

The background of the slide is a vibrant, abstract graphic. It features a series of overlapping, wavy bands of color in shades of red, orange, yellow, green, and blue, creating a sense of movement and energy. On the right side, there is a bright, multi-colored sunburst or starburst effect, with rays of light radiating outwards in various colors including blue, green, yellow, and orange. The overall composition is dynamic and visually appealing.

cisco *Live!*

Let's go

#CiscoLive



The bridge to possible

# Agile Management of Webex Calling

There's an API for that

Johannes Krohn, Principal Technical Marketing Engineer  
BRKCOL-3015



#CiscoLive



# Cisco Webex App

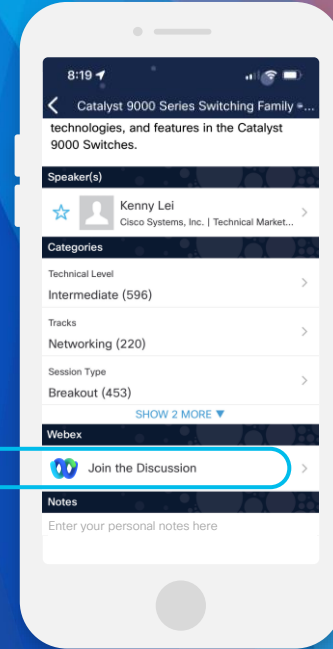
## Questions?

Use Cisco Webex App to chat with the speaker after the session

## How

- 1 Find this session in the Cisco Live Mobile App
- 2 Click “Join the Discussion”
- 3 Install the Webex App or go directly to the Webex space
- 4 Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until June 9, 2023.



<https://ciscolive.ciscoevents.com/ciscolivebot/#BRKCOL-3015>

# Agenda

- Why APIs?
- Coverage/Capabilities
- Getting started
- Use cases/Examples
- Closing

# Why APIs?



# API, What, Where, and Why?

- Definition: .. is a set of subroutine definitions, protocols, and tools for building application software. In general terms, it is a set of clearly defined methods of communication between various software components. .. Documentation for the API is usually provided to facilitate usage.”<sup>1</sup>
- APIs
  - Enabler for open systems integration
  - Universally available
  - Unleash developer innovation



# Webex Calling Provisioning Methods

	Control Hub	CSV	API
Ease of Use	+++	++	+
Speed	+	+++	+++
Customization		+	+++

# Coverage / Capabilities





# Webex APIs

- Documentation: <http://developer.webex.com>
- Various APIs available:
- Admin (licenses, locations, memberships, people, ..)
- Calling (call control, locations, people, org/location settings, ...)
- Devices (configuration, places, workspace locations, xAPI, ...)
- Meetings (invitees, participants, preferences, ...)
- ...
- OAuth access token used for authorization

## Webex APIs

- + Admin
- + Calling
- + Contact Center
- + Devices
- + Meetings
- + Messaging
- + Webex Assistant Skills
- + FedRAMP
- + Full API Reference

# Webex Calling API capabilities

- Provisioning
  - Users (incl. calling entitlements), locations (r/o), call pickups, call queues, hunt groups, auto attendant, call parks, schedules, voice messaging settings, ...
  - person settings: barge, call forwarding, call intercept, call recording, caller ID, voicemail settings, ...
  - Coverage continuously growing\*
- Call Control
  - Dial, answer, reject, hangup, hold/resume, divert, transfer, park/retrieve, start/stop/pause/resume recording, DTMF, push, pickup, barge
- Webhook Notifications/Events
  - Voice messages
  - Call events

\*Check <https://help.webex.com/en-us/article/rdmb0/What's-new-in-Webex-Calling> and <https://developer.webex.com/> for updates

## References:

<https://developer.webex.com/docs/webex-calling>  
<https://developer.webex.com/blog/calling-apis-overview>

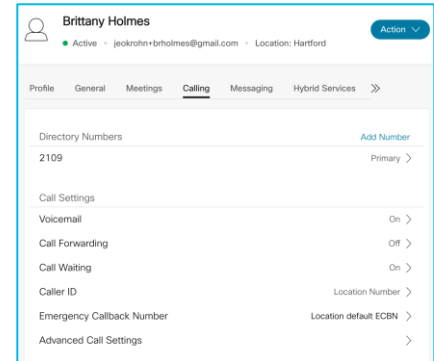
# Webex Calling Provisioning APIs

- Locations, <https://developer.webex.com/docs/api/v1/locations>
  - List locations
  - Get location details
  - Create/Update locations
  - Delete: not possible
- People, <https://developer.webex.com/docs/api/v1/people>
  - List
  - CRUD
  - `callingData` parameter to access calling data\*

# Webex Calling Provisioning APIs

Calling	Numbers	Locations	Call Routing	Features	PSTN Orders	Service Settings	Client Settings
<u>Auto Attendant</u>	Call Park Extension	Call Park Group	Call Pick-up	Call Queue	DECT Network	Hunt Group	Single Number Reach
Paging Group	Receptionist Client	Virtual Extension	Voicemail Group				

- Organization Settings,  
<https://developer.webex.com/docs/api/v1/webex-calling-organization-settings>
- Calling features found in Feature tab in Control Hub
- Person Settings,  
<https://developer.webex.com/docs/api/v1/webex-calling-person-settings>
- Settings found in person's Calling tab in Control Hub



# Webex Calling Voice Messaging APIs

- Voice Messaging,  
<https://developer.webex.com/docs/api/v1/webex-calling-voice-messaging>
- Handle voicemail and MWI
- User access only; no admin access
- Message summary, list messages, delete message, mark read/unread

GET	<a href="https://webexapis.com/v1/telephony/voiceMessages/summary">https://webexapis.com/v1/telephony/voiceMessages/summary</a>	Get Message Summary
GET	<a href="https://webexapis.com/v1/telephony/voiceMessages">https://webexapis.com/v1/telephony/voiceMessages</a>	List Messages
DELETE	<a href="https://webexapis.com/v1/telephony/voiceMessages/{messageId}">https://webexapis.com/v1/telephony/voiceMessages/{messageId}</a>	Delete Message
POST	<a href="https://webexapis.com/v1/telephony/voiceMessages/markAsRead">https://webexapis.com/v1/telephony/voiceMessages/markAsRead</a>	Mark As Read
POST	<a href="https://webexapis.com/v1/telephony/voiceMessages/markAsUnread">https://webexapis.com/v1/telephony/voiceMessages/markAsUnread</a>	Mark As Unread

# Webex Calling Call Controls

- Actions
  - Dial, answer, reject, hangup, hold/resume, divert, transfer, park/retrieve, start/stop/pause/resume recording, DTMF, push, pickup, barge
- Management
  - List, get details, call history
  - List/Details use common call object
- Requires user access token
  - No org level (admin) operations

Method	Description
POST <a href="https://webexapis.com/v1/telephony/calls/dial">https://webexapis.com/v1/telephony/calls/dial</a>	Dial
POST <a href="https://webexapis.com/v1/telephony/calls/answer">https://webexapis.com/v1/telephony/calls/answer</a>	Answer
POST <a href="https://webexapis.com/v1/telephony/calls/reject">https://webexapis.com/v1/telephony/calls/reject</a>	Reject
POST <a href="https://webexapis.com/v1/telephony/calls/hangup">https://webexapis.com/v1/telephony/calls/hangup</a>	Hangup
POST <a href="https://webexapis.com/v1/telephony/calls/hold">https://webexapis.com/v1/telephony/calls/hold</a>	Hold
POST <a href="https://webexapis.com/v1/telephony/calls/resume">https://webexapis.com/v1/telephony/calls/resume</a>	Resume
POST <a href="https://webexapis.com/v1/telephony/calls/divert">https://webexapis.com/v1/telephony/calls/divert</a>	Divert
POST <a href="https://webexapis.com/v1/telephony/calls/transfer">https://webexapis.com/v1/telephony/calls/transfer</a>	Transfer
POST <a href="https://webexapis.com/v1/telephony/calls/park">https://webexapis.com/v1/telephony/calls/park</a>	Park
POST <a href="https://webexapis.com/v1/telephony/calls/retrieve">https://webexapis.com/v1/telephony/calls/retrieve</a>	Retrieve
POST <a href="https://webexapis.com/v1/telephony/calls/startRecording">https://webexapis.com/v1/telephony/calls/startRecording</a>	Start Recording
POST <a href="https://webexapis.com/v1/telephony/calls/stopRecording">https://webexapis.com/v1/telephony/calls/stopRecording</a>	Stop Recording
POST <a href="https://webexapis.com/v1/telephony/calls/pauseRecording">https://webexapis.com/v1/telephony/calls/pauseRecording</a>	Pause Recording
POST <a href="https://webexapis.com/v1/telephony/calls/resumeRecording">https://webexapis.com/v1/telephony/calls/resumeRecording</a>	Resume Recording
POST <a href="https://webexapis.com/v1/telephony/calls/transmitDtmf">https://webexapis.com/v1/telephony/calls/transmitDtmf</a>	Transmit DTMF
POST <a href="https://webexapis.com/v1/telephony/calls/push">https://webexapis.com/v1/telephony/calls/push</a>	Push
POST <a href="https://webexapis.com/v1/telephony/calls/pickup">https://webexapis.com/v1/telephony/calls/pickup</a>	Pickup
POST <a href="https://webexapis.com/v1/telephony/calls/bargeIn">https://webexapis.com/v1/telephony/calls/bargeIn</a>	Barge In
GET <a href="https://webexapis.com/v1/telephony/calls">https://webexapis.com/v1/telephony/calls</a>	List Calls
GET <a href="https://webexapis.com/v1/telephony/calls/{callId}">https://webexapis.com/v1/telephony/calls/{callId}</a>	Get Call Details
GET <a href="https://webexapis.com/v1/telephony/calls/history">https://webexapis.com/v1/telephony/calls/history</a>	List Call History

# Webhook Notifications/Events

- Webhook API to manage webhooks:  
<https://developer.webex.com/docs/api/v1/webhooks>
- Resource: telephony\_calls
- Events: created, updated, deleted

