300-920 Q&As

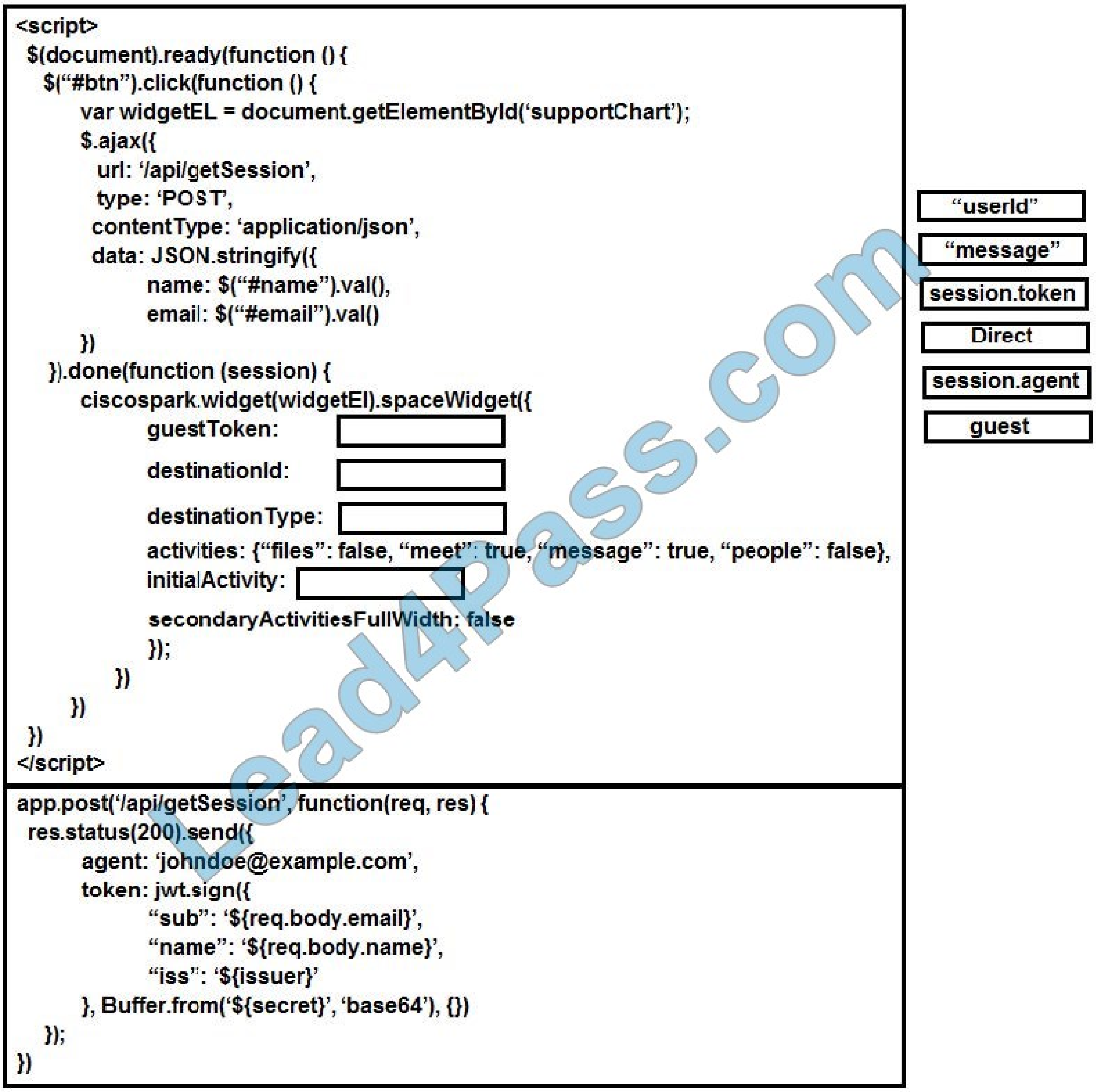
Developing Applications for Cisco Webex and Webex Devices (DEVWBX)

QUESTION 1

DRAG DROP

Drag and drop the expressions to create a Webex Teams widget that uses Guest Issuer to enable customers to chat and meet with agents. Not all options are used.

Select and Place:

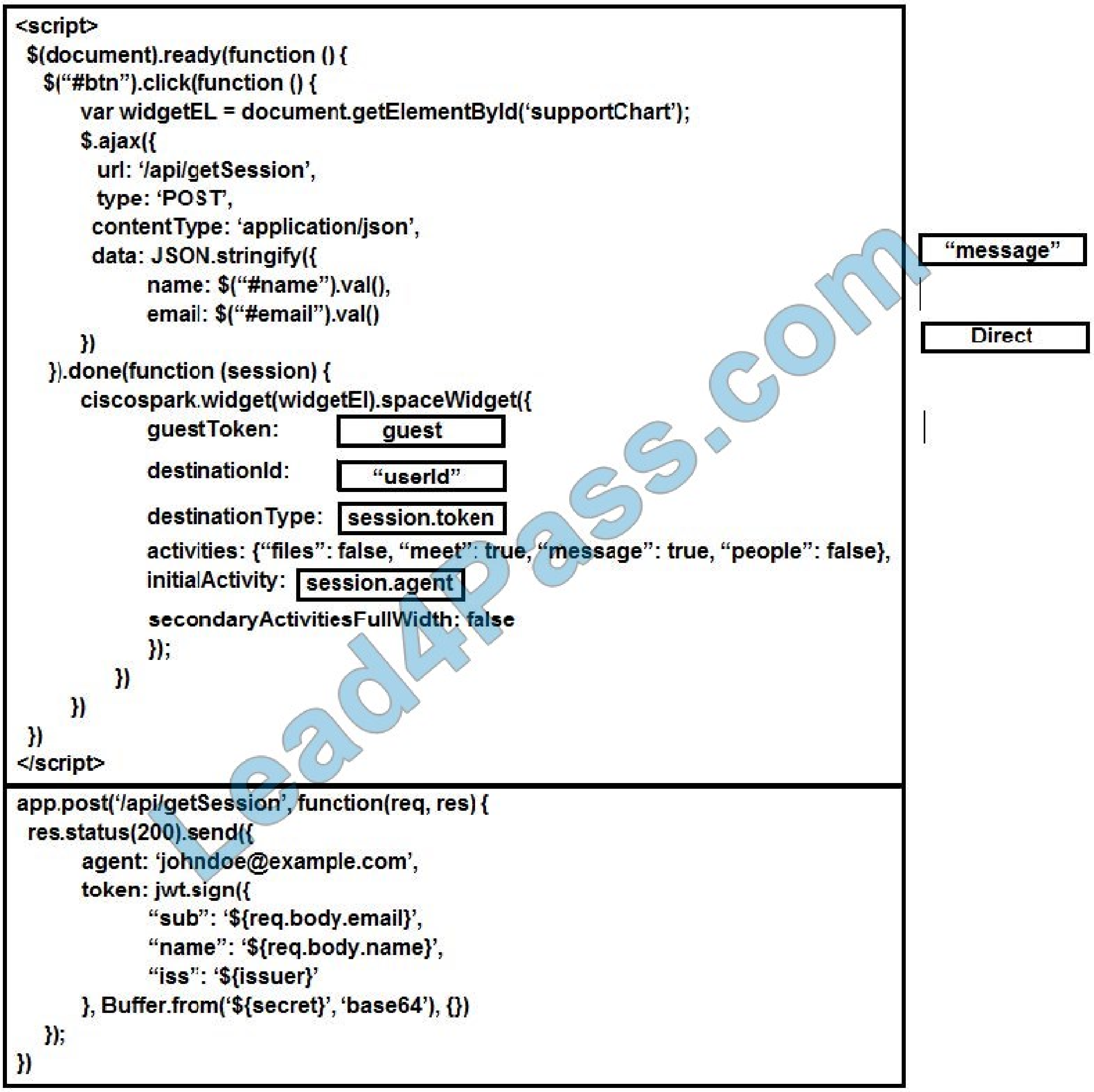


```
<script>
 $(document).ready(function () {
  $("#btn").click(function () {
      var widgetEL = document.getElementById('supportChart');
      $.ajax({
        url: '/api/getSession',
        type: 'POST',
       contentType: 'application/json',
       data: JSON.stringify({
             name: $("#name").val(),
             email: $("#email").val()
      })
   }).done(function (session) {
      ciscospark.widget(widgetEl).spaceWidget({
             guestToken: [          ]
             destinationId: [          ]
             destinationType: [          ]
             activities: {"files": false, "meet": true, "message": true, "people": false},
             initialActivity: [          ]
             secondaryActivitiesFullWidth: false
             });
          })
       })
 })
</script>
```

```
app.post('/api/getSession', function(req, res) {
 res.status(200).send({
      agent: 'johndoe@example.com',
      token: jwt.sign({
             "sub": '${req.body.email}',
             "name": '${req.body.name}',
             "iss": '${issuer}'
      }, Buffer.from('${secret}', 'base64'), {})
   });
})
```

