

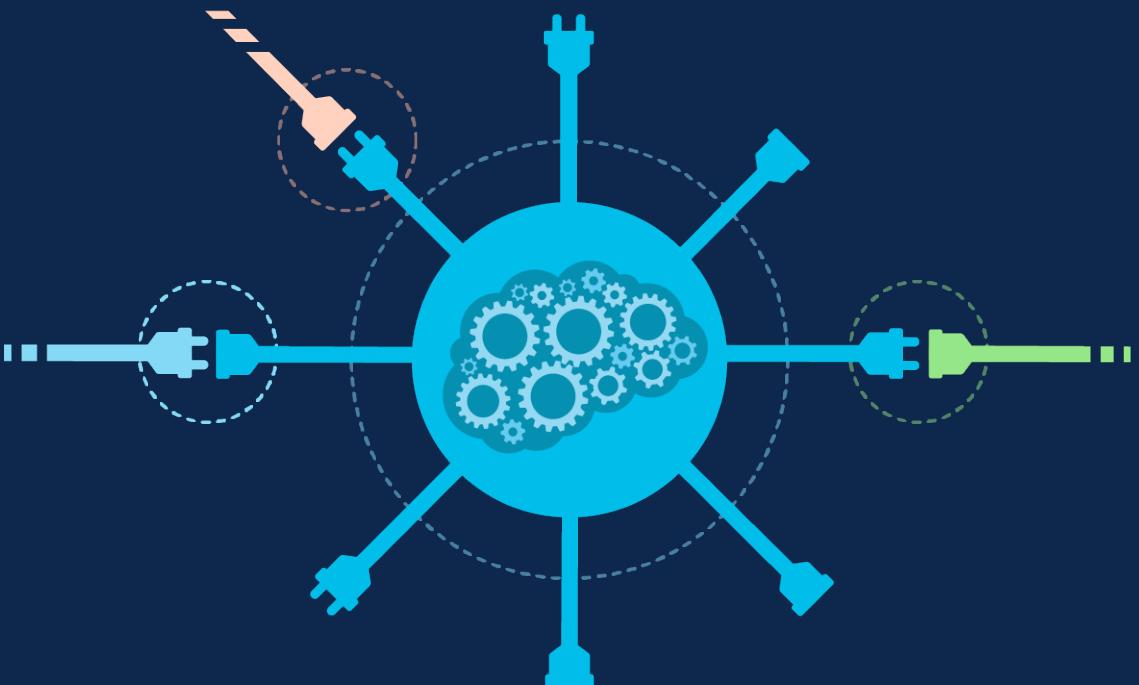


# Webex Contact Center APIs

Programmatic Contact Center & Developer Portal

Shrishail  
Solution Success Engineer – CCBU  
April 2023

# Agenda



- API Fundamentals
  - API terminology
- All New Webex CC APIs
  - Reimagining the Contact Center where anything is possible
- Developer Portal, New features
  - Overview of developer portal, new features
- OAuth2
  - Understanding the access mechanism
- Use Cases
  - What is possible using these API's, Live demo.
- Road Map, Sample & Support
  - What's next and where to get help



# 00

# API Fundamentals

# Lightning Round – Acronyms!

Demystifying Terms

Representational State Transfer

## RESTful

Simple ask and answer, like asking for the blue pen on the table.

Going to a website is a “GET” rest call.

# Lightning Round – Acronyms!

Demystifying Terms

HTTPS POST Event to your App

## Webhooks

Asking someone to let you know when Bob arrives. When Bob arrives, you get informed so that you can greet Bob.

# Lightning Round – Acronyms!

Demystifying Terms

Full Duplex HTTPS 1.1 Upgrade

## Websockets

Is like either listening to the radio (one way) or talking on the phone (two way), it is a dedicated communication channel that is used to stream data.

# Lightning Round – Acronyms!

Demystifying Terms

HTTP 429 - Too Many Requests

## Rate Limiting

Like a circuit breaker, it stops the flow of requests to protect the system.

# Lightning Round – Acronyms!

Demystifying Terms

## Client-Server

Think Customer << >> Provider  
Client Requests  
Server Responds

# Lightning Round – Acronyms!

Demystifying Terms

JavaScript Object Notation

## JSON

Is a format for representing the data.  
Key-value pairs, Easy to read/extract data.

# Lightning Round – Acronyms!

Demystifying Terms

Graph Query Language

## GraphQL

Query language used to extract data.  
Request specific data – Get only that data.  
Sent as part of HTTP POST request.

# Lightning Round – Acronyms!

Demystifying Terms

Open Authorization 2

## OAuth2

Uses Tokens instead of passwords.

Like Airbnb - owner uses their “key” to create a door code.

You use that to access the property.

When the code expires, you no longer have access, and  
you never touch the owner’s key!

# API = Contract = Specification

Functionality. Backward Compatible. Constraints detailed.

Documentation

Specification

Authentication  
Authorization  
Accounting

Community  
& Support

Schema

Terms of Service

Rate Limits

Examples

Self—Service development.

Build innovative solutions.

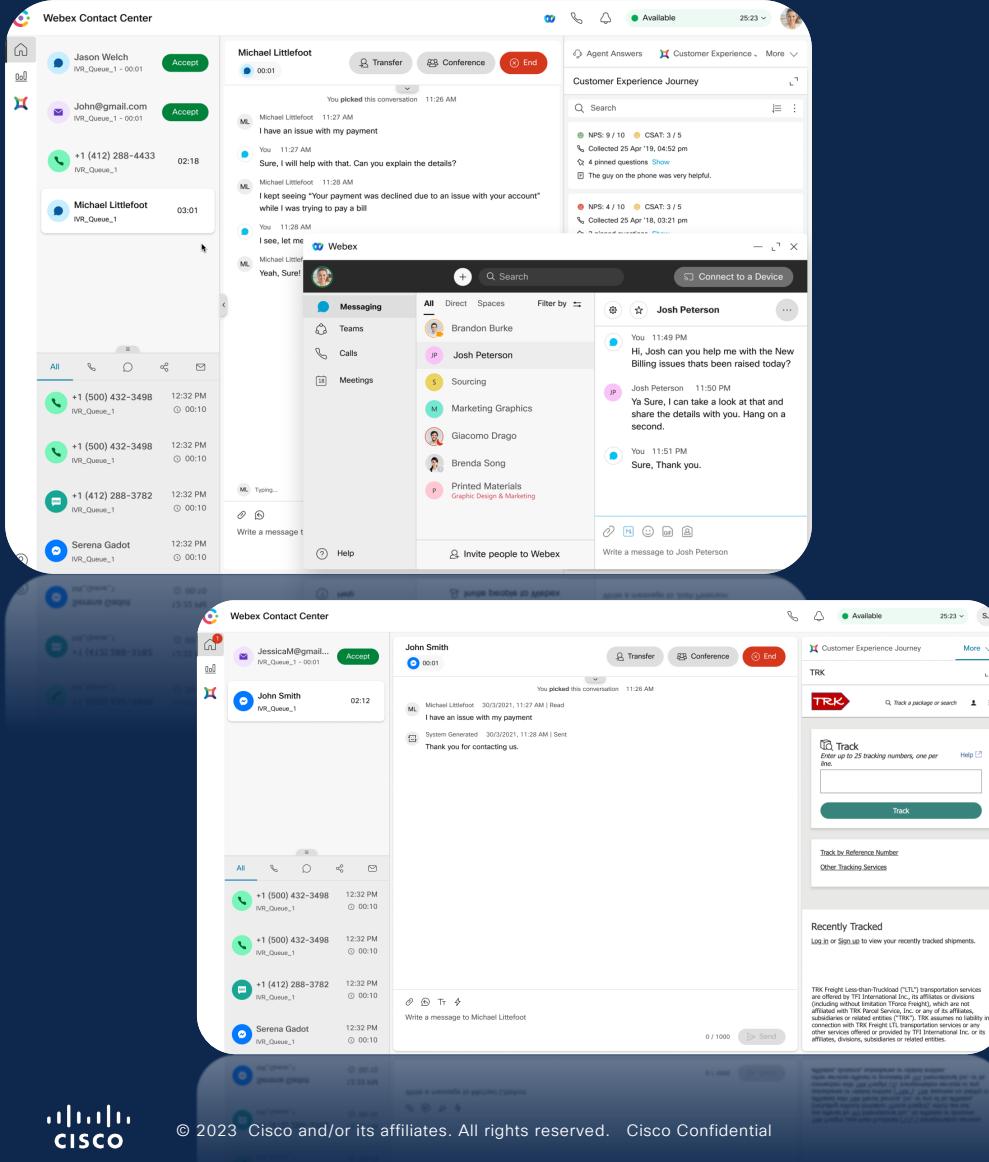
Application extensions that Differentiate.



01

# Webex Contact Center APIs

# New API Platform - Flexible & Extensible



- **Customize** every step of the customer and agent journey



- **Integrate** into existing business applications and tools



- **Build** new features that extend the existing out-of-box features



- **Track** and improve KPIs

# APIs Available today

Configuration

Reporting

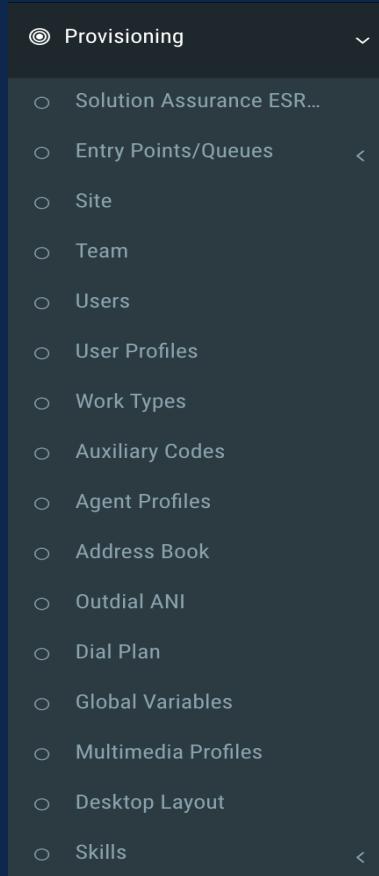
Agent

Call Control

Journey

Webhooks

# Configuration APIs



Address Book  
Agent Profile  
Audio files  
Auxiliary codes  
Queue  
Desktop layout  
Dial Number  
Dial plan  
Entry point  
Users

Global Variable  
Multimedia Profile  
Out dial ANI  
Site  
Skill  
Skill Profile  
Team  
User Profile  
Work Types

(Automated provisioning, Create/Read/Delete/Update operations)

# Reporting API

- GraphQL/Search.
- Raw data access to analyzer.
- Get only what you want.
- ANI, Agent.

The screenshot shows a Swagger UI interface for a 'Search Tasks' endpoint. At the top, there are tabs for 'Sample Code' and 'Try Out'. Below the tabs, a 'POST /search' button is visible. The main content area has a title 'Search Tasks' and a sub-section titled 'POST /search'. A detailed description follows:

Search supports the following queries - task, taskDetails and agentSession. For task, the max page size is 300 whereas for taskDetails and agentSession, the max page size is 100. Apart from the optional filter and aggregation input parameters for the query, each query accepts from and to parameters (datetime in epoch format), as mandatory parameters. The from parameter is the start time for the query (in epoch milliseconds) and it cannot be older than 36 months from the current time. The to parameter is the end time for the query (in epoch milliseconds) and it cannot be greater than the current time. The duration between to and from must not be more than 365 days. In a graphql query, the aggregationInterval parameter is only applicable if the query has aggregation input parameters. Refer 'IntervalData' description in GraphQL Schema to check allowed interval values that user can provide for specific duration. When Using Grouping along with aggregation, only field with scalar types are allowed for grouping. Also, in case of grouping by nested fields parent of selected field should not be an Array. For this API, response compression using gzip can be enabled by including 'Accept-Encoding' header in the request with its value as 'gzip'. The response will be compressed only if its size exceeds 1 MB. If the header is not present in the request or if gzip is not listed as one of the encodings in the header's value (comma separated encodings), then API response will not be compressed and this can impact the latency as observed from clients.

**Sample Code**

```
curl --request POST \
--url 'https://api.wxcc-us1.cisco.com/search?orgId=97cdbf45-ebe2-4683-8f2d-1a2a2e0a23c1'
--header 'Accept: application/json' \
--header 'Authorization: Bearer YOUR_TOKEN' \
--header 'Content-Type: application/json' \
--data '{"query": ""}'
```

**Response**

```
1 {
2   "data": {
3     "task": {
4       "tasks": [
5         {
6           "id": "fb53f6d1-5535-4ac8-b081-53834e17d6f5",
7           "channelType": "telephony",
8           "createdTime": 1629450000000,
9           "endedTime": 1630380960406,
10          "captureRequested": true,
11          "isActive": false,
12          "status": "ended",
```

# Webhooks

Subscribe to webhook in order to receive events

## Agents

The Agent resource encapsulates the entities which handle the Tasks through interactions with customers. An Agent can be either a human or bot, and can be either the entity directly handling the Task, or an assisting entity i.e. a supervisor. Agents are summarized by the groups they belong to (organization and team), details for the Tasks they handle, and their overall availability. The difference in duration between the from and to dates cannot be more than 1 day. Requires one of the following scopes `cjp:config`, `cjp:config_read`.

REST APIs (7) Webhooks (3)

**Agent Login**

**EVENT** `agent:login`

An event indicating that an Agent has logged in.

**Agent Logout**

**EVENT** `agent:logout`

An event indicating that an Agent has logged out.

**Agent State Change**

**EVENT** `agent:state_change`

An event indicating that an Agent's state has changed.

REST APIs (16) Webhooks (5)

**Task Connect**

**EVENT** `task:connect`

An event indicating that a task is routed to an agent, and the agent has yet to accept the work.

**Task Connected**

**EVENT** `task:connected`

An event indicating that a task has been successfully accepted by the Agent.

**Task Ended**

**EVENT** `task:ended`

An event indicating that a task is ended either by the Agent or by the Customer.

**Task New**

**EVENT** `task:new`

An event indicating that a task is formed.

**Task Parked**

**EVENT** `task:parked`

An event indicating that a task is queued because no agents are currently available to handle the task.

Captures > Capture Available Version 1

**Capture Available**

**EVENT** `capture:available`

An event indicating that the recording for a call is available after the call has ended.

- Events from server to client
- One way, Opposite to APIs
- Subscribe

# Agent APIs

## Login

**POST** /v1/agents/login

Allows the user to login to their Desktop. It does not allow a duplicate login and sends an error message over websocket, if an active session already exists. Requires 'cjp:user','id\_full\_admin','id\_READONLY\_admin','atlas-portal.partner.salesadmin','cjp.admin','cjp.supervisor','atlas-portal.partner.helpdesk','atlas-portal.partner.provision\_admin' scope for authorization. For a list of possible response messages, see the Call Control API Guide.

