LTRCCT-2296

From Good to Great: Enhancing Customer Experience with the Webex Contact Center Flow Designer

Dimitri Bokatov, Yaroslav Bondar, Taylan Kucuk

Table of contents

1. Getting Started

1.0.1 Please submit the form below with your Attendee ID.

All configuration entries in the lab guide will be renamed to include your Attendee ID.

Attendee ID:

Save

Your stored Attendee ID is: **No ID stored**

1.1 Overview

1.1.1 Learning Objectives

Welcome to **"From Good to Great - Enhancing Customer Experience with the Webex Contact Center Flow Designer"** instructor-led Lab!

This advanced lab is designed to empower you with the skills to craft exceptional customer journeys using the **Webex Contact Center Flow Designer**. Over the course of this lab, you'll work hands-on with features and integrations that bring intelligence and efficiency to every interaction. Take your time to explore and complete each step—you have 24 hours of pod access, and the lab content will remain available even after your pod expires for future reference. In this lab, you will:

- **Master Workflow Creation:** Learn how to build seamless workflows tailored to customer needs, including routing based on preferences and leveraging real-time data.
- **Optimize Routing Logic:** Implement advanced routing capabilities, such as callback handling, last agent routing and using Global Variables to facilitate routing logic .
- **Invoking Flow API:** Advance decision-making by using the Analyzer database on the fly.

Additionally, you will explore side missions for optional deep dives into:

- Event handling functionality for agent efficiency.
- Creating Post Call survey to measure customer satisfaction
- Changing Contact Center flow logic by using your phone only.

1.1.2 Disclaimer

The lab design and configuration examples provided are for educational purposes. For production design queries, please consult your Cisco representative or an authorized Cisco partner.

Let's get started and discover how **Webex Contact Center Flow Designer** takes customer experiences from good to great!

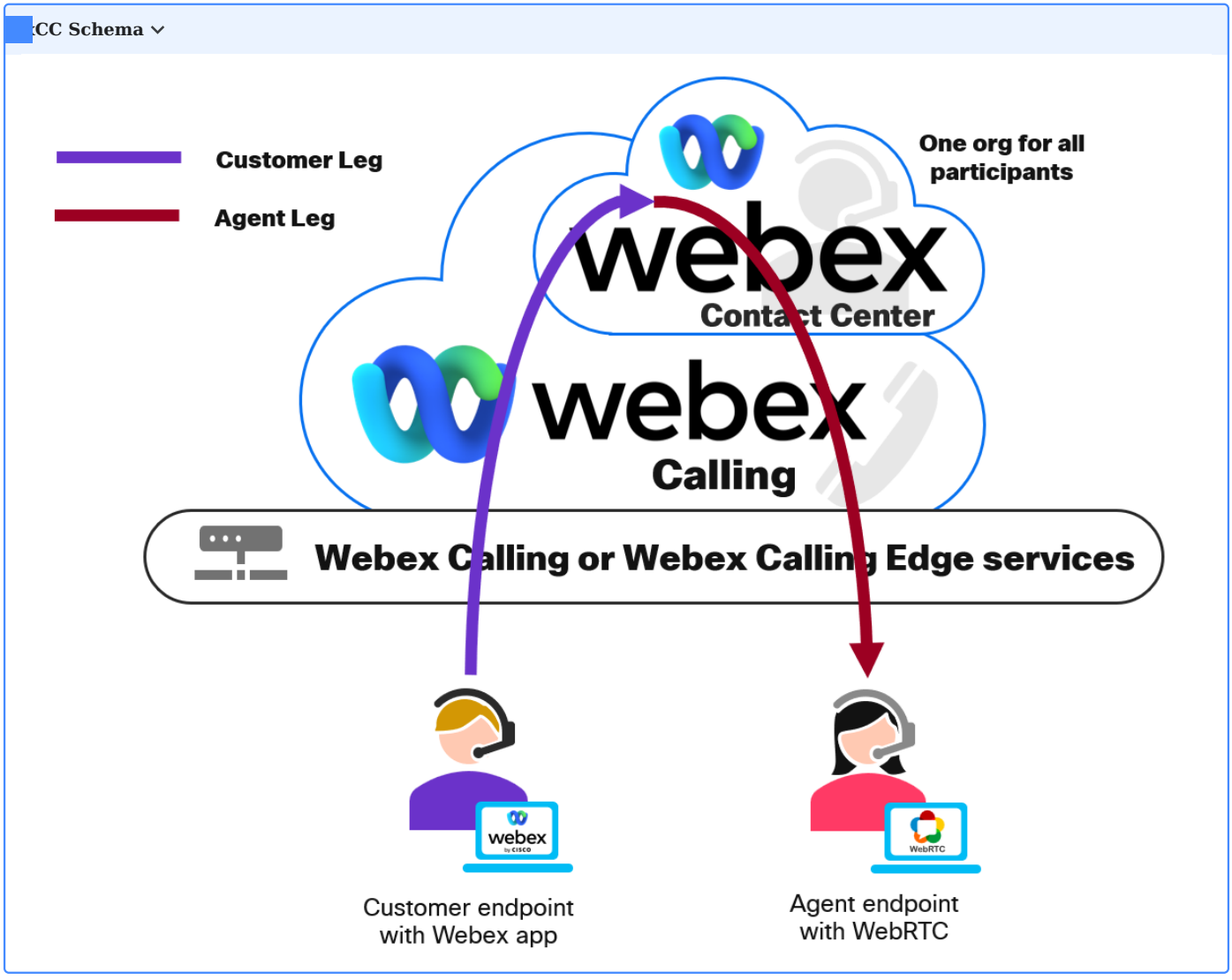
1.2 Getting to know your environment

1.2.1 Learning Objectives

1. Ensure that you have **"POD Your_Attendee_ID.pdf"** file on your desktop with instructions and credentials to access your lab. If you do not, please ask your lab proctor now.
2. Understand your configuration instructions
3. Familiarize yourself how we will use Google Chrome profiles to simulate various scenarios covered in the next labs.

Know before you start

1. We will be using a shared lab tenant for simulations, meaning all attendees will work within the same Webex Contact Center environment. To avoid conflicts, ensure that any entities you configure are tagged with the Attendee ID assigned to you.



2. The majority of the configuration in Control Hub is already set up, allowing you to focus primarily on Flow Design. Of course, there may still be some elements to adjust, but these should be minimal, letting you concentrate on building and refining the flow logic rather than spending time on initial setup.
3. The Agents have been configured for you. You will be performing the rest of the configurations to route voice calls
4. All your configurations should contain your attendee ID so the lab users don't step over each other's configurations

5. Each of you has been provided with the phone number to dial (Entry Point DN), 1 Agent, 1 Supervisor and 1 Admin account.
6. We are going to use built-in Cisco Text to Speech for playing all messages in the lab.
7. Please ask for help when you need it. You can do it by clicking on **"Ask a Question"** or by raising your hand and calling the proctor.

Predefined configuration

Entry Point/Channels: **Your_Attendee_ID_Channel**

Queue: **Your_Attendee_ID_Queue**

Agent: **wxcclabs+agent_IDYour_Attendee_ID@gmail.com**

Supervisor: **wxcclabs+supvr_IDYour_Attendee_ID@gmail.com**

Business Hours: **Your_Attendee_ID_Bussiness_Hours**

Webex App has been pre-installed on your Lab PC

Assigned Inbound Channel Number: **Provided by Lab Instructor**

More pre-configured entities will be mentioned during the lab missions if they have any.

Testing

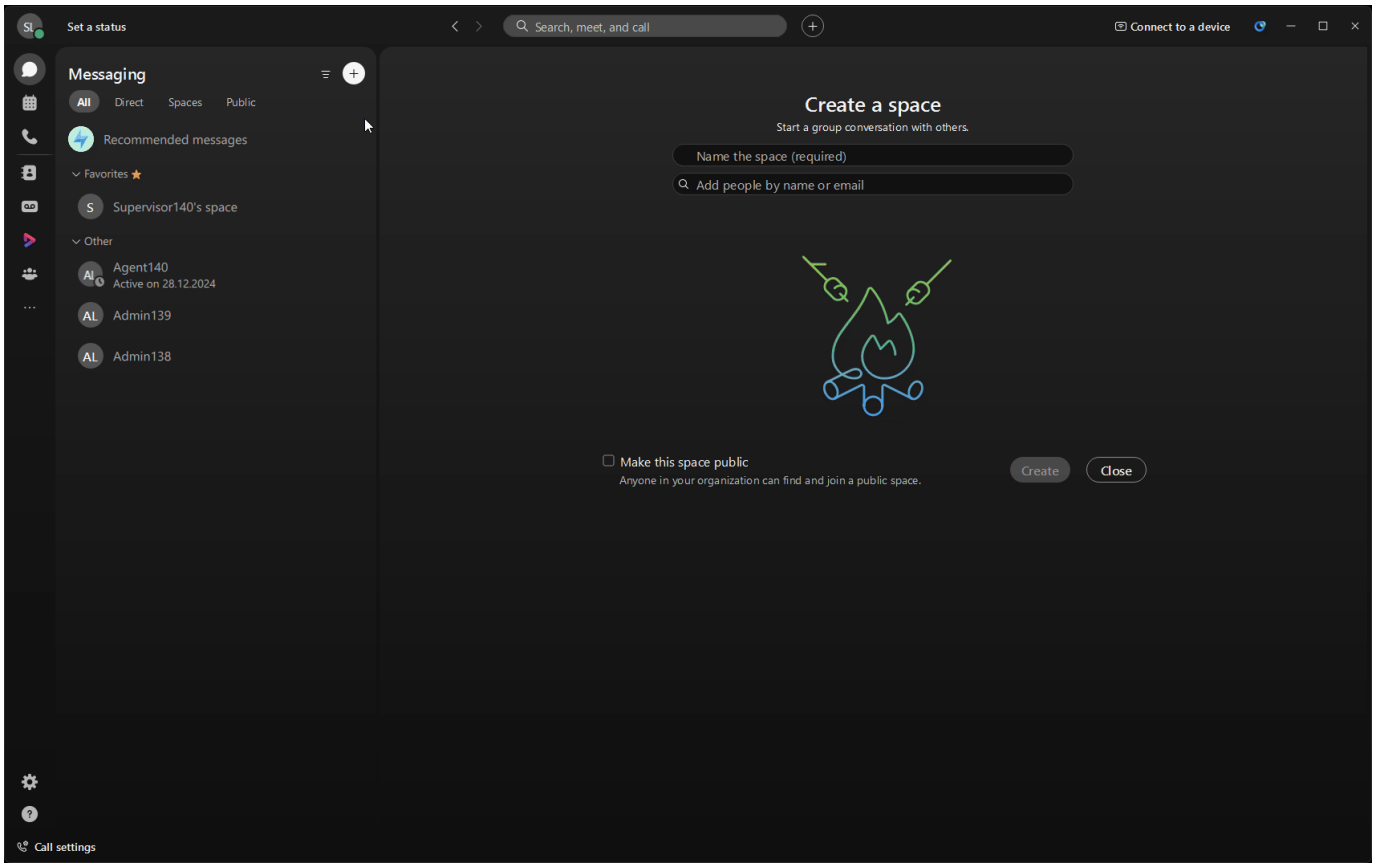
AGENT DESKTOP

Use Agent Desktop application  pre-installed on your workstation to login your agent. In addition, the Desktop profile has been set up so that you don't need to manually select a Team or Telephony option. By default, only the **Desktop** telephony option (also known as WebRTC) is enabled for each agent, allowing call audio to be delivered directly to the Agent Desktop app. Agents do not need to use any other tools, such as softphone, desk phone, or mobile phone to manage calls. Therefore, the Desktop telephony option, along with a pre-configured dedicated Team, will be automatically selected upon sign-in.

