Vendor: Cisco

Exam Code:300-920

Exam Name: Developing Applications for Cisco Webex and Webex Devices (DEVWBX)

Version: Demo

QUESTION 1

A Webex Teams bot is deployed but soon it stops responding. Which two explanations are the cause of the issue? (Choose two.)

A. A new webhook was created, which marks the old webhook as inactive.

B. The web server that is set to receive webhooks is not configured to return a 200 message. And the webhook is disabled.

C. The webhook secret is expired and must be refreshed.

D. The refresh token is not being used.

E. The bot owner regenerated the access token on developer.webex.com.

Correct Answer: AB

Reference: https://developer.authorize.net/api/reference/features/webhooks.html

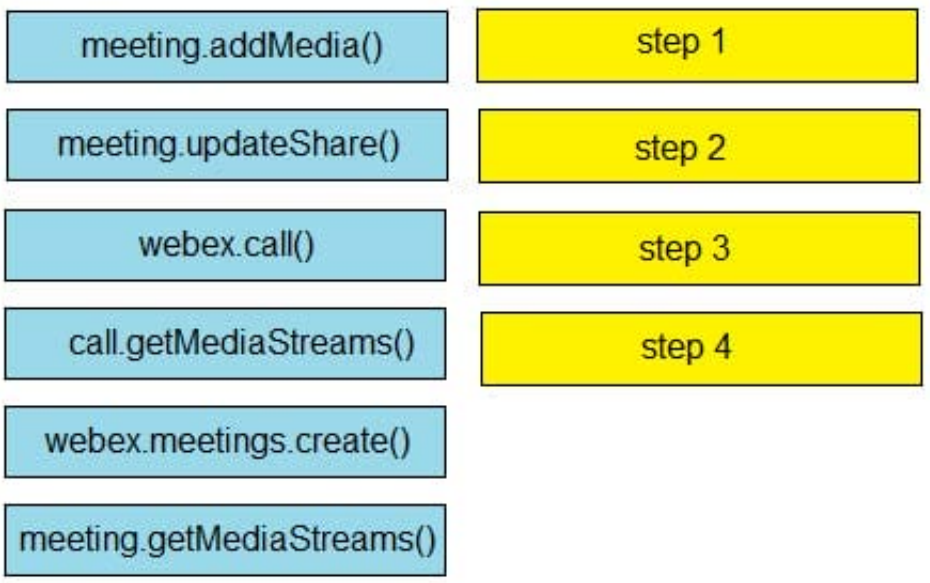
---

QUESTION 2

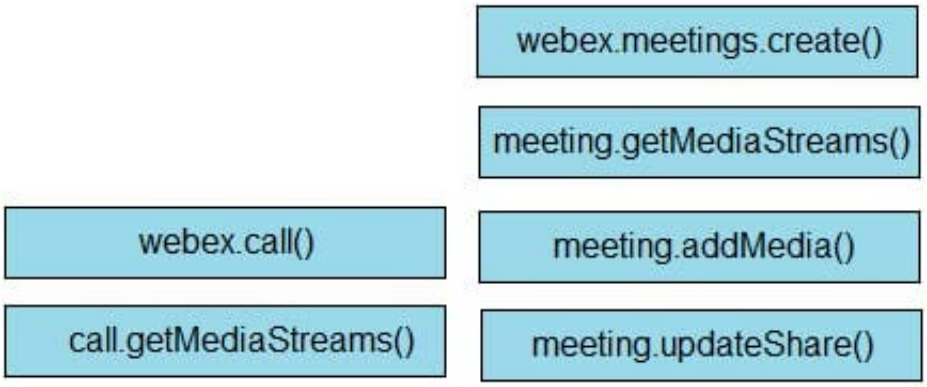
DRAG DROP

Drag and drop the methods from the left into the correct order of execution on the right to use webex-js-sdk in a browser to call and share the screen with another Webex user. Not all methods are used.

Select and Place:



Correct Answer:



Reference: https://github.com/webex/webex-js-sdk/blob/master/packages/node_modules/%40webex/plugin-meetings/README.md (see start wireless share)

QUESTION 3

Which two capabilities are currently supported by the Webex Meetings XML API? (Choose two.)