<https://developer.webex.com/docs/webhooks>

# Telephony\_call event example

```
{
  "id": "Y2lzY2...wMTc5",
  "name": "d9c193c3-4787-4726-b9fa-6acff173e15a",
  "targetUrl": "https://c780-149-249-133-109.ngrok.io/callevnt/Y2lzY29...ZWizZGE",
  "resource": "telephony_calls",
  "event": "created",
  "orgId": "Y2lz...mUzZTc",
  "createdBy": "Y2lz...ZGE",
  "appId": "Y2lzY29zc...5NzZlZWQ0ODM1",
  "ownedBy": "creator",
  "status": "active",
  "created": "2022-03-18T14:53:51.669Z",
  "actorId": "Y2lzY2...2FjZWizZGE",
  "data": {
    "eventType": "received",
    "eventTimestamp": "2022-03-18T14:54:02.442Z",
    "callId": "Y2lzY2...AxNDYxOTow",
    "callSessionId": "Zjg1OWExYTytNDI5NS00OTU0LWEwYzktMDY0MjFjOTY5Mzk3",
    "personality": "terminator",
    "state": "alerting",
    "remoteParty": {
      "name": "Henry Green",
      "number": "7101",
      "personId": "Y2lzY29zc...zNTg",
      "privacyEnabled": false,
      "callType": "location"
    },
    "appearance": 1,
    "created": "2022-03-18T14:54:02.440Z"
  }
}
```

Webhook ID

Webhook name

Target URL

Resource "telephony\_calls" → call event

Created → new call

Id of app used to create the webhook

Information about the actual call



# Webex Calling APIs Overview

## PROVISIONING

## CALL CONTROL

## ANALYTICS & REPORTING

### Customer Journey

Setup, Onboard, Manage

Call, Meet, Collaborate

Achieve Customer Success

- Manage users, phone #s, locations, & services
- Assign licenses
- Create and manage location features

- Place, answer, hang up calls
- Stop / start / pause recording
- Transmit DTMF digits
- List active calls / get history

- Detailed call records
- Onboarding, usage, & quality reporting
- Automated reporting setup

- Installation, activation, & onboarding
- Ongoing services management & care
- Self-service via partner portal

- Custom enterprise calling integrations
- Cloud business platform integration
- Custom app development

- User training & adoption services
- Business process design & optimization
- Vertical solutions design & oversight

### Representative Tasks

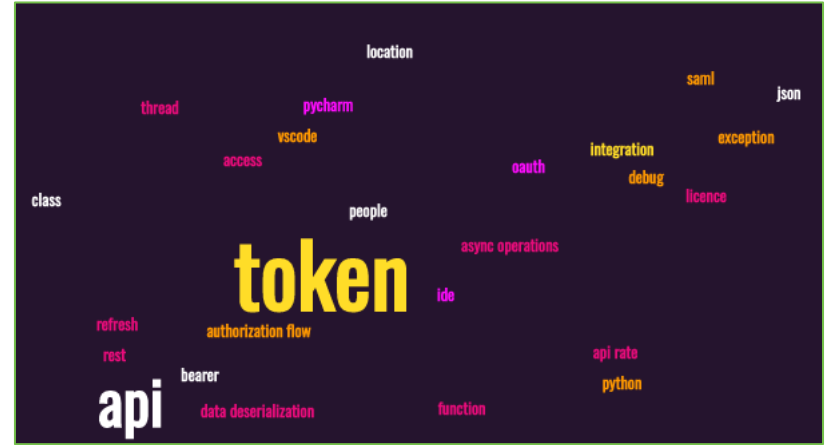
### Sample Solutions

# Getting Started



# Using Webex APIs

- Documentation at: <https://developer.webex.com/>
- But: Steep learning curve
- A lot of concepts to master
- SDK helps to abstract from the “dirty details”



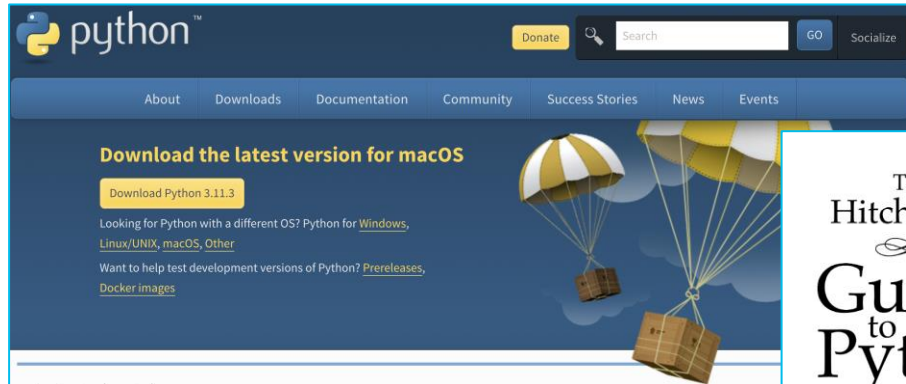
# Developer Sandbox

- Sandbox
  - playground to test API calls
  - Avoid impact on production org
- Limited to 10 users
- Allows to test capabilities not available w/ Webex free plans
- No Cisco PSTN
  - Can add Local Gateway for PSTN access
  - Working on a solution to get PSTN added to sandbox

<https://developer.webex.com/docs/developer-sandbox-guide>

# Installing Python

- Installers are available at <https://www.python.org/downloads/>
- Mac tip: install Python via Homebrew: <https://docs.python-guide.org/starting/install3/osx/>
  - Avoids issues with GNU readline (for example when using <https://pypi.org/project/cmd2/>)



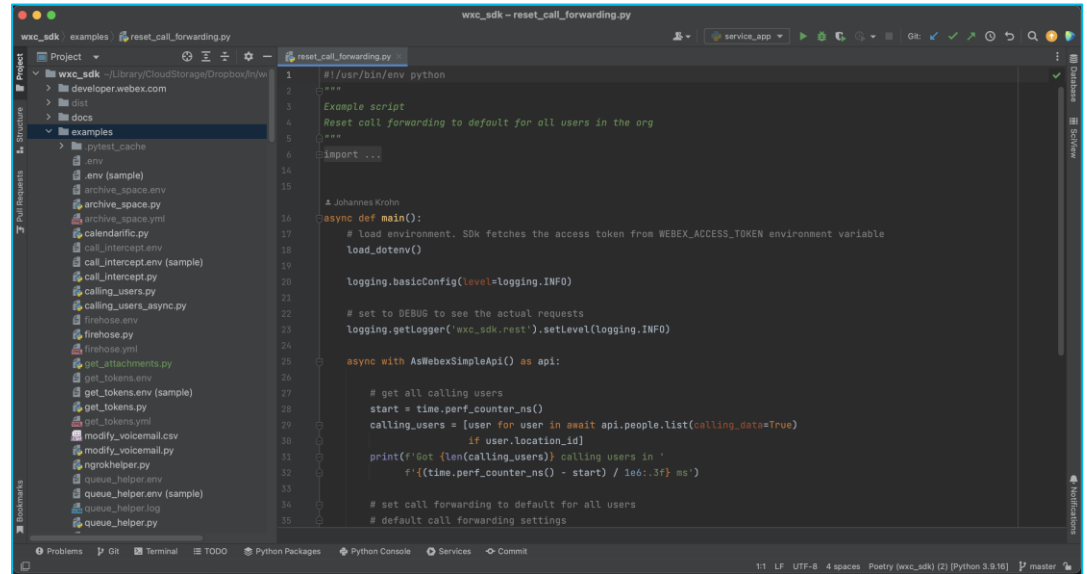
# Tools

# IDE – Integrated Development Environment

- Helps to develop and test your application

- Features

- GUI
- Editor
- Build automation
- Syntax highlighting
- Debugger
- Integration w/ revision control system (e.g. Git)
- ...



# Syntax Highlighting

- What Do you prefer?
- This?

```
def get_attachments():  
    def assert_folder(p_state, base_path, room_id, room_folder):  
        ''' make sure that the folder is created for the room  
        '''  
        if not os.path.lexists(base_path):  
            # base directory needs to be created  
            logging.debug('Base directory %s does not exist' % base_path)  
            os.mkdir(base_path)  
  
        full_path = os.path.join(base_path, room_folder)  
  
        if room_id not in p_state:  
            p_state[room_id] = {}  
            room_state = p_state[room_id]  
  
        if 'folder' not in room_state:  
            logging.debug('No previous folder for room %s' % room_folder)  
            # the folder for this room hasn't been created before  
            i = 0  
            base_folder = room_folder  
            while True:  
                full_path = os.path.join(base_path, room_folder)  
                try:  
                    os.mkdir(full_path)  
                    logging.debug('Created folder %s' % full_path)  
                except FileExistsError:
```



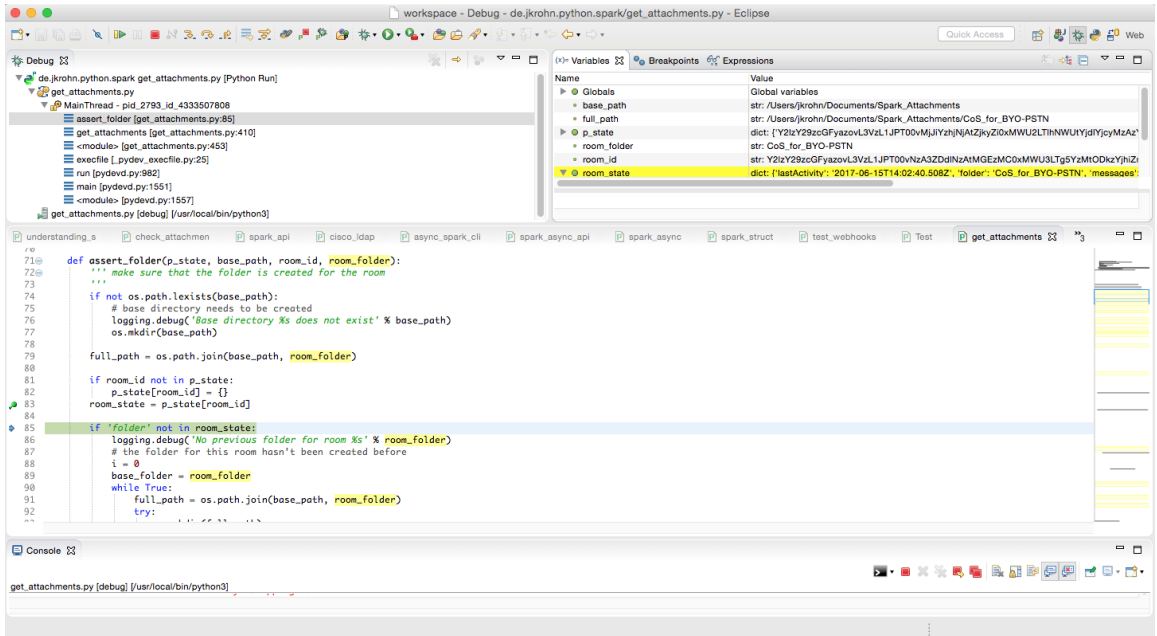
# Syntax Highlighting

- What Do you prefer?
- Or this?

```
def get_attachments():  
  
def assert_folder(p_state, base_path, room_id, room_folder):  
    """ make sure that the folder is created for the room  
    """  
    if not os.path.lexists(base_path):  
        # base directory needs to be created  
        logging.debug('Base directory %s does not exist' % base_path)  
        os.mkdir(base_path)  
  
    full_path = os.path.join(base_path, room_folder)  
  
    if room_id not in p_state:  
        p_state[room_id] = {}  
        room_state = p_state[room_id]  
  
    if 'folder' not in room_state:  
        logging.debug('No previous folder for room %s' % room_folder)  
        # the folder for this room hasn't been created before  
        i = 0  
        base_folder = room_folder  
        while True:  
            full_path = os.path.join(base_path, room_folder)  
            try:  
                os.mkdir(full_path)  
                logging.debug('Created folder %s' % full_path)  
            except FileExistsError:
```