"userId"
"message"
session.token
Direct
session.agent
guest

Correct Answer:



```
<script>
 $(document).ready(function () {
  $("#btn").click(function () {
       var widgetEL = document.getElementById('supportChart');
       $.ajax({
         url: '/api/getSession',
         type: 'POST',
        contentType: 'application/json',
        data: JSON.stringify({
              name: $("#name").val(),
              email: $("#email").val()
       })
   }).done(function (session) {
       ciscospark.widget(widgetEl).spaceWidget({
              guestToken:   [ guest ]
              destinationId:   [ "userId" ]
              destinationType:   [ session.token ]
              activities: {"files": false, "meet": true, "message": true, "people": false},
              initialActivity:   [ session.agent ]
              secondaryActivitiesFullWidth: false
              });
          })
      })
 })
</script>
```

```
app.post('/api/getSession', function(req, res) {
 res.status(200).send({
       agent: 'johndoe@example.com',
       token: jwt.sign({
              "sub": '${req.body.email}',
              "name": '${req.body.name}',
              "iss": '${issuer}'
       }, Buffer.from('${secret}', 'base64'), {})
    });
})
```

"message"

Direct

QUESTION 2

Which two filters are valid for limiting a webhook? (Choose two.)

A. roomId=

B. personId!=

C. spaceId=

D. personId=$spaceId=

E. personId=$roomId=

Correct Answer: AB

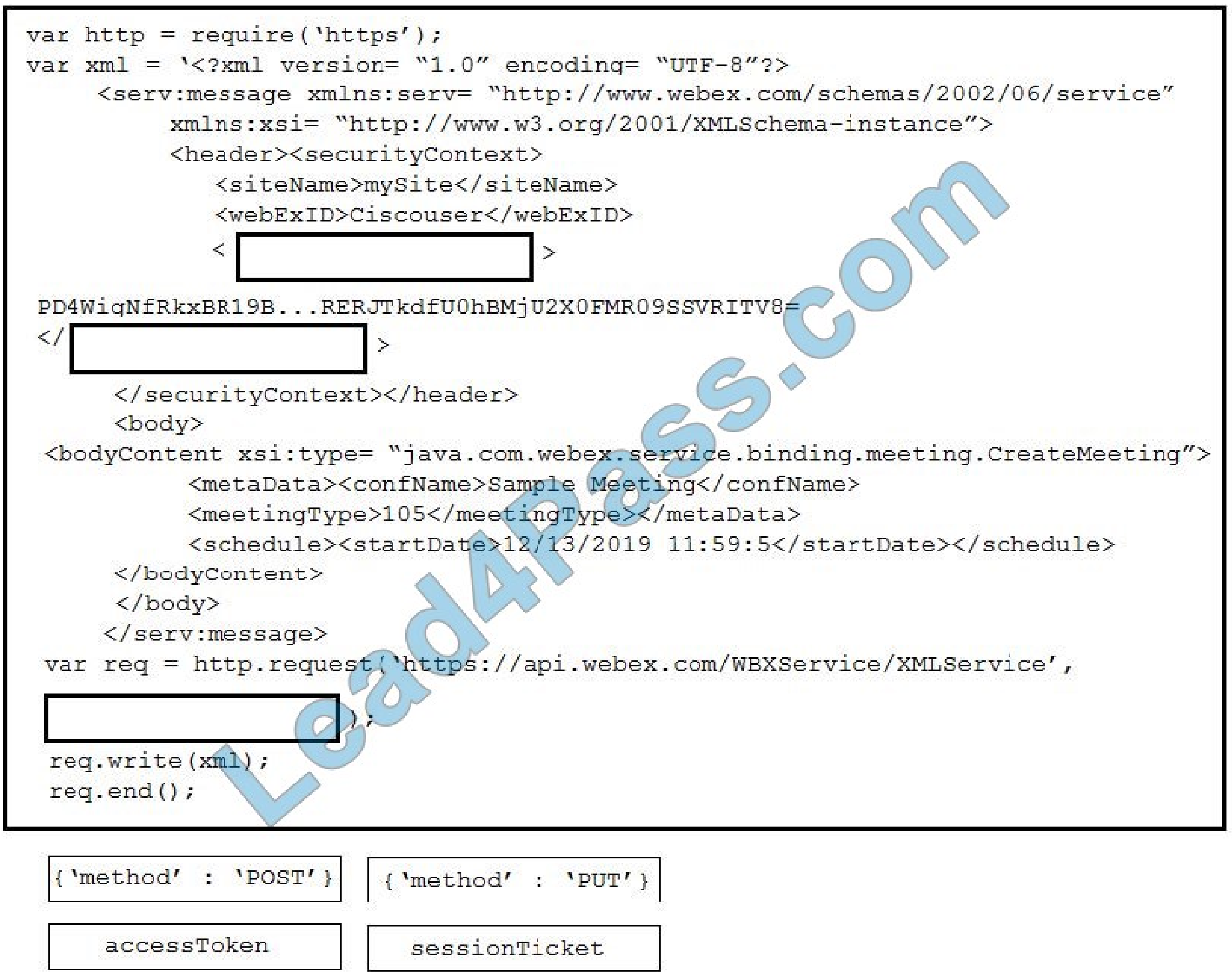
Reference: https://developer.webex.com/docs/api/guides/webhooks

QUESTION 3

DRAG DROP

Drag and drop the code onto the snippet to construct the JavaScript to create a new meeting with the Webex Meetings XML API. Options can be used more than once.

Select and Place:

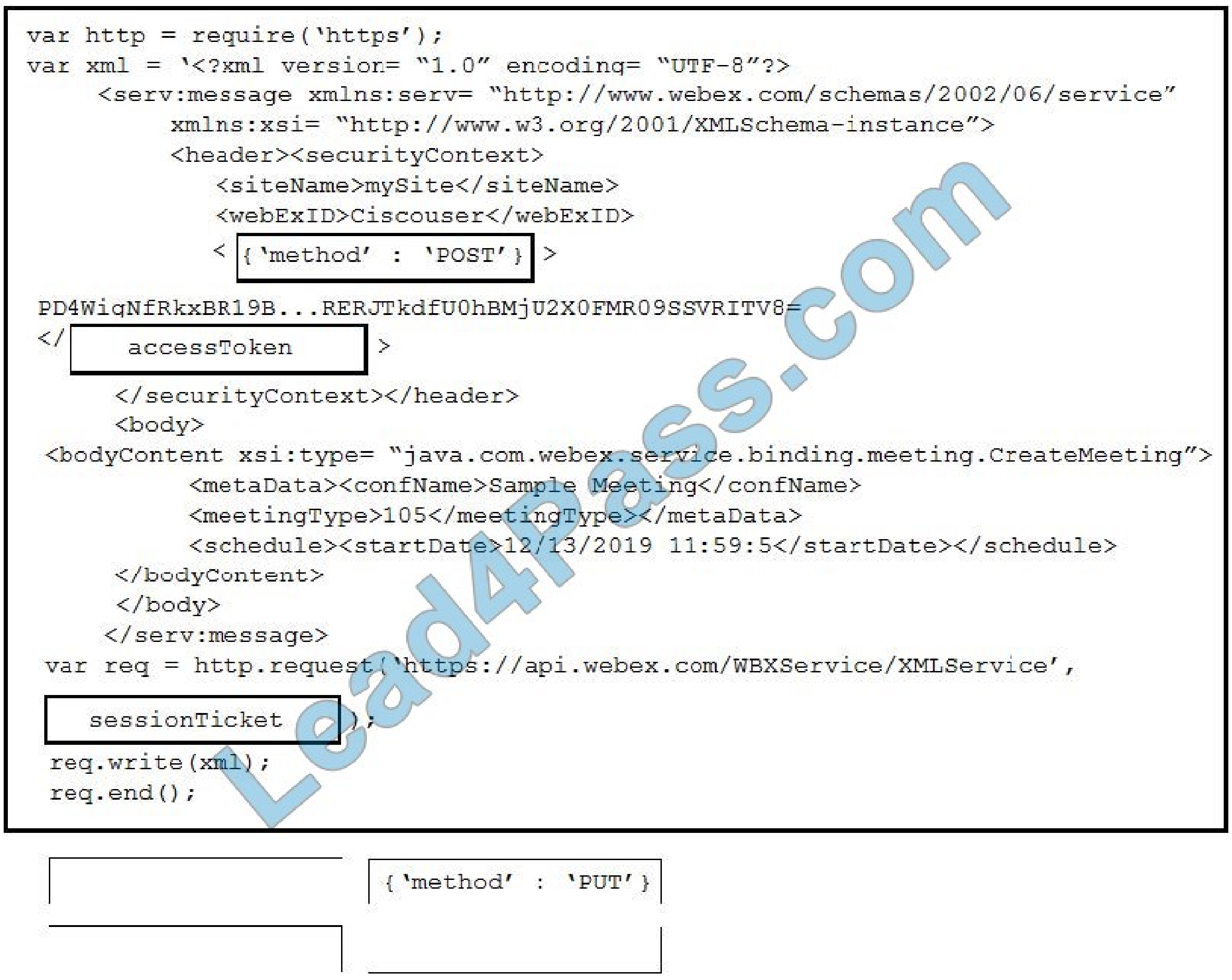


```
var http = require('https');
var xml = '<?xml version= "1.0" encoding= "UTF-8"?>
    <serv:message xmlns:serv= "http://www.webex.com/schemas/2002/06/service"
        xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance">
        <header><securityContext>
           <siteName>mySite</siteName>
           <webExID>Ciscouser</webExID>
           <                    >
 PD4WiqNfRkxBR19B...RERJTkdfU0hBMjU2X0FMR09SSVRITV8=
 </                    >
      </securityContext></header>
      <body>
 <bodyContent xsi:type= "java.com.webex.service.binding.meeting.CreateMeeting">
          <metaData><confName>Sample Meeting</confName>
          <meetingType>105</meetingType></metaData>
          <schedule><startDate>12/13/2019 11:59:5</startDate></schedule>
      </bodyContent>
      </body>
     </serv:message>
 var req = http.request('https://api.webex.com/WBXService/XMLService',
                      );
 req.write(xml);
 req.end();
```

```
{'method' : 'POST'}    {'method' : 'PUT'}

accessToken            sessionTicket
```