A. Request a recording link for playback.

B. Send a text message to the meeting host.

C. Request permissions to schedule on someone else\\'s behalf.

D. Schedule a new meeting.

E. Send a problem report.

Correct Answer: CD

Reference: https://community.cisco.com/t5/cloud-collaboration/api-for-scheduling-a-webex-meeting-on-behalf-of-other-user/td-p/3472319

QUESTION 4

```
document.getElementById('share-screen').addEventListener('click', () => {
     if (activeMeeting) {
        const mediaSettings = {
          receiveShare: true,
          sendShare: true,
     };

     console.info('SHARE-SCREEN: Preparing to share screen via 'getMediaStreams'');
     activeMeeting.getMediaStreams(mediaSettings)
       // '[, localShare]' is grabbing index 1 from the mediaSettingsResultsArray
     // and storing it in a variable called localShare.
     .then((mediaSettingsResultsArray) => {
          const [, localShare] = mediaSettingsResultsArray;

          console.info('SHARE-SCREEN: Add local share via 'updateShare'');

          return << missing code >>
          })
          .then(() => {
          console.info('SHARE-SCREEN: Screen successfully added to meeting.');
        })
       .catch(e) => {
          console.error('SHARE-SCREEN: Unable to share screen, error:');
          console.error(e);
     });
  }
  else {
      console.error('No active meeting available to share screen.);
   }
 });
```

Refer to the exhibit.Which code completes the return statement that initiates local screen sharing on the active meeting?

A. activeMeeting.updateShare({ sendShare: true receiveShare: true, stream: null

})

B. activeMeeting.updateShare({ sendShare: true receiveShare: false, stream: remoteShare })

C. activeMeeting.updateShare({ sendShare: true receiveShare: true, stream: localShare })

D. activeMeeting.updateShare({ sendShare: false receiveShare: false, stream: null })

Correct Answer: C

Reference: https://github.com/webex/webex-js-sdk/tree/master/packages/node_modules/%40webex/plugin-meetings

---

QUESTION 5

```
xcommand UserInterface Extensions List
OK
*r ExtensionsListResult (status=OK):
*r ExtensionsListResult Extensions Version: "1.5"
*r ExtensionsListResult Extensions Panel 1 Icon: Lightbulb
*r ExtensionsListResult Extensions Panel 1 Type: Statusbar
*r ExtensionsListResult Extensions Panel 1 Name: Panel
*r ExtensionsListResult Extensions Panel 1 PanelId: "panel_1"
*r ExtensionsListResult Extensions Panel 1 Order: 1
*r ExtensionsListResult Extensions Panel 1 Page 1 Name: "Page"
*r ExtensionsListResult Extensions Panel 1 Page 1 Row 1 Name: "Row"
*r ExtensionsListResult Extensions Panel 1 Page 1 Row 1 Widget 1 WidgetId: "widget_1"
*r ExtensionsListResult Extensions Panel 1 Page 1 Row 1 Widget 1 Type: Slider
*r ExtensionsListResult Extensions Panel 1 Page 1 Row 1 Widget 1 Options: "size=3"
** end
--
```

Refer to the exhibit. With this in-room configuration, which command successfully sets the slider to 50?

A. xcommand UserInterface Extensions Widget SetSlider Value: 50

B. xcommand UserInterface Extensions Widget Slider: 50 Widget: "widget_1"

C. xconfiguration UserInterface Extensions widget_1 50

D. xcommand UserInterface Extensions Widget SetValue WidgetId: "widget_1" Value: 50

Correct Answer: A

Reference: https://www.cisco.com/c/dam/en/us/td/docs/telepresence/endpoint/ce98/sx-mx-dx-room-kit-boards-customization-guide-ce98.pdf

---

QUESTION 6

DRAG DROP

Drag and drop the code to complete the JavaScript snippet so that it:

1.

retrieves the details of an individual user

2.

checks what licenses they have already

3.

updates their account with a new license Options can be used more than once.

Select and Place:

```
const request = require ('require');
var adminToken = 'VALID_ADMIN_TOKEN";
var personId = "VALID_PERSON_ID";
var licenseId = "VALID_LICENSE_ID";
function buildOptions(sendURL) {
     return (
      url:sendURL,
      headers: {
           'Content-Type': 'application/json',
           'Authorization': 'Bearer $ (adminToken}'
     }
  }
}
var peopleOptions = buildOptions('https://api.ciscospark.com/vl/ [          ] /${personId}');
/${personId}');
request.get(peopleOptions, function(error, response) {
 var json = JSON.parse(response.body);

if(json. [          ] .indexOf( [          ] ) < 0) {

json. [          ] .push( [          ] ) :

peopleOptions['body'] = JSON.stringify[json] :
request.put(peopleOptions, function(error, response) {
     var json = JSON.parse(response.body) ;
     console.log (json) ;
     }) ;
   }
}) ;
```