CALLING TO CONTACT CENTER

All calls to Webex Contact center should be done from Webex App which has been pre-installed for you as well as pre-logined to it. To make a test just open Webex App and dial the provided Support Number assigned to you.

> **Note**
>
> - By default, the Support Number is not linked to any flow in Webex CC prior to this. Therefore, you will hear an error tone or prompt if you attempt to call that number before finishing corresponding lab task.
> - International dialing is not allowed so you won't be able to dial your cell phones unless you have a US number.



1.3 Choose Your Adventure

1.3.1 Welcome to Your Lab Adventure!

Overview

In this session, we've designed 15 unique labs for you to explore, grouped into 4 distinct tracks to create a focused and engaging learning experience. Each track is crafted to help you develop specific skills.

Here's what you need to know:

Each track offers a different number of labs, guiding you through a cohesive learning journey.

Completing just one track is enough to achieve the session's goals.

Want to push further? Finish 2 tracks to excel, 3 tracks to become an expert, or take on all 4 tracks to earn the ultimate title of Mega Superstar!

Take a moment to review the tracks, choose the ones that excite you, and dive in. This is your adventure—make it unforgettable!

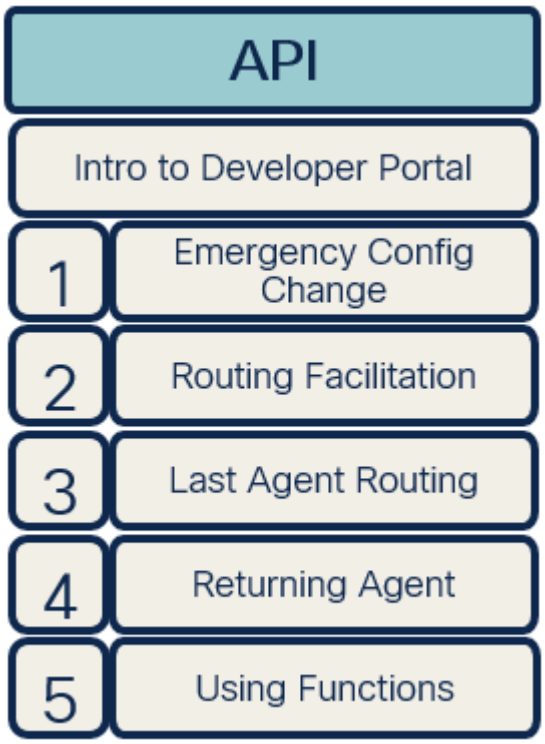
There's no set order—start with any track that interests you most, but we recommend starting with the **Core Track**.

1.3.2 Track 1: Core Track

This track introduces the fundamental features of Flow Designer. Participants will explore flow templates, business hours, and event flows while learning to utilize additional tools like the Debugger and Analyzer.

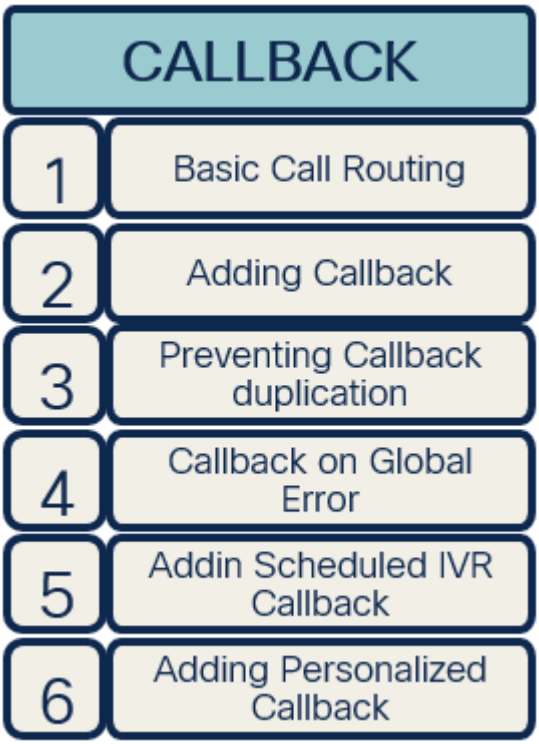
1.3.3 Track 2: API Track

In this track, participants will work on customizing flows using a variety of API requests to interact with different data sources.



1.3.4 Track 3: CallBack Track

The Callback track includes a series of labs focused on various callback scenarios. It begins with basic callback configuration and progresses to advanced GraphQL techniques to eliminate duplicate callbacks.

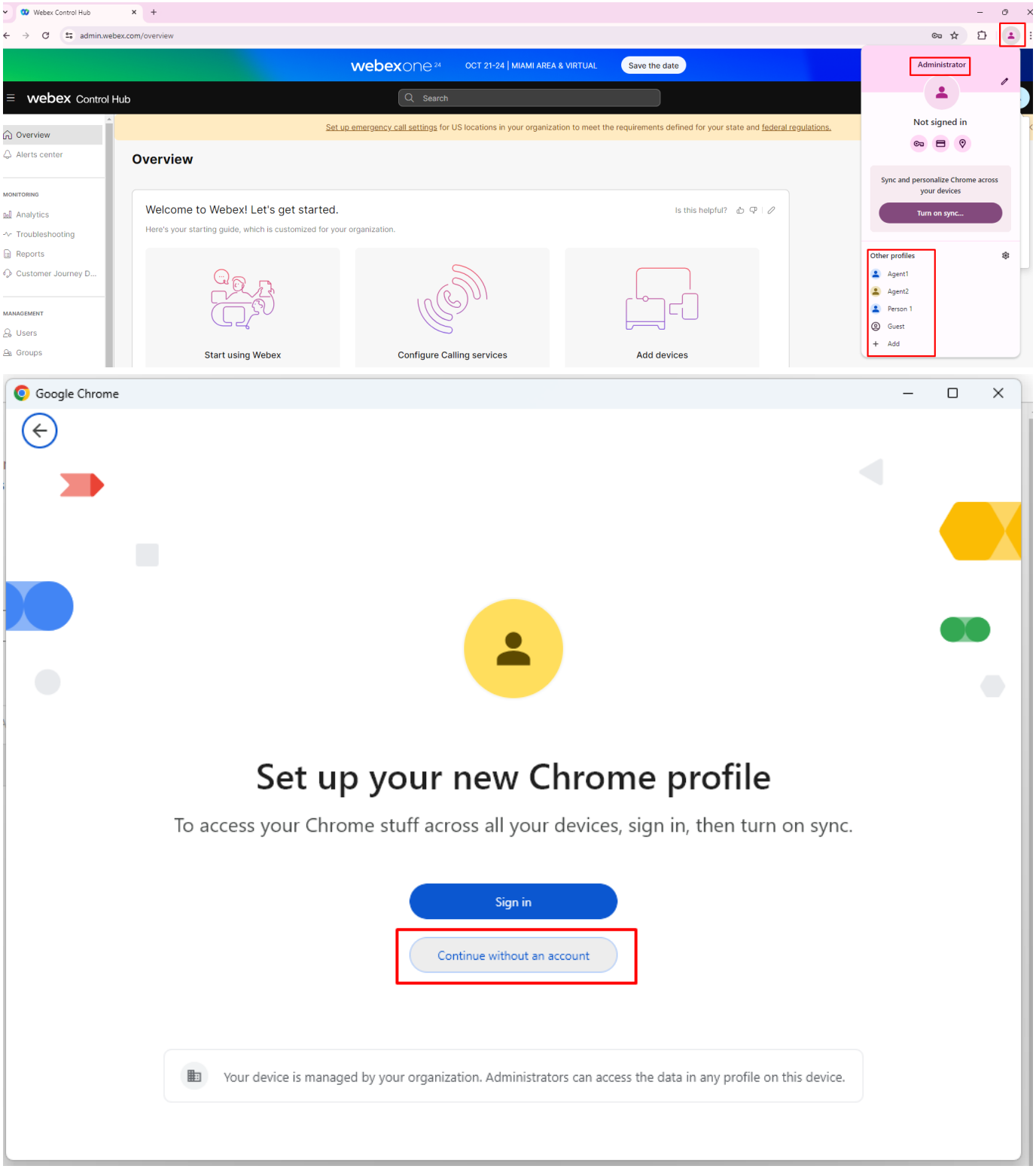


1.4 Chrome and Webex App Setup [Home Work]

Browser Setup

Since we will be using the same Chrome browser for different roles we will use the **Chrome Browser profiles** to allow multiple logins into the different components of the Webex contact center. For the control hub, use the Administrator profile created for you in the Chrome browser. Always offer Chrome to **remember your credentials and password** for this lab. For Agent Desktop, use Agent Desktop Application  pre-insalled on your working station by using Agent profile.

We will create the user profiles below - Admin, Agent.