## Logout

**PUT** /v1/agents/logout

Allows the user to logout from their Desktop. This API needs to be called once the WSS session has been successfully established. Requires 'cjp:user','id\_full\_admin','id\_READONLY\_admin','atlas-portal.partner.salesadmin','cjp.admin','cjp.supervisor','atlas-portal.partner.helpdesk','atlas-portal.partner.provision\_admin' scope for authorization. For a list of possible response messages, see the Call Control API Guide.

## Reload

**POST** /v1/agents/reload

Allows the user to receive all the contact assigned to particular agent and state. Requires 'cjp:user','id\_full\_admin','id\_READONLY\_admin','atlas-portal.partner.salesadmin','cjp.admin','cjp.supervisor','atlas-portal.partner.helpdesk','atlas-portal.partner.provision\_admin' scope for authorization.

## State Change

**PUT** /v1/agents/session/state

Allows the user to toggle between the states (Available, Idle, Busy etc). Requires 'cjp:user','id\_full\_admin','id\_READONLY\_admin','atlas-portal.partner.salesadmin','cjp.admin','cjp.supervisor','atlas-portal.partner.helpdesk','atlas-portal.partner.provision\_admin' scope for authorization. For a list of possible response messages, see the Call Control API Guide.

- WebSocket.
- Subscribe to WebSocket to get URL <-> start listening.

# Call Control APIs

## Accept Task

**POST** /v1/tasks/{taskId}/accept

Access this Endpoint when the user has to accept either an Inbound or an outbound requests. The request can be a call, a chat or an email. Requires 'cjp:user' scope for authorization. For a list of possible response messages, see the Call Control API Guide.

## Consult Task

**POST** /v1/tasks/{taskId}/consult

Access this Endpoint when the user has to consult a call or a chat to another user. Requires 'cjp:user' scope for authorization. For a list of possible response messages, see the Call Control API Guide.

## Consult Accept Task

**POST** /v1/tasks/{taskId}/consult/accept

Access this Endpoint when the user has to accept a call to the consulting user. Requires 'cjp:user' scope for authorization. For a list of possible response messages, see the Call Control API Guide.

## Consult Conference Task

**POST** /v1/tasks/{taskId}/consult/conference

Access this Endpoint when the user has to initiate a conference with the consulting user. Requires 'cjp:user' scope for authorization. For a list of possible response messages, see the Call Control API Guide.

## Consult End Task

**POST** /v1/tasks/{taskId}/consult/end

Access this Endpoint when the user has to end a call with the consulting user. Requires 'cjp:user' scope for authorization. For a list of possible response messages, see the Call Control API Guide.

## Consult Transfer Task

**POST** /v1/tasks/{taskId}/consult/transfer

Access this Endpoint when the user has to transfer a call to the consulting user. Requires 'cjp:user' scope for authorization. For a list of possible response messages, see the Call Control API Guide.

## End Task

**POST** /v1/tasks/{taskId}/end

Access this Endpoint when the user has to end either an Inbound or an outbound requests. The request can be a call, a chat or an email. Requires 'cjp:user' scope for authorization. For a list of possible response messages, see the Call Control API Guide.



# Journey APIs

- Store/Retrieve customer data.
- Configure JDS Objects.
- Store journey events.
- Retrieve journey events (historic/Realtime).

## Journey

The Journey resource represents the complete sum of experiences that customers go through when interacting with the representatives. Some of the highlights are building profile view based on the templates configured and actions can be configured and triggered based on the events. Requires appropriate spark:people\_read, cjp:config\_read or cjp:config\_write scopes. Please note we use **Shared Access Signature (SAS)** based Access Control to our APIs. For more details on JDS setup and authentication process, please visit the [Getting Started](#) page.

### Publish Events Via POST

**POST** /events/v1/journey

API accepts events that describe what occurred, when, and by whom on every interaction across touch points and applications. Data Ingestion is based on Cloud Events specification for describing event data in a common way. API accepts data in the form of POST with support for Header based authorization. SAS Token Requirements:

- ss=ds
- sp=w

◦ ws=ds  
◦ ss=ss  
◦ sas=  
◦ token=  
◦ auth=  
◦ base64=  
◦ token=  
◦ auth=  
◦ base64=





# 02

# Developer Portal

[developer.webex-cx.com](https://developer.webex-cx.com)

# Global, single, unified URL, New features

The screenshot shows the webex Customer Experience developer portal. On the left, there's a sidebar with navigation links like Overview, Getting Started, Authentication, Common API Errors, Integrations, Introduction to APIs, Rate Limiting, API Reference, Address Book, Agent Profile, Agents, Auxiliary Codes, Captures, Dial Plan, Dialed Number To Entry Point, EntryPoints, Estimated Wait Time, and Journey. The main area has tabs for Sample Code and Try Out. Under Request, there's a Header section with a toggle for 'Use personal access token' (which is selected), and a Parameters section with 'orgid' set to '6ea87036-890d-4118-be56-c1f4d92a9e7a' and 'page' set to '4'. A 'Run' button is present. Below the request area is a 200 Response section with some JSON placeholder data. At the top right, there are Documentation, AB testing, and Sign Out buttons.

Try it out – on developer portal, defaults to the Region

The screenshot shows the developer portal with a curl command example for a GET request. The URL is /organization/{orgid}/entry-point, and the headers include Accept: application/json and Authorization: Bearer bearerToken. The curl command is:

```
1 curl --request GET \
2 --url 'https://api.wxcc-us1.cisco.com/organization/2f9eecc5-0472-4549-\
3 --header 'Accept: application/json' \
4 --header 'Authorization: Bearer bearerToken'
```

API Root URLs – Ensure you're using the right region's URI

Region	Developer Portal
US,ANZ, UK, EU	<a href="https://developer.webex-cx.com/">https://developer.webex-cx.com/</a>

Region	API Root URL
US	<a href="https://api.wxcc-us1.cisco.com">https://api.wxcc-us1.cisco.com</a>
ANZ	<a href="https://api.wxcc-anz1.cisco.com">https://api.wxcc-anz1.cisco.com</a>
UK	<a href="https://api.wxcc-eu1.cisco.com">https://api.wxcc-eu1.cisco.com</a>
EU (Frankfurt)	<a href="https://api.wxcc-eu2.cisco.com">https://api.wxcc-eu2.cisco.com</a>

# Change log, New features

New	Newly released API resources, endpoints, or properties
Warning	API behavior change, usually related to unintended usage of an endpoint or because of unexpected behavior
Deprecation	API resource, endpoint, or property is still available but is no longer recommended to use and won't be updated further.
Major	Change which may effect specific use cases (like a more specific breaking change)
Breaking	Change which will affect all developers, regardless of API usage

## API Changelog

This page lists recent and upcoming changes to the Webex Contact Center APIs.

### November 14 2022

- User Profile resource has been updated to improve the readability of the API usage. Improved documentation with better examples, to facilitate ease of use.

Currently both endpoints to List User Profile "/user-profile" and "/v2/user-profile" already exists. Beginning February 2023, endpoint /user-profile will be deprecated.

See [User Profile](#) page for more details.

### November 14 2022

#### Major

- Users resource has been updated.

The Users API endpoint to "Create A User Record" as well as "Delete A User Record" will be deprecated beginning February 2023 as this operation can be performed only from Cisco Webex Control Hub.

A new endpoint to List Users "/v2/users" is now available. The existing endpoint to "List Users" /users will be deprecated beginning February 2023.

The endpoint to "List Users Along With User Profile" & "List Users Along With User Profile by ID" will be deprecated beginning February 2023 as this operation can be performed using APIs of User Profile resource.

See [Users](#) page for more details.

see also for aged see

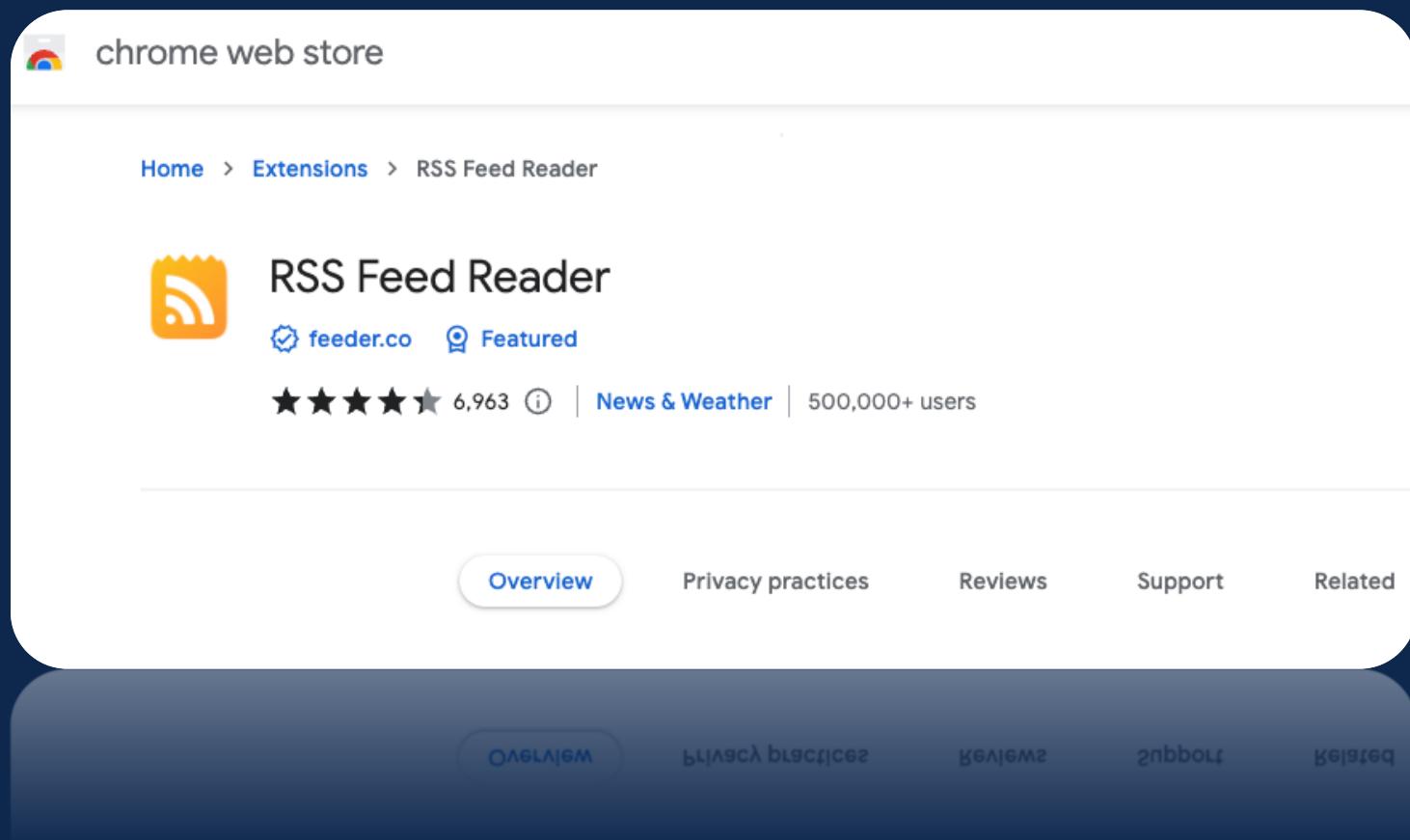
changes to user profile and users endpoint are now available starting in 2023. The old endpoints will be removed after February 2023.

- Changes Portal is going through.
- NEW API being released
- CHANGES to existing API to avoid breakage.
- Information on GUIDES and DOC changes.
- Github SAMPLES



# Change log, New features

## 1) Install chrome extension



# Change log, New features

2) Go to <https://developer.webex-cx.com/documentation/api-changelog>

Click on RSS feed.

OVERVIEW

- Getting Started
- Authentication
- Common API Errors
- Integrations
- Introduction to APIs
- Rate Limiting
- Sandbox

API REFERENCE

- API Changelog

API Changelog

## API Changelog

This page lists recent and upcoming changes to the Webex Contact Center APIs.

**January 11 2023**

New

- Captures API response has been modified. List of historical transcriptions will be available in the modified response along with the recordings details.

---

**November 14 2022**

- User Profile resource has been updated to improve the readability of the API usage. Improved documentation
- User Profile resource has been updated to improve the readability of the API usage. Improved documentation

**Contents**

- 2023
- January
- 2022
- November



# Change log, New features

3) New tab will open using chrome extension.

Click follow.

FOLLOW THIS FEED WITH Feeder + Follow

 WEBEX CONTACT CENTER FOR DEVELOPERS API CHANGELOG • JUST NOW

**New 2023-01-11**

Captures API response has been modified. List of historical transcriptions will be available in the modified response along with the recordings details.



# Change log, New features

4) Any updates on change log will be sent to your browser.

The screenshot shows a web browser window with a dark theme. The address bar displays the URL "Webex Contact Center for Developers API CHANGELOG". The main content area shows a list of changes:

- New 2023-01-11**  
Captures API response has been modified. List of historical transcripts will be available in the modified response along with the recording details.
- New 2022-11-14**
- Major 2022-11-14**
- New 2022-11-14**
- New 2022-11-14**
- New 2022-11-14**





# 03

# Understanding OAuth2

# Create Your App

**New App Integration**

---

**Integration Name\***  
Name of your integration

WxCCIntegrationDemo

**Description\***  
Provide some details about what your integration does, how it benefits users, and most importantly, how a user can get started using it. The description should be under 1024 characters.

WxCC Integration Demo

**Redirect URI(s)\***  
One or more URIs that a user will be redirected to when completing an OAuth grant flow.

https://oauth.pstmn.io/v1/callback

https://solutionassuranceeu220.eu.webexconnect.io/callback

http://localhost:3000/oauth/redirect

http://localhost:5000/auth/webex/callback

[Add URI](#)

**Scopes\***  
Scopes define the level of access that your integration requires.

cjp:config  
 cjp:config\_write  
 cjp:config\_read  
 spark:people\_read  
 cjp:user

By creating this app, you accept the [Terms of Service](#) and [Privacy Policy](#).

Cancel

Add Integration

Documentation
My Webex Apps
Sign Out

- Building secure connection to your App
  - One integration for multiple Apps

webex Contact Center for Developers

Documentation

YB

## Congratulations! 🎉

Your integration has been created. Use the OAuth credentials below to finish building your integration.

### WxCCIntegrationDemo

#### OAuth settings

##### Client ID

C8639ebdd80caee9706388c2dd0024463b11f58d502921751b2d9002dc2b7d1

[Copy](#)

##### Client Secret

3359b2d8b89df1209709bd5eb4873764725fdd14a57fd5ab005116f1597d1de7

[Copy](#)

##### OAuth Authorization URL

You can use the URL below to initiate an OAuth permission request for this app. It is configured with your redirect URI and app scopes. Be sure to update the state parameter.

