Guide for Managing Users on Control Hub Managed Sites via API

Create and Authorize Webex Integration (OAuth 2.0) to get Access Token:

1. User or Admin create Integration (https://developer.webex.com/docs/integrations) with following scopes (at a minimum, others may be added based on your other requirements):
    a. **spark:all** (used to obtain XML API token)
    b. **spark-admin:licenses_read** (listing licenses available for organization)
    c. **spark-admin:people_read** (Read access to Webex Org user directory)
    d. **spark-admin:people_write** (Write access to Webex Org user directory)
2. Admin Authenticates/Authorizes Integration (https://developer.webex.com/blog/real-world-walkthrough-of-building-an-oauth-webex-integration)

Use REST API to Create/Update users to assign Meeting Licenses:

1. Use the /v1/licenses/list-licenses REST API to get the available Organization Licenses.
2. Use the /v1/people REST API to Create, Update users to assign the Meeting license.
    a. POST /v1/people (create-a-person)
    b. GET /v1/people/{personId} (get-person-details)
    c. PUT /v1/people/{personId} (update-a-person)

Managing user Session Types and Tracking Codes:

1. First you must use the XML API AuthenticateUser request to exchange the OAuth 2.0 Access Token for a Session Ticket which is used for subsequent XML API requests.
2. With the resulting session ticket use the following XML APIs:
    a. The GetSite XML API can return the Tracking Code configurations for the site as well as the Session Type ID list.
    b. The LstMeetingType XML API can also be used to return more detailed information about the Session Types available on the Webe Meeting site.
3. The SetUser XML API is used to modify the users Meeting/Session Types and Tracking Code values.

Refresh tokens/tickets

CI Tokens needs periodic refresh (at least every 14 days)
https://developer.webex.com/blog/real-world-walkthrough-of-building-an-oauth-webex-integration
Session Tickets need to be re-generated at least every 90 minutes

Other non-API options

Control Hub User Management:

Directory Connector for MS Active Directory(guide)
Manual User management in Cisco Webex Control Hub (create / modify)
Bulk CSV upload in Control Hub (create / modify)

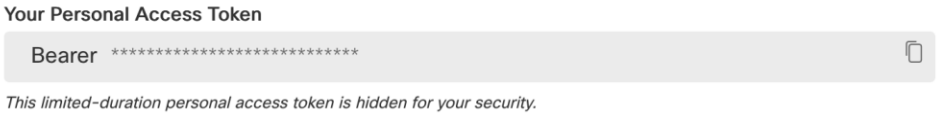
Control Hub Meeting Setting management:

Configuring Tracking Codes for Meeting site and Users (help.webex.com)
Managing Session Types for Meeting site and Users (help.webex.com)

Details

Testing the XML API without creating an integration.

1.1. Skip to the next step for directions on creating and using the Oauth2.0 Integrations instead.
1.2. The Getting Started page on the Webex developer portal (https://developer.webex.com/docs/api/getting-started) provides an Access Token for testing that has a short lifetime—only 12 hours after logging into this site—so it shouldn't be used outside of app development. When using this token, any actions taken through the API will be done as you.
1.3. Look for the following field and click the rectangular icon to copy your token to your OS clipboard.



1.4. With this token you can now skip to step 5 "Accessing the Meetings XML API" below to begin testing the XML API's.

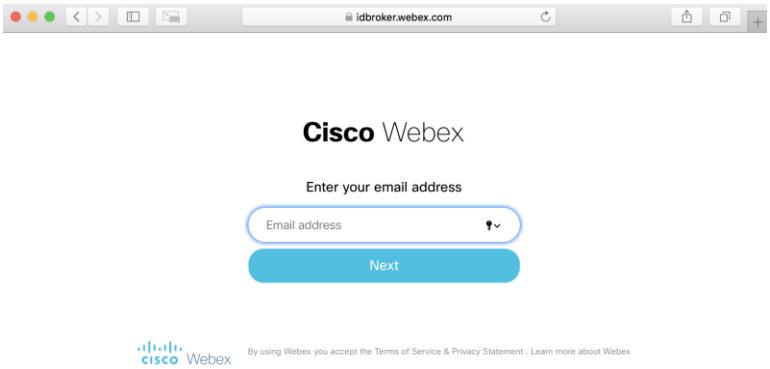
2. Create a Webex Teams Integration at https://developer.webex.com/docs/integrations

2.1. Must use the **spark:all** scope.
2.2. Detailed walkthrough guide for creating Webex Teams Integrations available at https://developer.webex.com/blog/real-world-walkthrough-of-building-an-oauth-webex-integration

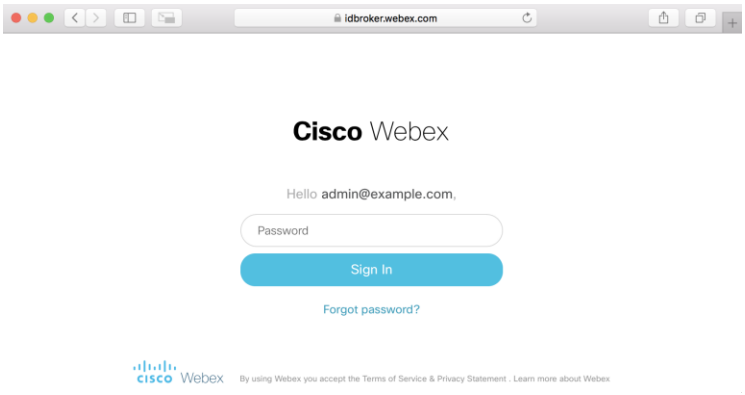
3. Get the Access Token: Use the Integration created in Step 1 to Authenticate/Authorize your Admin account.