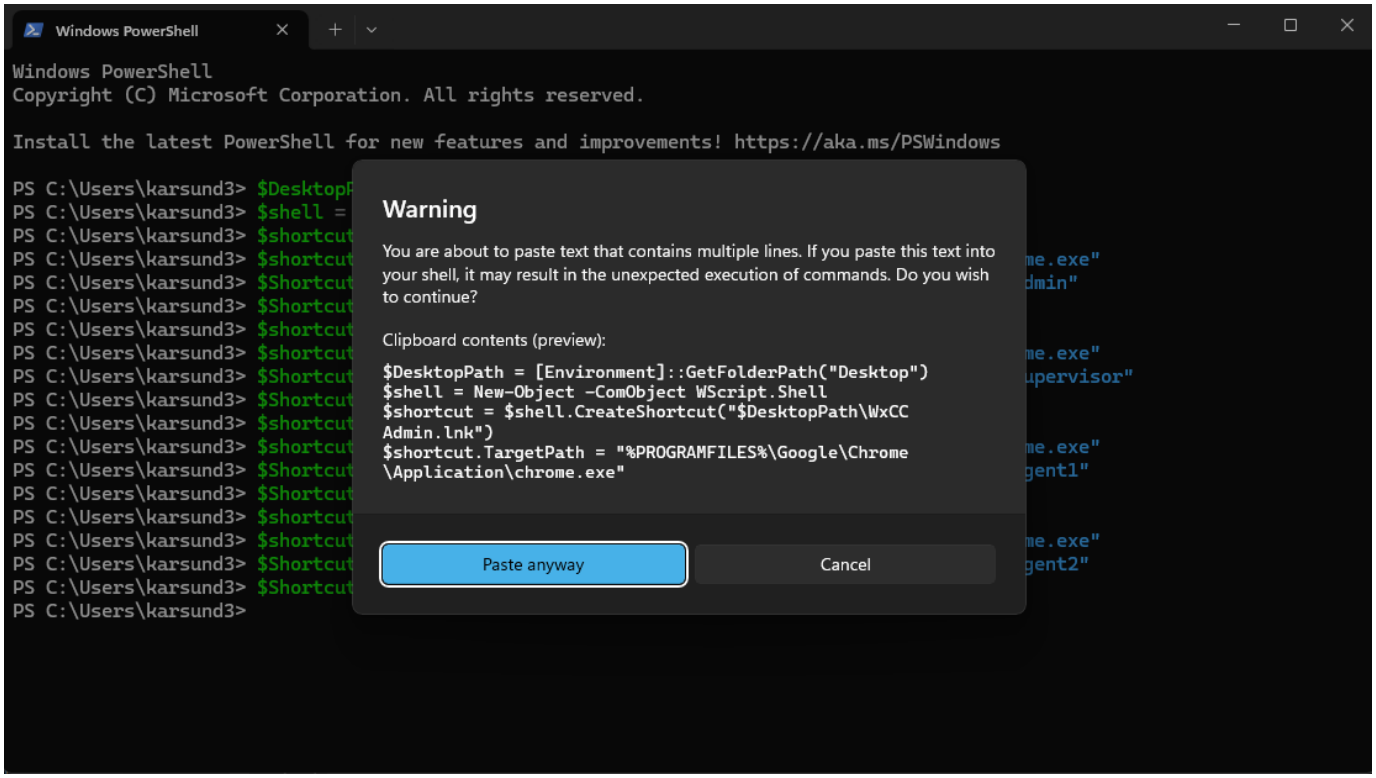


Creating Chrome user profiles

Open the Windows Terminal (Windows key and type **Powershell**). Paste and run the following code. You will see 2 new Chrome shortcut icons on the desktop

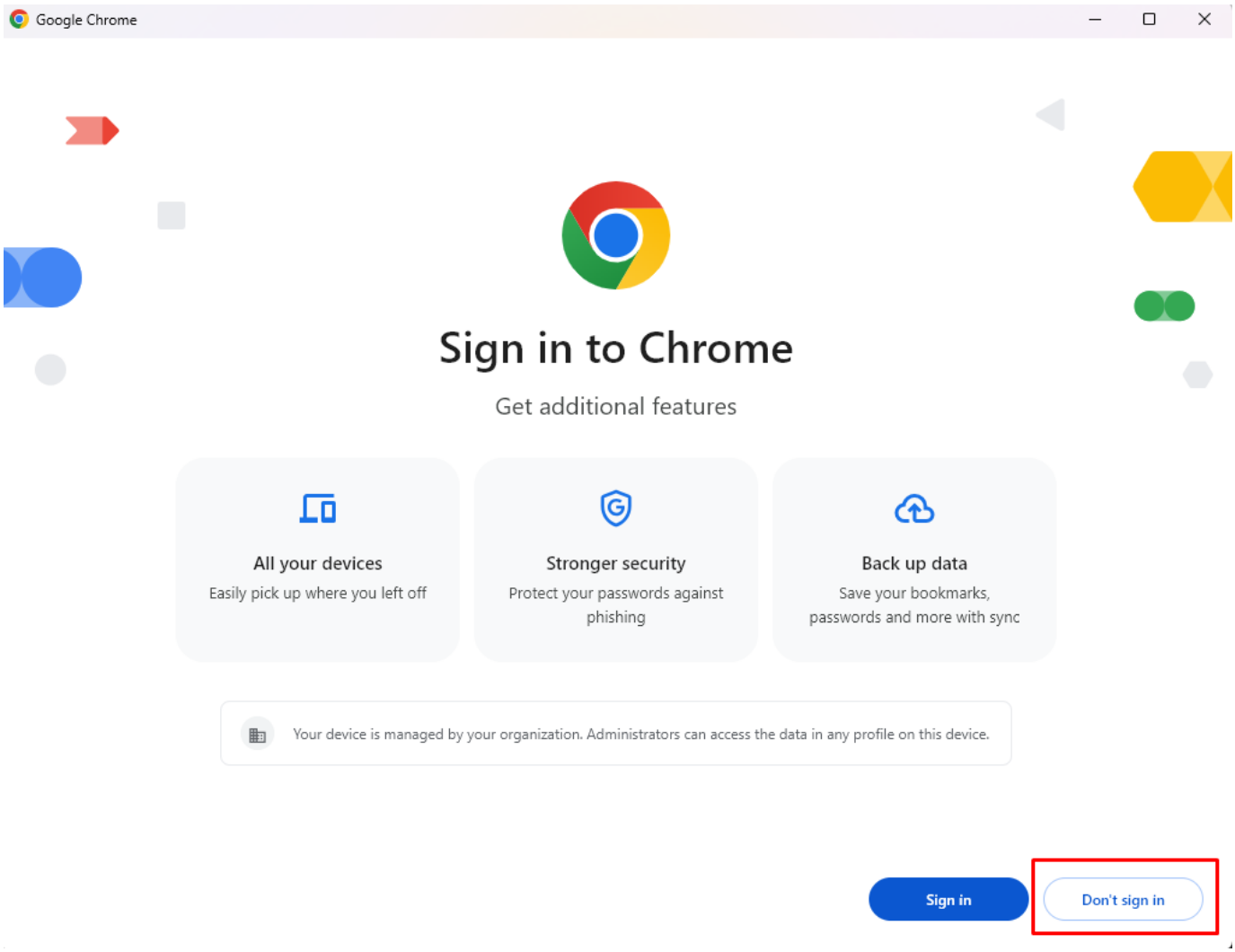
```
$DesktopPath = [Environment]::GetFolderPath("Desktop")
$shell = New-Object -ComObject WScript.Shell
$shortcut = $shell.CreateShortcut("$DesktopPath\WxCC Admin.lnk")
$shortcut.TargetPath = "%PROGRAMFILES%\Google\Chrome\Application\chrome.exe"
```

```
$Shortcut.Arguments = "--user-data-dir=%USERPROFILE%\chromeProfiles\admin"
$Shortcut.Save()
$shortcut = $shell.CreateShortcut("$DesktopPath\WxCC Agent1.lnk")
$shortcut.TargetPath = "%PROGRAMFILES%\Google\Chrome\Application\chrome.exe"
$Shortcut.Arguments = "--user-data-dir=%USERPROFILE%\chromeProfiles\Agent1"
$Shortcut.Save()
```

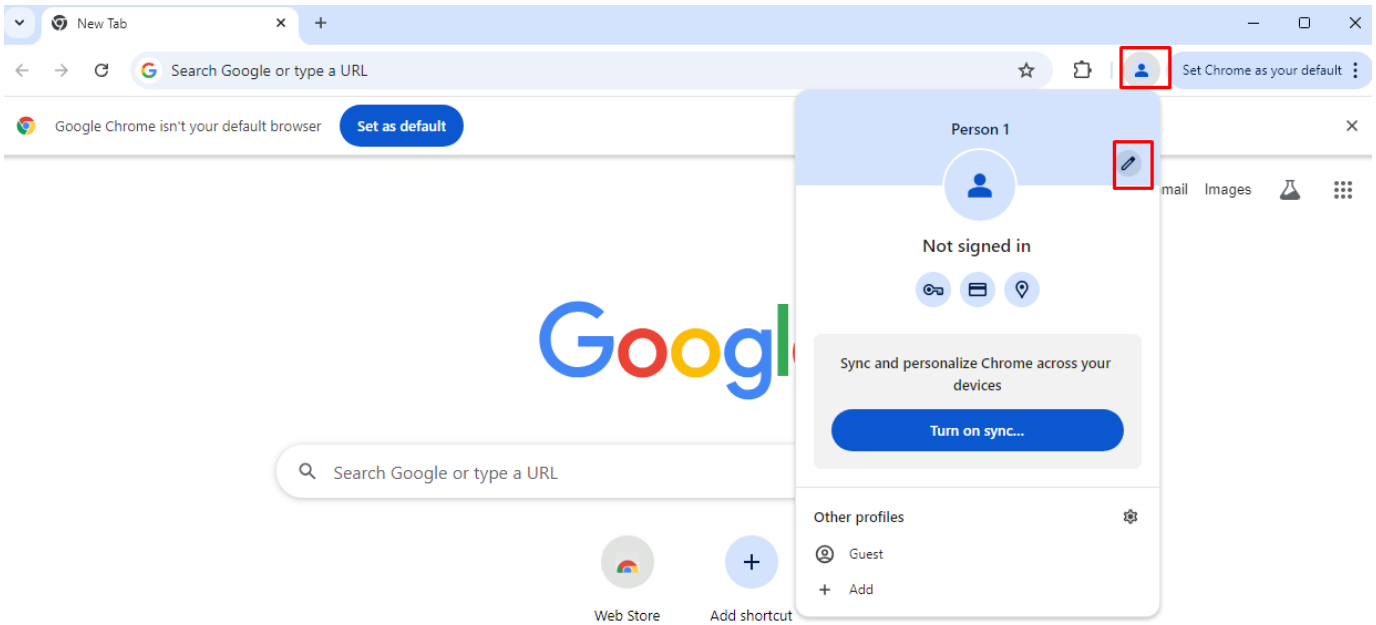


Check the desktop of your lab PC. You should find 2 Chrome shortcuts created - **WxCC Admin** and **WxCC Agent1**

When you click on the links



You can customize each profile to be easily identifiable with a name and/or icon of your choice

