Configure and Troubleshoot Guest and Host Access on CMS Spaces

Contents

Introduction

This document describes how to set up Guest and Host access on spaces of your Cisco Meeting Server (CMS) by using API commands.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Cisco Meeting Server (CMS) with spaces set up and able to make calls into it
- API client (like Poster, Postman) or
- CMS API guide

Components Used

The information in this document is based on CMS version 2.1

The information in this document was created from a device in a specific lab environment. If your

network is live, ensure that you understand the potential impact of any command.

Background Information

The document outlines types of scenarios:

- Different URIs or call-IDs are used by guest and host participants
- Same URI is used by guest and host participants where differentiation is made based on PIN or passcode entry (both non-empty)
- Same URI is used by guest and host participants where differentiation is made based on PIN or passcode entry (mix of empty/non-empty)
- A host user is a member of the space and authorized via webRTC log in, guest users join the meeting with callID. Same URI and callID is used by guest and host participants with empty or non-empty PIN/passcodes for guest users

Configure

There are four possibilities for differentiation between Guest and Host participants in CMS, described in the next 4 examples, and are mainly based on different **callLegProfiles** that determine the in-call behavior for those participants joining in on the space.

First, the method by using a different **URI** (or call-ID) for guest and host participants is explained, and afterwards that gets appended by using different passcodes (or timeout) on the same **URI**, to make the differentiation between guest and host participants. The third method of a timeout or empty PIN entry for Guest users was introduced as a new feature on CMS 2.1 as shown on section 2.4 of the release notes. The fourth method explains how to set up Guest and Host access on spaces with assigned owner/members and make the member of the space (owner) to be the host of the space.

1) Configuration using different URIs

This is the basic configuration that has been available before CMS 2.1 release and is the same as for a different call-ID. The next steps need to be performed to get the Guest/Hoost access differentiation on the same space:

1. Create a guest **callLegProfile** (**needsActivation** = **true**)
2. Create a host **callLegProfile** (**needsActivation** = **false**)
3. Assign the guest **callLegProfile** to an existing or new space (being the default access method)
4. Create a new **accessMethod** on that same space with a different **URI** (and call-ID) and assign the host **callLegProfile** to it

Step 1. Create a guest **callLegProfile** (**needsActivation** = **true**).

A **callLegProfile** determines the in-call behavior and by default you are assign the guest in-call behavior to the space so that you can later on have a different access method on that same space, as well for host to be able to join in.

Note: You can also assign this on tenant level (**/api/v1/tenants/<tenant-ID>**) or system level (**/api/v1/system/profiles**) for example to apply this for all spaces (or per tenant), however here it is shown on the space itself. Take into account that the most specific allocation of the **callLegProfile** is taken into account for the in-call behavior.

The **needsActivation** parameter is the most important one here for the guest/host behavior since if set to **true**, the participant is unable to receive or contribute audio and video until one or more **full/activator** (host) participants join. Other parameters on the **callLegProfile** can be found on section 8.4.3 of the API guide, under which the showed ones can be relevant in this setup as well (depending on your requirements):

- presentationContributionAllowed
- rxAudioMute
- rxVideoMute
- deactivationMode (deactive | disconnect | remainActivated) and deactivationModeTime [action to be performed when last activator leaves the call]

To create the guest **callLegProfile**, make a **POST** request on **/api/v1/callLegProfiles** with the preferred parameters and **needsActivation** parameter set to **true** so that you can perform a **GET** request on that **callLegProfile-ID** afterwards with this outcome for example:

```
<?xml version="1.0"?><callLegProfile id="d4bfe12d-68cd-41c0-a671-
48395ee170ab"><needsActivation>true</needsActivation><defaultLayout>speakerOnly</defaultLayout><
presentationContributionAllowed>false</presentationContributionAllowed><rxAudioMute>true</rxAudi
oMute><rxVideoMute>false</rxVideoMute><deactivationMode>deactivate</deactivationMode></callLegPr
ofile>
```

Note down the **callLegProfile-ID** as marked in bold as this has to be applied on the space in step 3 for the (default) guest access.

Step 2. Create a host **callLegProfile** (**needsActivation** = **false**).

Similarly create the host **callLegProfile** for the host in-call behavior. The same parameters as mentioned previously apply, although the parameters can be selected according to your own preference and requirements. The main element here, is to set the **needsActivation** parameter to **false** to give it the host role.

You create it by a **POST** request on **/api/v1/callLegProfiles** with the preferred parameters and **needsActivation** parameter set to **false** so that you can perform a **GET** request on that **callLegProfile-ID** afterwards with the this outcome for example:

```
<?xml version="1.0"?><callLegProfile id="7306d2c1-bc15-4dbf-ab4a-
1cbdaabd1912"><needsActivation>false</needsActivation><defaultLayout>speakerOnly</defaultLayout>
<presentationContributionAllowed>true</presentationContributionAllowed><rxAudioMute>false</rxAud
ioMute><rxVideoMute>false</rxVideoMute></callLegProfile>
```

Note down the **callLegProfile-ID** as marked in bold as this has to be applied on the space **accessMethod** in step 4 for the host access.