[https://webexapis.com/v1/authorize?](https://webexapis.com/v1/authorize?client_id=C8639ebdd80caee9706388c2dd0024463b11f58d502921751b2d9002dc2b7d1&response_type=code&redirect_uri=http%3A%2F%2Flocalhost%3A5000%2Fauth%2Fwebex%2Fcallback&scope=cj%3Auser%20cj%3Aconfig_write%20cj%3Aconfig%20cj%3Aconfig_read&state=set_state_here)

client\_id=C8639ebdd80caee9706388c2dd0024463b11f58d502921751b2d9002dc2b7d1&response\_type=code&redirect\_uri=http%3A%2F%2Flocalhost%3A5000%2Fauth%2Fwebex%2Fcallback&scope=cj%3Auser%20cj%3Aconfig\_write%20cj%3Aconfig%20cj%3Aconfig\_read&state=set\_state\_here

#### Integration Name\*

Name of your integration

WxCCIntegrationDemo

[Edit](#)

# Webex OAuth2 Mechanism - Overview

The screenshot shows a two-step process. Step 1: A browser window with a 'Cisco Webex' logo and a 'Welcome to We...' message. It has an 'Email address' input field and a 'Sign In' button. Step 2: A modal window titled '200 / OK' showing a JSON API response. The response includes an 'intervalStartTime' of 1605499200000, an 'agentId' of AXU\_1xy7NNx5c-IBprEB, an 'agentName' of Grace Loh, a 'teamId' of AXVKitNLwb-tZrANaW6S, a 'teamName' of WxOneTeam, and a 'channels' array with one item. This item contains a 'channelType' of 'social', and various task-related metrics like total assigned, offered, accepted, rejected, transferred, engaged duration, and hold duration, all set to 0. Below the modal is a browser developer tools Network tab showing a request to 'localhost:5000/auth/webex/callback?code=NmE5Y'. The Network tab also displays the JSON response with fields like 'access\_token', 'refresh\_token', and 'token\_type'.

```
200 / OK
```

```
{"meta": {"orgId": "6608122d-50f5-4558-be72-4e9f26c6a42f"}, "data": [{"intervalStartTime": 1605499200000, "agentId": "AXU_1xy7NNx5c-IBprEB", "agentName": "Grace Loh", "teamId": "AXVKitNLwb-tZrANaW6S", "teamName": "WxOneTeam", "channels": [{"channelType": "social", "totalAssignedTasks": 0, "totalOfferedTasks": 0, "totalAcceptedTasks": 0, "totalRejectedTasks": 0, "totalTransferredTasks": 0, "totalEngagedDuration": 0, "totalHoldDuration": 0}]}]}
```

```
localhost:5000/auth/webex/callback X +
```

```
localhost:5000/auth/webex/callback?code=NmE5Y
```

JSON Raw Data Headers

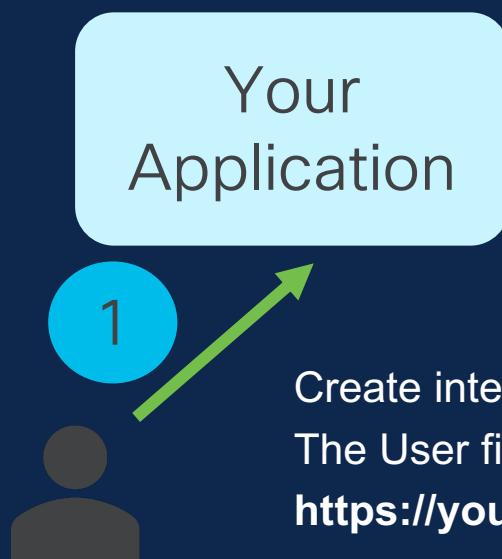
Save Copy Collapse All Expand All Filter JSON

```
access_token: "ODlmNWY4ZDYtYzH0S00ZDMzLWEyY2YtYjNlMDMwNDUwYjI2NzBhNzUzM0Dl...  
expires_in: 43199  
refresh_token: "NDU1NWEyMDAtNWYyNi00YTk0LWI1MzctZmEyNTg40DhiMzBhNDljNwM4Y...  
refresh_token_expires_in: 5182014  
token_type: "Bearer"
```

CISCO Webex © 2023 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

- Create an integration to obtain clientid and client secret.
- Once you obtain an Org's **access\_token** and **refresh\_token**, you can access their data.
- Use the **refresh\_token** before expiry (12 hours) to get another **access\_token**

# OAuth2 Access Token Flow

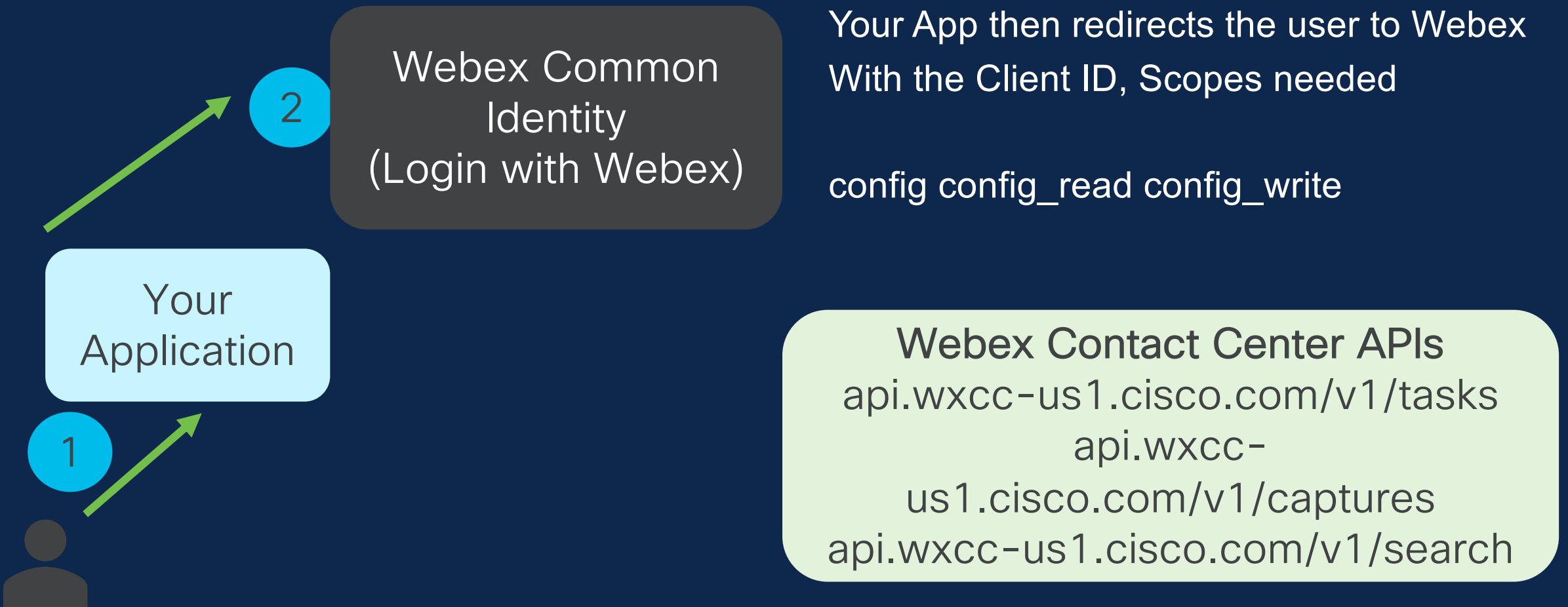


Webex Common Identity  
(Login with Webex)

- User Profiles and Scopes apply.
- You need a Contact Center Administrator Role to Read and Write to all endpoints.

Webex Contact Center APIs  
`api.wxcc-us1.cisco.com/v1/tasks`  
`api.wxcc-`  
`us1.cisco.com/v1/captures`  
`api.wxcc-us1.cisco.com/v1/search`

