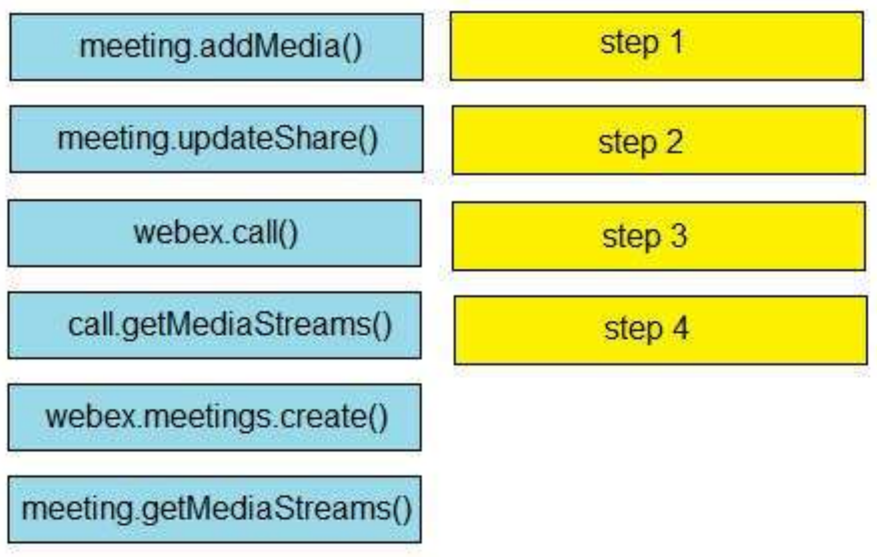
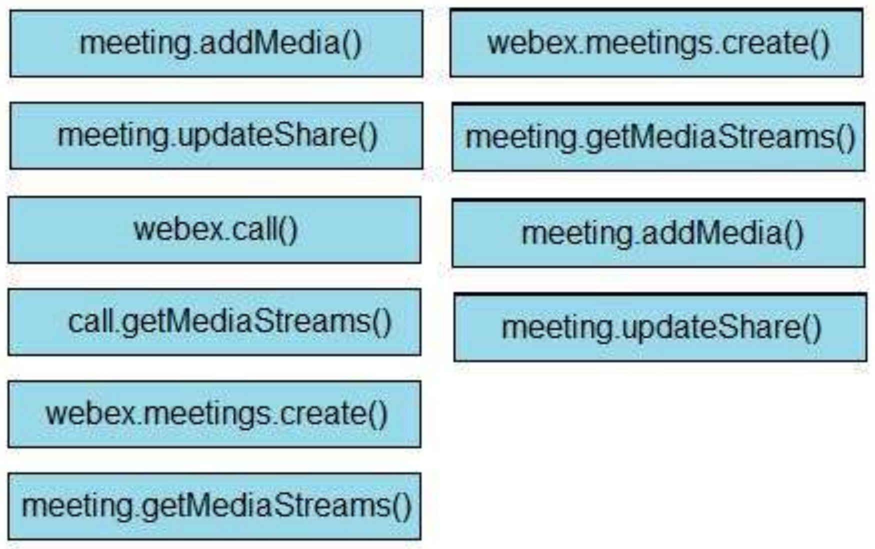
QUESTION 34
DRAG DROP

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:

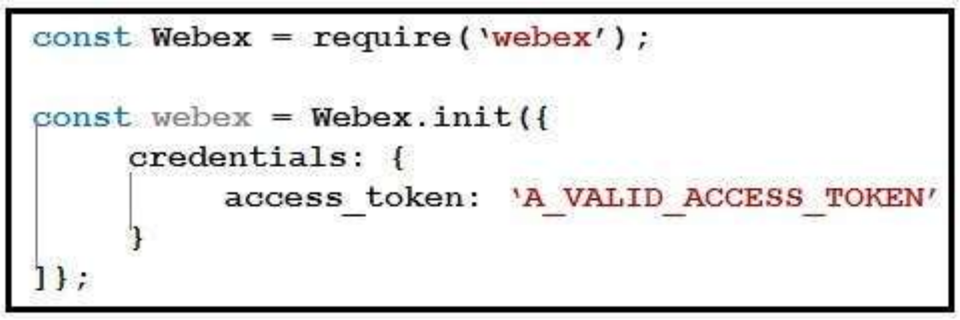


meeting.addMedia()	webex.meetings.create()
meeting.updateShare()	meeting.getMediaStreams()
webex.call()	meeting.addMedia()
call.getMediaStreams()	meeting.updateShare()
webex.meetings.create()	
meeting.getMediaStreams()	

