Cisco

(300-435)

Automating Cisco Enterprise Solutions (ENAUTO)

Total: **149 Questions**
Link:

Question: 1

What are two characteristics of RPC API calls? (Choose two.)

A.They can be used only on network devices.
B.They use only UDP for communications.
C.Parameters can be passed to the calls.
D.They must use SSL/TLS.
E.They call a single function or service.

Answer: CE

Explanation:

The correct answer is **C. Parameters can be passed to the calls** and **E. They call a single function or service**. RPC (Remote Procedure Call) APIs are designed to execute functions or procedures on a remote server as if they were local. This mechanism inherently requires the ability to pass parameters or arguments to the remote function. This allows the client to provide input to the function, influencing its behavior and the results it generates. Additionally, RPC calls are typically designed to invoke a specific, well-defined procedure or service on the remote server. Each call is intended to perform a specific task, rather than a broad, multi-purpose interaction. This promotes modularity and simplifies the communication process. RPC is a procedural paradigm that emphasizes the specific nature of the call and its purpose.

Option A is incorrect because RPC can be used across many systems, not just network devices. Option B is wrong because RPC can use various protocols for communications, like TCP, not solely UDP. Also, while RPCs can use SSL/TLS, using them is not a must and is optional hence option D is incorrect.

Remote Procedure Call (RPC):https://en.wikipedia.org/wiki/Remote_procedure_call
Introduction to REST APIs:https://www.redhat.com/en/topics/api/what-are-rest-apis (While not directly RPC, it contrasts with it, highlighting RPC's single-function focus).

Question: 2

Which two actions do Python virtual environments allow users to perform? (Choose two.)

A.Simplify the CI/CD pipeline when checking a project into a version control system, such as Git.
B.Efficiently port code between different languages, such as JavaScript and Python.
C.Run and simulate other operating systems within a development environment.
D.Quickly create any Python environment for testing and debugging purposes.
E.Quickly create an isolated Python environment with module dependencies.

Answer: DE

Explanation:

The correct answers are **D and E.** Python virtual environments are fundamentally designed to address dependency management and isolation within Python development. They achieve this by creating self-contained directories where specific versions of Python and its packages can be installed, independent of the system-wide Python installation or other virtual environments.

Option **D** is correct because virtual environments facilitate rapid creation of dedicated, configurable Python environments. This allows developers to experiment with different package combinations and versions without altering the global Python installation or impacting other projects. This capability is crucial for testing and debugging, as developers can easily recreate specific scenarios with exact package specifications.

Option **E** is also correct. A core function of virtual environments is to isolate dependencies for projects. By installing packages within a virtual environment, you ensure that a project's required libraries are separated from other projects, avoiding conflicts between different version requirements. This is crucial to ensure project portability and reproducible results.

Option **A** is incorrect because CI/CD pipelines might utilize virtual environments, but the virtual environments themselves do not simplify the check-in process itself. While virtual environments ensure consistency during testing phases within CI/CD, that's different from the act of checking in code.Option **B** is incorrect; virtual environments are specific to Python and do not assist in porting code to different languages. Language conversion is a complex process requiring other specialized tools.Option **C** is incorrect as virtual environments do not simulate operating systems. Tools like Docker or virtual machines handle that kind of isolation, not Python virtual environments.

In essence, virtual environments are focused on providing localized and controlled Python environments for development, enabling efficient dependency management and reproducible projects.

Further reading:

Python venv documentation: https://docs.python.org/3/library/venv.html
Python virtual environments guide: https://realpython.com/python-virtual-environments-primer/

Question: 3

What are two benefits of leveraging Ansible for automation of Cisco IOS XE Software? (Choose two.)

A.Ansible playbooks are packaged and installed on IOS XE devices for automatic execution when an IOS device reboots.
B.All IOS XE operating systems include Ansible playbooks for basic system administration tasks.
C.It is a device-independent method for automation and can be used with any type of device or operating system.
D.Ansible playbooks can be written from the IOS XE EXEC command line to configure the device itself.
E.It does not require any modules of software except SSH to be loaded on the network device.

Answer: CE

Explanation:

The correct answers are C and E.

Let's break down why:

C. It is a device-independent method for automation and can be used with any type of device or operating system. Ansible is designed to be platform-agnostic. While specific modules cater to different operating systems and devices, the core logic of using playbooks written in YAML remains consistent. This allows network engineers to leverage the same automation framework across diverse environments, including Cisco IOS XE, Linux servers, and other network devices. This eliminates the need to learn multiple automation languages, streamlining processes and promoting consistency.
https://docs.ansible.com/ansible/latest/getting_started/index.html

E. It does not require any modules of software except SSH to be loaded on the network device. Ansible operates on a push-based model, where the control node executes instructions on managed nodes. Typically, it uses SSH to communicate and push configuration changes. No agents are required on the target Cisco IOS XE devices, simplifying deployment and reducing resource consumption on these devices. This is a key advantage over agent-based solutions which can add overhead and complexity to manage.
https://docs.ansible.com/ansible/latest/network/getting_started/network_getting_started.html

Now let's address why the other options are incorrect:

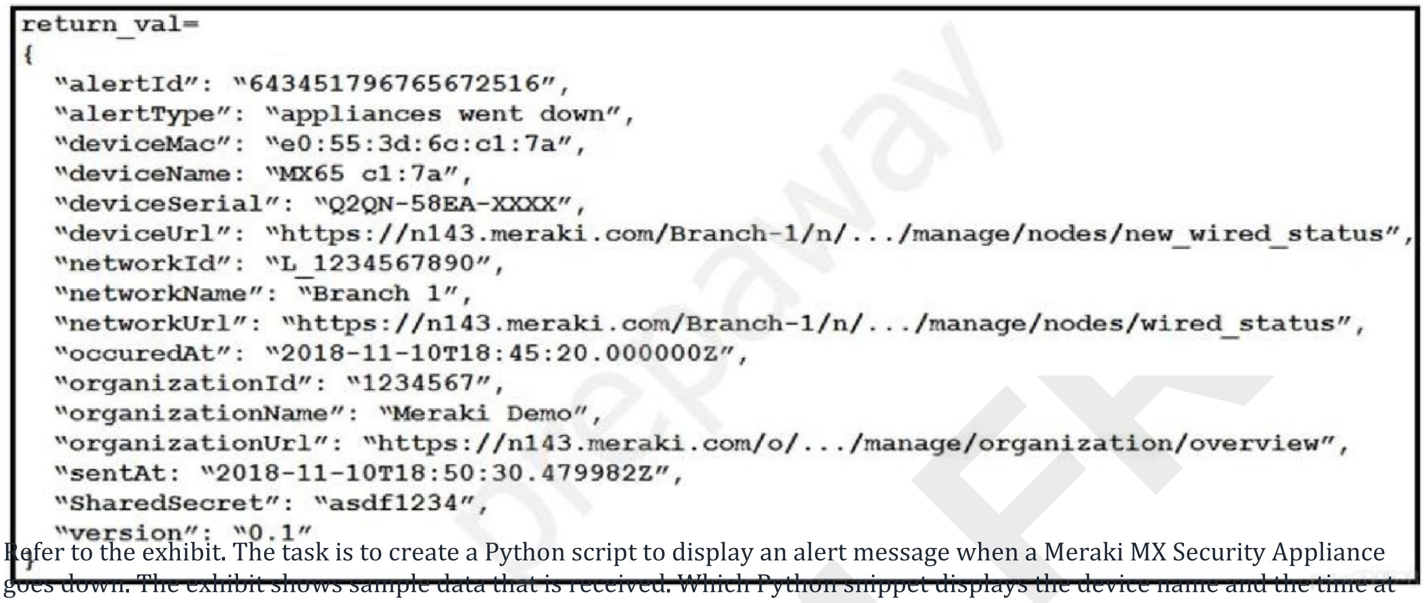
A. Ansible playbooks are packaged and installed on IOS XE devices for automatic execution when an IOS device reboots. Ansible playbooks are executed from a central control node and are not installed or executed directly on the IOS XE devices. They are not triggered by device reboots.

B. All IOS XE operating systems include Ansible playbooks for basic system administration tasks. IOS XE doesn't inherently include pre-built Ansible playbooks. Users need to create or obtain them separately.

D. Ansible playbooks can be written from the IOS XE EXEC command line to configure the device itself. Ansible playbooks are created on a machine where Ansible is installed and typically aren't written directly on the IOS XE device's EXEC command line.

In conclusion, Ansible's device-independent nature and agentless architecture provide compelling advantages for automating Cisco IOS XE, making options C and E the correct choices.

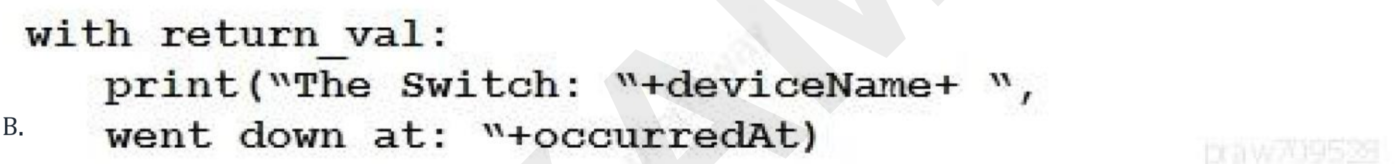
Question: 4



```
return_val=
{
  "alertId": "643451796765672516",
  "alertType": "appliances went down",
  "deviceMac": "e0:55:3d:6c:c1:7a",
  "deviceName: "MX65 c1:7a",
  "deviceSerial": "Q2QN-58EA-XXXX",
  "deviceUrl": "https://n143.meraki.com/Branch-1/n/.../manage/nodes/new_wired_status",
  "networkId": "L_1234567890",
  "networkName": "Branch 1",
  "networkUrl": "https://n143.meraki.com/Branch-1/n/.../manage/nodes/wired_status",
  "occuredAt": "2018-11-10T18:45:20.000000Z",
  "organizationId": "1234567",
  "organizationName": "Meraki Demo",
  "organizationUrl": "https://n143.meraki.com/o/.../manage/organization/overview",
  "sentAt: "2018-11-10T18:50:30.479982Z",
  "SharedSecret": "asdf1234",
  "version": "0.1"
}
```

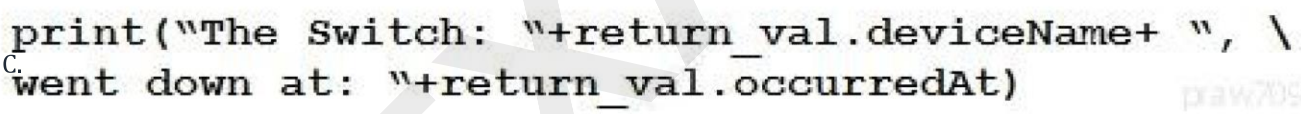
Refer to the exhibit. The task is to create a Python script to display an alert message when a Meraki MX Security Appliance goes down. The exhibit shows sample data that is received. Which Python snippet displays the device name and the time at which the switch went down?

A.



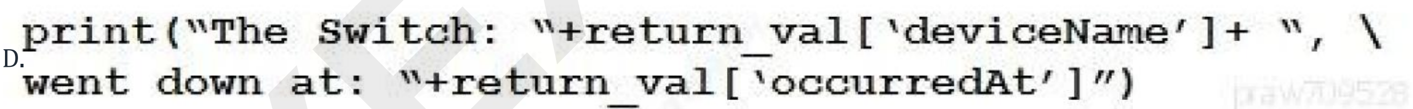
```
with return_val:
    print("The Switch: "+deviceName+ ",
    went down at: "+occurredAt)
```

B.



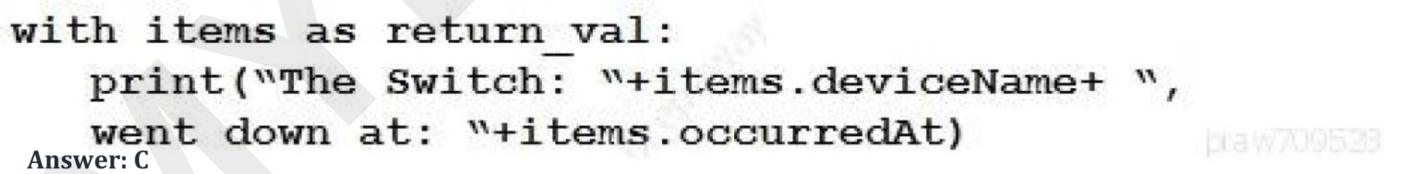
```
print("The Switch: "+return_val.deviceName+ ", \
went down at: "+return_val.occurredAt)
```

C.



```
print("The Switch: "+return_val['deviceName']+ ", \
went down at: "+return_val['occurredAt']")
```

D.



```
with items as return_val:
   print("The Switch: "+items.deviceName+ ",
   went down at: "+items.occurredAt)
```

Answer: C

Explanation:

```
print(“The Switch: “+return_val[‘deviceName’]+ “, \
went down at: “+return_val[‘occurredAt’]”)
```

Question: 5

```
{
  “alertData”: {
    “countNode”: 1,
      “bssids”: [
       “aa:bb:cc:dd:ee:ff”,
       “11:22:33:44:55:66”
      ],
      “minFirstSeen”: 1548512334,
      “maxLastSeen”: 1548512802,
      “countIsContained”: 0,
      “reason”: “Seen on LAN”,
      “wiredMac”: “aa:bb:cc:dd:ee:f0”
  },
  “alertId”: “629378047939282802”,
  “alertType”: “Air Marshal -Roque AP detected”,
  “occuredAt”: “2019-01-26T14:18:54.000000Z”,
  “organizationId”: “123456”,
  “organizationName”: “Organization”,
  “organizationUrl”: “https://n1.meraki.com/o/.../manage/organization/overview”,
  “networkId”: “L_123456789012345678”,
  “networkName”: “Network”,
  “networkUrl”: “https://n1.meraki.com/.../manage/nodes/list”,
  “version”: “0.1”
  “SharedSecret”: “supersecret”,
  “sentAt: “2019-01-26T14:35:20.442869Z”,
}
```

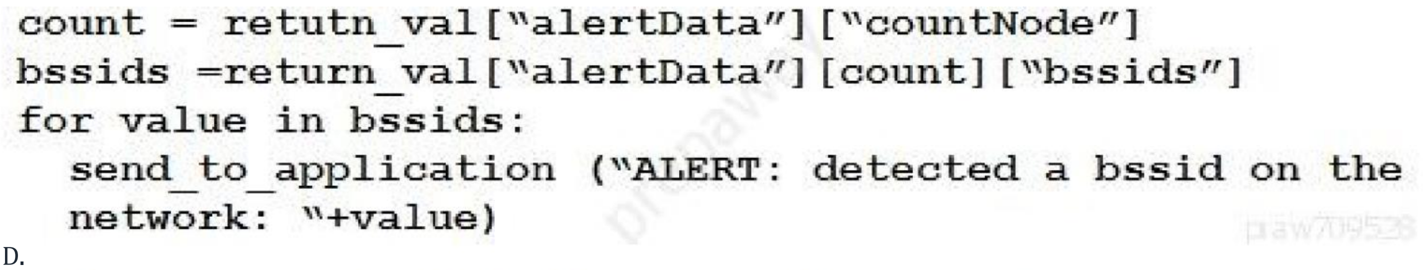
Refer to the exhibit. The goal is to write a Python script to automatically send a message to an external messaging application when a rogue AP is detected on the network. The message should include the broadcast SSID that is in the alert.
A function called `send_to_application` is created, and this is the declaration:
send_to_application(message)
The exhibit also shows the data that is received by the application and stored in the variable return_val. Which Python code completes the task?
A.

```
bssids =return_val[“bssids”]
for number in range(return_val[“alertData”][“countNode”]):
  send_to_application (“ALERT: detected a bssid on the
  network: “+ return_val[“alertData”][bssids][number])
```

B.

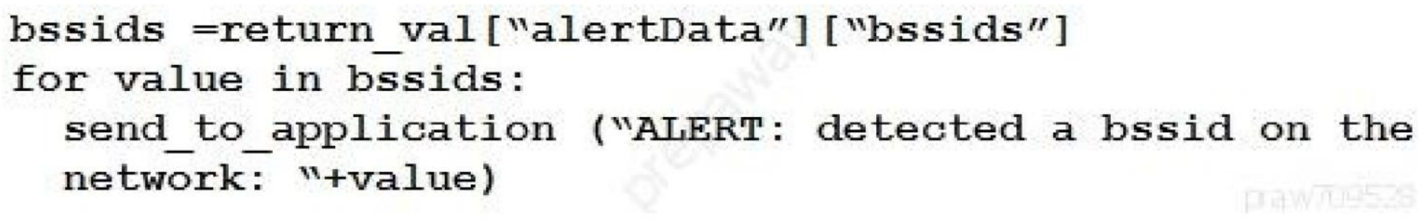
```
bssids =return_val[“bssids”]
for value in bssids:
  send_to_application (“ALERT: detected a bssid on the
  network: “+value)
```

C.



```
count = retutn_val["alertData"]["countNode"]
bssids =return_val["alertData"][count]["bssids"]
for value in bssids:
  send_to_application ("ALERT: detected a bssid on the
  network: "+value)
```

D.



```
bssids =return_val["alertData"]["bssids"]
for value in bssids:
  send_to_application ("ALERT: detected a bssid on the
  network: "+value)
```

Answer: A

Explanation:
Explanation -
For number in range value is required for the application to send the alert. Bssids are also included.

Question: 6

Which two features are foundations of a software-defined network instead of a traditional network? (Choose two.)

A.control plane and data plane are tightly coupled
B.build upon a robust software stack
C.requires device by device-level configurations
D.automated through expressed intent to a software controller
E.requires significant physical hardware resources

Answer: BD

Explanation:

Here's a justification for why options B and D are features of software-defined networks (SDNs), while A, C, and E are not:

Software-defined networking fundamentally shifts network control away from individual hardware devices to a centralized software controller. **Option B, "build upon a robust software stack,"** is crucial because SDN relies on software for all control and management functions. This software stack provides the abstraction layer necessary for network automation, orchestration, and programmability. Key components include the SDN controller, APIs for communication, and applications for network services.

Option D, "automated through expressed intent to a software controller," highlights the declarative nature of SDN. Instead of configuring individual devices, administrators define the desired network behavior, and the controller automatically implements the necessary configurations. This intent-based approach simplifies management and allows for rapid network changes, moving away from traditional imperative, device-centric workflows.

In contrast, **option A, "control plane and data plane are tightly coupled,"** describes the traditional networking model. SDNs decouple these planes, allowing for centralized control (control plane) and distributed packet forwarding (data plane). **Option C, "requires device by device-level configurations,"** is a characteristic of traditional networking, not SDN, which abstracts away device-specific configurations. Finally, **option E, "requires significant physical hardware resources,"** doesn't fundamentally differentiate SDN; while physical resources are always needed, SDN emphasizes the abstract management and efficient utilization of those resources by software. The core value of SDN comes from the software, not an increase in physical hardware.