# Live Debugger

- Live Debugger allows to
    - Set breakpoints
    - Check variables
    - Evaluate expressions
- Essential for effective SW development



# IDEs for Python

- [IDLE \(Standard IDE\)](#)
- [PyCharm](#)
- [VS Code](#)
- [PythonAnywhere](#)
- [Cloud9 \(AWS\)](#)



**PyCharm**



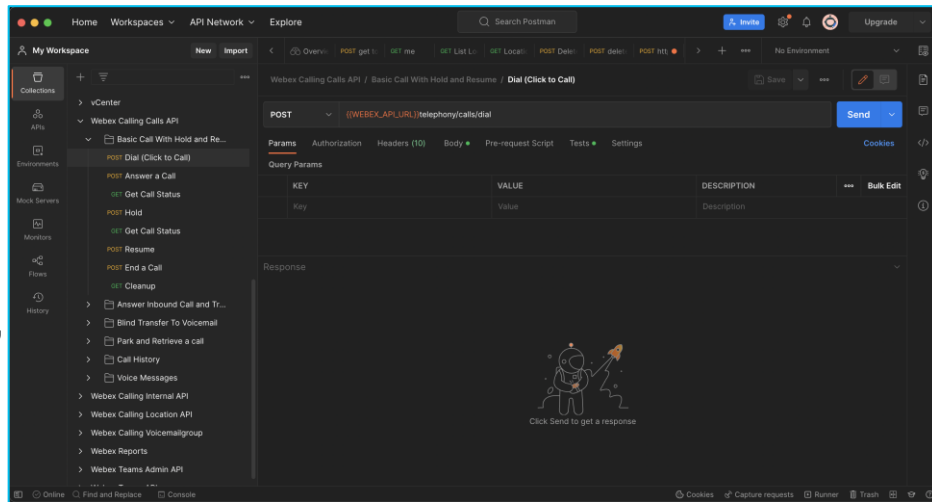
pythonanywhere





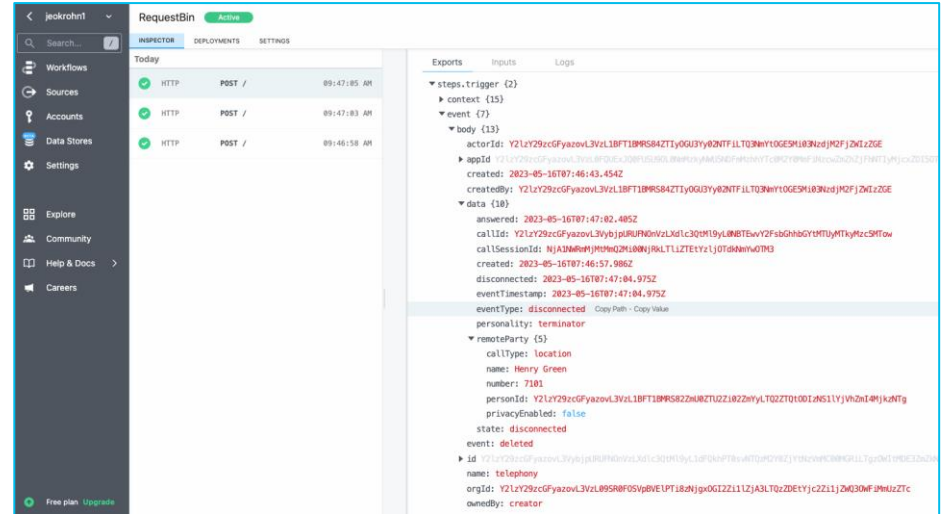
# Postman: Test APIs

- Share, test, document & monitor APIs
- Easily test API calls
- Generate code (Python, curl, ..)
- Create collections
- Available for Mac, Windows, Linux, and Chrome apps
- <https://www.getpostman.com/>
- Postman collection for Webex Teams:  
<https://github.com/CiscoDevNet/postman-webex>



# RequestBin: See Webhooks in Action

- Free service: <https://pipedream.com/requestbin>
- Creates unique URL
- Use case: Webex webhook pointing to Requestbin to test webhook operation
- Provides real-time view on requests hitting the URL



# GitHub

- Git repository hosting service
- Offers
  - Revision control
  - Source code management
- THE place to share your code



# Consuming APIs

# Calling a Webex API Endpoint

## Listing Webex Calling Locations

### List Locations

List locations for an organization.  
Use query parameters to filter the response.  
Long result sets will be split into [pages](#).

GET /v1/locations

```
11 def main():
12     # load .env file
13     load_dotenv()
14
15     # after reading .env file all variables defined in the file are accessible as environment variables
16     access_token = os.getenv('WEBEX_TOKEN')
17     if access_token is None:
18         raise
19
20     url = 'https://webexapis.com/v1/locations'
21
22     with requests.Session() as session:
23         headers = {'Authorization': f'Bearer {access_token}'}
24         response = session.get(url=url, headers=headers)
25         response.raise_for_status()
26         data = response.json()
27         print(f'{len(data["items"])} locations found')
28         for location in data['items']:
29             print(location)
30
31     # look for locations in California
32     ca_locations = [location for location in data['items']
33                     if location['address']['state'] == 'CA']
34     print(f'{len(ca_locations)} locations in CA')
35     print(', '.join(loc['name'] for loc in ca_locations))
```

URL of the endpoint

Session() from requests module is used

Fabricate the Authorization header

Call the endpoint

Check for errors

Parse the JSON response into a dict

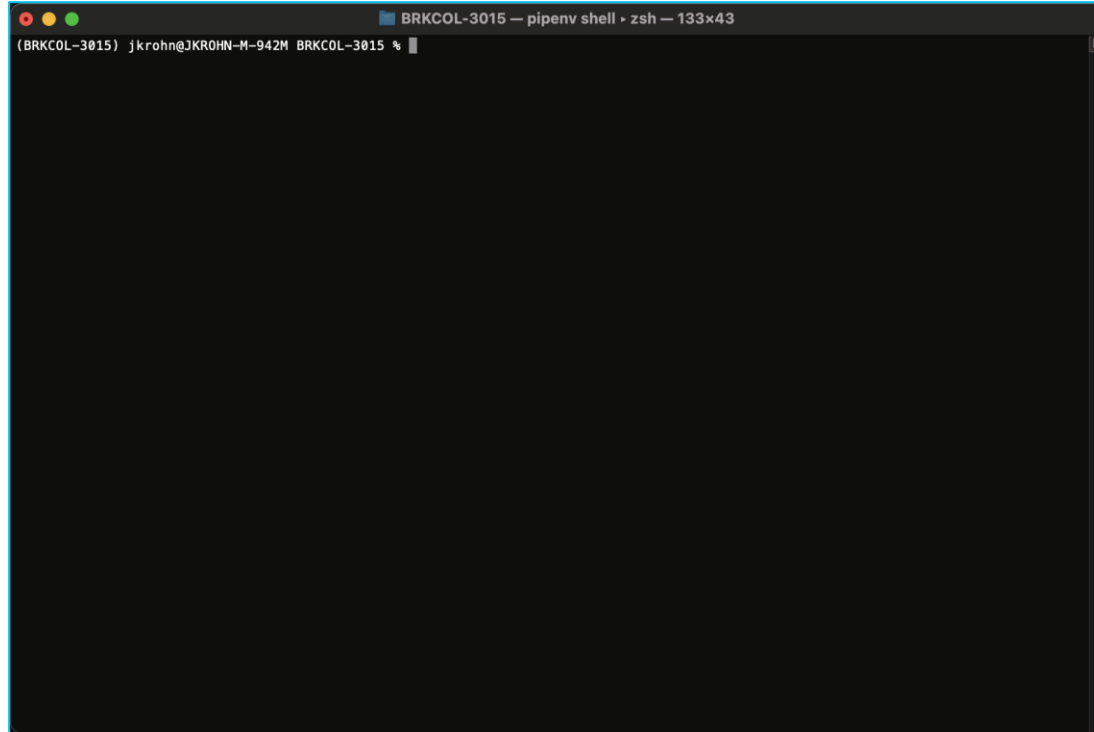
Accessing the response values as dict keys

[https://github.com/jeokrohn/BRKCOL-3015/blob/main/list\\_locations\\_direct.py](https://github.com/jeokrohn/BRKCOL-3015/blob/main/list_locations_direct.py)



# Calling a Webex API Endpoint

## Listing Webex Calling Locations



# Calling a Webex API Endpoint

## Listing Webex Calling Locations

### List Locations

List locations for an organization.  
Use query parameters to filter the response.  
Long result sets will be split into [pages](#).

GET /v1/locations

```
11 def main():
12     # load .env file
13     load_dotenv()
14
15     # after reading .env fi
16     access_token = os.geten
17     if access_token is None
18         raise
19
20     url = 'https://webexapi
21     with requests.Session()
22         headers = {'Authori
23         response = session.
24         response.raise_for_
25         data = response.js
26         print(f'{len(data["
27         for location in dat
28             print(location)
29
30     # look for locations in California
31     ca_locations = [location for location in data['items']
32                     if location['address']['state'] == 'CA']
33     print(f'{len(ca_locations)} locations in CA')
34     print(', '.join(loc['name'] for loc in ca_locations))
```

That was easy, but..

- Accessing dictionary values by key is hard and error prone
- Missing handling of 429 responses (throttling)
- Missing pagination handling
- Handling of additional parameters (name, id)

There has to be a better way?!