Explanation

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

QUESTION 35



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

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

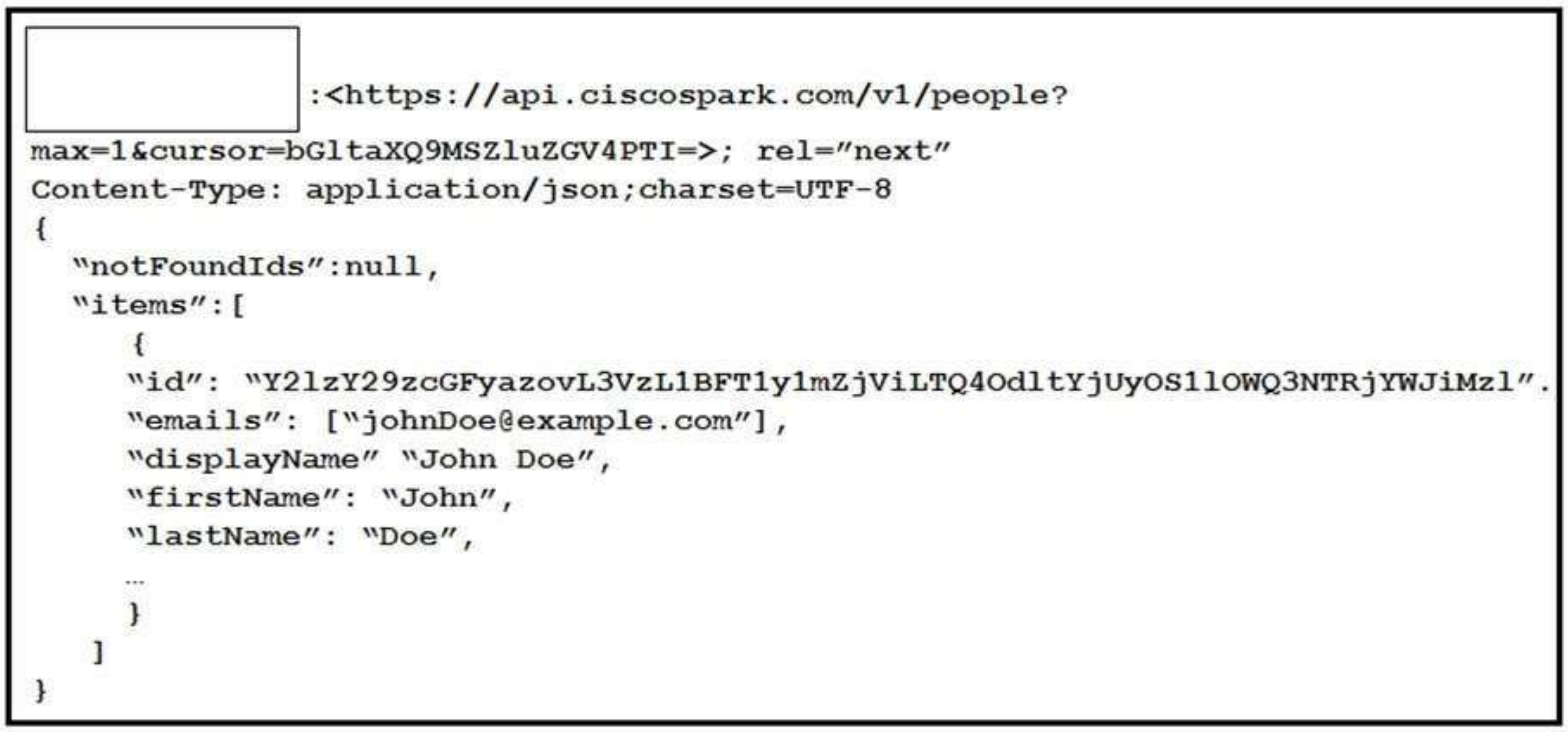
Refer to the exhibit. When using the Webex Browser SDK to create calls and share screens, which two statements are valid given a 'webex' object such as displayed in the exhibit? (Choose two.)

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
Explanation

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

QUESTION 36



```
[          ]:<https://api.ciscospark.com/v1/people?
max=1&cursor=bGltaXQ9MSZluZGV4PTI=>; rel="next"
Content-Type: application/json;charset=UTF-8
{
  "notFoundIds":null,
  "items":[
     {
     "id": "Y2lzY29zcGFyazovL3VzL1BFT1y1mZjViLTQ4OdltYjUyOS1lOWQ3NTRjYWJiMzl".
     "emails": ["johnDoe@example.com"],
     "displayName" "John Doe",
     "firstName": "John",
     "lastName": "Doe",
     ...
     }
   ]
}
```

Refer to the exhibit. A Webex Teams REST API response is shown with the HTTP Header missing. Which HTTP header expected in this response?

