PassQueen
a.k.a dumpsqueen.com

Automating Cisco Collaboration Solutions (CLAUTO)

Cisco 300-835

Version Demo

Total Demo Questions: 14

Total Premium Questions: 145

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Cisco Collaboration APIs	125
Topic 2, Cisco Collaboration Infrastructure	4
Topic 3, Webex	16
Total	145

QUESTION NO: 1

Refer to the exhibit.

```
def test(definedArgument, *args, **kwargs):
    print (definedArgument)
    print (args)
    print (kwargs["test1"])
```

Which output is expected when this code runs?

test("USER",1,2,True,"Hello World",test1 ="Test Passed!",test2=2254)

A)

```
definedArgument
Args
Kwargs
```

B)

```
USER
1
Test Passed!
```

C)

```
USER
(1, 2, True, 'Hello World')
test1
```

D)

```
USER
(1, 2, True, 'Hello World')
Test Passed!
```

A. Option

B. Option

C. Option

D. ('USER', 1, 2, True, 'Hello World') {'test1': 'Test Passed!', 'test2': 2254}

ANSWER: D

Explanation:

The expected output is `('USER', 1, 2, True, 'Hello World')` followed by `{'test1': 'Test Passed!', 'test2': 2254}`. The function call supplies five positional arguments before any named arguments. When a function collects positional arguments with `*args`, Python places them in a tuple, retaining their supplied order and their original types. Therefore, the string `USER`, integers `1` and `2`, Boolean `True`, and the string `Hello World` appear as the tuple shown. The named arguments are collected by `**kwargs` into a dictionary. Their names become dictionary keys and their assigned values become the corresponding values, producing the `test1` and `test2` entries. Printing these variables displays their normal Python tuple and dictionary representations. This is the standard mechanism for accepting an arbitrary number of positional and keyword arguments in a Python function. Python documents argument unpacking and collection in its arbitrary argument lists tutorial section, and defines the relevant call behavior in the Python language reference for calls.

QUESTION NO: 2

Which two actions are required to collect Cisco Unified Communications Manager Perfmon counter data for a selected set of counters? (Choose two.)

A. Call perfmonOpenSession.

B. Call perfmonAddCounter for each required counter.

C. Call perfmonResetCounter before each collection.

D. Call perfmonListCounterData after every counter is added.

E. Call perfmonRestartService to activate the session.

ANSWER: A B

Explanation:

An application must first call `perfmonOpenSession` to establish a Perfmon monitoring session and obtain a session handle. It then uses `perfmonAddCounter` to register each required counter with that session. After the counters are registered, the application can call `perfmonCollectCounterData` to retrieve their values.

Opening a session and adding counters define the scope of the collection operation. The application should close the session when monitoring is complete to release server-side resources. Cisco documents the session-based Perfmon workflow in the Cisco Performance Monitoring API documentation.

QUESTION NO: 3

Which two application use cases can be developed using Cisco UCM CTI (TAPI/JTAPI)? (Choose two.)

A. collaborative file sharing tool

B. interactive voice response system

C. contact center solution

D. instant messaging client

E. calendaring application

ANSWER: B C

Explanation:

interactive voice response system and **contact center solution** are appropriate Cisco Unified Communications Manager CTI application use cases. Unified CM exposes telephony events and call-control functions through CTI interfaces, including JTAPI and TAPI. An application can monitor device and call state, answer or redirect calls, initiate calls, transfer calls, and receive notifications as calls progress. Those capabilities allow an interactive voice response application to programmatically answer an inbound call, play or coordinate prompts through associated media resources, collect caller input, and route the call according to application logic.

The same call-control and event model is fundamental to a contact center solution. A contact-center application needs to observe agent and endpoint status, track call lifecycle events, control call delivery, and apply routing or queueing decisions. CTI provides the integration layer that lets the application coordinate these telephony actions with its own customer, agent, and workflow data. Cisco documents JTAPI as an interface through which applications can control and monitor Unified CM calls and devices, and identifies CTI as a core component for integrating telephony applications. See the Cisco JTAPI Developer Guide and Cisco's Unified CM programming reference guides.