# wxc\_sdk: SDK for Webex Calling APIs

- PyPi: <https://pypi.org/project/wxc-sdk/>
- Homepage: [https://github.com/jeokrohn/wxc\\_sdk](https://github.com/jeokrohn/wxc_sdk)
- Documentation: <https://wxc-sdk.readthedocs.io/en/latest/>
- Simple SDK to work with Webex APIs
  - Focus on Webex Calling specific endpoints ... and more
- Takes care of all the “ugly” stuff
  - JSON (de-)serialisation, authentication, 429 retries,
  - Pagination, ...
  - Logging
- Python objects for all API objects
  - Tab completion → efficient coding
- Actively maintained
  - New API endpoints will be added continuously
- Foundation for your provisioning automation and other projects around Webex Calling

```
#!/usr/bin/env python
"""
Demonstration of how to call a Webex API endpoint using the SDK
"""
import os

import wxc_sdk
from dotenv import load_dotenv

def main():
    load_dotenv()

    # after reading .env file all variables defined in the file are accessible as environment variables
    access_token = os.getenv('WEBEX_TOKEN')

    with wxc_sdk.WebexSimpleApi(tokens=access_token) as api:
        locations = list(api.locations.list())
        print(f'{len(locations)} locations found')
        for location in locations:
            print(location)

        ca_locations = [location for location in locations
                        if location.address.state == 'CA']
        print()
        print(f'{len(ca_locations)} locations in CA')
        print(', '.join(loc.name for loc in ca_locations))

if __name__ == '__main__':
    main()
```

# Calling a Webex API Endpoint

## Listing Webex Calling Locations using the SDK

```
#!/usr/bin/env python
"""
Demonstration of how to call a Webex API endpoint using the SDK
"""
import os

import wxc_sdk
from dotenv import load_dotenv

def main():
    load_dotenv()

    # after reading .env file all variables defined in the file are accessible as environment variables
    access_token = os.getenv('WEBEX_TOKEN')

    with wxc_sdk.WebexSimpleApi(tokens=access_token) as api:
        locations = list(api.locations.list())
        print(f'{len(locations)} locations found')
        for location in locations:
            print(location)

        ca_locations = [location for location in locations
                        if location.address.state == 'CA']
        print()
        print(f'{len(ca_locations)} locations in CA')
        print(', '.join(loc.name for loc in ca_locations))

if __name__ == '__main__':
    main()
```

→ The API object

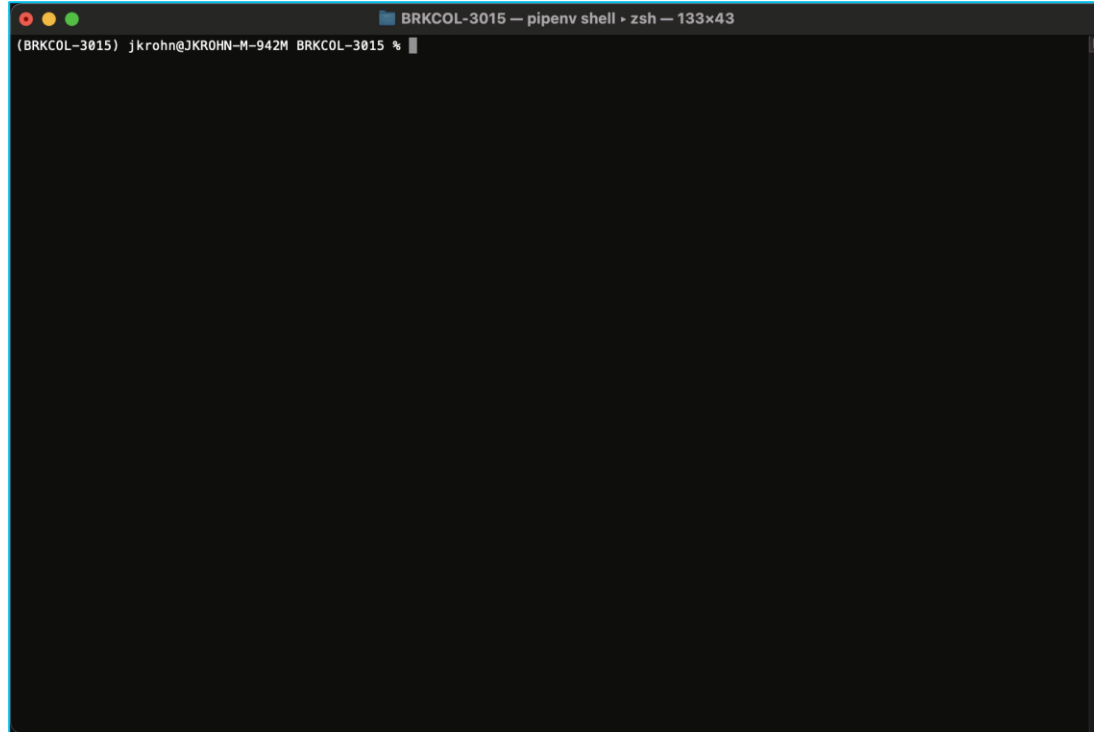
→ Get list of locations

→ Access data using attributes  
of Python classes

[https://github.com/jeokrohn/BRKCOL-2015/blob/main/list\\_locations\\_sdk.py](https://github.com/jeokrohn/BRKCOL-2015/blob/main/list_locations_sdk.py)

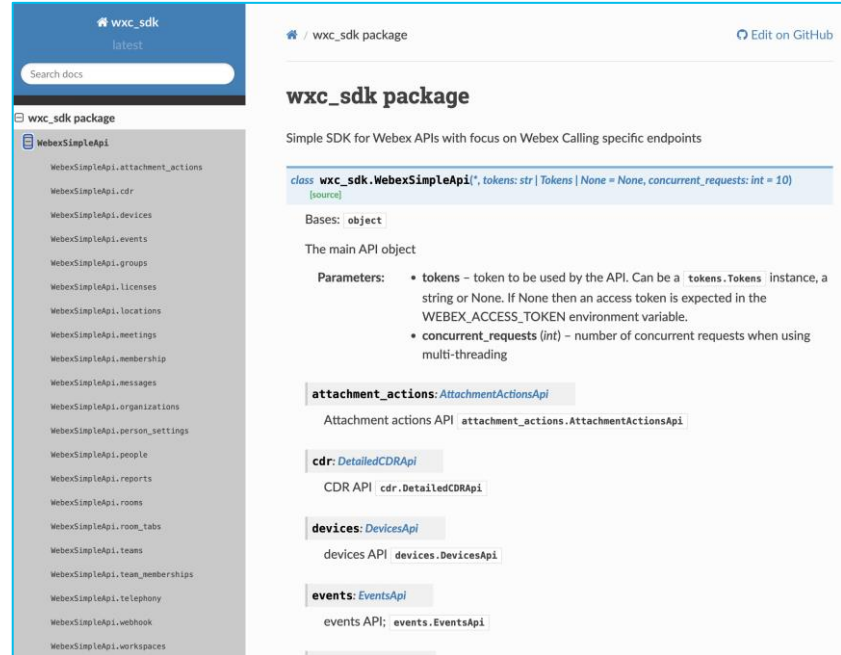
# Calling a Webex API Endpoint

## Listing Webex Calling Locations using the SDK



# wxc\_sdk: Comprehensive Coverage

- SDK covers all Webex Calling specific API endpoints
- Additionally:
  - Licenses, memberships, messages, people, teams, team memberships, webhooks, ...
- Easy token management



The screenshot displays the documentation for the `wxc_sdk` package. On the left, a sidebar lists various API endpoints under the `WebexSimpleApi` class, including `attachment_actions`, `cdr`, `devices`, `events`, `groups`, `licenses`, `locations`, `meetings`, `membership`, `messages`, `organizations`, `person_settings`, `people`, `reports`, `rooms`, `room_tabs`, `teams`, `team_memberships`, `telephony`, `webhook`, and `workspaces`. The main content area shows the package description: "Simple SDK for Webex APIs with focus on Webex Calling specific endpoints". It includes the class definition `wxc_sdk.WebexSimpleApi` with parameters `tokens` and `concurrent_requests`. Below this, it lists the main API object and its parameters, followed by links to specific API classes like `AttachmentActionsApi`, `DetailedCDRApi`, `DevicesApi`, and `EventsApi`.

<https://wxc-sdk.readthedocs.io/en/latest/>

# Tokens

# Tokens

## Why and How?

- Access tokens are required to authorize API access
- .. Can be obtained in different ways:
  - Personal access token (developer token)
  - Integration (OAuth2 authorization flow)
  - Service App
- **NEVER(!!!) store tokens in your source files**
- **NEVER(!!!) push tokens to GitHub repositories**
- **NEVER(!!!) share tokens in any shape or form**
- Best practice:
  - In your code read access token from environment variable
  - Use `dotenv.loadenv()` to load `.env` file with environment variables
  - Exclude `.env` from version control (Git) by adding exclusion in `.gitignore`
  - Integration tokens can be cached in local files .. but make sure to restrict access and not push to GitHub

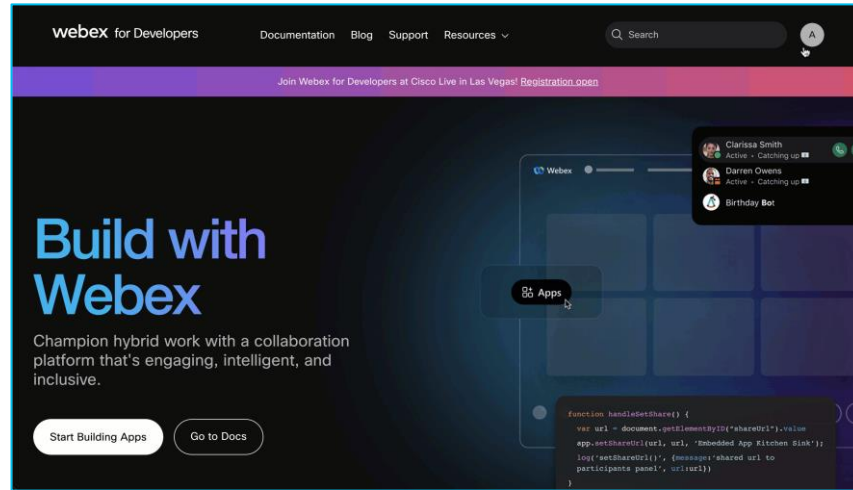
```
GET https://webexapis.com/v1/people
User-Agent: python-requests/2.30.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Authorization: Bearer MGY4MzNjNzgt***
content-type: application/json; charset=utf-8
```