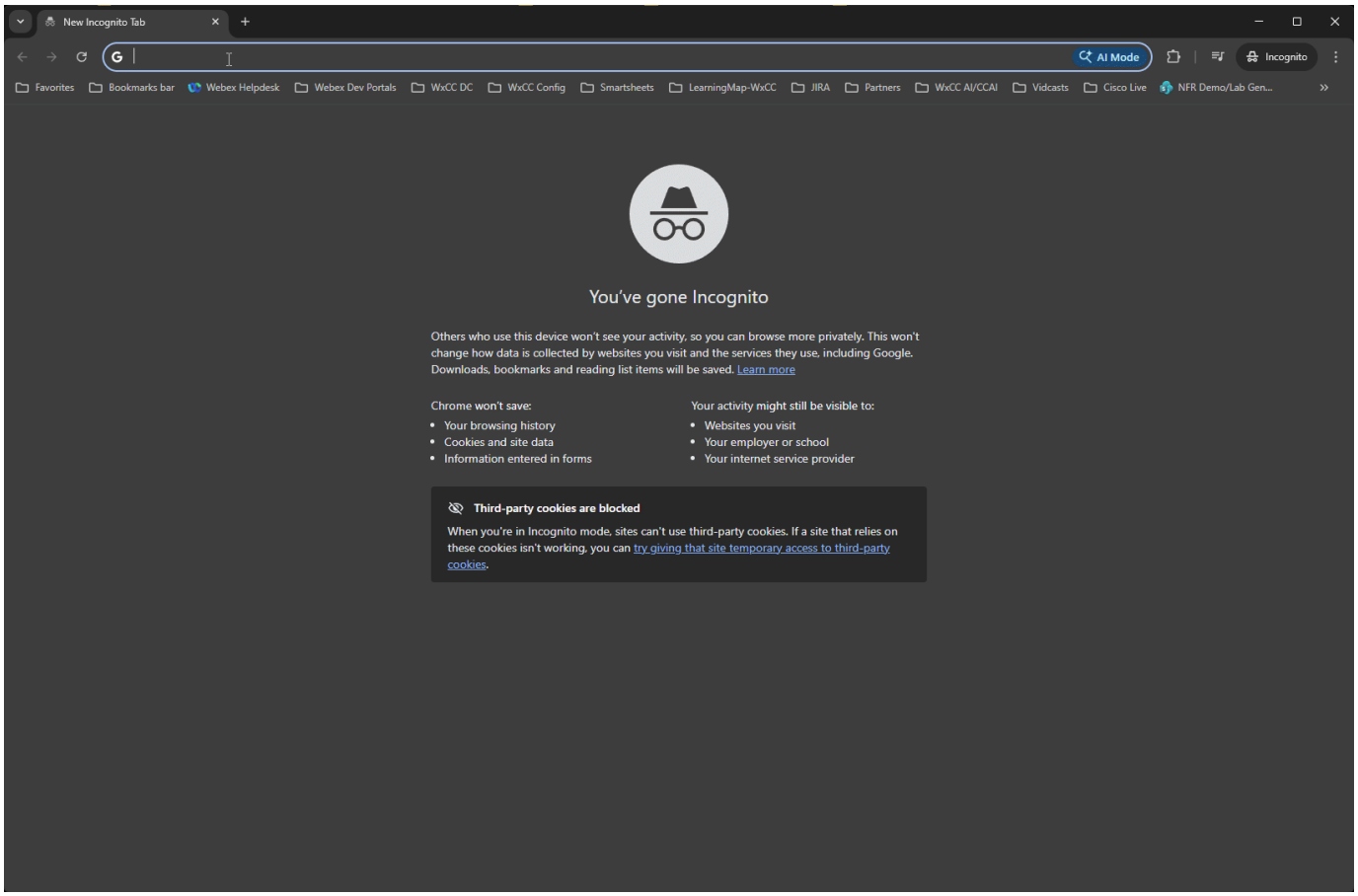


We will use the **Admin** profile first in the next section.

1.5 LAB ROOM SETUP

1.5.1 Steps to setup each laptop in the room.

1. Download and save on desktop a txt file for current pod from **User Credentials**. Example **ID *[id]* - LTRCCT-2966 Credentials**, where ***[id]*** is the POD ID of current desktop.
2. On each laptop access lab-assistant from the browser. Enter pin code 191518
3. Click on **Documentation** button to open Lab Guide.
4. In the Lab Guide go to Getting Started -> **LAB ROOM SETUP**



Creating Chrome user profiles

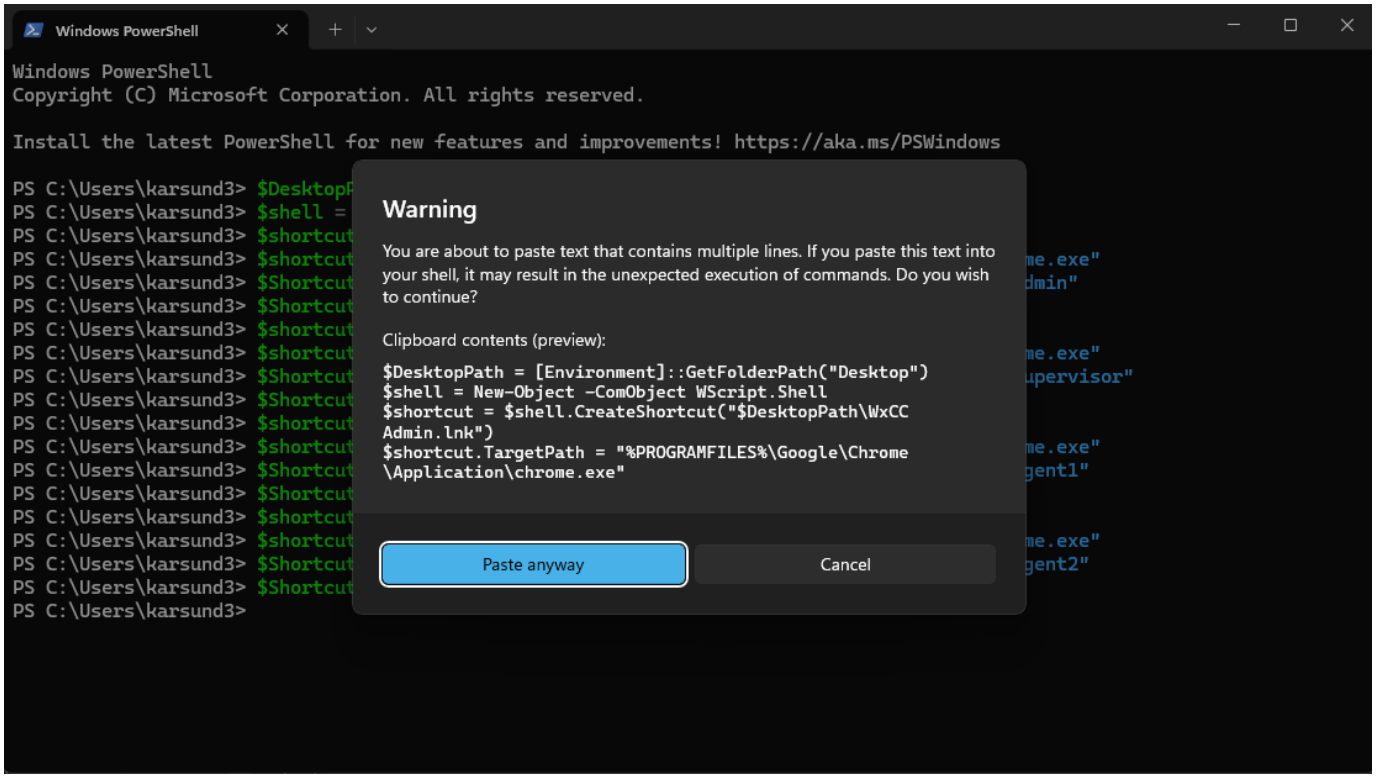
Open the Windows Terminal (Windows key and type **Powershell**). Paste and run the following code. After running the script you should find 2 Chrome shortcuts created - **WxCC Admin** and **WxCC Agent1**.

```
# ================================
# Helper function to create Chrome profile, shortcut, profile name, and bookmarks
# ================================
function New-ChromeProfile {
    param(
        [string]$ProfileFolderName,   # Folder under chromeProfiles (e.g. 'admin', 'agent')
        [string]$ProfileDisplayName,  # Name Chrome will show (e.g. 'Admin', 'Agent')
        [string]$ShortcutName,        # Shortcut file name
        [array]$Bookmarks             # Array of bookmark objects @{ name=""; url="" }
    )

    $DesktopPath = [Environment]::GetFolderPath("Desktop")
    $ProfilePath = "$env:USERPROFILE\chromeProfiles\$ProfileFolderName"
    $DefaultProfilePath = "$ProfilePath\Default"
    $PreferencesFile = "$DefaultProfilePath\Preferences"
    $BookmarksFile = "$DefaultProfilePath\Bookmarks"
```

```
    # --- Ensure profile directory exists ---
    if (!(Test-Path $DefaultProfilePath)) {
        New-Item -ItemType Directory -Path $DefaultProfilePath -Force | Out-Null
    }

    # --- Create desktop shortcut ---
    $shell = New-Object -ComObject WScript.Shell
    $shortcut = $shell.CreateShortcut("$DesktopPath\$ShortcutName")
    $shortcut.TargetPath = "$env:ProgramFiles\Google\Chrome\Application\chrome.exe"
    $shortcut.Arguments = "--user-data-dir=$ProfilePath"
    $shortcut.Save()

    # --- Update Preferences file (set profile name) ---
    if (!(Test-Path $PreferencesFile)) {
        @"
{
  "profile": {
    "name": "$ProfileDisplayName"
  }
}
"@ | Set-Content -Path $PreferencesFile -Encoding UTF8
    } else {
        $json = Get-Content $PreferencesFile -Raw | ConvertFrom-Json
        if (-not $json.profile) {
            $json | Add-Member -MemberType NoteProperty -Name 'profile' -Value @{ name = $ProfileDisplayName }
        } else {
            $json.profile.name = $ProfileDisplayName
        }
        $json | ConvertTo-Json -Depth 10 | Set-Content $PreferencesFile -Encoding UTF8
    }

    # --- Build bookmark children array dynamically ---
    $children = @()
    $id = 1
    foreach ($bm in $Bookmarks) {
        $children += @{
            date_added = "13200000000000000"
            id = "$id"
            name = $bm.name
            type = "url"
            url  = $bm.url
        }
        $id++
    }

    # --- Complete bookmark JSON ---
    $BookmarkJson = @{
        roots = @{
            bookmark_bar = @{
                children = $children
                date_added = "13200000000000000"
                date_modified = "13200000000000000"
                id = "100"
                name = "Bookmarks bar"
                type = "folder"
            }
            other = @{
                children = @()
                id = "101"
                name = "Other bookmarks"
                type = "folder"
            }
            synced = @{
                children = @()
                id = "102"
                name = "Synced bookmarks"
                type = "folder"
            }
        }
        version = 1
    }

    $BookmarkJson | ConvertTo-Json -Depth 10 | Set-Content -Path $BookmarksFile -Encoding UTF8

    Write-Host "Profile '$ProfileDisplayName' created with $($Bookmarks.Count) bookmark(s)."
}

# ================================
# Create Admin profile (2 bookmarks)
# ================================
New-ChromeProfile `
    -ProfileFolderName "admin" `
    -ProfileDisplayName "Admin" `
    -ShortcutName "WxCC Admin.lnk" `
    -Bookmarks @(
        @{ name = "LTRCCT-2966"; url = "https://webexcc-sa.github.io/LTRCCT-2296/" },
        @{ name = "Webex Control Hub"; url = "https://admin.webex.com/" }
    )

# ================================
# Create Agent profile (1 bookmark)
# ================================
New-ChromeProfile `
```

```
    -ProfileFolderName "agent" `
    -ProfileDisplayName "Agent" `
    -ShortcutName "WxCC Agent.lnk" `
    -Bookmarks @(
        @{ name = "Agent Desktop"; url = "https://desktop.wxcc-us1.cisco.com/" }
    )
```

- Click on **Past anyway** if asks.