Correct Answer:



```
var http = require('https');
var xml = '<?xml version= "1.0" encoding= "UTF-8"?>
     <serv:message xmlns:serv= "http://www.webex.com/schemas/2002/06/service"
          xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance">
          <header><securityContext>
             <siteName>mySite</siteName>
             <webExID>Ciscouser</webExID>
             < {'method' : 'POST'} >
 PD4WiqNfRkxBR19B...RERJTkdfU0hBMjU2X0FMR09SSVRITV8=
 </   accessToken   >
      </securityContext></header>
      <body>
 <bodyContent xsi:type= "java.com.webex.service.binding.meeting.CreateMeeting">
          <metaData><confName>Sample Meeting</confName>
          <meetingType>105</meetingType></metaData>
          <schedule><startDate>12/13/2019 11:59:5</startDate></schedule>
      </bodyContent>
      </body>
     </serv:message>
 var req = http.request('https://api.webex.com/WBXService/XMLService',
  sessionTicket   );
 req.write(xml);
 req.end();
```

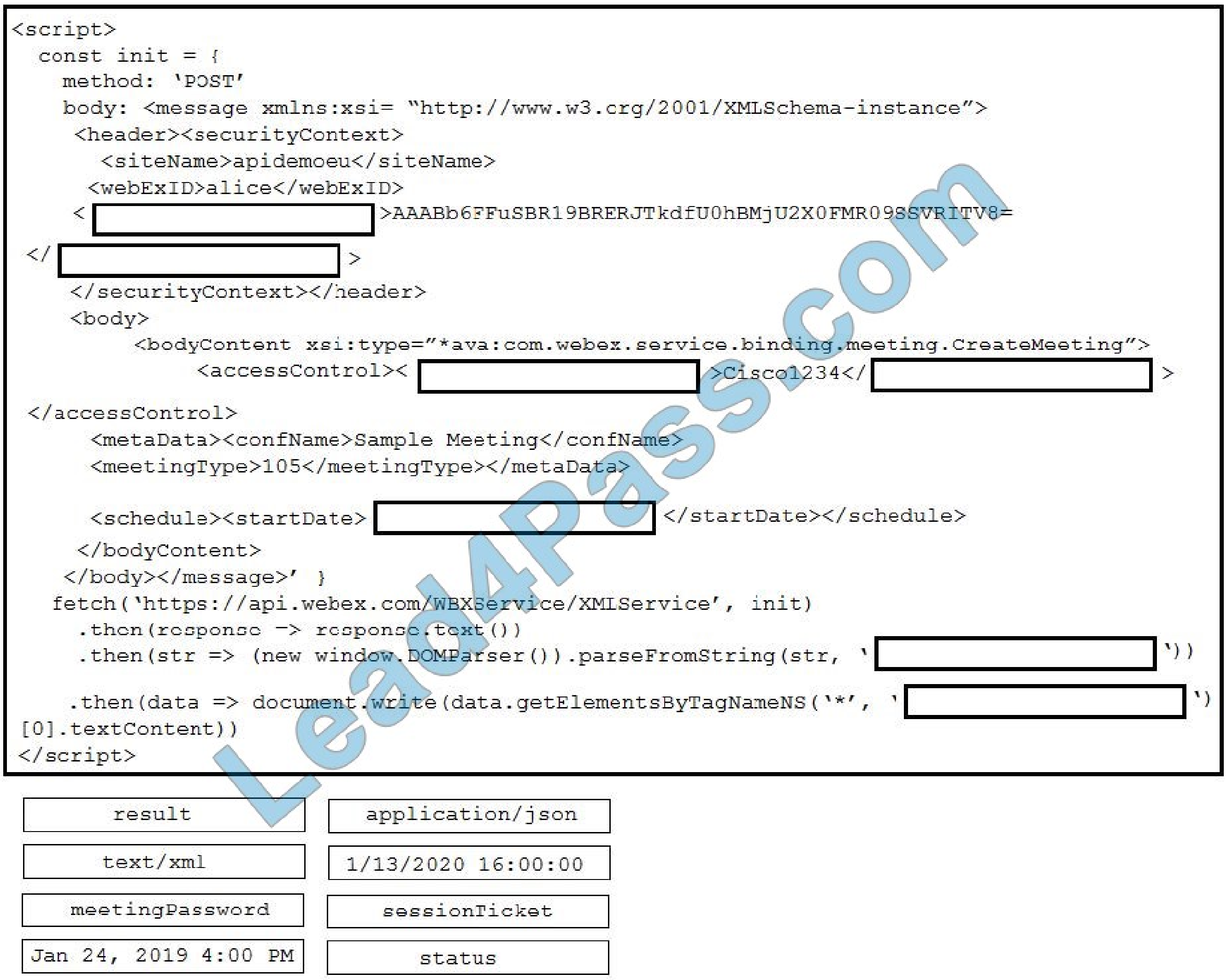
```
{'method' : 'PUT'}
```

QUESTION 4

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:



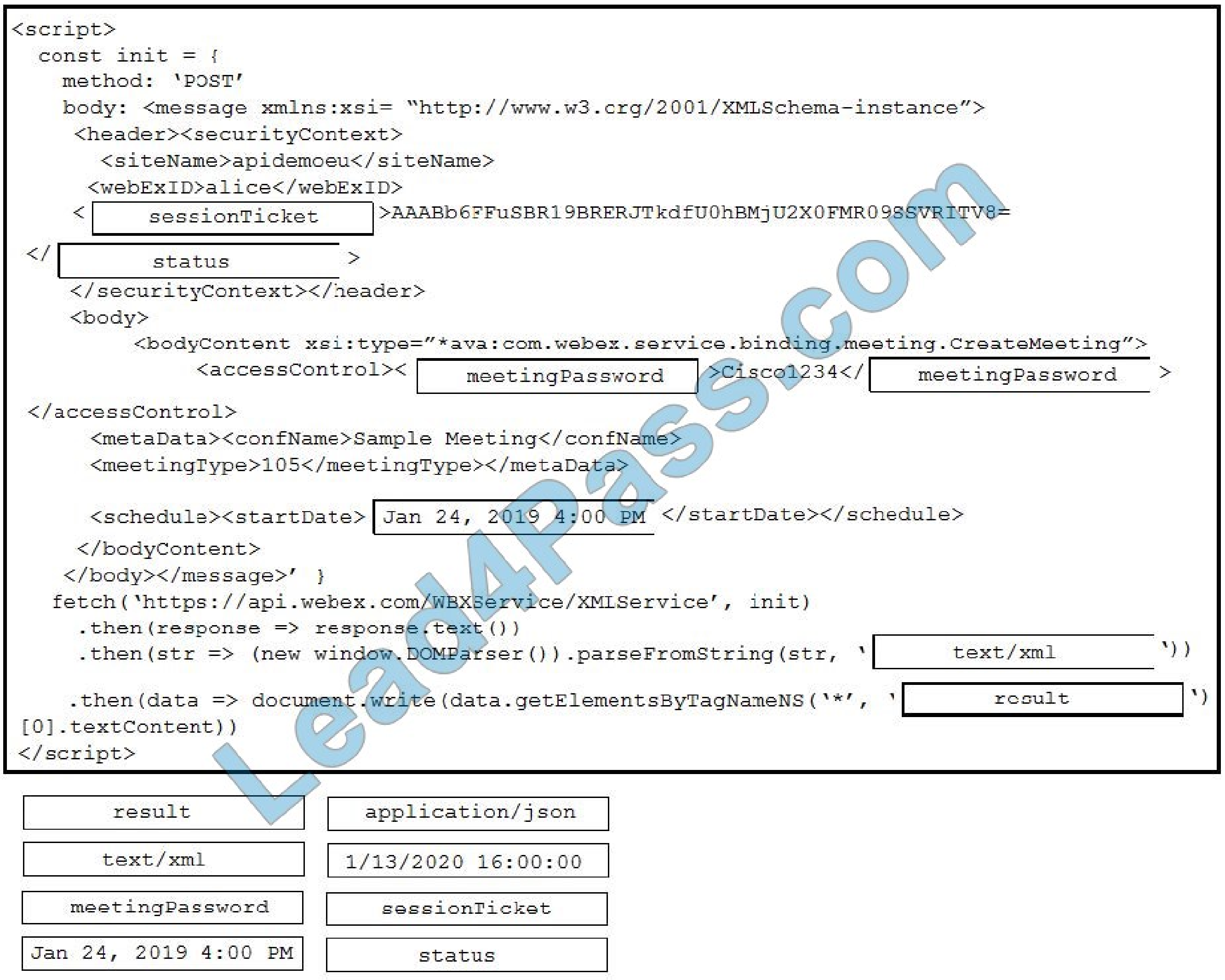
```
<script>
  const init = {
    method: 'POST'
    body: <message xmlns:xsi= "http://www.w3.crg/2001/XMLSchema-instance">
     <header><securityContext>
       <siteName>apidemoeu</siteName>
      <webExID>alice</webExID>
     <                    >AAABb6FFuSBR19BRERJTkdfU0hBMjU2X0FMR09SSVRITV8=
 </                    >
    </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.crg/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 5

Which element is needed to build a Web application that authenticates Webex users and can post messages under the user\\'s identity?

A. OAuth integration configured with the `messages_write\\' scope

B. bot access token

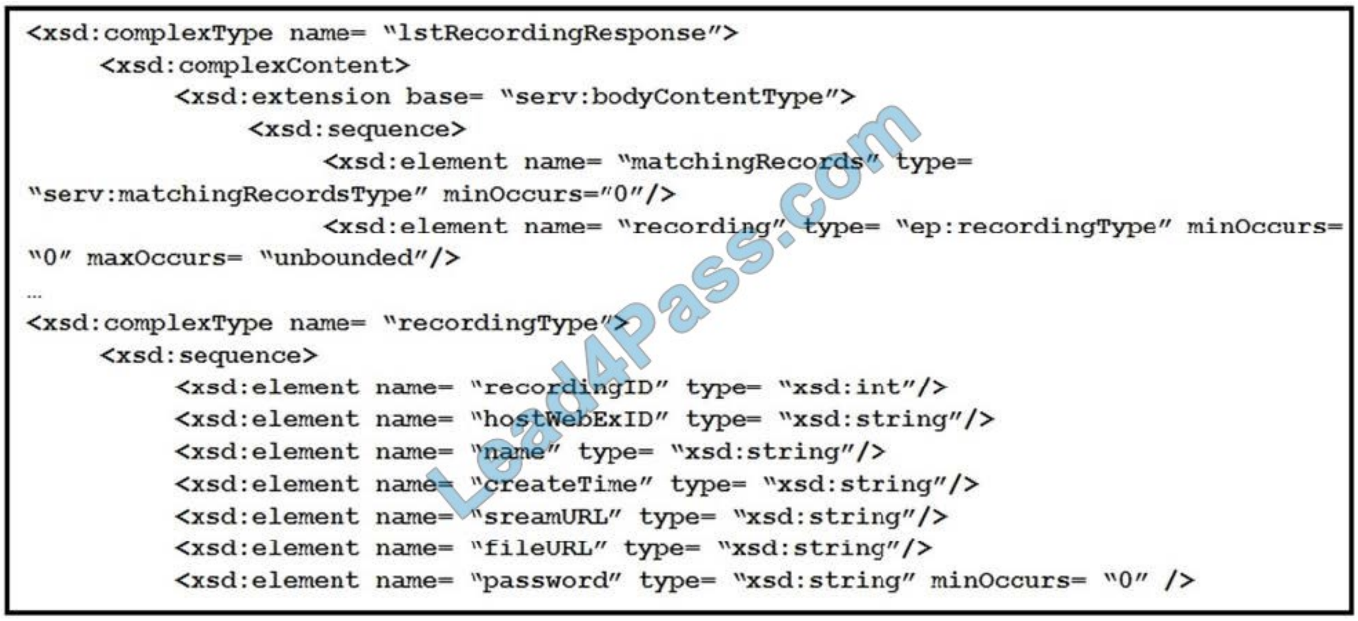
C. Guest Issuer application

D. self-signed certificate that is created from a public authority

Correct Answer: A

Reference: https://developer.webex.com/blog/real-world-walkthrough-of-building-an-oauth-webex-integration

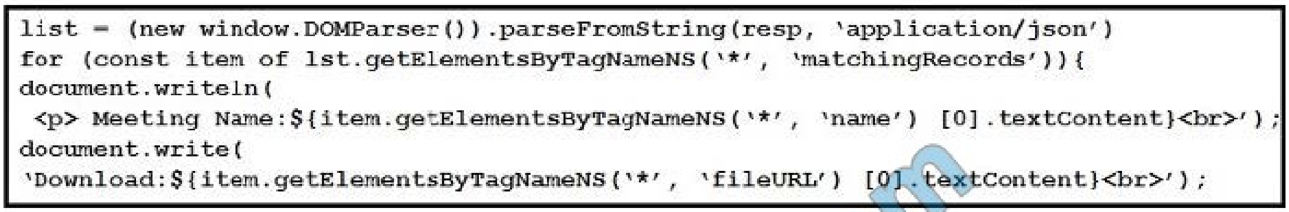
QUESTION 6