Step 3. Assign the guest **callLegProfile** to an existing or new space (being the default **accessMethod**).

Perform either a **PUT** command on an existing space (**/api/v1/coSpaces/<coSpace-ID>**) to adapt the space or a **POST** command on **/api/v1/coSpaces** to create a new one with the guest

callLegProfile parameter as created in step 1 as the in-call behavior for that space. You can also set the **URI**, **passcode** and **call-ID** parameters for that space as well to your desire as per section 6.2 of the API guide.

Perform a **GET** request on that space (**/api/v1/coSpaces/<coSpace-ID>**) to verify that the guest **callLegProfile** is associated with it, as well as the **URI** and **call-ID** value. An example output with this example created guest **callLegProfile** in step 1 is this one with a **URI** value of **guest.space** and **call-ID** of 628821815 (no passcode set):


```
<?xml version="1.0"?><coSpace id="7cc797c9-c0a8-47cf-b519-8dc5a01f1ade"><name>Guest.space</name><autoGenerated>true</autoGenerated><uri>guest.space</uri><callId>628821815</callId><callLegProfile>d4bfe12d-68cd-41c0-a671-48395ee170ab</callLegProfile><ownerId>bc392aaa-8c6d-4619-ad2f-cb30c4c53766</ownerId><ownerJid>Guest@cms.steven.lab</ownerJid><secret>iWqZQ.tTMIleeQHKMB.JYg</secret><numAccessMethods>1</numAccessMethods></coSpace>
```


Note down the space-ID as marked in bold as this has to be used to create the **accessMethod** on that particular space in step 4.

Step 4. Create a new **accessMethod** on that space with a different **URI** (and **call-ID**) and assign the host **callLegProfile** to it.

You want to create a different way of accessing the space than the guest access which is currently the default one. This is done by specifying an **accessMethod** on the space itself by a **POST** command on **/api/v1/coSpaces/<coSpace-ID>/accessMethods** with here the **coSpace-ID** being the bold marked value in step 3 (**7cc797c9-c0a8-47cf-b519-8dc5a01f1ade**) on which the host **callLegProfile** of step 2 is applied as well as the different **URI** and **call-ID** field.

After a GET reqeust on that space accessMethod (**/api/v1/coSpaces/<coSpace-ID>/accessMethods/<accessMethod-ID>**), you must be able to see a similar kind of output as this one, where you can see a different **URI** (host.space) and **call-ID** (888) as opposed to the default **accessMethod** of the space as well as the specially associated host **callLegProfile** as set up on step 2:


```
<?xml version="1.0"?><accessMethod id="447c61c5-44e5-465e-a8a1-5dd4e10048c0"><uri>host.space</uri><callId>888</callId><passcode></passcode><callLegProfile>7306d2c1-bc15-4dbf-ab4a-1cbdaabd1912</callLegProfile><secret>r8.QXRrOMFp719gDL5ck6Q</secret></accessMethod>
```


Verify

Now you can dial into the same meeting:

- As a guest

  - by dialing to guest.space URI (followed by the domain as configured on your call matching rules)

  - by entering the call-ID value 628821815 via IVR or WebRTC join (no passcode)

- As a host

  - by dialing to host.space URI (followed by the domain as configured on your call matching rules)

- by entering the call-ID value 888 via IVR or WebRTC join (no passcode)

When there are only guests joined to the space, they are all put in a lobby room waiting for the host to join in. Once a host joins, all of the guests and hosts are put in conference. If there are no hosts joined on the space anymore but still some guests, they return back to the lobby screen as per the configuration of **deactivate** on **deactivationMode** parameter on the guest **callLegProfile** as shown on Step 1.

2) Configuration using same URIs but non-empty guest and host PINs/passcodes

This configuration is similar as the one in the previous example, and also available already before CMS 2.1 release. It requires both the guest and host to enter a non-empty PIN or passcode so that differentiation can be made upon that, as they dial to the same **URI**.

The configuration steps are quite similar to the previous configurational example:

1. Create a guest **callLegProfile** (**needsActivation** = **true**)
2. Create a host **callLegProfile** (**needsActivation** = **false**)
3. Assign the guest **callLegProfile** to an existing or new space specifying a guest passcode (PIN) (being the default access method)
4. Create a new **accessMethod** on that same space with the same **URI** (different **call-ID**) and assign the host **callLegProfile** to it including a host passcode (PIN)

Step 1. Create a guest **callLegProfile** (**needsActivation** = **true**).

