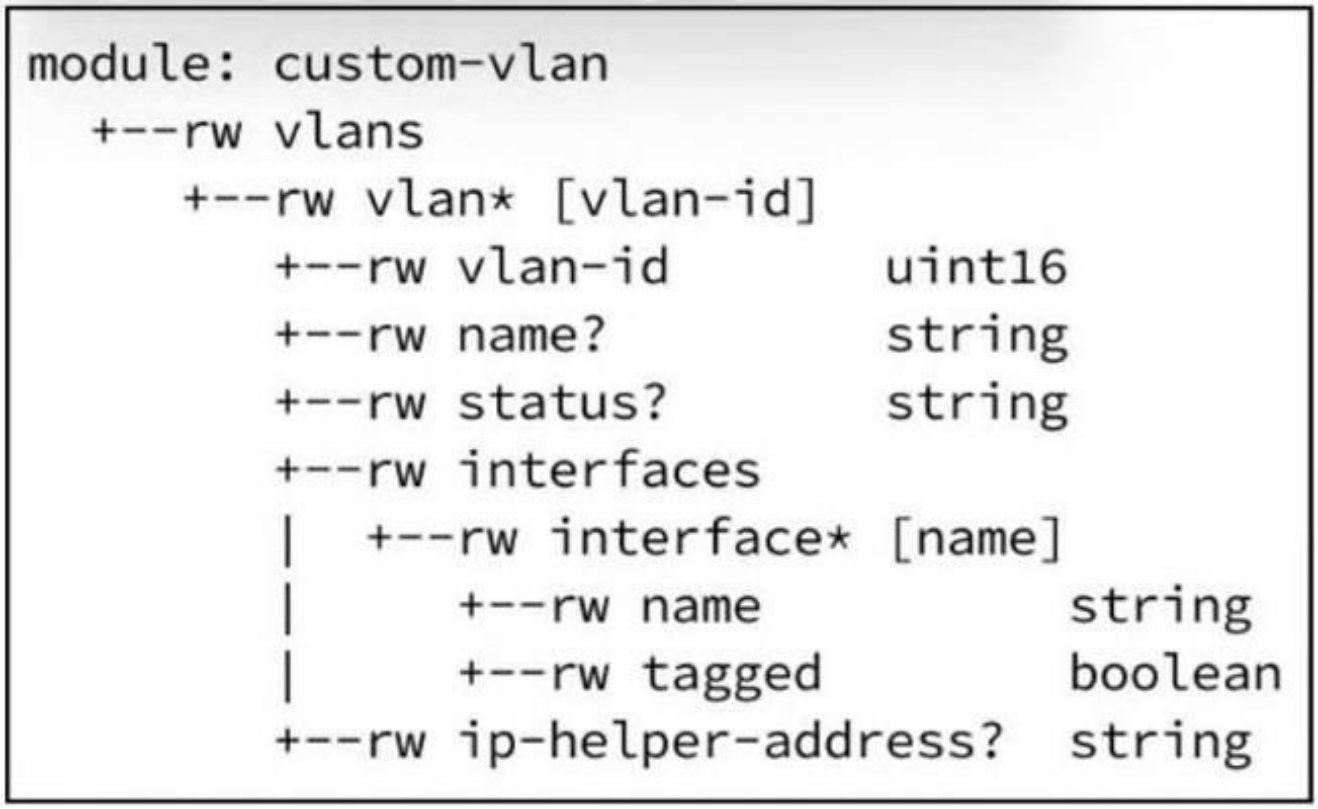
Answer:



#cloud-header | files_write
commands

Explanation:
A cloud-init Day-0 artifact must begin with the cloud-init header, then use the write_files section to place the bootstrap configuration on disk. Inside that section, path defines where the startup CLI file is written, and content provides the multiline CLI configuration that will later be merged into the running configuration by the boot-time commands.

NO.22 Refer to the exhibit. A network automation engineer is validating a VLAN YANG module tree to interpret hierarchical placement for VLAN membership and interface. The engineer wants to automate precise configuration actions in a network controller. Which node path meets these requirements?