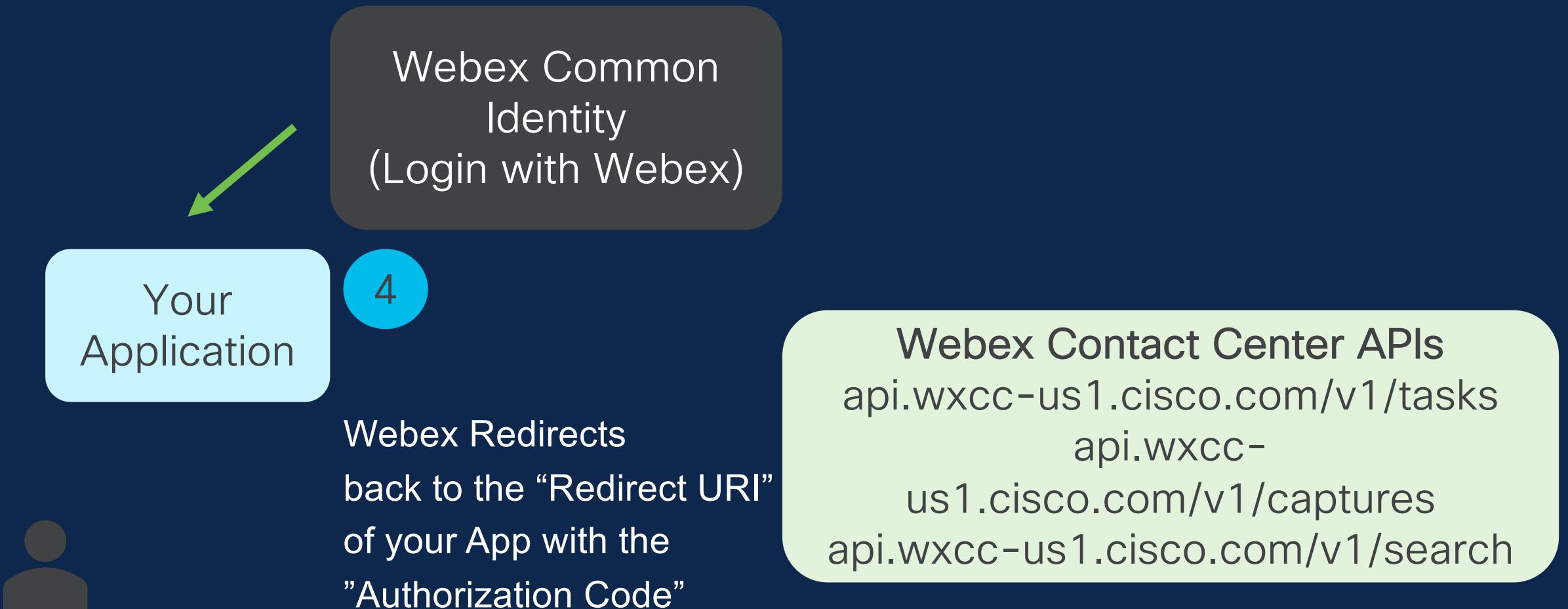
# OAuth2 Access Token Flow



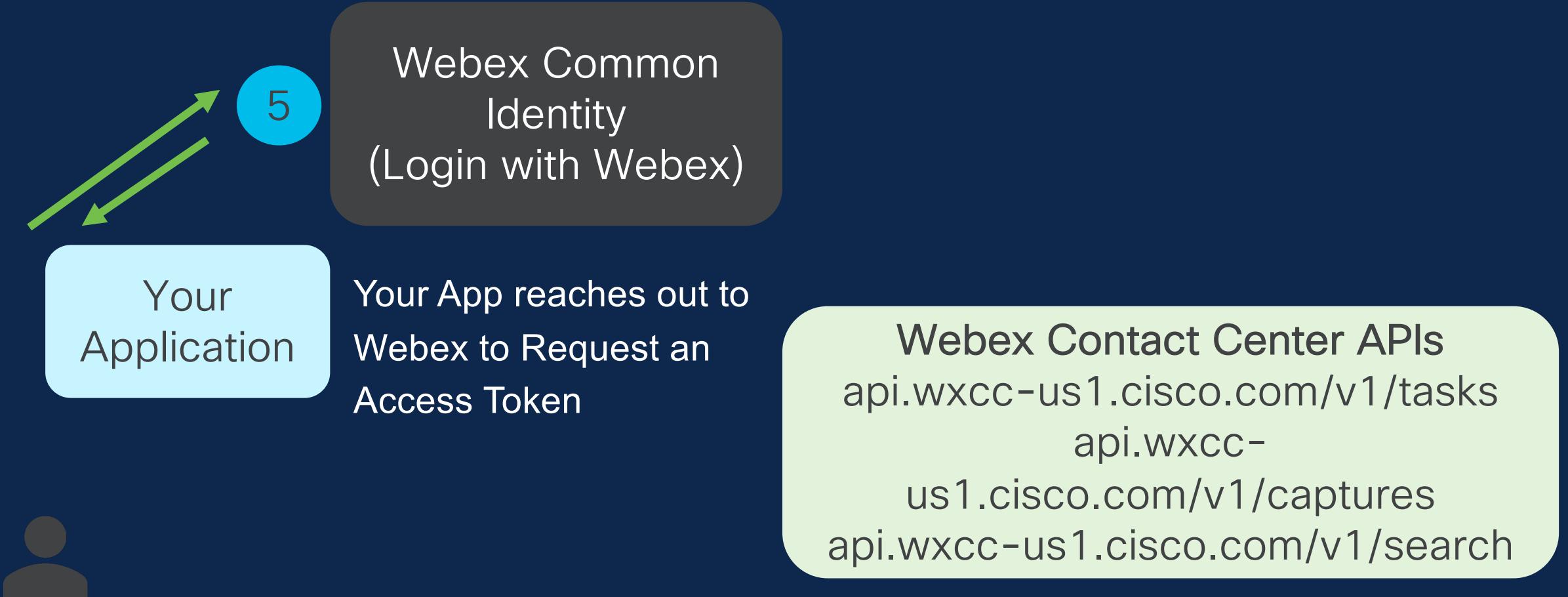
# OAuth2 Access Token Flow



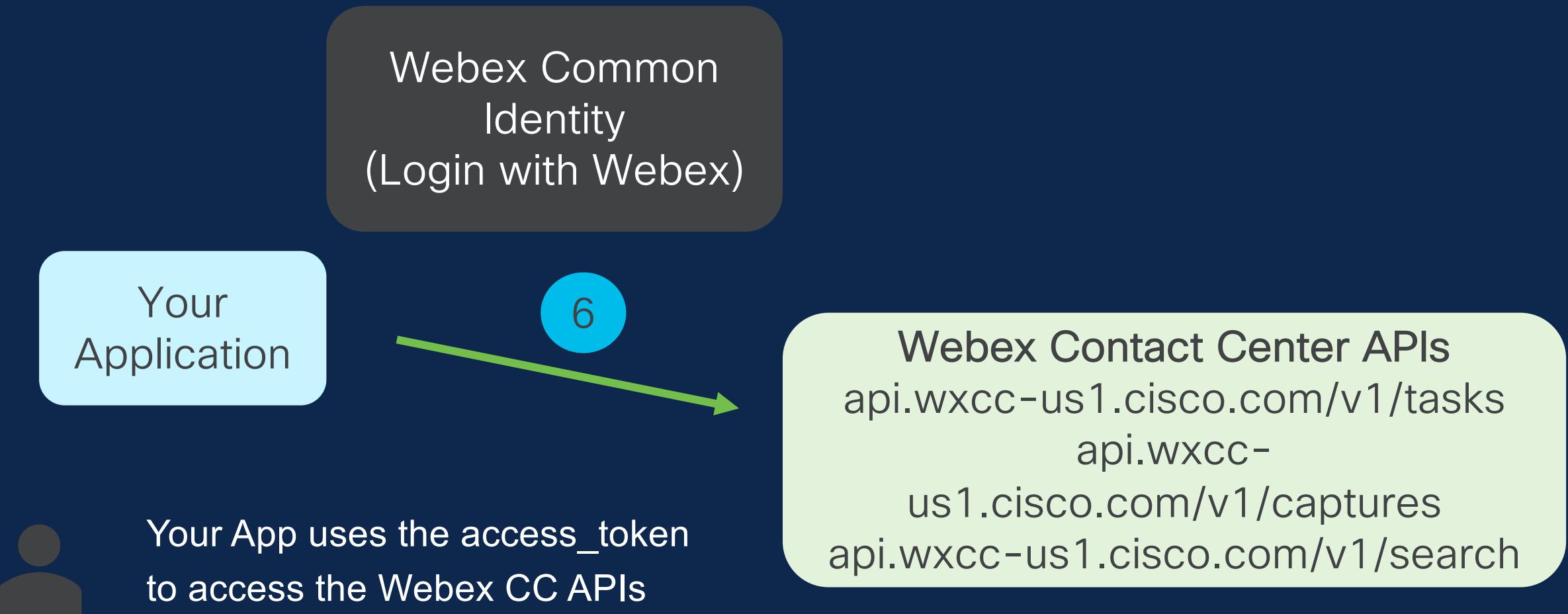
# OAuth2 Access Token Flow



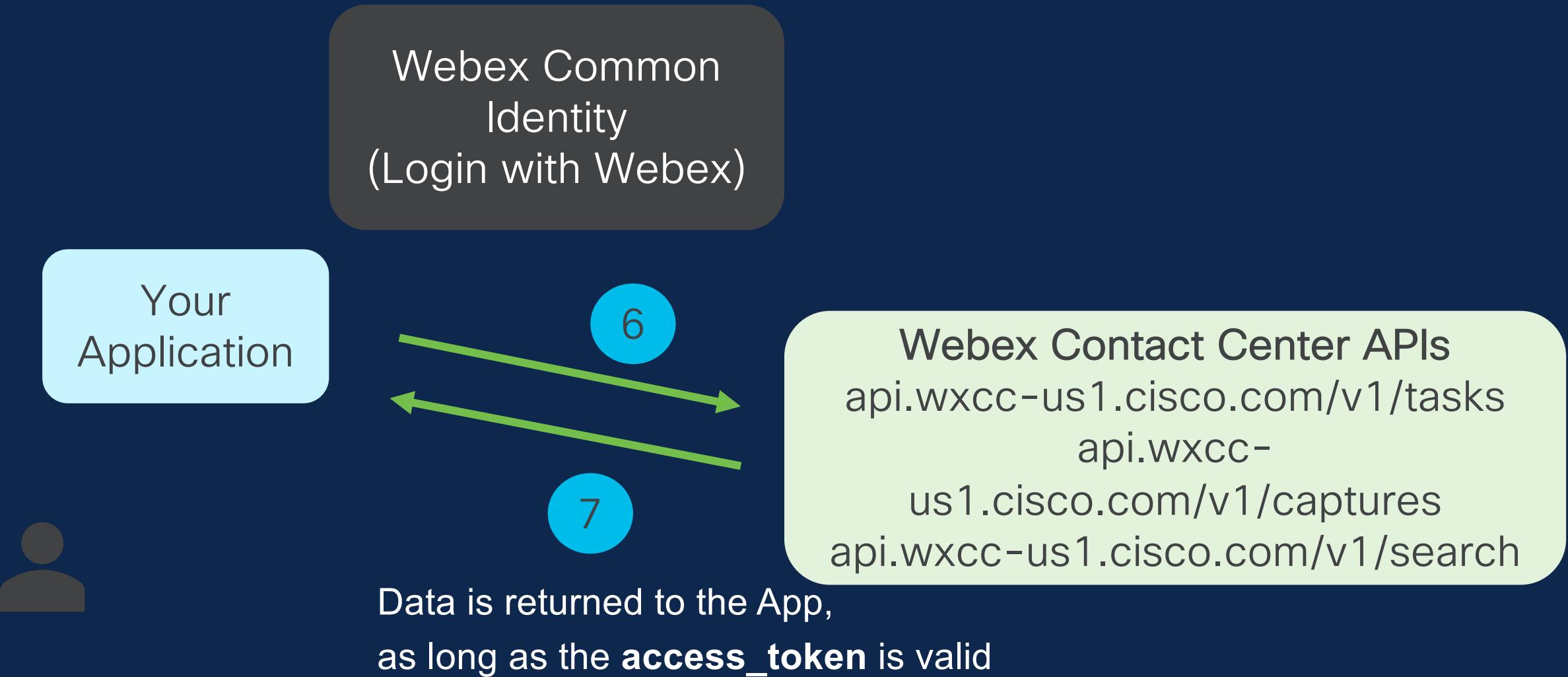
# OAuth2 Access Token Flow



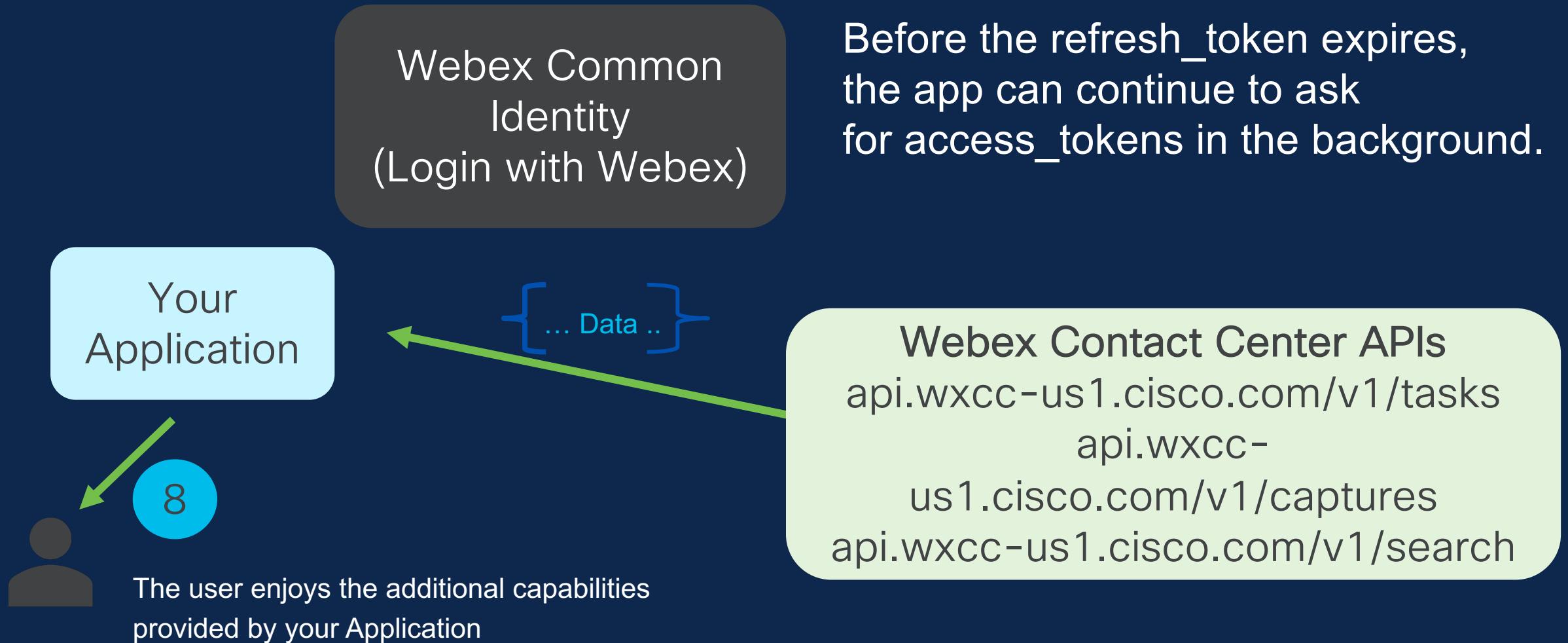
# OAuth2 Access Token Flow



# OAuth2 Access Token Flow



# OAuth2 Access Token Flow





# 04

# Use Cases

# Callback from website

# Use case - Callback from website, how ?

1) Client: The website where the user is making the request from.

Client

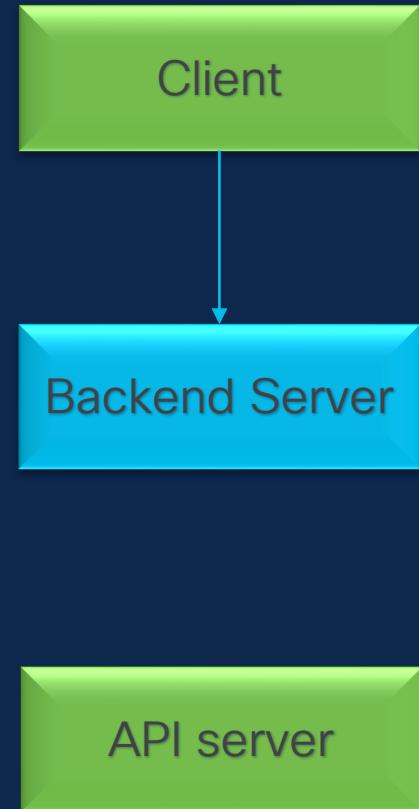
Backend Server

API server

# Use case - Callback from website, how ?

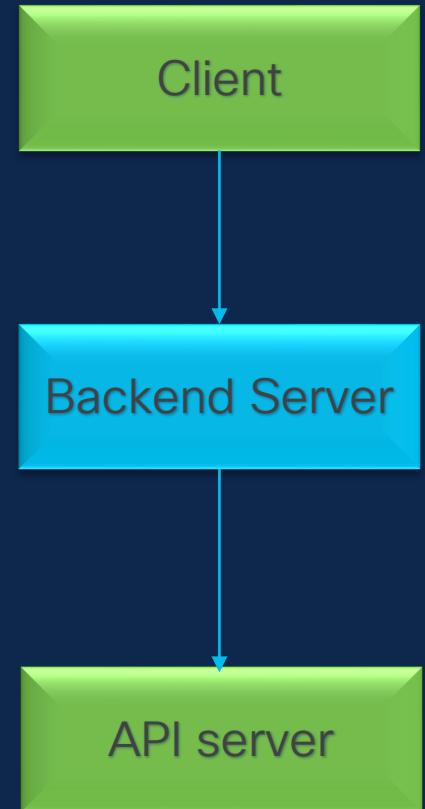
1) Client: The website where the user is making the request from.

2) POST request from client to backend server.



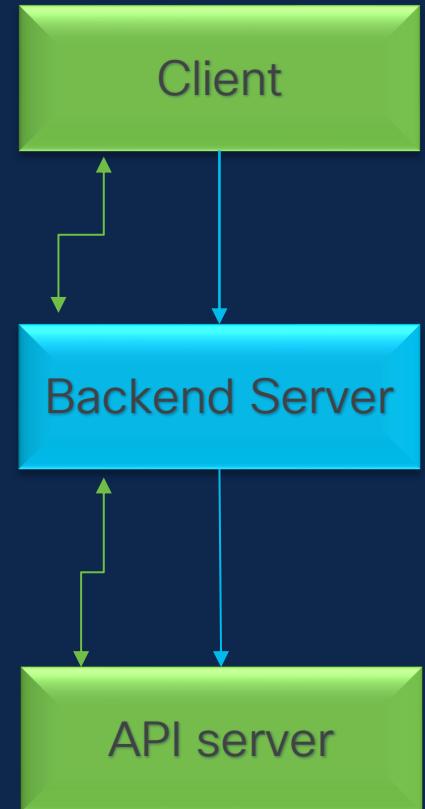
# Use case - Callback from website, how ?

- 1) Client: The website where the user is making the request from.
- 2) POST request from client to backend server.
- 3) Backend server - To receive the POST request.



# Use case - Callback from website, how ?

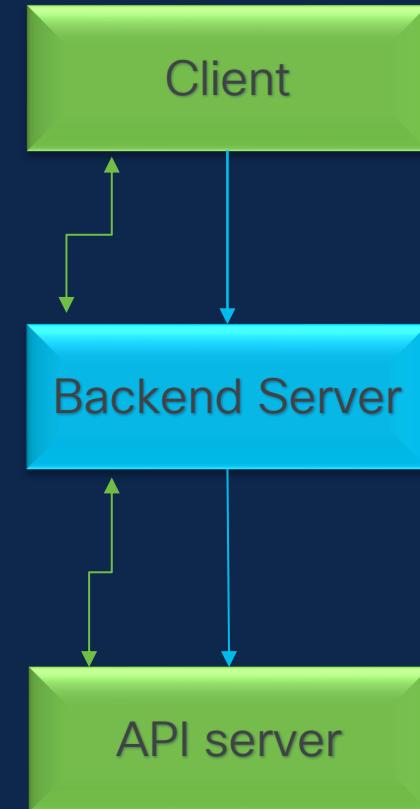
- 1) Client: The website where the user is making the request from.
- 2) POST request from client to backend server.
- 3) Backend server - To receive the POST request.
- 4) Backend server - Call Webex CC **Task API** to initiate callback request.  
Wait for response code and continue accordingly. (CONTRACT)  
**POST Parameters - Outbound Type – Callback, EP ID – Out Bound.**



# Use case - Callback from website, how ?

- 1) Client: The website where the user is making the request from.
- 2) POST request from client to backend server.
- 3) Backend server - To receive the POST request.
- 4) Backend server - Call Webex CC Task API to initiate callback request.  
Wait for response code and continue accordingly.