3.1. The Oauth 2.0 flow starts with accessing the OAuth Authorization URL from a web browser.
Example:
https://api.ciscospark.com/v1/authorize?client_id={CLIENT_ID}&response_type=code&redirect_uri={Redirect_URI}&scope=spark%3Aall%20spark%3Akms&state={set_state_here}

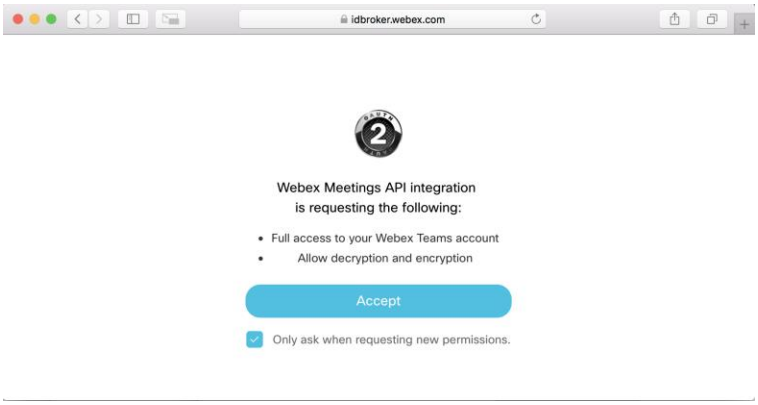
3.2. Once the Authorization URL has loaded you will provide your admin email address and click Next.



3.3. Next you will be prompted for your Webex Password (If using SSO you will be redirected to your SSO Service to complete authentication as usual).



3.4. After Authentication has completed the you will be prompted to Authorize the Webex Integration you had created in step 1. Click the "Accept" button to grant access to the Integration.



3.5. After the Authorization has been completed Webex will redirect the web browser to the "Redirect URI" specified when the Integration was created.

The "Redirect URI" represents the endpoint where your browser will redirect to after you have authorized the Integration. This is typically a valid HTTP endpoint you would host/control (supporting TLS recommended), so the Authorization Code or Access Tokens can be transmitted to your server securely to avoid attacks.

Example: https://your_http_redirect.example.com/?code={Authorization Code}&state={set_state_here}

3.6. In this final step you will exchange the Authorization Code for the actual Access Token your app can use to begin interacting with the Meetings XML API's.

To do this, the app will need to perform an HTTP POST to the following URL with a standard set of OAuth parameters (this endpoint will only accept an x-www-form-urlencoded body):
https://api.ciscospark.com/v1/access_token

The required parameters are:

- grant_type — This should be set to "authorization_code"
- client_id — Issued when creating your integration.
- client_secret — Issued when creating your integration.
- code — The Authorization Code from the previous step.
- redirect_uri — Must match the one used when creating your integration.

Example Request (curl):

```
curl -X POST 'https://api.ciscospark.com/v1/access_token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=authorization_code' \
--data-urlencode 'client_id= Your Client ID provided after creating Integration ' \
--data-urlencode 'client_secret= Your Client Secret provided after creating Integration ' \
--data-urlencode 'code= Authorization Code from step 2.5 ' \
--data-urlencode 'redirect_uri=https://your_http_redirect.example.com/'
```

Example Response:

```
{
  "access_token": "{Webex_Access_Token}",
  "expires_in": 1209599,   //value is total time in seconds = 14 days
  "refresh_token": "{Webex_Refresh_Token}",
  "refresh_token_expires_in": 7775999 //value is total time in seconds = 90 days
}
```

4. Refreshing the Access Token (at least every 14 days)

4.1. Now that you have Authenticated/Authorized the integration you can use the refresh token to generate a new Access Token without going through the Oauth2.0 process again, allowing you to essentially remain authenticated forever.