In summary, SDNs embrace programmability and automation through a robust software stack and centralized intent-based management, differentiating them from traditional networking models.

Authoritative links:

Open Networking Foundation (ONF):https://opennetworking.org/ **-** A leading organization for SDN standards and development.

SDX Central:https://www.sdxcentral.com/ **-** A comprehensive resource for information on SDNs and related technologies.

Cisco DevNet:https://developer.cisco.com/ **-** Cisco's developer portal, often containing information on automation and programmability with a focus on the network.

Question: 7

A new project called `device_status` must be stored in a central Git repository called `device_status` with the first file named `device_status.py`. The Git repository is created using the account python_programmer. Which set of commands inserts the project into Git?

A.

```
git init
git add device status.py
git commit -m "Initial Revision"
git remote add origin \
     https://git.cisco.com/python_programmer/device_status.git
git push -u origin master
```

B.

```
git init
git remote add origin \
     https://git.cisco.com/python_programmer/device_status.git
git add device_status.py
git pull
```

C.

```
git init
git remote add origin \
     https://git.cisco.com/python_programmer/device_status.git
git add device_status.py
git commit -m "Initial Revision"
git pull -u origin master
```

D.

```
git init
git add device_status.py
git remote add python_programmer/device_status
git push
```

Answer: A

Explanation:

Reference:
https://help.github.com/en/github/importing-your-projects-to-github/adding-an-existing-project-to-github-us

ing-the-command-line

Question: 8

What are two characteristics of synchronous calls to APIs? (Choose two.)

A.They can be used only with certain programming languages.
B.They make your application less portable, so asynchronous calls are preferred.
C.They can add perceived latency to your application if data is not received.
D.They block until a response is returned from the servers.
E.They do not block while waiting for the API to be processed.

Answer: CD

Explanation:

Okay, here's a detailed justification for why options C and D are the correct characteristics of synchronous API calls, along with supporting concepts and links:

Synchronous API calls operate in a request-response pattern where the client making the call waits for a response from the server before proceeding with its next task. Option D, "They block until a response is returned from the servers," accurately describes this behavior. The calling application's thread is essentially paused, or blocked, until the API call completes and a response is received. This blocking nature is a fundamental characteristic of synchronous operations.

Option C, "They can add perceived latency to your application if data is not received," is also correct due to this blocking behavior. Since the application is waiting, any delays in the server processing the request or transmitting the response are directly translated into perceived latency for the user. For example, if a user clicks a button that triggers a synchronous API call and that call takes a few seconds, the application will appear frozen or slow until the API call completes. This perceived delay is often undesirable in user-facing applications.

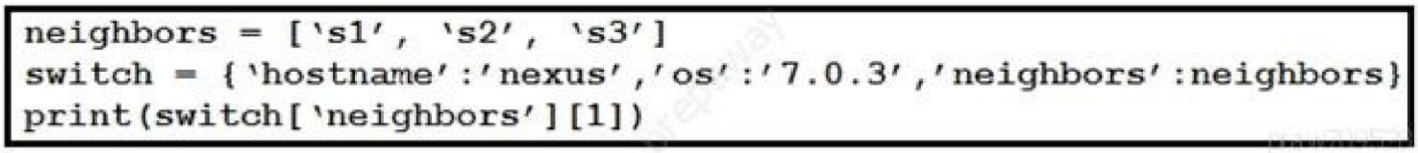
Option A is incorrect. Synchronous API calls are not restricted to specific programming languages. Most languages support synchronous communication patterns. Option B is misleading; while asynchronous calls can be preferred in certain scenarios to improve responsiveness, synchronous calls aren't inherently less portable. Synchronous and asynchronous have use cases, therefore it's not a blanket rule to prefer one over the other. Finally, option E is incorrect because, by definition, synchronous calls do block while waiting for a response.

In summary, synchronous API calls are characterized by their blocking nature and the resulting potential for perceived latency, while they are not dependent on language or portability. Their simple request-response pattern, though straightforward, makes them unsuitable for latency-sensitive situations.

Authoritative Links for Further Research:

1. **Understanding Synchronous and Asynchronous Calls:**https://www.geeksforgeeks.org/difference-between-synchronous-and-asynchronous-calls/ - This article provides a clear explanation of the difference between synchronous and asynchronous operations with examples.
2. **API Design Patterns**: https://cloud.google.com/apis/design - This link by Google covers API design and includes a discussion on synchronous and asynchronous patterns.
3. **Synchronous vs Asynchronous Operations**:https://www.baeldung.com/cs/synchronous-vs-asynchronous - This article provides an in-depth discussion of both models with programming examples.

Question: 9



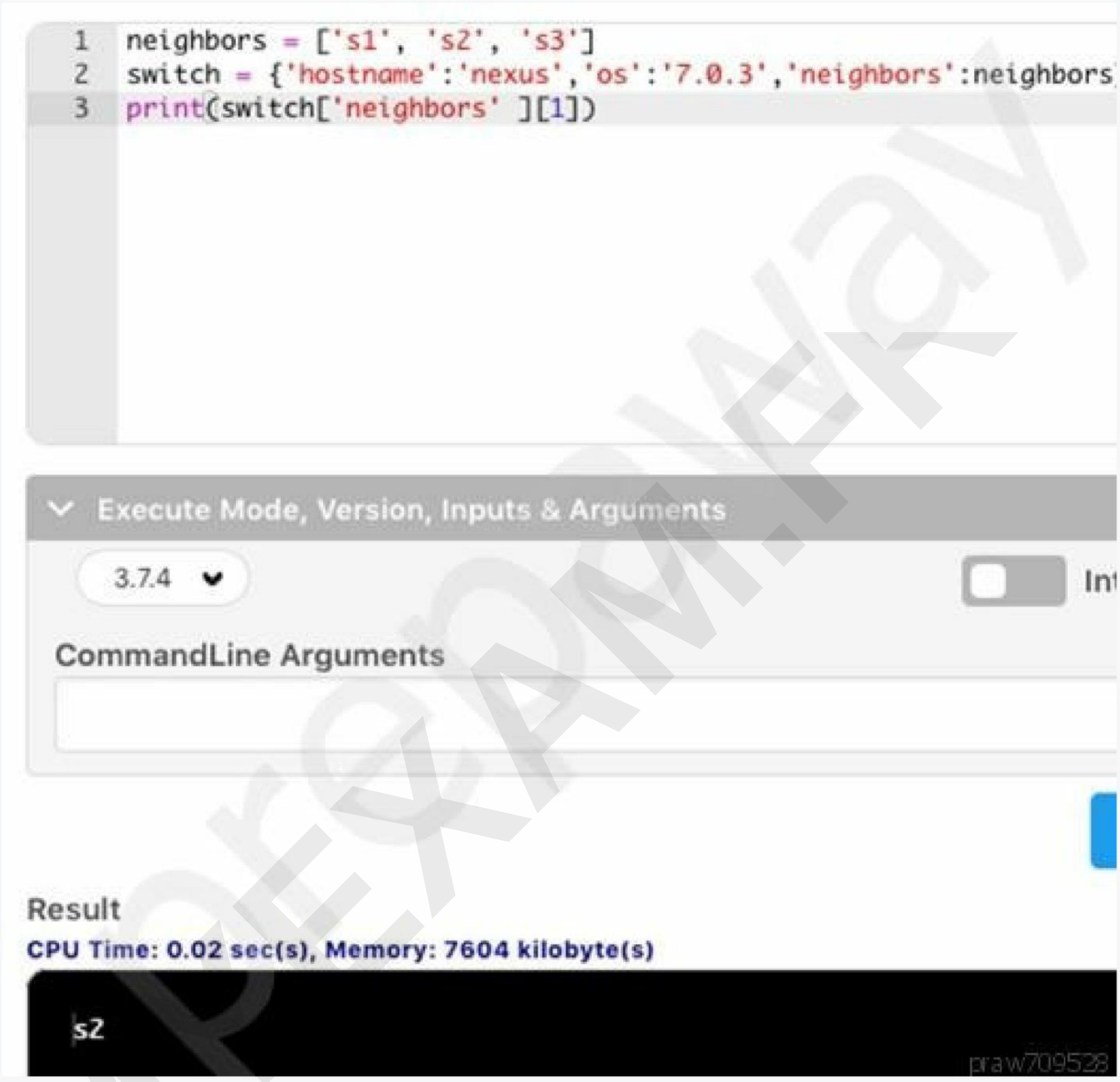
```
neighbors = ['s1', 's2', 's3']
switch = {'hostname':'nexus','os':'7.0.3','neighbors':neighbors}
print(switch['neighbors'][1])
```

Refer to the exhibit. What is the result when running the Python scripts?

A.s1
B.s2
C.s1, s2, s3
D.s3

Answer: B

Explanation:



```
neighbors = ['s1', 's2', 's3']
switch = {'hostname':'nexus','os':'7.0.3','neighbors':neighbors
print(switch['neighbors' ][1])
```

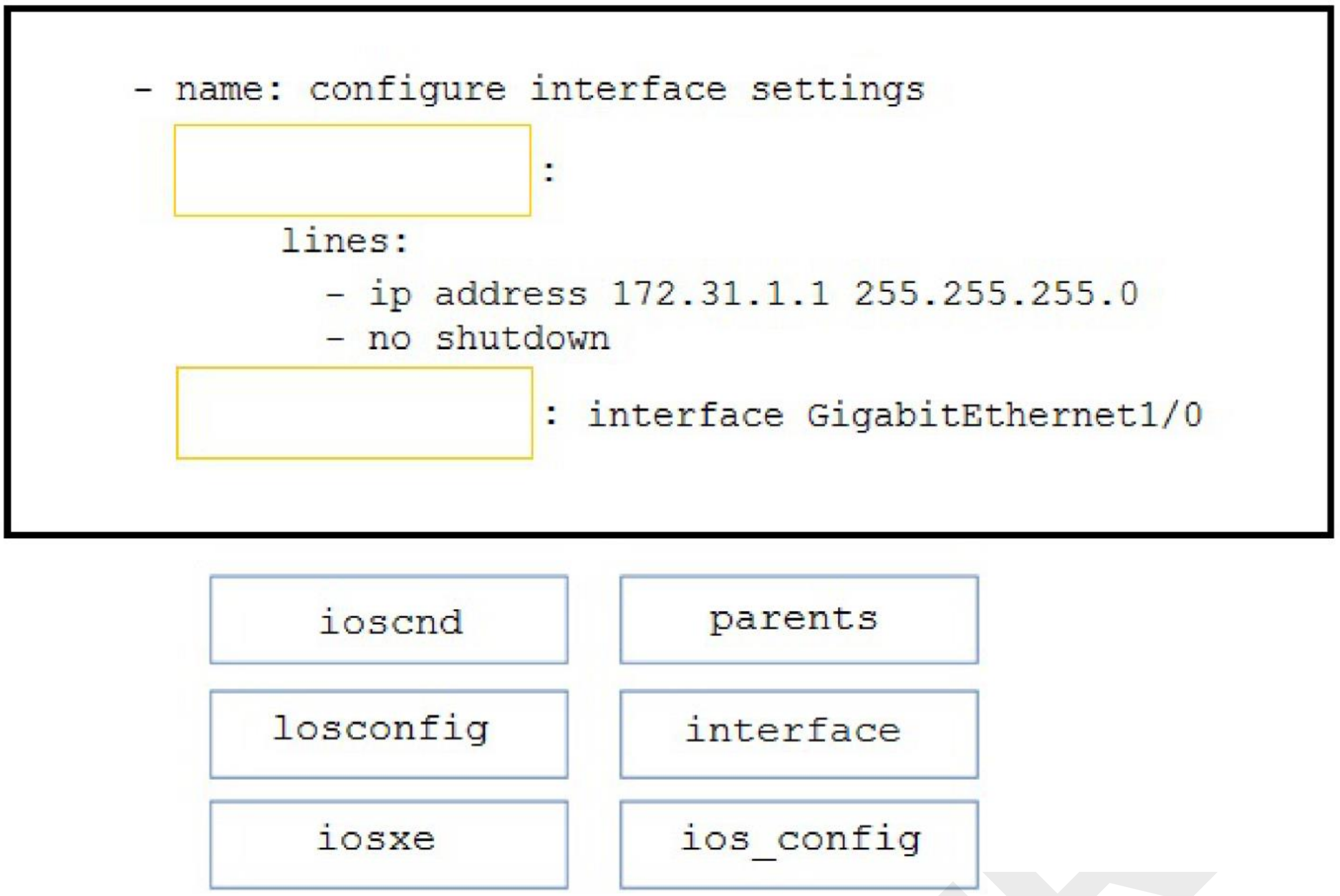
Result

CPU Time: 0.02 sec(s), Memory: 7604 kilobyte(s)

```
s2
```

Question: 10

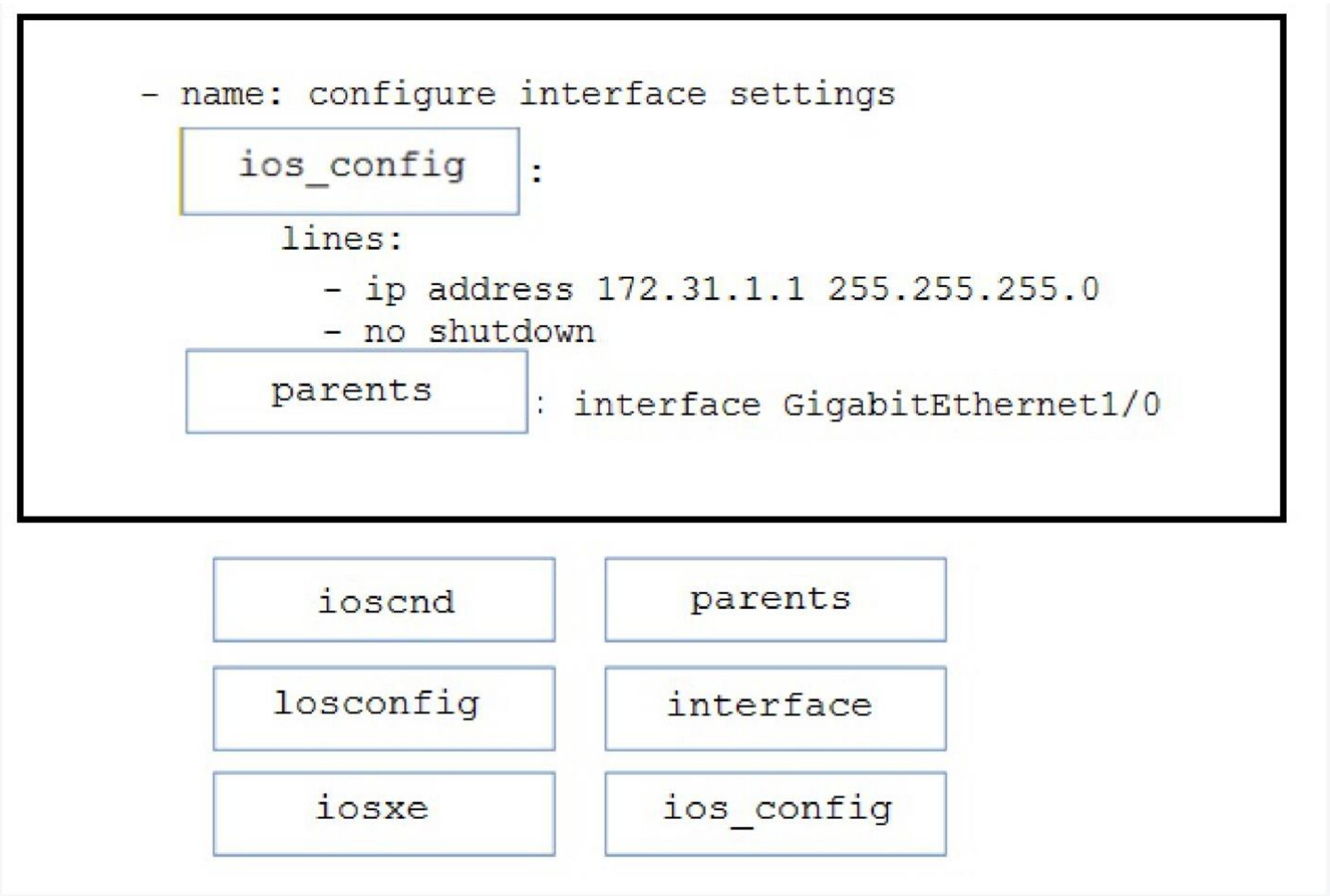
DRAG DROP -
Drag and drop the code from the bottom onto the box where the code is missing in the Ansible playbook to apply the configuration to an interface on a Cisco IOS
XE device. Not all options are used.
Select and Place:



```
- name: configure interface settings
                    :
     lines:
       - ip address 172.31.1.1 255.255.255.0
       - no shutdown
                    : interface GigabitEthernet1/0
```

```
ioscnd        parents
losconfig     interface
iosxe         ios_config
```

Answer:



```
- name: configure interface settings
  ios_config :
    lines:
      - ip address 172.31.1.1 255.255.255.0
      - no shutdown
  parents : interface GigabitEthernet1/0
```

Explanation:

IOS_Config.

Parents.

Question: 11

An engineer stored source code in a Git repository and is ready to develop a new feature. The production release is stored in the `master` branch. Which commands create the new feature in a separate branch called `feature` and check out the new version?

A.

```
git branch \
  https://git.cisco.com/python_programmer/device_status:feature
git checkout device_status:feature
```

B.

```
git remote add branch python_programmer/device_status:feature
git pull
```

C.

```
git branch feature
git checkout feature
```

D.

```
git remote add branch \
  https://git.cisco.com/python_programmer/device_status:feature
git push
```

Answer: C

Explanation:

Reference:
https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow

Question: 12

What is a benefit of developing an application in a Python virtual environment?

A.The application operates in multiple target systems simultaneously.
B.The application supports concurrency or multithreading.
C.The application operates across systems that have different operating systems.
D.The development environment is isolated from Python projects that already exist.

