

Agile Management of Webex Calling

There is an API for that

Johannes Krohn

Principal Technical Marketing Engineer

Join the conversation!

Scan the QR code to be added to the Webex space for Q&A and more



Agenda

01 Why APIs?

02 Coverage/Capabilities

03 Getting started

04 Use cases/Examples

05 Closing

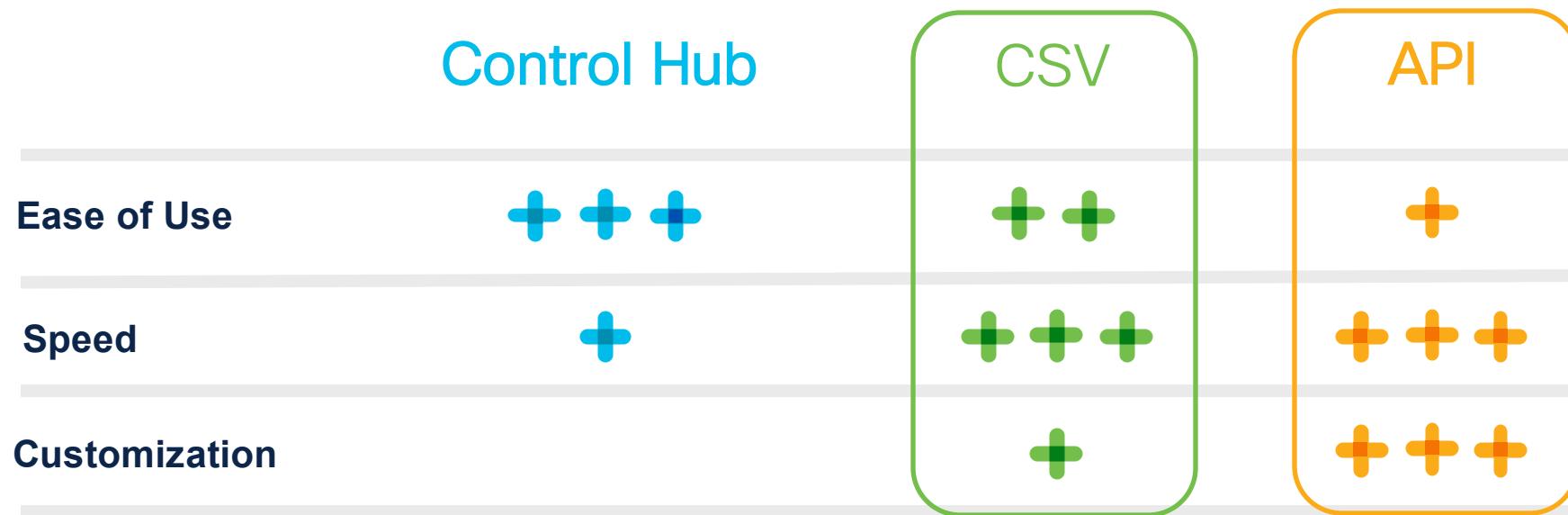
Why APIs?

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.”¹
- APIs
 - Enabler for open systems integration
 - Universally available
 - Unleash developer innovation



Webex Calling Provisioning Methods



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, ...)
 - ...
- Comprehensive coverage of Webex Calling
- 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, mute/unmute, 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/docs/api/changelog> for updates

References:

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

Webex Calling Provisioning APIs

- Locations, <https://developer.webex.com/calling/docs/api/v1/locations>
 - List locations
 - Get location details
 - Create/Update locations
 - Delete: not possible
- People, <https://developer.webex.com/admin/docs/api/v1/people>*
 - List
 - CRUD
 - callingData parameter to access calling data (set, modify TN/extension)

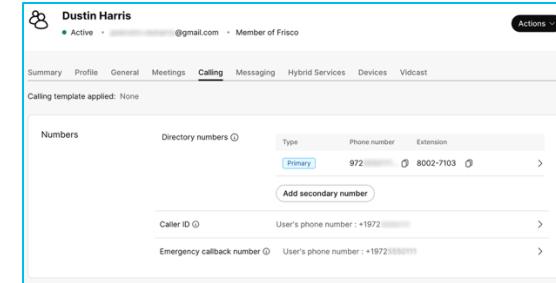
*For user management use of the SCIMv2 user API is preferred:
<https://developer.webex.com/admin/docs/api/v1/scim-2-users>

Preferred endpoint for TN/extension assignment/updates:
[PATCH /licenses/users](#)

Webex Calling Provisioning APIs

- Person Settings,
<https://developer.webex.com/calling/docs/api/v1/user-call-settings-1-2>
<https://developer.webex.com/calling/docs/api/v1/user-call-settings-2-2>
 - Settings found in person's Calling tab in Control Hub
- Comprehensive coverage

- API REFERENCE
- ▽ All APIs
 - > Call Controls
 - > Call Routing
 - > Call Settings For Me
 - > Calling Service Settings
 - > Client Call Settings
 - > Conference Controls
 - > Contact Center with Calling For Me
 - > Converged Recordings
 - > DECT Devices Settings
 - > Device Call Settings
 - > Devices
 - > Emergency Services Settings
- Features: Auto Attendant
 - > Features: Call Park
 - > Features: Call Pickup
 - > Features: Call Queue
 - > Features: Hunt Group
 - > Features: Paging Group
 - > Features: Announcement Playlist
 - > Features: Announcement Repository
 - > Features: Call Recording
 - > Features: Customer Experience Essentials
- Features: Hot Desking
 - > Features: Operating Modes
 - > Features: Single Number Reach
 - > Features: Virtual Extensions
- People
 - > PSTN
 - > Recording Report
 - > Reports
 - > Reports: Detailed Call History
 - > Send Activation Email
 - > User Call Settings (1/2)
 - > User Call Settings (2/2)
 - > Virtual Line Call Settings
 - > Workspace Call Settings (1/2)
 - > Workspace Call Settings (2/2)
 - > Workspaces
- Numbers
 - > Directory numbers
 - > Type
 - > Phone number
 - > Extension
- Calmer ID O
 - > User's phone number : +1972 8002-7103
- Emergency callback number O
 - > User's phone number : +1972 8002-7103



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

Answer	Post
Barge In	Post
Dial	Post
Divert	Post
Get Call Details	Get
Hangup	Post
Hold	Post
List Call History	Get
List Calls	Get
Mute	Post
Park	Post
Pause Recording	Post
Pickup	Post
Push	Post
Reject	Post
Resume	Post
Resume Recording	Post
Retrieve	Post
Start Recording	Post
Stop Recording	Post
Transfer	Post
Transmit DTMF	Post
Unmute	Post

Webhook Notifications/Events

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

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

Telephony_call event example

```
{  
  "id": "Y2lzY2...wMTc5",  
  "name": "d9c193c3-4787-4726-b9fa-6acff173e15a",  
  "targetUrl": "https://c780-149-249-133-109.ngrok.io/callevent/Y2lzY29...ZWIzZGE",  
  "resource": "telephony_calls",  
  "event": "created",  
  "orgId": "Y2lz...mUzZTc",  
  "createdBy": "Y2lz...ZGE",  
  "appId": "Y2lzY29zc...5Nzz1ZWQ0ODM1",  
  "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": "Zjg1OWExYTtNDI5NS00OTU0LWEwYzktMDY0MjFjOTY5Mzk3",  
    "personality": "terminator",  
    "state": "alerting",  
    "remoteParty": {  
      "name": "Henry Green",  
      "number": "7101",  
      "personId": "Y2lzY29z...zNTg",  
      "privacyEnabled": false,  
      "callType": "location"  
    },  
    "appearance": 1,  
    "created": "2022-03-18T14:54:02.440Z"  
  }  
}
```

Diagram annotations:

- Webhook ID: Points to the "id" field.
- Webhook name: Points to the "name" field.
- Target URL: Points to the "targetUrl" field.
- Resource “telephony_calls” → call event: Points to the "resource" field.
- Created → new call: Points to the "event" field.
- Id of app used to create the webhook: Points to the "appId" field.
- Information about the actual call: Points to the "data" object.

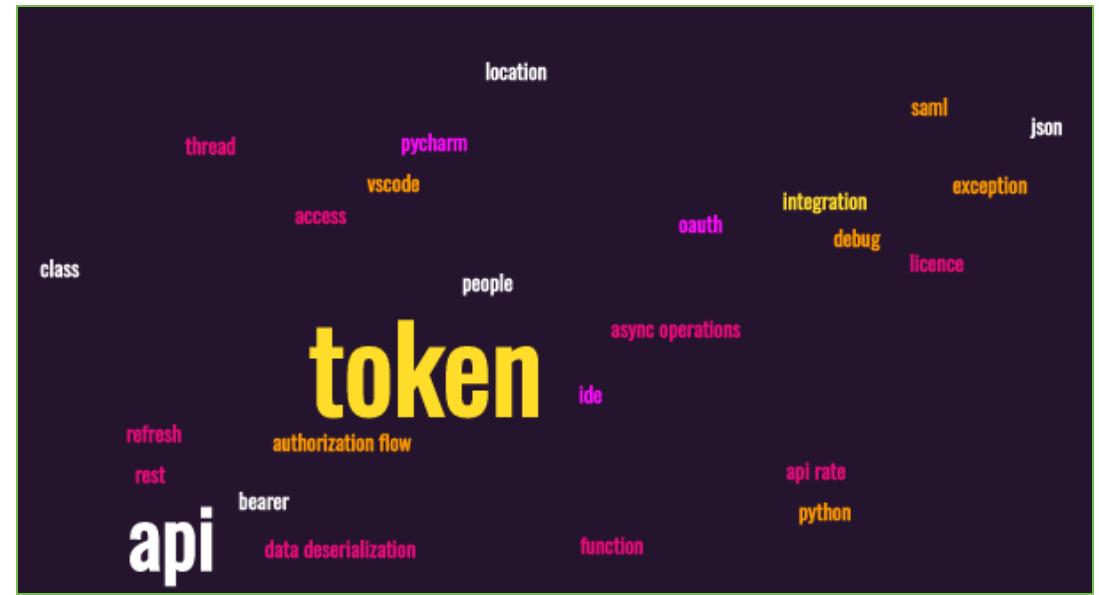
Webex Calling APIs Overview

	PROVISIONING	CALL CONTROL	ANALYTICS & REPORTING
Customer Journey	Setup, Onboard, Manage	Call, Meet, Collaborate	Achieve Customer Success
Representative Tasks	<ul style="list-style-type: none">Manage users, phone #s, locations, & servicesAssign licensesCreate and manage location features	<ul style="list-style-type: none">Place, answer, hang up callsStop / start / pause recordingTransmit DTMF digitsList active calls / get history	<ul style="list-style-type: none">Detailed call recordsOnboarding, usage, & quality reportingAutomated reporting setup
Sample Solutions	<ul style="list-style-type: none">Installation, activation, & onboardingOngoing services management & careSelf-service via partner portal	<ul style="list-style-type: none">Custom enterprise calling integrationsCloud business platform integrationCustom app development	<ul style="list-style-type: none">User training & adoption servicesBusiness process design & optimizationVertical solutions design & oversight