Getting a new Access Token using the Refresh Token is done much like getting the initial Access Token but with different parameters:

- grant_type — This should be set to "refresh_token"
- client_id — Same client_id you received when you created your integration
- client_secret — Same secret you used to get the Access Token originally
- refresh_token — This is the Refresh Token returned when you got the initial Access Token

Example Request (curl):

```
curl -X POST 'https://api.ciscospark.com/v1/access_token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=refresh_token' \
```

```
--data-urlencode 'client_id= Your Client ID provided after creating Integration' \
--data-urlencode 'client_secret= Your Client Secret provided after creating Integration' \
--data-urlencode 'refresh_token= Refresh Token returned from previous step'
```

Example Response:

```
{
   "access_token": "{Webex_Access_Token}",
   "expires_in": 1209599,   //value is total time in seconds = 14 days
   "refresh_token": "{Webex_Refresh_Token}",
   "refresh_token_expires_in": 7775999 //value is total time in seconds = 90 days
}
```

5. Accessing the Meetings XML API.

5.1. The XML API is XML-RPC based, using XML requests sent over HTTP POST to exchange data with Webex.
5.2. The XML API endpoint uses the Webex Meeting Site URL. Example:
'https://{MeetingSiteURL}.webex.com/WBXService/XMLService'

5.3. Example GetAPIVersion request (curl):

```
curl -X POST 'https://{MeetingSiteURL}.webex.com/WBXService/XMLService' \
--header 'Content-Type: application/xml' \
--data '<?xml version="1.0" encoding="UTF-8"?>
<serv:message xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <header>
  <securityContext>
   <siteName>{MeetingSiteURL}</siteName>
  </securityContext>
 </header>
 <body>
  <bodyContent xsi:type="java:com.webex.service.binding.ep.GetAPIVersion">
  </bodyContent>
 </body>
</serv:message>'
```

6. Exchange the Access Token for a Session Ticket using the XML API AuthenticateUser request.

6.1. Documentation: https://developer.cisco.com/docs/webex-xml-api-reference-guide/#!authenticateuser
6.2. **XML API request:**

```
<?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">
 <header>
  <securityContext>
   <siteName>{MeetingSiteURL}</siteName>
   <webExID>{Administrator_Email_address}</webExID>
  </securityContext>
 </header>
 <body>
  <bodyContent xsi:type="java:com.webex.service.binding.user.AuthenticateUser">
   <accessToken>{Webex_Access_Token}</accessToken>
```

```
  </bodyContent>
 </body>
</serv:message>
```

XML API response:

```
<?xml version="1.0" encoding="UTF-8"?>
<serv:message>
 <serv:header>
  <serv:response>
   <serv:result>SUCCESS</serv:result>
   <serv:gsbStatus>PRIMARY</serv:gsbStatus>
  </serv:response>
 </serv:header>
 <serv:body>
  <serv:bodyContent xsi:type="use:authenticateUserResponse">
   <use:sessionTicket>{SESSION_TICKET}</use:sessionTicket>
   <use:createTime>1580795693342</use:createTime>
   <use:timeToLive>5400</use:timeToLive> <!—expires in seconds 5400 = 90 minutes -->
  </serv:bodyContent>
 </serv:body>
</serv:message>
```

6.3. About the Session Ticket:

6.3.1. default expiration time of 90 minutes (defined and managed by webex).

6.3.2. Cannot be refreshed, or kept alive with usage.

6.3.3. Can be revoked with the XML API revokeSessionTicket request
- https://developer.cisco.com/docs/webex-meetings/#xml-api-10-0-release-notes/security-enhancement-single-logout-api-xml-api

6.3.4. Session ticket needs to be regenerated at least every 90 minutes

7. Use the Session Ticket from above step to authenticate subsequent XML API requests.

7.1. Documentation: https://developer.cisco.com/docs/webex-xml-api-reference-guide/#!getuser

XML request:

```
<?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">
 <header>
  <securityContext>
   <siteName>{MeetingSiteURL}</siteName>
   <webExID>{Administrator_Email_address}</webExID>
   <sessionTicket>{SESSION_TICKET}</sessionTicket>
  </securityContext>
 </header>
 <body>
```

```
  <bodyContent xsi:type="java:com.webex.service.binding.user.GetMe">
  </bodyContent>
 </body>
</serv:message>
```

XML response:

```
<?xml version="1.0" encoding="UTF-8"?>
<serv:message>
 <serv:header>
  <serv:response>
   <serv:result>SUCCESS</serv:result>
   <serv:gsbStatus>PRIMARY</serv:gsbStatus>
  </serv:response>
 </serv:header>
 <serv:body>
  <serv:bodyContent xsi:type="use:getMeResponse">
   <use:firstName>Firstname</use:firstName>
   <use:lastName>Lastname</use:lastName>
   <use:email>First.Last@example.com</use:email>
   <use:webExID>First.Last@example.com</use:webExID>
   <use:userRole>siteAdmin</use:userRole>
   <use:personalMeetingRoom>
    <use:hostPin>1234</use:hostPin>
    <use:title>Example's Personal Room</use:title>
    <use:pmrUrl>https://example.webex.com/meet/asd8qcji9f2j9c3mpcmp9hc2</use:pmrUrl>
    <use:sipUrl>asd8qcji9f2j9c3mpcmp9hc2@example.webex.com</use:sipUrl>
    <use:displayMeetingUrl>asd8qcji9f2j9c3mpcmp9hc2@example.webex.com</use:displayMeetingUrl>
   </use:personalMeetingRoom>
  </serv:bodyContent>
 </serv:body>
</serv:message>
```