**POST Parameters - Outbound Type – Callback, EP ID – Out Bound.**



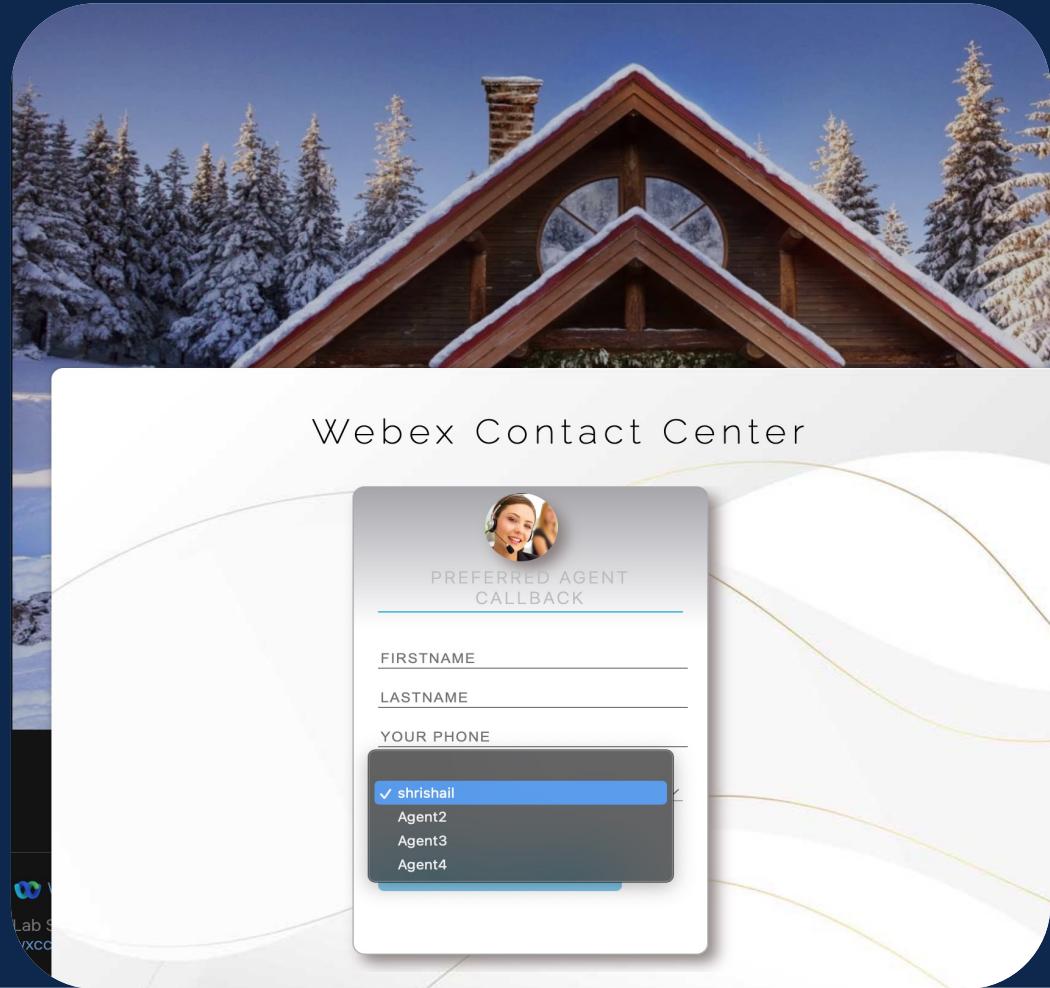
# Preferred Agent Callback.

# Use case – Preferred agent callback, how?

- Client: Browser, Request from website.
- Form - Programmatically pull agent name.
- Submit request.
- Initiate a POST request to **Task API**  
with **outboundType as ‘EXECUTE\_FLOW’**  
**and EP ID of the inbound entry point to**  
**trigger Flow.**
- Flow with QtoAgent node  
and a flow variable that will receive  
agent email ID.

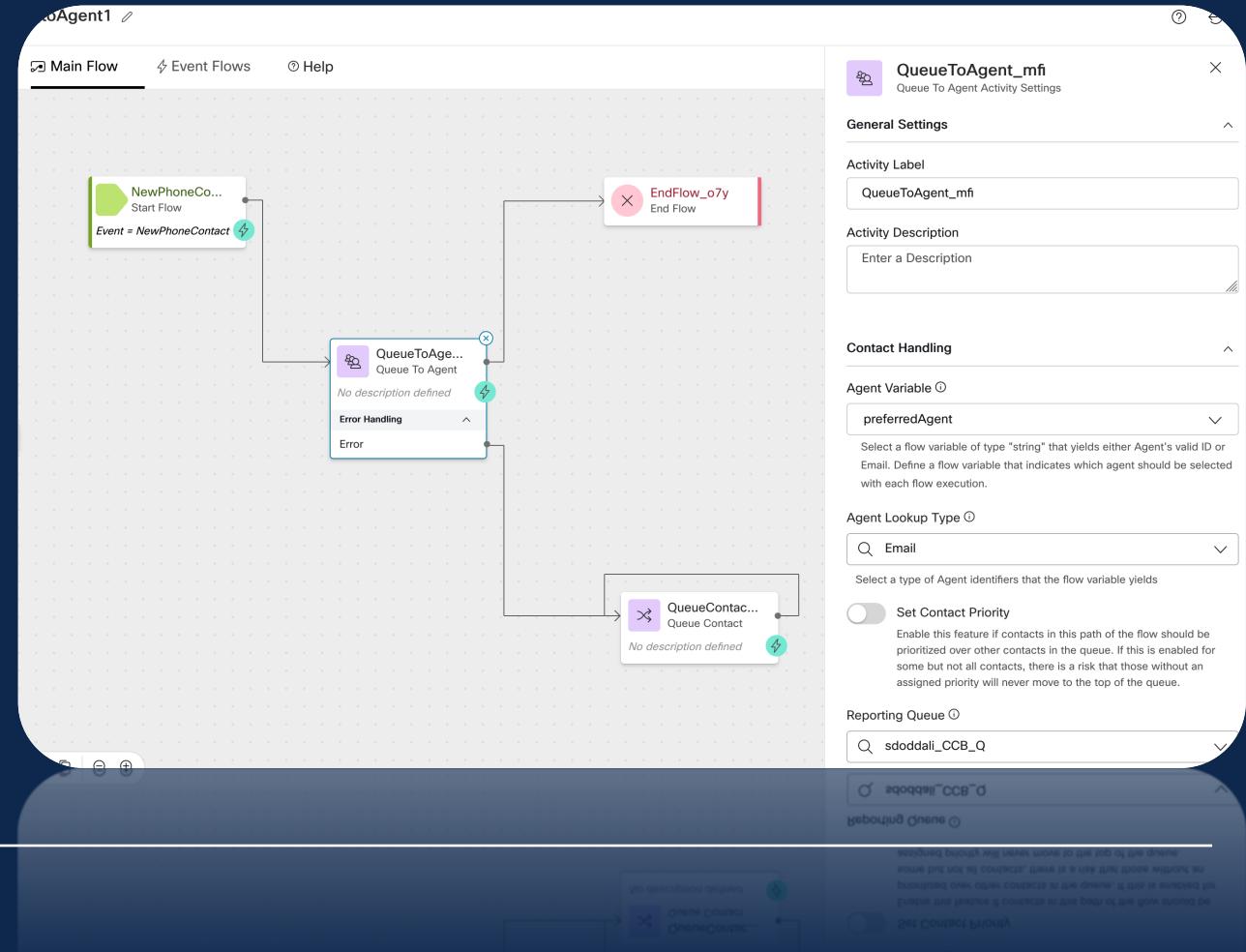
# Use case – Preferred agent callback, how?

- Client: Browser, Request from website.
- Form - Programmatically pull agent name.
- Submit request.
- Initiate a POST request to **Task API** with **outboundType as ‘EXECUTE\_FLOW’** and **EP ID of the inbound entry point to trigger Flow.**
- Flow with QtoAgent node and a flow variable that will receive agent email ID.



# Use case - Preferred agent callback, how?

- Client: Browser, Request from website.
- Form - Programmatically pull agent name.
- Submit request.
- Initiate a POST request to **Task API** with **outboundType as ‘EXECUTE\_FLOW’** and **EP ID of the inbound entry point to trigger Flow.**
- Flow with QtoAgent node and a flow variable that will receive agent email ID.



# Play Emergency Message

# Use case – Play emergency message.

Consider a scenario where there is an emergency and you must play an emergency message to the callers.

# Use case - Play emergency message.

Question - How many contact center APIs are being used for this demo ?

# Use case – Play emergency message.

Question - How many contact center APIs are being used for this demo ?

Answer – 2.

# Use case - Play emergency message.

The screenshot shows a REST API documentation interface for the 'Create Audio File' endpoint. The top navigation bar includes 'Audio Files > Create Audio File' and 'Version 1'. The main section is titled 'Create Audio File' and contains the following details:

- Method:** POST /organization/{orgid}/audio-file
- Description:** Create a new Audio File for a given organization.
- Path Parameters:**
  - orgid \*** **uuid**: Organization ID to be used for this operation. The specified security token must have permission to interact with the organization.  
Example: "b94a28f3-8b27-43aa-9794-2b9a846050e2"
- Request Body:**
  - audioFile \*** **binary**: The audio file to be uploaded.
  - blobId**: A unique identifier for the blob.
  - contentType**: The content type of the audio file.

# Use case - Play emergency message.

The screenshot displays a REST API documentation interface with three main sections:

- Create Audio File**:
  - Method:** POST /organization/{orgid}/audio-file
  - Description:** Create a new Audio File for a given organization.
  - Path Parameters**: orgid \* uuid
  - Request Body**: audioFile \* binary
- Sample Code**: curl --request POST \  
--url https://api.wxcc-us1.cisco.com/organization/b94a28f3-8b27-43aa-9794-2b9a846050e2 \  
--header 'Accept: \*/\*' \  
--header 'Authorization: Bearer YOUR\_TOKEN' \  
--header 'Content-Type: multipart/form-data' \  
--form audioFile= \  
--form blobId= \  
--form contentType=
- Try Out**: A button to execute the API call.

**Update Global Variables**:

- Method:** PUT /organization/{orgid}/cad-variable/{id}
- Description:** Update Global Variables by using the specific ID for an Org ID. Required fields in payload are agentEditable, variableType, agentViewable, reportable, active, defaultValue.
- Path Parameters**: orgid \* uuid
- Request Body**: { "agentViewable": false, "active": false, "agentEditable": false, "name": "", "reportable": false, "organizationId": "" } (with example values)

A third section is partially visible on the right, showing a Request tab with an Authorization section (using personal access token) and a Parameters section (with orgid and id fields).



# Use case - Play emergency message.

The screenshot shows a web-based application interface for managing global variables. The top navigation bar includes tabs for 'Solution Assurance ESR GT', 'Dashboard', 'Address Book', and 'Global Variables'. The current page is titled 'Global Variable'.

**General Settings**

Name	EmergencyGlobalVariable
Description	DO NOT DELETE_SUMMIT
Variable Type	String
Default Value	HoldMusic
Strings support alphanumerics and spaces.	
Status	Active



# Use case - Play emergency message.

The image shows two screenshots of a web-based management interface. The left screenshot displays the 'Global Variable' configuration for an 'EmergencyGlobalVariable'. The right screenshot shows a list of 'Audio Files' with two entries: 'holdmusic 1.wav' and 'HoldMusic.wav'.

**Global Variable**

**General Settings**

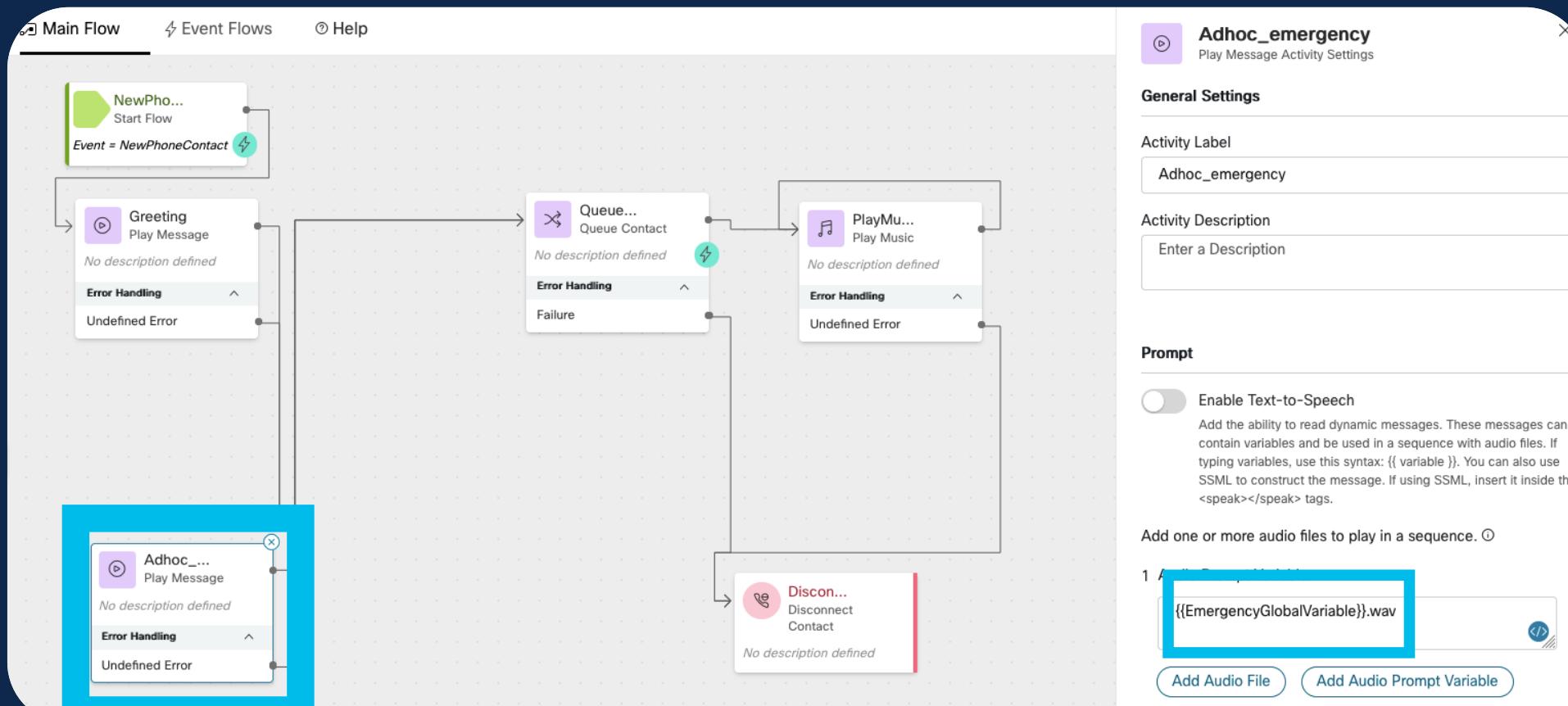
Name	EmergencyGlobalVariable
Description	DO NOT DELETE_SUMMIT
Variable Type	String
Default Value	HoldMusic
Strings support alphanumerics and spaces.	
Status	Active