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 on-premises PSTN (Local Gateway) for PSTN access

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

Installing Python

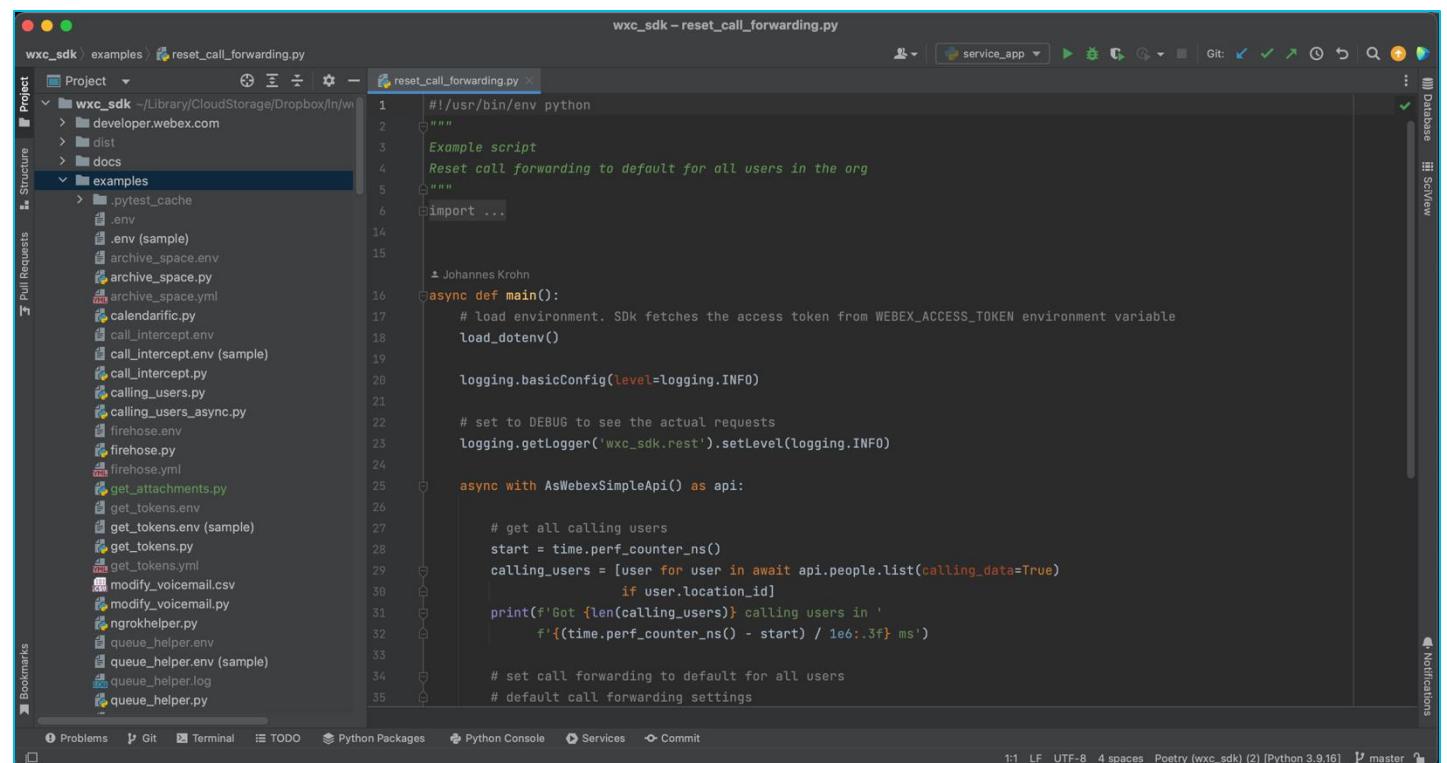
- Installers are available at <https://www.python.org/downloads/>
- Mac tip: install Python via Homebrew: <https://docs.python-guide.org/starting/install3/osx/>
 - Avoids issues with GNU readline (for example when using <https://pypi.org/project/cmd2/>)

The screenshot shows the Python.org homepage. On the left, there's a section for downloading Python 3.11.3 for macOS. It includes links for Windows, Linux/UNIX, macOS, and Other operating systems. Below this, there's information about Docker images and pre-releases. In the center, there's a large graphic of two boxes descending from the sky on yellow and white striped parachutes. To the right of the graphic is the title "The Hitchhiker's Guide to Python" in a large serif font, with a "Star" button and a count of 24,679 stars below it. At the bottom of this section is a search bar labeled "Search the doc". On the far right, there's a promotional banner for "The Hitchhiker's Guide to Python" with the text "Master Real-World Python Skills With a Community of Experts", "Watch Now", and "Level Up With Unlimited Access to Our Vast Library of Python Tutorials and Video Lessons". Below the banner, the text "Installing Python 3 on Mac OS X" is displayed, followed by a small thumbnail image.

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)
 - ...



The screenshot shows a dark-themed IDE interface with a Python file named 'reset_call_forwarding.py' open in the center editor window. The code in the file is as follows:

```
#!/usr/bin/env python
"""
Example script
Reset call forwarding to default for all users in the org
"""

import ...

# Johannes Krohn

async def main():
    # load environment. SDK fetches the access token from WEBEX_ACCESS_TOKEN environment variable
    load_dotenv()

    logging.basicConfig(level=logging.INFO)

    # set to DEBUG to see the actual requests
    logging.getLogger('wxc_sdk.rest').setLevel(logging.DEBUG)

    async with AsWebexSimpleApi() as api:

        # get all calling users
        start = time.perf_counter_ns()
        calling_users = [user for user in await 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
        # default call forwarding settings
```

The left sidebar shows a project structure with a 'examples' folder containing various files like archive_space.py, calendarific.py, etc. The bottom navigation bar includes tabs for Problems, Git, Terminal, TODO, Python Packages, Python Console, Services, and Commit.

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

The screenshot shows the Eclipse IDE interface during a Python debugging session. The top status bar indicates "workspace - Debug - de.jkrohn.python.spark/get_attachments.py - Eclipse". The left pane displays the call stack for the current thread (MainThread - pid_2793_id_4333507808), with the current frame at line 85 of get_attachments.py. The code editor window shows the following Python code:

```
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:
```

The right pane shows the "Variables" view, which lists global variables and their values. The variable "room_state" is highlighted in yellow, indicating it is the current focus of the debugger. The "Value" column for "room_state" shows its dictionary structure.

IDEs for Python

- IDLE (Standard IDE)
- PyCharm
- VS Code
- PythonAnywhere



PyCharm





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 Messaging and Admin APIs:
<https://github.com/CiscoDevNet/postman-webex>

The screenshot shows the Postman application interface. On the left, the sidebar displays 'My Workspace' with various sections like Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. Under 'APIs', there is a collection for 'Webex Calling Calls API' containing several requests such as 'Basic Call With Hold and Resume', 'Dial (Click to Call)', 'Answer a Call', 'Get Call Status', 'Hold', 'Resume', 'End a Call', and 'Cleanup'. The main panel shows a specific POST request for 'Dial (Click to Call)' with the URL `{(WEBEX_API_URL)}telephony/calls/dial`. The 'Headers' tab is selected, showing 'Content-Type: application/json'. Below the request, there's a table for 'Query Params' with columns for 'KEY', 'VALUE', and 'DESCRIPTION'. A note says 'Click Send to get a response' with an illustration of a person holding a rocket. At the bottom, there are tabs for 'Online', 'Find and Replace', and 'Console'.

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

The screenshot shows the RequestBin interface. On the left, there's a sidebar with options like Workflows, Sources, Accounts, Data Stores, and Settings. The main area has tabs for INSPECTOR, DEPLOYMENTS, and SETTINGS, with the INSPECTOR tab active. Below the tabs is a table titled "Today" showing three recent requests:

Type	Method	URL	Time
HTTP	POST	/	09:47:05 AM
HTTP	POST	/	09:47:03 AM
HTTP	POST	/	09:46:58 AM

On the right, there's a detailed view of the first request. It shows the "Exports", "Inputs", and "Logs" sections. The "Logs" section is expanded and displays the following JSON data:

```
steps.trigger {2}
  context {15}
  event {7}
    body {13}
      actorId: Y2lzcGyazovL3VzL1BFT1BMR84ZTiy0GU3y02NTf1LTQ3NmYt0GE5M103NzdjM2FjZWIZG
      appId: Y2lzcGyazovL3VzL0PQEJxJ0fUSU9OLNnIzkyMuNSDhIiIzhyYTc0fIzY0nf1NzceMzJPNNTiyMjcxZD1501
      created: 2023-05-16T07:46:43.454Z
      createdBy: Y2lzcGyazovL3VzL1BFT1BMR84ZTiy0GU3y02NTf1LTQ3NmYt0GE5M103NzdjM2FjZWIZG
      data {10}
        answered: 2023-05-16T07:47:02.405Z
        callId: Y2lzcGyazovL3VybjpURUFNOnVzLXd1c3Qth19yL0NBTEwY2FsbGhhbGYtMTUYMTkyMzc5MTow
        callSessionId: NjAI1MmRmJMMmQ2M100NjRkLT1iZTETy2lJ0TdkNnw0TM3
        created: 2023-05-16T07:46:57.906Z
        disconnected: 2023-05-16T07:47:04.975Z
        eventTimestamp: 2023-05-16T07:47:04.975Z
        eventType: disconnected
        personality: terminator
      remoteParty {5}
        callType: location
        name: Henry Green
        number: 7101
        personId: Y2lzcGyazovL3VzL1BFT1BMR82ZmU0ZTU2Zi02ZmYyLT02ZTQt0D1zN51LyjVhZmI4MjkzNtg
        privacyEnabled: false
        state: disconnected
      event: deleted
      id: Y2lzcGyazovL3VybjpURUFNOnVzLXd1c3Qth19yL1dFQkhPT0svNT0zK2Y0ZjYtNzVnHCB0HGRLLTg2WIT1HDE32mZk
      name: telephony
      orgId: Y2lzcGyazovL3VzL095R0F05p8VElPT18nJgx0G1Z2Z11ZjA3LTQzZDEtYjczZ11jZWQ30WFjMmUzZtC
      ownedBy: creator
```