Same configuration as in previous example 1 and even the same guest **callLegProfile** (**`d4bfe12d-68cd-41c0-a671-48395ee170ab`**) can be used as demonstrated.

Step 2. Create a host **callLegProfile** (**needsActivation** = **false**)

Same configuration as inprevious example 1 and even the same host **callLegProfile** (**`7306d2c1-bc15-4dbf-ab4a-1cbdaabd1912`**) can be used as demonstrated.

Step 3. Assign the guest **callLegProfile** to an existing or new space specifying a guest passcode (PIN) (being the default **accessMethod**).

Similarly as before, you can either perform a **PUT** operation on an existing space (**/api/v1/coSpaces/<cospace-ID>**) or a **POST** operation to create a new space (**/api/v1/coSpaces**) with the desired parameters for the **URI**, passcode and **call-ID** for example as well as the guest **callLegProfile** (from step 1) that you assigned to it as per section 6.2 of the API guide.

If you perform a **GET** request on that space, you must be able to see a similar kind of output as this one where you see a **URI** of guestpin.space, a **call-ID** of 189, our previously created guest **callLegProfile** and a **passcode** of 789:

```
<?xml version="1.0"?><coSpace id="22d9f4ca-8b88-4d11-bba9-e2a2f7428c46"><name>Guest/Host
PIN</name><autoGenerated>false</autoGenerated><uri>guestpin.space</uri><callId>189</callId><call
LegProfile>d4bfe12d-68cd-41c0-a671-
48395ee170ab</callLegProfile><passcode>789</passcode><secret>X7f83UX7PHcIYp0JbT0fUA</secret><num
AccessMethods>1</numAccessMethods></coSpace>
```

Note down the space-ID as marked in bold as this has to be used to create the **accessMethod** on that particular space in step 4.

Step 4. Create a new **accessMethod** on that space with the same **URI** (different **call-ID**) and assign the host **callLegProfile** to it including a host passcode (PIN).

On this space you also create a different access method for the hosts (as the guest **callLegProfile** is assigned on the space itself as the default join option), just like on the first configuration example. This is done using a **POST** command on **/api/v1/coSpaces/<coSpace-ID>/accessMethods** with the **coSpace-ID** value for our space being **22d9f4ca-8b88-4d11-bba9-e2a2f7428c46** as highlighted in the previous step. On this **POST** command, you can provide the different parameters like the **URI** (guestpin.space, the same as the original one), **call-ID** (889), host **callLegProfile** as defined in step 2 and the host **passcode** or **PIN** (1234 in this case).

If you perform a **GET** request on that **accessMethod**, you must be able to see a similar kind of output showing the same **URI** of guestpin.space, a **call-ID** of 889, the host **callLegProfile** reference and the host **PIN** of 1234:

```
<?xml version="1.0"?><accessMethod id="760c0e17-55c0-4232-ba72-
2e9207916330"><uri>guestpin.space</uri><callId>889</callId><passcode>1234</passcode><callLegProf
ile>7306d2c1-bc15-4dbf-ab4a-
1cbdaabd1912</callLegProfile><secret>c0wnqI1qB9JGRdmekHEO1w</secret></accessMethod>
```

Verify

Now you can dial into the same meeting:

- As a guest

  - by dialing to guestpin.space **URI** (followed by the domain as configured on your call matching rules) and entering **PIN** 789

  - by entering the **call-ID** value 189 via IVR or WebRTC join with **PIN** 789

- As a host

  - by dialing to guestpin.space **URI** (followed by the domain as configured on your call matching rules) and entering **PIN** 1234

  - by entering the **call-ID** value 889 via IVR or WebRTC join with **PIN** 1234

When there are only guests joined to the space, they are all put in a lobby room waiting for the host to join in. Once a host joins, all of the guests and hosts are put in conference. If there are no hosts joined on the space anymore but still some guests, they return back to the lobby screen as per the configuration of **deactivate** on **deactivationMode** parameter on the guest **callLegProfile** as shown on Step 1.

3) Configuration using same URIs with mix of empty guest PIN and non-empty host PIN

This configuration is only available starting from version 2.1 of CMS onwards due to some newly

added API commands of passcodeMode and **passcodeTimeout** on the **callProfile** section. This allows for an empty PIN for guests to join (either entering # or timeout) while the host has a PIN to access the space and activate it. The **callProfile** controls the in-call experience for SIP (including Lync) calls and thus is not applicable for CMA clients (both thick client and WebRTC).

The configuration steps are similar as the ones of example 2, with the addition of the callProfile:

1. Create a guest **callLegProfile** (**needsActivation** = **true**)
2. Create a host **callLegProfile** (**needsActivation** = **false**)
3. Create a **callProfile** with the desired **passcodeMode** and **passcodeTimeout** configuration
4. Assign the guest **callLegProfile** and **callProfile** of step 3 to an existing or new space specifying a guest passcode (PIN) (being the default access method)
5. Create a new **accessMethod** on that same space with the same **URI** (different **call-ID**) and assign the host **callLegProfile** to it including a host passcode (PIN)

As the configurations are quite identical to the configuration examples 1 and 2, there are references to those ones. In fact for the test, the same space was used as in example 2, but added with the callProfile now.