A. vlans/vlan/interfaces/interface

B. vlans/vlan/data/interface

C. vlans/vlan/interfaces/name

D. vlans/vlan/interfaces/interface/name

Answer: A

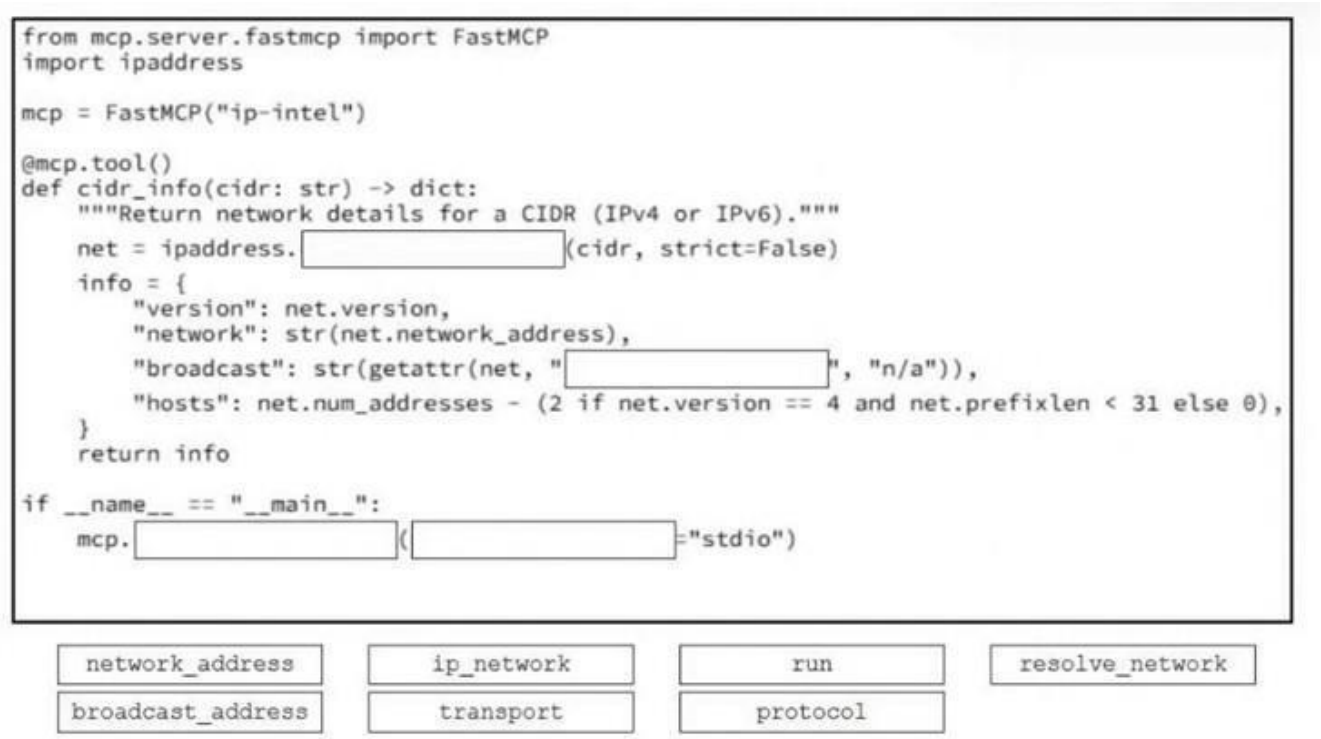
Explanation:

The YANG tree shows that VLAN membership interfaces are organized under the interfaces container within each vlan list entry, and each member is represented as an interface list item.

