

## 300-635 Dumps

# Automating and Programming Cisco Data Center Solutions (DCAUTO)

<https://www.certleader.com/300-635-dumps.html>



**NEW QUESTION 1****DRAG DROP**

A file named myfunc.py has been edited. Drag and drop the steps from the left that ensure that this file is committed to the local Git repository and verify its status into the correct order on the right. Not all options are used.

Select and Place:

- A. Mastered
- B. Not Mastered

**Answer:** A

**NEW QUESTION 2****DRAG DROP**

Drag and drop the code to complete an Ansible playbook that creates a new tenant. Not all options are used.

Select and Place:

- A. Mastered
- B. Not Mastered

**Answer:** A

**NEW QUESTION 3**

Refer to the exhibit.

```
mo_dir = cobra.mit.access.MoDirectory(cobra.mit.session.LoginSession(apic_url, username, password))
mo_dir.login()
cq = cobra.mit.access.ClassQuery('fvCEP')
cq.subtree = 'full'
objlist = mo_dir.query(cq)
for mo in objlist:
    print "MAC: " + mo.mac + "|" + "IP: " + mo.ip
```

Which action does the execution of this ACI Cobra Python code perform?

- A. It prints all LLDP neighbor MAC and IP addresses.
- B. It prints all Cisco Discovery Protocol neighbor MAC and IP addresses.
- C. It prints all endpoint MAC and IP addresses.
- D. It prints all APIC MAC and IP addresses.

**Answer:** C

**NEW QUESTION 4**

Refer to the exhibit.

```
def add_tenant():
    token = apic_login.aaaLogin()
    for tenant in range(1,10):
        try:
            response = requests.post(
                url=constant.APIC_URL + "/api/node/mo/uni/tn-exam%s.json" % (tenant),
                headers={
                    "Cookie": "APIC-cookie=" + token,
                    "Content-Type": "application/json; charset=utf-8",
                },
                data=json.dumps({
                    "fvTenant": {
                        "attributes": {
                            "status": "created",
                            "dn": "uni/tn-exam%s" % (tenant),
                            "name": "exam%s" % (tenant),
                            "rn": "tn-exam%s" % (tenant)
                        },
                        "children": [
                        ]
                    }
                })
            )
            print("Response HTTP Status Code: {status_code}".format(
                status_code=response.status_code))
            print("Response HTTP Response Body: {content}".format(
                content=response.content))
        except requests.exceptions.RequestException:
            print("HTTP Request failed")
    add_tenant()
```

Assuming a new ACI instance, what is the result when this script is run?

- A. Ten objects are created and subsequently deleted.

- B. Nine objects are created.
- C. An exception is thrown.
- D. Ten objects are created.

Answer: D

#### NEW QUESTION 5

```
import requests

USER = "admin"
PASS = "password"
APIC = "https://apic.supereats.com"

OPERATION = 'api/aaaLogin.json'
DATA = {"aaaUser": {"attributes": {"name": USER, "pwd": PASS}}}
RESPONSE = requests.post(APIC+OPERATION, json=DATA, verify=False)

TOKEN = RESPONSE.json()["imdata"][0]["aaaLogin"]["attributes"]["token"]
COOKIE = {'APIC-cookie': TOKEN}

OPERATION = 'api/aaaLogout.json'
DATA = {
    "aaaLogout": {
        "attributes": {
            "token": TOKEN
        }
    }
}
RESPONSE = requests.post(APIC+OPERATION, json=DATA, cookies=COOKIE, verify=False)
```

Which Python snippets create an application policy named OrderProcess that contains two application endpoint groups under Tenant SuperEats using direct calls to the ACI REST API? Assume that authentication and library imports are correct. A.

```
OPERATION = 'api/node/mo/uni.json'
DATA = {
    "FVTenant": {"attributes": {"name": "SuperEats"},
    "children": [{"FVAp": {"attributes": {"name": "OrderProcess"},
        "children": [
            {"FVAEPg": {"attributes": {"name": "app"}}},
            {"FVAEPg": {"attributes": {"name": "web"}}}
        ]
    }
    ]}
}
RESPONSE = requests.post(APIC+OPERATION, json=DATA, cookies=COOKIE)
```

A.

```
OPERATION = 'api/node/mo/uni.json'
DATA = {
    "fvTenant": {"attributes": {"name": "SuperEats"},
    "children": [{"fvAp": {"attributes": {"name": "OrderProcess"},
        "children": [
            {"fvAEPg": {"attributes": {"name": "app"}}},
            {"fvAEPg": {"attributes": {"name": "web"}}}
        ]
    }
    ]}
}
RESPONSE = requests.get(APIC+OPERATION, cookies=COOKIE)
```

B.

```
OPERATION = 'api/node/mo/uni.json'
DATA = {
    "fvTenant": {"attributes": {"rn": "SuperEats"},
    "children": [{"fvAp": {"attributes": {"rn": "OrderProcess"},
        "children": [
            {"fvAEPg": {"attributes": {"rn": "app"}}},
            {"fvAEPg": {"attributes": {"rn": "web"}}}
        ]
    }
    ]}
}
RESPONSE = requests.post(APIC+OPERATION, json=DATA, cookies=COOKIE)
```