Step 1. Create a guest **callLegProfile** (**needsActivation** = **true**).

Same configuration as inprevious example 1 and even the same guest callLegProfile (**`d4bfe12d-68cd-41c0-a671-48395ee170ab`**) can be used as demonstrated.

Step 2. Create a host **callLegProfile** (**needsActivation** = **false**).

Same configuration as inprevious example 1 and even the same host callLegProfile (**`7306d2c1-bc15-4dbf-ab4a-1cbdaabd1912`**) can be used as demonstrated.

Step 3. Create a **callProfile** with the desired **passcodeMode** and **passcodeTimeout** configuration.

You can create a callProfile that determines the in-call experience for SIP (including Lync) calls. There are a few possible configurations possible here, like allowing of recording or streaming or the maximum participant limit for example but the focus here is on the new API additions from CMS 2.1 relating to the passcode handling. The other parameters can be found on section 8.2 of the API guide.

Two parameters determine the passcode behavior here, being:

- ***passcodeMode***

  - **required** : the IVR waits forever for a user to enter the PIN or # for an empty PIN (for guests)

  - **timeout** : the IVR waits for a ***passcodeTimeout*** amount of seconds for the participant to enter the PIN and if no entry has been made within that time, it assumes a blank (#) PIN has been entered

- ***passcodeTimeout :*** only needs to be set when **passcodeMode** is set to timeout and controls the amount of time before interpreting passcode as a blank one

In order to create the **callProfile**, perform a **POST** command on **/api/v1/callProfiles** (or **PUT** on **/api/v1/callProfiles/<callProfile-ID>** if you want to modify an existing one) with the desired

parameters for **passcodeMode** and **passcodeTimeout**. If you perform a **GET** command on that specific **callProfile**, you must get a similar kind of outcome for example where you have set up the mode as timeout and a timeout value of 5 seconds:

<?xml version="1.0"?><callProfile id="**4b0eff60-e4aa-4303-8646-a7e800a4eac6**"><**passcodeMode**>timeout</**passcodeMode**><**passcodeTimeout**>5</**passcodeTimeout**></callProfile>

Note down the **callProfile-ID** as marked in bold as this has to be used to assign to the space to have this in-call behavior in step 4.

Step 4. Assign the guest **callLegProfile** and **callProfile** of step 3 to an existing or new space specifying a guest passcode (PIN) (being the default access method).

Similarly as before, you can either do a **PUT** operation on an existing space (**/api/v1/coSpaces/<cospace-ID>**) or a **POST** operation to create a new space (**/api/v1/coSpaces**) with the desired parameters for the **URI** and **call-ID** for example as well as the guest **callLegProfile** (from Step 1). The difference from the previous examples is the **callProfile** from step 3 and the fact that no passcode is assigned for it.

If you perform a **GET** request on that space, you must be able to see a similar kind of output as this example, where you see the **URI** of guestpin.space, a **call-ID** of 189, the previously created guest **callLegProfile** and the **callProfile** as set up in step 3:

<?xml version="1.0"?><coSpace id="**22d9f4ca-8b88-4d11-bba9-e2a2f7428c46**"><name>Guest/Host PIN</name><autoGenerated>false</autoGenerated><*uri*>guestpin.space</*uri*><*callId*>189</*callId*><**callLegProfile**>**d4bfe12d-68cd-41c0-a671-48395ee170ab**</**callLegProfile**><**callProfile**>**4b0eff60-e4aa-4303-8646-a7e800a4eac6**</**callProfile**><secret>X7f83UX7PHcIYp0JbT0fUA</secret><numAccessMethods>1</numAccessMethods></coSpace>

Note down the space ID as marked in bold as this has to be used to create the accessMethod on that particular space in step 5.

Step 5. Create a new **accessMethod** on that same space with the same **URI** (different **call-ID**) and assign the host **callLegProfile** to it including a host passcode (PIN).

On this space you also create a different access method for the hosts (as the guest **callLegProfile** is assigned on the space itself as the default join option), just like on the first configuration example. This is done using a **POST** command on **/api/v1/coSpaces/<coSpace-ID>/accessMethods** where the **coSpace-ID** value is replaced with the value for your space being **22d9f4ca-8b88-4d11-bba9-e2a2f7428c46** as highlighted in the previous step for this case. On this **POST** command, you can provide the different parameters like the **URI** (guestpin.space, the same as the original one), **call-ID** (889), host **callLegProfile** as defined in Step 2 and the host passcode or PIN (1234 in this case).