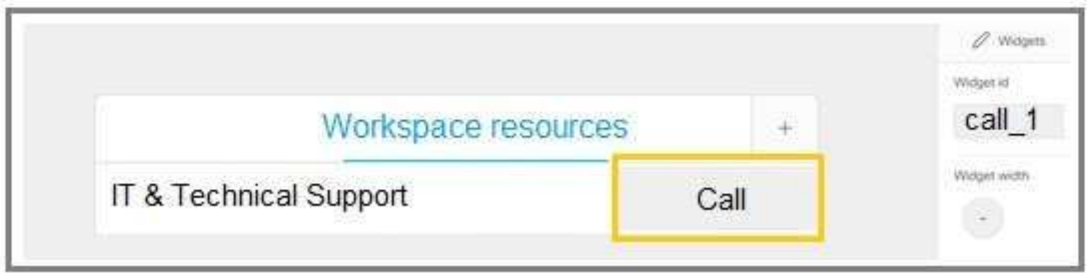
A. Push
B. Link
C. Patch
D. Put

Correct Answer: B

Explanation

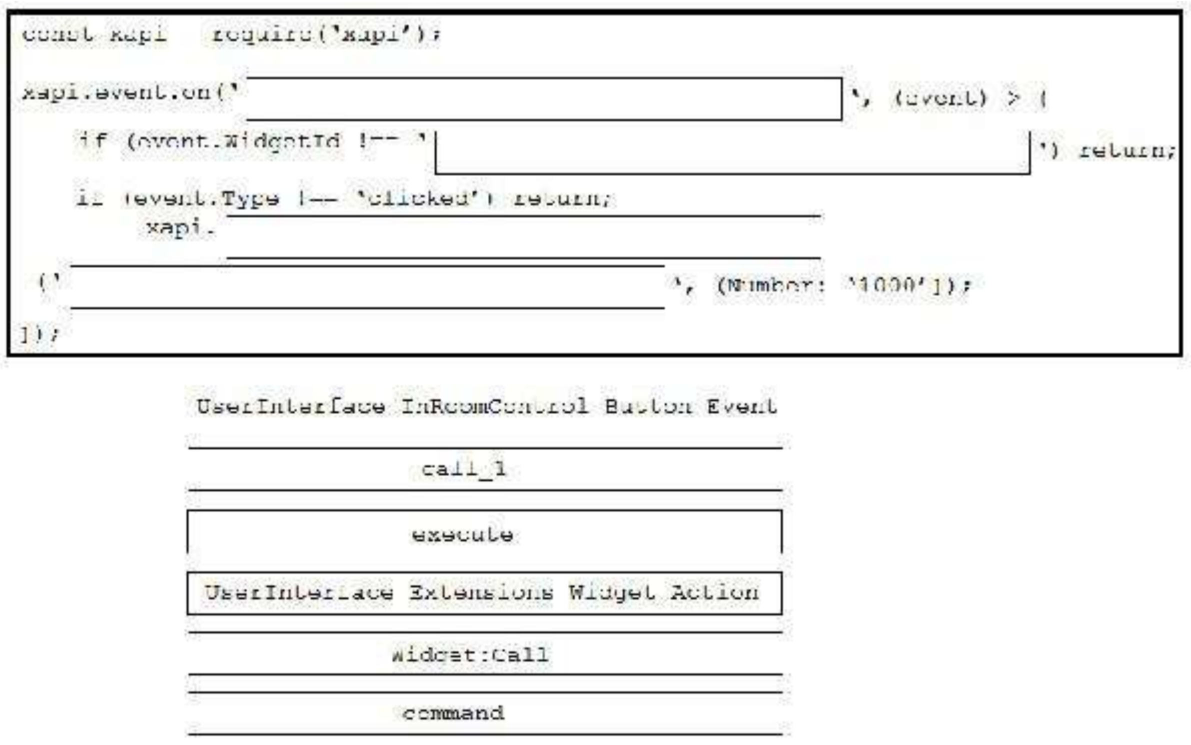
Explanation/Reference:
Reference: https://developer-portal-intb.ciscospark.com/docs/api/basics

QUESTION 37
DRAG DROP



Refer to the exhibit. A Webex device In-Room Control editor screenshot and associated Macro code is shown. Drag and drop the code snippets to complete the JavaScript Macro that launches a call when the Call button on the custom control panel is touched. Not all options are used.

Select and Place:



```
const xapi = require('xapi');

xapi.event.on('                              ', (event) => {
    if (event.WidgetId !== '                              ') return;
    if (event.Type !== 'clicked') return;
      xapi.
                              
  ('                              ', {Number: '1000'});
});
```