<https://developer.webex.com/docs/getting-started>



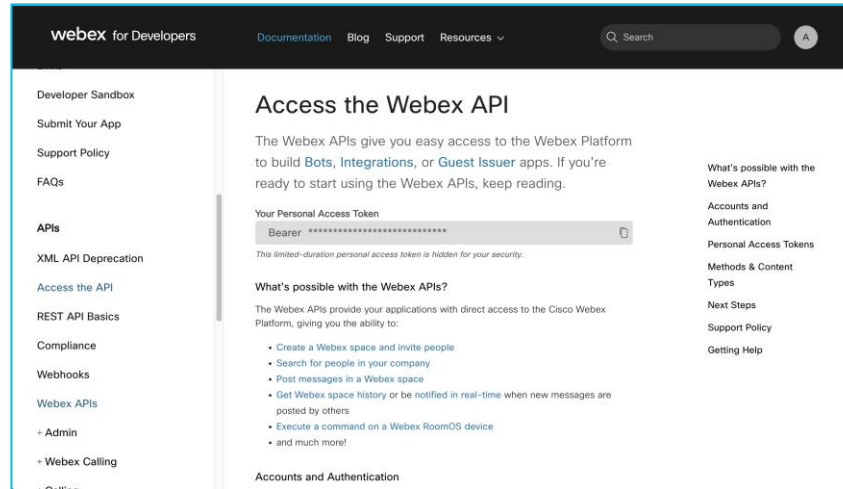
# Personal Access Token

- From developer.webex.com
- Limited lifetime (12 h)
- Should NEVER be used in production
- Testing only



# Personal Access Token

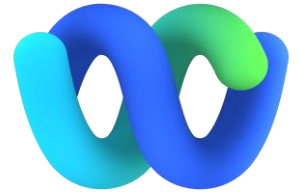
- From developer.webex.com
- Limited lifetime (12 h)
- Should NEVER be used in production
- Testing only



# Integration – The Better Way to Obtain Tokens

- Act on behalf of a Webex user
  - Access equivalent to a real Webex User (limited by authorized scopes)
- Invoke Webex APIs on behalf of user
- Requires authorization of integration by user
  - OAuth Grant Flow to authenticate user and ask for authorization
  - User approves authorization levels (scopes) requested by the integration
- Each Integration has a client ID, client secret and redirect URI
- Documentation: <https://developer.webex.com/docs/integrations>

# OAuth Authorization Code Flow



1. Application Requests *auth code*

Browser redirect to Webex Authentication

2. Webex returns the *auth code* to application

Browser redirect to Application

3. Request an *access token*

HTTP GET request to Webex API

4. Application gets *access token* and *refresh token*

HTTP GET response from Webex API

# Integration Tokens in Scripts Using wxc\_sdk

- `wxc_sdk` offers an easy way to work with cached tokens in scripts
- Cache tokens in YAML file
- Get tokens from OAuth flow redirecting to <http://localhost:6001/redirect>
- Spin up temporary (primitive) server to handle final step of OAuth flow

Prepare integration based on values read from environment

Read, create, refresh, cache tokens

```
access_token: ZGVlY2***
expires_at: '2023-05-30T14:35:37.246110+00:00'
refresh_token: NTlk***
refresh_token_expires_at: '2023-08-14T14:08:57.246110+00:00'
token_type: Bearer
```

```
#!/usr/bin/env python
"""
Demonstration of how to call a Webex API endpoint using the SDK with cached integration tokens
"""
import os
from os.path import splitext, basename

from dotenv import load_dotenv

from wxc_sdk import WebexSimpleApi
from wxc_sdk.integration import Integration
from wxc_sdk.scopes import parse_scopes


def get_tokens():
    """
    get (cached) integration tokens
    """
    env_vars = ('INTEGRATION_CLIENT_ID', 'INTEGRATION_CLIENT_SECRET', 'INTEGRATION_SCOPES')
    if not all(os.getenv(s) for s in env_vars):
        raise KeyError(f'Not all required environment variables ({", ".join(env_vars)}) defined.')

    client_id = os.getenv('INTEGRATION_CLIENT_ID')
    client_secret = os.getenv('INTEGRATION_CLIENT_SECRET')
    scopes = parse_scopes(os.getenv('INTEGRATION_SCOPES'))
    integration = Integration(client_id=client_id,
                             client_secret=client_secret,
                             scopes=scopes,
                             redirect_url='http://localhost:6001/redirect')

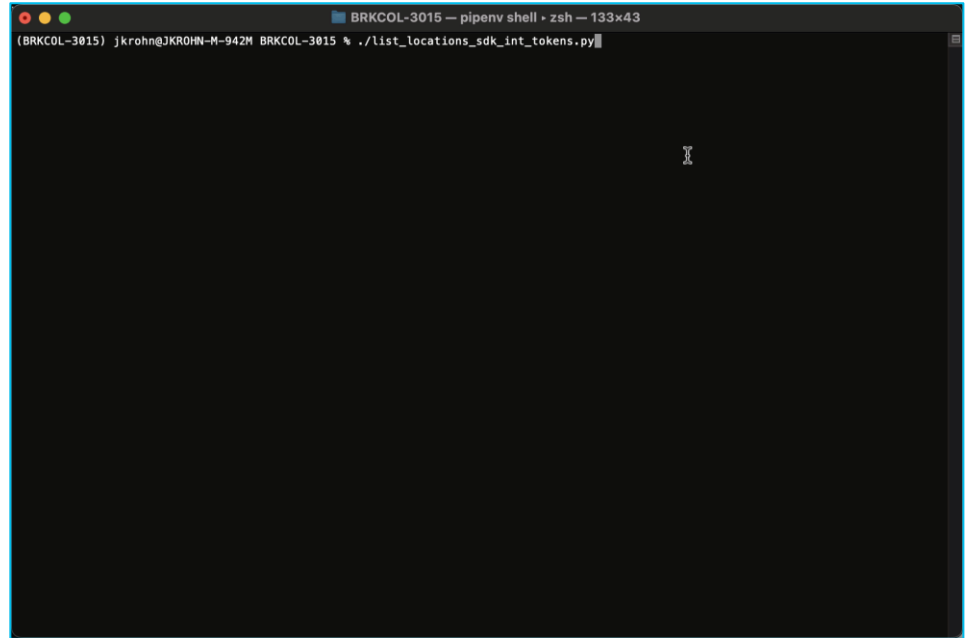
    yml_path = f'{splitext(basename(__file__))[0]}.yml'
    tokens = integration.get_cached_tokens_from_yaml(yml_path=yml_path)

    return tokens
```

[https://github.com/jeokrohn/BRKCOL-3015/blob/main/list\\_locations\\_sdk\\_int\\_tokens.py](https://github.com/jeokrohn/BRKCOL-3015/blob/main/list_locations_sdk_int_tokens.py)

# Integration Tokens in Scripts Using wxc\_sdk

- Without cached tokens OAuth flow gets initiated
- Auth code exchanged for tokens
- Tokens are cached
- Next execution uses cached tokens

A terminal window titled "BRKCOL-3015 — pipenv shell • zsh — 133x43". The prompt shows the user is "jkrohn@JKROHN-M-942M" on the "BRKCOL-3015" host. The command being executed is `./list_locations_sdk_int_tokens.py`. The terminal output is currently blank, with a cursor visible on the line.

```
BRKCOL-3015 — pipenv shell • zsh — 133x43
(BRKCOL-3015) jkrohn@JKROHN-M-942M BRKCOL-3015 % ./list_locations_sdk_int_tokens.py
```

[https://github.com/jeokrohn/BRKCOL-3015/blob/main/list\\_locations\\_sdk\\_int\\_tokens.py](https://github.com/jeokrohn/BRKCOL-3015/blob/main/list_locations_sdk_int_tokens.py)

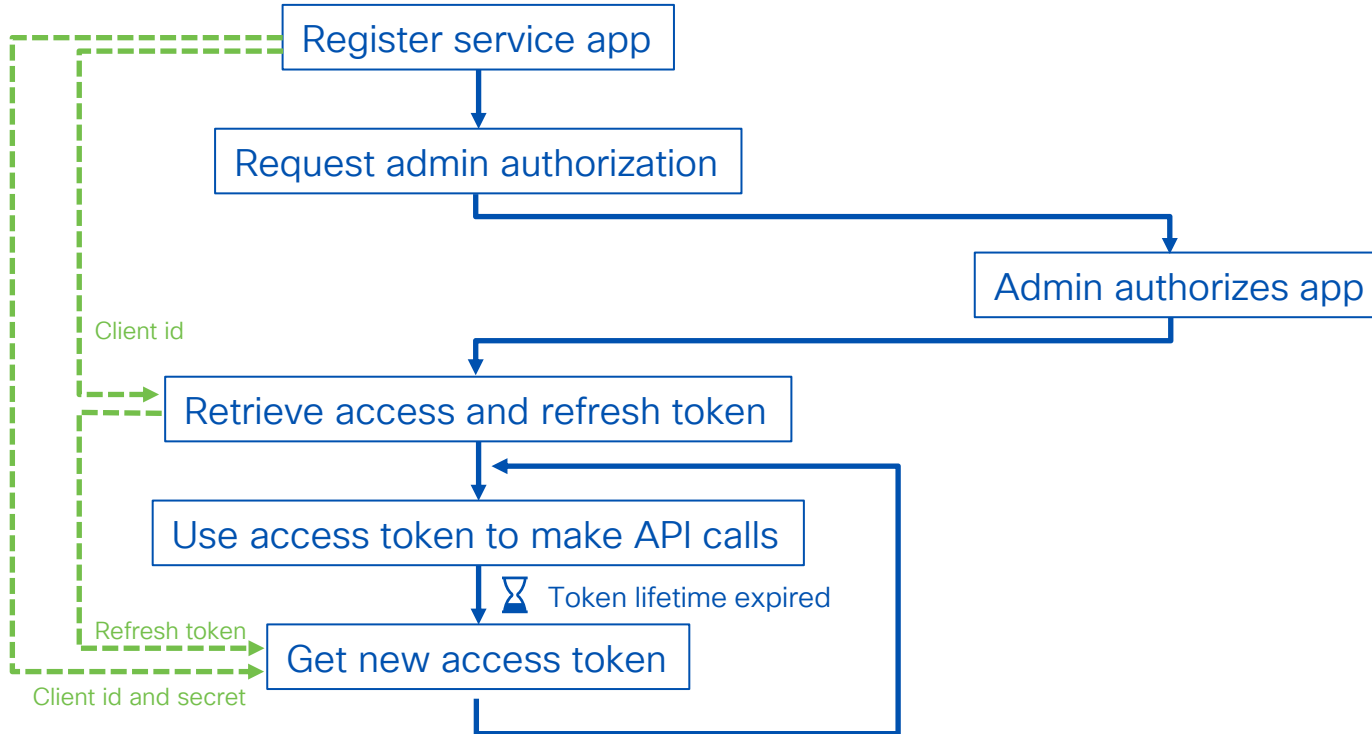
# Service Apps

- Machine account
- Request admin permission independent of user account
- Use cases
  - Provisioning
  - Reporting
  - Scheduling systems
  - ...
- Similar to integrations .. but no user specific authorization flow