licenseId | person

people | licenses

Correct Answer:

```
const request = require ('require');
var adminToken = 'VALID_ADMIN_TOKEN";
var personId = "VALID_PERSON_ID";
var licenseId = "VALID_LICENSE_ID";
function buildOptions(sendURL) {
     return (
      url:sendURL,
      headers: {
          'Content-Type': 'application/json',
          'Authorization': 'Bearer $ (adminToken}'
     }
  }
}
var peopleOptions = buildOptions('https://api.ciscospark.com/vl/ [ people ] /${personId}');
/${personId}');
request.get(peopleOptions, function(error, response) {
 var json = JSON.parse(response.body);

if(json. [ licenses ] .indexOf( [ licenseId ] ) < 0) {

json. [ people ] .push( [ person ] ) :

peopleOptions['body'] = JSON.stringify[json] :
request.put(peopleOptions, function(error, response) {
     var json = JSON.parse(response.body) ;
     console.log (json) ;
     }) ;
   }
}) ;
```

[licenseId] [person]

[people] [licenses]

QUESTION 7

Which two statements about using the Compliance Office role to manage data in Webex Teams are true? (Choose two.)

A. For group rooms with members from multiple organizations, only the compliance officer of the owning organization can manage all data.

B. The compliance officer can manage data only for users who have granted the spark-compliance messages scope.

C. For direct rooms between two organizations, each compliance officer can manage data only from their organization.

D. For group rooms with members from multiple organizations, either compliance offcer can manage all data.

E. For direct rooms between two organizations, the compliance officer of either organization can manage all data.

Correct Answer: CE

Reference: https://developer-portal-intb.ciscospark.com/docs/api/guides/compliance

---

QUESTION 8

DRAG DROP

Drag and drop the code to complete the JavaScript code snippet to create a meeting using the Webex Meetings XML API. Options may be used more than once.

Select and Place:

```
<script>
  const init = {
    method: 'POST'
    body: <message xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance">
     <header><securityContext>
       <siteName>apidemoeu</siteName>
      <webExID>alice</webExID>
     <[                    ]>AAABb6FFuSBR19BRERJTkdfU0hBMjU2X0FMR09SSVRITV8=
 </[                    ]>
     </securityContext></header>
     <body>
          <bodyContent xsi:type="*ava:com.webex.service.binding.meeting.CreateMeeting">
               <accessControl><[                    ]>Cisco1234</[                    ]>
 </accessControl>
      <metaData><confName>Sample Meeting</confName>
      <meetingType>105</meetingType></metaData>

      <schedule><startDate>[                    ]</startDate></schedule>
     </bodyContent>
    </body></message>' }
   fetch('https://api.webex.com/WBXService/XMLService', init)
     .then(response => response.text())
     .then(str => (new window.DOMParser()).parseFromString(str, '[                    ]'))

    .then(data => document.write(data.getElementsByTagNameNS('*', '[                    ]')
[0].textContent))
</script>
```

result	application/json
text/xml	1/13/2020 16:00:00
meetingPassword	sessionTicket
Jan 24, 2019 4:00 PM	status

Correct Answer:

```
<script>
  const init = {
    method: 'POST'
    body: <message xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance">
     <header><securityContext>
       <siteName>apidemoeu</siteName>
      <webExID>alice</webExID>
     <[ sessionTicket ]>AAABb6FFuSBR19BRERJTkdfU0hBMjU2X0FMR09SSVRITV8=
 </[ status ]>
     </securityContext></header>
     <body>
          <bodyContent xsi:type="*ava:com.webex.service.binding.meeting.CreateMeeting">
               <accessControl><[ meetingPassword ]>Cisco1234</[ meetingPassword ]>
 </accessControl>
      <metaData><confName>Sample Meeting</confName>
      <meetingType>105</meetingType></metaData>

      <schedule><startDate>[ Jan 24, 2019 4:00 PM ]</startDate></schedule>
     </bodyContent>
    </body></message>' }
   fetch('https://api.webex.com/WBXService/XMLService', init)
     .then(response => response.text())
     .then(str => (new window.DOMParser()).parseFromString(str, '[ text/xml ]'))
    .then(data => document.write(data.getElementsByTagNameNS('*', '[ result ]')
[0].textContent))
</script>
```

result	application/json
text/xml	1/13/2020 16:00:00
meetingPassword	sessionTicket
Jan 24, 2019 4:00 PM	status

QUESTION 9

Which user role implicitly has permission in the Events API to retrieve "message created" events from other users in the same organization?