UserInterface InRoomControl Button Event

call_1

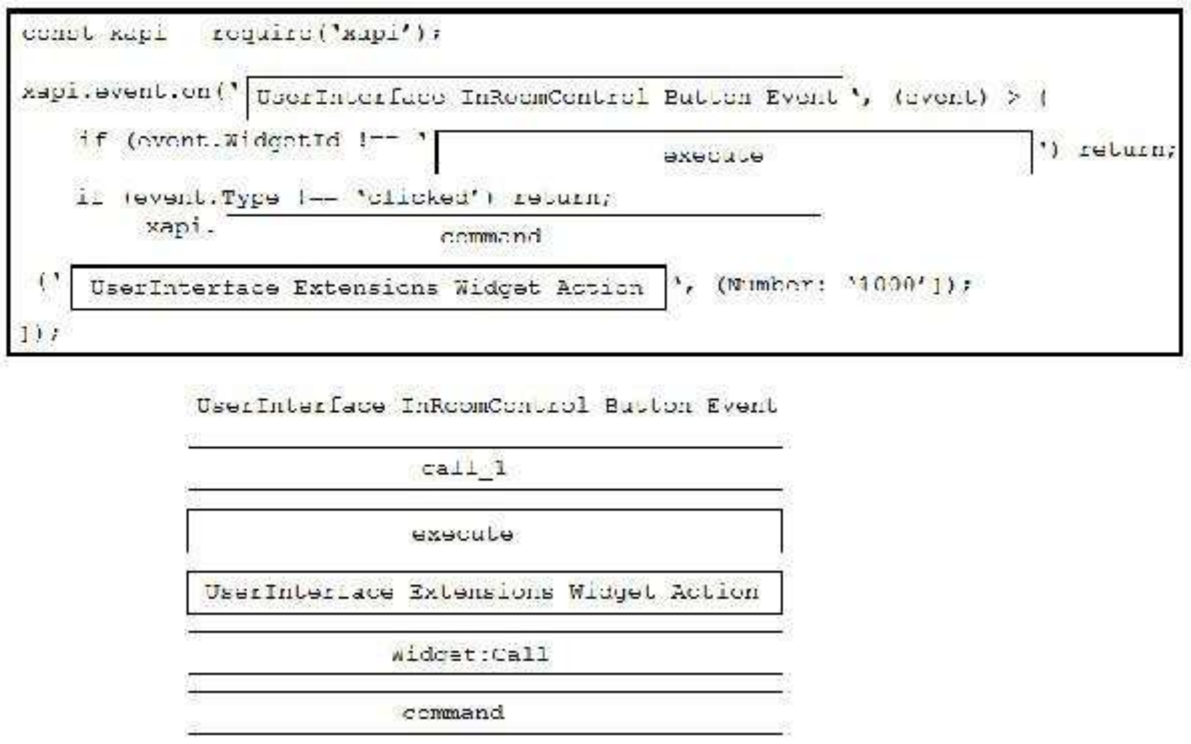
execute

UserInterface Extensions Widget Action

widget:Call

command

Correct Answer:



```
const xapi = require('xapi');

xapi.event.on('UserInterface InRoomControl Button Event', (event) => {
    if (event.WidgetId !== '          execute          ') return;
    if (event.Type !== 'clicked') return;
      xapi.
                      command
  ('UserInterface Extensions Widget Action', {Number: '1000'});
});
```

UserInterface InRoomControl Button Event

call_1

execute

UserInterface Extensions Widget Action

widget:Call

command

Explanation

Explanation/Reference:
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 38
A company wants to adopt Webex Teams as a messaging platform and use REST APIs to automate the creation of teams and rooms. Which sequence of REST API requests is needed to create and populate a new Webex team and create a populated Webex room for the team?