Answer: D

Explanation:

Justification:

The correct answer is **D. The development environment is isolated from Python projects that already exist.**

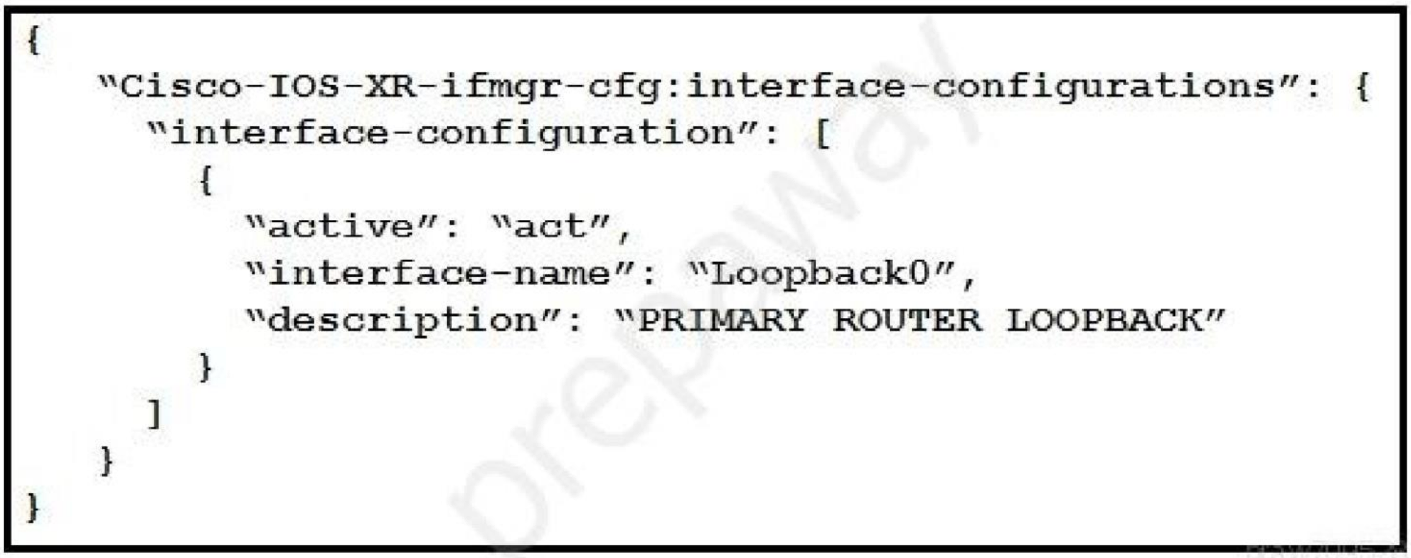
Python virtual environments are a fundamental practice in Python development for managing project dependencies and ensuring consistency. They create an isolated space for each project, preventing conflicts between package versions required by different projects. This isolation is crucial because various projects might rely on different versions of the same library. Without virtual environments, installing a package for one project might inadvertently break another project using an incompatible version. Option D precisely captures this core benefit of isolation, ensuring that the development environment remains clean and dedicated to the specific needs of the project. Options A, B, and C are not primary benefits of using virtual environments.

Virtual environments don't inherently make an application run on multiple systems (A), offer concurrency or multithreading (B), or directly support cross-OS operations (C). Instead, they promote modularity and avoid dependency clashes, leading to more reliable and maintainable applications. The isolation offered by a virtual environment makes collaboration among developers easier because they can recreate identical environments and avoid "works on my machine" scenarios.

Authoritative Links:

Python Virtual Environments Documentation:https://docs.python.org/3/tutorial/venv.html
Real Python - Python Virtual Environments:https://realpython.com/python-virtual-environments-primer/
GeeksforGeeks - Python Virtual Environment:https://www.geeksforgeeks.org/python-virtual-environment/

Question: 13



```
{
   "Cisco-IOS-XR-ifmgr-cfg:interface-configurations": {
     "interface-configuration": [
       {
         "active": "act",
         "interface-name": "Loopback0",
         "description": "PRIMARY ROUTER LOOPBACK"
       }
     ]
   }
}
```

Refer to the exhibit. Which type of YANG container is described by the JSON instance provided?

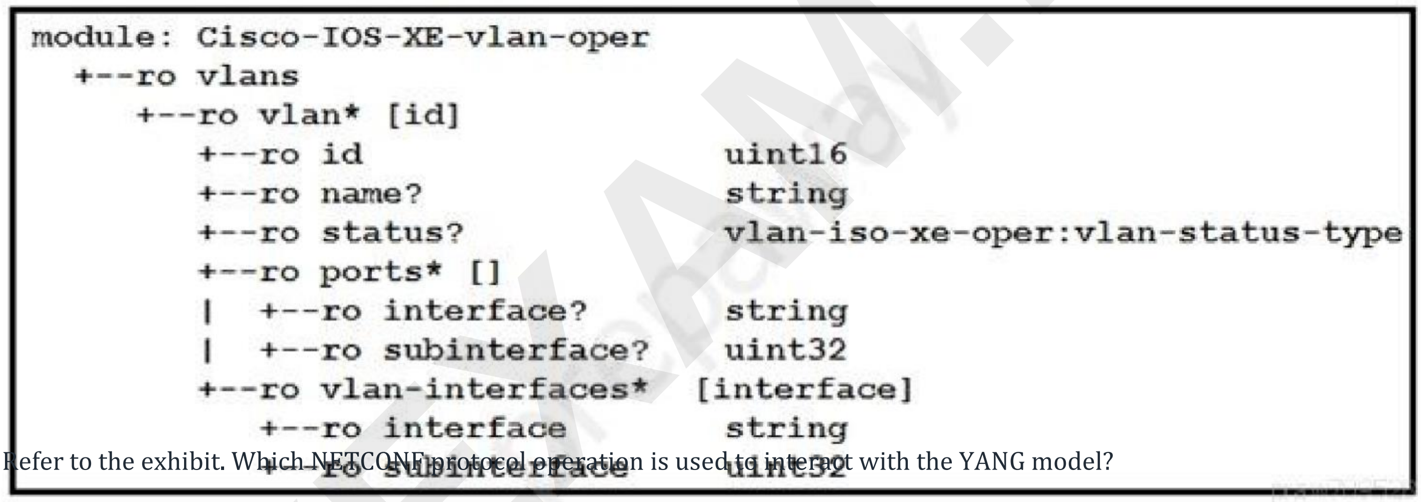
A.interface-configurations
B.active
C.interface-name
D.description

Answer: A

Explanation:

Reference:
https://www.cisco.com/c/en/us/td/docs/routers/asr9000/software/asr9k-r7-0/programmability/configuration/guide/b-programmability-cg-asr9000-70x/b- programmability-cg-asr9000-70x_chapter_011.html

Question: 14



```
module: Cisco-IOS-XE-vlan-oper
  +--ro vlans
     +--ro vlan* [id]
        +--ro id                    uint16
        +--ro name?                 string
        +--ro status?               vlan-iso-xe-oper:vlan-status-type
        +--ro ports* []
        |  +--ro interface?         string
        |  +--ro subinterface?      uint32
        +--ro vlan-interfaces*   [interface]
           +--ro interface          string
           +--ro subinterface       uint32
```

Refer to the exhibit. Which NETCONF protocol operation is used to interact with the YANG model?

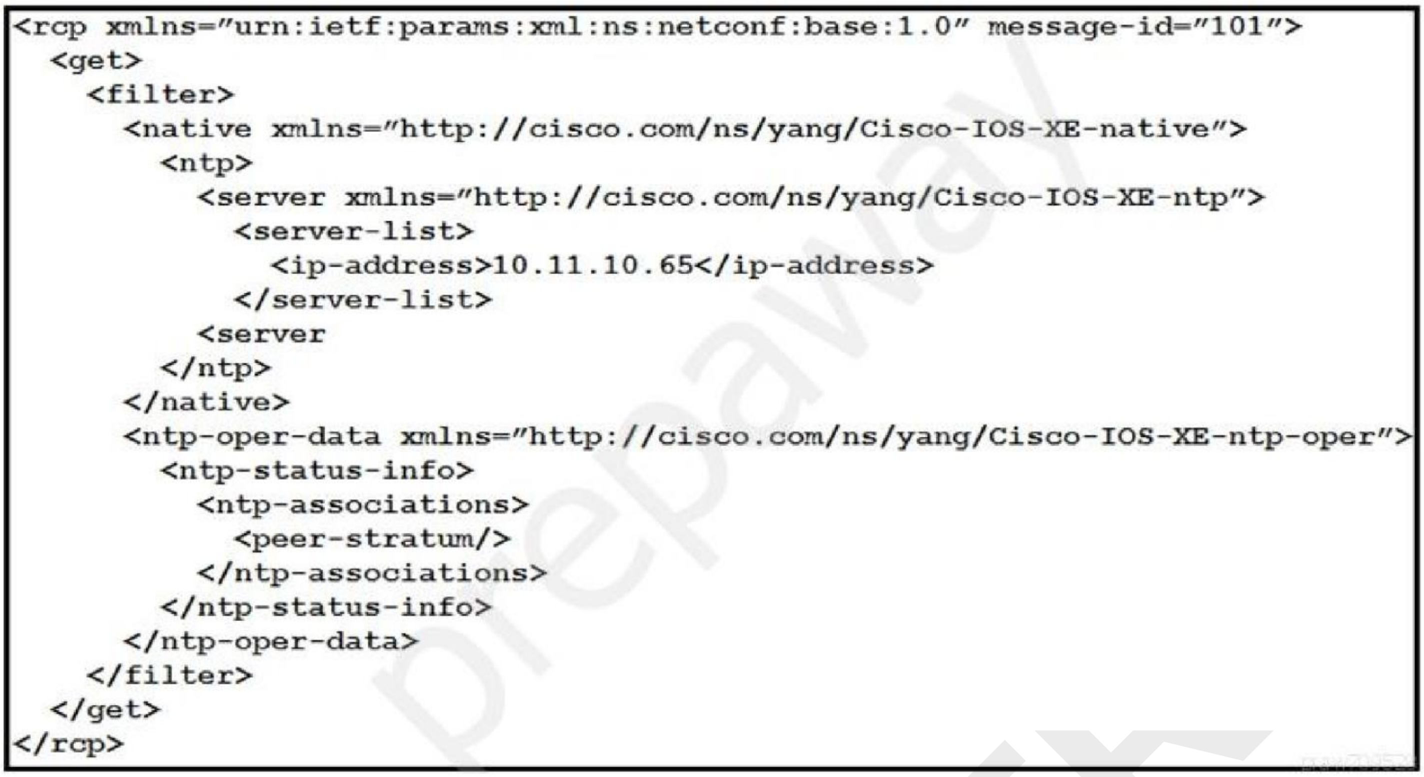
A.<edit-config>
B.<get>
C.<get-config>
D.<copy-config>

Answer: B

Explanation:

The exhibit is a read-only YANG leaf ("ro vlan") therefore is classed as "state" date and not config data. As such the <get> statement is what you need to use to interact with this particular YANG model.

Question: 15



```
<rcp xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <get>
    <filter>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <ntp>
          <server xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ntp">
            <server-list>
              <ip-address>10.11.10.65</ip-address>
            </server-list>
          <server
        </ntp>
      </native>
      <ntp-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ntp-oper">
        <ntp-status-info>
          <ntp-associations>
            <peer-stratum/>
          </ntp-associations>
        </ntp-status-info>
      </ntp-oper-data>
    </filter>
  </get>
</rcp>
```

Refer to the exhibit. How many YANG models does the NETCONF <get> operation interact with?

A.one
B.two
C.three
D.four

Answer: C

Explanation:

C - 3 cause <get> will get both config and oper data.

Question: 16

Which statement describe the difference between OpenConfig and native YANG data models?

A.Native models are designed to be independent of the underlying platform and are developed by vendors and standards bodies, such as the IETF.

B.Native models are developed by individual developers and designed to apply configurations on platforms.
C.OpenConfig models are developed by vendors and designed to integrate to features or configurations that are relevant only to that platform.

D.Native models are developed by vendors and designed to integrate to features or configurations that are relevant only to that platform.

Answer: D

Explanation:

The correct answer is **D. Native models are developed by vendors and designed to integrate to features or configurations that are relevant only to that platform.**

Here's a detailed justification:

YANG (Yet Another Next Generation) is a data modeling language used to describe configuration and operational data for network devices. Both OpenConfig and native YANG models use YANG as their foundation but differ significantly in their scope and purpose.

Native YANG Models: These are developed specifically by network equipment vendors like Cisco, Juniper, or Arista. They are tailored to the unique features, functionalities, and hardware specifications of their particular devices. Native models allow for granular control over all aspects of a device, including specific vendor-defined features. This also means they can be very specific to the vendor's device and may not be compatible across different vendors' products. Because native models expose the whole of the vendor's CLI capabilities through a structured data model, they tend to have more vendor-specific components and are tightly coupled with the specific implementation of the underlying platform.These are the data models provided by vendors that provide access to the unique CLI and capabilities of their particular operating system.

OpenConfig Models: In contrast, OpenConfig models are created by a collaborative group of network operators and technology vendors with the goal of developing vendor-agnostic network APIs and data models. OpenConfig aims for consistency and interoperability across different vendors’ equipment. These models are designed to be abstract and are not necessarily a direct mapping of the device's underlying CLI. They focus on widely used, fundamental networking concepts, enabling operators to manage networks from various vendors more consistently, aiming to simplify the process of automation. OpenConfig models use higher-level concepts that abstract away the details of the underlying hardware and platform.

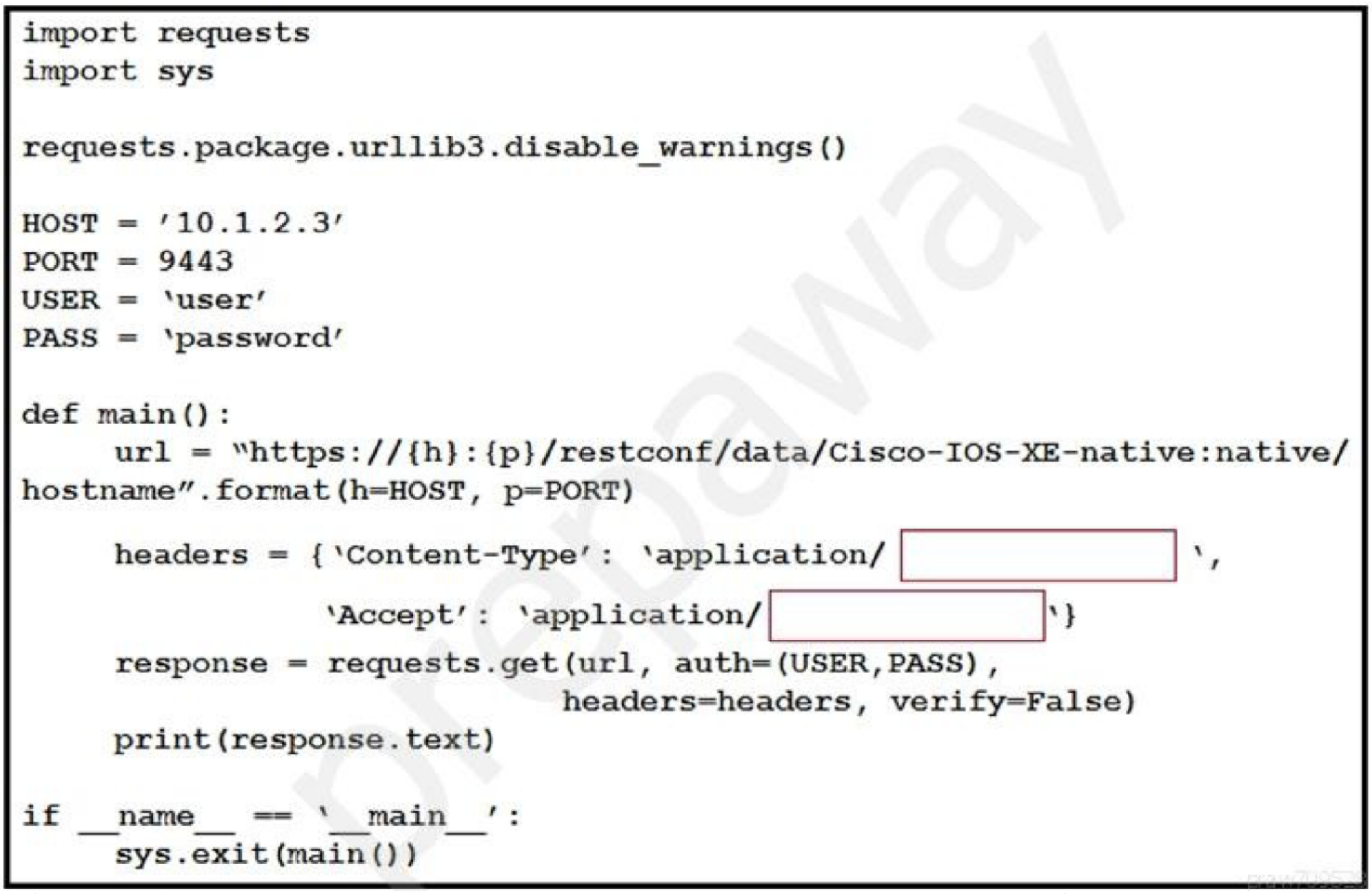
Option A is incorrect because it reverses the characteristics of native and OpenConfig models; OpenConfig models are designed for platform independence, not native models. Option B is inaccurate as native models are vendor-developed, not by individual developers. Option C is incorrect because OpenConfig models are designed for interoperability between different platforms.

In summary, while both are YANG based, native models expose vendor-specific device features, while OpenConfig aims for abstraction and vendor-agnostic management. This difference in design philosophy is why option D accurately distinguishes between the two.

Authoritative Links:

OpenConfig:https://www.openconfig.net/
YANG (RFC 7950):https://datatracker.ietf.org/doc/rfc7950/
Cisco's Introduction to Network Programmability:https://developer.cisco.com/site/programmability/ (Search for information about YANG and model-driven programmability)

Question: 17



```
import requests
import sys

requests.package.urllib3.disable_warnings()

HOST = '10.1.2.3'
PORT = 9443
USER = 'user'
PASS = 'password'

def main():
     url = "https://{h}:{p}/restconf/data/Cisco-IOS-XE-native:native/
hostname".format(h=HOST, p=PORT)

     headers = {'Content-Type': 'application/[            ]',
                'Accept': 'application/[            ]'}
     response = requests.get(url, auth=(USER,PASS),
                             headers=headers, verify=False)
     print(response.text)

if __name__ == '__main__':
     sys.exit(main())
```

Refer to the exhibit. An engineer creates a Python script using RESTCONF to display hostname information. The code must be completed so that it can be tested.
Which string completes the highlighted areas in the exhibit?

A.yang-data+json
B.yang +json
C.yang.data+json
D.json

Answer: A

Explanation:

Reference:
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/166/b_166_programmability_cg/restconf_prog_int.html

Question: 18

Which statement is true for Cisco IOS XE Software?

A.RESTCONF supports JSON and XML and NETCONF supports XML.
B.RESTCONF supports XML and NETCONF supports JSON and XML.
C.RESTCONF and NETCONF supports JSON and XML.
D.RESTCONF supports XML and NETCONF supports JSON.

Answer: A

Explanation:

The correct answer is A: RESTCONF supports JSON and XML, and NETCONF supports XML.

Let's break down why:

RESTCONF is an HTTP-based protocol that uses REST principles for network configuration and management. It leverages standard web technologies, making it easy to integrate with existing IT tools. A key aspect of REST is its ability to transfer data using various formats. Specifically, RESTCONF on Cisco IOS XE supports both JSON (JavaScript Object Notation) and XML (Extensible Markup Language) for representing data. JSON is often favored for its simplicity and ease of parsing in web applications, while XML offers more structured options, especially when data schemas are more complex.