A. Compliance Officer

B. User and Device Administrator

C. Full Administrator

D. User Administrator

Correct Answer: A

Reference: https://developer.webex.com/docs/api/guides/compliance

---

QUESTION 10

```
1   const jsxapi = require('jsxapi');
2   const speaker = require('./speaker')
3   const xapi = jsxapi.connect('ssh://10.131.202.230', {
4       username: 'admin',
5       password: ''
6   });
7   let state = 0
8   function toggle(state) {
9       if (state === 0) {return 1}
10      else {return 0}
11  }
12  xapi.on('ready', () => {
13     xapi.event
14       .on('                    ', (x) => {
15           if (x.PanelId === 'panel_1') {
16             state = toggle(state)
17            if (state === 1) {
18               speaker.control({"on": true});
19            } else {
20               speaker.control({"on": false})
21            }
22          }
23       })
24  });
```

Refer to the exhibit. An `Action Button\\' with identifier `panel_1\\' is deployed to the Touch10 interface of a Room Series device. Which event must be inserted into line 14 to turn the music on/off in the conference room?

A. UserInterface Extensions Event Pressed

B. UserInterface Extensions Panel Clicked

C. UserInterface Extensions Widget GetValue

D. UserInterface Extensions Widget Action

Correct Answer: A

Reference: https://www.cisco.com/c/dam/en/us/td/docs/telepresence/endpoint/ce81/sx-mx-in-room-control-guide-ce81.pdf

---

QUESTION 11

DRAG DROP

Drag and drop the code to complete the JavaScript to display the first URL from a user\\'s list of Webex Meetings

recordings. Not all options are used.

Select and Place:

```
const init = { [                              ] ,
body: '<message xmlns:xsi= "http://www.v3.org/2001/XMLSchema-instance">
    <header><securityContext>
        <siteName>apidemoeu</siteName>
        <webExID>alice</webExID>
        <sessionTicket>AAABb6CkTRgAABUYARZAD2X0FMR09SSVRITV8=</sessionTicket>
    </securityContext></header>
    <body>
        <bodyContent xsi:type="[                              ]">
         <createTimeScope>
                        <createTimeStart>11/21/2019 0:0:0</createTimeStart>
                        <createTimeEnd>11/21/2019 23:59:59</createTimeEnd>
             </createTimeScope>
         </bodyContent></body></message>  };
         fetch{'[                              ]',init)
            .then(response => response.text())
            .then(str => (new window.DOMParser()).parseFromString(str, 'text/xml'))
            .then(data => document.write(data.getElementByTagNameNS(
                '[                              ]',
                'fileURL') [0].textContent));
```

- http://www.webex.com/schemas/2002/06/service/meeting
- method: 'POST'
- java:com.webex.service.binding.ep.LstRecording
- java:com.webex.service.binding.meeting.Recordings
- http://www.webex.com/schemas/2002/06/service/ep
- method: 'GET'
- https://api.webex.com/WBXService/XMLService
- https://api.webex.com/v1/recordings

Correct Answer:

```
const init = { [ method: 'POST' ] ,
body: '<message xmlns:xsi= "http://www.v3.org/2001/XMLSchema-instance">
     <header><securityContext>
          <siteName>apidemoeu</siteName>
          <webExID>alice</webExID>
          <sessionTicket>AAABb6CkTRgAABUYARZAD2X0FMR09SSVRITV8=</sessionTicket>
     </securityContext></header>
     <body>
          <bodyContent xsi:type="[ java:com.webex.service.binding .ep.LstRecording ]">
           <createTimeScope>
                          <createTimeStart>11/21/2019 0:0:0</createTimeStart>
                          <createTimeEnd>11/21/2019 23:59:59</createTimeEnd>
                </createTimeScope>
           </bodyContent></body></message>  };
           fetch('[ https://api.webex.com/WBX Service/XMLService ]',init)
               .then(response => response.text())
               .then(str => (new window.DOMParser()).parseFromString(str, 'text/xml'))
               .then(data => document.write(data.getElementByTagNameNS(
                    '[ http://www.webex.com/schemas/ 2002/06/service/ep ]',
                   'fileURL') [0].textContent));
```