A. POST /teams, POST /memberships, POST /rooms
B. POST /teams, POST /people, POST /rooms
C. POST /teams, POST /team/memberships, POST /rooms
D. POST /teams, POST /team/memberships, POST /rooms, POST /memberships

Correct Answer: B
Explanation

Explanation/Reference:
Reference: https://developer.webex.com/docs/api/basics

QUESTION 39
Which Webex Teams webhook resource type indicates that a user interacted with a card?

A. buttonActions
B. attachmentActions
C. webhookCardActions
D. cardActions

Correct Answer: B
Explanation

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

QUESTION 40
DRAG DROP

Drag and drop the code segments to construct a script that carries out these functions:
- adds a new room
- adds two users, Bob, and Alice, to the room
- when both the users have been added, sends a welcome message

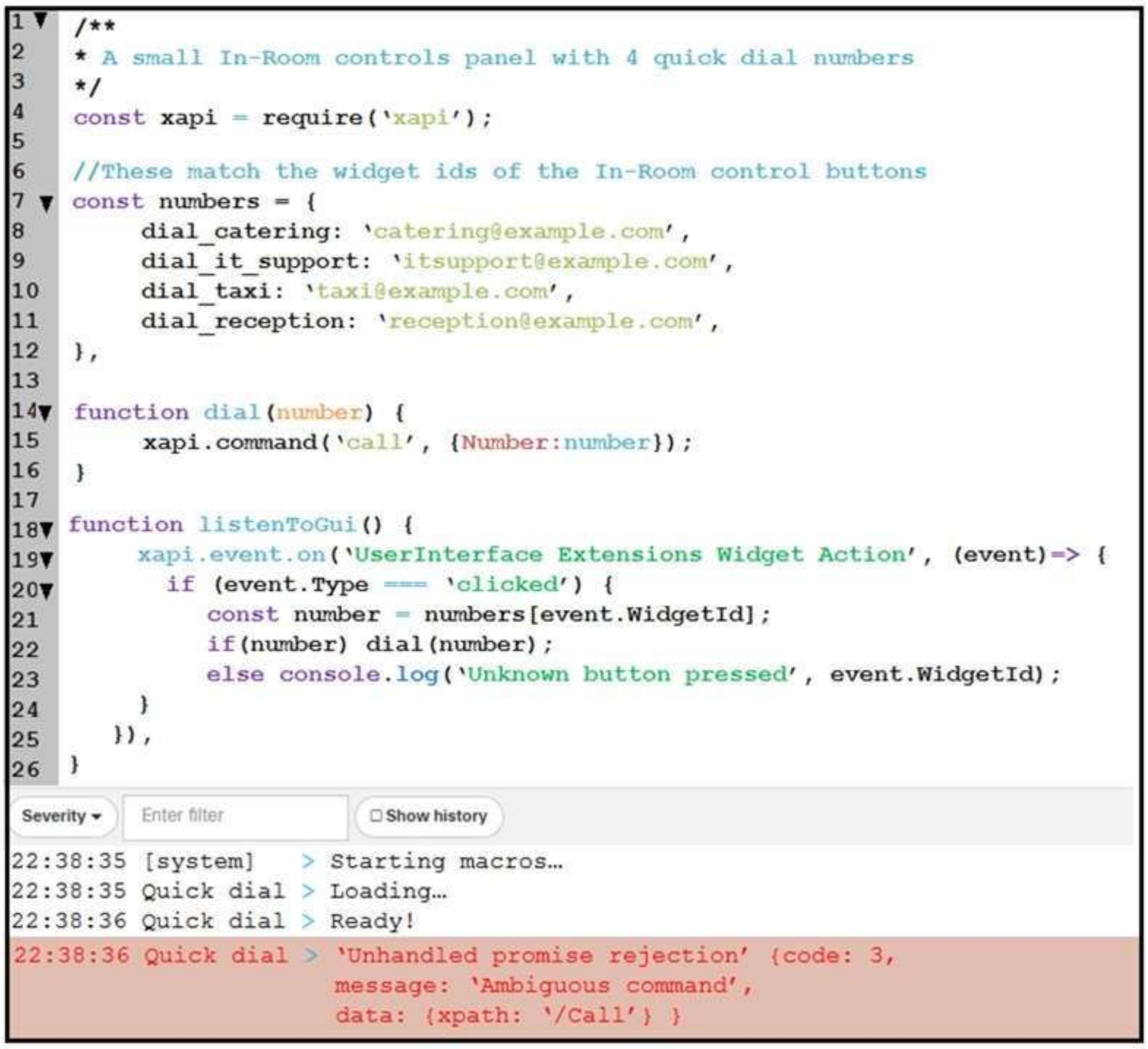
Select and Place:

Correct Answer:

Explanation

Explanation/Reference:
Reference: https://github.com/webex/webex-js-sdk/tree/master/packages/node_modules/webex (see usage)

QUESTION 41



```
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 }
```

Severity ▾ | Enter filter | ☐ Show history

```
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

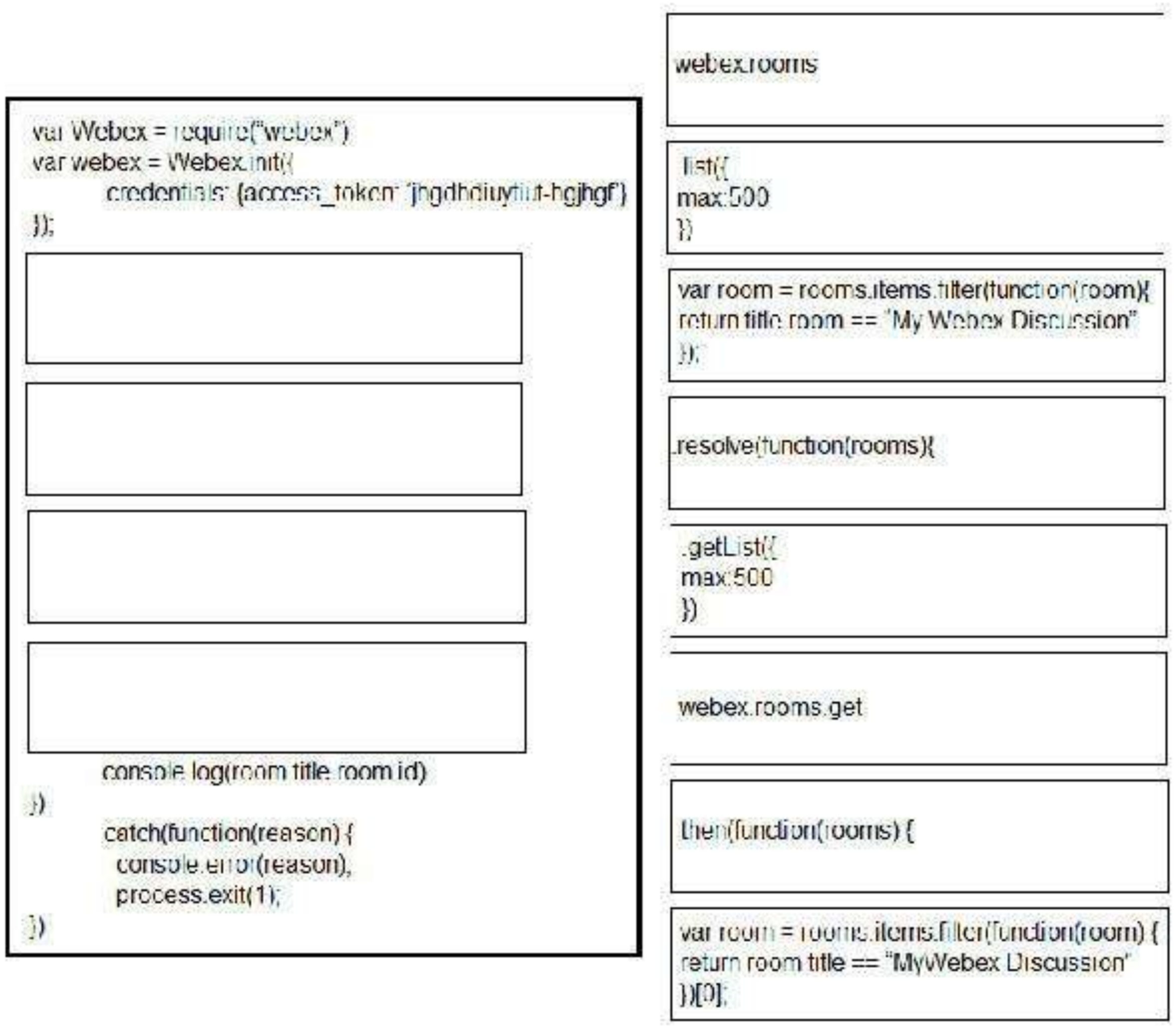
Correct Answer: C
Explanation

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

QUESTION 42
DRAG DROP

Drag and drop the code segments from the right of the screen into the targets on the left to create a script that retrieves a list of rooms using the Webex JavaScript API and outputs the title and ID to the console for one that matches the title "My Webex discussion". Not all code segments are used.

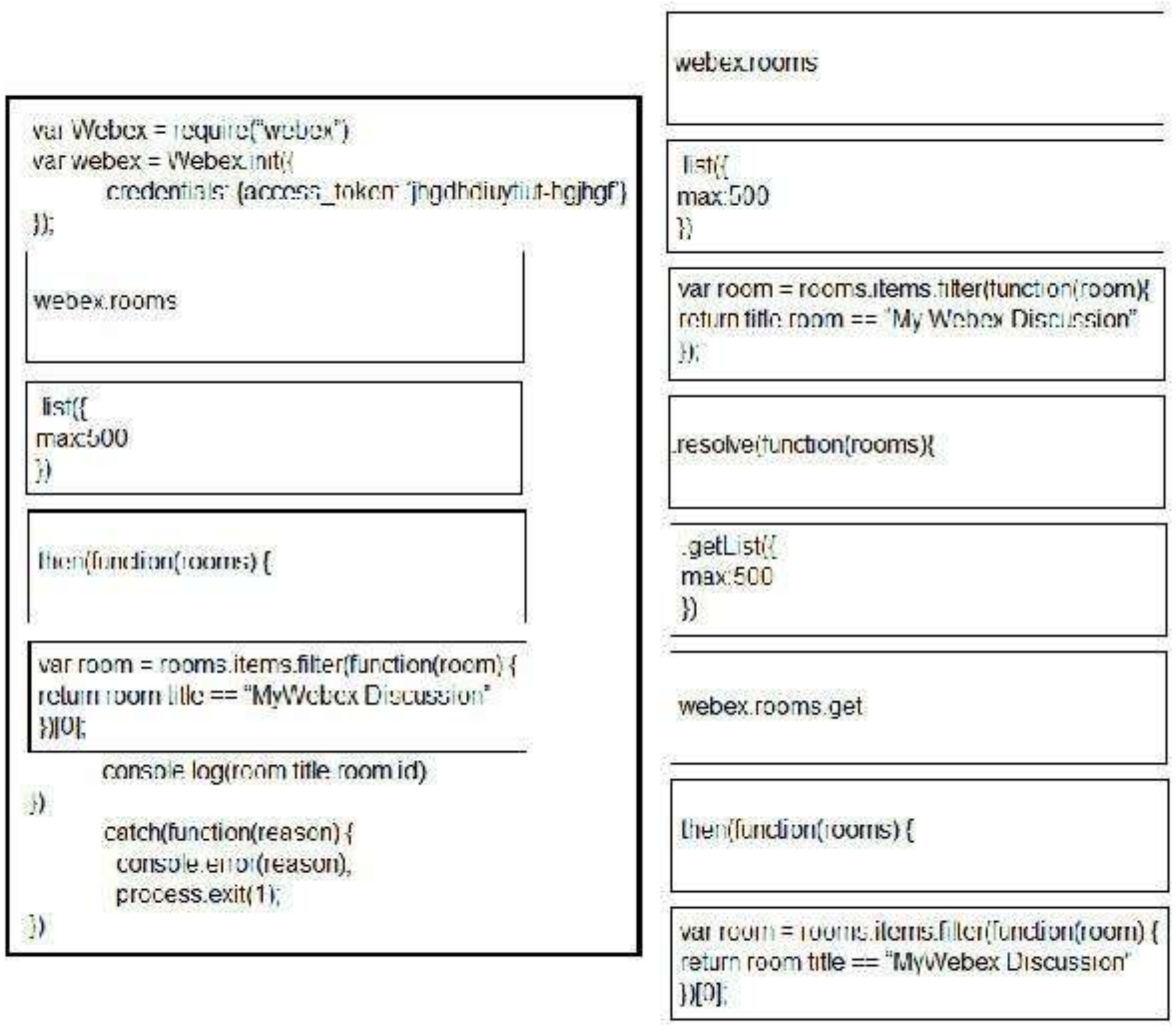
Select and Place:



```
var Webex = require("webex")
var webex = Webex.init({
        credentials: {access_token: 'jhgdhdiuytiut-hgjhgf'}
});
[          ]
[          ]
[          ]
[          ]
        console.log(room.title, room.id)
})
        catch(function(reason) {
          console.error(reason),
          process.exit(1);
})
```

```
webex.rooms

list({
max:500
})

var room = rooms.items.filter(function(room){
return title.room == "My Webex Discussion"
});

.resolve(function(rooms){

.getList({
max:500
})

webex.rooms.get

then(function(rooms) {

var room = rooms.items.filter(function(room) {
return room.title == "MyWebex Discussion"
})[0];
```

Correct Answer:



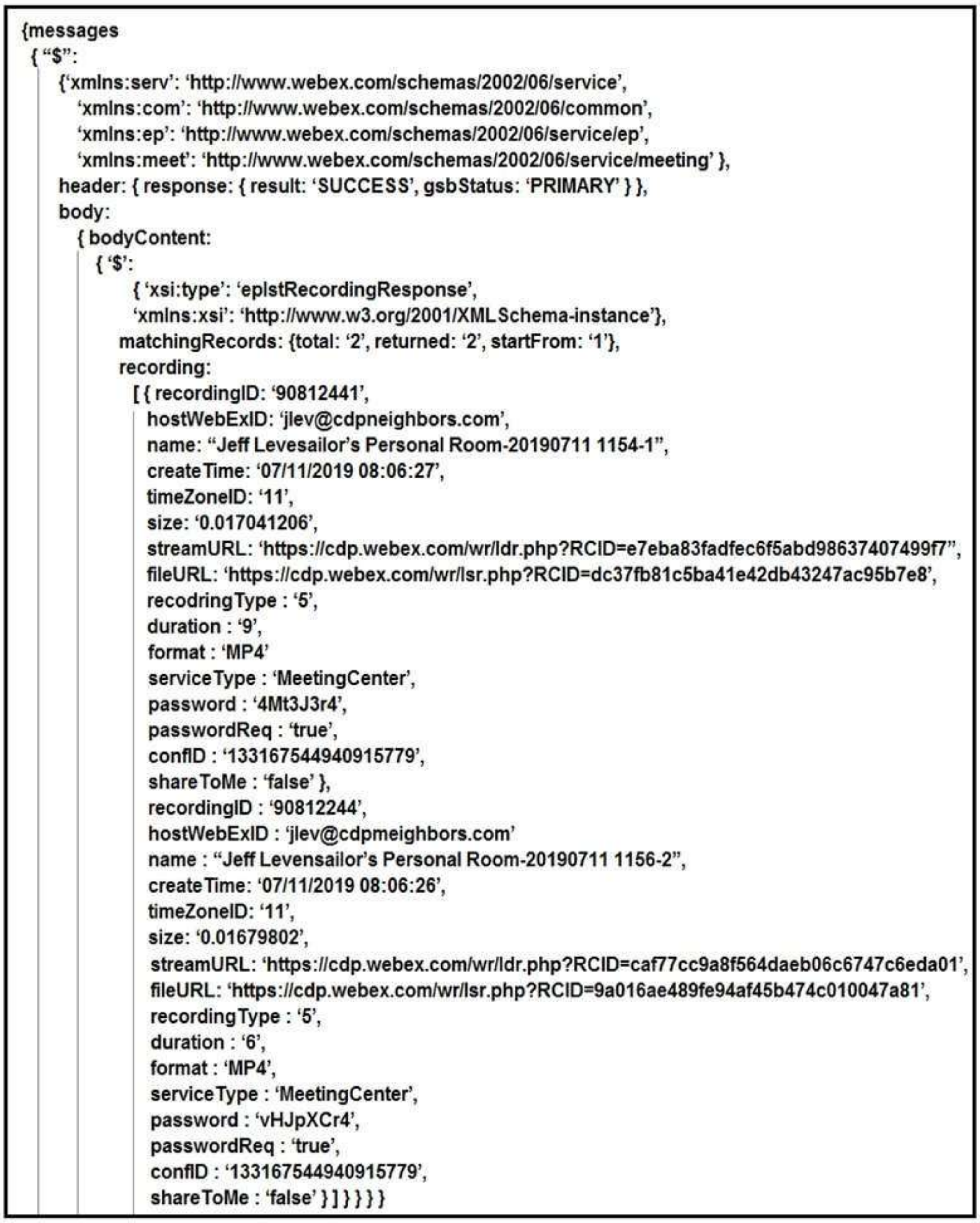
```
var Webex = require("webex")
var webex = Webex.init({
        credentials: {access_token: 'jhgdhdiuytiut-hgjhgf'}
});
webex.rooms
list({
max:500
})
then(function(rooms) {
var room = rooms.items.filter(function(room) {
return room.title == "MyWebex Discussion"
})[0];
        console.log(room.title, room.id)
})
        catch(function(reason) {
          console.error(reason),
          process.exit(1);
})
```

```
webex.rooms

list({
max:500
})

var room = rooms.items.filter(function(room){
return title.room == "My Webex Discussion"
});

.resolve(function(rooms){

.getList({
max:500
})

webex.rooms.get

then(function(rooms) {

var room = rooms.items.filter(function(room) {
return room.title == "MyWebex Discussion"
})[0];
```

Explanation

Explanation/Reference:
Reference: https://github.com/webex/webex-js-sdk/blob/master/README.md

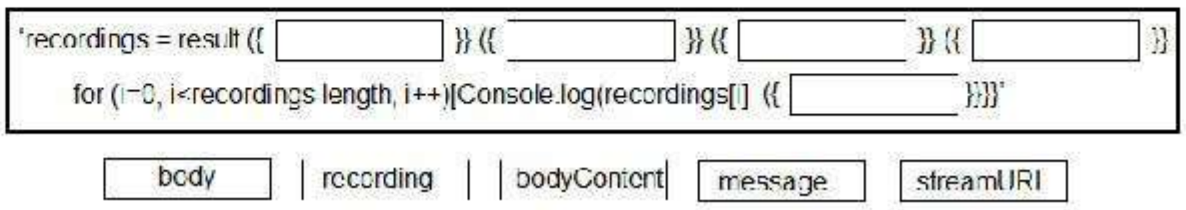
QUESTION 43
DRAG DROP