- Check the desktop of your lab PC. You should find 2 Chrome shortcuts created - **WxCC Admin** and **WxCC Agent1**

Configure apps for Admin, Agent and Supervisor

1. Open **WxCC Admin**, check bookmarked URLs. Login to **Control Hub** by using Admin account from TXT file you downloaded from the **BOX**.
2. Save the password in the browser.
3. Open **WxCC Agent**, check bookmarked URL. Login to **Agent Desktop** by using Agent account from TXT file you downloaded from the **BOX**.
4. Save the password in the browser.
5. While logged in, click on **Install App** to install **Webex Desktop** Application.
6. After installation, add **Webex Desktop App** link to desktop.
7. Open **Webex App**, and login by using Supervisor account.
8. Make a test call to a DN from TXT file you downloaded from the **BOX**.. Make sure you hear **Welcome to Cisco Live** message.

2. CORE TRACK

2.1 Lab Guide

2.1.1 Mission 1: Basic Call Routing (Flow Template, TTS, Language)

> **Note**
>
> The input in the images that follow are only examples. They do not reflect the input you need to use in the lab exercises. In some cases, the input in the images may not follow the same attendee or pod ID from previous images. They are for representation only

Story

Imagine calling a contact center, seeking quick, personalized help. Behind the scenes, a flow smoothly routes your call based on your needs.

CALL FLOW OVERVIEW

1. A new call enters the flow. *(This initiates the interaction and triggers the defined call-handling process.)*
2. The flow determines the caller's language preference and plays a preconfigured Text-to-Speech (TTS) prompt. *(This ensures the caller receives information in their preferred language.)*
3. The call is routed to the appropriate queue. *(This directs the caller to the right team on the flow logic.)*

MISSION DETAILS

Your mission is to:

1. Configure key flow elements for efficient caller journeys.
2. Explore Flow Templates to streamline flow creation.
3. Set up routing with conditions, such as language preference.
4. Gain the skills to design flows for real-world scenarios.

y Flow Templates? [Optional]

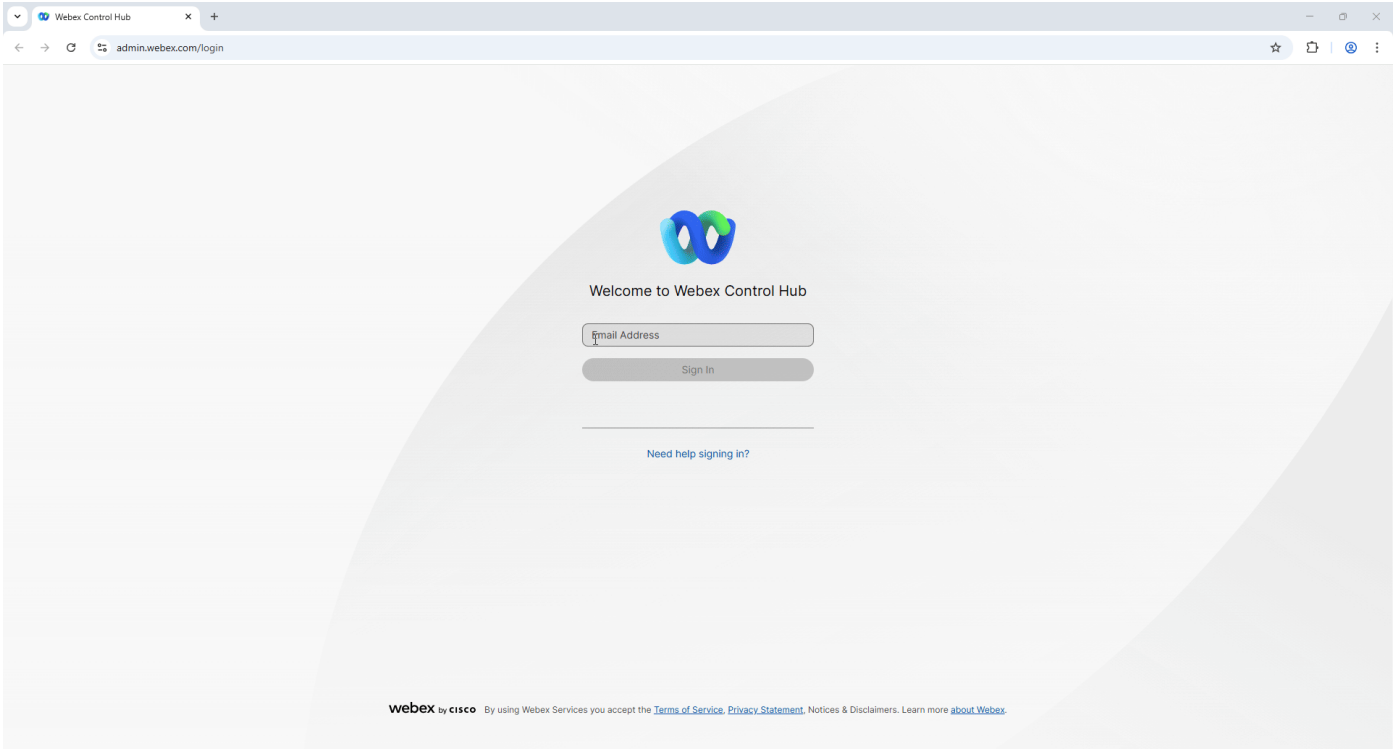
Flow Templates in Webex Contact Center are an essential feature for flow developers, offering a range of benefits that streamline the development process and enhance the efficiency and consistency of flow creation. Here's what they bring to the table:

- **Consistency and Standards**: Templates ensure that flows adhere to best practices, creating consistent experiences across multiple projects.
- **Time Savings**: Prebuilt structures reduce the need to start from scratch, enabling faster setup and allowing more focus on customization.
- **Reduced Errors**: Using tested templates lowers the risk of mistakes and minimizes troubleshooting.
- **Easy Onboarding**: New developers or partners can learn quickly by using templates as guides.
- **Scalability**: Templates allow developers to replicate and adapt solutions efficiently across different flows or deployments.
- **Innovation**: Developers can spend more time on unique features and integrations rather than reconfiguring basics.

Flow Templates are designed to empower developers, speed up the development lifecycle, and maintain high-quality standards across flows, making them a core asset in Webex Contact Center flow design.

BUILD

1. Login into Webex Control Hub by using your Admin profile. Your login will be of the format **wxcclabs+admin_IDYour_Attendee_ID@gmail.com**. You will see another login screen with Webex logo on it where you may need to enter the email address again and the password provided to you.



Note

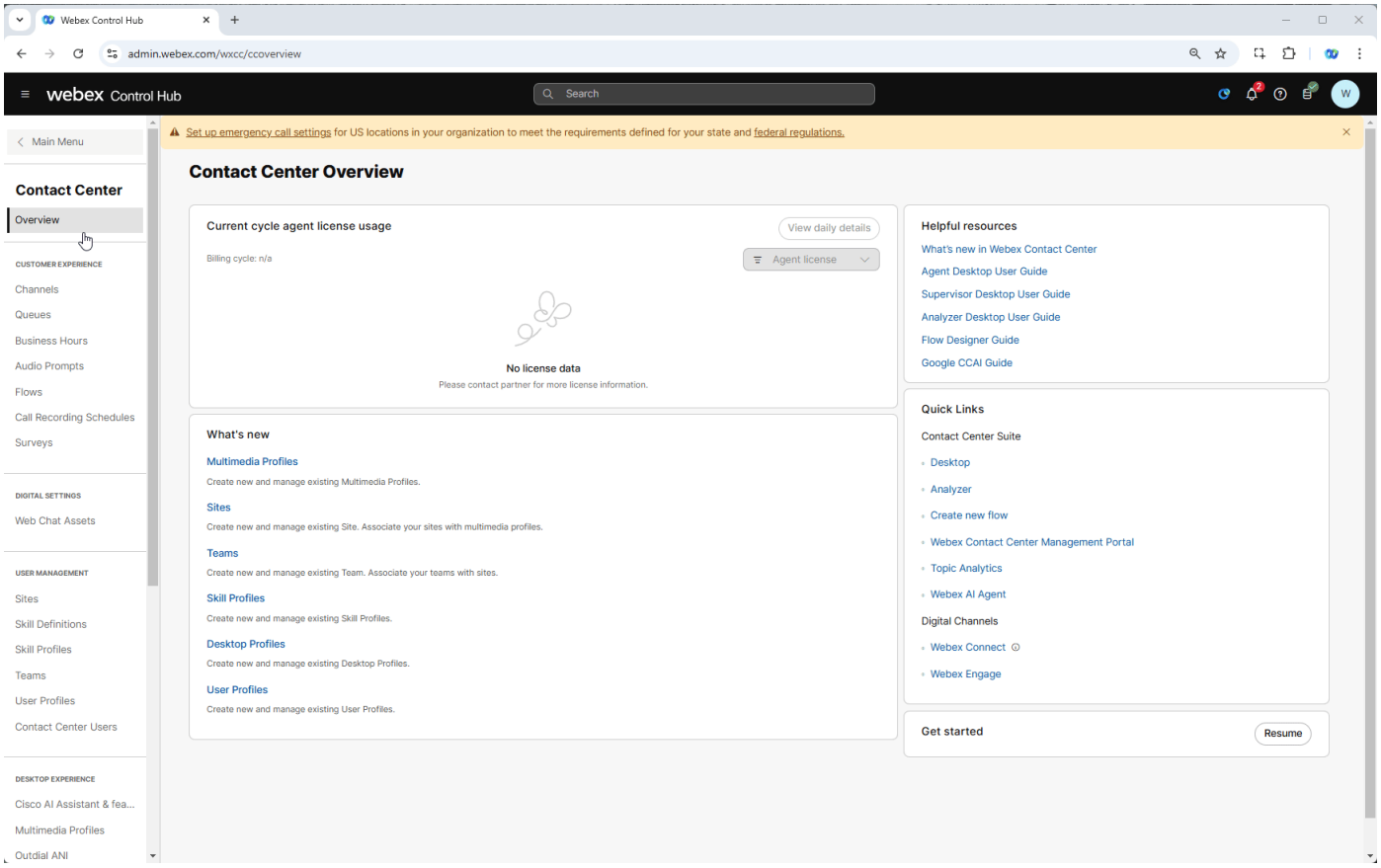
Remember to take up the offer from Chrome to save your password

2. This is the **Administration interface** for Webex Contact Center and is also known as the Control Hub. Look for the contact center option in the left pane under **SERVICES – Contact Center** and click on it.
3. Navigate to **Flows**, click on **Manage Flows** dropdown list and select **Create Flows**.