GitHub

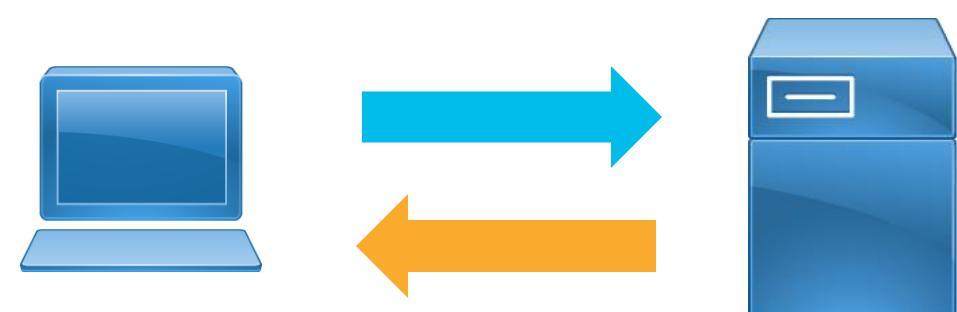
- Git repository hosting service
- Offers
 - Revision control
 - Source code management
- THE place to share your code



Consuming APIs

REST – Representational State Transfer

- Not really a standard – more an architecture
- Uses existing standards: for example HTTP(S) for transport
- All about client-server
- Conceptually similar to web browser accessing web server
- Resources
 - Every resource can be addressed by a URI
 - Methods: GET, PUT, POST, DELETE, HEAD
 - Uniform representation: typically JSON
- Protocol
 - Stateless
 - Client-server



Calling a Webex API Endpoint

Listing Webex Calling 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     with requests.Session() as session:
22         headers = {'Authorization': f'Bearer {access_token}'}
23         response = session.get(url=url, headers=headers)
24         response.raise_for_status()
25         data = response.json()
26         print(f'{len(data["items"])} locations found')
27         for location in data['items']:
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))
```

List Locations

Operation Id: *List_Locations*

Description: List locations for an organization.

- Use query parameters to filter the result set by location name, ID, or organization.
- Long result sets will be split into pages.
- Searching and viewing locations in your organization requires an administrator or location administrator auth token with any of the following scopes: `spark-admin:locations_read`, `spark-admin:people_read` or `spark-admin:device_read`.

GET /locations

<https://developer.webex.com/calling/docs/api/v1/locations/list-locations>

URL of the endpoint

Session() from requests module is used

Fabricate the Authorization header

Call the endpoint

Check for errors

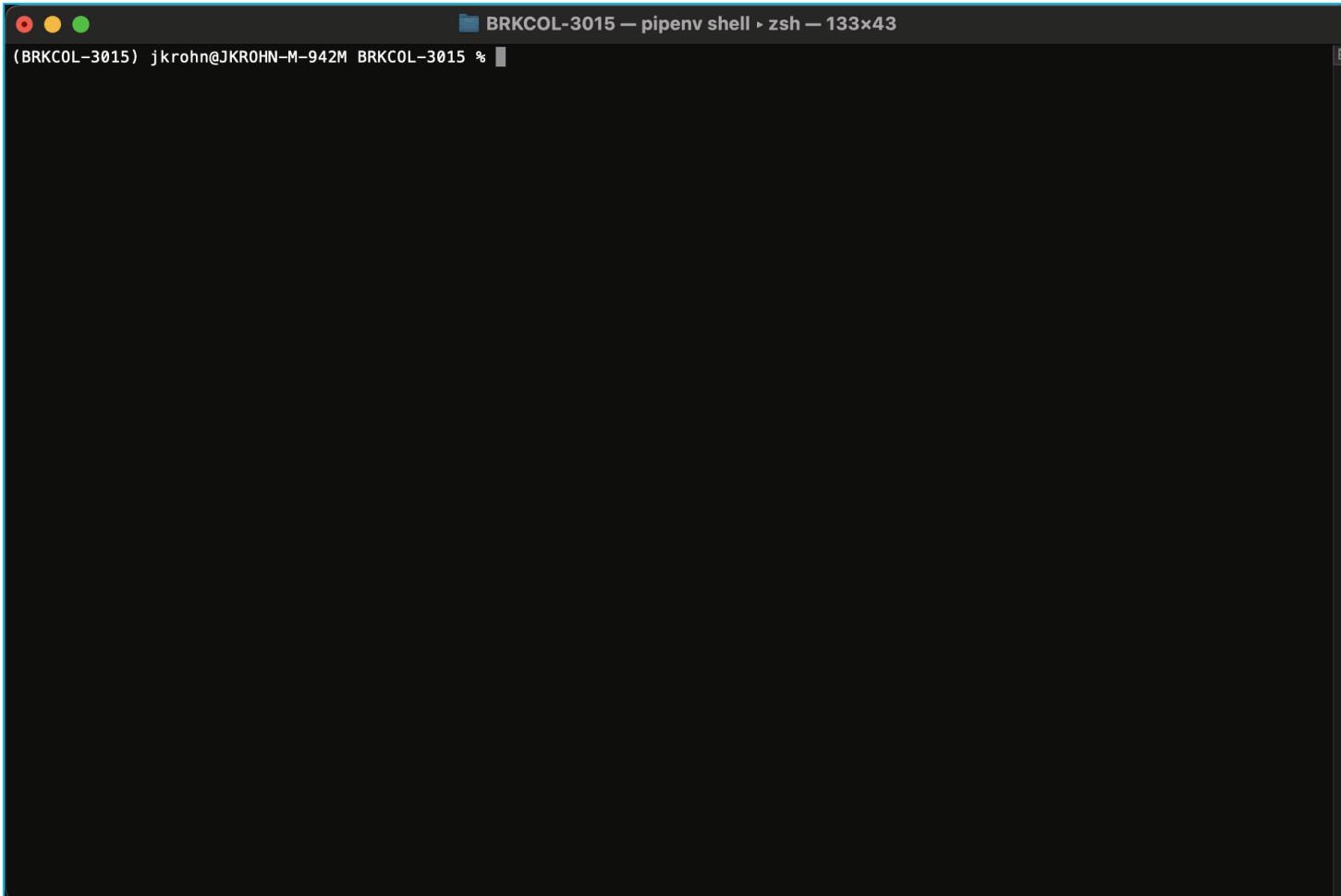
Parse the JSON response into a dict

Accessing the response values as dict keys

https://github.com/jeokrohn/BRKCOL-3015/blob/main/list_locations_direct.py

Calling a Webex API Endpoint

Listing Webex Calling Locations

A screenshot of a terminal window titled "BRKCOL-3015 — pipenv shell > zsh — 133x43". The window has three colored window control buttons (red, yellow, green) at the top left. The title bar also shows the host name "(BRKCOL-3015)" and user name "jkrohn@JKROHN-M-942M". The main area of the terminal is a solid black rectangle, indicating no output or a blank screen.

Calling a Webex API Endpoint

Listing Webex Calling Locations

```
11 def main():
12     # load .env file
13     dotenv()
14
15     # after reading .env file
16     access_token = os.getenv('WEBEX_ACCESS_TOKEN')
17     if access_token is None:
18         raise ValueError("WEBEX_ACCESS_TOKEN environment variable is not set")
19
20     url = 'https://webexapi'
21     with requests.Session() as session:
22         headers = {'Authorization': f'Bearer {access_token}'}
23         response = session.get(url)
24         response.raise_for_status()
25         data = response.json()
26         print(f'{len(data["locations"])} locations found')
27         for location in data["locations"]:
28             print(location)
29
30     # look for locations in California
31     ca_locations = [location for location in data["locations"]
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?!

List Locations

Operation Id: *List_Locations*

Description: List locations for an organization.

- Use query parameters to filter the result set by location name, ID, or organization.
- Long result sets will be split into pages.
- Searching and viewing locations in your organization requires an administrator or location administrator auth token with any of the following scopes: `spark-admin:locations_read`, `spark-admin:people_read` or `spark-admin:device_read`.