**Audio Files**

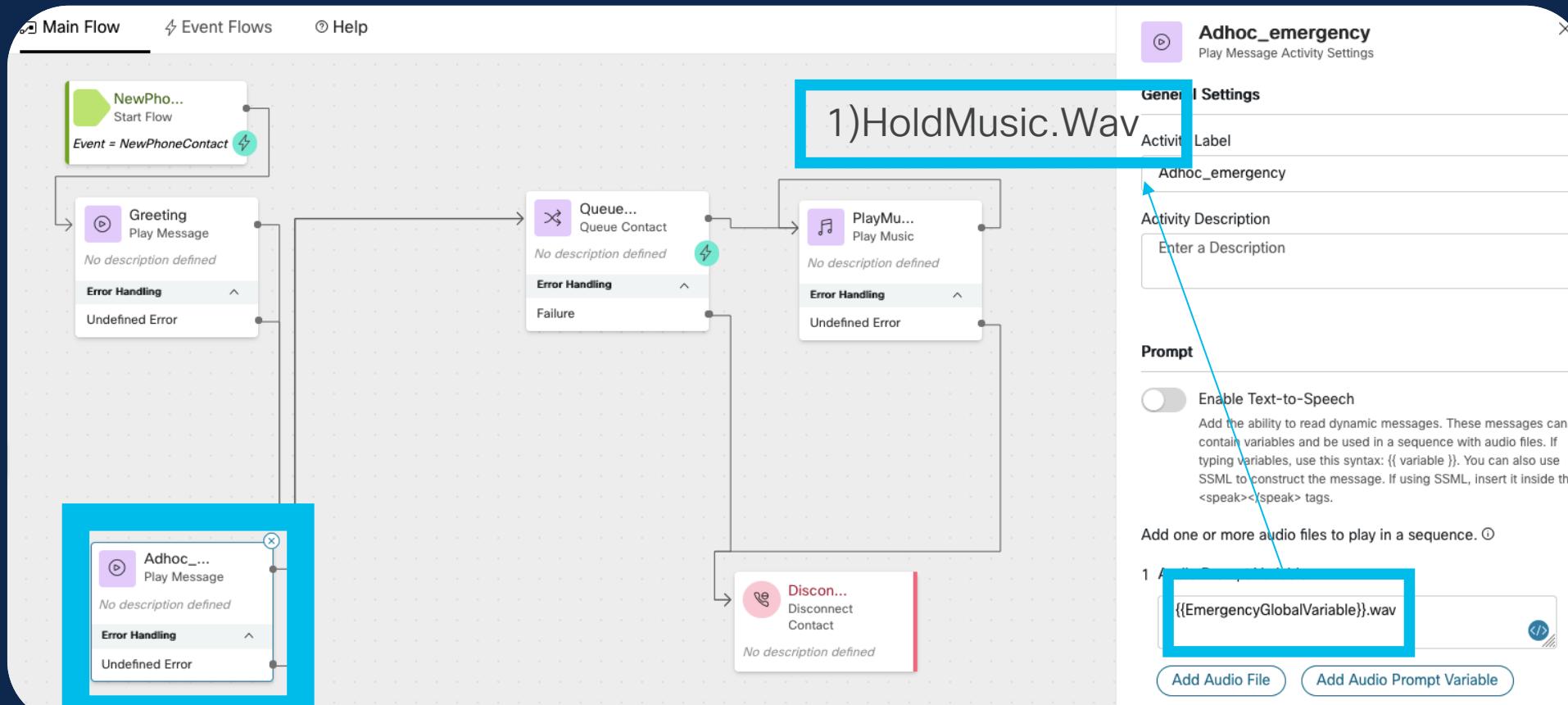
**+ New**

Resource Name
holdmusic 1.wav
HoldMusic.wav

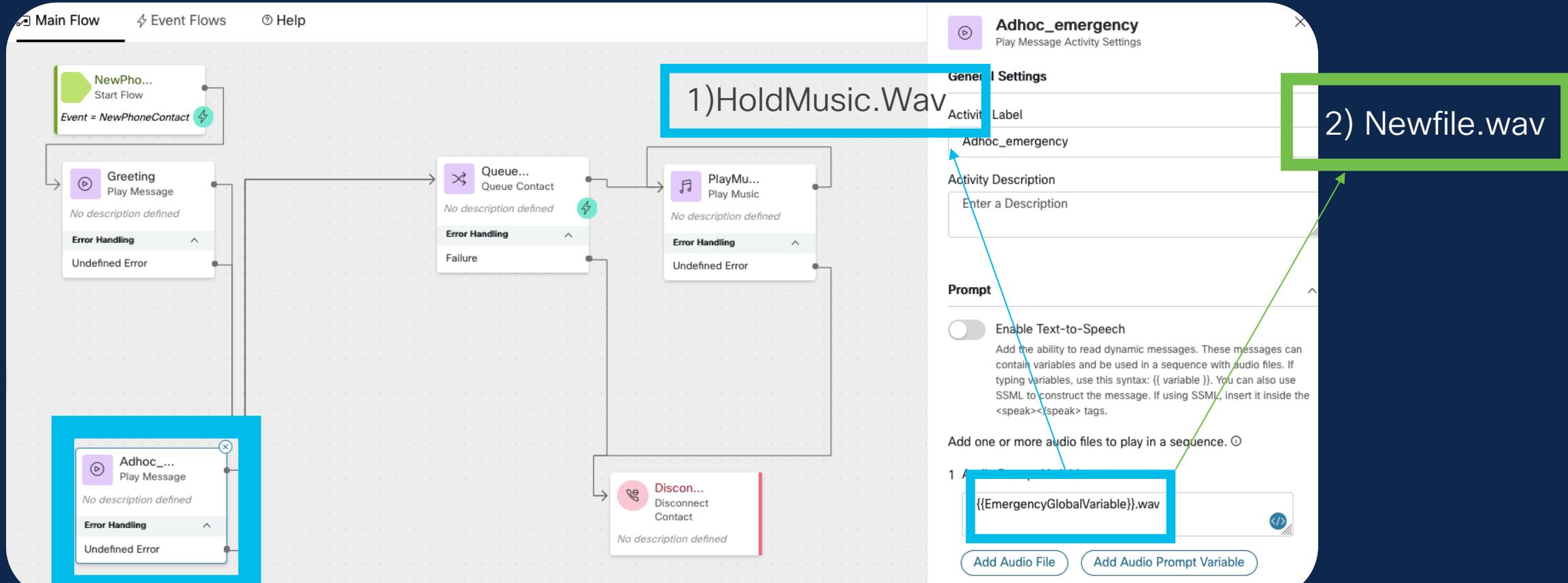
# Use case - Play emergency message.



# Use case - Play emergency message.

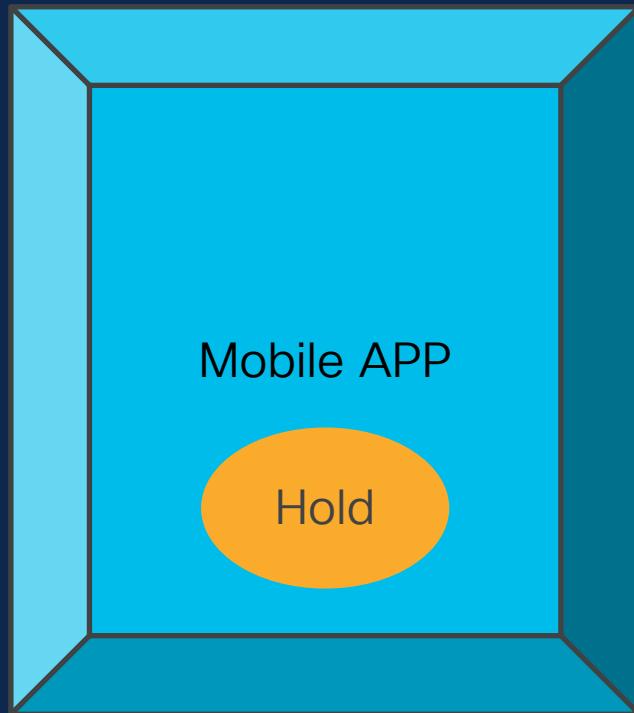


# Use case - Play emergency message.



# Fully Custom Agent Desktop - Mobile App.

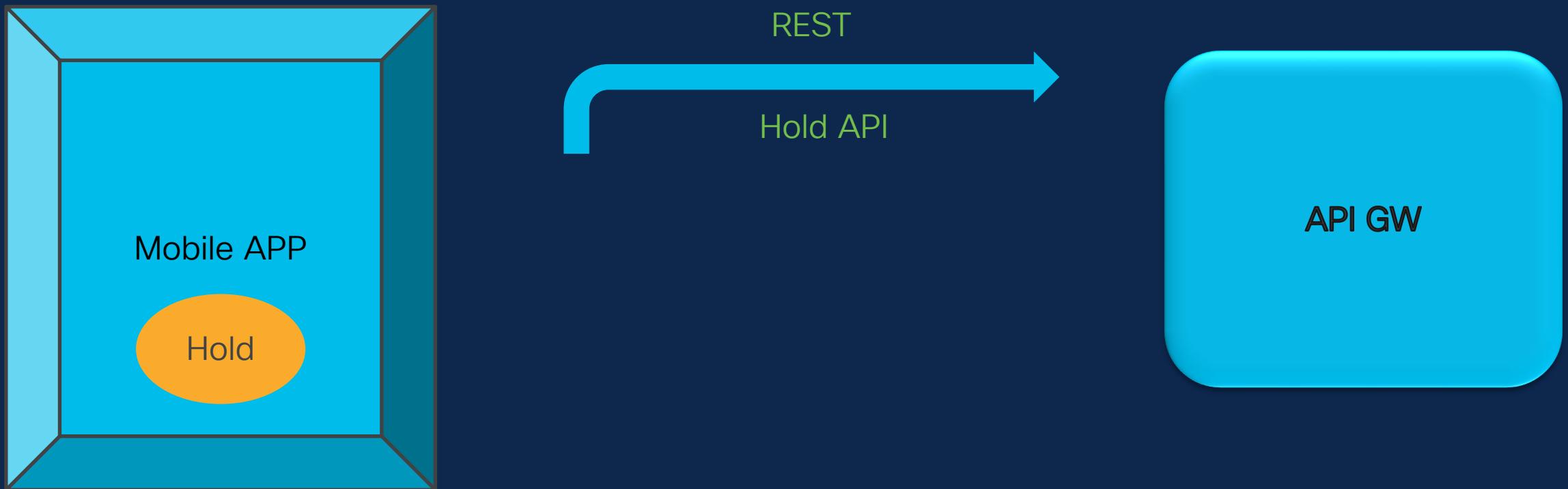
# Use case - Fully custom Agent desktop app



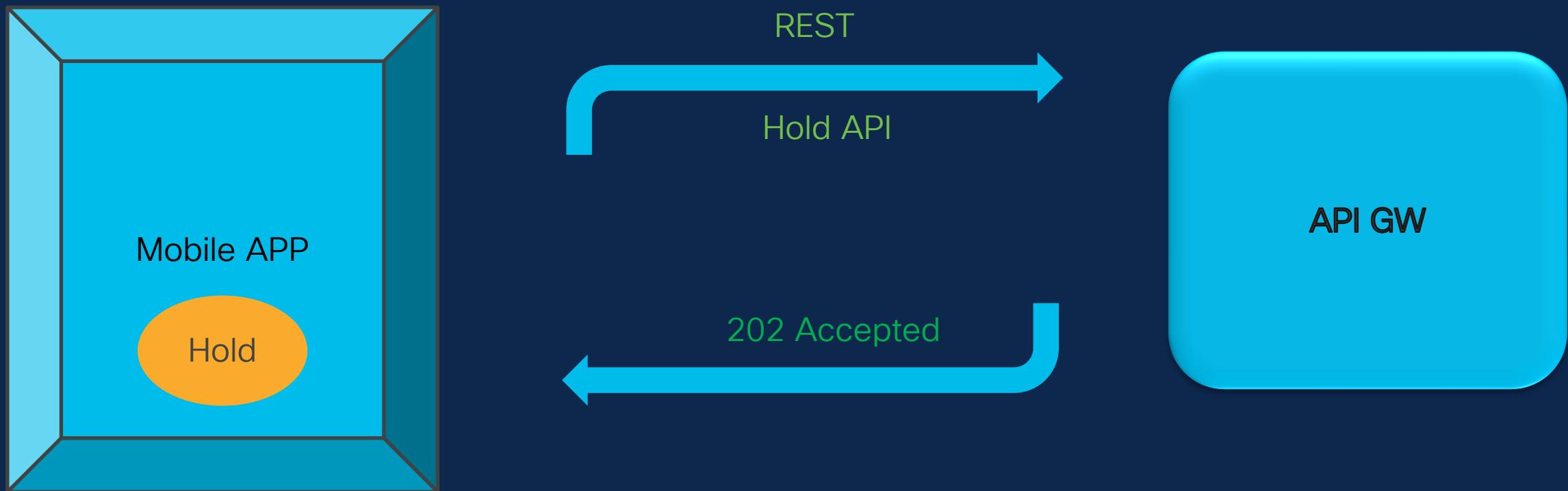
# Use case - Fully custom Agent desktop app



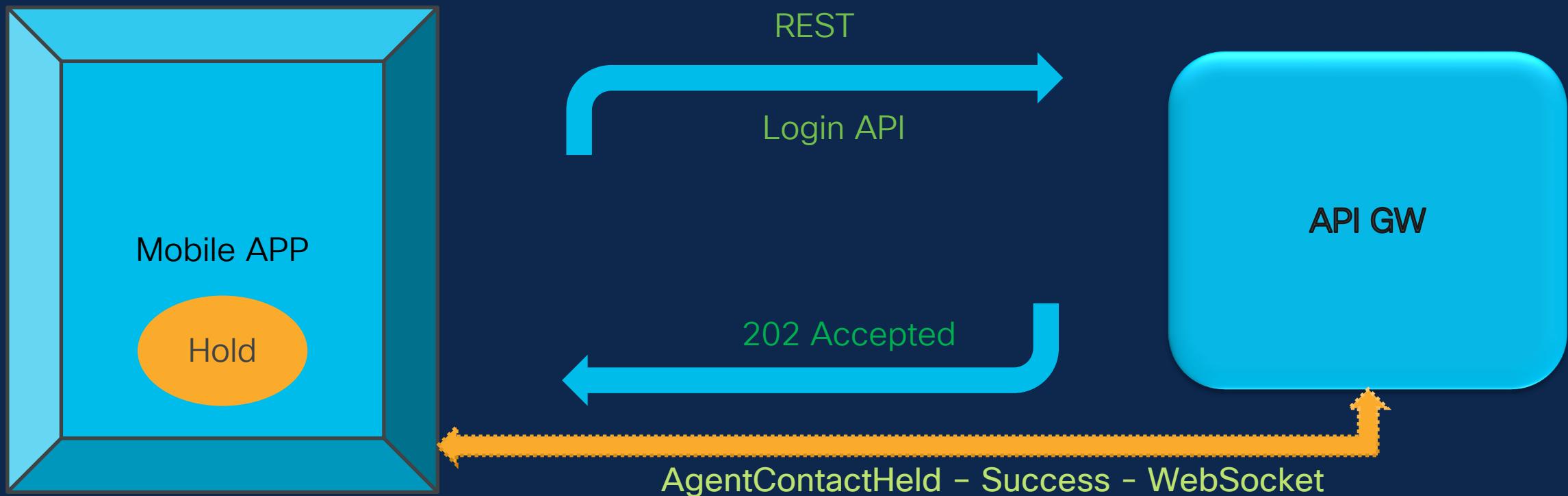
# Use case - Fully custom Agent desktop app