# Using Service App Tokens

Developer

Admin





# Using Service App Tokens

Developer

Admin

Register service app

Request a

Client id

Retrieve acc

Use access to

Refresh token

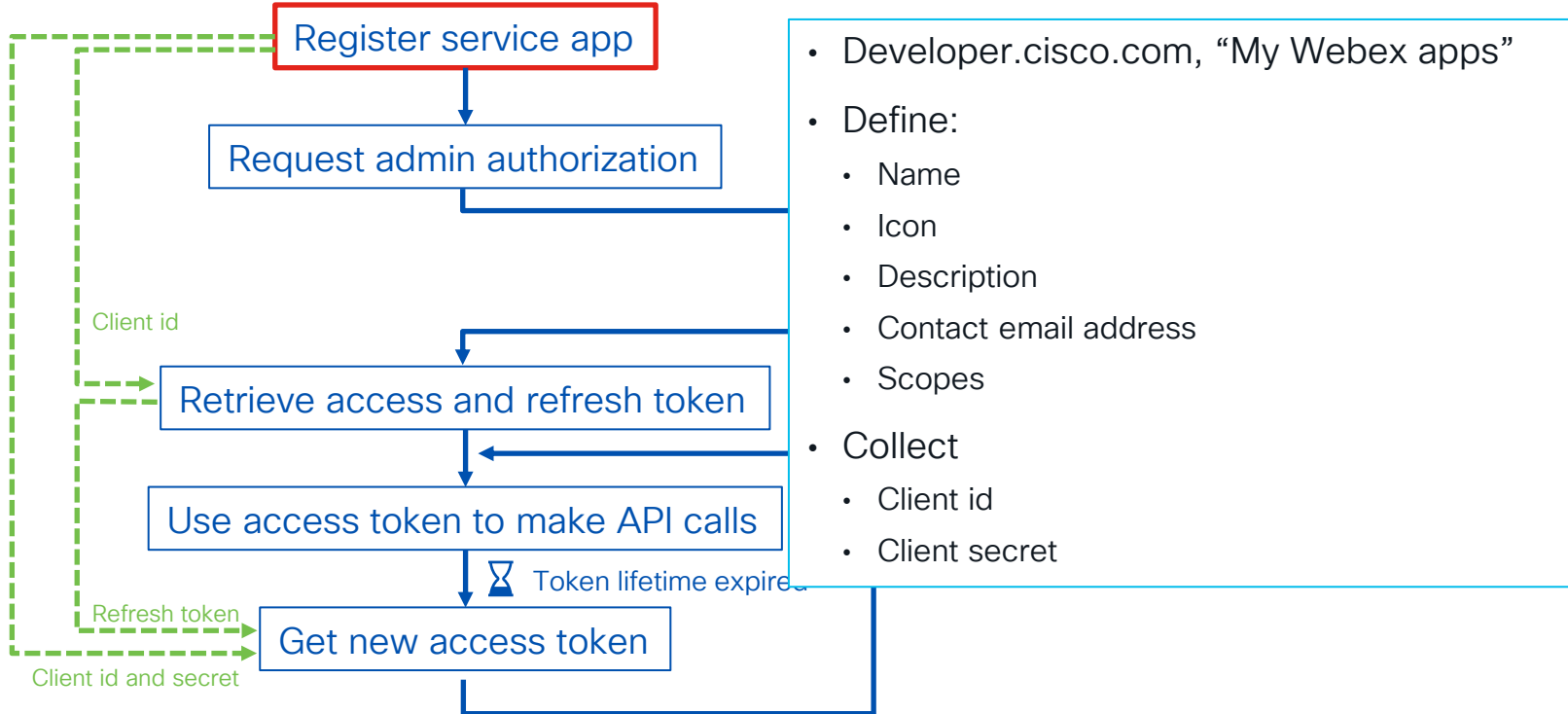
Get new

Client id and secret

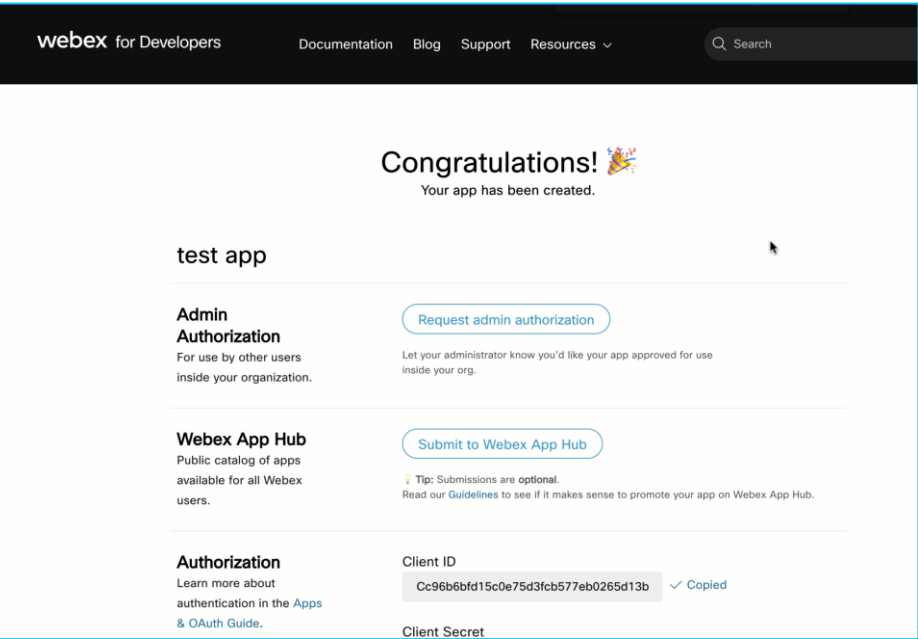
# Using Service App Tokens

Developer

Admin



**CISCO** *Live!*



# Using Service App Tokens

Developer

Admin

webex Control Hub

Search

Overview

Getting Started Guide

Alerts center

MONITORING

Analytics

Troubleshooting

Reports

MANAGEMENT

Customers

Users

Groups

Locations

Workspaces

Devices

Apps

jkrohn-sandbox.wbx.ai

Set up emergency call settings for US locations in your organization to meet the requirements defined for your state and federal regulations.

### Overview

**Getting Started Guide**

37 %

3 of 8 tasks completed

[View the Getting Started Guide and the recommended tasks](#)

**Updates**

Update your services to the new Webex experience. [Learn more](#)

Update Webex Meetings to the ne...

**New offers**

Allow everyone in your organization to host a Webex Meeting with a Basic Meetings license. [Learn more](#)

**Onboarding**

33 Total users

There is no CSV upload within 180 days

Potential new users: 10

[Review](#) [Enable directory sync](#)

**Webex services** ALL ONLINE

Webex, Calling, Meetings, Hybrid Services, Control Hub, Developer API, Room Devices, Contact Center

**Devices**

8 Total devices

Online 3, Online with issues 0, Offline 5, Expired 0

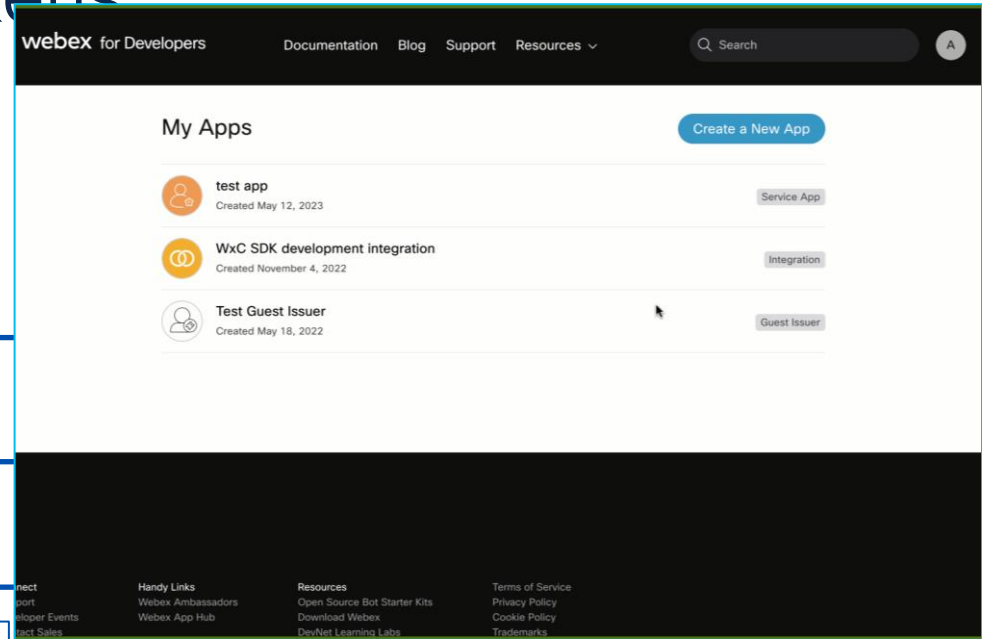
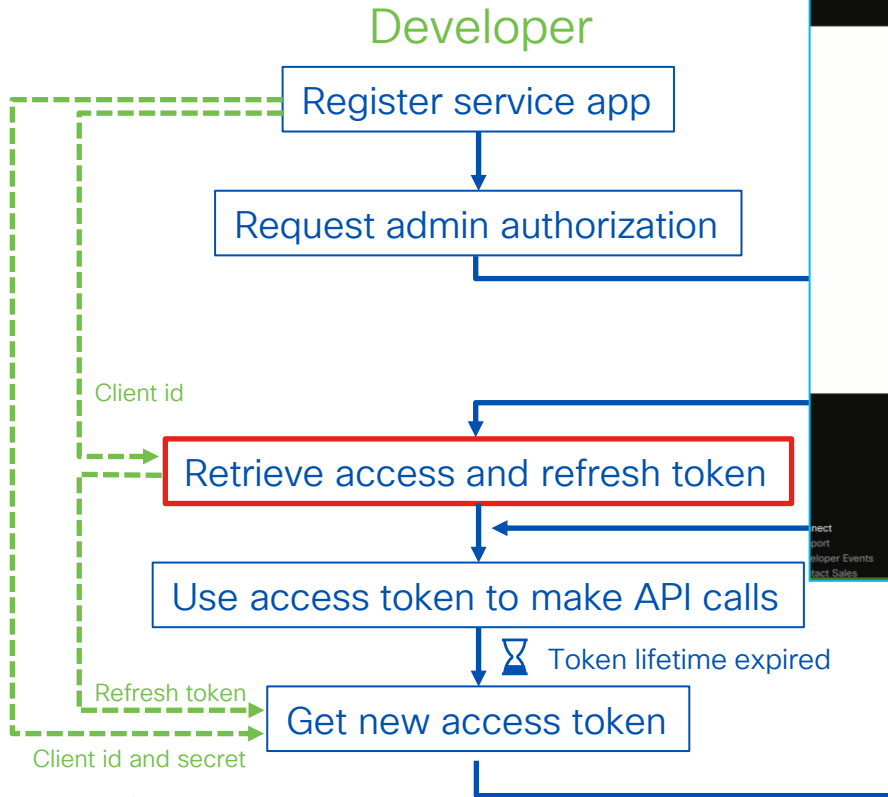
Admin authorizes app