C.

```
OPERATION = 'api/node/mo/uni.json'
DATA = {
    "fvTenant": {"attributes": {"name": "SuperEats"},
    "children": [{"fvAp": {"attributes": {"name": "OrderProcess"},
        "children": [
            {"fvAEPg": {"attributes": {"name": "app"}}},
            {"fvAEPg": {"attributes": {"name": "web"}}}
        ]
    }
    ]}
}
RESPONSE = requests.post(APIC+OPERATION, json=DATA, cookies=COOKIE)
```

**Answer:** D

#### NEW QUESTION 6

DRAG DROP

Drag and drop the correct code snippets into the Python code to create a new application profile "WebApp" using the ACI REST API. Not all options are used. Select and Place:

- A. Mastered
- B. Not Mastered

**Answer:** A

#### NEW QUESTION 7

DRAG DROP

Drag and drop the correct YAML components from the bottom onto the correct blanks within the Ansible playbook to create a new application profile called "DbApp" using the Ansible ACI module. Not all options are used. Select and Place:

- A. Mastered
- B. Not Mastered

**Answer:** A

#### NEW QUESTION 8

Refer to the exhibit.

```
from acitoolkit.acitoolkit import (
    AppProfile, BridgeDomain, Context,
    EPG, Session, Subnet, Tenant
)

def create_tenant():
    session = Session(
        "https://apic", "admin", "ciscopsdt"
    )
    session.login()
    my_tenant = Tenant("DevNet_Tenant")
    my_vrf = Context("DevNet_VRF", my_tenant)
    my_bd = BridgeDomain("DevNet_BD", my_tenant)
    my_bd.add_context(my_vrf)
    my_subnet = Subnet("DevNet_Subnet", my_bd)
    my_subnet.set_scope("public")
    my_subnet.set_addr("10.10.10.1/24")
    my_app = AppProfile("DevNet_App", my_tenant)
    my_epg = EPG("DevNet_EPG", my_app)
    my_epg.add_bd(my_bd)
    session.push_to_apic(
        my_tenant.get_url(),
        my_tenant.get_json()
    )

if __name__ == '__main__':
    create_tenant()
```

Which two actions does this Python code perform with the Cisco ACI? (Choose two.)

- A. It creates a subnet "DevNet\_Subnet" inside VRF "DevNet\_VRF" located in ACI tenant "DevNet\_Tenant" and sets the scope to "private".
- B. It creates a subnet "DevNet\_Subnet" inside AppProfile "DevNet\_App" located in ACI tenant "DevNet\_Tenant" and sets the network address to "10.10.10.1/24".
- C. It creates an EPG "DevNet\_EPG" inside AppProfile "DevNet\_App" located in ACI tenant "DevNet\_Tenant" and link the EPG with BridgeDomain "DevNet\_BD".
- D. It creates a subnet "DevNet\_Subnet" inside VRF "DevNet\_VRF" located in ACI tenant "DevNet\_Tenant" and sets the network address to "10.10.10.1/24".
- E. It creates an EPG "DevNet\_EPG" inside VRF "DevNet\_VRF" located in ACI tenant "DevNet\_Tenant" and link the EPG with BridgeDomain "DevNet\_BD".

**Answer:** CD

#### NEW QUESTION 9

What is a key characteristic of an ACI policy model?



- A. Logical and concrete domains are separated.
- B. All configuration is carried out against concrete entities.
- C. It allows communications with newly connected devices.
- D. Network administrators configure logical and physical system resources directly.

**Answer:** A

#### NEW QUESTION 10

Refer to the exhibit.

**Switch configuration**

```
!Command: show running-config
!
feature hsrp
!
ip access-list allow_http_traffic
 10 permit tcp any any eq www
!
vrf context management
 ip route 0.0.0.0/0 192.168.151.2
!
interface mgmt0
 ip address 192.168.251.129 255.255.255.0
 vrf member management
```

**Ansible playbook**

```
---
- name: Vlan Provisioning
  hosts: nxos
  gather_facts: no

  vars:
    nxos_provider:
      username: "{{ un }}"
      password: "{{ pwd }}"
      transport: nxapi
      host: "{{ inventory_hostname }}"

  tasks:

    - name: CREATE VLANS AND ASSIGN A NAME, USING VLAN_ID
      nxos_vlan:
        vlan_id: "{{ item.vlan_id }}"
        name: "{{ item.name }}"
        provider: "{{ nxos_provider }}"
      with_items:
        - vlan_id: 2
          name: Native
        - vlan_id: 15
          name: Web
        - vlan_id: 20
          name: App
        - vlan_id: 30
          name: DB
```