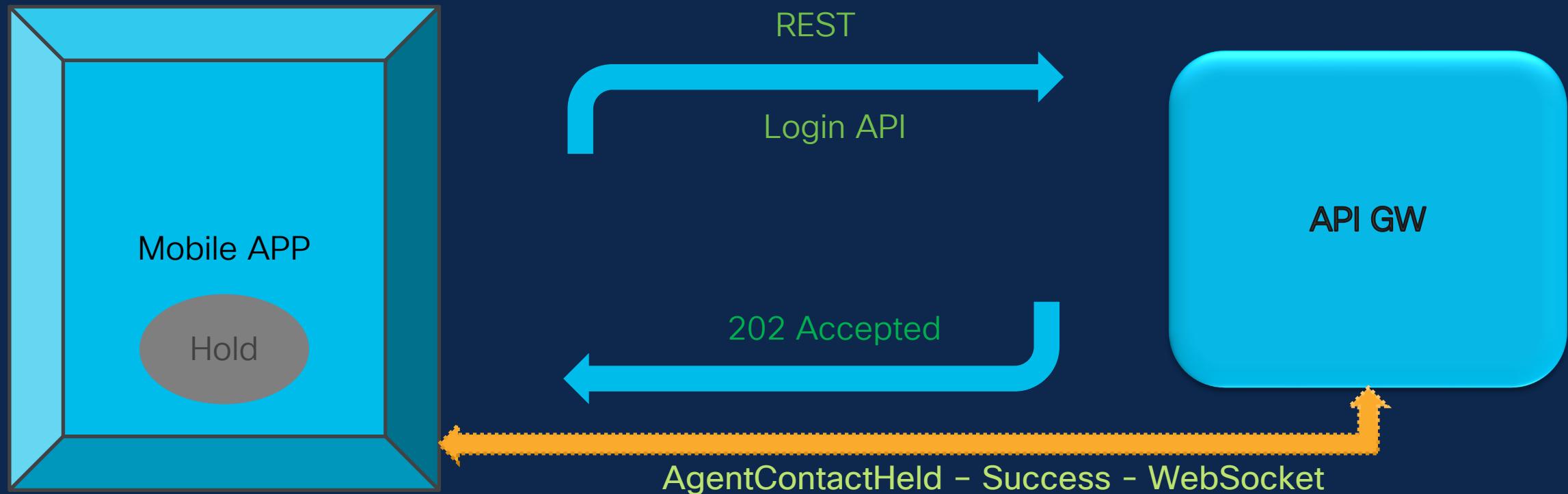
# Use case - Fully custom Agent desktop app



# Use case - Fully custom Agent desktop app



# Use case - Fully custom Agent desktop app



# Wallboard



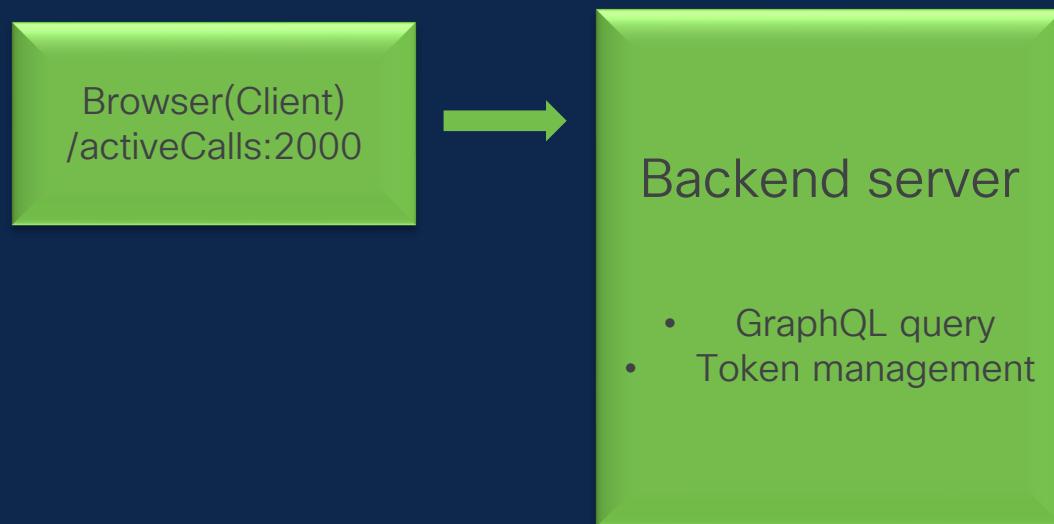
# Use case – WallBoard

- Making a backend API calls every X seconds *from browser.*

Browser(Client)  
/activeCalls:2000

# Use case – WallBoard

- Making a backend API calls every X seconds from browser.
- Design the query depending upon the data you would like to see on wallboard.
- Token management to rotate access token every 10 hours.



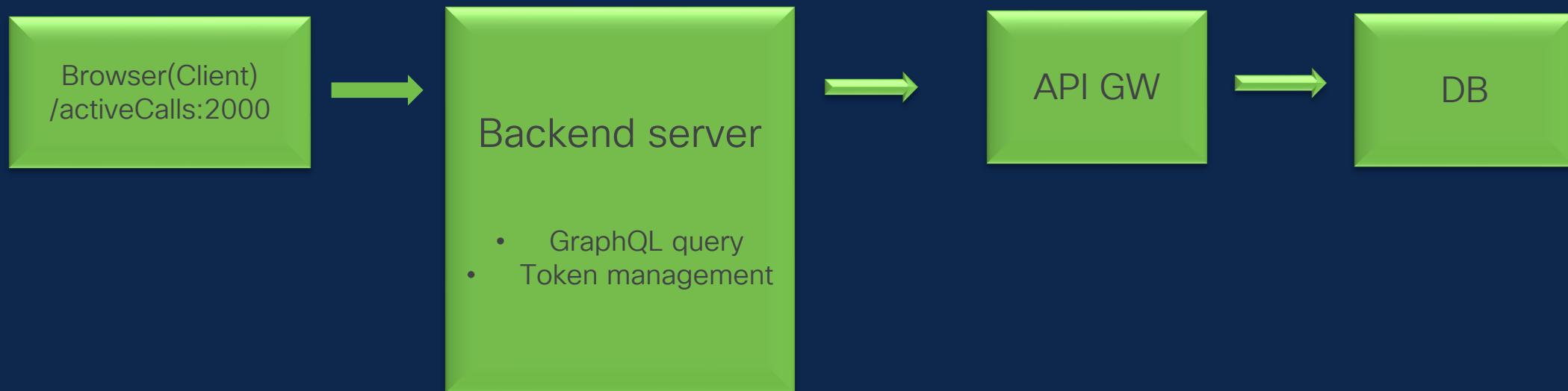
# Use case – WallBoard

- Making a backend API calls every X seconds from browser.
- Design the query depending upon the data you would like to see on wallboard.
- Token management to rotate access token every 10 hours.
- POST API call to API GW (authorization, rate limiting)



# Use case – WallBoard

- Making a backend API calls every X seconds from *browser*.
- Design the query depending upon the data you would like to see on wallboard.
- Token management to rotate access token every 10 hours.
- POST API call to API GW (authorization, rate limiting)
- Call to DB.



# BUT WAIT THERE'S MORE



# Query....?

```
1  {
2    #FILTER: Fetch Real-time (Active) Queued Tasks on the System – using filters.
3    task(
4      from: {{time_milliseconds_lookback}} #This can be set to Date.now() – (days * 24 * 60 * 60 * 1000) for look back in days
5      to: {{time_milliseconds}} #This can be set to Date.now() in ms
6      filter: {
7        #The main filter for active tasks is isActive: true
8        and: [
9          { channelType: { equals: telephony } }
10         { status: { equals: "parked" } }
11         #direction: { equals: "inbound" } }
12         { isActive: { equals: true } }
13       ]
14     aggregations: {
15       field: "id"
16       type: count
17       name: "Total parked Calls"
18     }
19   ) {
20     tasks {
21       #Add more fields here as needed
22
23       aggregation {
24         name
25         value
26       }
27     }
28   }
29 }
30 }
31 }
32 }
33 }
34 }
```

# Query....?

```
1  {
2    #FILTER: Fetch Real-time (Active) Queued Tasks on the System - using filters.
3    task(
4      from: {{time_milliseconds_lookback}} #This can be set to Date.now() - (days * 24 * 60 * 60 * 1000) for look back in days
5      to: {{time_milliseconds}} #This can be set to Date.now() in ms
6      filter: {
7        #The main filter for active tasks is isActive: true
8        and: [
9          { channelType: { equals: telephony } }
10         { status: { equals: "parked" } }
11         #direction: { equals: "inbound" } }
12         { isActive: { equals: true } }
13       ]
14     }aggregations: {
15       field: "id"
16       type: count
17       name: "Total parked Calls"
18     }
19   ) {
20     tasks {
21       #Add more fields here as needed
22
23       aggregation {
24         name
25         value
26       }
27     }
28   }
29 }
30 }
31 }
32 }
33 }
34 }
```

# Query....?

```
1  {
2    #FILTER: Fetch Real-time (Active) Queued Tasks on the System - using filters.
3    task(
4      from: {{time_milliseconds_lookback}} #This can be set to Date.now() - (days * 24 * 60 * 60 * 1000) for look back in days
5      to: {{time_milliseconds}} #This can be set to Date.now() in ms
6      filter: {
7        and: [
8          { channelType: { equals: telephony } }
9          { status: { equals: "parked" } }
10         #direction: { equals: "inbound" } }
11         { isActive: { equals: true } }
12       ]
13     }
14     aggregations: {
15       field: "id"
16       type: count
17       name: "Total parked Calls"
18     }
19   )
20   tasks {
21     #Add more fields here as needed
22
23     aggregation {
24       name
25       value
26     }
27   }
28 }
29
30 }
31 }
32 }
33 }
34 }
```

# Query....?

```
1  {
2    #FILTER: Fetch Real-time (Active) Queued Tasks on the System – using filters.
3    task(
4      from: {{time_milliseconds_lookback}} #This can be set to Date.now() – (days * 24 * 60 * 60 * 1000) for look back in days
5      to: {{time_milliseconds}} #This can be set to Date.now() in ms
6      filter: {
7        #The main filter for active tasks is isActive: true
8        and: [
9          { channelType: { equals: telephony } }
10         { status: { equals: "parked" } }
11         #direction: { equals: "inbound" } }
12         { isActive: { equals: true } }

13       aggregations: {
14         field: "id"
15         type: count
16         name: "Total parked Calls"
17       }
18     )
19   }
20   tasks {
21     #Add more fields here as needed
22
23     aggregation {
24       name
25       value
26     }
27   }
28 }
29
30 }
31
32 }
33
34 }
```

# Query....?

```
1  {
2    #FILTER: Fetch Real-time (Active) Queued Tasks on the System – using filters.
3    task(
4      from: {{time_milliseconds_lookback}} #This can be set to Date.now() – (days * 24 * 60 * 60 * 1000) for look back in days
5      to: {{time_milliseconds}} #This can be set to Date.now() in ms
6      filter: {
7        #The main filter for active tasks is isActive: true
8        and: [
9          { channelType: { equals: telephony } }
10         { status: { equals: "parked" } }
11         #direction: { equals: "inbound" } }
12         { isActive: { equals: true } }
13       ]
14     }aggregations: {
15       field: "id"
16       type: count
17     }
18   )
19   {
20     tasks {
21       #Add more fields here as needed
22
23       aggregation {
24         name
25         value
26       }
27     }
28   }
29 }
30 }
31 }
32 }
33 }
34 }
```

# 05 Availability & Roadmap

# Developer Experience Roadmap (April'23)

## Available Today

- Detailed API reference, guides and getting started documents
- API flavors supported – REST, gRPC, GraphQL, Webhook events
- SDKs (Desktop JS SDK)
- Sample code and API calls in 6 variants (cURL, JavaScript, Java, C#, Python, Go)
- Embedded interactive editor & try it out.
- Changelog and Subscription
- Partner Showcase
- Support (FAQs, Developer Community)
- Marketplace – Apphub
- Github Sample code & Demos

## Coming Soon

### Developer Portal

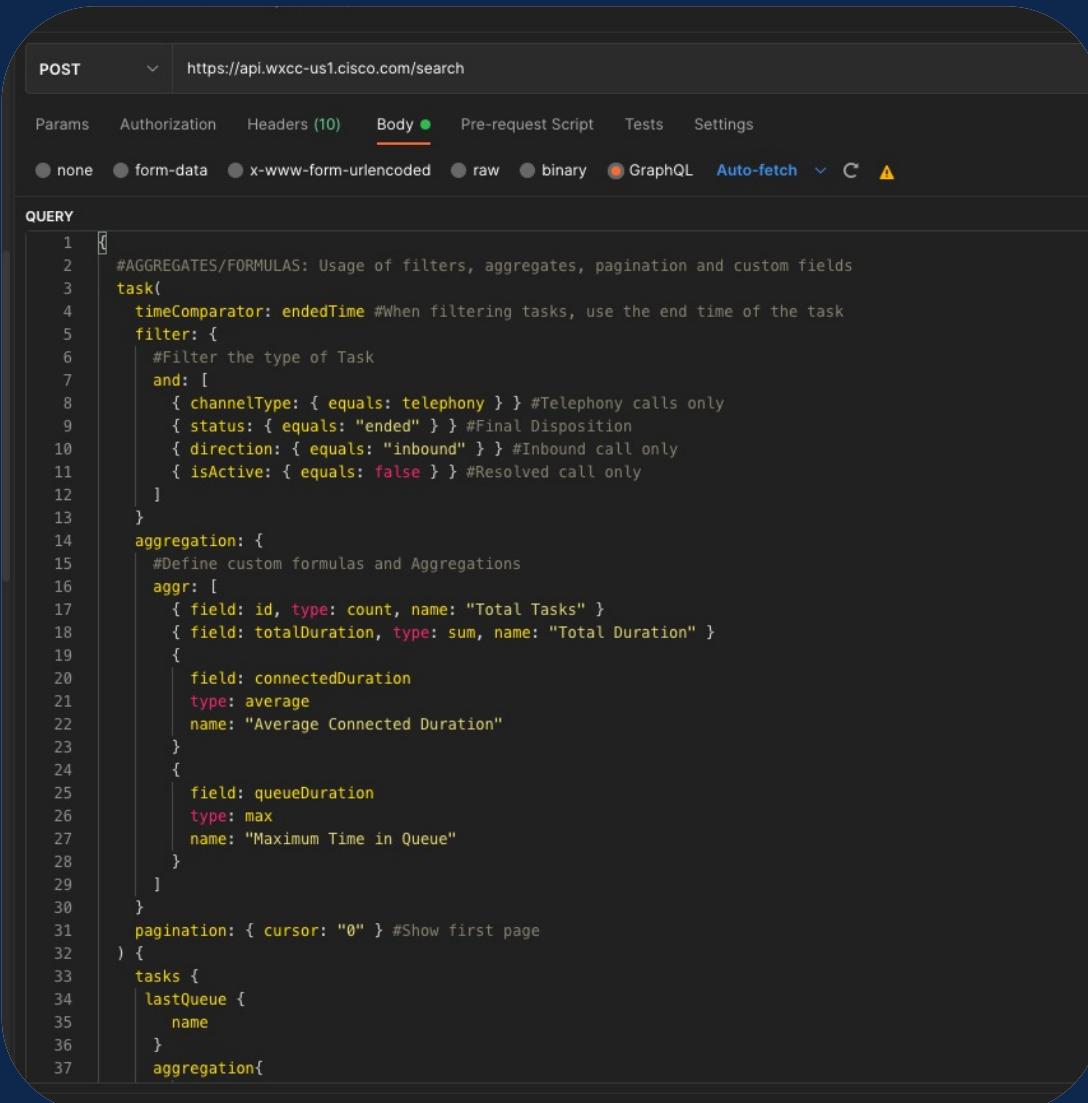
- Organize API by use cases
- Sandbox
- Search functionality
- APIs support from flow.