GET /locations

<https://developer.webex.com/calling/docs/api/v1/locations/list-locations>

URI of the endpoint

ests module is used
rization header

onse into a dict
onse values as dict keys

wxc_sdk: SDK for Webex Calling APIs

- PyPi: <https://pypi.org/project/wxc-sdk/>
- Homepage: https://github.com/jeokrohn/wxc_sdk
- Documentation: <https://wxc-sdk.readthedocs.io/en>
- Simple SDK to work with Webex APIs
 - Focus on Webex Calling specific endpoints ... and more
- Takes care of all the “ugly” stuff
 - JSON (de-)serialization, 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()
```

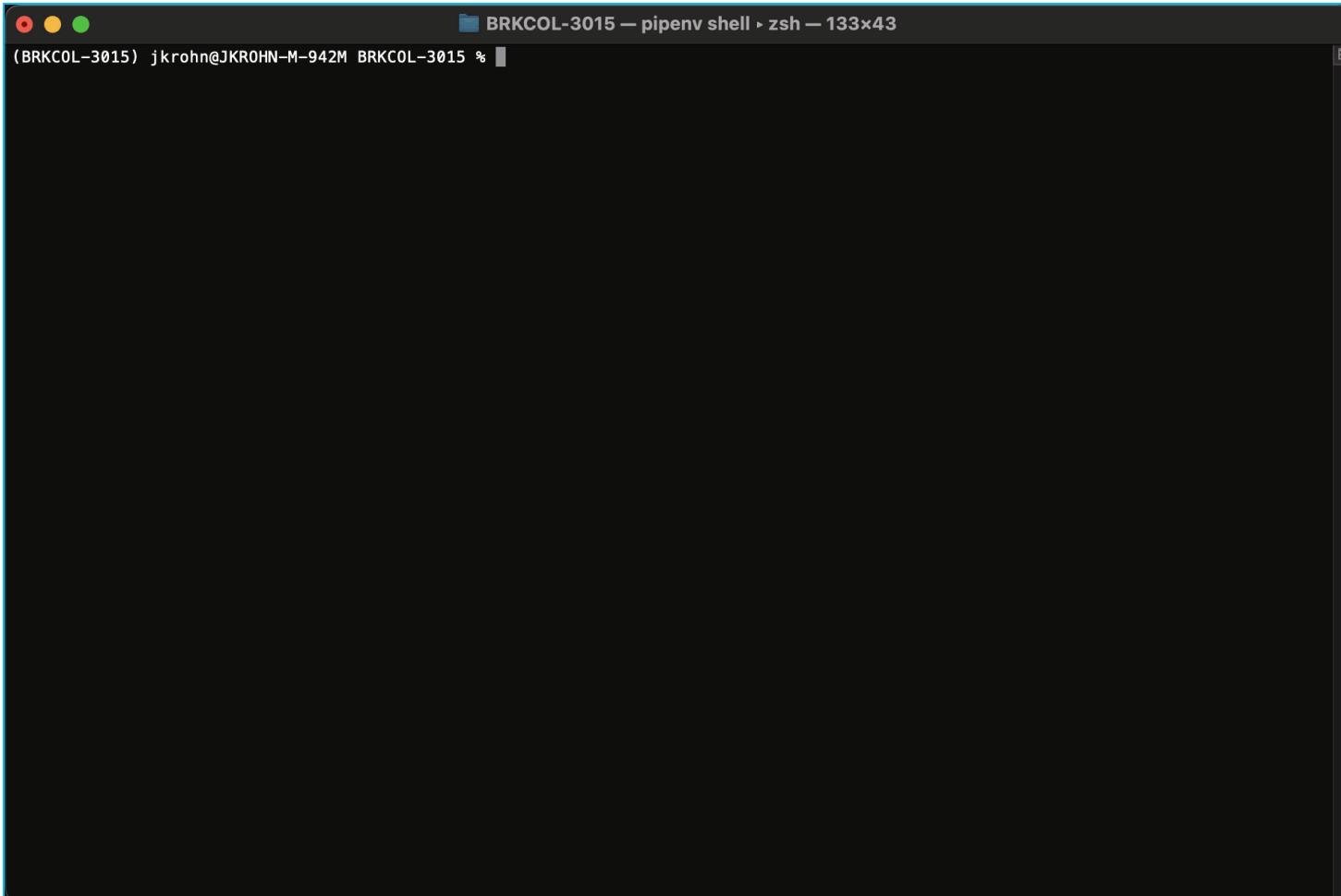
The API object

Get list of locations

Access data using attributes
of Python classes

Calling a Webex API Endpoint

Listing Webex Calling Locations using the SDK



```
BRKCOL-3015 — pipenv shell ▶ zsh — 133x43
(jkrohn@JKROHN-M-942M BRKCOL-3015 %)
```

wxc_sdk: Comprehensive Coverage

- SDK covers all Webex Calling specific API endpoints
- Additionally:
 - Licenses, memberships, messages, people, teams, team memberships, webhooks, ...
- Available methods:
 - https://wxc-sdk.readthedocs.io/en/1.26.0/user/method_ref.html
- Easy token management

The screenshot shows the ReadTheDocs documentation for the `wxc_sdk` package. The left 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 is titled "wxc_sdk package" and describes it as a "Simple SDK for Webex APIs with focus on Webex Calling specific endpoints". It includes a code block for the `WebexSimpleApi` class, which takes parameters for tokens (a string or `Tokens` instance) and concurrent requests (an integer). Below the class definition, there are sections for `attachment_actions`, `cdr`, `devices`, and `events`, each linking to their respective API documentation.

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

Tokens

Tokens

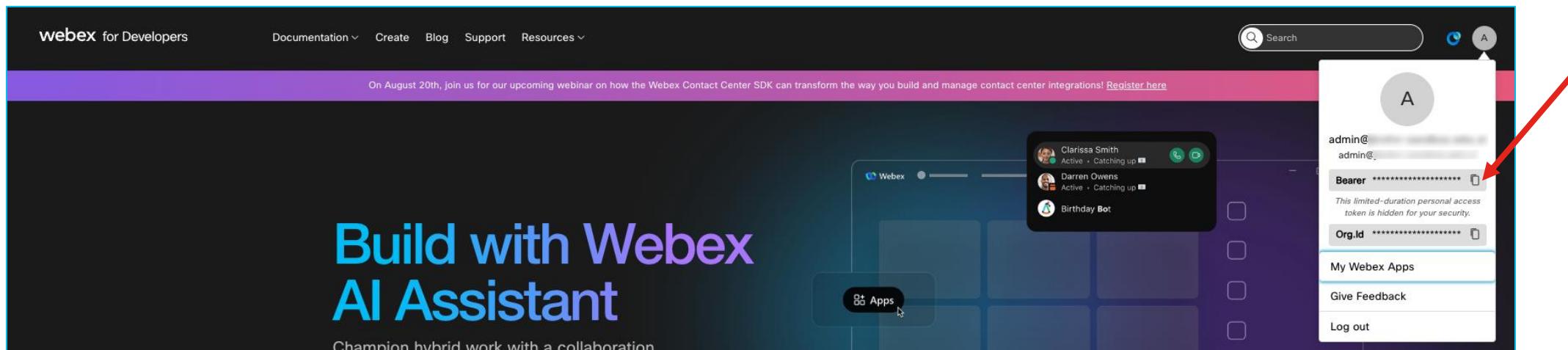
Why and How?

- 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
```