NETCONF is a protocol designed explicitly for network configuration, utilizing a client-server model. It employs XML as its sole encoding method for representing configuration data and operations. NETCONF focuses on providing robust mechanisms for managing network devices and implementing configuration transactions. While it has a richer set of capabilities than RESTCONF, its restriction to XML is a fundamental design decision.

Therefore, statement A accurately captures the supported data formats for RESTCONF and NETCONF within the context of Cisco IOS XE. Option B is incorrect because it reverses the data format support. Option C is incorrect because NETCONF does not support JSON. Option D is incorrect as RESTCONF supports both JSON and XML.

In summary, Cisco IOS XE uses RESTCONF with the flexibility of JSON and XML, while NETCONF exclusively employs XML for its network management operations. This difference in data representation highlights the varied design philosophies of each protocol. RESTCONF is more versatile for web-based integrations, whereas NETCONF is a more tightly defined protocol optimized for complex network configuration.

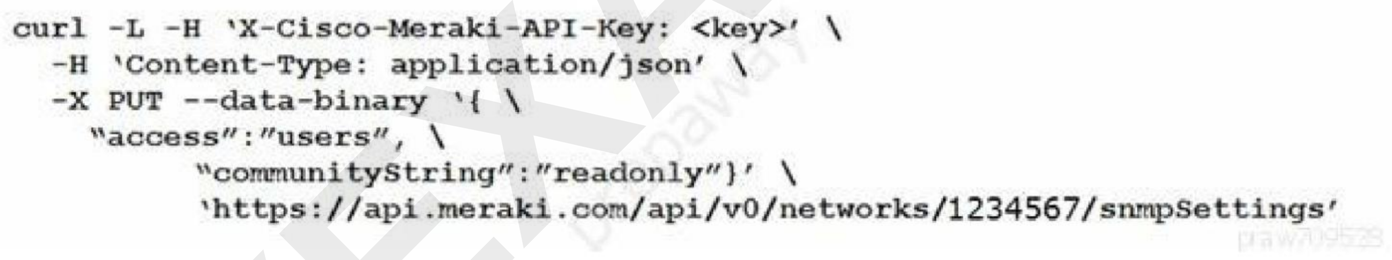
Authoritative Links:

1. **Cisco DevNet - RESTCONF:**https://developer.cisco.com/docs/ios-xe/#!restconf-on-ios-xe 2. **Cisco DevNet - NETCONF:**https://developer.cisco.com/docs/ios-xe/#!netconf-on-ios-xe 3. **IETF RFC 6241 - Network Configuration Protocol (NETCONF):** https://datatracker.ietf.org/doc/html/rfc6241
4. **IETF RFC 8040 - RESTCONF Protocol:**https://datatracker.ietf.org/doc/html/rfc8040

Question: 19

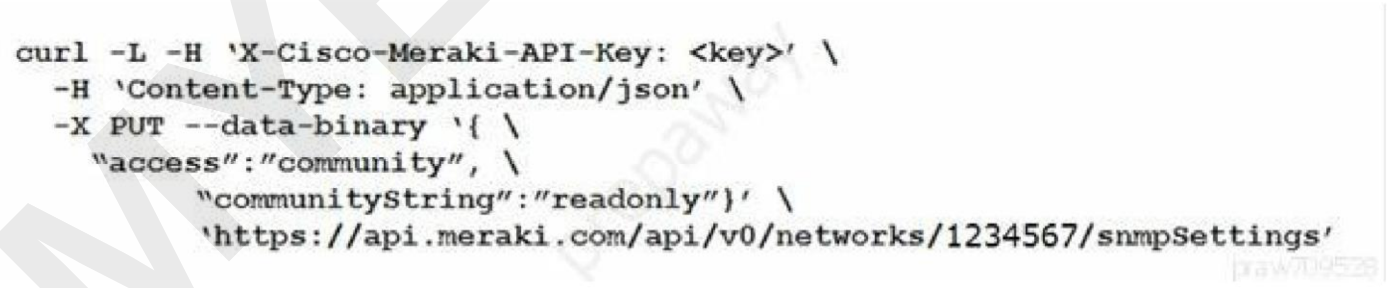
Which curl command is used to update the SNMP community of network ID `1234567` to read-only? A.

B.

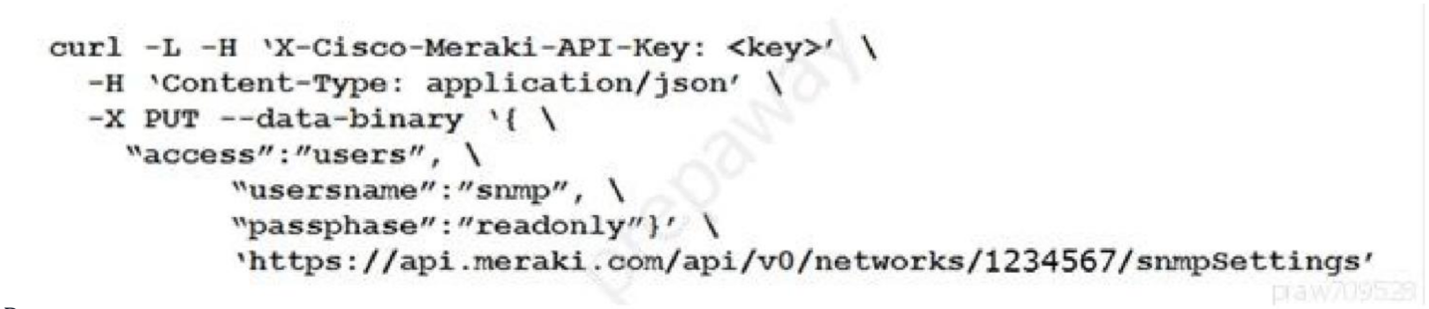


```
curl -L -H 'X-Cisco-Meraki-API-Key: <key>' \
  -H 'Content-Type: application/json' \
  -X PUT --data-binary '{ \
    "access":"users", \
           "communityString":"readonly"}' \
           'https://api.meraki.com/api/v0/networks/1234567/snmpSettings'
```

C.

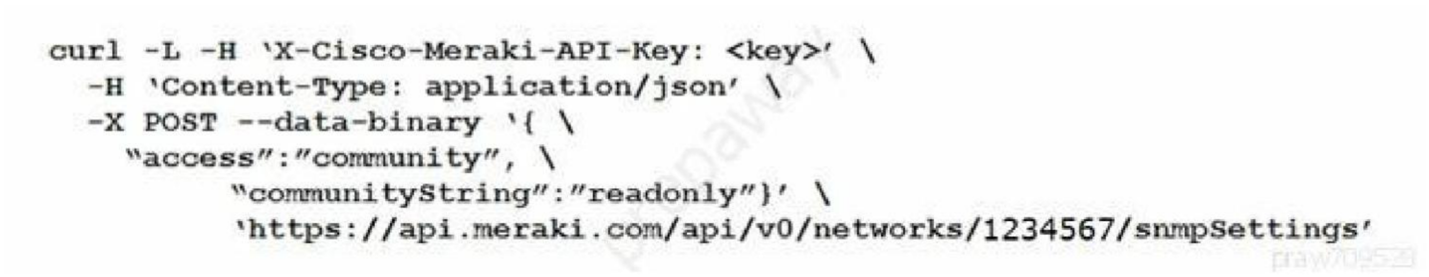


```
curl -L -H 'X-Cisco-Meraki-API-Key: <key>' \
  -H 'Content-Type: application/json' \
  -X PUT --data-binary '{ \
    "access":"community", \
           "communityString":"readonly"}' \
           'https://api.meraki.com/api/v0/networks/1234567/snmpSettings'
```



```
curl -L -H 'X-Cisco-Meraki-API-Key: <key>' \
  -H 'Content-Type: application/json' \
  -X PUT --data-binary '{ \
    "access":"users", \
          "usersname":"snmp", \
          "passphase":"readonly"}' \
          'https://api.meraki.com/api/v0/networks/1234567/snmpSettings'
```

D.



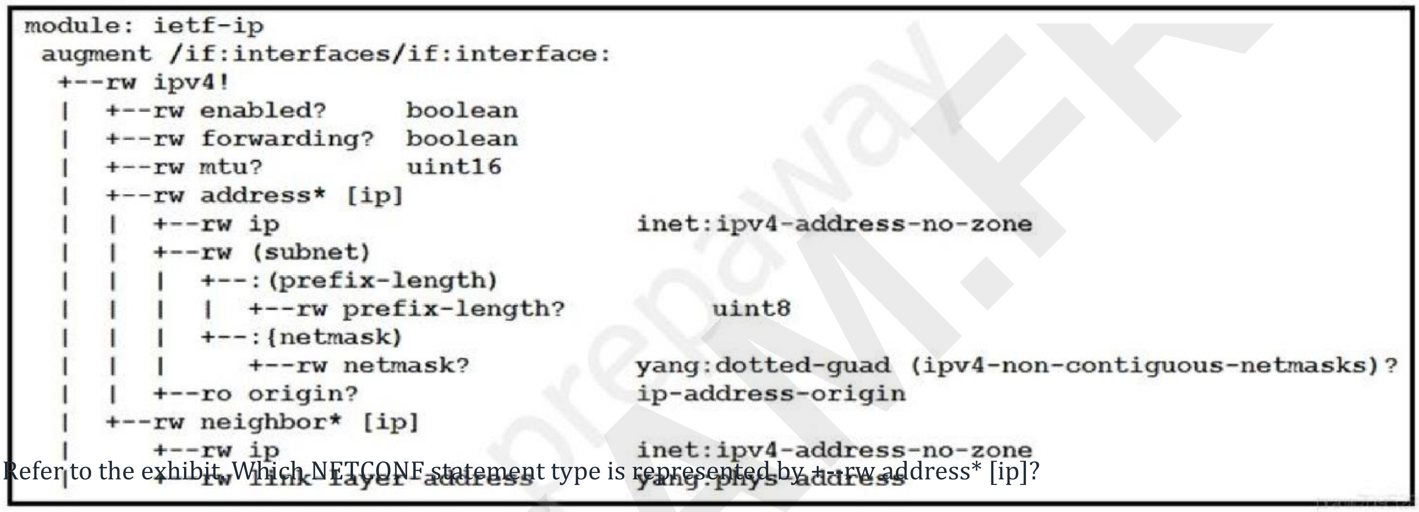
```
curl -L -H 'X-Cisco-Meraki-API-Key: <key>' \
  -H 'Content-Type: application/json' \
  -X POST --data-binary '{ \
    "access":"community", \
          "communityString":"readonly"}' \
          'https://api.meraki.com/api/v0/networks/1234567/snmpSettings'
```

Answer: B

Explanation:

PUT is used to update the snmp network ID. The access has to be community and not users. option B is correct.

Question: 20



```
module: ietf-ip
 augment /if:interfaces/if:interface:
  +--rw ipv4!
   |  +--rw enabled?     boolean
   |  +--rw forwarding?  boolean
   |  +--rw mtu?         uint16
   |  +--rw address* [ip]
   |  |  +--rw ip                    inet:ipv4-address-no-zone
   |  |  +--rw (subnet)
   |  |  |  +--:(prefix-length)
   |  |  |  |  +--rw prefix-length?        uint8
   |  |  |  +--:(netmask)
   |  |  |     +--rw netmask?        yang:dotted-guad (ipv4-non-contiguous-netmasks)?
   |  |  +--ro origin?               ip-address-origin
   |  +--rw neighbor* [ip]
   |     +--rw ip                    inet:ipv4-address-no-zone
   |     +--rw link-layer-address    yang:phys-address
```

Refer to the exhibit. Which NETCONF statement type is represented by +--rw address* [ip]?

A.list
B.leaf-list
C.container
D.submodule

Answer: A

Explanation:
Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.

Question: 21

The automation engineer must replace device configuration using RESTCONF. How is this configured using the Python library Requests?

A.delete()
B.post()
C.put()
D.patch()

Answer: C

Explanation:

The correct answer is **C. put()**. RESTCONF utilizes standard HTTP methods to interact with network devices. When the goal is to completely replace a device's configuration, the PUT method is the appropriate choice.

PUT requests are idempotent, meaning that making the same request multiple times will result in the same outcome - the resource will be replaced with the provided payload. This behavior makes PUT ideal for configuration replacement, as it ensures a consistent state regardless of prior configurations. In contrast, the POST method is typically used to create new resources, while DELETE is used to remove them. The PATCH method, although also used for updates, is meant for partial modifications, not complete replacements. Since the question specifies a full replacement of the configuration, using patch() could lead to unintended, residual configurations. Using the Python requests library, requests.put() sends an HTTP PUT request to the specified RESTCONF endpoint, effectively applying the new configuration in place of the old one. This operation overwrites the resource at the target URI. The provided payload will define the complete, new configuration for the device. If using a different HTTP method, such as post(), the request would either create an additional, unwanted resource or result in an error. Therefore, put() is the only appropriate method to replace the existing configuration. For further understanding, please refer to:

RFC 7231 (HTTP/1.1):https://www.rfc-editor.org/rfc/rfc7231#section-4.3.4 This document provides official specifications on the semantics of the PUT method.

RESTCONF Overview:https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/16-6/b_16-6_programmability_cg/m_restconf.html This document from Cisco provides a general explanation of RESTCONF usage, illustrating the methods and their functionalities.

Python Requests library:https://requests.readthedocs.io/en/latest/ This is the official documentation for the requests library, explaining how to use methods like put(), post(), delete(), and patch().

Question: 22

Which function is available in NETCONF and unavailable in RESTCONF?

A.configuration changes are automatically activated
B.uses the YANG data models to communicate
C.supports JSON and data encoding
D.validates the content of a candidate datastore

Answer: D

Explanation:

The correct answer is D: validates the content of a candidate datastore. NETCONF and RESTCONF are both network management protocols that leverage YANG data models for structured data exchange, and both support various data encodings like JSON. A key distinction, however, lies in how they handle configuration

changes. NETCONF offers a more sophisticated, transaction-oriented approach using a "candidate datastore". This datastore acts as a staging area where configuration changes can be validated against the defined YANG model before being committed to the "running datastore". This validation ensures that only valid
configurations are applied, reducing the risk of errors and network disruptions. RESTCONF, while using YANG for data definitions and supporting data encoding, directly modifies the running configuration through HTTP methods. It lacks the explicit concept of a candidate datastore and the built-in validation process that NETCONF provides. This makes NETCONF better suited for complex configuration scenarios that require high reliability and consistency. Specifically, NETCONF's <validate> operation can be used to check configuration before committing, which is not available in RESTCONF. In summary, although both protocols use YANG models and support encoding, NETCONF's transaction-based configuration with candidate datastore and validation makes it distinct from RESTCONF.

Relevant Links:

NETCONF Protocol:https://datatracker.ietf.org/doc/html/rfc6241
RESTCONF Protocol:https://datatracker.ietf.org/doc/html/rfc8040
YANG Data Modeling Language:https://datatracker.ietf.org/doc/html/rfc7950

Question: 23

Which setting is used for the dampening period when configuring an on-change publication for YANG-push versus OpenConfig?

A.null
B.-1
C.0
D.1000

Answer: C

Explanation:

The correct answer is **C. 0**.

When configuring on-change publications with YANG-push, the dampening period determines how long a system waits after a change before sending a notification. A dampening period helps to prevent a flood of notifications when rapid, transient changes occur. It allows for a more stable and less noisy stream of updates. YANG-push is designed to be highly efficient and responsive; therefore, it's often desired to send notifications as immediately as possible following a relevant change, rather than introduce a delay.

In this context, a value of "0" indicates that no dampening is applied and notifications are sent as soon as a change is detected. While a non-zero value is used to introduce a delay, the question specifically seeks the setting for minimal delay and immediate publication. When using a non-zero value, the notification will be sent after that amount of seconds even if the resource is changed again.

OpenConfig uses a different dampening mechanism, while the question refers to "YANG-push" and hence the correct answer is “0”. Values like "-1" or "null" are typically not recognized for dampening purposes and would most likely result in errors or default behaviors, not the intended immediate publishing. A large value like "1000" would introduce a significant delay, going against the concept of immediate on-change publication.

Therefore, option C, 0, accurately represents the setting for no dampening or immediate change notification using YANG-push.

Authoritative Links:

RFC 8639 - Subscription to YANG Notifications:https://datatracker.ietf.org/doc/html/rfc8639 (This is the primary RFC defining YANG push, and while it doesn't explicitly define the dampening setting, it provides the basis on which such settings are built).

Cisco Documentation on YANG-push: (Refer to Cisco's official documentation on configuring YANG-push on specific devices; this will provide context on how they implement the dampening settings). Note that direct links may vary with the Cisco device and software versions, you can search the device documentation on cisco.com for the keyword "yang push" and your device model to find the relevant page.

Question: 24

Setting is used for the dampening period when configuring an on-change publication for YANG-push versus OpenConfig.
What are two characteristics of synchronous calls to APIs? (Choose two.)

A.They block until a response is returned from the servers

B.They make an application less portable, so asynchronous calls are preferred

C.They add perceived latency to an application if data is not received

D.Calls are limited to specific programming languages

E.They do not block while waiting for the API to be processed

Answer: AC

Explanation:

The correct answers are A and C. Synchronous API calls, by their nature, operate in a blocking manner. This means that when an application initiates a synchronous request, it pauses its execution and waits until it receives a response from the server. This "waiting" period directly translates to perceived latency for the user, especially if the server takes time to process the request. Option A accurately reflects this behavior.

Furthermore, the application's flow is held up because it's dependent on the server's response, leading to a direct increase in latency. Option C is also correct as perceived delays are unavoidable while the client waits for a synchronous API call to return.