If you perform a **GET** request on that **accessMethod**, you must be able to see a similar kind of output showing the same **URI** of guestpin.space, a **call-ID** of 889, the host **callLegProfile** reference and the host **PIN** of 1234:

<?xml version="1.0"?><accessMethod id="760c0e17-55c0-4232-ba72-2e9207916330"><*uri*>guestpin.space</*uri*><*callId*>889</*callId*><**passcode**>1234</**passcode**><**callLegProfile**>**7306d2c1-bc15-4dbf-ab4a-1cbdaabd1912**</**callLegProfile**><secret>c0wnqI1qB9JGRdmekHEO1w</secret></accessMethod>

Verify

Now you can dial into the same meeting:

- As a guest

  - by dialing to guestpin.space **URI** (followed by the domain as configured on your call matching rules) and entering # as **PIN** or let it timeout after 5 seconds

  - by entering the **call-ID** value 189 via IVR or WebRTC join

- As a host

  - by dialing to guestpin.space **URI** (followed by the domain as configured on your call matching rules) and entering **PIN** 1234

  - by entering the **call-ID** value 889 via IVR or WebRTC join with **PIN** 1234

4) A host user is a member of the space and authorized via webRTC log in, guest users join the meeting with callID. Same URI and callID is used by guest and host participants with empty or non-empty PIN/passcodes for guest users

The next steps need to be performed to get the Guest/Host access differentiation on the same space for members and non-members of the space:

1. Create a guest **callLegProfile**(**needsActivation** = **true**)
2. Create a host **callLegProfile**(**needsActivation** = **false**)
3. Assign the guest **callLegProfile** to an existing or new space (being the default access method)
4. Create a new **accessMethod** on that same space with the same **URI** (and call-ID) and assign the host **callLegProfile** to it
5. Assign user's**ownerJID**to same space. (if not assigned)
6. Add that**ownerID**as the member user to same space and assign**hostcallLegProfile**to that member user

Step 1. Create a guest callLegProfile (needsActivation = true).

Same configuration as in previous example 1 and the guest callLegProfile (**bfe7d07f-c7cb-4e90-a46e-4811bbaf6978**) is used in this example.

Note down the **callLegProfile-ID** as marked in bold as this has to be applied on the space in step 3 for the guest access.

Step 2. Create a host callLegProfile (needsActivation = false)

Same configuration as inprevious example 1 and the host callLegProfile (**0e76e943-6d90-43df-9f23-7f1985a74639**) is used in this example.

Note down the **callLegProfile-ID** as marked in bold as this has to be applied on the space **accessMethod** in step 4 for the host access and on the coSpace member in step 6.

Step 3. Assign the guest **callLegProfile** to an existing or new space (being the default accessMethod).

Perform either a **PUT** command on an existing space (**/api/v1/coSpaces/<coSpace-ID>**) to adapt the space or a **POST** command on **/api/v1/coSpaces** to create a new one with the guest **callLegProfile** parameter as created in step 1 as the in-call behavior for that space. You can also set the **URI** and **call-ID** parameters for that space as well to your desire as per section 6.2 of the API guide.

Perform a **GET** request on that space (**/api/v1/coSpaces/<coSpace-ID>**) to verify that the guest **callLegProfile** is associated with it, as well as the **URI** and **call-ID** value. An example output with this example created guest **callLegProfile** in step 1 is this one with a **URI** value of**global** and **call-ID** of 1234 (no passcode set), **nonMemberAccess**set to**true**:

```
<?xml version="1.0" ?>
<coSpace id="96d28acb-86c6-478d-b81a-a37ffb0adafc">
        <name>Global</name>
        <autoGenerated>false</autoGenerated>
        <uri>global</uri>
        <callId>1234</callId>
        <callLegProfile>bfe7d07f-c7cb-4e90-a46e-4811bbaf6978</callLegProfile>
        <nonMemberAccess>true</nonMemberAccess>
        <secret>0w4O2zTTF0WdL4ymF8D0_A</secret>
        <defaultLayout>allEqual</defaultLayout>
</coSpace>
```

Note down the space-ID as marked in bold as this has to be used to create the **accessMethod** on that particular space in step 4.

Step 4. Create a new **accessMethod** on that space with the same **URI** (and **call-ID**) and assign the host **callLegProfile** to it.

You want to create a different way of accessing the space than the guest access which is currently the default one. This is done by specifying an **accessMethod** on the space itself by a **POST**command on **/api/v1/coSpaces/<coSpace-ID>/accessMethods** with here the **coSpace-ID** being the bold marked value in step 3 (**96d28acb-86c6-478d-b81a-a37ffb0adafc**) on which the host **callLegProfile** of step 2 is applied as well as the same **URI** and **call-ID** field. You can add non-empty passcode for the hosts who connect via callID (without being logged in as a user via webRTC).