- “Apps” section in Control Hub
- Review scopes
- Authorizing user is documented

Get new access token

Client id and secret

# Using Service App Tokens

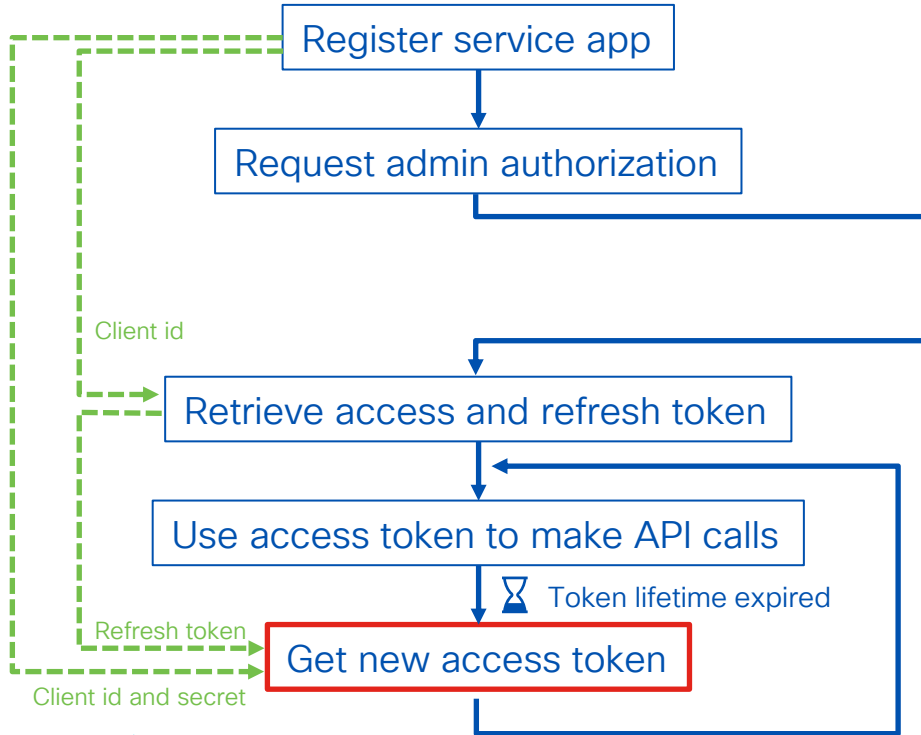


- Developer requests tokens for authorized org
- Client secret needed to authorize the request

# Using Service App Tokens

Developer

Admin



- When getting close to token lifetime
- New access token can be obtained using the refresh token

```
POST https://webexapis.com/v1/access_token
Content-Type: application/x-www-form-urlencoded
--- body ---
grant_type: refresh_token
client_id: ***
client_secret: ***
refresh_token: ***
Response
Content-Type: application/json
--- response body ---
{
  "access_token": "****",
  "expires_in": 1209599,
  "refresh_token": "****",
  "refresh_token_expires_in": 7775954,
  "token_type": "Bearer",
  "scope": "spark:kms ..."
}
```

# Token Overview

	Developer Token	Integration Token	Service App Token
How to get	From developer.webex.com	Requires OAuth auth code authorization flow, web server required	Service app access to org granted by org admin.  Service app owner creates token on developer.webex.com
Granular Access Control	No, always has all scopes	Set of scopes assigned to integration	Set of scopes assigned to service app
Actor	Owner of developer token	User granting authorization	Service app
Lifetime	12 hrs	Access token: 14 d Refresh token: 90 d (extended when getting new access token)	
Use cases	Development, Tests	(Web) applications acting on behalf of user	(Web) services explicitly authorized by org admin

# Use Cases / Examples





# Use Cases

- Scripts
  - Bulk Provisioning: user, calling features, ...
  - Validation: check/verify settings; scripting saves you from navigating through individual menus in ControlHub
  - Automation reduces the risk of errors in repetitive tasks
  - CLI tools
- Integration with existing enterprise management systems/tools
- Backend for web services (portal applications)

# SDK Examples

- Examples available at:

<https://wxc-sdk.readthedocs.io/en/latest/examples.html>

[https://github.com/jeokrohn/wxc\\_sdk/tree/master/examples](https://github.com/jeokrohn/wxc_sdk/tree/master/examples)

```
(wxc-sdk-NNVrdgRm-py3.9) jkrohn@JKROHN-M-942M examples % ./reset_call_forwarding.py
```

```
api = WebexSimpleApi()

# get all calling users
start = time.perf_counter_ns()
calling_users = [user for user in api.people.list(calling_data=True)
                  if user.location_id]
print(f'Got {len(calling_users)} calling users in '
      f'{(time.perf_counter_ns() - start) / 1e6:.3f} ms')

# set call forwarding to default for all users
with ThreadPoolExecutor() as pool:
    # default call forwarding settings
    forwarding = PersonForwardingSetting.default()

    # schedule update for each user and wait for completion
    start = time.perf_counter_ns()
    list(pool.map(
        lambda user: api.person_settings.forwarding.configure(person_id=user.person_id,
                                                                forwarding=forwarding),
        calling_users))
    print(f'Reset call forwarding to default for {len(calling_users)} users in '
          f'{(time.perf_counter_ns() - start) / 1e6:.3f} ms')
```

# Examples

- `queue_helper.py`: read/update call queue join state of users
- `call_intercept.py`: read/update call intercept settings of a user
- `us_holidays_async.py`: create schedule with US holidays for all US location
- `reset_call_forwarding.py`: reset call forwarding settings for all users
- `users_wo_devices.py`: identify users without devices
- `catch_tns.py`: pool unassigned TNs on hunt groups to catch calls to unassigned TNs

# Example: User Web Portal

# User Web Portal

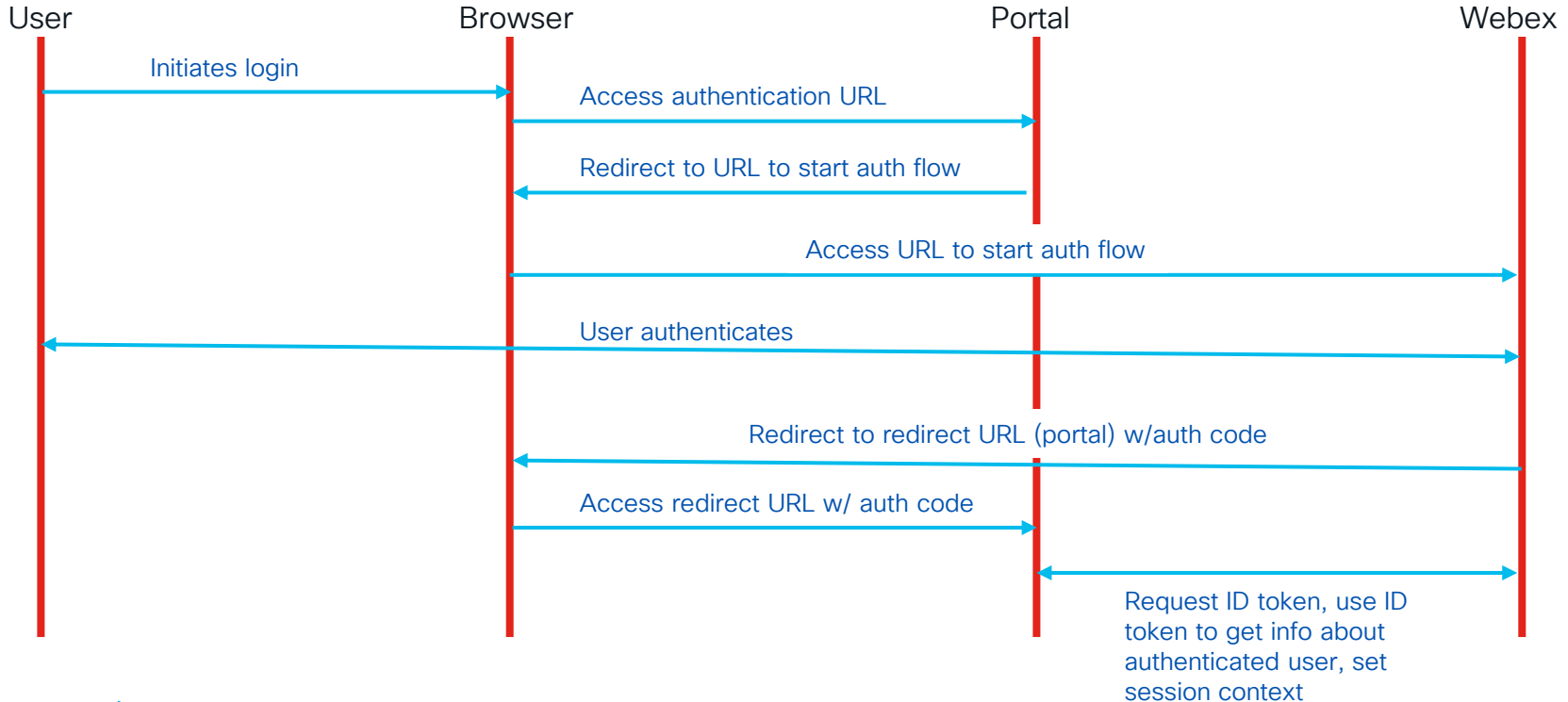
- Only a limited set of configuration options is available for users on settings.webex.com
- User portal can expose additional options (which usually require admin privileges)
- Perfect example for using service app tokens
  - User does not have the required privileges → integration token granted by user not sufficient