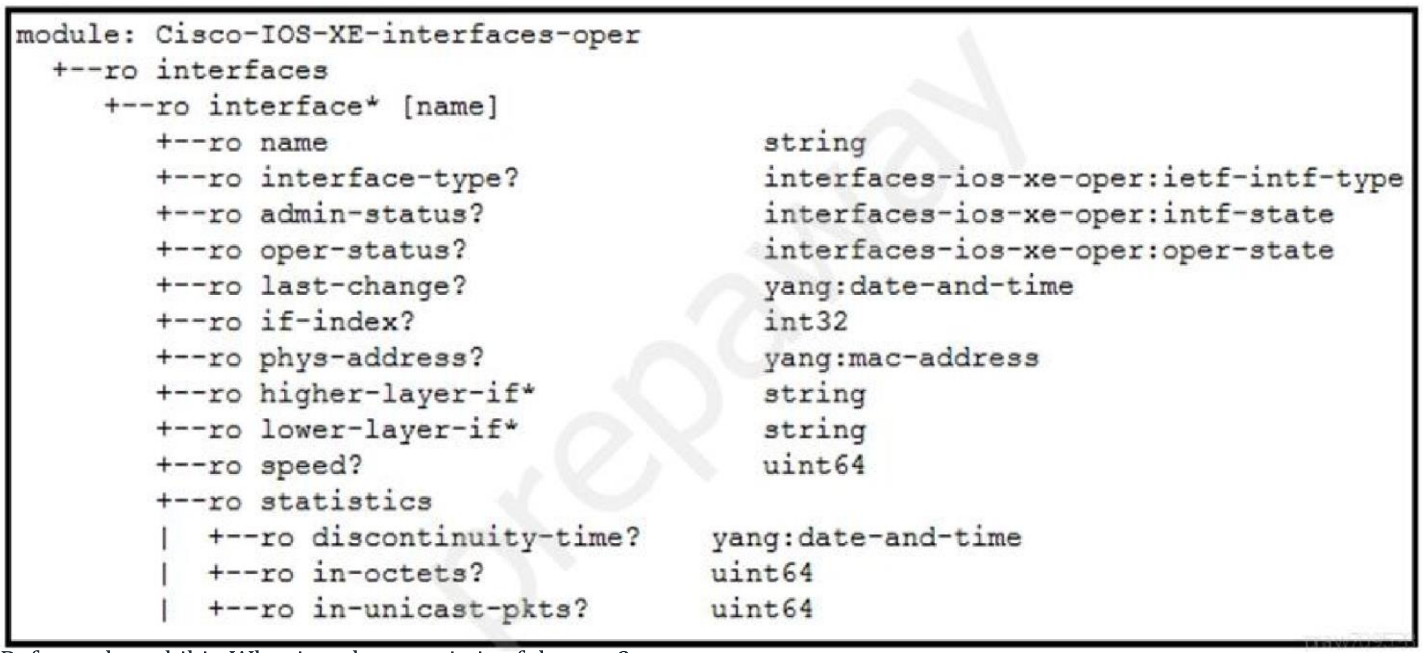
Options B, D, and E are incorrect. Synchronous calls don't intrinsically make an application less portable (B), as both synchronous and asynchronous methods can be implemented across different platforms and languages. Synchronous calls are not limited to specific languages (D); most programming languages support making both synchronous and asynchronous API calls. Finally, synchronous calls inherently do block while waiting for API processing to complete (E).

Further research on synchronous versus asynchronous API calls and their impact on application performance can be found here:

Microsoft Docs on Asynchronous Programming:https://docs.microsoft.com/en-us/dotnet/standard/async-in-depth
Red Hat - Understanding synchronous and asynchronous processing in APIs:
https://developers.redhat.com/blog/2018/10/23/understanding-synchronous-and-asynchronous-processing-in-apis

These resources provide a more comprehensive understanding of the characteristics and use cases of synchronous and asynchronous communication patterns.

Question: 25



```
module: Cisco-IOS-XE-interfaces-oper
  +--ro interfaces
     +--ro interface* [name]
        +--ro name                          string
        +--ro interface-type?               interfaces-ios-xe-oper:ietf-intf-type
        +--ro admin-status?                 interfaces-ios-xe-oper:intf-state
        +--ro oper-status?                  interfaces-ios-xe-oper:oper-state
        +--ro last-change?                  yang:date-and-time
        +--ro if-index?                     int32
        +--ro phys-address?                 yang:mac-address
        +--ro higher-layer-if*              string
        +--ro lower-layer-if*               string
        +--ro speed?                        uint64
        +--ro statistics
        |  +--ro discontinuity-time?    yang:date-and-time
        |  +--ro in-octets?             uint64
        |  +--ro in-unicast-pkts?       uint64
```

Refer to the exhibit. What is a characteristic of the tree?

A.three optional metrics
B.two leaf-lists
C.ten leaf-lists
D.three containers

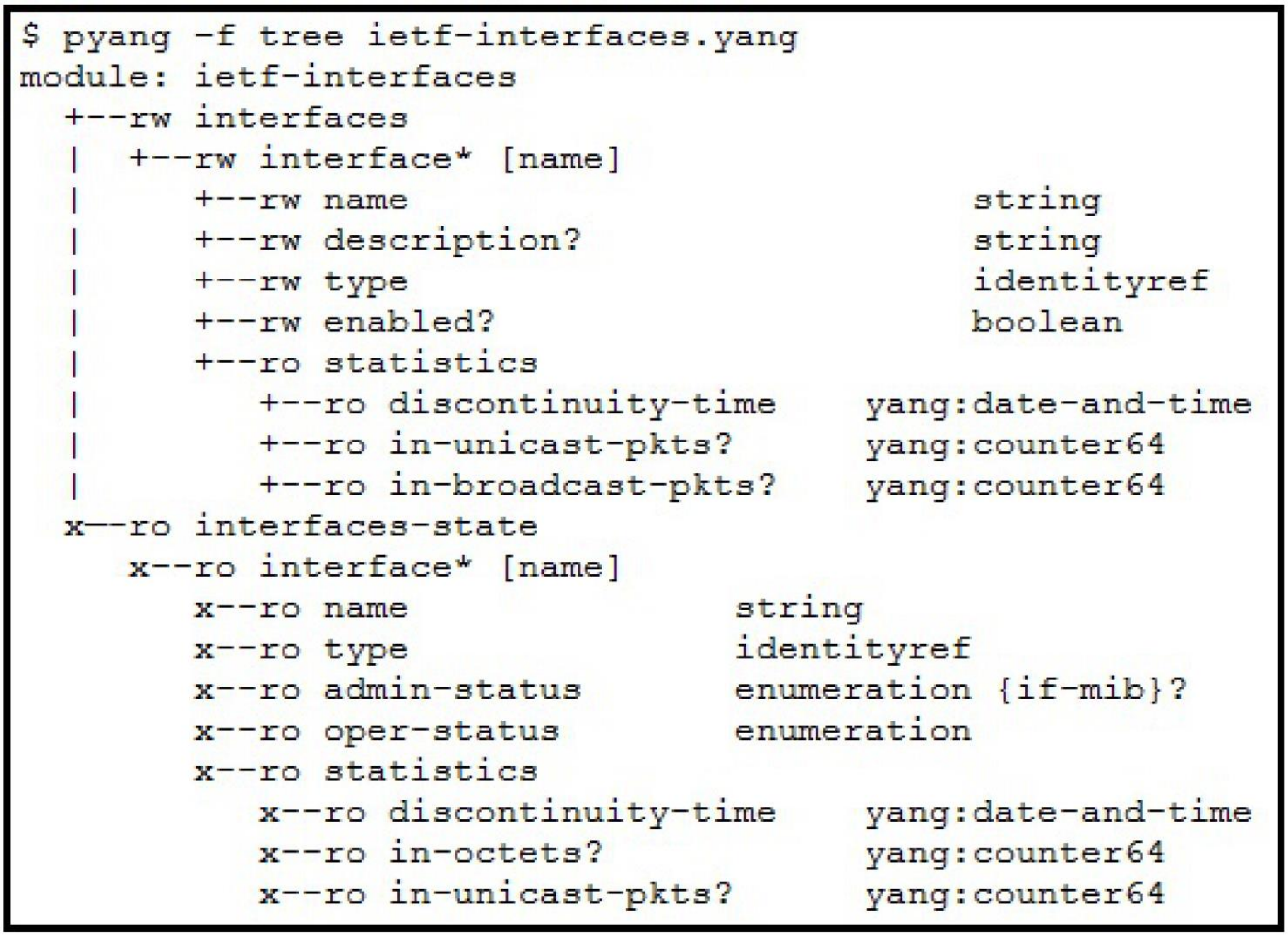
Answer: B

Explanation:

Correct answer is B:two leaf-lists.

Question: 26

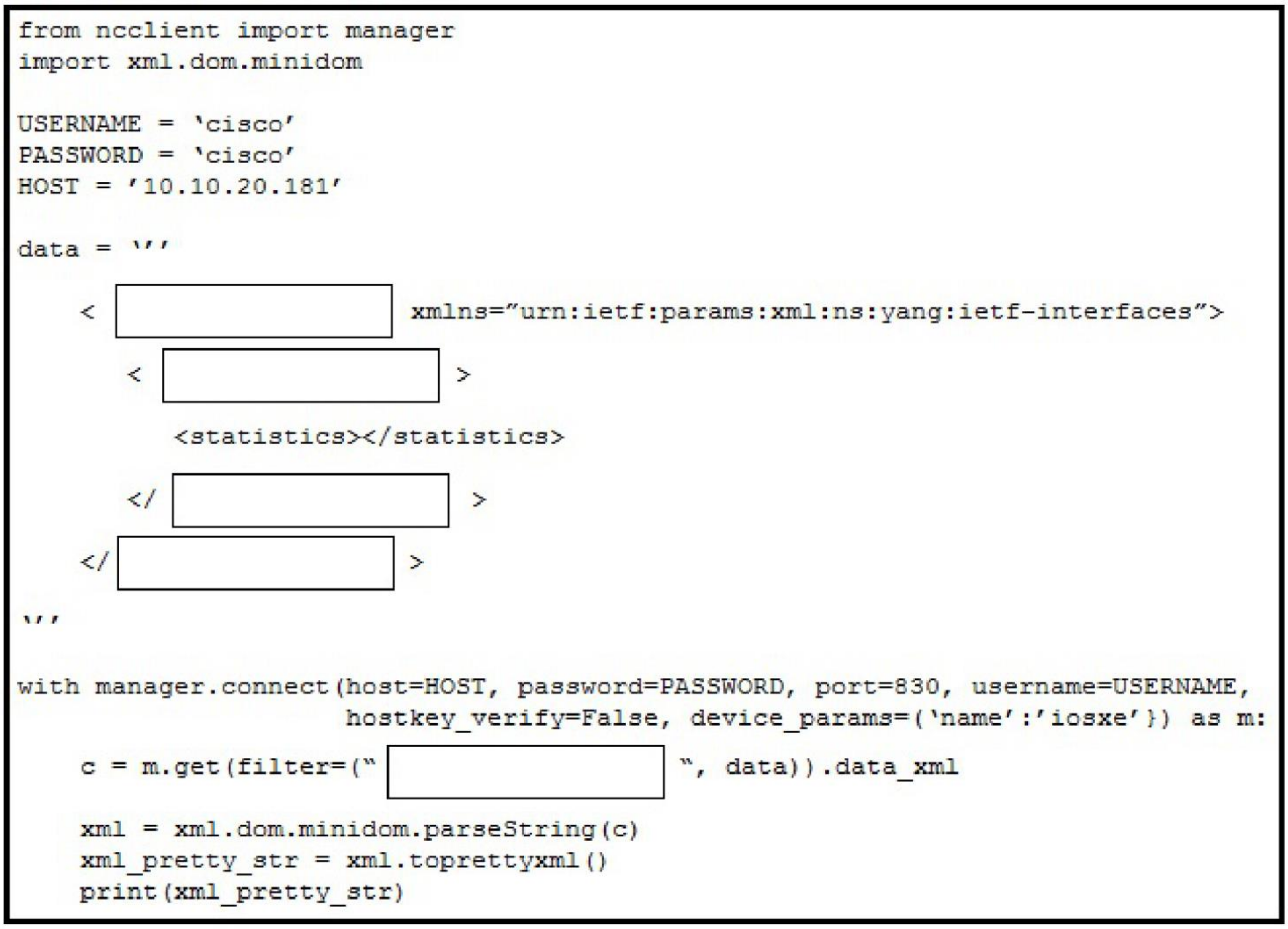
DRAG DROP -



```
$ pyang -f tree ietf-interfaces.yang
module: ietf-interfaces
  +--rw interfaces
  |  +--rw interface* [name]
  |     +--rw name                                  string
  |     +--rw description?                          string
  |     +--rw type                                  identityref
  |     +--rw enabled?                              boolean
  |     +--ro statistics
  |        +--ro discontinuity-time    yang:date-and-time
  |        +--ro in-unicast-pkts?      yang:counter64
  |        +--ro in-broadcast-pkts?    yang:counter64
  x--ro interfaces-state
     x--ro interface* [name]
        x--ro name                 string
        x--ro type                 identityref
        x--ro admin-status         enumeration {if-mib}?
        x--ro oper-status          enumeration
        x--ro statistics
           x--ro discontinuity-time    yang:date-and-time
           x--ro in-octets?            yang:counter64
           x--ro in-unicast-pkts?      yang:counter64
```

Refer to the exhibit. Drag and drop the code from the bottom onto the box where the code is missing to complete the ncclient request that captures the operational data of the interfaces of a Cisco IOS XE device. Options may be used once, more than once, or not at all.

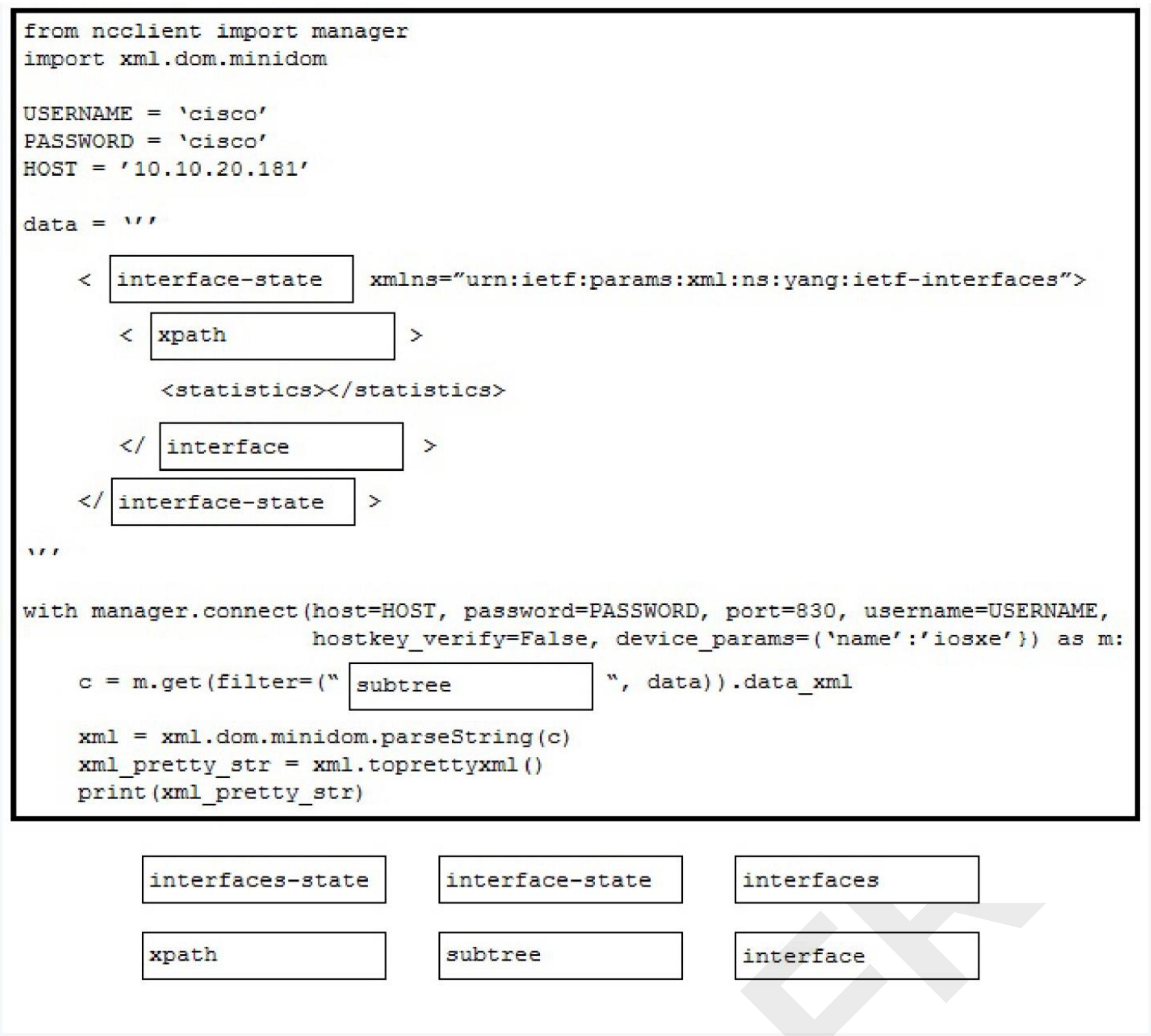
Select and Place:



```
from ncclient import manager
import xml.dom.minidom

USERNAME = 'cisco'
PASSWORD = 'cisco'
HOST = '10.10.20.181'

data = '''

    <                  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">

       <                  >

          <statistics></statistics>

       </                  >

    </                  >

'''

with manager.connect(host=HOST, password=PASSWORD, port=830, username=USERNAME,
                     hostkey_verify=False, device_params=('name':'iosxe'}) as m:

    c = m.get(filter=("                  ", data)).data_xml

    xml = xml.dom.minidom.parseString(c)
    xml_pretty_str = xml.toprettyxml()
    print(xml_pretty_str)
```

interfaces-state	interface-state	interfaces
xpath	subtree	interface

Answer:



```
from ncclient import manager
import xml.dom.minidom

USERNAME = 'cisco'
PASSWORD = 'cisco'
HOST = '10.10.20.181'

data = '''

    < interface-state  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">

       < xpath >

          <statistics></statistics>

       </ interface >

    </ interface-state >

'''

with manager.connect(host=HOST, password=PASSWORD, port=830, username=USERNAME,
                     hostkey_verify=False, device_params=('name':'iosxe'}) as m:

    c = m.get(filter=(" subtree ", data)).data_xml

    xml = xml.dom.minidom.parseString(c)
    xml_pretty_str = xml.toprettyxml()
    print(xml_pretty_str)
```

interfaces-state	interface-state	interfaces
xpath	subtree	interface

Question: 27

When working with MV Sense APIs, which type of protocol is MQTT based upon?

A.publish-subscribe messaging protocol
B.simple mail transport protocol
C.heavyweight messaging protocol
D.computer vision protocol

Answer: A

Explanation:

The correct answer is **A. publish-subscribe messaging protocol**. MQTT (Message Queuing Telemetry Transport) is fundamentally built upon the publish-subscribe architectural pattern. This pattern decouples message senders (publishers) from message receivers (subscribers). Publishers send messages to a central broker, without knowing specific recipients. The broker then filters and distributes messages to subscribed clients based on topics.

This is crucial for efficient communication in scenarios like IoT and sensor networks, where devices may have intermittent connectivity or limited resources. The lightweight nature of MQTT makes it well-suited for constrained environments often encountered when working with sensors, such as those found in the MV Sense environment. These sensors may frequently send data updates about their readings, and publish-subscribe allows for efficient consumption of this data by interested applications.

MQTT avoids the need for direct point-to-point connections between each sensor and each application, which would be much more complex and resource intensive. Using a central broker simplifies network management and message routing. In the context of MV Sense APIs, which provide access to sensor data, MQTT enables a scalable and adaptable way to handle many sensor updates.