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

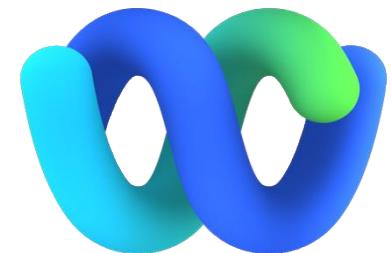
- Act on behalf of a Webex user
 - Access equivalent to a real Webex user (limited by authorized scopes)
- Invoke Webex APIs on behalf of user
- Requires authorization of integration by user
 - OAuth Grant Flow to authenticate user and ask for authorization
 - User approves authorization levels (scopes) 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 YML 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: ZGV1Y2***  
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_yml(yml_path=yml_path)  
    return tokens
```

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

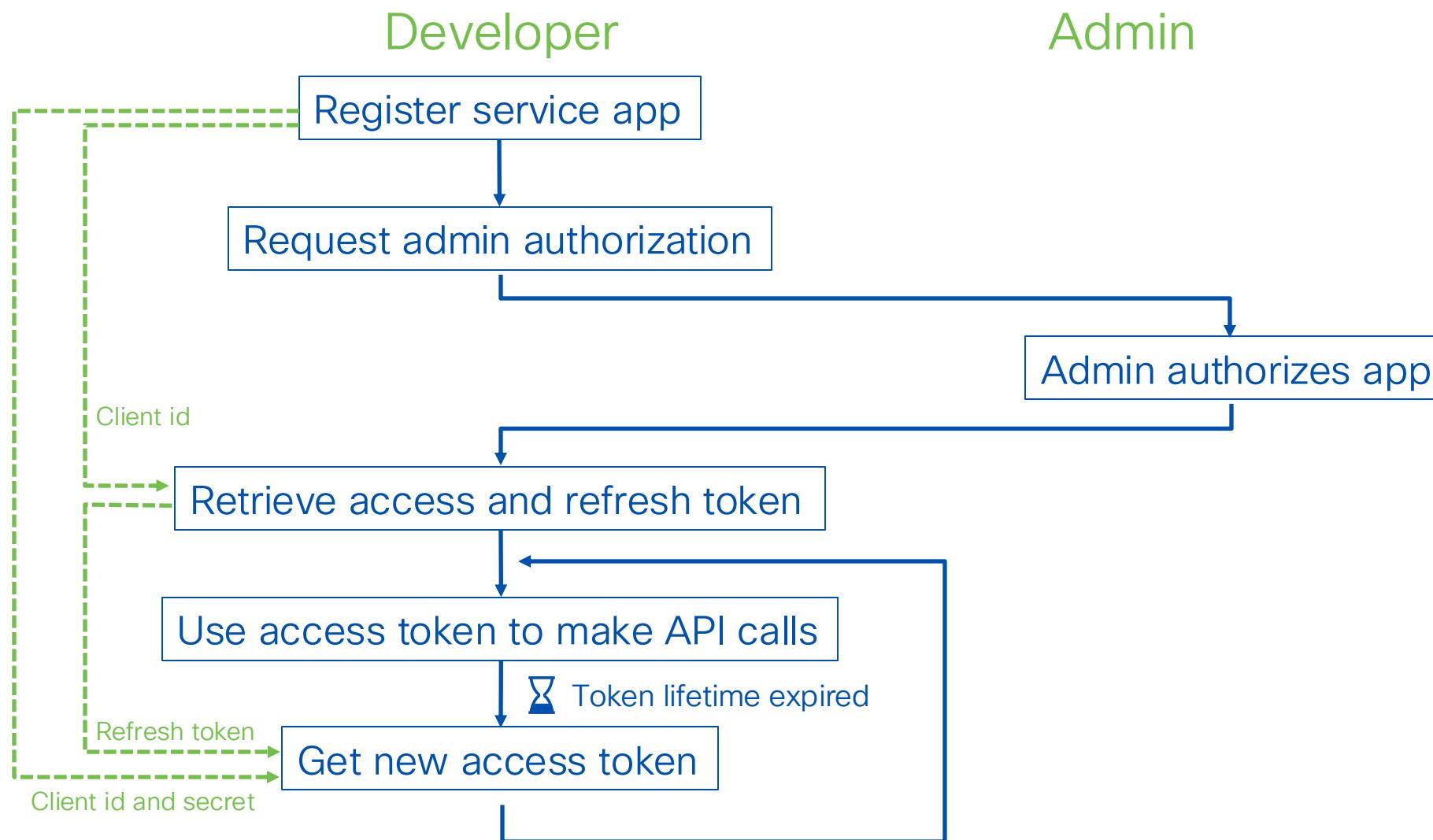
https://github.com/jeokrohn/BRKCOL-3015/blob/main/list_locations_sdk_int_tokens.py

Service Apps

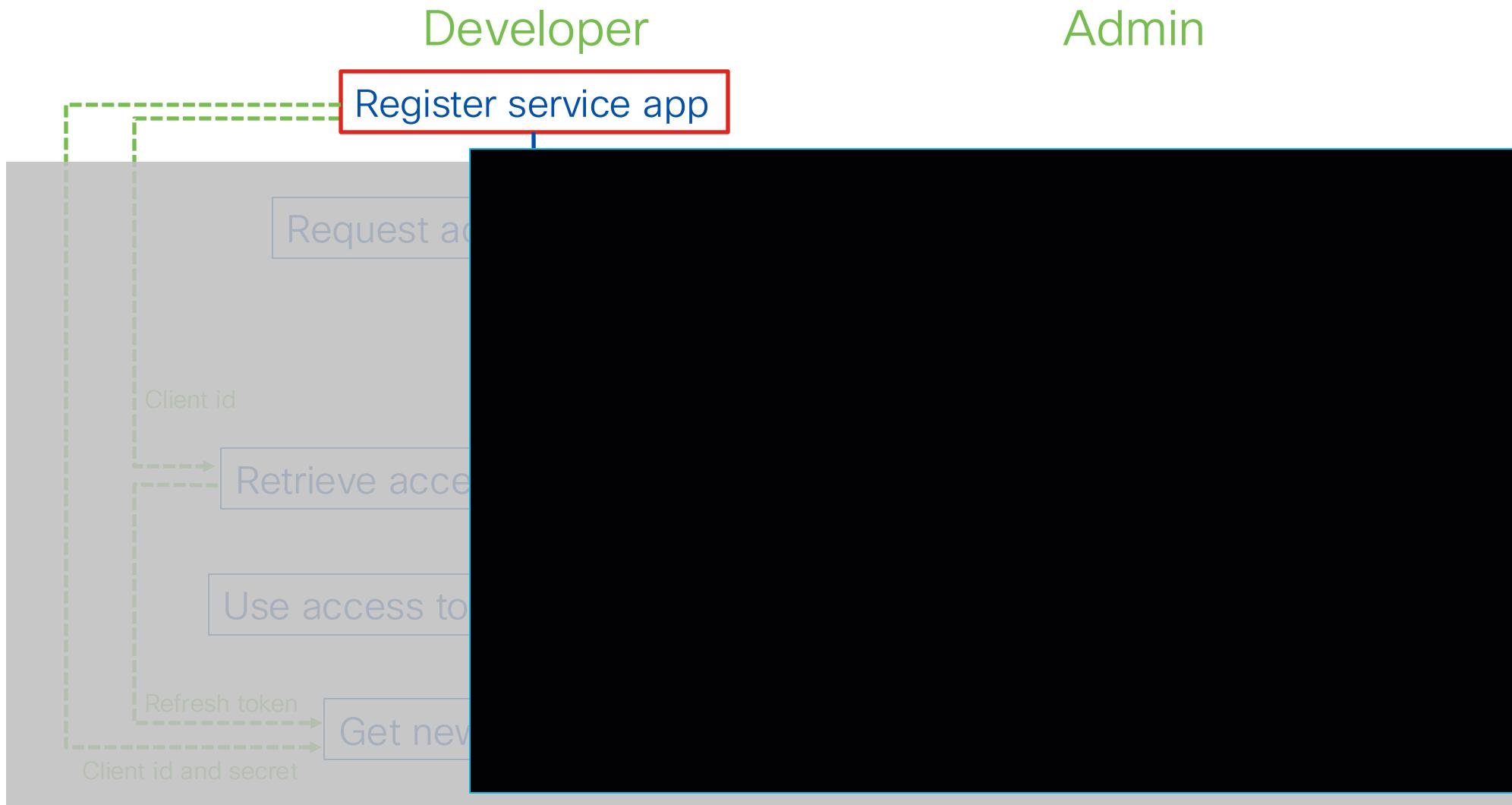
- Machine account
- Request admin permission independent of user account
- Use cases
 - Provisioning
 - Reporting
 - Scheduling systems
 - ...
- Similar to integrations .. but no user specific authorization flow

<https://developer.webex.com/admin/docs/service-apps>

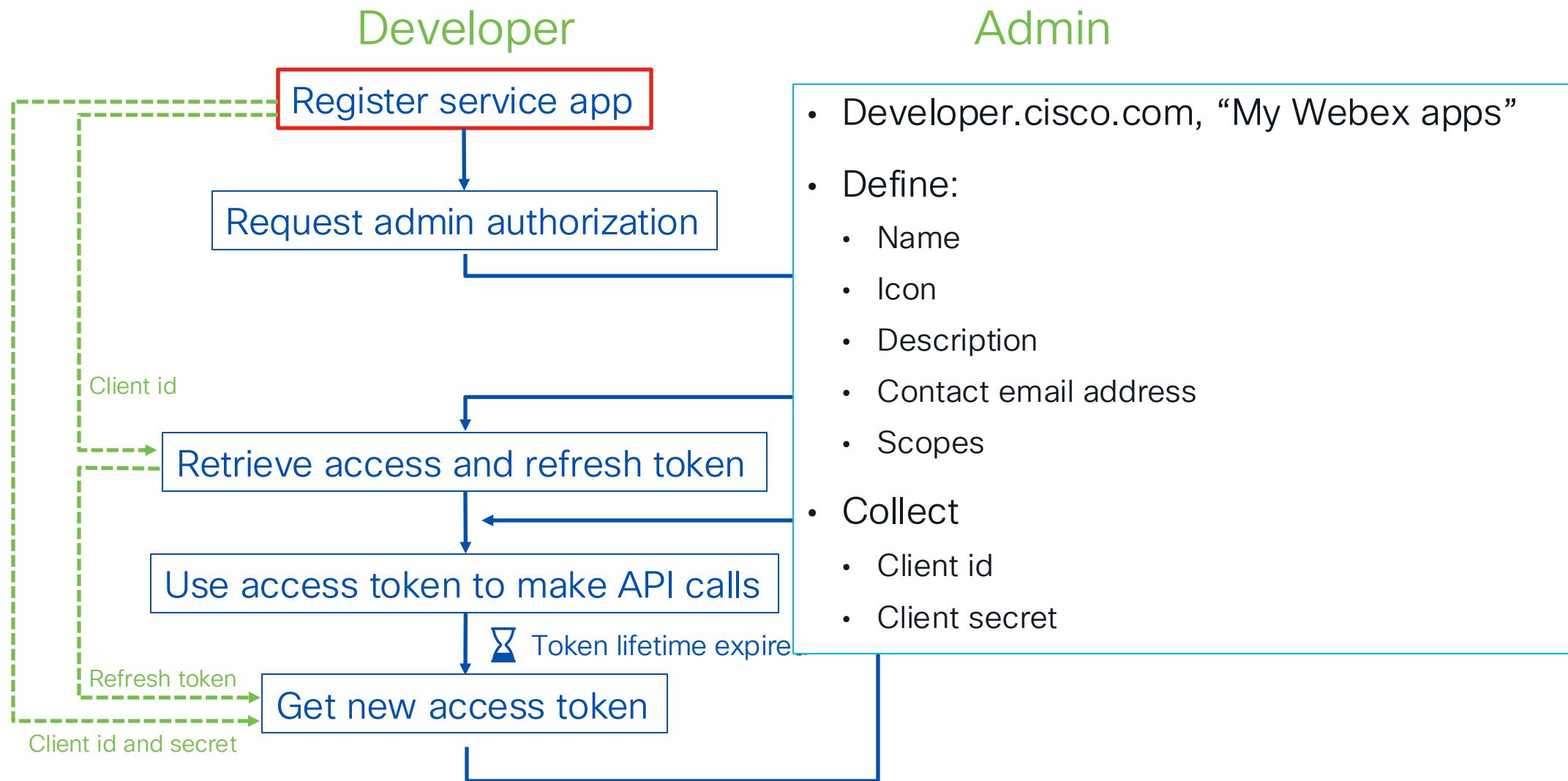
Using Service App Tokens



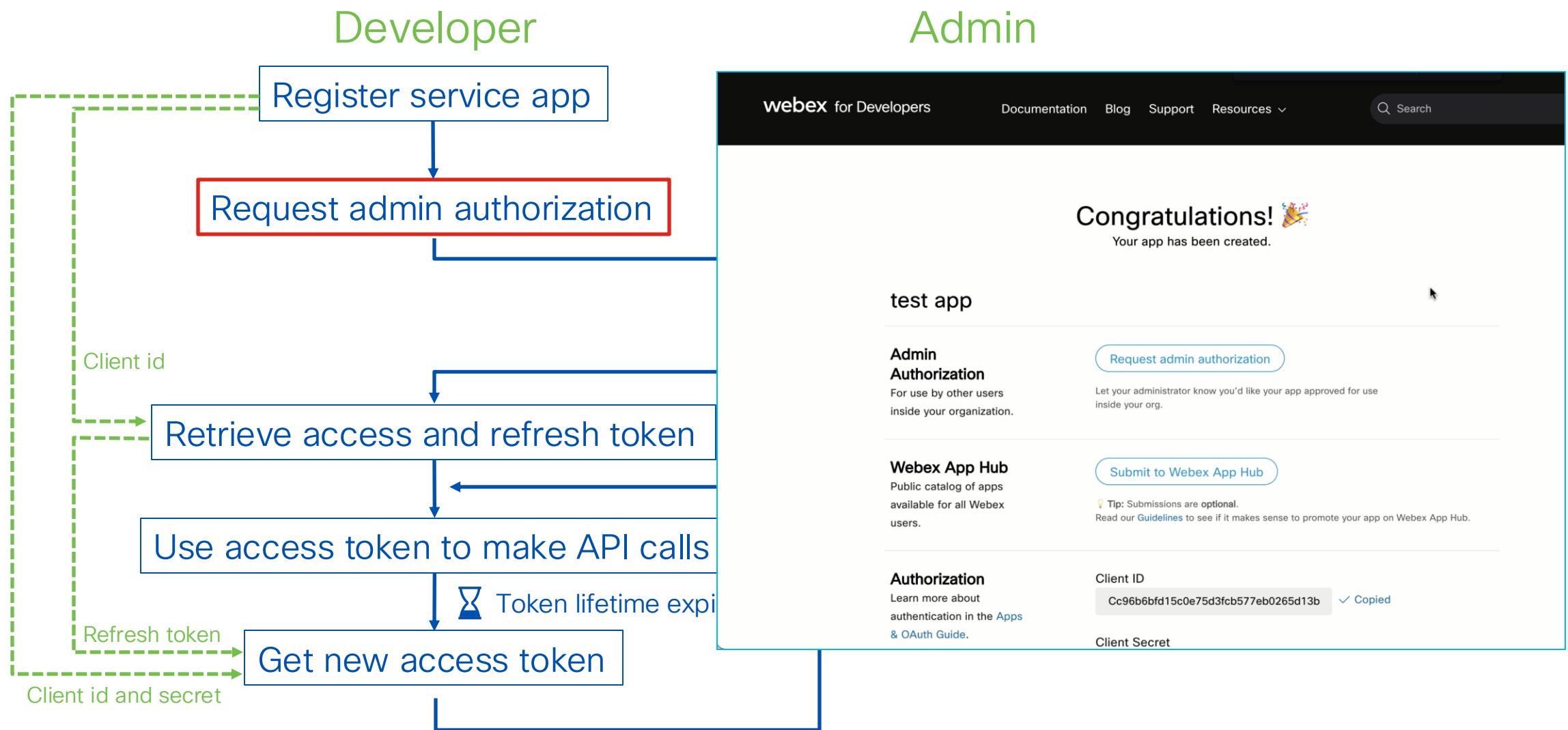
Using Service App Tokens



Using Service App Tokens



Using Service App Tokens



Using Service App Tokens

Developer

Admin

The screenshot shows the webex Control Hub interface. On the left, a sidebar lists navigation options like Overview, Getting Started Guide, Alerts center, MONITORING (Analytics, Troubleshooting, Reports), MANAGEMENT (Customers, Users, Groups, Locations, Workspaces, Devices, Apps), and a specific account section for jkrohn-sandbox.wbx.ai. The main area is titled 'Overview' and contains several cards:

- Getting Started Guide:** Shows 37% completion (3 of 8 tasks completed) and a link to 'View the Getting Started Guide and the recommended tasks'.