After a GET request on that space accessMethod (**/api/v1/coSpaces/<coSpace-ID>/accessMethods/<accessMethod-ID>**), you must be able to see a similar kind of output as this one, where you can see the same**URI** (global) and **call-ID** (1234) as well as the specially associated host **callLegProfile** as set up on step 2 and**host passcode**(12345):

```
<?xml version="1.0" ?>
<accessMethod id="c4ecc16e-945f-4e35-ba03-d9b69107b32c">
        <uri>global</uri>
        <callId>1234</callId>
        <passcode>12345</passcode>
        <callLegProfile>0e76e943-6d90-43df-9f23-7f1985a74639</callLegProfile>
        <secret>kffO1zTTE0feL4fsdf43w_B </secret>
</accessMethod>
```

Step 5. Assign user's **ownerJid**to the space. (if not assigned). Add ownerJID to the space by specifying**ownerJid (user1@evacanoalone.net)**on the space by a**PUT**command on**/api/v1/coSpaces/<coSpace-ID>**

After a GET request on that space, you must be able to see that **ownerId**and **ownerJid**have been

assigned to the space:

```
<?xml version="1.0" ?>
<coSpace id="96d28acb-86c6-478d-b81a-a37ffb0adafc">
        <name>Global</name>
        <autoGenerated>false</autoGenerated>
        <uri>global</uri>
        <callId>1234</callId>
        <callLegProfile>bfe7d07f-c7cb-4e90-a46e-4811bbaf6978</callLegProfile>
        <nonMemberAccess>true</nonMemberAccess>
        <ownerId>1d942281-413e-4a2a-b776-91a674c3a5a9</ownerId>
        <ownerJid>user1@evacanoalone.net</ownerJid>
        <secret>0w4O2zTTF0WdL4ymF8D0_A</secret>
        <numAccessMethods>1</numAccessMethods>
        <defaultLayout>allEqual</defaultLayout>
</coSpace>
```

Note down the ownerID (**1d942281-413e-4a2a-b776-91a674c3a5a9**).

Step 6. Add that ownerID (1d942281-413e-4a2a-b776-91a674c3a5a9) from the step 5 as the member user to the space and assign**hostcallLegProfile**to that member user. This is done by specifying specifying**userJId**and**host callLegProfile**on the space itself (specifying**coSpaceID**) by a **POST**command (**/api/v1/coSpaces/<coSpaceID>/coSpaceUsers).**Other parameters on the **coSpaceUsers** can be found on section 6.4.2 of the API guide, under which the showed ones can be relevant in this setup as well:

<canDestroy>true</canDestroy>

<canAddRemoveMember>true</canAddRemoveMember>

<canChangeName>true</canChangeName>

<canChangeUri>false</canChangeUri>

<canChangeCallId>false</canChangeCallId>

<canChangePasscode>true</canChangePasscode>

<canPostMessage>true</canPostMessage>

<canDeleteAllMessages>false</canDeleteAllMessages>

<canRemoveSelf>false</canRemoveSelf>

Veryfy that member user has been added to the space by a**GET**command (**/api/v1/coSpaces/<coSpaceID>/coSpaceUsers?**)

```
<?xml version="1.0" ?>
<coSpaceUsers total="1">
<coSpaceUser id="1d942281-413e-4a2a-b776-91a674c3a5a9">
<userId>1d942281-413e-4a2a-b776-91a674c3a5a9</userId>
<userJid>user1@evacanoalone.net</userJid>
<autoGenerated>false</autoGenerated>
</coSpaceUser>
</coSpaceUsers>
```

Note down the userID (if different form ownerID form step 5). Verify that**host callLegProfile**has been assigned to coSpaceUser by a**GET**request specifying**coSpaceID**and**userID (/api/v1/coSpaces/<coSpaceID>/coSpaceUsers/<userID>)**

```
<?xml version="1.0" ?>
<coSpaceUser id="1d942281-413e-4a2a-b776-91a674c3a5a9">
        <userId>1d942281-413e-4a2a-b776-91a674c3a5a9</userId>
        <userJid>user1@evacanoalone.net</userJid>
        <autoGenerated>false</autoGenerated>
        <canDestroy>true</canDestroy>
        <canAddRemoveMember>true</canAddRemoveMember>
        <canChangeName>true</canChangeName>
        <canChangeUri>false</canChangeUri>
        <canChangeCallId>false</canChangeCallId>
        <canChangePasscode>true</canChangePasscode>
        <canPostMessage>true</canPostMessage>
        <canDeleteAllMessages>false</canDeleteAllMessages>
        <canRemoveSelf>false</canRemoveSelf>
        <canChangeNonMemberAccessAllowed>true</canChangeNonMemberAccessAllowed>
        <callLegProfile>0e76e943-6d90-43df-9f23-7f1985a74639</callLegProfile>
</coSpaceUser>
```