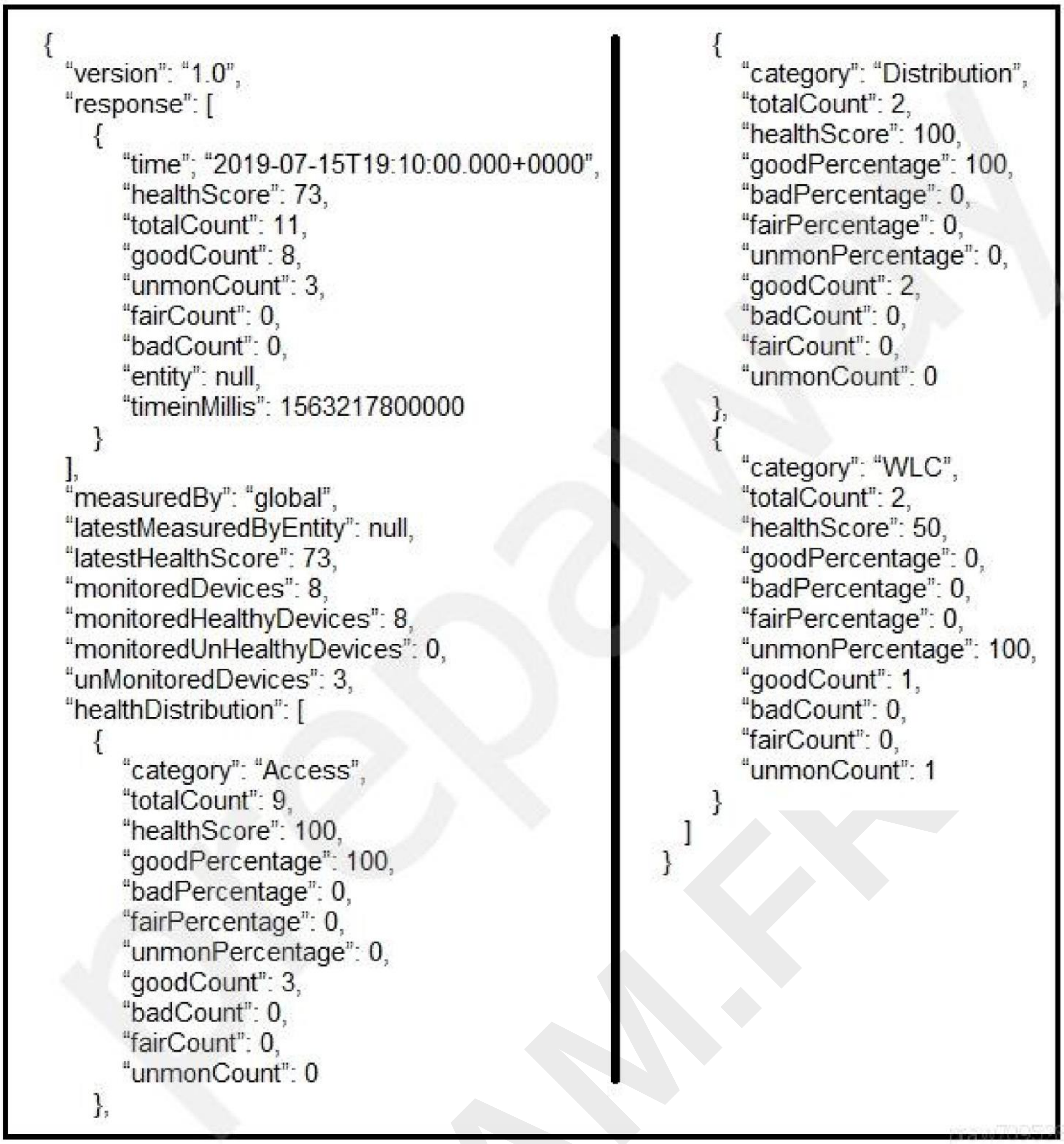
This is quite different from protocols like Simple Mail Transport Protocol (SMTP) which is designed for email transport. SMTP operates on a push model, specifically for user-to-user messaging. A "heavyweight" protocol is a mischaracterization of MQTT, and it is definitively not a computer vision protocol.

In summary, the inherent publish-subscribe nature of MQTT is its defining characteristic and what makes it perfect for use in sensor networks and data streaming environments like MV Sense APIs.

Authoritative links for further research:

MQTT Specification:http://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html
MQTT Explained (IBM):https://www.ibm.com/topics/mqtt
Understanding MQTT:https://www.hivemq.com/mqtt-essentials/

Question: 28



```
{
  "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
    }
  ]
}
```

Refer to the exhibit. Which device type is functioning in a degraded state?

A.access point
B.distribution switch
C.access switch
D.wireless LAN controller

Answer: D

Explanation:

Correct answer is D:wireless LAN controller.

Question: 29

Which two Netmiko methods are used to configure a device? (Choose two.)

A.send_config()
B.send_control_from_file()
C.send_config_set()
D.send_command()
E.send_config_from_file()

Answer: CE

Explanation:

The correct Netmiko methods for configuring a device are send_config_set() and send_config_from_file().

send_config_set() is designed to take a list of configuration commands as input and apply them to the device. This method is suitable for dynamic configurations where the commands are generated programmatically or stored in a list within your automation script. It allows for granular control over the configuration changes being applied. Netmiko intelligently handles the necessary command sequences to enter configuration mode, send the commands, and exit configuration mode.

send_config_from_file() is used when configuration commands are stored in an external text file. This is useful for standard configurations, where the entire device configuration or a segment of it can be pre-written and applied. The method reads the commands from the specified file and sends them to the device, simplifying the process of managing large or complex configurations.

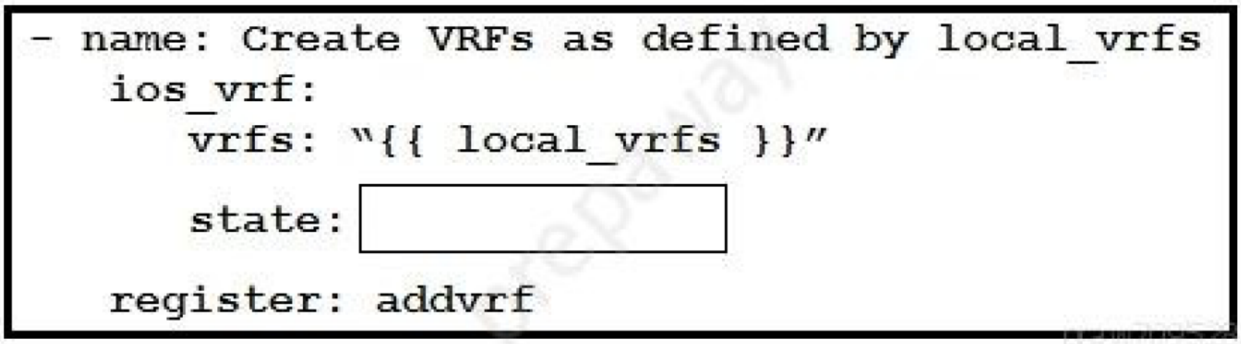
send_command() is primarily for executing show commands or other operational commands, not for making configuration changes. send_control_from_file() is not a standard Netmiko method, and therefore, it is not an answer. send_config() is also not a standard Netmiko method.

These two methods, send_config_set() and send_config_from_file(), provide a flexible way to push configurations to network devices using Netmiko, enabling the automation of network administration tasks such as changing interface parameters, configuring routing protocols, and setting access lists. They work through an SSH connection established by Netmiko, communicating via the device's command-line interface.

Here are authoritative resources for further study:

1. **Netmiko Documentation (Configuring Devices):**https://pynet.twb-tech.com/blog/netmiko-config.html
2. **Practical Python with Netmiko:**https://www.youtube.com/watch?v=d36a96-U8-g
3. **Network Automation with Python and Netmiko**https://realpython.com/network-automation-python-netmiko/

Question: 30



```
- name: Create VRFs as defined by local_vrfs
  ios_vrf:
     vrfs: "{{ local_vrfs }}"
     state: [          ]
  register: addvrf
```

Refer to the exhibit. An engineer creates an Ansible playbook to configure VRF information using a local_vrfs variable. The code must be completed so that it can be tested. Which string completes the code?

A.present
B.up
C.on
D.active

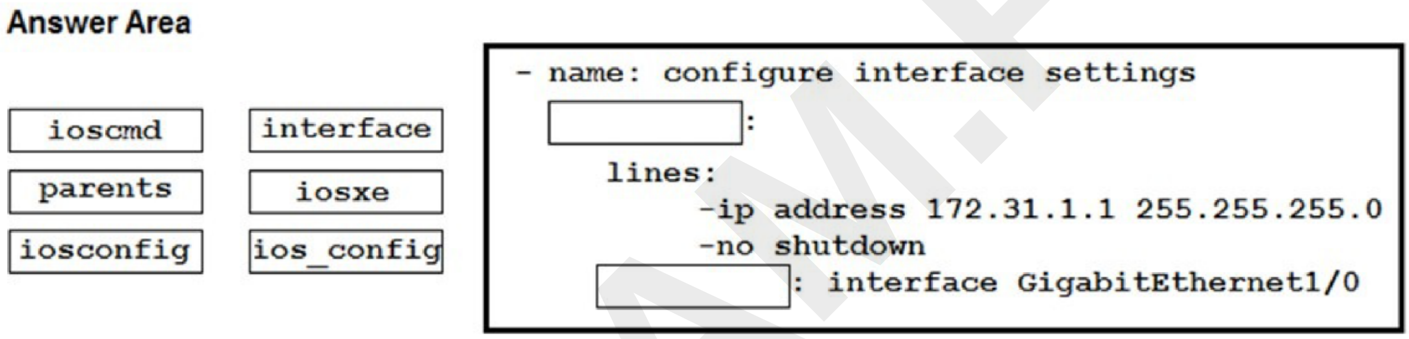
Answer: A

Explanation:

Reference:
https://docs.ansible.com/ansible/latest/modules/ios_vrf_module.html

Question: 31

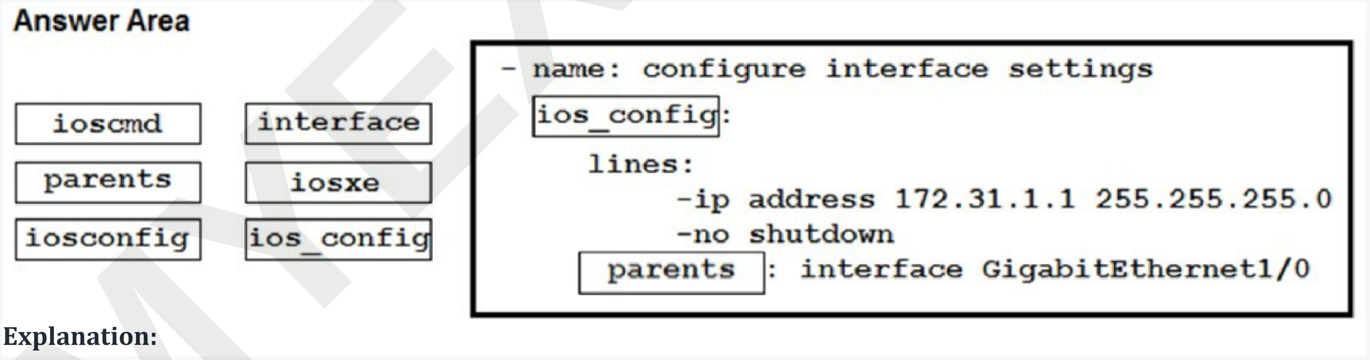
DRAG DROP -
Drag and drop the commands to the Ansible playbook that applies configuration to an interface on a Cisco IOS XE device. Not all options are used.
Select and Place:



Answer Area

```
- name: configure interface settings
  [          ]:
     lines:
         -ip address 172.31.1.1 255.255.255.0
         -no shutdown
    [          ]: interface GigabitEthernet1/0
```

Answer:



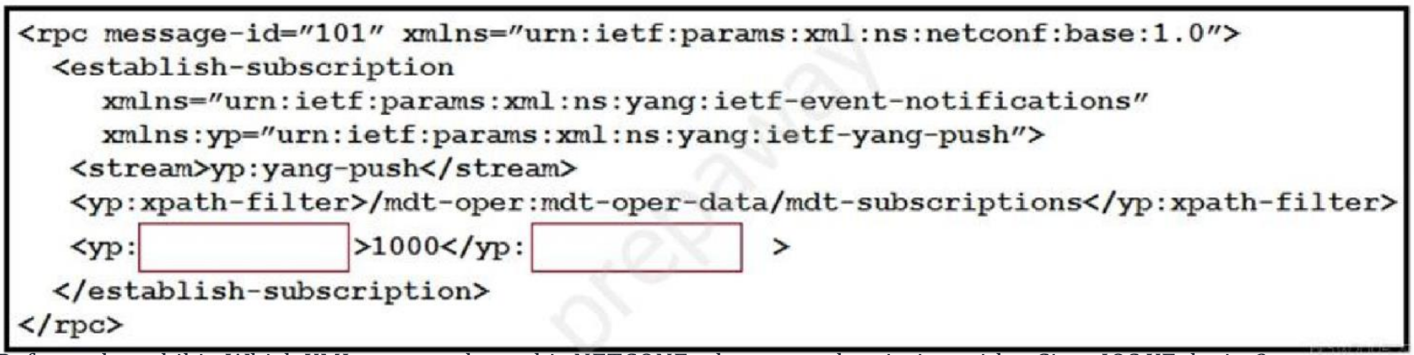
Answer Area

```
- name: configure interface settings
  ios_config:
     lines:
         -ip address 172.31.1.1 255.255.255.0
         -no shutdown
    parents: interface GigabitEthernet1/0
```

Explanation:

Reference:
http://imxing.cn/?p=464

Question: 32



```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
     xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
     xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
   <stream>yp:yang-push</stream>
   <yp:xpath-filter>/mdt-oper:mdt-oper-data/mdt-subscriptions</yp:xpath-filter>
   <yp:            >1000</yp:            >
  </establish-subscription>
</rpc>
```

Refer to the exhibit. Which XML tag completes this NETCONF telemetry subscription with a Cisco IOS XE device?

A.crontab
B.cadence
C.frequency
D.period

Answer: D

Explanation:

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

Question: 33

Which two statements are benefits of YANG-push telemetry data over traditional data collection methods? (Choose two.)

A.The subscription requests use less bandwidth than SNMP polls. B.It uses UDP rather than TCP.

C.You can precisely define data subscriptions.
D.It scales better than SNMP.

E.It is supported on more devices than SNMP.

Answer: CD

Explanation:

Here's a detailed justification for why options C and D are the correct benefits of YANG-push telemetry over traditional methods like SNMP, while A, B, and E are incorrect:

Correct Answers (C and D):

C. You can precisely define data subscriptions: YANG-push allows for granular control over the specific data that is streamed. Instead of pulling broad sets of metrics like SNMP does, you define exactly which data points are needed using a YANG model's data tree and XPath or similar expressions. This precise subscription reduces the volume of unnecessary data, optimizing bandwidth and processing overhead. This capability is crucial for efficient and targeted network monitoring.

D. It scales better than SNMP: Traditional methods like SNMP rely on polling, where a central collector repeatedly asks each device for data. As the network grows, this polling approach becomes inefficient due to increased network traffic and processing load on both the collector and the devices. YANG-push, being event-driven, only sends updates when data changes. This minimizes network chatter and allows for handling large numbers of devices effectively, demonstrating superior scalability. It's aligned with principles of efficient resource utilization in cloud and network infrastructure.

Incorrect Answers (A, B, and E):

A. The subscription requests use less bandwidth than SNMP polls: While the data delivery in YANG-push often uses less bandwidth due to sending only change-based updates, the initial subscription requests themselves can be more complex, containing detailed YANG paths. This statement is therefore an oversimplification.

B. It uses UDP rather than TCP: YANG-push can use various transports, with gRPC over TCP being a common choice due to reliability and built-in security features. While UDP can be used, it is not a defining feature or a benefit in all scenarios. The protocol choice is implementation dependent.

E. It is supported on more devices than SNMP: SNMP is a very mature and widely supported protocol that is present across almost all network devices. While YANG-push adoption is growing, it is not as ubiquitous as SNMP. It is not a practical benefit to assume higher device support.

In Summary: YANG-push's advantages center on its ability to provide targeted and scalable telemetry. This aligns with cloud computing best practices of optimized resource usage and efficient monitoring of distributed environments. Its event-driven model contributes to less network traffic and lower processing demands, making it a superior choice for large-scale enterprise networks.

Authoritative Links:

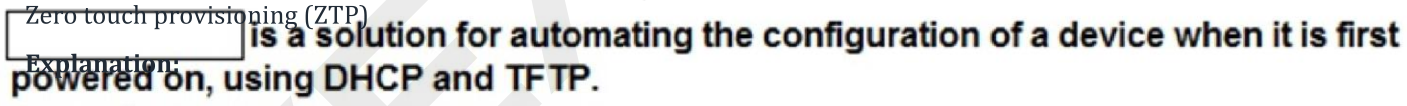
Cisco's guide on Model-Driven Telemetry:https://www.cisco.com/c/en/us/solutions/enterprise/white-paper-c11-741278.html - This document explains the principles and use cases of model-driven telemetry, focusing on its advantages for network automation.

IETF RFC 7950: The YANG 1.1 Data Modeling Language:https://datatracker.ietf.org/doc/html/rfc7950 - This is the authoritative source on the YANG data modeling language used in many model-driven telemetry solutions.

gRPC documentation:https://grpc.io/ - gRPC is commonly used as a transport mechanism for YANG-push, and this provides relevant information regarding its features.

Question: 34

FILL BLANK -
Fill in the blank to complete the statement.


[ ] is a solution for automating the configuration of a device when it is first powered on, using DHCP and TFTP.

Answer:

Zero touch provisioning (ZTP)

Explanation:

Reference:

https://developer.cisco.com/docs/ios-xe/#!day-zero-provisioning-quick-start-guide

Question: 35

Which tag is required when establishing a YANG-push subscription with a Cisco IOS XE device?

A.<yp:period>
B.<yp:subscription-result>
C.<yp:subscription-id>
D.<yp:xpath-filter>

Answer: D

Explanation:

The correct answer is **D. yp:xpath-filter**. When establishing a YANG-push subscription with a Cisco IOS XE device, the <yp:xpath-filter> tag is **mandatory** to specify the precise data nodes the subscriber wishes to receive. YANG-push, unlike traditional polling, pushes data updates only when changes occur to the specified data nodes. This minimizes resource utilization and enhances efficiency. Without an XPath filter, the device wouldn't know which data to stream, rendering the subscription ineffective. The <yp:xpath-filter> uses XPath syntax to pinpoint specific data points within the YANG data model. This granularity is crucial for targeted data collection and analysis. In contrast, tags such as <yp:period> define the reporting interval for periodic subscriptions (not relevant to change-driven YANG-push). The <yp:subscription-result> tag is typically part of the response and not the request for establishing a subscription and <yp:subscription-id> is assigned by the server after successful subscription setup, not used during initial request. Hence, while other elements may play roles in various aspects of subscription management, the <yp:xpath-filter> tag is specifically required during the initial establishment of a YANG-push subscription for filtering desired data.