QUESTION NO: 4

Refer to the exhibit.

```
<?xml version="1.0"?>
<inboundDialPlanRules total="3">
     <inboundDialPlanRule id="eedeeca5-c73b-441e-bc5a-71d9b256180b">
          <domain>cms.heleus.lab</domain>
          <priority>100</priority>
     </inboundDialPlanRule>
     <inboundDialPlanRule id="647511a0-0c7d-484f-8796-ee52dd3128f1">
          <domain>space.heleus.lab</domain>
          <priority>2</priority>
     </inboundDialPlanRule>
     <inboundDialPlanRule id="2c5662ab-7e5a-45a8-a133-5f4d6c6cb509">
          <domain>cms.lab</domain>
          <priority>1</priority>
     </inboundDialPlanRule>
</inboundDialPlanRules>
```

Which API call made to Cisco Meeting Server removes the inbound dial plan rule matching cms.lab domain?

A. DELETE to /api/v1/inboundDialPalnRules/ with body attribute id="2c5662ab-7e5a-45a8-a133- 5f4d6c6cb509".

B. DELETE to /api/v1/inboundDialPlanRules/2c5662ab-7e5a-45a8-a133-5f4d6c6cb509.

C. PUT to /api/v1/inboundDialPalnRules/2c5662ab-7e5a-45a8-a133-5f4d6c6cb509 with body attribute delete="True".

D. PUT to /api/v1/inboundDialPalnRules/2c5662ab-7e5a-45a8-a133-5f4d6c6cb509.

ANSWER: B

Explanation:

The correct API call is **DELETE to /api/v1/inboundDialPlanRules/2c5662ab-7e5a-45a8-a133-5f4d6c6cb509.** Cisco Meeting Server exposes inbound dial-plan rules as individual REST resources. The rule shown for the `cms.lab` domain is identified by the UUID `2c5662ab-7e5a-45a8-a133-5f4d6c6cb509`; therefore, the deletion request must address that specific resource by appending its identifier to the inbound dial-plan-rules collection URI. The HTTP `DELETE` method expresses the requested operation directly: removal of the identified configuration object. No request-body deletion attribute is required for this resource operation. The endpoint uses the camel-case collection name `inboundDialPlanRules`, so preserving that spelling is important when constructing the request. After Cisco Meeting Server accepts the request, the inbound dial plan rule represented by that UUID is removed and will no longer be evaluated for inbound calls. Cisco's Meeting Server programming references describe the REST API resource model and methods, while the product documentation index provides the applicable API reference for the deployed CMS release. See the Cisco Meeting Server Programming Reference Guides and the Cisco Meeting Server documentation page.

QUESTION NO: 5

After the AXL query ns:updatePhone is used to upgrade a phone configuration successfully, the phone does not reflect the change. Which other method must be performed for the change to take effect?

A. ns:getPhone

B. ns:restartPhone

C. ns:rebootPhone

D. ns:savePhone

E. ns:resetPhone

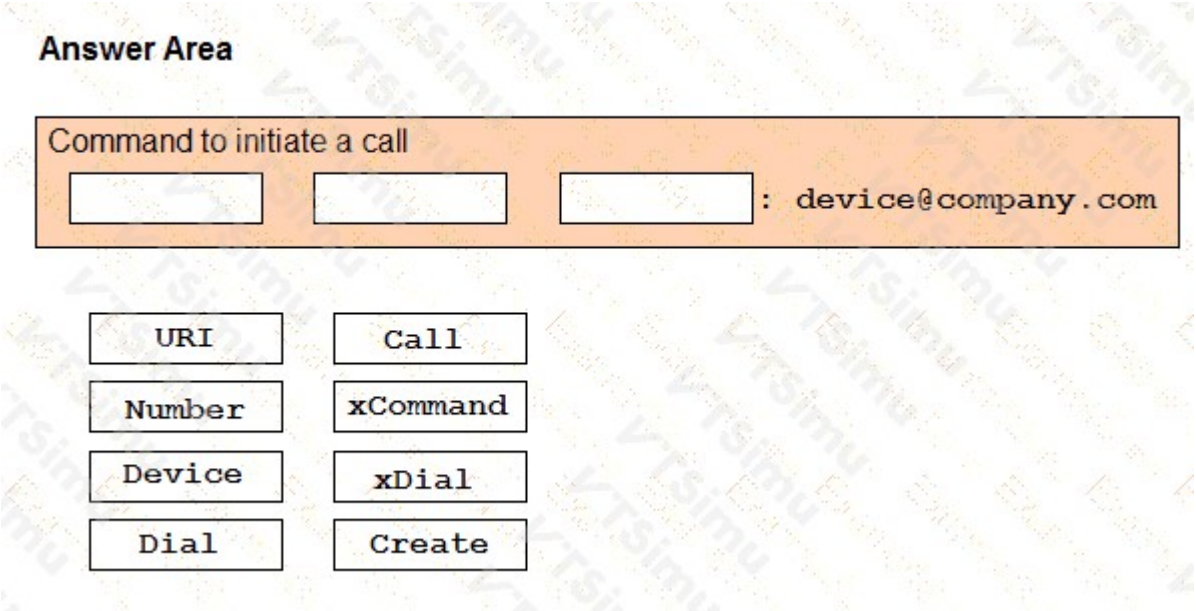
ANSWER: E

Explanation:

`ns:resetPhone` must be performed after `ns:updatePhone` when the changed phone setting requires the device to reload its configuration. The `updatePhone` AXL request successfully writes the revised phone configuration to Cisco Unified Communications Manager's database, but a phone that is already registered can continue operating with the configuration it obtained previously. A reset instructs Unified CM to reset the specified device, causing it to re-register and download its current configuration, thereby applying the updated values. This is the programmatic equivalent of using the Reset action for a phone in Unified CM administration when configuration changes must be pushed to the endpoint. The application should use the appropriate reset type and schedule the operation carefully because resetting an active device temporarily interrupts its call-processing registration and may affect an ongoing call. Cisco's AXL documentation describes `updatePhone` as the operation for modifying phone data and provides `resetPhone` as the related operation for resetting one or more phones. See the Cisco AXL API documentation and the Cisco Unified Communications Manager Administration Guides for phone configuration and reset behavior.

QUESTION NO: 6 - (DRAG DROP)

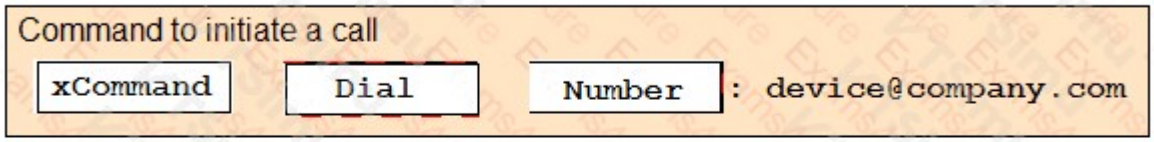
Drag and drop the elements to create the command to initiate a call to device@company.com using the Webex Devices xAPI SSH Interface. Not all options are used.

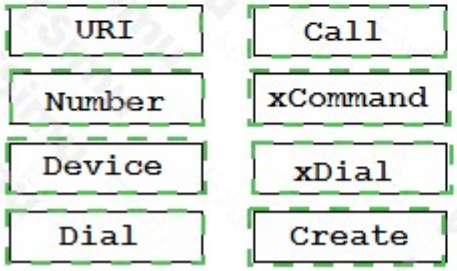


ANSWER:

Answer:

Answer Area





Answer:

Explanation:

To initiate an outbound call through the Webex Devices xAPI SSH interface, the command is constructed using the xAPI command prefix, the call-control command, and the parameter that supplies the destination. The correct full syntax is `xCommand Dial Number: device@company.com`.

`xCommand` identifies that the instruction is an executable xAPI command rather than a status query or configuration operation. `Dial` is the endpoint call-control command used to start a new outgoing call. The destination is then provided through the `Number` parameter, followed by a colon and the address to call. In this case, `device@company.com` is a URI-style destination, but it is still supplied to the Dial command through `Number`, producing a valid command line for the SSH interface.

This format follows the Webex Devices xAPI convention in which commands begin with `xCommand` and accept named arguments in the form `Parameter: value`. Cisco documents the Dial command and its Number argument in the xAPI command reference. See the Cisco RoomOS xAPI Command.Dial reference and the Cisco RoomOS xAPI documentation for the command model and SSH/API usage. Thus, the displayed arrangement correctly generates the command needed to call the specified device address.

QUESTION NO: 7

When the behavior of a Cisco collaboration device is customized, which use case requires an external control system because implementing JavaScript macro does not suffice?

A. Add a Join Webex meeting button to the touch panel.

B. Move the shutters up and down.

C. Trigger a “room-reset” to restore default configurations.

D. Implement an in-room control panel for speed-dialing.

ANSWER: B

Explanation:

“Move the shutters up and down.” is correct because operating motorized window shutters requires control of equipment that is external to the Cisco collaboration endpoint. JavaScript macros execute on the endpoint and can automate endpoint behavior, react to events, invoke endpoint commands, and extend the device user interface. They do not themselves provide the physical relay, serial, infrared, or building-automation interface needed to drive shutter motors. An external room-control processor, automation gateway, or shutter-control system is therefore required to perform the physical action.