Verify

Now you can dial into the same meeting:

- As a guest
  - by dialing to URI (followed by the domain as configured on your call matching rules)

  - by entering the call-ID value 1234 via IVR or WebRTC join (no passcode)

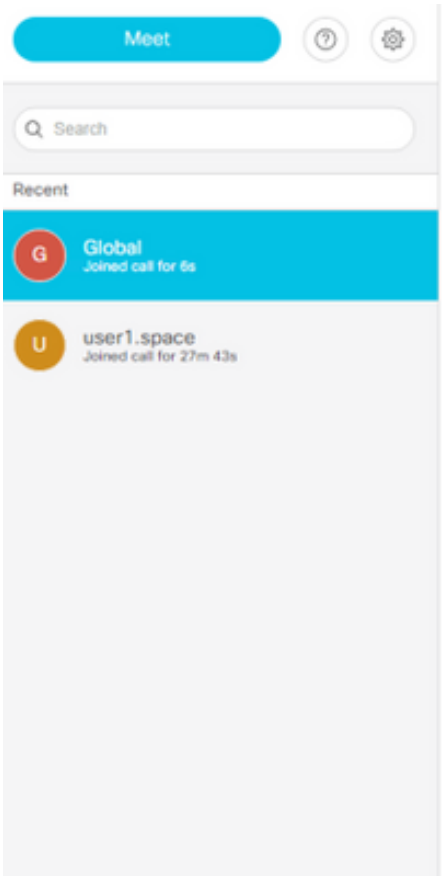
- As a host
  By logging in as a user (a member of the space with assigned “host” callLegProfile, with user1@evacanoalone.net in this scenario) via webRTC and join the space (“global” URI).

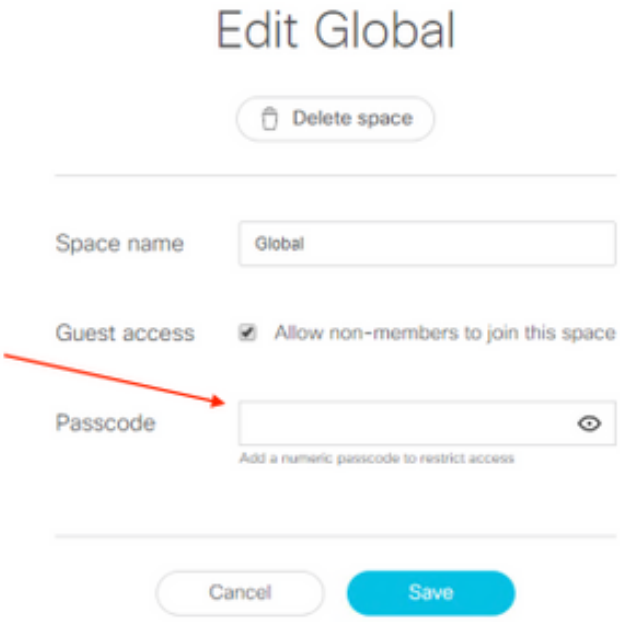
  - by dialing to “global” URI (followed by the domain as configured on your call matching rules) and passcode 12345.

  - by entering the call-ID value 1234 via IVR or WebRTC join (with a host passcode 12345)

When there are only guests joined to the space, they are all put in a lobby room waiting for the host to join in. Once a host joins, all of the guests and hosts are put in the conference. If there are no hosts joined on the space anymore but still some guests, they return back to the lobby screen as per the configuration of **deactivate** on **deactivationMode** parameter on the guest **callLegProfile** as shown on Step 1.

Host (owner/member) can set(edit/remove) a password for guests directly in webRTC app or completely disable a non-member (guest) access for the space:





Troubleshoot

This section provides information you can use in order to troubleshoot your configuration.

The logging of CMS does show us briefly when you join as a guest or when the first host joins but it is best to verify using GET requests the callProfile as well as guest and host callLegProfile definitions and the allocation of them on the respective accessMethods (or the default access method) or at higher level (global level or tenant level).

You can follow on this structure to get all of the information:

1. **GET on /api/v1/callProfiles** (if using this with the **passcodeMode**)
   >verify in detail the desired callProfile-ID using **GET** on **/api/v1/callProfiles/<callProfile-ID>**
2. **GET** on **/api/v1/callLegProfiles**
   >verify in detail the desired callLegProfile-IDs of guest and host using **GET** on **/api/v1/callProfiles/<callProfile-ID>**
3. **GET** on **/api/v1/coSpaces**
   >verify in detail the desired space-ID using **GET** on **/api/v1/coSpaces/<coSpace-ID>**
   >check if the desired callProfile-ID (step 1) and guest callLegProfile (step 2) are associated to this space
   [if it is not there, check on the less specific elements like tenant (**/api/v1/tenants/<tenant-ID>**) or global (**/api/v1/system/profiles**) level]
4. **GET** on **/api/v1/coSpaces/<coSpace-ID>/accessMethods**
   >verify in detail the desired accessMethod using **GET** on **/api/v1/coSpaces/<coSpace-ID>/accessMethods/<accessMethod-ID>** to check if the host **callLegProfile** is assigned