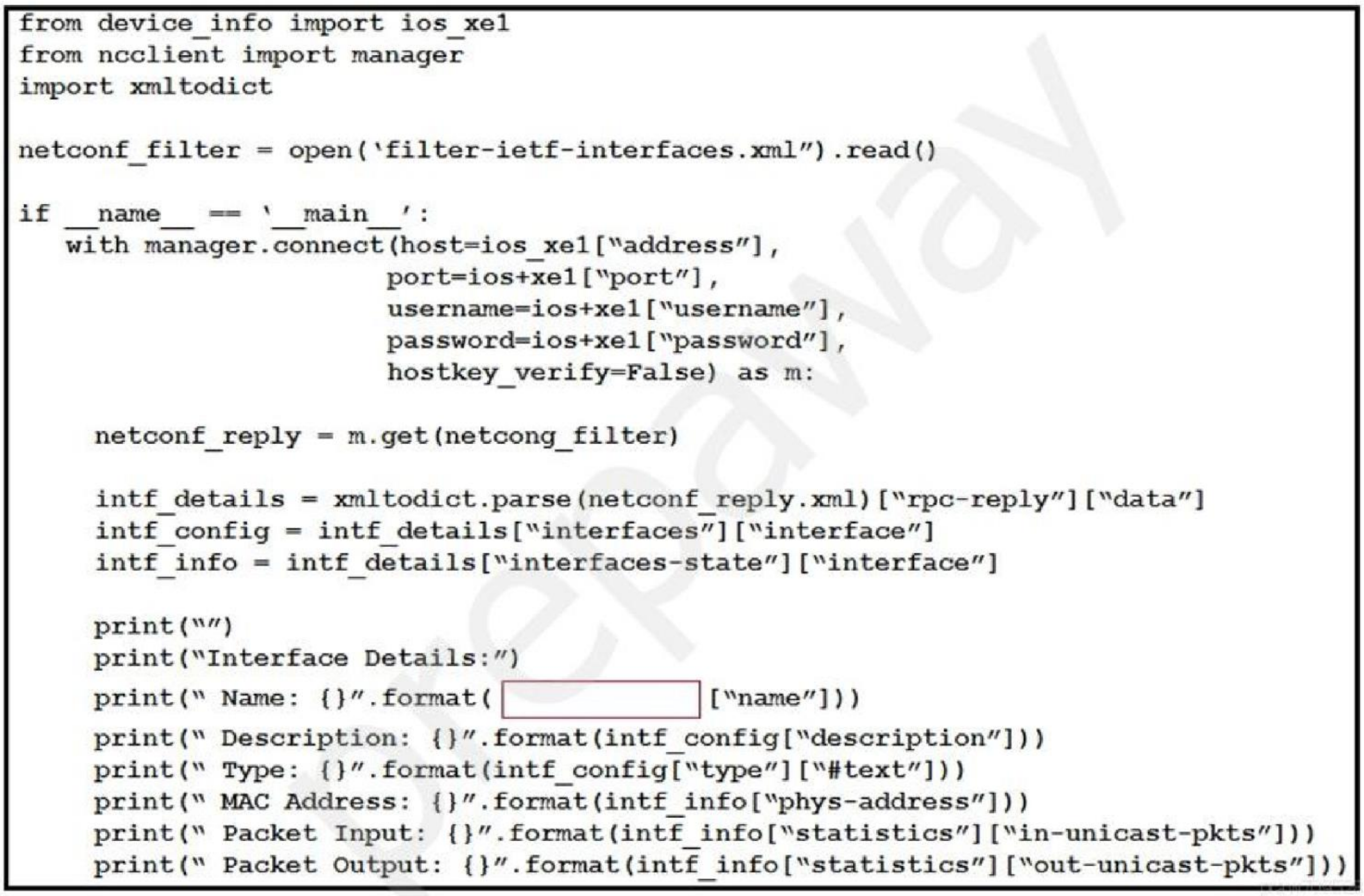
Authoritative Links:

Cisco YANG Data Model Programmability Guide:https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/16-12/b_16-12-programmability-config-guide/b_16-12-programmability-config-guide_chapter_0101.html This document provides a comprehensive understanding of YANG data modeling and its programmability aspects on Cisco IOS XE. Specifically look for sections concerning YANG-push, filters, and data streaming.

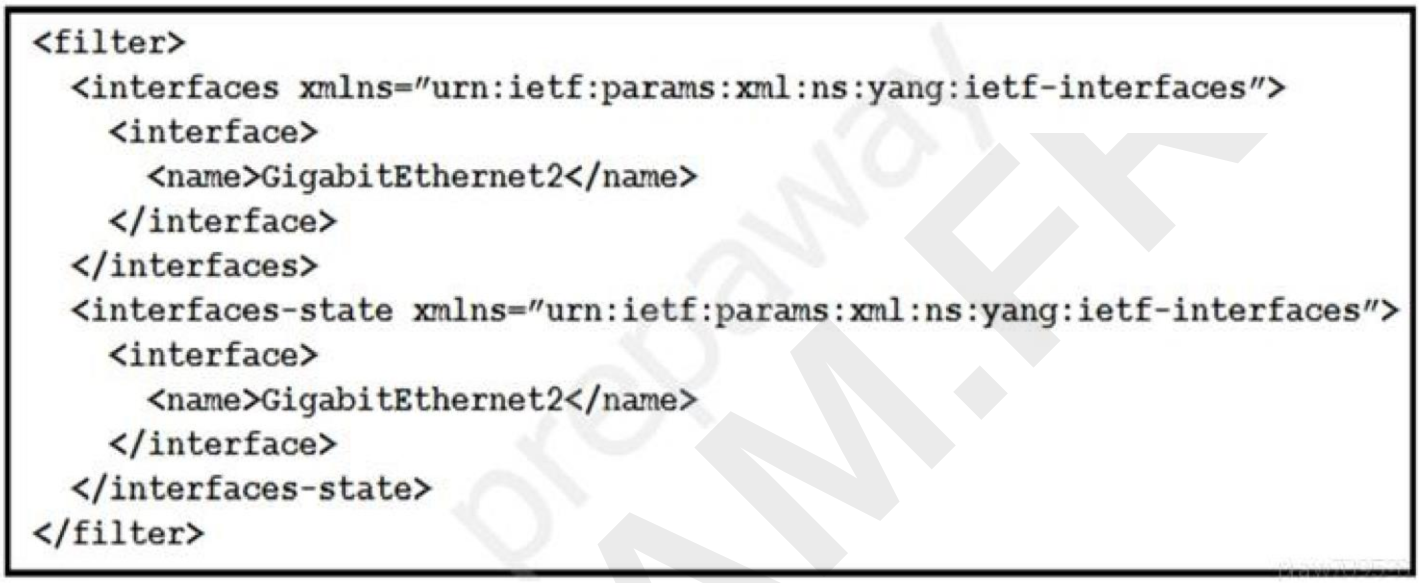
IETF RFC 7950 (The YANG 1.1 Data Modeling Language):https://datatracker.ietf.org/doc/html/rfc7950 This is the official RFC defining the YANG language. Reading specific sections on data node filtering within YANG push subscription would provide better understanding of the concepts.

IETF RFC 8641 (Subscription to YANG Notifications):https://datatracker.ietf.org/doc/html/rfc8641 Refer to this RFC for specifics on subscription mechanisms and their parameters in context of YANG-based streaming, and where xpath filters play a role.

Question: 36



```
from device_info import ios_xe1
from ncclient import manager
import xmltodict

netconf_filter = open('filter-ietf-interfaces.xml").read()

if __name__ == '__main__':
   with manager.connect(host=ios_xe1["address"],
                        port=ios+xe1["port"],
                        username=ios+xe1["username"],
                        password=ios+xe1["password"],
                        hostkey_verify=False) as m:

     netconf_reply = m.get(netcong_filter)

     intf_details = xmltodict.parse(netconf_reply.xml)["rpc-reply"]["data"]
     intf_config = intf_details["interfaces"]["interface"]
     intf_info = intf_details["interfaces-state"]["interface"]

     print("")
     print("Interface Details:")
     print(" Name: {}".format(             ["name"]))
     print(" Description: {}".format(intf_config["description"]))
     print(" Type: {}".format(intf_config["type"]["#text"]))
     print(" MAC Address: {}".format(intf_info["phys-address"]))
     print(" Packet Input: {}".format(intf_info["statistics"]["in-unicast-pkts"]))
     print(" Packet Output: {}".format(intf_info["statistics"]["out-unicast-pkts"]))
```



```
<filter>
  <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    <interface>
      <name>GigabitEthernet2</name>
    </interface>
  </interfaces>
  <interfaces-state xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    <interface>
      <name>GigabitEthernet2</name>
    </interface>
  </interfaces-state>
</filter>
```

Refer to the exhibits. An engineer creates a Python scripts using ncclient to display interface information. The code must be completed so that it can be tested.
Which expression completes the highlighted section in the format call?

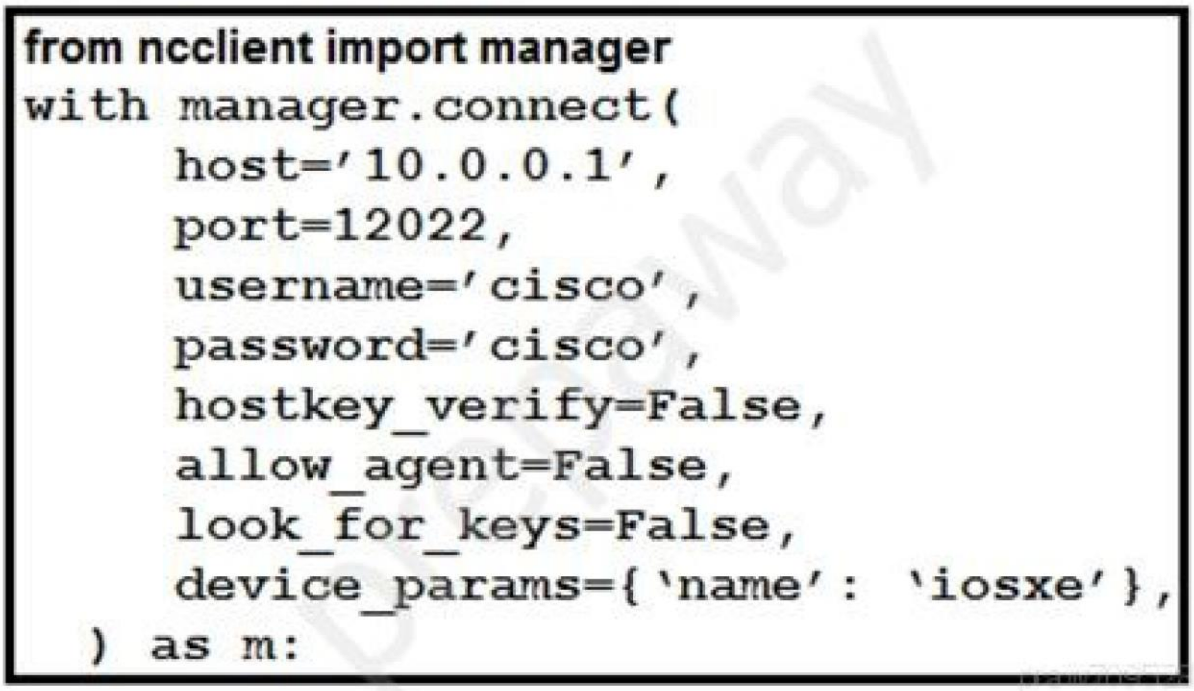
A.intf_info
B.intf_config
C.intf_get
D.intf_config[0]

Answer: B

Explanation:

Correct answer is B:intf_config.

Question: 37



```
from ncclient import manager
with manager.connect(
      host='10.0.0.1',
      port=12022,
      username='cisco',
      password='cisco',
      hostkey_verify=False,
      allow_agent=False,
      look_for_keys=False,
      device_params={'name': 'iosxe'},
  ) as m:
```

Refer to the exhibit. What is the correct ncclient method to use to collect the running configuration of a Cisco IOS XE device that uses NETCONF?

A.config=m.copy_config(source='running')
B.config=m.get(source='running')
C.config=m.collect_config(source='running')
D.config=m.get_config(source='running')

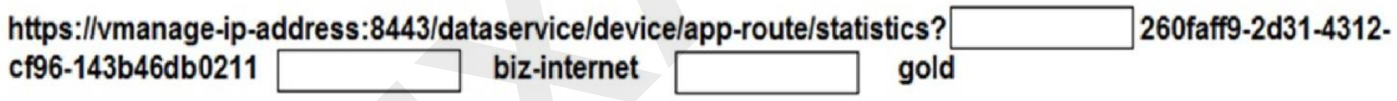
Answer: D

Explanation:

config=m.get_config(source='running').

Question: 38

FILL BLANK -
Fill in the blanks to complete this API request against the Cisco SD_WAN vManage Statistics API, which specified a deviceId of 260faff9-2d31-4312-cf96-
143b46db0211, a local-color of biz-internet, and a remote-color of gold.



https://vmanage-ip-address:8443/dataservice/device/app-route/statistics? ______ 260faff9-2d31-4312-cf96-143b46db0211 ______ biz-internet ______ gold

Answer:

deviceID=, local-color, remote-color

Explanation:

Reference:
https://sdwan-docs.cisco.com/Product_Documentation/Command_Reference/Command_Reference/vManage_REST_APIs/Real-Time_Monitoring_APIs/
Application-Aware_Routing#Statistics

Question: 39

What does the command boot ipxe forever switch 1 perform when executed on a Cisco IOS XE device?

A.It continuously sends DHCP requests for iPXE until the device boots with an image.
B.It continuously sends DNS requests for iPXE until the device restarts.
C.It continuously sends DNS requests for iPXE until the device boots with an image.
D.It continuously sends DHCP requests for iPXE until the device restarts.

Answer: A

Explanation:

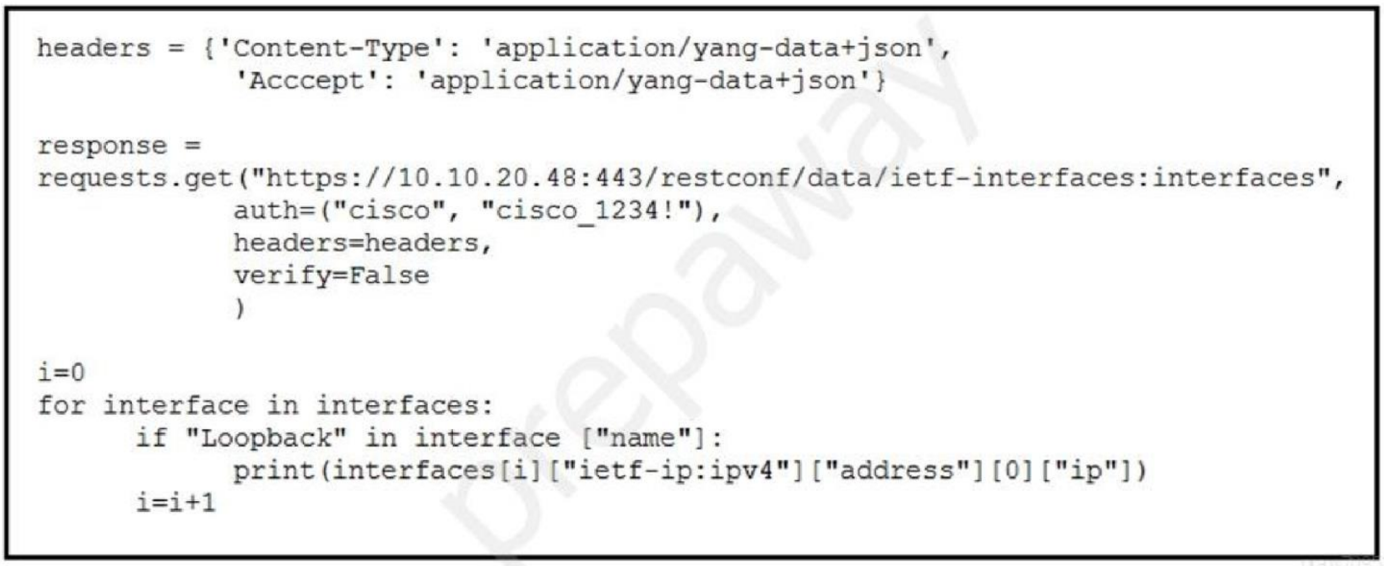
The command boot ipxe forever switch 1 on a Cisco IOS XE device initiates a continuous loop of iPXE boot attempts using network interface 'switch 1'. iPXE (Preboot Execution Environment) is a network bootloader that allows a device to load its operating system over the network rather than from local storage. When boot ipxe is invoked, the device first attempts to obtain an IP address via DHCP (Dynamic Host Configuration Protocol). The 'forever' keyword specifies that this process should continue indefinitely. The 'switch 1' part indicates the specific network interface used for the iPXE boot process. Therefore, the command results in a persistent cycle of the device attempting to retrieve DHCP information to subsequently initiate the iPXE process and boot from an image. The process involves DHCP discovery, offer, request and ack. It continuously sends out DHCP requests. If an IP is received, it then attempts the iPXE boot process and the cycle repeats.

Option A aligns perfectly with this functionality, as it describes the process of continuously sending DHCP requests for iPXE until the device boots from a provided image over the network. The device is not rebooted as such, it remains up and is trying to achieve a successful network boot. Options B and C are incorrect because iPXE primarily relies on DHCP for IP assignment, not DNS, during its initial startup. Option D is incorrect because the device is not rebooted when it fails; it instead restarts the iPXE boot process from a failed DHCP request cycle. This persistent network boot attempt facilitates zero-touch provisioning and remote device management.

Further research can be done via the following links:

Cisco IOS XE Configuration Guides:https://www.cisco.com/c/en/us/support/ios-xml/index.html (Search for 'iPXE boot' or 'network boot' within the relevant IOS XE release documentation.)
iPXE Project:https://ipxe.org/
DHCP Protocol RFC:https://datatracker.ietf.org/doc/html/rfc2131

Question: 40



```
headers = {'Content-Type': 'application/yang-data+json',
           'Acccept': 'application/yang-data+json'}

response =
requests.get("https://10.10.20.48:443/restconf/data/ietf-interfaces:interfaces",
           auth=("cisco", "cisco_1234!"),
           headers=headers,
           verify=False
           )

i=0
for interface in interfaces:
     if "Loopback" in interface ["name"]:
           print(interfaces[i]["ietf-ip:ipv4"]["address"][0]["ip"])
     i=i+1
```

Refer to the exhibit. A Python script is used to configure a Cisco IOS XE device. The script must be updated to print the IP addresses of all the loopback interfaces. Which statement should be added before the loop?