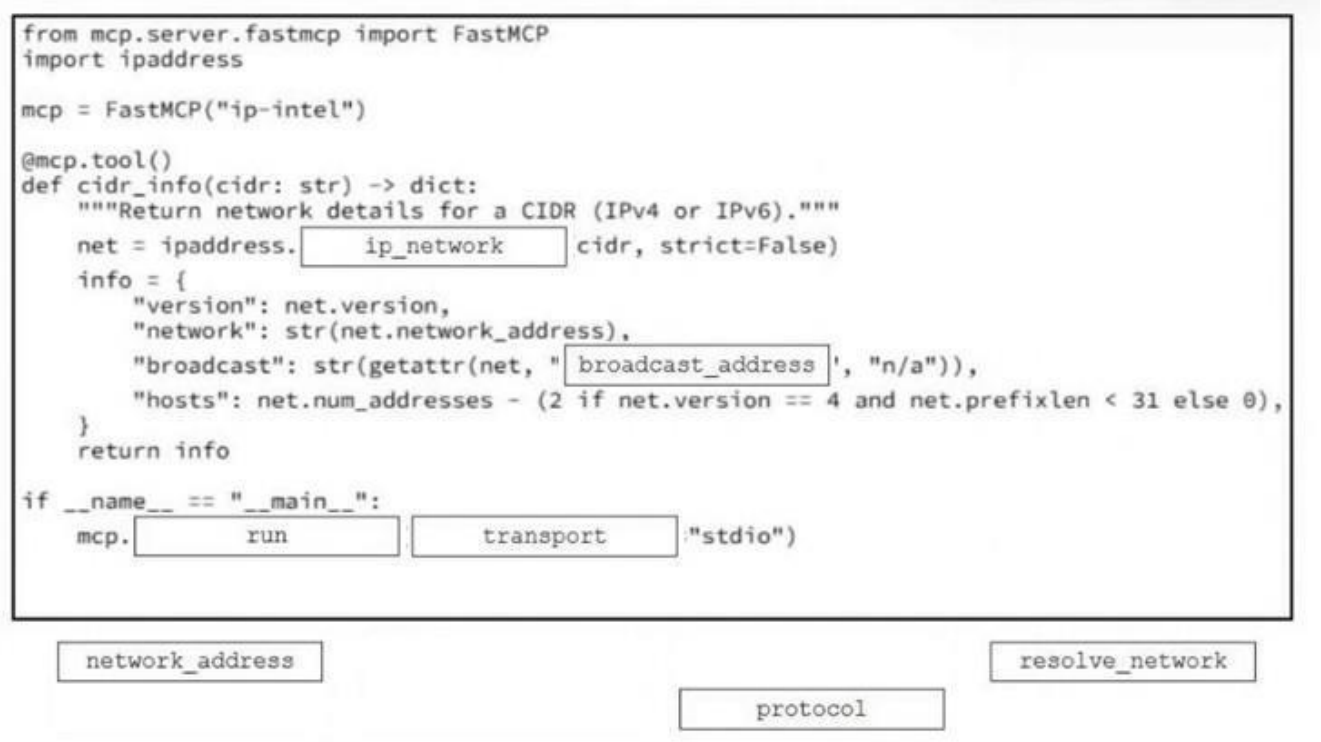
That makes vlans/vlan/interfaces/interface the correct node path for automating configuration of VLAN interface membership.

NO.23 Drag and Drop Question

An engineer must provide subnet intelligence to an AI agent so it can reason about addressing during change planning. The engineer must implement a FastMCP tool that accepts a CIDR and returns network, broadcast, and host counts, then start the server over stdio. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used.



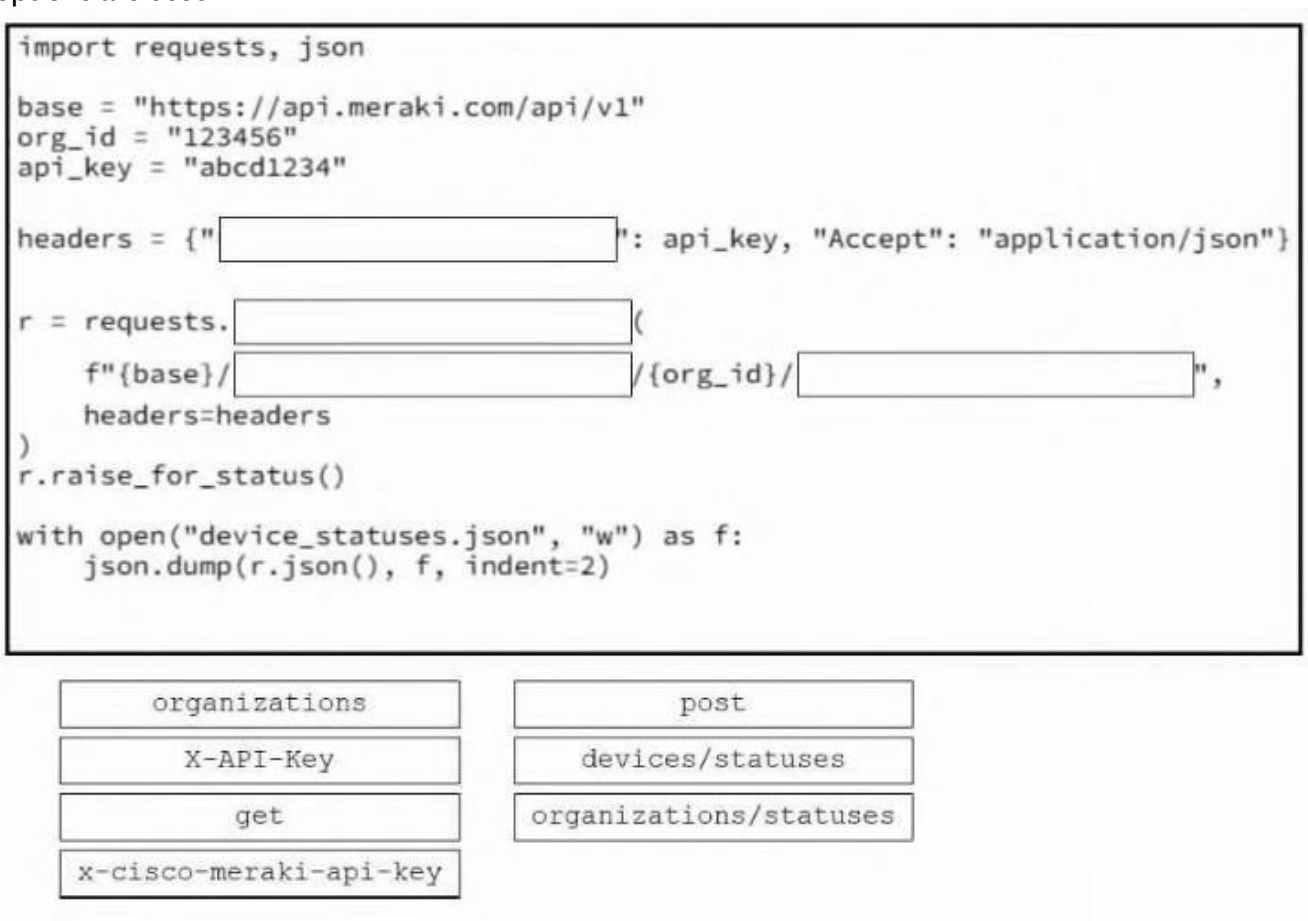
Answer:



Explanation:
The ip_network function parses the CIDR into a network object that exposes the addressing attributes needed by the tool. The broadcast value is obtained from the broadcast_address attribute, which is used when present and safely handled for IPv6 with the fallback already shown. To start the FastMCP server over standard I/O, the server is launched with run and the mode is specified with the transport argument set to stdio.

NO.24 Drag and Drop Question

An engineer must monitor device health across a Meraki organization. The engineer must retrieve device statuses via the organizations devices/statuses endpoint and archive the JSON. Drag and drop the code snippets from the bottom onto the boxes in the code to construct the artifact. Not all options are used.



Answer:

```
import requests, json

base = "https://api.meraki.com/api/v1"
org_id = "123456"
api_key = "abcd1234"

headers = {"X-API-Key": api_key, "Accept": "application/json"}

r = requests.get(
    f"{base}/organizations/{org_id}/devices/statuses",
    headers=headers
)
r.raise_for_status()

with open("device_statuses.json", "w") as f:
    json.dump(r.json(), f, indent=2)
```

post

organizations/statuses

x-cisco-meraki-api-key

Explanation:
The Meraki Dashboard API authenticates requests with the X-API-Key header. Retrieving device health data is a read operation, so the correct method is get. The required endpoint structure is the organization-level path organizations/{org_id}/devices/statuses, which returns the device status JSON that can then be archived to disk.

NO.25 Refer to the exhibit. An engineer must erase the network ENAUTO by using Cisco Meraki APIs .
Which code snippet must be added to the box in the code to complete the Python code?

```
import requests

url = "https://api.meraki.com/api/v1/networks/ENAUTO/split"

payload = None

headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
    "": "",
    "X-Cisco-Meraki-API-Key": "6bec40cf957de430a6f1f2012345678a4fac9ea0"
}

[                                                              ]

print(response.text.encode('utf8'))
```

A. response = requests.response('REMOVE', url, headers=headers, data = payload)

B. response = requests.requests('ERASE', url, headers=headers, data = payload)
C. response = requests.response('DELETE', url, headers=headers, data = payload)
D. response = requests.request('DELETE', url, headers=headers, data = payload)
Answer: D
Explanation:
To delete a Meraki network using the API, the correct HTTP method is DELETE, and the proper syntax with the requests library is:
response = requests.request('DELETE', url, headers=headers, data=payload) This line sends a DELETE request to the specified Meraki API endpoint to erase the network identified by "ENAUTO".

NO.26 Refer to the exhibit. An engineer must split the network ENAUTO by using Cisco Meraki APIs. Which code snippet must be added to the box in the code to complete the Python code?

```
import requests

url = "https://api.meraki.com/api/v1/networks/ENAUTO/split"

payload = None

headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
    "": "",
    "X-Cisco-Meraki-API-Key": "6bec40cf957de430a6f1f2012345678a4fac9ea0"
}

[                                                                    ]

print(response.text.encode('utf8'))
```

A. response = requests.request('POST', url, headers=headers, data = payload)
B. response = requests.request('PATCH', url, headers=headers, data = payload)
C. response = requests.request('PUT', url, headers=headers, payload = none)
D. response = requests.request('GET', url, headers=headers, payload = none)
Answer: A
Explanation:
To perform a network split using the Cisco Meraki API, the correct method is a POST request.
The code must use:
response = requests.request('POST', url, headers=headers, data=payload) This correctly sends the API request with the appropriate headers and payload (even if None).