4. **Create a new flow** tab will be opened. Navigate to **Flow Templates**.
5. Choose **Simple Inbound Call to Queue** template and click **Next**.

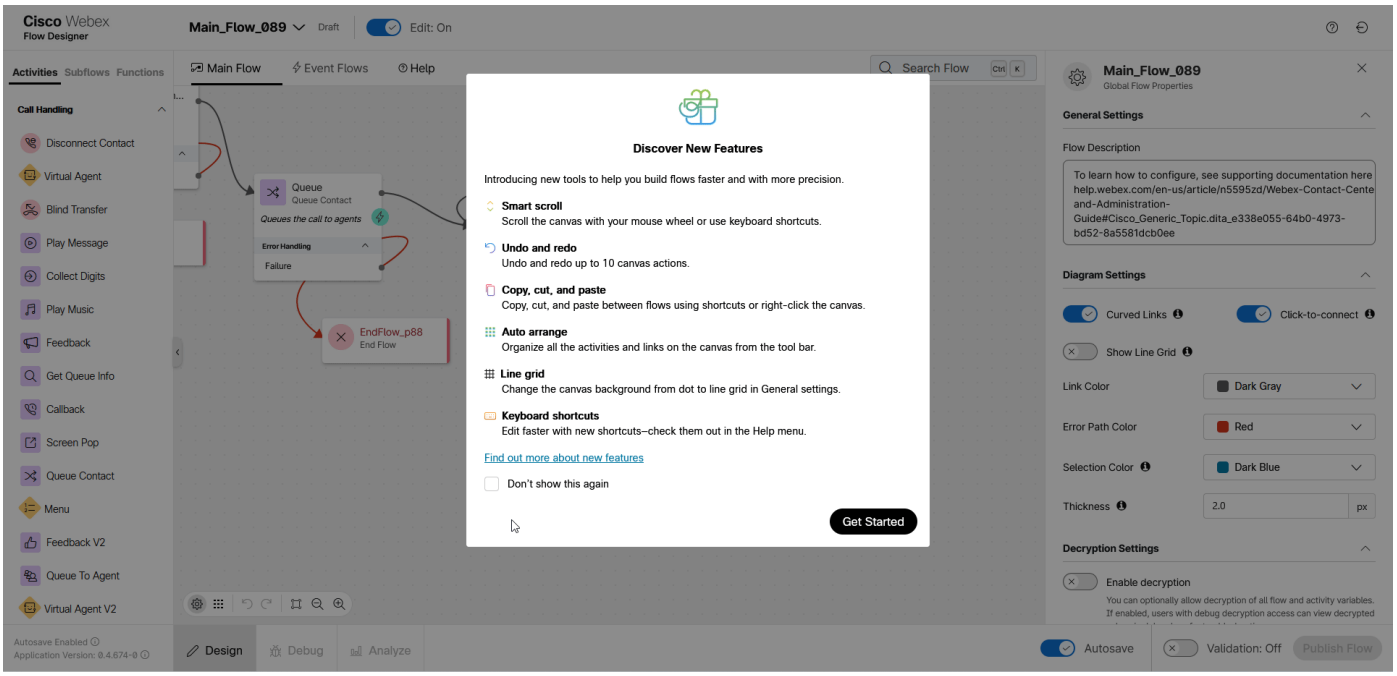
> **Note**
>
> You can press **View Details** link under the template name to observe flow structure and read flow description before proceeding with the template.

6. Name your flow as **Main_Flow_Your_Attendee_ID** Then click on **Create Flow**.



7. If **Discover New Features** popup window appears in front of flow canvas, set the checkbox **Don't show this again** at the bottom and press **Get Started** button to close it and and go to the flow designer.

> **Note**
>
> **Edit** should be set to **On** when you create new flow, but if not switch it from **Edit: Off** mode to **Edit: On** at the top of the page.

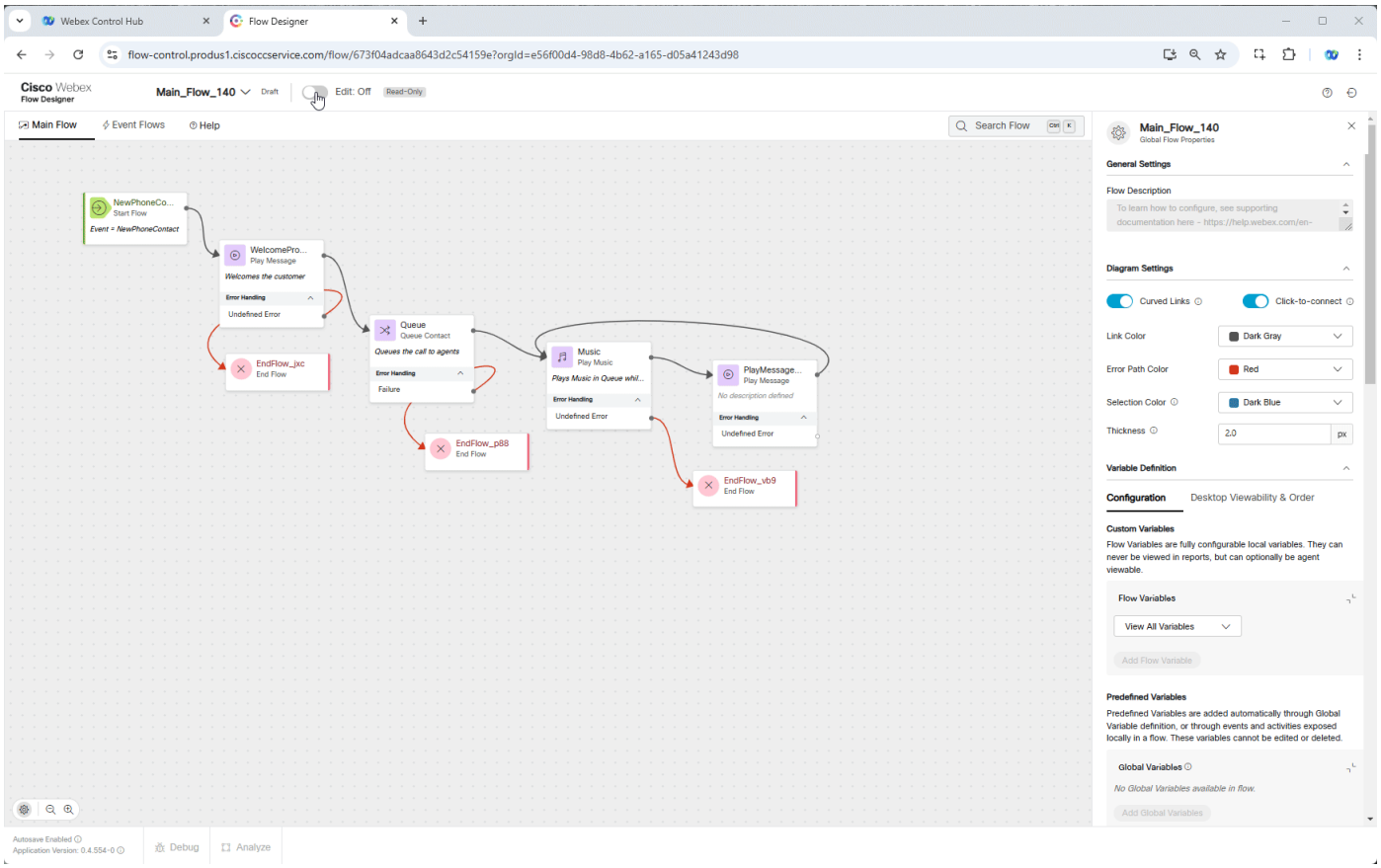


8. Select **Play Message** node with label **WelcomePrompt** and on the node settings modify **Text-to-Speech Message** to any greetings you like. This message will be the first message you hear while calling to your script.
9. Select **Queue** node. On the **General settings** keep Static Queue checked and select queue **Your_Attendee_ID_Queue** from the dropdown list

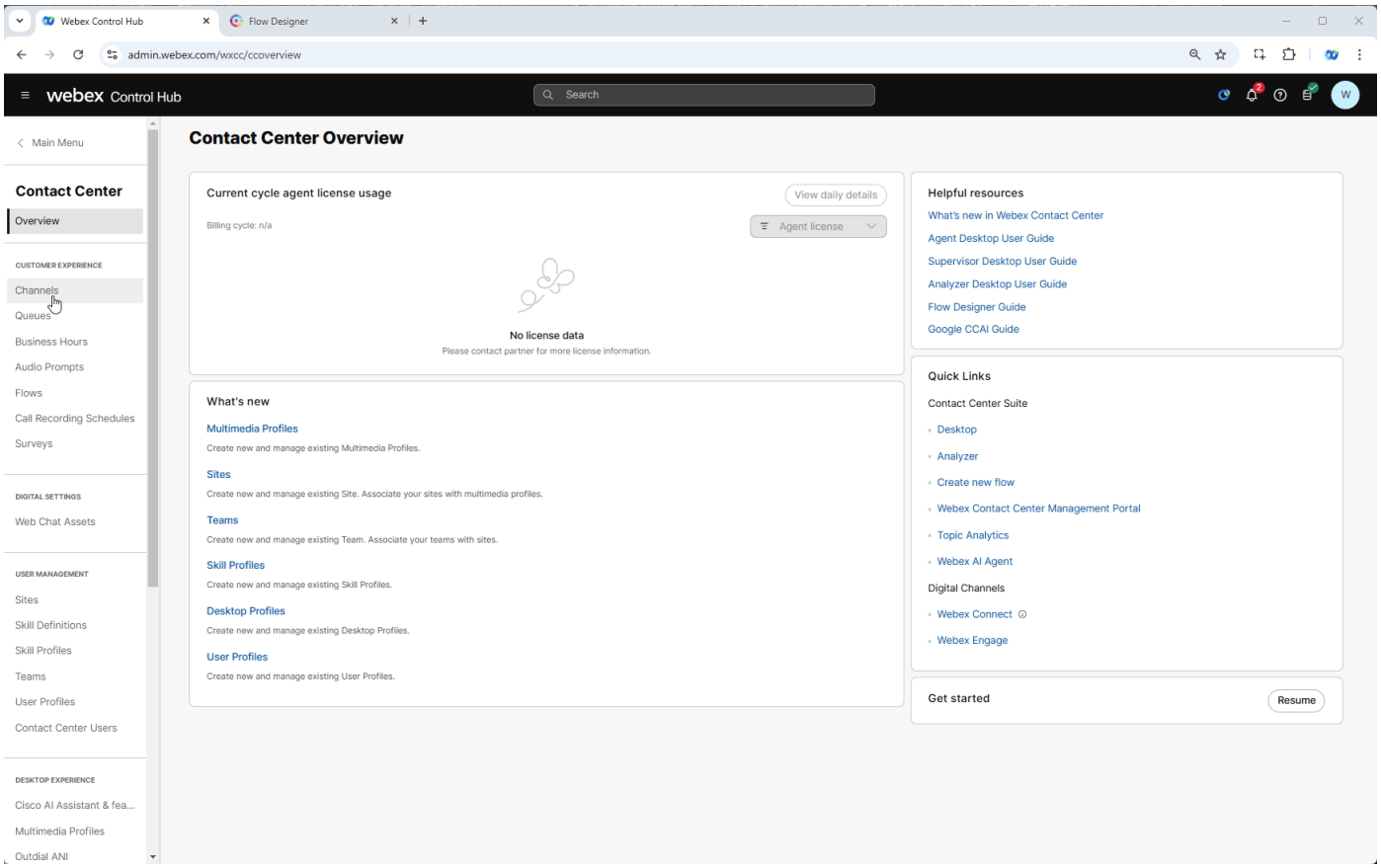
> **Note**
>
> As mentioned in **Getting Started**, all queues have been preconfigured so you don't need to change them at current step.