```
<xsd:complexType name= "lstRecordingResponse">
     <xsd:complexContent>
          <xsd:extension base= "serv:bodyContentType">
               <xsd:sequence>
                    <xsd:element name= "matchingRecords" type=
"serv:matchingRecordsType" minOccurs="0"/>
                    <xsd:element name= "recording" type= "ep:recordingType" minOccurs=
"0" maxOccurs= "unbounded"/>
...
<xsd:complexType name= "recordingType">
     <xsd:sequence>
          <xsd:element name= "recordingID" type= "xsd:int"/>
          <xsd:element name= "hostWebExID" type= "xsd:string"/>
          <xsd:element name= "name" type= "xsd:string"/>
          <xsd:element name= "createTime" type= "xsd:string"/>
          <xsd:element name= "sreamURL" type= "xsd:string"/>
          <xsd:element name= "fileURL" type= "xsd:string"/>
          <xsd:element name= "password" type= "xsd:string" minOccurs= "0" />
```

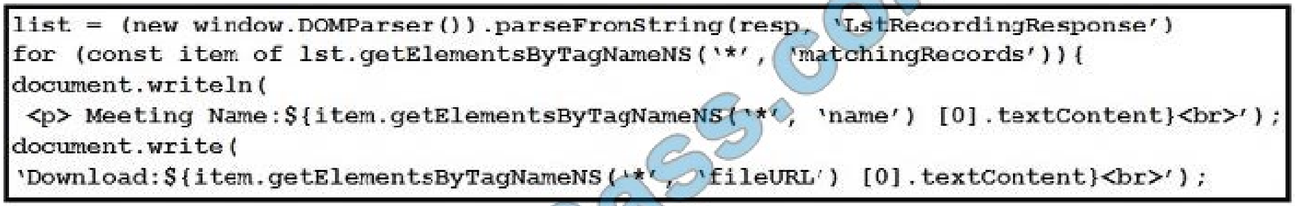
Refer to the exhibit. A snippet from the XSD schema of the Webex Meeting XML API `LstRecordingResponse\\' element is listed in the exhibit. Assuming that a variable named `resp\\' exists that contains the XML response from a successful `LstRecording\\' request, which code snippet correctly generates a simple report that lists meeting names and recording file download links?

A.



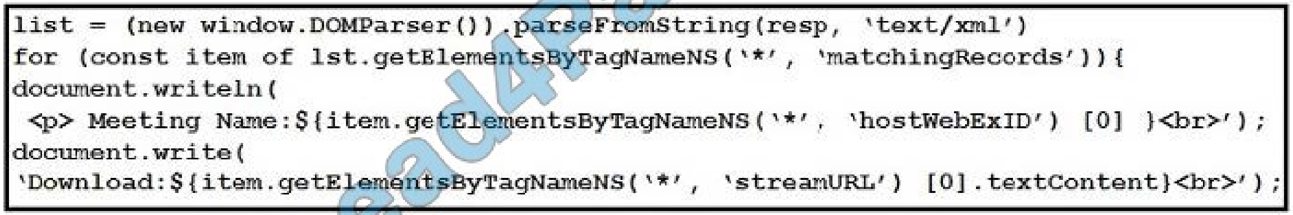
```
list = (new window.DOMParser()).parseFromString(resp, 'application/json')
for (const item of lst.getElementsByTagNameNS('*', 'matchingRecords')){
document.writeln(
 <p> Meeting Name:${item.getElementsByTagNameNS('*', 'name') [0].textContent}<br>');
document.write(
 'Download:${item.getElementsByTagNameNS('*', 'fileURL') [0].textContent}<br>');
```

B.



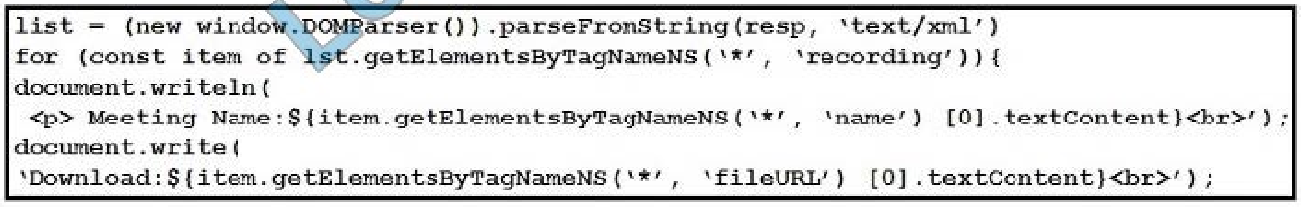
```
list = (new window.DOMParser()).parseFromString(resp, 'LstRecordingResponse')
for (const item of lst.getElementsByTagNameNS('*', 'matchingRecords')){
document.writeln(
 <p> Meeting Name:${item.getElementsByTagNameNS('*', 'name') [0].textContent}<br>');
document.write(
 'Download:${item.getElementsByTagNameNS('*', 'fileURL') [0].textContent}<br>');
```

C.



```
list = (new window.DOMParser()).parseFromString(resp, 'text/xml')
for (const item of lst.getElementsByTagNameNS('*', 'matchingRecords')){
document.writeln(
 <p> Meeting Name:${item.getElementsByTagNameNS('*', 'hostWebExID') [0] }<br>');
document.write(
 'Download:${item.getElementsByTagNameNS('*', 'streamURL') [0].textContent}<br>');
```

D.



```
list = (new window.DOMParser()).parseFromString(resp, 'text/xml')
for (const item of lst.getElementsByTagNameNS('*', 'recording')){
document.writeln(
 <p> Meeting Name:${item.getElementsByTagNameNS('*', 'name') [0].textContent}<br>');
document.write(
 'Download:${item.getElementsByTagNameNS('*', 'fileURL') [0].textContent}<br>');
```

A. Option A

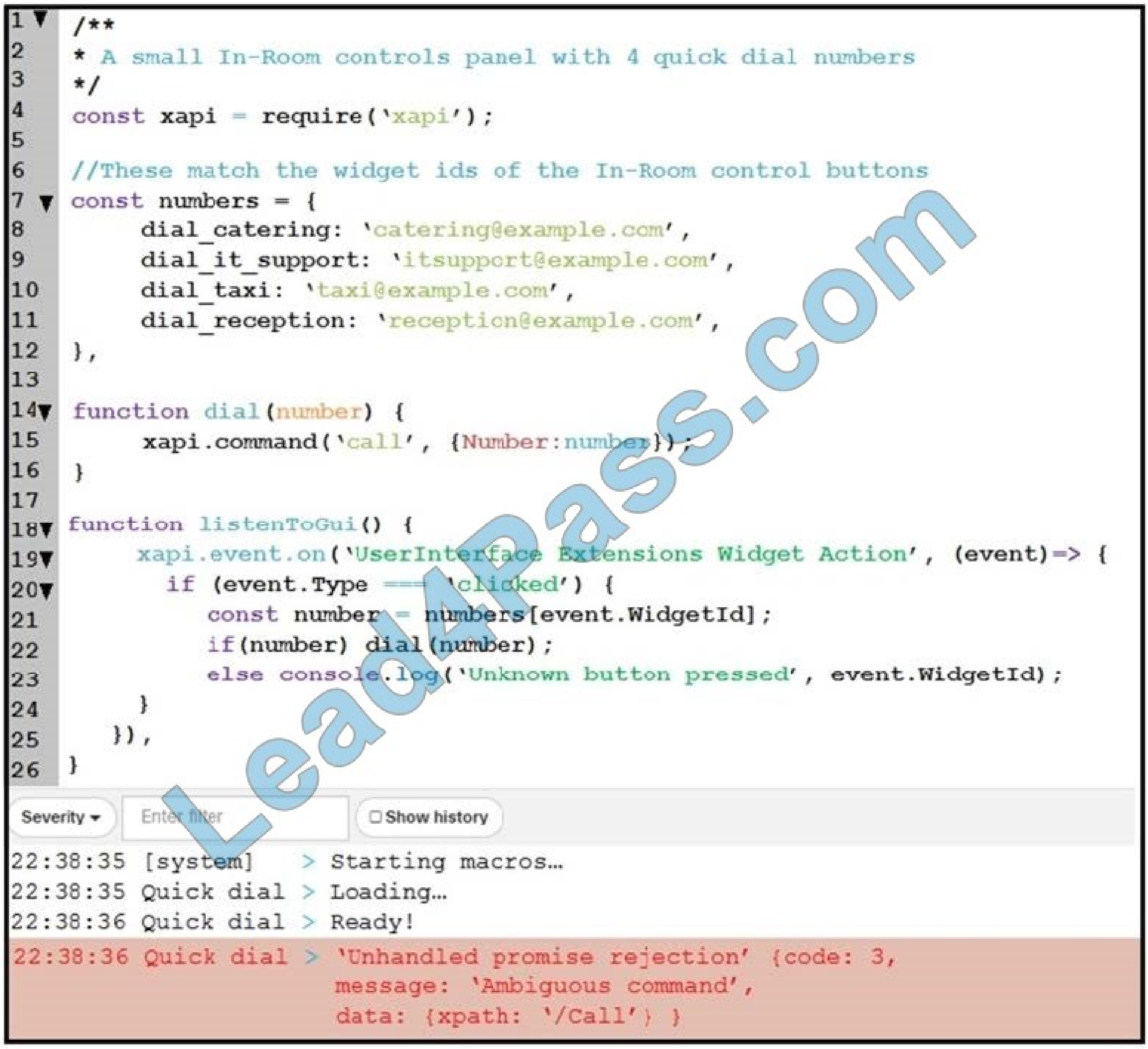
B. Option B

C. Option C

D. Option D

Correct Answer: A

QUESTION 7