- Updates:** Encourages updating services to the new Webex experience, with a link to 'Update Webex Meetings to the ne...'. It also has a 'Learn more' button.
- New offers:** Allows everyone in the organization to host a Webex Meeting with a Basic Meetings license, with a 'Learn more' button.
- Onboarding:** Shows 33 total users, indicating no CSV upload within 180 days. It includes a pie chart showing user status: Active (100%), Inactive (0%), Not verified (0%), and Verified (0%). Buttons for 'Review' and 'Enable directory sync' are present.
- Webex services:** Shows ALL ONLINE status for Webex, Calling, and Meetings.
- Devices:** Shows 8 total devices, with breakdowns for Online (3), Online with issues (0), Offline (5), and Expired (0).

A green dashed arrow at the bottom points from the 'Client id and secret' text to the 'Get new access token' button. The 'Get new access token' button is highlighted with a blue border.

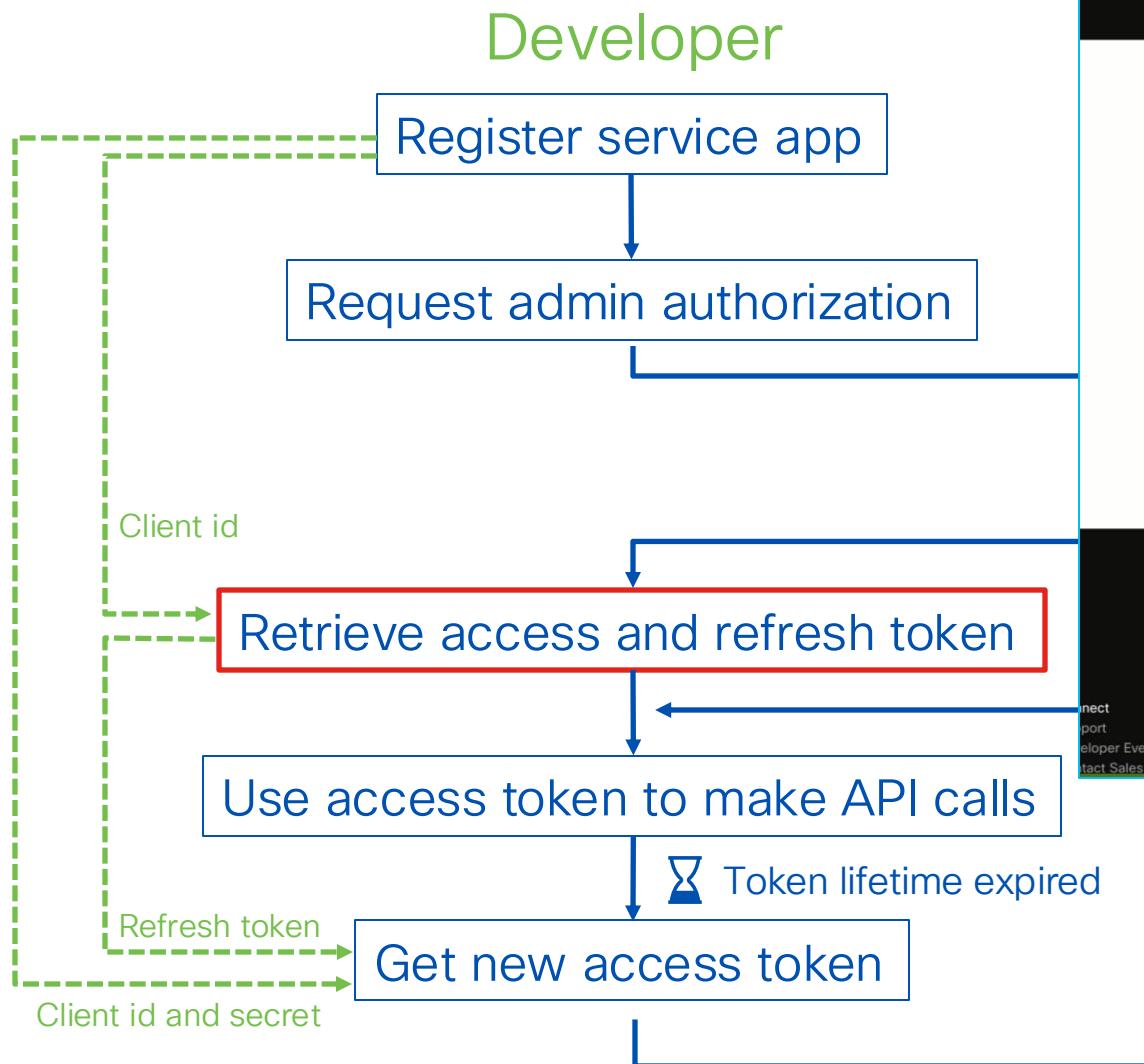
Client id and secret

Get new access token

Admin authorizes app

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

Using Service App Tokens



webex for Developers Documentation Blog Support Resources ▾ A Search

Create a New App

My Apps

- test app (Created May 12, 2023) Service App
- WxC SDK development integration (Created November 4, 2022) Integration
- Test Guest Issuer (Created May 18, 2022) Guest Issuer

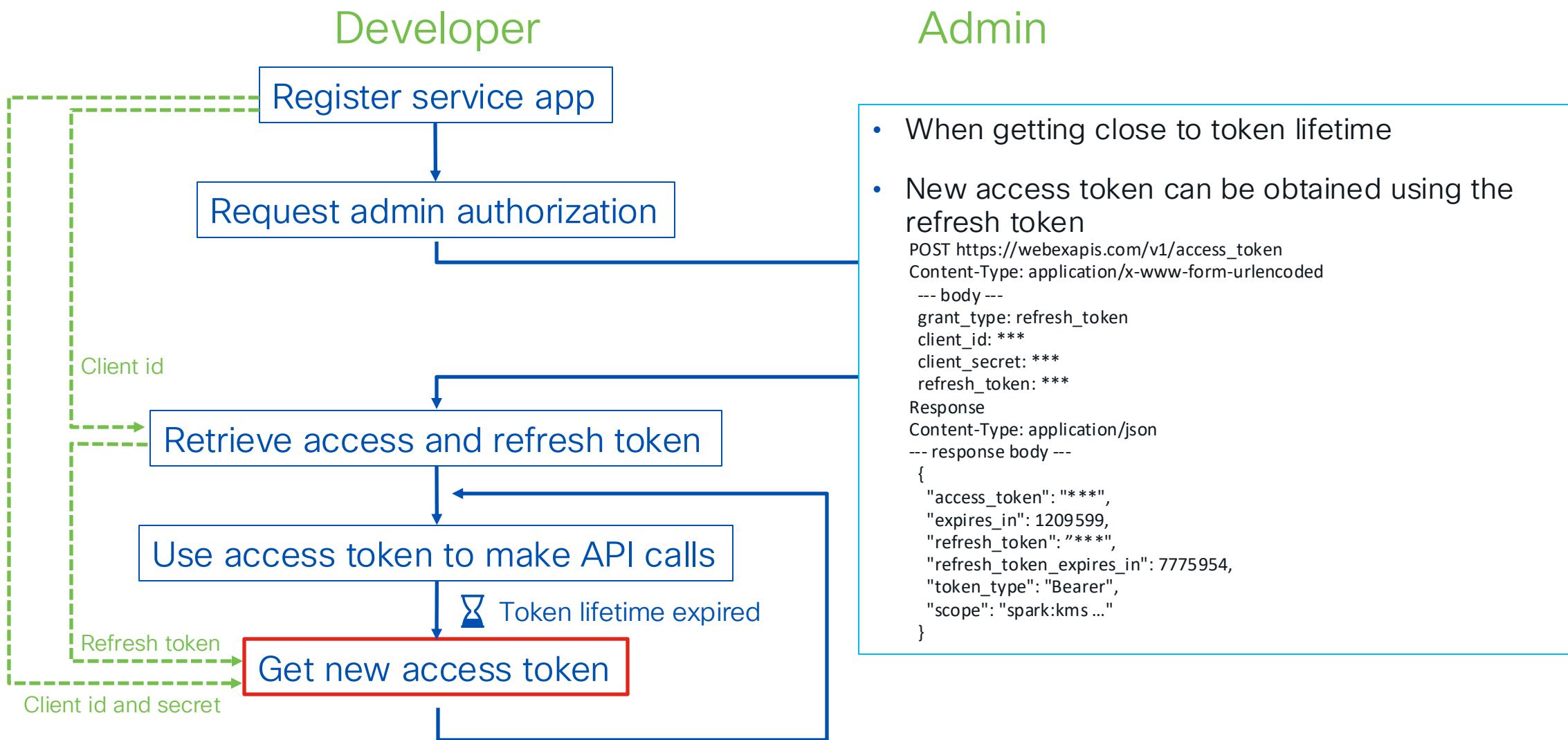
Handy Links: Webex Ambassadors, Webex App Hub

Resources: Open Source Bot Starter Kits, Download Webex, DevNet Learning Labs

Terms of Service, Privacy Policy, Cookie Policy, Trademarks

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

Using Service App Tokens



Using Service App with wxc_sdk

Read secrets from environment