10. [Optional] Select **Play Message** node (the one which goes after **Queue** and **Play Music** nodes) and on the **Node settings** modify **Text-to-Speech Message** to any message you like. This message will be played while the caller is waiting in the queue.
11. Validate and publish the flow:

- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.



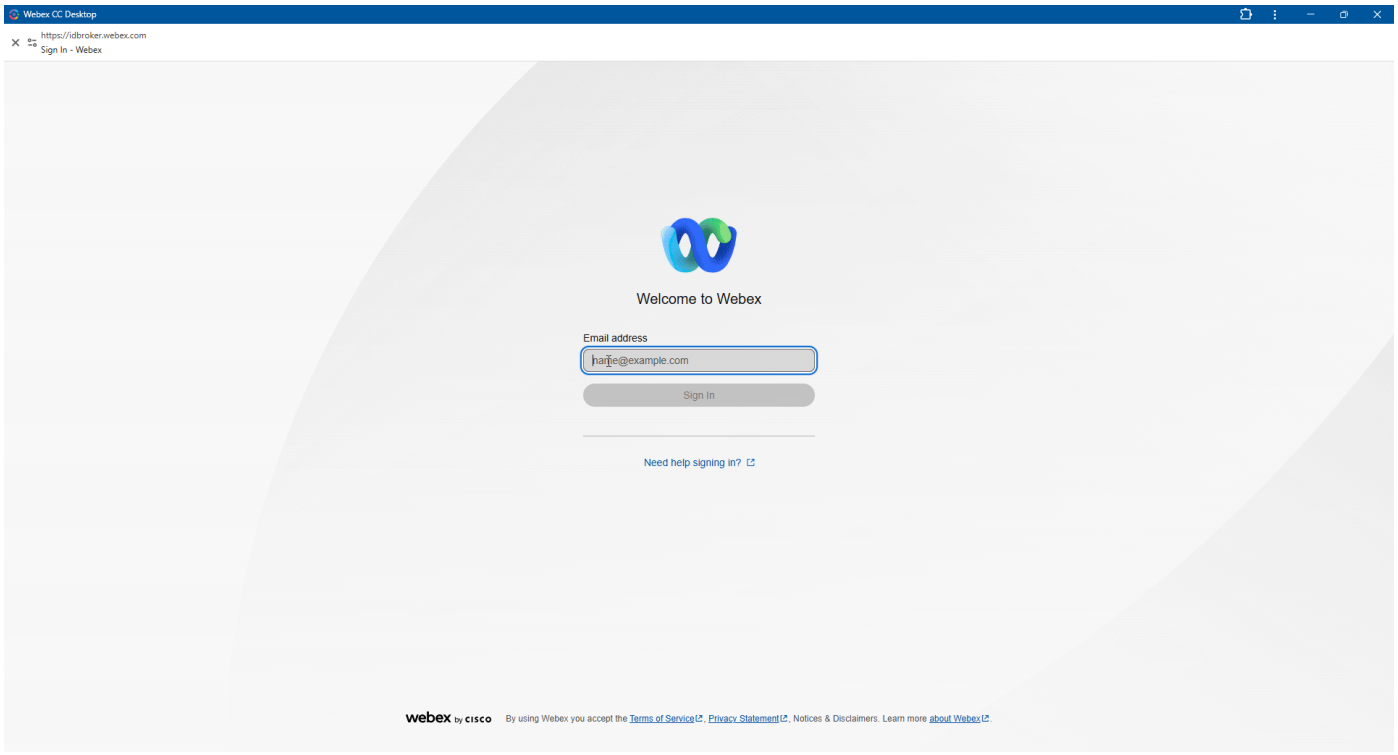
12. Return to Control Hub to assign the Flow to your **Channel (Entry Point)**. Go to **Channels**, search for your channel **Your_Attendee_ID_Channel**
13. Click on **Your_Attendee_ID_Channel**
14. In **Entry Point** settings section change the following, then click **Save** button:
    - Routing flow: **Main_Flow_Your_Attendee_ID**
    - Music on hold: **defaultmusic_on_hold.wav**
    - Version label: **Latest**



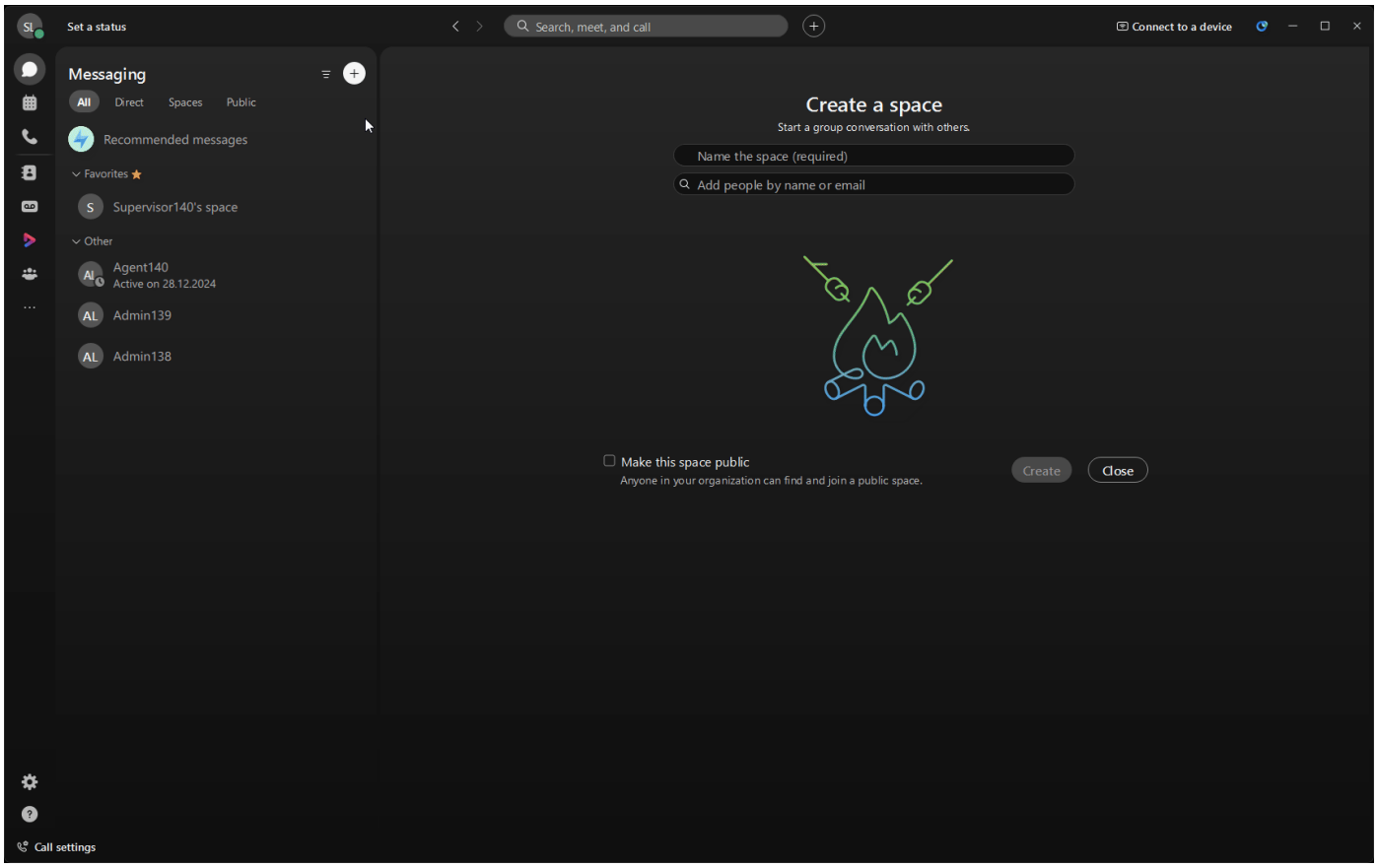
CHECKPOINT TEST

1. Launch Webex CC Desktop application and login with agent credentials you have been provided **wxcclabs+agent_IDYour_Attendee_ID@gmail.com** You will see another login screen with Webex icon on it where you may need to enter the email address again and the password provided to you.
2. Allow notifications and browser to access Microphone by clicking **Allow** in the relevant pop-up prompts.
3. Make your agent **Available** and you're ready to make a call.