In the CMS logging shown int his example, you have first two guest participants coming in (*call 38* from 2000@steven.lab and *call 39* from 1060@steven.lab) who time out to the guestpin.space@acano.steven.lab space and then the host joins. You can see from the snippet that for guests it informs us about what needs to be done with it (**to be deactivated**) and you can see this behavior for those calls change when the host (stejanss.movi@steven.lab) joins on the space (**ceasing to be deactivated**). Similarly you can see the same logging again when the guests move to the lobby again as soon as there are no hosts anymore on the space (**to be deactivated**).

```
2017-02-21  17:48:54.809  Info     call 38: incoming encrypted SIP call from
"sip:2000@steven.lab" to local URI "sip:guestpin.space@acano.steven.lab" 2017-02-21 17:48:54.822
Info call 38: setting up UDT RTP session for DTLS (combined media and control) 2017-02-21
17:48:54.837 Info call 38: compensating for far end not matching payload types 2017-02-21
17:48:54.847 Info sending prompt response (2) to BFCP message 2017-02-21 17:48:54.847 Info call
38: sending BFCP hello as client following receipt of hello when BFCP not active 2017-02-21
17:48:54.883 Warning call 38: replacing pending BFCP message "PrimitiveHelloAck" with
"PrimitiveHelloAck" 2017-02-21 17:48:54.883 Info call 38: BFCP (client role) now active 2017-02-
21 17:48:59.294 Info call 39: incoming encrypted SIP call from "sip:1060@steven.lab" to local
URI "sip:guestpin.space@acano.steven.lab" 2017-02-21 17:48:59.310 Info call 39: setting up UDT
RTP session for DTLS (combined media and control) 2017-02-21 17:48:59.323 Info call 39:
compensating for far end not matching payload types 2017-02-21 17:48:59.569 Info sending prompt
response (2) to BFCP message 2017-02-21 17:48:59.569 Info call 39: sending BFCP hello as client
following receipt of hello when BFCP not active 2017-02-21 17:48:59.746 Info call 39: BFCP
(client role) now active 2017-02-21 17:49:07.971 Info configuring call e2264fb0-483f-45bc-a4f3-
5a4ce326e72c to be deactivated
2017-02-21  17:49:07.972  Info     participant "2000@steven.lab" joined space 22d9f4ca-8b88-
4d11-bba9-e2a2f7428c46 (Guest/Host PIN)
2017-02-21  17:49:12.463  Info     configuring call b1b5d433-5ab5-49e1-9ae3-3f4f71703d1b to be
deactivated
2017-02-21  17:49:12.463  Info     participant "1060@steven.lab" joined space 22d9f4ca-8b88-
4d11-bba9-e2a2f7428c46 (Guest/Host PIN)
2017-02-21  17:49:12.463  Info     conference "Guest/Host PIN": unencrypted call legs now
present
2017-02-21  17:49:16.872  Info     call 40: incoming encrypted SIP call from
"sip:stejanss.movi@steven.lab" to local URI "sip:guestpin.space@acano.steven.lab" 2017-02-21
17:49:16.885 Info call 40: setting up UDT RTP session for DTLS (combined media and control)
2017-02-21 17:49:24.260 Info call 40: audio prompt play time out 2017-02-21 17:49:26.670 Info
participant "stejanss.movi@steven.lab" joined space 22d9f4ca-8b88-4d11-bba9-e2a2f7428c46
(Guest/Host PIN)
2017-02-21  17:49:26.670  Info     call e2264fb0-483f-45bc-a4f3-5a4ce326e72c ceasing to be
deactivated
2017-02-21  17:49:26.670  Info     call b1b5d433-5ab5-49e1-9ae3-3f4f71703d1b ceasing to be
deactivated
2017-02-21  17:49:30.832  Info     call 40: ending; remote SIP teardown - connected for 0:14
2017-02-21  17:49:30.833  Info     participant "stejanss.movi@steven.lab" left space 22d9f4ca-
8b88-4d11-bba9-e2a2f7428c46 (Guest/Host PIN)
2017-02-21  17:49:30.833  Info     configuring call e2264fb0-483f-45bc-a4f3-5a4ce326e72c to be
deactivated
2017-02-21  17:49:30.833  Info     configuring call b1b5d433-5ab5-49e1-9ae3-3f4f71703d1b to be
deactivated
```

Related Information

- Technical Support & Documentation - Cisco Systems
- CMS documentation