```
1  /**
2  * A small In-Room controls panel with 4 quick dial numbers
3  */
4  const xapi = require('xapi');
5
6  //These match the widget ids of the In-Room control buttons
7  const numbers = {
8        dial_catering: 'catering@example.com',
9        dial_it_support: 'itsupport@example.com',
10       dial_taxi: 'taxi@example.com',
11       dial_reception: 'reception@example.com',
12 },
13
14 function dial(number) {
15       xapi.command('call', {Number:number});
16 }
17
18 function listenToGui() {
19       xapi.event.on('UserInterface Extensions Widget Action', (event)=> {
20         if (event.Type === 'clicked') {
21            const number = numbers[event.WidgetId];
22            if(number) dial(number);
23            else console.log('Unknown button pressed', event.WidgetId);
24         }
25      }),
26 }
```

```
22:38:35 [system]   > Starting macros…
22:38:35 Quick dial > Loading…
22:38:36 Quick dial > Ready!
22:38:36 Quick dial > 'Unhandled promise rejection' {code: 3,
                      message: 'Ambiguous command',
                      data: {xpath: '/Call'} }
```

Refer to the exhibit. An end user reports that the speed dial button is not working on their Webex Device, and when loading into the Macro Editor, this error was presented. On which line is the incorrect syntax?

A. line 4

B. line 14

C. line 15

D. line 22

Correct Answer: C

Reference: https://community.cisco.com/t5/telepresence-and-video/ce9-2-1-macro-framework-discussions/td-p/3220093

QUESTION 8

Which expression is a valid Webex Teams webhook filter?

A. personEmail=person@example.com+roomId=abc123

B. personEmail=person@example.com-roomId=abc123

C. personEmail=person@example.comandroomId=abc123

D. personEmail=person@example.com,roomId=abc123

Correct Answer: C

You can also use more than one filter in a webhook. To use multiple filters, combine them with the "and" symbol. For example, to create a webhook that only sends notifications when a specific person performs an action in a specific room, such as sending a message or creating a membership, combine the personEmail and roomId filters.

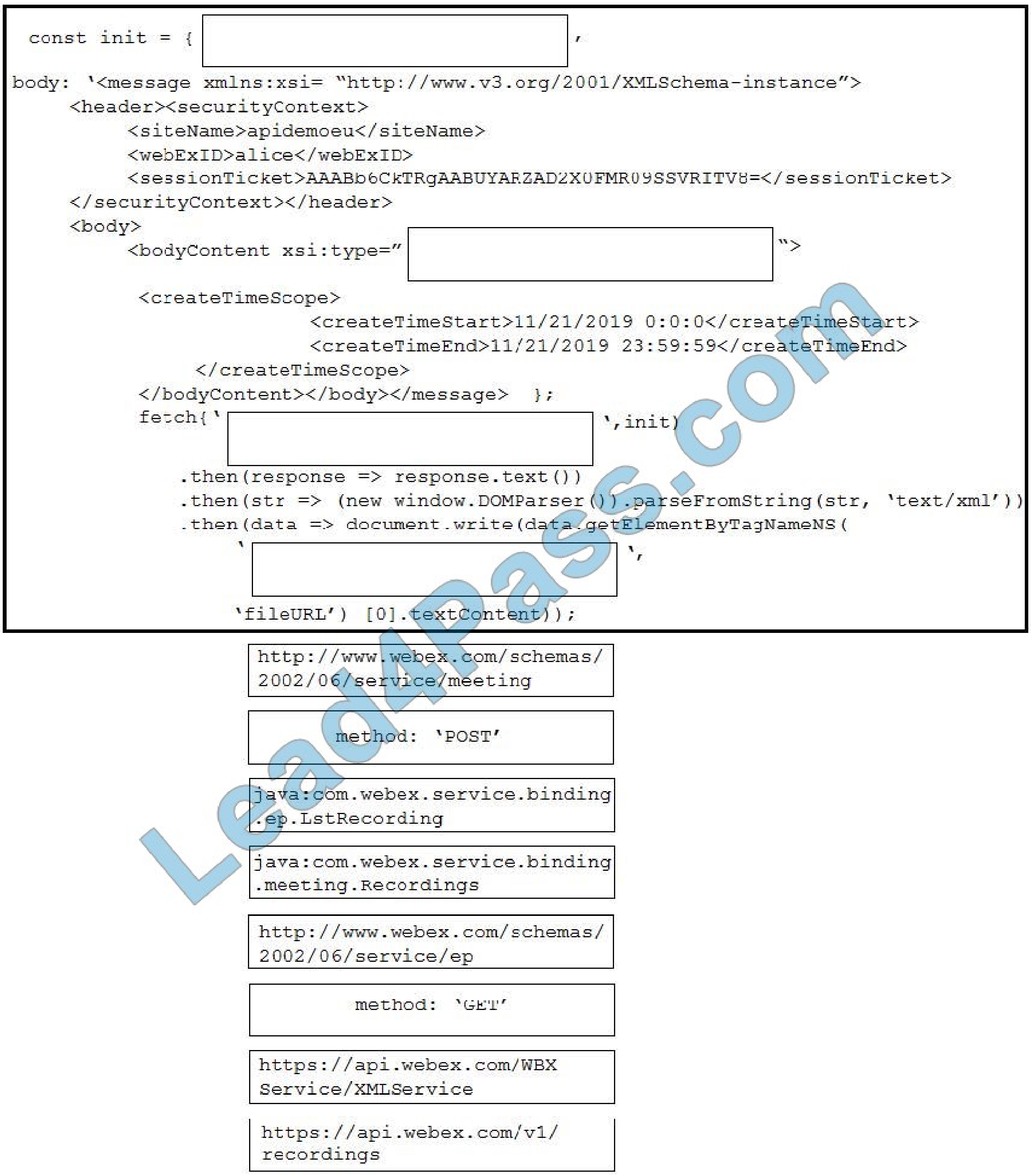
Reference: https://developer.webex.com/docs/api/guides/webhooks

QUESTION 9

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/WBX
Service/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));
```

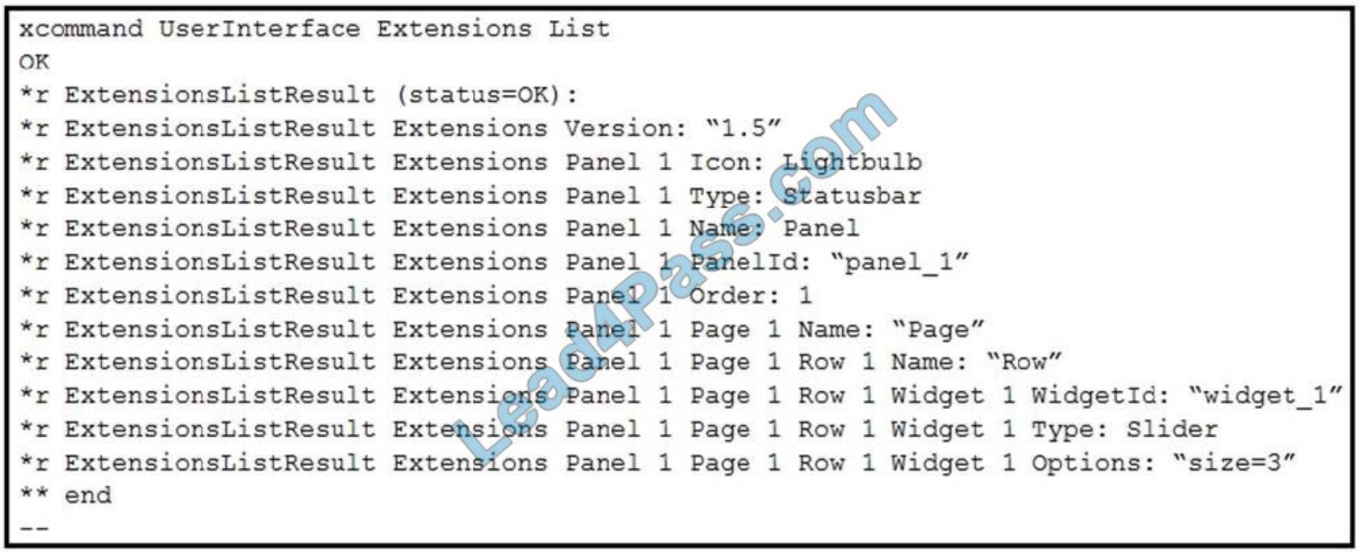
```
http://www.webex.com/schemas/
2002/06/service/meeting
```

```
java:com.webex.service.binding
.meeting.Recordings
```

```
method: 'GET'
```

```
https://api.webex.com/v1/
recordings
```