```
{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 : '133167544940915779',
              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' } ] } } } }
```

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.

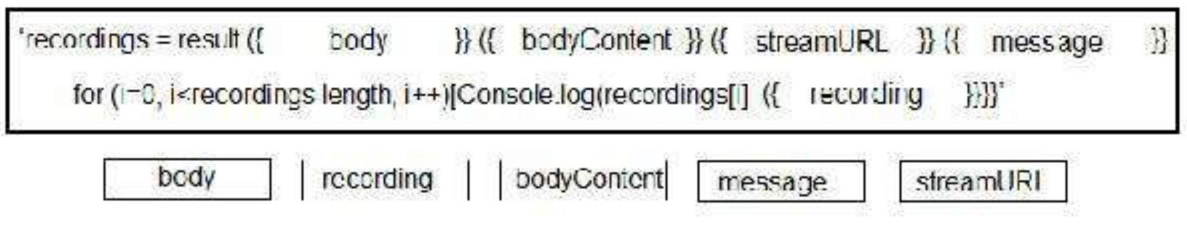
Select and Place:



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

body | recording | bodyContent | message | streamURI

Correct Answer:



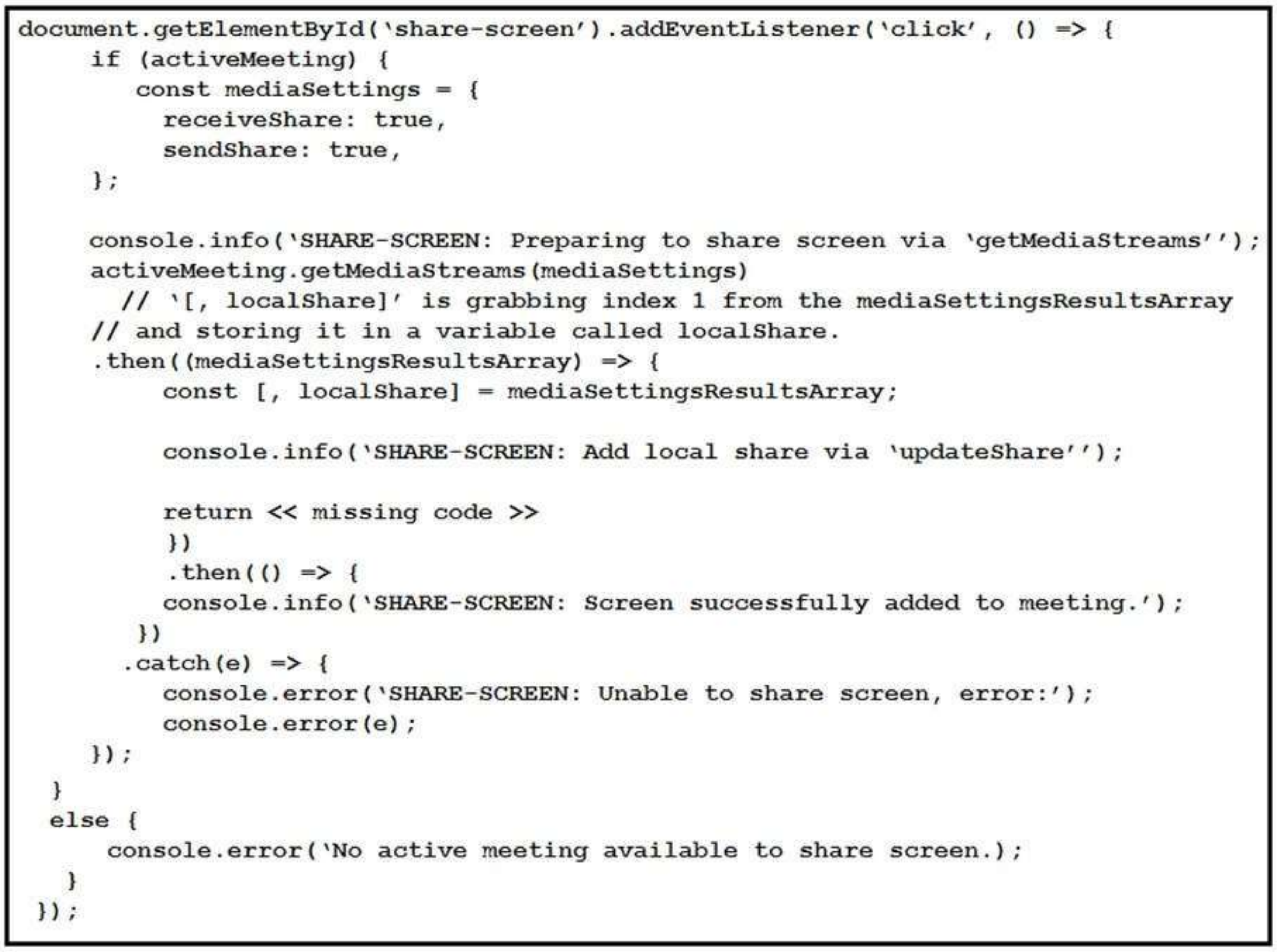
```
'recordings = result ({   body   }} ({  bodyContent  }} ({  streamURL  }} ({  message  }}
   for (i=0, i<recordings length, i++)[Console.log(recordings[] ({  recording  }}}}'
```

body | recording | bodyContent | message | streamURL

Explanation

Explanation/Reference:

QUESTION 44



```
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({ sen
   dShare: true receiveShare: true,
   stream: null
   })
B. activeMeeting.updateShare({ sen
   dShare: true receiveShare:
   false, stream: remoteShare
   })
C. activeMeeting.updateShare({ sen
   dShare: true receiveShare: true,
   stream: localShare
   })
D. activeMeeting.updateShare({ sen
   dShare: false receiveShare:
   false, stream: null
   })

Correct Answer: C
Explanation

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