A.interfaces = response.json()[ietf-interfaces:interfaces]
B.interface = response.json()[ietf-interfaces:interfaces]
C.interface = response.json()[ietf-interfaces:interfaces][interface]
D.interfaces = response.json()[ietf-interfaces:interfaces][interface]

Answer: D

Explanation:

Reference:
https://blog.wimwauters.com/networkprogrammability/2020-04-04_restconf_python/

Question: 41

Which environment must be enabled to complete the Zero-Touch Provisioning process on a Cisco IOS XE device?

A.TCL
B.ZTP Open Service Container
C.EEM
D.Guest Shell

Answer: D

Explanation:

The correct answer is **D. Guest Shell**. Zero-Touch Provisioning (ZTP) on Cisco IOS XE devices relies on a lightweight, containerized environment known as the Guest Shell. This environment provides the necessary execution space to run scripts and software required for automated configuration. Specifically, upon initial boot, the Cisco device attempts to discover a DHCP server offering option 43, which can contain a URL pointing to a configuration script. The Guest Shell, being a Linux-based virtual environment within the IOS XE device, executes this script. The script usually involves retrieving further configuration files, software images, or performing specific device setups. This automation eliminates the need for manual configuration of each device, promoting efficient and scalable deployments. The Guest Shell's Linux-based architecture supports commonly used scripting languages (e.g., Python) enabling customizable automation routines. Options A, B, and C do not directly enable ZTP. TCL is a scripting language, but not the primary enabler for this process.

ZTP Open Service Container and EEM are related to other automation aspects but not the foundational environment for ZTP. The guest shell provides the flexibility and programmability required for ZTP to effectively operate within Cisco IOS XE.

Relevant link:

Cisco documentation on Guest Shell: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/169/b_169_programmability_config_guide/guest_shell.html

Question: 42

FILL BLANK -
Fill in the blank to complete the statement.
________________ is a solution for automating the configuration of a device when it is first powered on, using DHCP and TFTP.

Answer:

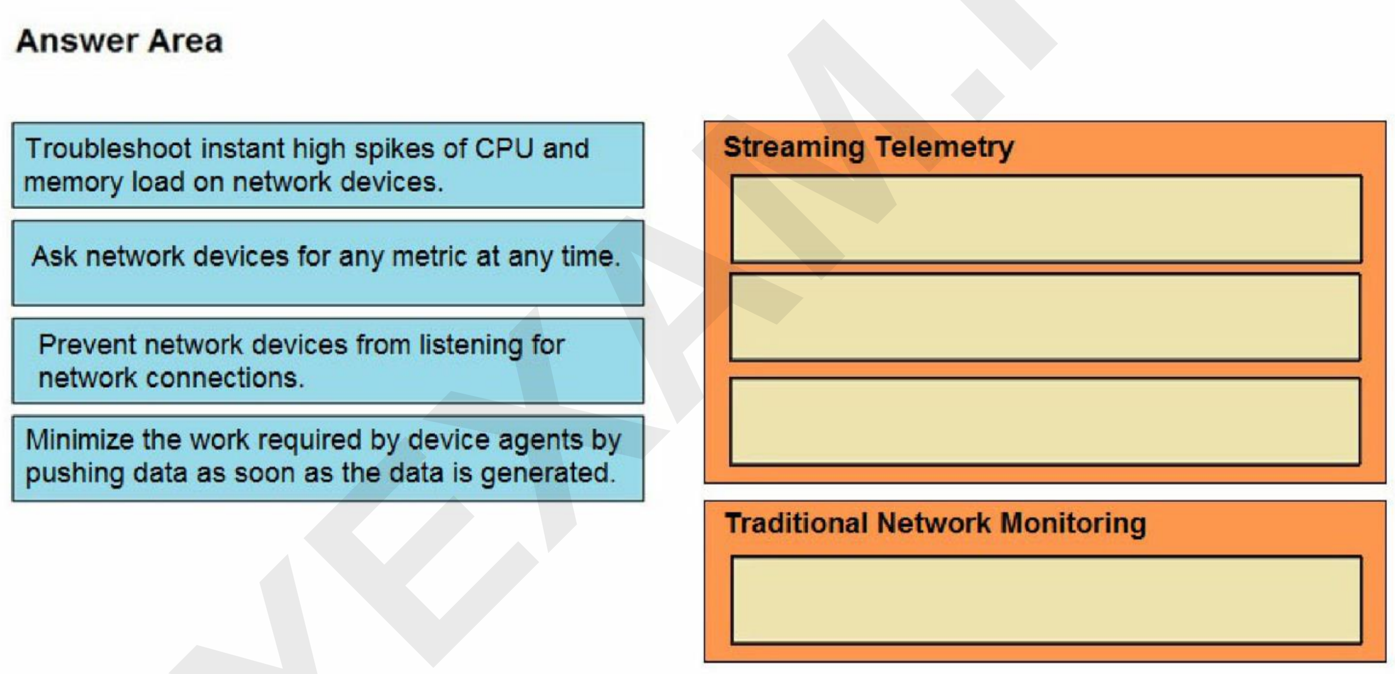
Zero-touch provisioning

Explanation:

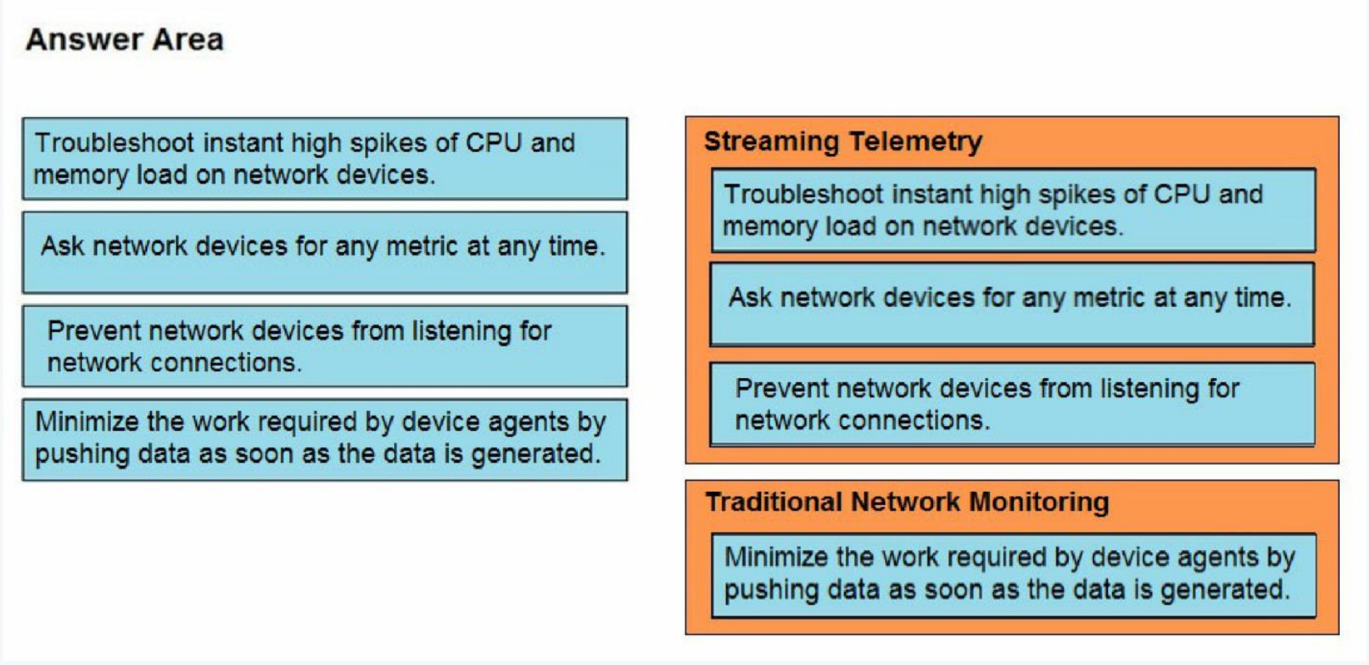
Reference:
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/169/b_169_programmability_cg/zero_touch_provisioning.html

Question: 43

DRAG DROP -
Drag and drop the characteristic from the left onto the monitoring type described on the right.
Select and Place:


Answer Area
Troubleshoot instant high spikes of CPU and memory load on network devices.
Ask network devices for any metric at any time.
Prevent network devices from listening for network connections.
Minimize the work required by device agents by pushing data as soon as the data is generated.
Streaming Telemetry
Traditional Network Monitoring

Answer:


Answer Area

Troubleshoot instant high spikes of CPU and memory load on network devices.

Ask network devices for any metric at any time.

Prevent network devices from listening for network connections.

Minimize the work required by device agents by pushing data as soon as the data is generated.

Streaming Telemetry

Troubleshoot instant high spikes of CPU and memory load on network devices.

Ask network devices for any metric at any time.

Prevent network devices from listening for network connections.

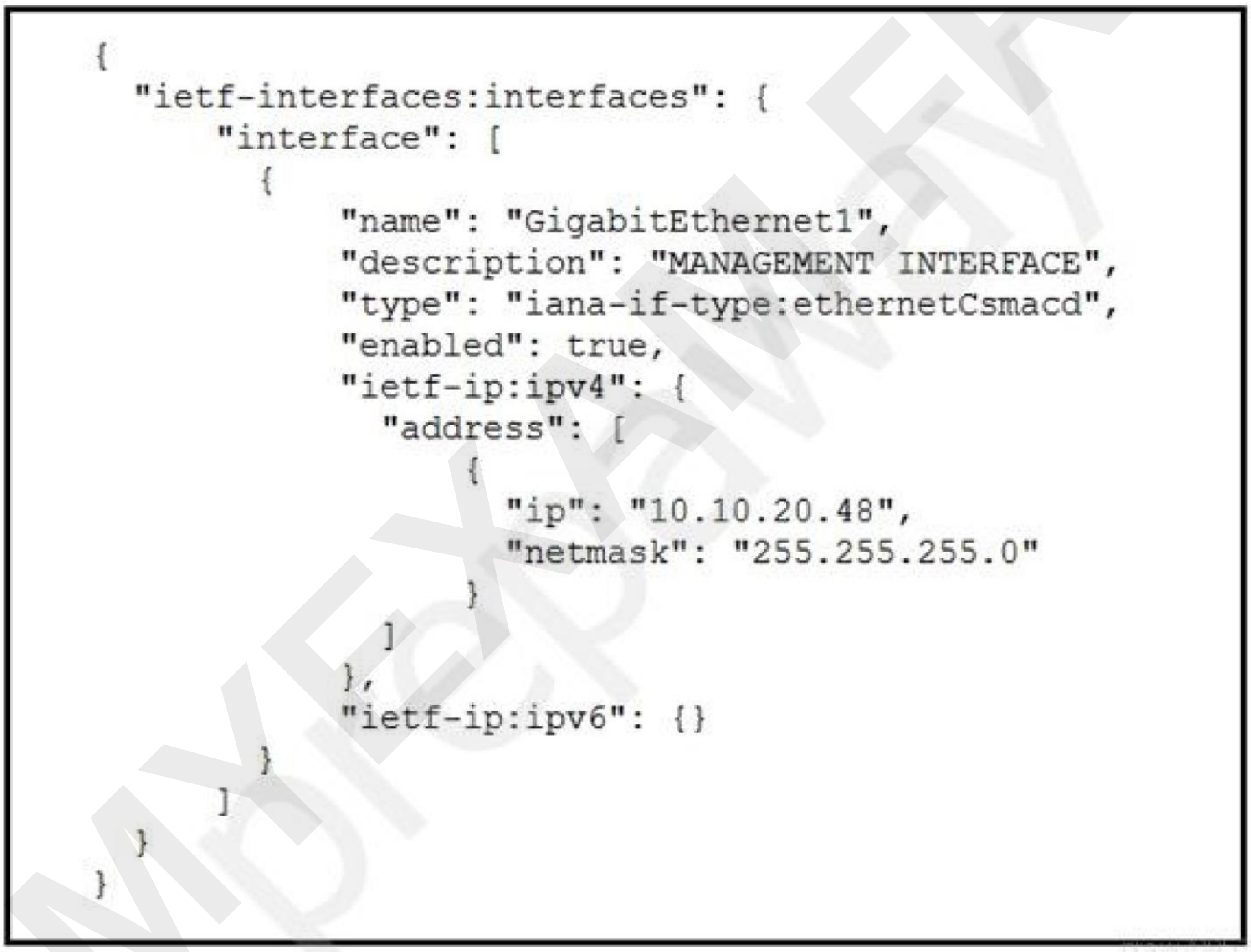
Traditional Network Monitoring

Minimize the work required by device agents by pushing data as soon as the data is generated.

Explanation:

Reference:
https://www.cisco.com/c/en/us/td/docs/iosxr/ncs5500/telemetry/70x/b-telemetry-cg-ncs5500-70x/b-teleme try-cg-ncs5500-70x_chapter_010.html

Question: 44



```
{
  "ietf-interfaces:interfaces": {
      "interface": [
        {
            "name": "GigabitEthernet1",
            "description": "MANAGEMENT INTERFACE",
            "type": "iana-if-type:ethernetCsmacd",
            "enabled": true,
            "ietf-ip:ipv4": {
              "address": [
                  {
                    "ip": "10.10.20.48",
                    "netmask": "255.255.255.0"
                  }
              ]
            },
            "ietf-ip:ipv6": {}
        }
      ]
    }
}
```

Refer to the exhibit. A RESTCONF GET request is sent to a Cisco IOS XE device. A portion of the response is shown in the exhibit. Which module name corresponds to the YANG model referenced in the request?

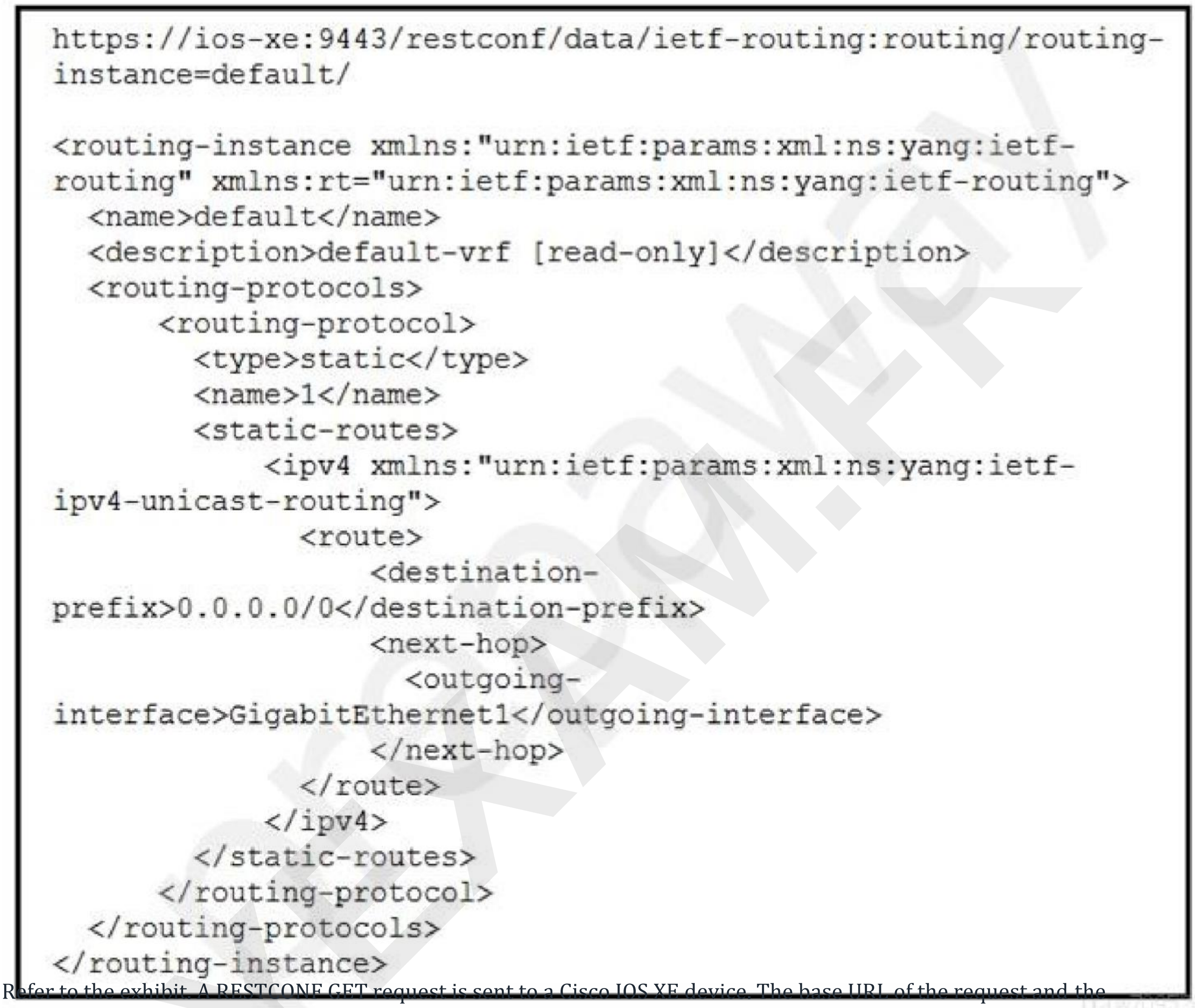
A.ietf-interfaces:ietf-ipv4
B.iana-if-type:ethernetCsmacd
C.ietf-interfaces:interfaces
D.ietf-interfaces

Answer: D

Explanation:

Dietf-interfaces > MODULE:interfaces > CONTAINERietf-ipv > MODULE Through augmentation, the ietf-interfaces model is "augmented" by the ietf-ip model to add IPv4.

Question: 45



```
https://ios-xe:9443/restconf/data/ietf-routing:routing/routing-
instance=default/

<routing-instance xmlns:"urn:ietf:params:xml:ns:yang:ietf-
routing" xmlns:rt="urn:ietf:params:xml:ns:yang:ietf-routing">
  <name>default</name>
  <description>default-vrf [read-only]</description>
  <routing-protocols>
      <routing-protocol>
        <type>static</type>
        <name>1</name>
        <static-routes>
            <ipv4 xmlns:"urn:ietf:params:xml:ns:yang:ietf-
ipv4-unicast-routing">
              <route>
                  <destination-
prefix>0.0.0.0/0</destination-prefix>
                  <next-hop>
                    <outgoing-
interface>GigabitEthernet1</outgoing-interface>
                  </next-hop>
              </route>
            </ipv4>
        </static-routes>
      </routing-protocol>
  </routing-protocols>
</routing-instance>
```

Refer to the exhibit. A RESTCONF GET request is sent to a Cisco IOS XE device. The base URL of the request and the response in XML format are shown in the exhibit. What is the YANG data node that is referenced in the response?

A.route is a leaf list
B.static-routes is a container

C.static-routes is a list
D.routing-instance is a container

Answer: B

Explanation:

static-routes is a container.