QUESTION 10



```
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

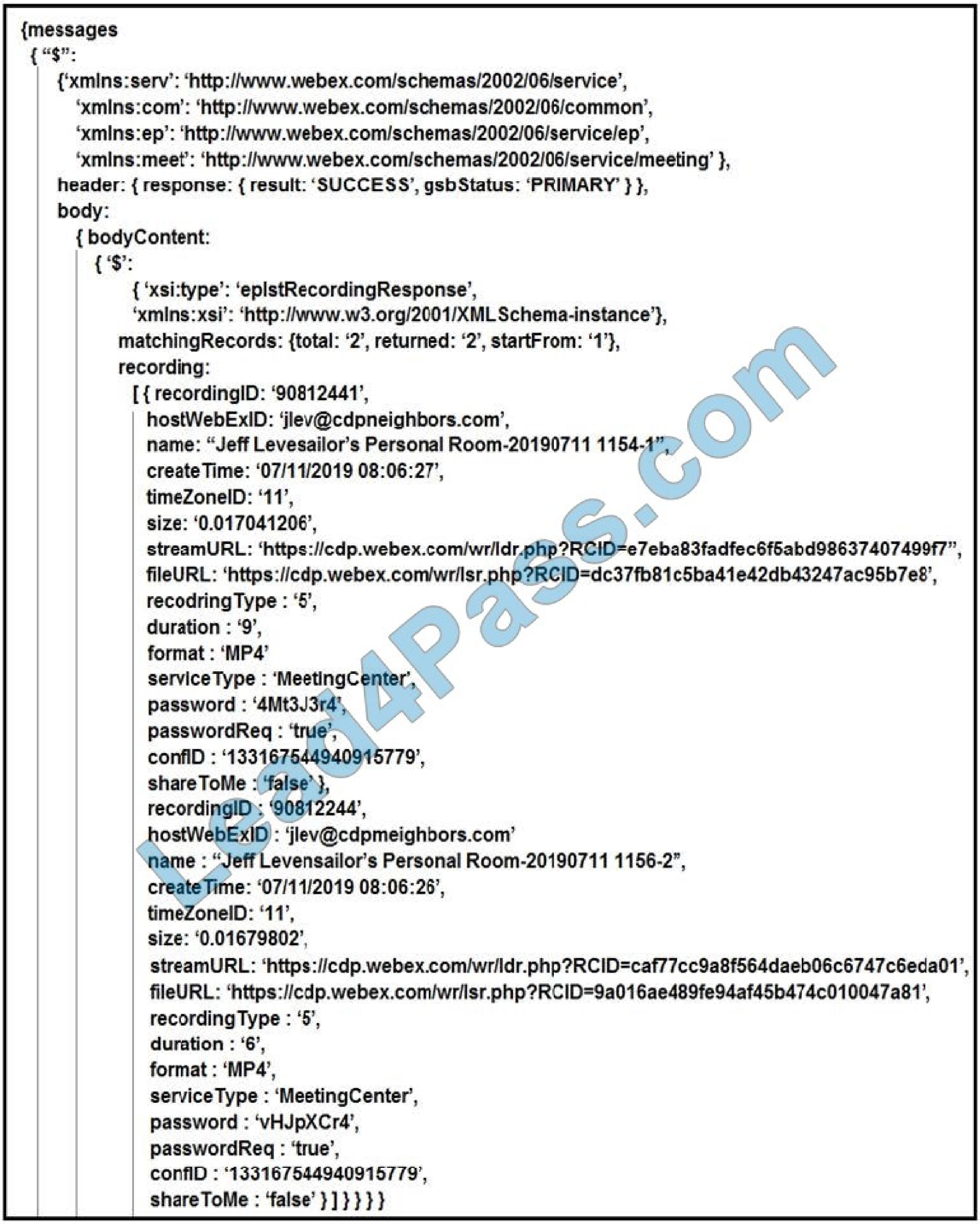
D. xcommand UserInterface Extensions Widget SetValue WidgetId: "widget_1" Value: 50

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

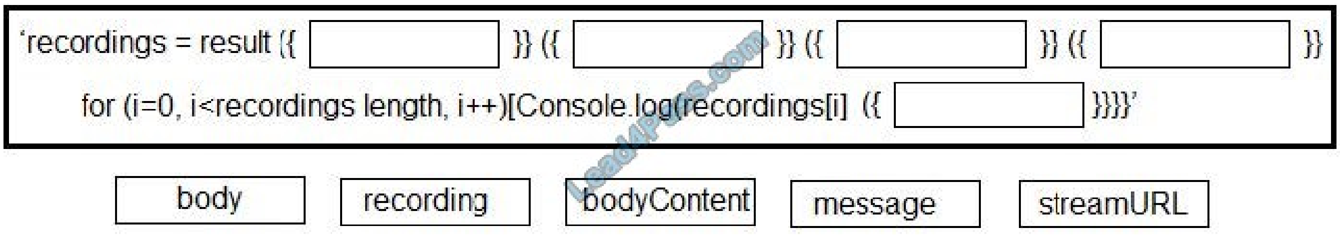
QUESTION 11

DRAG DROP Refer to the exhibit. A training coordinator must post links to Webex recordings on a company SharePoint site. This is usually a manual process, but a DevOps engineer wants to automate it using Webex XML APIs. After a sucessful LstRecording call wrapped in xml2js, the ‘console dir(result)’ output is shown in the exhibit. Using ‘dot notation’, drag and drop the code below onto the code snippet to output the streamURL for each recording.



```
{messages
 { "$":
    {'xmlns:serv': 'http://www.webex.com/schemas/2002/06/service',
      'xmlns:com': 'http://www.webex.com/schemas/2002/06/common',
      'xmlns:ep': 'http://www.webex.com/schemas/2002/06/service/ep',
      'xmlns:meet': 'http://www.webex.com/schemas/2002/06/service/meeting' },
    header: { response: { result: 'SUCCESS', gsbStatus: 'PRIMARY' } },
    body:
      { bodyContent:
        { '$':
            { 'xsi:type': 'eplstRecordingResponse',
            'xmlns:xsi': 'http://www.w3.org/2001/XMLSchema-instance'},
          matchingRecords: {total: '2', returned: '2', startFrom: '1'},
          recording:
            [ { recordingID: '90812441',
              hostWebExID: 'jlev@cdpneighbors.com',
              name: "Jeff Levesailor's Personal Room-20190711 1154-1",
              createTime: '07/11/2019 08:06:27',
              timeZoneID: '11',
              size: '0.017041206',
              streamURL: 'https://cdp.webex.com/wr/ldr.php?RCID=e7eba83fadfec6f5abd98637407499f7",
              fileURL: 'https://cdp.webex.com/wr/lsr.php?RCID=dc37fb81c5ba41e42db43247ac95b7e8',
              recodringType : '5',
              duration : '9',
              format : 'MP4'
              serviceType : 'MeetingCenter',
              password : '4Mt3J3r4',
              passwordReq : 'true',
              confID : '1331675449409l5779',
              shareToMe : 'false' },
              recordingID : '90812244',
              hostWebExID : 'jlev@cdpmeighbors.com'
              name : "Jeff Levensailor's Personal Room-20190711 1156-2",
              createTime: '07/11/2019 08:06:26',
              timeZoneID: '11',
              size: '0.01679802',
              streamURL: 'https://cdp.webex.com/wr/ldr.php?RCID=caf77cc9a8f564daeb06c6747c6eda01',
              fileURL: 'https://cdp.webex.com/wr/lsr.php?RCID=9a016ae489fe94af45b474c010047a81',
              recordingType : '5',
              duration : '6',
              format : 'MP4',
              serviceType : 'MeetingCenter',
              password : 'vHJpXCr4',
              passwordReq : 'true',
              confID : '133167544940915779',
              shareToMe : 'false' } ] } } } }
```

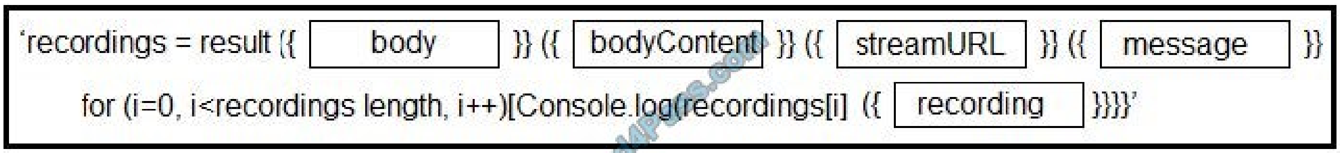
Select and Place:



```
'recordings = result ({ [        ] }} ({ [        ] }} ({ [        ] }} ({ [        ] }}
    for (i=0, i<recordings length, i++)[Console.log(recordings[i] ({ [        ] }}}}'
```

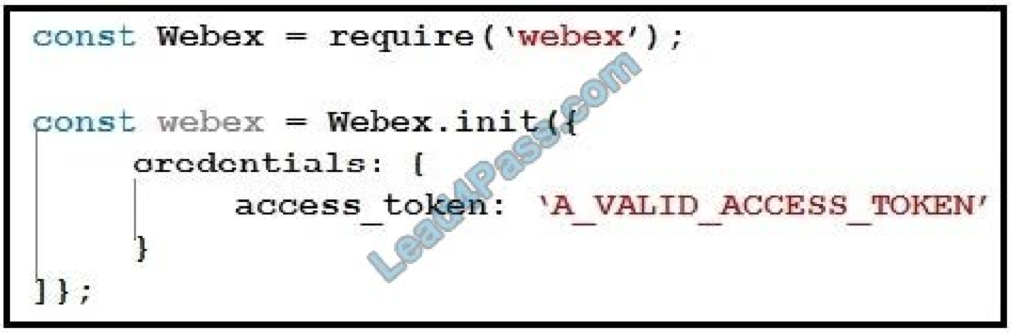
body | recording | bodyContent | message | streamURL

Correct Answer:



```
'recordings = result ({ [ body ] }} ({ [ bodyContent ] }} ({ [ streamURL ] }} ({ [ message ] }}
    for (i=0, i<recordings length, i++)[Console.log(recordings[i] ({ [ recording ] }}}}'
```

QUESTION 12



```
const Webex = require('webex');

const webex = Webex.init({
    credentials: {
        access_token: 'A_VALID_ACCESS_TOKEN'
    }
]};
```

Refer to the exhibit. When using the Webex Browser SDK to create calls and share screens, which two statements are valid given a `webex\\' object such as displayed in the exhibit? (Choose two.)

A. After a meeting is joined, it cannot be left programmatically until the host ends the meeting.

B. The webex meetings.register() function must be invoked before attempting to join any meeting.

C. The joinMeeting() function throws an error of type `media stopped\\' if a media stream is stopped.

D. Given a Webex meeting number the webex meetings join() function can be used to join the meeting.

E. The mediaSettings for a joined meeting accepts boolean attributes to send and receive audio, video, and screen share.

Correct Answer: AB

Reference: https://developer.webex.com/docs/sdks/browser

QUESTION 13

Which xAPI access mechanism requires separate connections for commands and notifications?

A. Serial

B. WebSocket

C. HTTP/HTTPS

D. SSH

Correct Answer: D

Reference: https://github.com/CiscoDevNet/labs-xapi/blob/master/labs/collab-xapi-intro/5.md

QUESTION 14

With CE 9.8 and above, which two statements are correct when an application is sending and receiving data over a connection established with an xAPI interface? (Choose two.)

A. All Serial, SSH, and WebSockets can be used to send and receive data.

B. HttpClient can be used to send requests but not receive responses.

C. HttpFeedback is the only option to receive data.

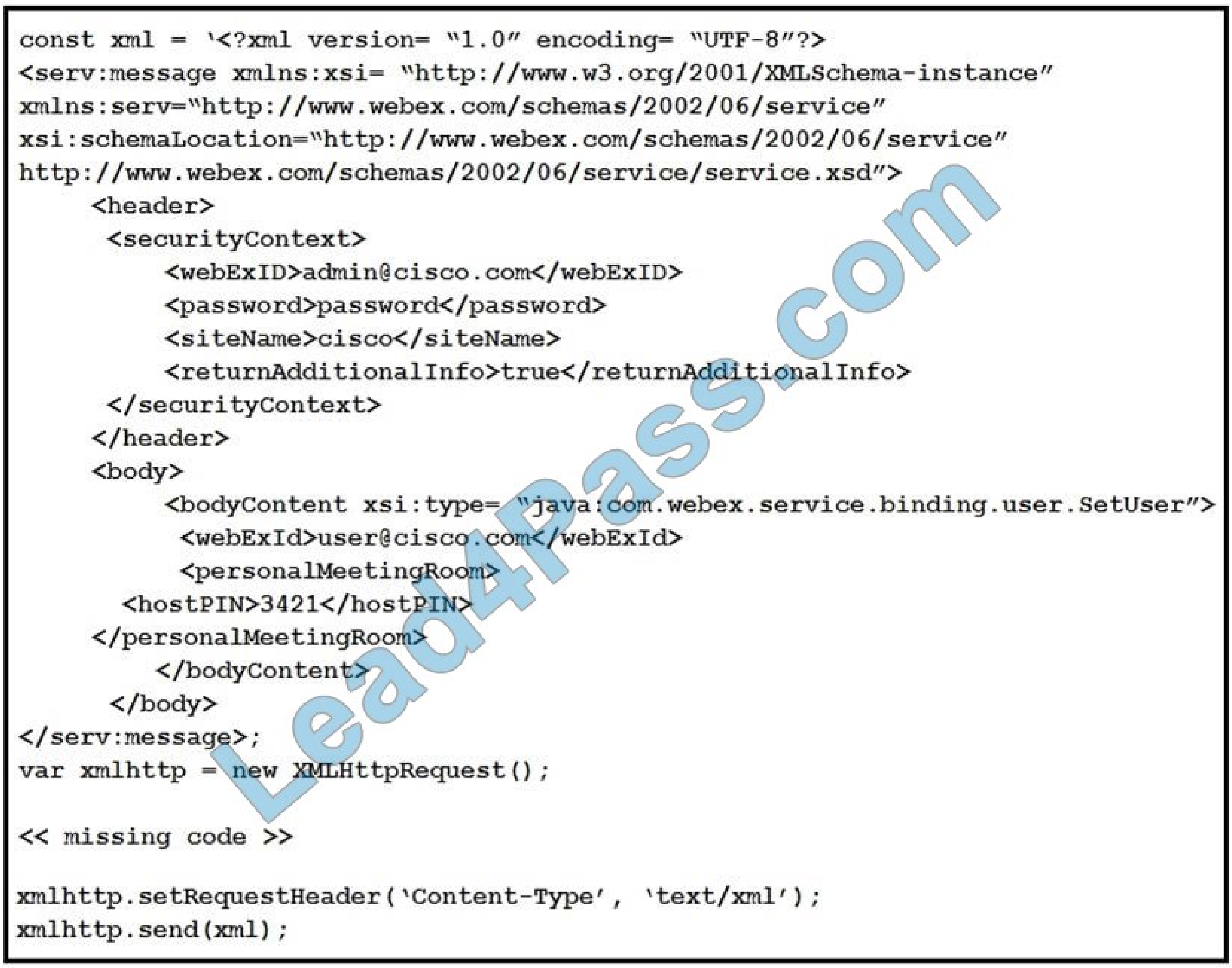
D. The HttpClient command can be used to send requests and read responses over HTTP.

E. WebSockets is the only option to send and receive data.

Correct Answer: BD

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 15



```
const xml = '<?xml version= "1.0" encoding= "UTF-8"?>
<serv:message xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
xmlns:serv="http://www.webex.com/schemas/2002/06/service"
xsi:schemaLocation="http://www.webex.com/schemas/2002/06/service"
http://www.webex.com/schemas/2002/06/service/service.xsd">
     <header>
      <securityContext>
          <webExID>admin@cisco.com</webExID>
          <password>password</password>
          <siteName>cisco</siteName>
          <returnAdditionalInfo>true</returnAdditionalInfo>
      </securityContext>
     </header>
     <body>
          <bodyContent xsi:type= "java:com.webex.service.binding.user.SetUser">
           <webExId>user@cisco.com</webExId>
           <personalMeetingRoom>
       <hostPIN>3421</hostPIN>
     </personalMeetingRoom>
         </bodyContent>
      </body>
</serv:message>;
var xmlhttp = new XMLHttpRequest();

<< missing code >>

xmlhttp.setRequestHeader('Content-Type', 'text/xml');
xmlhttp.send(xml);
```

Refer to the exhibit. A developer must construct an HTTP Request to use the XML API to set a Personal Meeting Room PIN for a given user. Which code completes the code to create the request?

A. xmlhttp.open("GET", "https://cisco.webex.com/WBXService/XMLService");

B. xmlhttp.open("PATCH", "https://cisco.webex.com/WBXService/XMLService");

C. xmlhttp.open("PUT", "https://cisco.webex.com/WBXService/XMLService");

D. xmlhttp.open("POST", "https://cisco.webex.com/WBXService/XMLService");

Correct Answer: D

The post method can be used for HTTP request that sets up a personal metting room PIN for a user.