```
SERVICE_APP_CLIENT_ID=Cc96b6bfd15c0e75d3f  
SERVICE_APP_CLIENT_SECRET=d602c66bd8cd32a  
SERVICE_APP_REFRESH_TOKEN=ZGI1ZTE5NDEtNzY
```

Create access token and persist tokens in YML file

```
access_token: NTU5NTYxNjItZjU  
expires_at: 2024-02-06 15:17:29.083739+00:00  
expires_in: 1209599  
refresh_token: ZGI1ZTE5NDEtNzY  
refresh_token_expires_at: 2024-04-22 15:17:29.083739+00:00  
refresh_token_expires_in: 7775999  
scope: spark-admin:workspaces_write Identity:one_time_password identity:placeonetimepassword_create  
    spark:people_read identity:tokens_write spark-admin:workspace_locations_read spark-admin:workspaces_read  
    spark:devices_write spark:devices_read spark:kms spark-admin:devices_read spark-admin:workspace_locations_write  
    identity:tokens_read spark-admin:licenses_read spark-admin:telephony_config_read  
    spark-admin:telephony_config_write spark-admin:devices_write spark-admin:people_read
```

Use tokens to call endpoints

https://github.com/jeokrohn/wxc_sdk/blob/master/examples/service_app.py

Using Service App with wxc_sdk

Read secrets from environment

```
SERVICE_APP_CLIENT_ID=Cc96b6bfd15c0e75d3f  
SERVICE_APP_CLIENT_SECRET=d602c66bd8cd32a  
SERVICE_APP_REFRESH_TOKEN=ZGI1ZTE5NDEtNzY
```

Create access token and persist

```
access_token: NTU5NTYxNjItZjU  
expires_at: 2024-02-06 15:17:29.083739+00:00  
expires_in: 1209599  
refresh_token: ZGI1ZTE5NDEtNzY  
refresh_token_expires_at: 2024-04-22 15:17:29.083739+00:00  
refresh_token_expires_in: 7775999  
scope: spark-admin:workspaces_write Identity:one_time_password identity:placeonet  
    spark:people_read identity:tokens_write spark-admin:workspace_locations_read spark  
    spark:devices_write spark:devices_read spark:kms spark-admin:devices_read spark  
    identity:tokens_read spark-admin:licenses_read spark-admin:telephony_config_rea  
    spark-admin:telephony_config_write spark-admin:devices_write spark-admin:people
```

Use tokens to call endpoints

```
def get_access_token() -> Tokens:  
    """  
    Get a new access token using refresh token, service app client id, service app client secret  
    """  
    tokens = Tokens(refresh_token=getenv('SERVICE_APP_REFRESH_TOKEN'))  
    integration = Integration(client_id=getenv('SERVICE_APP_CLIENT_ID'),  
                               client_secret=getenv('SERVICE_APP_CLIENT_SECRET'),  
                               scopes=[], redirect_url=None)  
    integration.refresh(tokens=tokens)  
    write_tokens_to_file(tokens)  
    return tokens  
  
def get_tokens() -> Optional[Tokens]:  
    """  
    Get tokens from cache or create new access token using service app credentials  
    """  
    # try to read from file  
    tokens = read_tokens_from_file()  
    # ... or create new access token using refresh token  
    if tokens is None:  
        tokens = get_access_token()  
    if tokens.remaining < 24 * 60 * 60:  
        tokens = get_access_token()  
    return tokens
```

https://github.com/jeokrohn/wxc_sdk/blob/master/examples/service_app.py

Using Service App with wxc_sdk

Read secrets from environment

```
SERVICE_APP_CLIENT_ID=Cc96b6bfd15c0e75d3f  
SERVICE_APP_CLIENT_SECRET=d602c66bd8cd32a  
SERVICE_APP_REFRESH_TOKEN=ZGI1ZTE5NDEtNzY
```

Create access token and persist tokens in YML file

```
access_token: NTU5NTYxNjItzjU  
expires_at: 2024-02-06 15:17:29.083739+00:00  
expires_in: 1209599  
refresh_token: ZGI1ZTE5NDEtNzY  
refresh_token_expires_at: 2024-04-22 15:17:29.083739+00:00  
refresh_token_expires_in: 7775999  
scope: spark-admin:workspaces_write Identity:one_time_password identity:placeonetimepassword_cr  
    spark:people_read identity:tokens_write spark-admin:workspace_locations_read spark-admin:work  
    spark:devices_write spark:devices_read spark:kms spark-admin:devices_read spark-admin:worksp  
    identity:tokens_read spark-admin:licenses_read spark-admin:telephony_config_read  
    spark-admin:telephony_config_write spark-admin:devices_write spark-admin:people_read
```

```
# get tokens and dump to console  
tokens = get_tokens()  
print(dumps.loads(tokens.json(), indent=2))  
print()  
print('scopes:')  
print('\n'.join(f' * {s}' for s in sorted(tokens.scope.split())))  
  
# use tokens to access APIs  
api = WebexSimpleApi(tokens=tokens)  
  
users = list(api.people.list())  
print(f'{len(users)} users')  
  
queues = list(api.telephony.callqueue.list())  
print(f'{len(queues)} call queues')
```

Use tokens to call endpoints

https://github.com/jeokrohn/wxc_sdk/blob/master/examples/service_app.py

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

```
(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
- room_devices.py: remove calling entitlements from selected Workspaces
- ...

Example: User Web Portal

User Web Portal

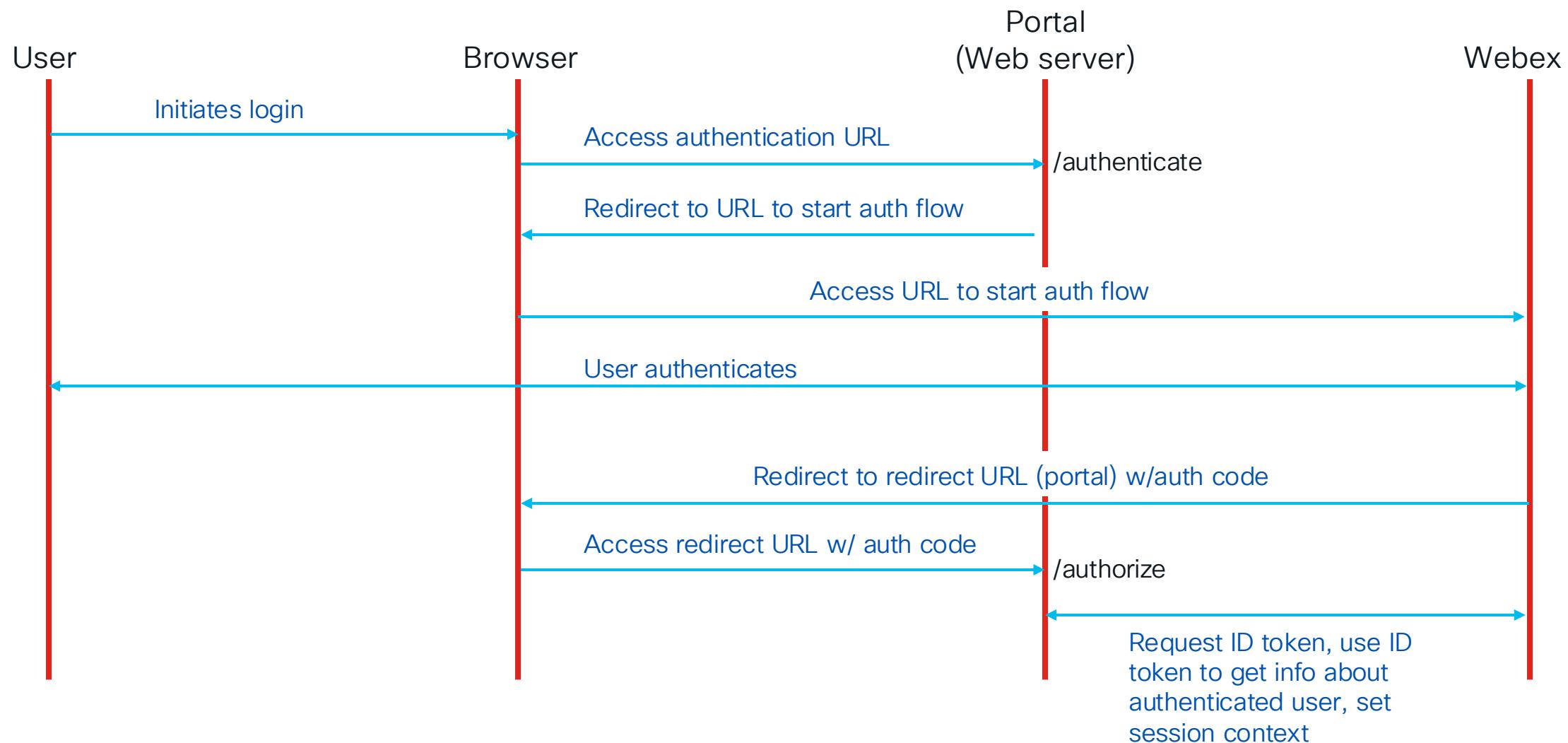
- Only a limited set of configuration options is available for users on settings.webex.com
- User portal can expose additional options (which usually require admin privileges)
- Perfect example for using service app tokens
 - User does not have the required privileges → integration token granted by user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