The collaboration device can still participate in the workflow: a custom interface control or macro can send a request to the external system through an available integration method, allowing a user action on the device to initiate the shutter movement. However, the external controller remains responsible for communicating with the shutter hardware and safely commanding its movement. Cisco describes macros and user-interface extensions as mechanisms for customizing endpoint workflows and integrating with external services, while room-control integrations address control of environmental devices. See Cisco's macro and user-interface extension documentation and room controls documentation.

QUESTION NO: 8

Which two statements describe advantages of consuming APIs with asynchronous versus synchronous requests? (Choose two.)

A. All Cisco APIs are designed to be invoked asynchronously.

B. APIs respond more quickly when invoked asynchronously.

C. Asynchronous request coding is less complex.

D. Application threads do not block waiting for an asynchronous response.

E. Multiple asynchronous requests can be sent simultaneously.

ANSWER: D E

Explanation:

Application threads do not block waiting for an asynchronous response because asynchronous API consumption separates initiating a request from handling its eventual completion. After sending the request, the application can return control to its event loop, worker pool, or user interface and register a callback, future, promise, or completion handler to process the result later. This improves responsiveness and helps an integration use its execution resources efficiently when network or service operations take an unpredictable amount of time.

Multiple asynchronous requests can be sent simultaneously because the caller is not required to wait for one request to complete before starting another. This is particularly useful for collaboration automation workflows that must retrieve or update data across several endpoints or services. The client can track each outstanding operation independently and process responses as they arrive, which can reduce the overall elapsed time for a workflow when requests are independent. HTTP supports concurrent exchanges, while asynchronous client patterns provide the application-side mechanism for continuing work and coordinating those outstanding requests. See the HTTP semantics specification at RFC 9110 and the asynchronous JavaScript programming guidance at MDN Web Docs.

QUESTION NO: 9

Which Webex SDK allows group space calling?

A. Java

B. Browser

C. ChromeOS

D. Node.js

ANSWER: B

Explanation:

Browser is correct because the Webex Browser SDK (also called the Webex JavaScript SDK) provides the browser-side media and calling capabilities needed for real-time communications in Webex spaces. It is designed for web applications and uses WebRTC to establish and manage audio/video media sessions from supported browsers. In the context of a Webex space, the SDK can use the authenticated user's Webex identity and space context to support calling workflows, including joining or initiating group interactions associated with that space.

The Browser SDK is therefore the appropriate choice when an application must embed Webex collaboration experiences directly in a web page. Developers install the JavaScript package, authenticate with Webex, and use its calling and media APIs to control call lifecycle events, local and remote media streams, device selection, and participant-related behavior. This browser-native architecture is what enables interactive group calling rather than merely making REST API requests. Cisco's Webex developer documentation describes the browser-focused SDK and its real-time collaboration capabilities at Webex Browser SDK documentation. The SDK source, usage guidance, and supported package information are also available in the Webex JavaScript SDK repository.

QUESTION NO: 10

An In-Room Control Panel can be configured as "global" (always available). Which order panel type is supported?

A. Background

B. Do Not Disturb

C. Never

D. Out-of-Call

ANSWER: D

Explanation:

Out-of-Call is the supported panel type for a global In-Room Control panel. Cisco room-device In-Room Controls use panel visibility and ordering settings to determine where a custom control appears in the user interface. A panel configured as global is intended to remain accessible regardless of the device's current call state. Pairing that global availability with the Out-of-Call panel type provides a consistent place for controls that are primarily associated with the room rather than with an active meeting, such as room-environment or local operational controls.

Cisco's In-Room Controls configuration model supports creating and placing custom panels through the device web interface, xAPI, or Control Hub. The panel definition includes information that determines the panel's placement and behavior in the RoomOS interface. When a panel is made global, it is not limited to a particular call workflow; therefore, Out-of-Call is the supported ordering context stated for this configuration. This lets the device present the panel predictably whenever users access the room controls interface. Cisco's guidance for configuring custom controls is available in the Webex In-Room Controls documentation and the Cisco RoomOS xAPI documentation.

QUESTION NO: 11

Which two statements describe the OAuth authorization-code flow for a Webex integration? (Choose two.)

A. The user is redirected to Webex to authorize the integration.

B. The authorization code is exchanged for access and refresh tokens.

C. The authorization code is used as the bearer token for Webex API requests.

D. The integration can use any redirect URI during token exchange.

E. The client secret is included in every Webex API request.

ANSWER: A B

Explanation:

The user is redirected to the Webex authorization URL so that the user can authenticate and approve the scopes requested by the integration. After authorization succeeds, Webex redirects the user agent to the integration's registered redirect URI with an authorization code.