**Playbook output**

```
$ ansible-playbook playbook.yml

PLAY [Vlan Provisioning]*****
*****

TASK [CREATE VLANS AND ASSIGN A NAME, USING VLAD_ID]*****
*****
failed: [192.168.251.129] (item={'vlan_id': 2, 'name': 'Native'}) => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "ansible_loop_var": "item", "changed": false, "item": {"name": "Native", "vlan_id": 2}, "msg": "Request failed: <urlopen error [Errno 61] Connection 'refused>"
"status": -1, "url": "http://192.168.251.129:80/ins")
```

The exhibit shows a Cisco NX-OS switch configuration, an Ansible playbook, and the output of running this playbook. The playbook failed due to error "msg" 'Request failed <urlopen error [Errno 61] Connection refused>', 'status' -1, "url" "http://192.168.251.129:80/ins".

Which Cisco NX-OS configuration command resolves this failure?

- A. feature nxapi
- B. http-server enabled
- C. interface mgmt0; ip access-group allow\_http\_traffic in
- D. feature http

**Answer:** A

#### NEW QUESTION 10

DRAG DROP

After a Cisco Nexus switch interface is enabled and in the up state, an engineer must automate the configuration of the interface descriptions using the EEM Python Module. Drag and drop the steps the engineer must take from the left into the correct order on the right.

Select and Place:

|  |        |
|--|--------|
| event syslog pattern "IF_UP"                     | step 1 |
| exit   | step 2 |
| event manager applet link monitor                | step 3 |
| conf t   | step 4 |
| action 1 cli command "source cdp_description.py" | step 5 |

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

|  |  |
|--|--|
| event syslog pattern "IF_UP"                     | conf t   |
| exit   | event manager applet link monitor                |
| event manager applet link monitor                | event syslog pattern "IF_UP"                     |
| conf t   | action 1 cli command "source cdp_description.py" |
| action 1 cli command "source cdp_description.py" | exit   |

#### NEW QUESTION 14

What are two capabilities of the DCNM REST API? (Choose two.)

- A. uses the dcnm-token header for requests authentication after initial basic authentication
- B. uses basic authentication without encoding for username and password
- C. supports HTTP until release 11.0(1)
- D. uses bearer key authorization
- E. is separated into Classic LAN, LAN Fabric, Media Controller, and SAN Management categories

**Answer:** AE

#### NEW QUESTION 19

Which two capabilities apply to the DCNM API? (Choose two.)

- A. DCNM provides an XML-based SOAP API.
- B. DCNM requires a license to use the API.
- C. Some features of DCNM must be configured through the GUI.
- D. All API operations can be performed using the DCNM GUI.
- E. DCNM provides a REST-based API.

**Answer:** AE

#### NEW QUESTION 23

Refer to the exhibit.

```
Dn
--
sys/chassis-4/blade-1
sys/chassis-4/blade-3
sys/chassis-4/blade-5
sys/chassis-4/blade-7
sys/chassis-5/blade-1
```

Which two Cisco UCS PowerTool commands provide this output? (Choose two.)

- A. Get-UcsServer | Select-Object Dn
- B. Get-UcsRack Systems | Select-Object Dn
- C. Get-UcsBlade | Select-Object Dn
- D. Get-UcsRackUnit | Select-Object Dn
- E. Get-UcsSystems | Select-Object Dn

**Answer:** AC

#### NEW QUESTION 24

What is a description of a Cisco UCS Director script module?

- A. function to convert internal workflow tasks into Python scripts
- B. place to store custom workflow scripts, jars, and custom lists of values for use in custom workflow tasks
- C. place to store external scripts that are not related to Cisco UCS Director
- D. place to store imported scripts, Bash, and custom Python code for use in custom workflow tasks

**Answer:** B

#### NEW QUESTION 25

Which programming language are the Cisco UCS Director custom workflow tasks written in?

- A. C++
- B. Python
- C. Java
- D. Cloupia Script

**Answer:** C

**NEW QUESTION 28**

DRAG DROP

Drag and drop the items to complete the request to retrieve the current firmware of Cisco UCS devices from the Cisco Intersight API. Not all items are used.  
Select and Place:

- A. Mastered
- B. Not Mastered

**Answer:** A

**NEW QUESTION 30**

DRAG DROP

A co-worker is using Cisco Intersight to determine the maximum available memory per server for their company's data center. Drag and drop the code to complete the Cisco Intersight API call that provides the desired results. Not all options are used.  
Select and Place:

- A. Mastered
- B. Not Mastered

**Answer:** A

**NEW QUESTION 32**

.....

## Thank You for Trying Our Product

\* 100% Pass or Money Back

All our products come with a 90-day Money Back Guarantee.

\* One year free update

You can enjoy free update one year. 24x7 online support.

\* Trusted by Millions

We currently serve more than 30,000,000 customers.

\* Shop Securely

All transactions are protected by VeriSign!

**100% Pass Your 300-635 Exam with Our Prep Materials Via below:**

<https://www.certleader.com/300-635-dumps.html>