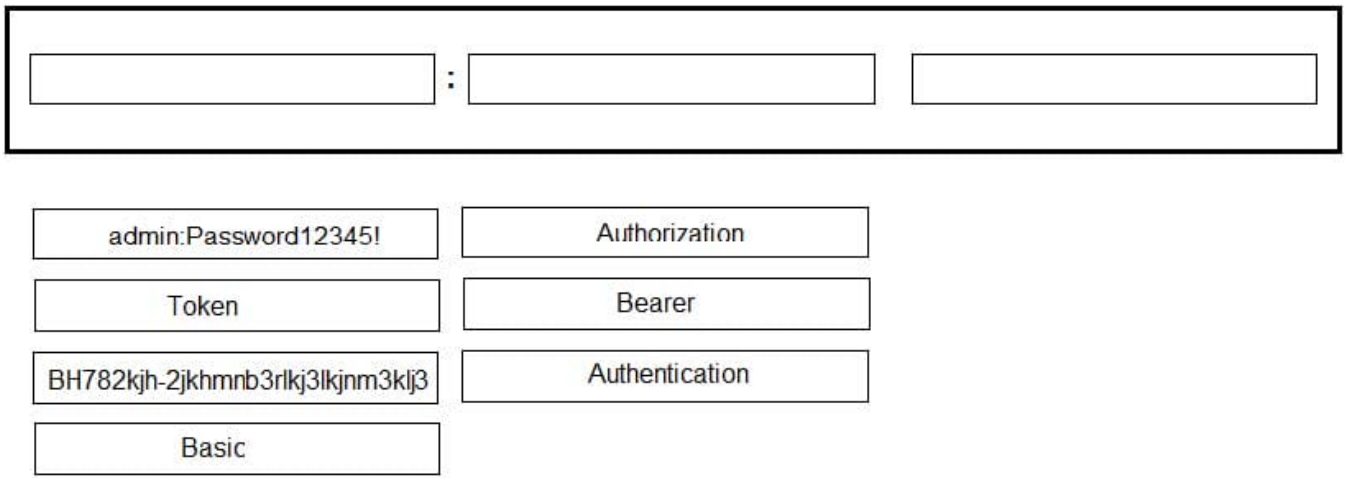
- http://www.webex.com/schemas/2002/06/service/meeting
- java:com.webex.service.binding.meeting.Recordings
- method: 'GET'
- https://api.webex.com/v1/recordings

QUESTION 12

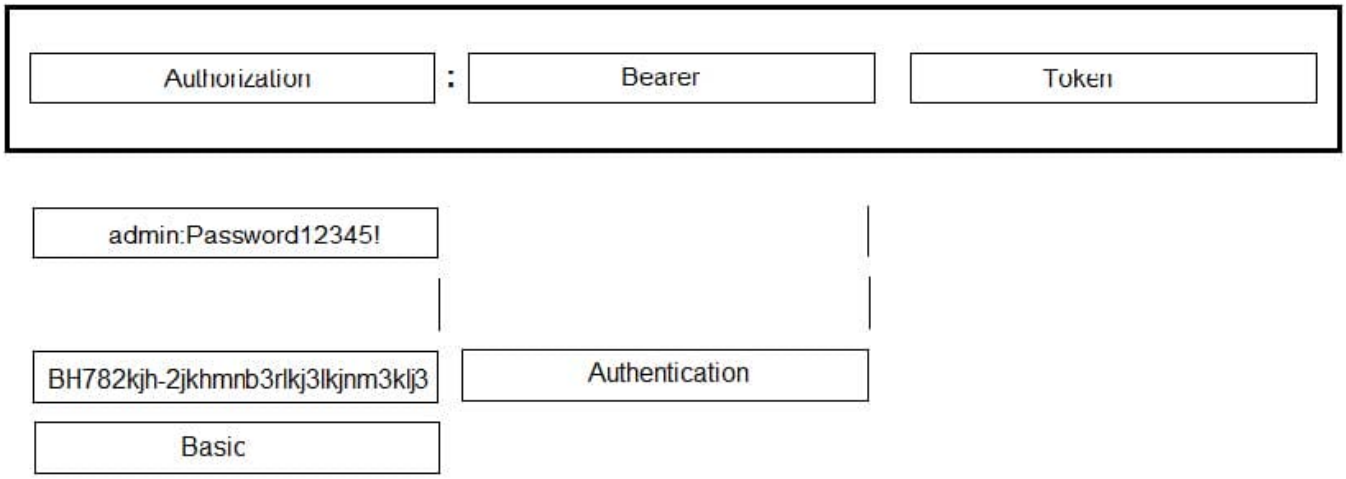
DRAG DROP

Drag and drop the components to create the authentication header used for the Webex Teams API. Not all options are used.

Select and Place:



Correct Answer:



Reference: https://developer.webex.com/docs/guest-issuer