The integration then exchanges that authorization code for access and refresh tokens by calling the Webex access-token endpoint. The authorization code is short-lived and is not itself used as the bearer credential for subsequent Webex API

calls. The registered redirect URI helps ensure that the authorization response is returned only to an approved integration endpoint. See the Webex Integrations documentation and the Webex authentication guide.

QUESTION NO: 12

Which two practices help an application process paginated responses from a Webex REST API? (Choose two.)

A. Follow the Link header URL identified by rel="next".

B. Use the max query parameter when it is supported by the resource.

C. Assume that a successful response contains every matching resource.

D. Use HTTP POST to request each additional page.

E. Increase the OAuth token lifetime to retrieve additional pages.

ANSWER: A B

Explanation:

A Webex API response can include a `Link` header that identifies the URL for the next page of results. An application should follow the URL identified with `rel="next"` until no next-page link is returned. This enables the application to retrieve all resources rather than only the first page.

An application can also use the `max` query parameter, when supported by the API, to request a page size appropriate for the workflow. The client must still process the returned pages because a maximum page size does not guarantee that all matching resources are returned in one response. See the Webex API getting-started documentation.

QUESTION NO: 13 - (DRAG DROP)

DRAG DROP

This Python script automates the creation of a Webex Teams space and adds participants to the space. Drag and drop code on the snippet to complete the script. Not all options are used.

Select and Place:

Answer Area

```
import requests

head = { 'Content-Type': '                    ',
    'Authorization': 'Bearer NWU4NjQ0YWUtNJYItMWMy_-417f-ae0e10f' }

res = requests.post(url = 'https://api.ciscospark.com/v1/rooms/',
   headers = head, json = { '                    ': 'Populate members' })

spaceId = res.json()['id']

members = ['ToEnglish@webex.bot', 'ToSpanish@webex.bot']
for members in members:
   requests.post(url='https://api.ciscospark.com/v1/                    /',
      headers = head,
      json = { 'roomId': spaceId, '                    ': member})
```

/memberships	person	title	/rooms
/spaces	personEmail	application/xml	
application/json	name	/members	

ANSWER:

Answer Area

```
import requests

head = { 'Content-Type': ' application/json ',
    'Authorization': 'Bearer NWU4NjQ0YWUtNJYItMWMy_-417f-ae0e10f' }

res = requests.post(url = 'https://api.ciscospark.com/v1/rooms/',
   headers = head, json = { '       title       ': 'Populate members' })

spaceId = res.json()['id']

members = ['ToEnglish@webex.bot', 'ToSpanish@webex.bot']
for members in members:
   requests.post(url='https://api.ciscospark.com/v1/  /memberships  /',
      headers = head,
      json = { 'roomId': spaceId, '  personEmail  ': member})
```

/memberships	person	title	/rooms
/spaces	personEmail	application/xml	
application/json	name	/members	

Explanation:

The script first creates a Webex space by making a POST request to the Rooms resource. Webex APIs use JSON request bodies for this operation, so the request header must specify `application/json` as its content type. The body sent to the Rooms resource must use `title` for the human-readable name of the new space. In this script, that produces a room titled "Populate members," and the returned room object provides its `id`, which is saved in `spaceId`.

The loop then creates one membership for each email address in the `members` list. Creating a membership is the Webex API operation that adds a person to an existing space, so the POST URL must end with `/memberships`. Its JSON body needs both the ID of the room being updated and the email address of the person being added. The room identifier belongs in `roomId`, and the individual email address from the loop belongs in `personEmail`. This creates a membership record connecting that person to the newly created room. Cisco's Webex developer documentation describes the required `title` field for room creation in the Create a Room API, and documents `roomId` and `personEmail` for adding users through the Create a Membership API.

QUESTION NO: 14 - (SIMULATION)

A developer for a large company must change the logo on all Cisco collaboration room devices and a base64 image has already been provided. Drag

and drop the code snippets from the bottom onto the boxes in the Python script to update the logo on devices in the 192.168.1.1 to 192.168.1.50 IP

range by using the xAPI. Not all options are used.

Select and Place:

ANSWER: See the explanation for the answer

Explanation:

Place the snippets in this order, from the top box to the bottom box:

The script loops through host values 1 through 50. Cisco collaboration endpoints accept XML xAPI commands at the `/putxml` endpoint; the branding image is uploaded with the `UserInterface/Branding` command using HTTP Basic authentication and a `POST` request.