# Concepts

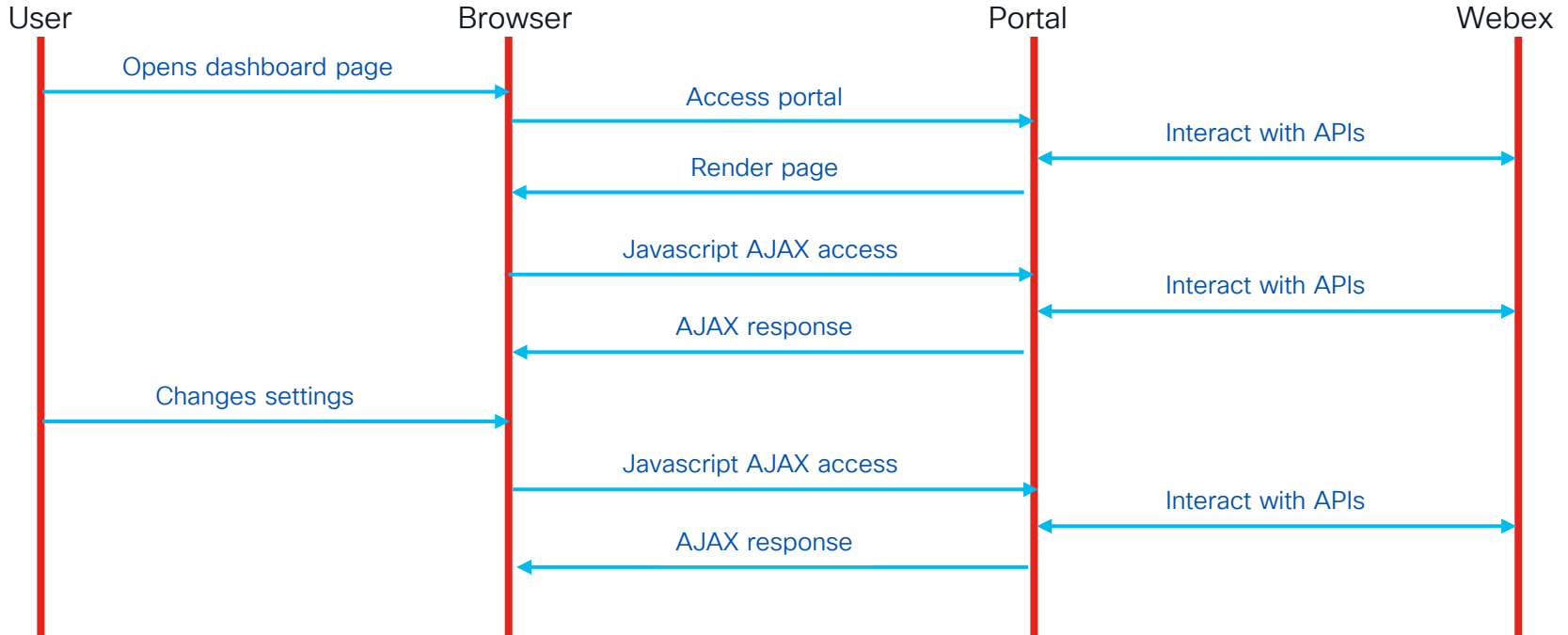
- Web based portal
- Authenticate users using “Login with Webex”
- For provisioning operations portal uses service app tokens
- Browser only interacts with portal endpoints
  - Includes Javascript Ajax access
  - Portal acts as proxy between browser and Webex APIs

<https://developer.webex.com/docs/login-with-webex>  
<https://github.com/jeokrohn/BRKCOL-3015>

# Login with Webex



# Portal Transactions





# Demo Web App

# Disclaimer, Where to Go Next

- Demo code is not “production ready”
  - Missing token lifetime monitoring
  - Filesystem based session backend
  - ...
- Where to go next
  - Add roles: can be based on groups in Webex
  - Additional functions: manage devices, bulk update, ...

# Closing



# References

- Webex for Developers: <https://developer.webex.com/>
- Examples for the session on GitHub  
<https://github.com/jeokrohn/BRKCOL-3015>
- Python SDK: <https://pypi.org/project/wxc-sdk/>
- SDK Examples available at:  
<https://wxc-sdk.readthedocs.io/en/latest/examples.html>  
[https://github.com/jeokrohn/wxc\\_sdk/tree/master/examples](https://github.com/jeokrohn/wxc_sdk/tree/master/examples)
- Call control bot: <https://wxc-callcontrol.readthedocs.io/en/latest/>

# Fill out your session surveys!



Attendees who fill out a minimum of four session surveys and the overall event survey will get **Cisco Live-branded socks** (while supplies last)!



Attendees will also earn 100 points in the **Cisco Live Challenge** for every survey completed.



**These points** help you get on the leaderboard and increase your chances of winning daily and grand prizes

# Continue your education



- Visit the Cisco Showcase for related demos
- Book your one-on-one Meet the Engineer meeting
- Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs
- Visit the On-Demand Library for more sessions at [www.CiscoLive.com/on-demand](https://www.CiscoLive.com/on-demand)

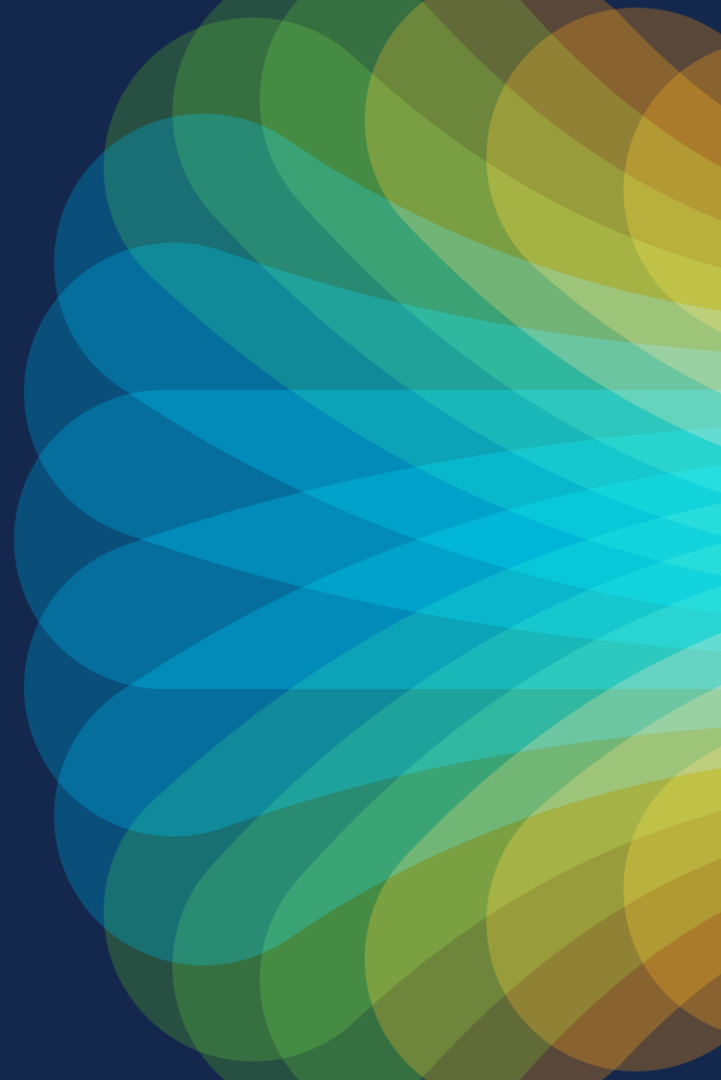


The bridge to possible

# Thank you



#CiscoLive

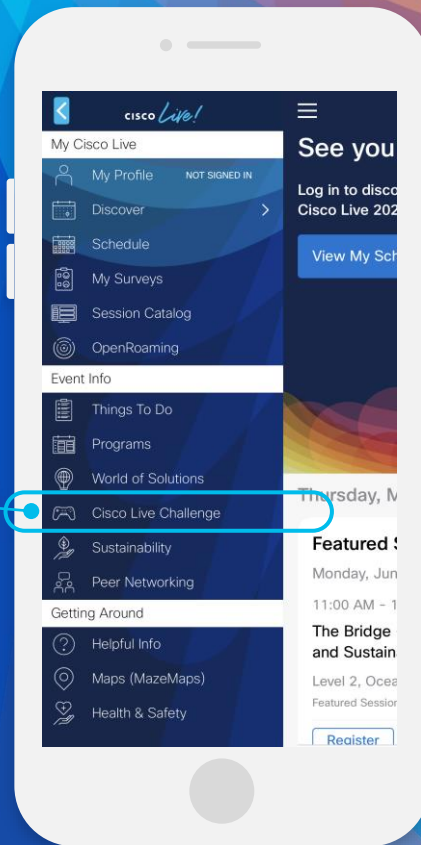
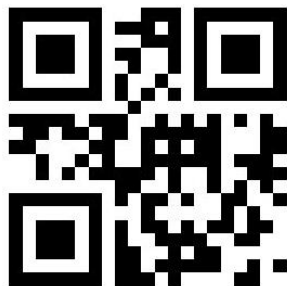


# Cisco Live Challenge

Gamify your Cisco Live experience!  
Get points for attending this session!

## How:

- 1 Open the Cisco Events App.
- 2 Click on 'Cisco Live Challenge' in the side menu.
- 3 Click on View Your Badges at the top.
- 4 Click the + at the bottom of the screen and scan the QR code:





The background is a vibrant, abstract graphic. It features a central bright white light source from which numerous colorful rays emanate, creating a sunburst or starburst effect. The rays transition through a spectrum of colors including yellow, orange, red, and various shades of blue and green. Overlaid on this are several large, semi-transparent, wavy shapes in similar color tones, giving the overall image a sense of motion and energy.

cisco *Live!*

Let's go

#CiscoLive