### Contact Center APIs

- Flow Import/Export APIs
- Bulk Data Export APIs
- Outbound Dialer APIs
- Supervisor Desktop APIs

# 06 Samples & Support

# API Use case Samples Available on GitHub



The screenshot shows a Postman collection interface. The URL is `https://api.wxcc-us1.cisco.com/search`. The 'Body' tab is selected, showing a GraphQL query. The 'Headers' tab shows `Content-Type: application/json`. The 'Params' tab shows `cursor: "0"`.

```
query {
  tasks {
    lastQueue {
      name
    }
    aggregation{
      #AGGREGATES/FORMULAS: Usage of filters, aggregates, pagination and custom fields
      task(
        timeComparator: endedTime #When filtering tasks, use the end time of the task
        filter: {
          #Filter the type of Task
          and: [
            { channelType: { equals: telephony } } #Telephony calls only
            { status: { equals: "ended" } } #Final Disposition
            { direction: { equals: "inbound" } } #Inbound call only
            { isActive: { equals: false } } #Resolved call only
          ]
        }
        aggregation: {
          #Define custom formulas and Aggregations
          agrs: [
            { field: id, type: count, name: "Total Tasks" }
            { field: totalDuration, type: sum, name: "Total Duration" }
            {
              field: connectedDuration
              type: average
              name: "Average Connected Duration"
            }
            {
              field: queueDuration
              type: max
              name: "Maximum Time in Queue"
            }
          ]
        }
        pagination: { cursor: "0" } #Show first page
      ) {
        tasks {
          lastQueue {
            name
          }
          aggregation{

```

- GitHub Repo - CiscoDevNet [webex-contact-center-api-samples](https://github.com/CiscoDevNet/webex-contact-center-api-samples)
- Sample Code - Clone
- ReadMe - Environment.
- Video Tutorials - Use Cases, API, code

<https://github.com/CiscoDevNet/webex-contact-center-api-samples>

# API Use case Samples Available on GitHub

#	Folder Name	Comments	Link
0	postman-sample	Getting Started with the Webex Contact Center APIs and how to create a pair of Client ID or Client Secret for OAuth2 etc.	<a href="#">Postman Sample</a>
1	app-auth-sample	This is a sample application that shows you how to obtain an access token. It also has sample GET calls for Tasks, Agent Stats, Queue Stats, Users, Sites, etc.	<a href="#">OAuth2 Sample</a>
2	configuration-app-sample	This is a sample frontend application that shows you how to use the Configuration APIs for EPs, Queues, Teams, and expose the capabilities on the front-end	<a href="#">Configuration App Sample</a>
3	graphql-sample	This is a sample application that has example calls for the new /search endpoint that is powered by GraphQL. One can formulate multiple request types that support the GraphQL syntax. Both Tasks and Agent Sessions are supported.	<a href="#">GraphQL /search API Sample</a>
4	webhook-email-notify-sample	This is a sample application that shows you how to use Webhooks and send an email notification using webhooks.	<a href="#">Webhook Sample with Email Notifications</a>
5	call-recording-download-sample	This is a sample application that shows you how to use Webhooks - the capture:created Webhook allows you to download a new call recording on the system, to a local file.	<a href="#">Call Recording Download Sample</a>
6	token-app-sample	This is a sample application that shows you how to build a scheduler service that obtains a new access token every 10 hours from Webex and persist this onto your local datastore. The example uses a simple SQLite DB as an example.	<a href="#">Token Management Service Sample</a>
7	callback-sample	This is a sample application that shows you how to build a front end Form to Leverage our brand new Webcallback API that injects a callback call into Webex Contact Center. The example uses a simple JavaScript injection technique to inject the form in any webpage of your choice.	<a href="#">Web Callback API Sample</a>
8	widget-sample-101	This is a sample web component widget that shows you how to build a web component widget from scratch. It also covers how you can read the Desktop STORE to retrieve information and pass these values into the widget via the layout.	<a href="#">Web component Widget Sample &amp; 101</a>
8	samples-javascript-for-web	Four level of API samples for the Web component API. This includes samples for the Desktop API, the Web component API, the Web component API, and the Web component API.	<a href="#">Four Level of API Samples for Web Component API</a>

- Checkout out the GitHub - CiscoDevNet repository:  
**[webex-contact-center-api-samples](https://github.com/CiscoDevNet/webex-contact-center-api-samples)**
- Sample Code
- ReadMe
- Video Tutorials
- Use Cases

<https://github.com/CiscoDevNet/webex-contact-center-api-samples>



# Support - Break Fix

- Open a TAC case.
- Tracking ID.

The screenshot shows a user interface for making API requests. At the top, there's a 'Header' section with a toggle switch labeled 'Use personal access token' which is turned on. Below it is a text input field containing a long, obscured string of characters, with a small info icon next to it. A note below the input says: 'This limited-duration personal access token is hidden for your security.' In the middle, there's a 'Parameters' section with fields for 'orgId\*' (containing '3cd15de4-efeb-4718-97de-6d23fb69e30') and 'uuid'. There are also 'pccc' and 'pcc' fields, both currently empty. A large green 'Run' button is centered below these fields. At the bottom, there's a '403 Response' section showing a JSON error message:

```
{  
  "trackingId": "ccconfig_06894ea1-54f3-4fdd-8ad7-1bf3aacda9a5",  
  "error": {  
    "key": "403",  
    "message": [  
      {  
        "description": "Access denied - Client is forbidden access to the resource"  
      }  
    ]  
  }  
}
```

# Webex Contact Center APIs Support Community

Cisco Community > Technology and Support > For Developers > Developer Collaboration > Contact Center

The screenshot shows the Cisco Community Contact Center APIs Support page. At the top, it displays statistics: 893,945 Discussions, 174,384 Solutions, 1,054,563 Members, and 14,900 Online. Below these are six navigation links: Technology & Support (selected), For Partners, Customer Connection, Webex, Events, and Members & Recognition. The main content area features a section titled "Ask a Question | Answer a Question" with ten topic cards arranged in two rows of five. The topics are: Developer Networking (3923 Discussions), Developer Collaboration (6722 Discussions, highlighted with a blue border), Developer Security (169 Discussions), Developer Data Center (150 Discussions), Developer Internet of Things (334 Discussions); and Developer Cloud (28 Discussions), Developer Services (308 Discussions), Developer DevNet Site (2103 Discussions), Developer Mobility (411 Discussions), and Developer Analytics and Automation Software (50 Discussions). A "View All Topics" button is located at the bottom of this section. The footer contains a "Search" bar and several small text links.

Topic	Discussions
Developer Networking	3923 Discussions
Developer Collaboration	6722 Discussions
Developer Security	169 Discussions
Developer Data Center	150 Discussions
Developer Internet of Things	334 Discussions
Developer Cloud	28 Discussions
Developer Services	308 Discussions
Developer DevNet Site	2103 Discussions
Developer Mobility	411 Discussions
Developer Analytics and Automation Software	50 Discussions

- Label – ‘Webex Contact Center APIs’
- Various discussions available already.

# Developer support



Community

Find answers, post questions, join other Webex developers.

[Join Community](#)



Submit Request

For proper support requests, create a ticket.  
Best for asking private questions or sharing sensitive logs.

[Submit Request](#)

Or email [wxccdevsupport@webex.com](mailto:wxccdevsupport@webex.com)



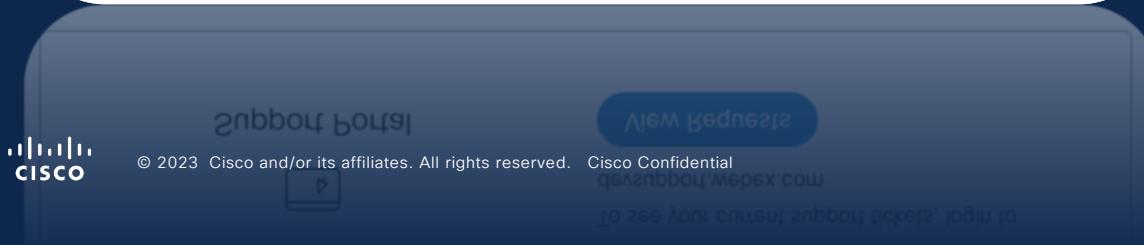
Support Portal

To see your current support tickets, login to [devsupport.webex.com](https://devsupport.webex.com)

[View Requests](#)

- 1) Participate in developer forum.
- 2) Submit support request from developer portal.

Manage your support requests.



# Sandbox

The screenshot shows the "Sandbox" section of the Webex Contact Center for Developers website. On the left, there's a sidebar with a navigation menu:

- OVERVIEW
- Getting Started
- Authentication
- Common API Errors
- Integrations
- Introduction to APIs
- Rate Limiting
- Sandbox** (This item is highlighted with a red border)

Below the sidebar, there's a "API REFERENCE" section with a "Sandbox" button (also highlighted with a red border).

The main content area has a large heading "Webex Contact Center" and a prominent blue button labeled "Request Sandbox".

## Introduction

A Contact Center Developer Sandbox provides you with a licensed org lets you create and test capabilities of the We

If you are signed in to the Developer Portal when you request a sandbox, it will be sent to the email associated with your Webex account; if you sign up.



# Sandbox

≡ webex Contact Center for Developers

OVERVIEW  
Getting Started  
Authentication  
Common API Errors  
Integrations  
Introduction to APIs  
Rate Limiting  
**Sandbox**

## Webex Contact Center

**Request Sandbox**

### Introduction

A Contact Center Developer Sandbox provides you with a licensed org lets you create and test capabilities of the Webex Contact Center.

If you are signed in to the Developer Portal when you request a sandbox, it will be sent to the email associated with your Webex account; if you are not signed in, you will receive an email to sign up.

Request a Sandbox

### Contact Center Developer Sandbox

Webex Contact Center Developer Sandbox is a test environment which grants you administrative access to contact center portals.

This sandbox will be tied to the email address associated with your Webex Account. You can setup only ONE sandbox at any given time which expires in 60 days.

User email ID: **sdoddali@cisco.com**

No Sandboxes associated with this user

By creating this app, you accept the [Terms of Service](#).

**Cancel** **Request a Sandbox**



# Sandbox

## Getting Started

The provisioning email you receive contains the following information for your sandbox organization:

- Organization name: (e.g. jdoessandboxtest-1xep)
- Username/email: (e.g. admin@jdoessandboxtest-1xep.wbx.ai)
- Password
- Webex Site URL: (e.g. jdoesandboxtest-1xep.webex.com)

**Note:** This sandbox will remain active only for a 60 day period. To avoid any potential conflicts with your primary Webex account, it's recommended that you use a private or incognito browser window to perform the sign in.

### 12. Agent/User - Few users are configured

- The users are assigned
  - 1. Default "Agent Based" team, default Agent Profile and default Site and a default Multimedia Profile.
- All the users are enabled for contact center.
- The following user have been pre-configured.
  - 1. Premium Agent 1
    - Username: (e.g. user1Email@email.com)
    - Password
    - Extension (e.g. 2727)
  - 2. Supervisor Agent 2
    - Username: (e.g. user2Email@email.com)
    - Password:
    - Extension: (e.g. 3728)

### 13. Flow - A default basic flow is configured to play a default hold music and redirect the call to the default inbound queue.

### 14. Routing strategy - A routing strategy has been configured mapped to the default inbound entry point and the default basic flow that is active for 60 days from the time of creation. This means the routing strategy is immediately active.

## Sandbox contents

Every sandbox requested from the developer portal has the following contents so that you can test all of the contact center features end to end:

1. **Webex calling for Voice** - Webex calling is the default calling platform.
  - Two (2) PSTN numbers using Cisco PSTN provider. The main number for the default location is reserved.
    1. Main number - Assigned to the default location for Webex calling. This is reserved.
    2. Entry Point number - The second number is configured for Contact center default entry point.
2. **Site** - One (1) site is configured by default.
3. **Team** - An default agent-based team and a capacity based team is available for use.
4. **Multimedia profile** - One (1) default multimedia profile is available
5. **Agent profile** - Two (2) agent profiles available. A default agent profile without Auto wrap-up and an agent profile with auto wrap up is configured.
6. **User profile** - Five (5) pre-configured user profiles are available
  - Administrator Only Profile
  - Administrator Profile
  - Premium Agent User Profile
  - Standard Agent User Profile
  - Supervisor Profile
7. **Administrator** - One (1) administrator user is configured.
8. **Entry Point** - One (1) default inbound entry point and one (1) default out-dial entry point are available
9. **Queue** - One (1) default inbound queue is preconfigured. The inbound queue is associated with the default agent based team
10. **Entry Point mappings** - One PSTN direct number is mapped to the default entry point and default inbound queue.
  - Direct number -
11. **Audio files** - A default hold music is already uploaded.



# Takeaways

The screenshot shows the webex Customer Experience API documentation. On the left, there's a sidebar with navigation links like Overview, Getting Started, Authentication, Common API Errors, Integrations, Introduction to APIs, Rate Limiting, API Reference, Address Book, Agent Profile, Agents, Auxiliary Codes, Captures, Dial Plan, Dialed Number To Entry Point, Entry Points, Estimated Wait Time, Journey, Multimedia Profiles, and Outdial ANI. The main area has tabs for Sample Code and Try Out. Under Sample Code, there's a Request section with Header (Authorization: Use personal access token) and Parameters (orgId: 97cdbf45-ebe2-4687-8341-44d5c7abf101). Below that is a GraphQL section with a query editor containing the following code:

```
1 task(from:1648780752000,
2   to: 1650940752000,
3   filter: {},
4   pagination:{cursor: "0"})
5 {
6   tasks
7 }
```

The right side of the interface shows the results of the GraphQL query, which includes data such as createdTime, channelType, entryPoint, and name.

- Login to [developer.webex-cx.com](https://developer.webex-cx.com) with a WebexCC Admin.
- Try out the APIs live & use the `access_token` for 12 hours -> In a Demo!
- Check out the samples on GitHub
- Request Sandbox.
- Participate in the Cisco Developer Community
- Try out lab.

# References

- Developer Portal: <https://developer.webex-cx.com/>
- Desktop SDK: <https://developer.webex-cx.com/documentation/guides/desktop>
- Partner Community:  
<https://community.cisco.com/t5/contact-center/bd-p/j-disc-dev-contact-center>
- GitHub Samples:  
<https://github.com/CiscoDevNet/webex-contact-center-api-samples>



external

Portal

<https://developer.webex-cx.com/>



The bridge to possible