Concepts

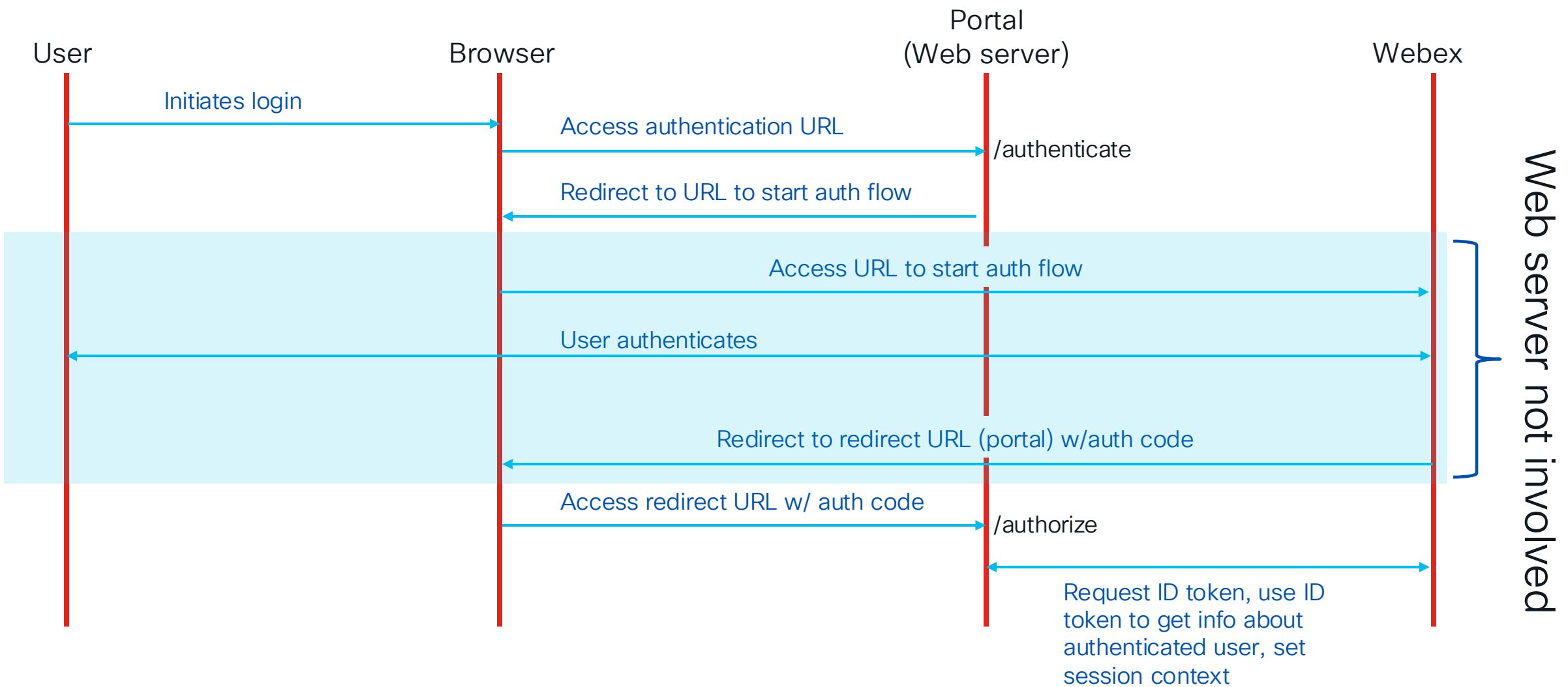
- Web based portal
- Authenticate users using “Login with Webex”
- For provisioning operations portal uses service app tokens
 - No tokens exposed in browser
- Browser only interacts with portal endpoints
 - Includes Javascript Ajax access
 - Portal acts as proxy between browser and Webex APIs
- APIs implemented using flask-restx → Swagger UI for free

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

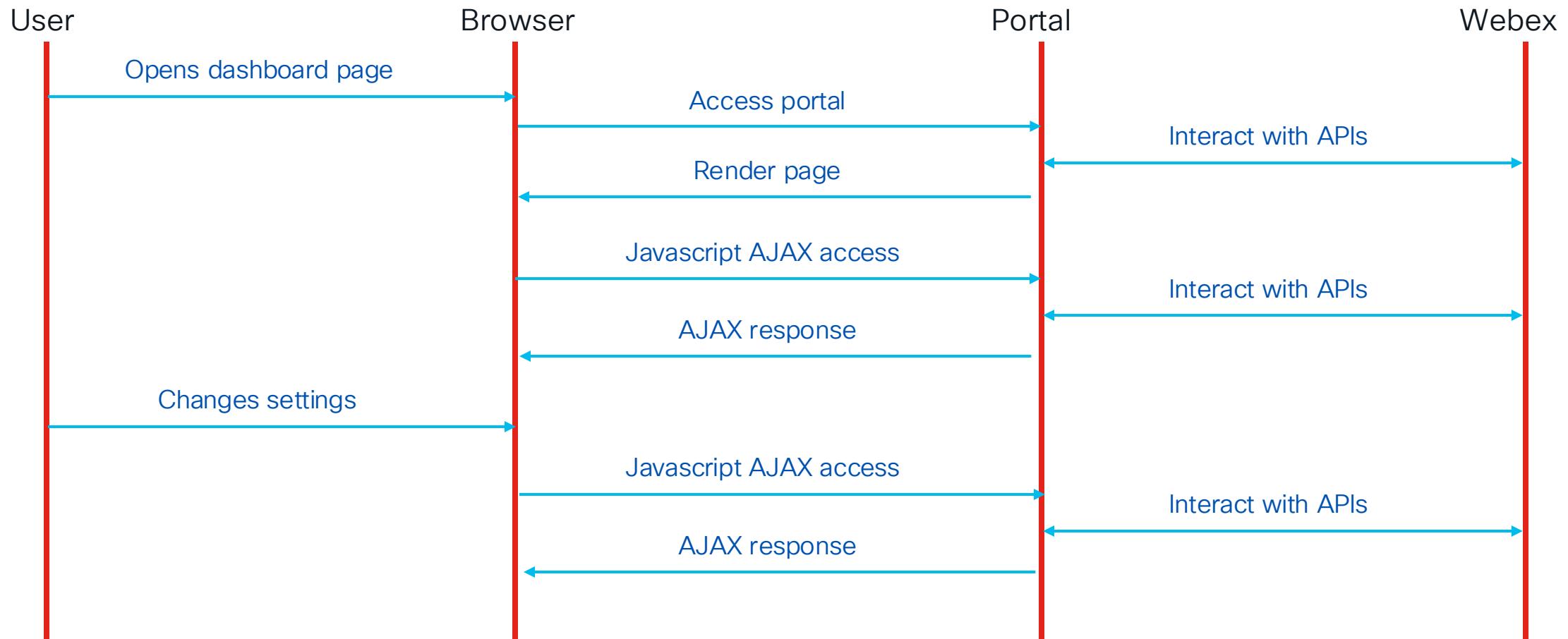
Login with Webex



Login with Webex



Portal Transactions



Rendering Data

index.html

```
<!-- phones -->
<div class="row">
  <div class="col-12 col-lg-7 mb-4" id="phoneCard">
    <div class="card shadow mb-4 h-100">
      <div class="card-header py-3">Phones</div>
      <div class="card-body">
        <div class="table-responsive">
          <table class="table table-striped table-bordered dt-responsive nowrap" id="userPhones">
            <!-- the table is empty and will be set in $(document).ready -->
          </table>
        </div>
        <div class="row">
          <div class="text-xs" id="status">
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

index.html

```
{% block scripts %}
<script type="text/javascript">
$(document).ready(function () {
  ...
  // set up DataTable for phones
  $('#userPhones').DataTable({
    searching: false,
    data: [],
    columns: [
      {title: "Model", data: "model"},
      {title: "Mac", data: "mac"},
      {title: "Status", data: "status"}
    ]
  });
  update_user_phones()
})
```

Rendering Data

user_phones.js

```
function get_user_phones() {
  return $.ajax({type: 'GET', url: '/userphones'})
}

function update_user_phones() {
  const card = document.getElementById("phoneCard")
  const status = card.querySelector("#status")
  status.textContent = "...getting phone information"

  //get phones from server and update table
  get_user_phones().then(function (data) {
    console.log(data)

    if (data["success"] == false) {
      status.textContent = "Error: " + data["message"]
    } else {
      status.textContent = ""
      $('#userPhones').DataTable().clear()
      $('#userPhones').DataTable().rows.add(data['rows'])
      $('#userPhones').DataTable().columns.adjust().draw()
    }
  })
}
```

index.html

```
{% block scripts %}
<script type="text/javascript">
$(document).ready(function () {

  ...
  //set up DataTable for phones
  $('#userPhones').DataTable({
    searching: false,
    data: [],
    columns: [
      {title: "Model", data: "model"},
      {title: "Mac", data: "mac"},
      {title: "Status", data: "status"}
    ]
  });
  update_user_phones()
})
```

es">

Rendering Data

user_phones.js

```
function get_user_phones() {
    return $.ajax({type: 'GET', url: 'api/userphones'})
}

function update_user_phones() {
    const card = document.getElementById("phoneCard")
    const status = card.querySelector("#status")
    status.textContent = "...getting phone information"

    //get phones from server and update table
    get_user_phones().then(function (data) {
        console.log(data)

        if (data["success"] == false) {
            status.textContent = "Error: " + data["message"]
        } else {
            status.textContent = ""
            $('#userPhones').DataTable().clear()
            $('#userPhones').DataTable().rows.add(data['rows'])
            $('#userPhones').DataTable().columns.adjust().draw()
        }
    })
}
```

api/__init__.py

```
@api.route('/userphones')
class UserPhones(Resource):
    """
    Get phones of current user
    """

    @staticmethod
    @assert_user
    def get():
        """
        Get phones of current user.
        Returns a JSON object with:
        * success: True if the operation was successful
        * rows: List of phone data rows. Each row in the table will contain:
            * product: Product name of the phone
            * mac: MAC address of the phone in colon-separated format
            * connection_status: Connection status of the phone
        """
        def mac_with_colons(mac: str) -> str:
            octets = (mac[i:i + 2] for i in range(0, len(mac), 2))
            return ":".join(octets)

        user = session.get('user')
        user: Person
        ca: AppWithTokens = current_app
        path = urlparse(request.url).path
        try:
            log.debug(f"{path}: getting user phones")
            devices = list(ca.api.devices.list(person_id=user.person_id))
        except RestError as e:
            log.error(f"{path}: getting user phones failed: {e}")
            return {'success': False,
                    'message': f'{e}'}
        log.debug(f"{path}: returning device data")
        return {'success': True,
                'rows': [{model: device.product,
                          'mac': mac_with_colons(device.mac),
                          'status': device.connection_status}
                         for device in devices
                         if device.product_type == ProductType.phone]}
```

Demo Web App

Web App

- Run app in docker container
 - docker-compose up -d --build -build to re-build the image
 - docker-compose logs -f show logs
- Run `web_app/app.py`
- Portal URL: <http://localhost:5010>
- Swagger UI: <http://localhost:5010/api/docs>
 - Review/test the API endpoints
 - User must be authenticated

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
- Call control bot: <https://wxc-callcontrol.readthedocs.io/en/latest/>
- Other repositories: <https://github.com/jeokrohn?tab=repositories>

Share your
experience!

Scan the QR code for the
post-session survey



The background features a dynamic arrangement of overlapping circles in various colors, including shades of blue, green, purple, and orange, set against a black background.

webexone²⁵