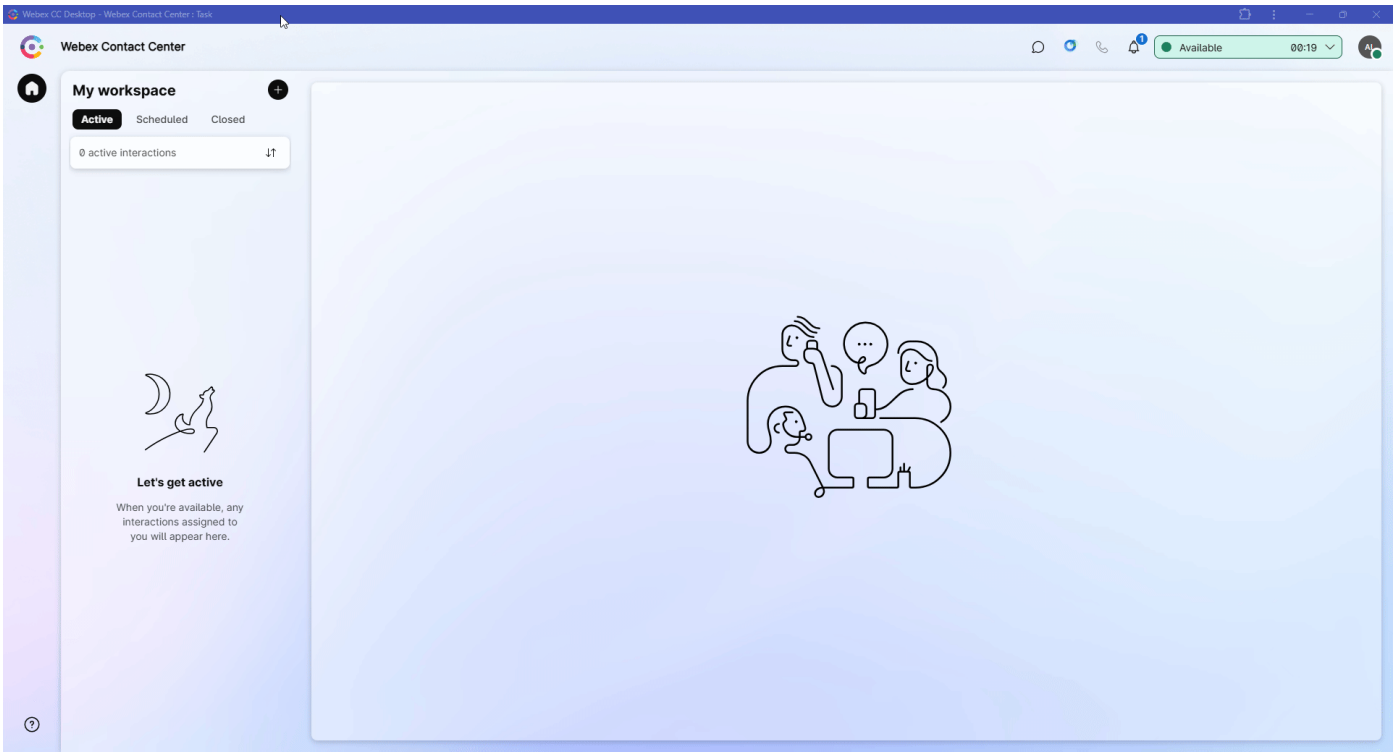
> **Note**
>
> This is the only time during the lab when you need to log in to the Webex CC Desktop application. It has been configured to keep your agent logged into the application for the entire duration of the lab. If, for any reason, you are logged out manually or due to a network error, please log in again as explained above.



4. Open your Webex App and dial the Support Number provided to you, which is configured in your **Your_Attendee_ID_Channel** configuration.



5. Ensure that you hear a welcome prompt configured in your flow and receive an incoming call in your agent desktop. Answer the call in the agent desktop and confirm that it has been successfully established. Then end the call, select any value from the **Wrap Up Reasons** dropdown list on the agent desktop, and press the **Submit Wrap Up** button.



Enhance you flow with Language and Voice Name

ENHANCEMENT DETAILS

Your mission is to:

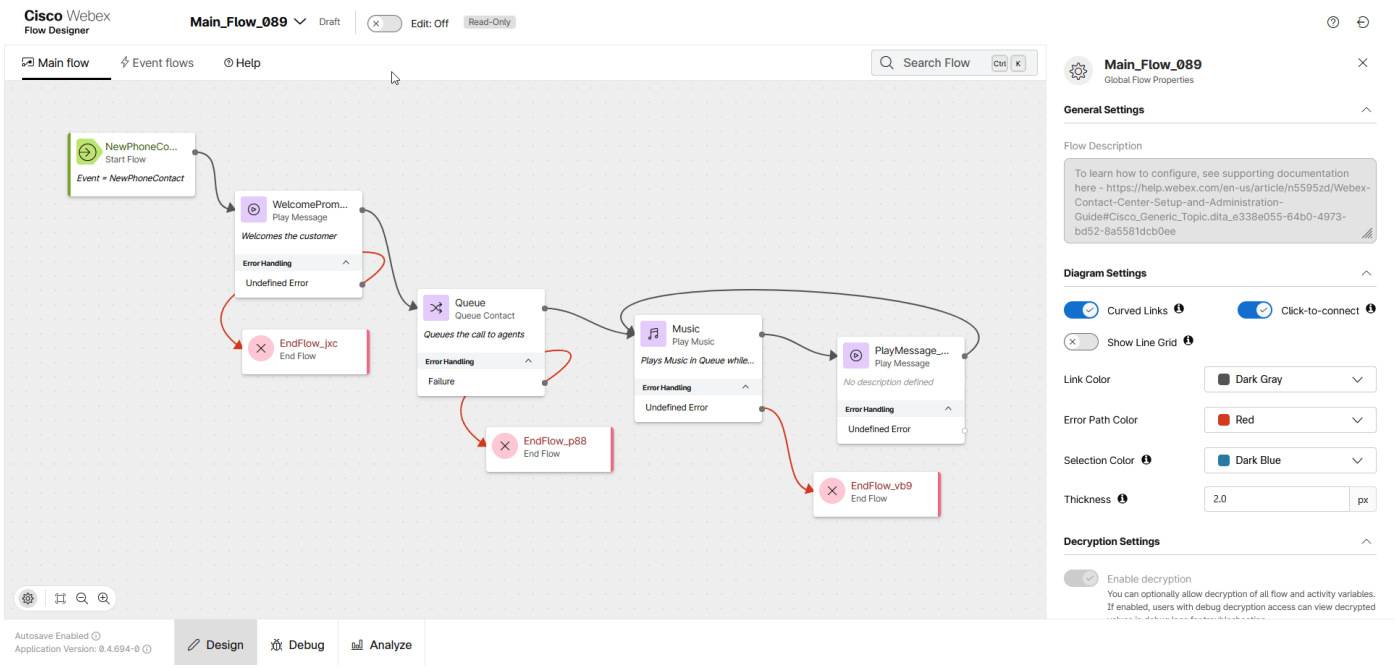
- Use the same flow created in the previous section.
- Add Language and Voice Name global variables to the flow.
- Modify Text-To-Speech (TTS) settings to use Australian English language and Male Voice.
- Place a call to verify and validate the speech functionality.

> **xt-to-Speech (TTS) in Webex Contact Center [Optional]**
>
> All supported TTS Languages and Voice Names can be found in the table here: Text-to-Speech-(TTS)-in-Webex-Contact-Center

BUILD

1. Open your flow **Main_Flow_Your_Attendee_ID**. Make sure **Edit** toggle is **ON**.
2. Navigate to the **Global Flow Properties** panel on the right-hand side.

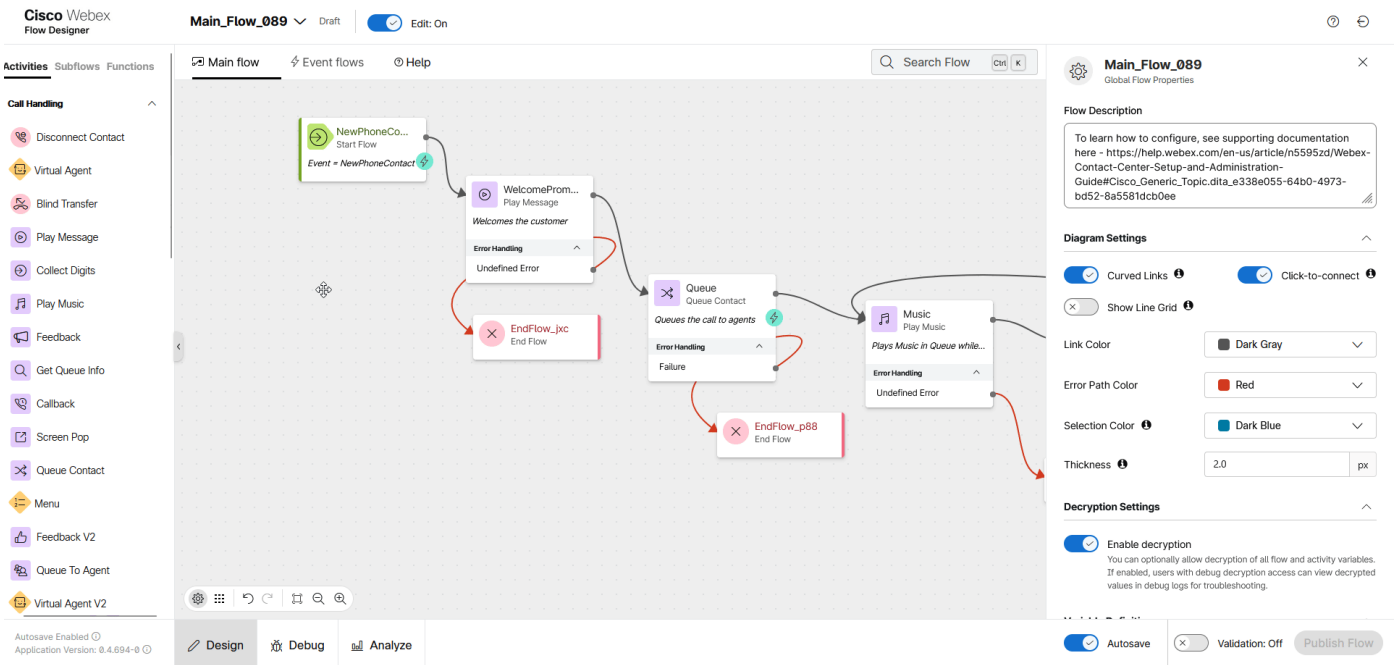
- Scroll down and Locate the **Predefined Variables** section.
- Click on the **Add Global Variables** button.
- Search for **Global_Language** variable and click on **Add** button.
- Click on the **Add Global Variables** button one more time.
- Search for **Global_VoiceName** variable and click on **Add** button.



3. Add a **Set Variable** node from the activity library on the left with following configuration:

- Navigate to the **Variable Settings** section.
- Click on the **Variable** drop-down list and select **Global_Language**.
- Set Value: **en-AU**.
- Click on the **+ Add New** button under the variable you have just added to add another one.
- Click on the **Variable** drop-down list and select **Global_VoiceName**.
- Set Value: **en-AU-Chris**.

- Return to the flow canvas and delete connection between **NewPhoneContact** and **WelcomePrompt** nodes.
- Connect **NewPhoneContact** to **Set Variable**.
- Connect **Set Variable** to **WelcomePrompt**.

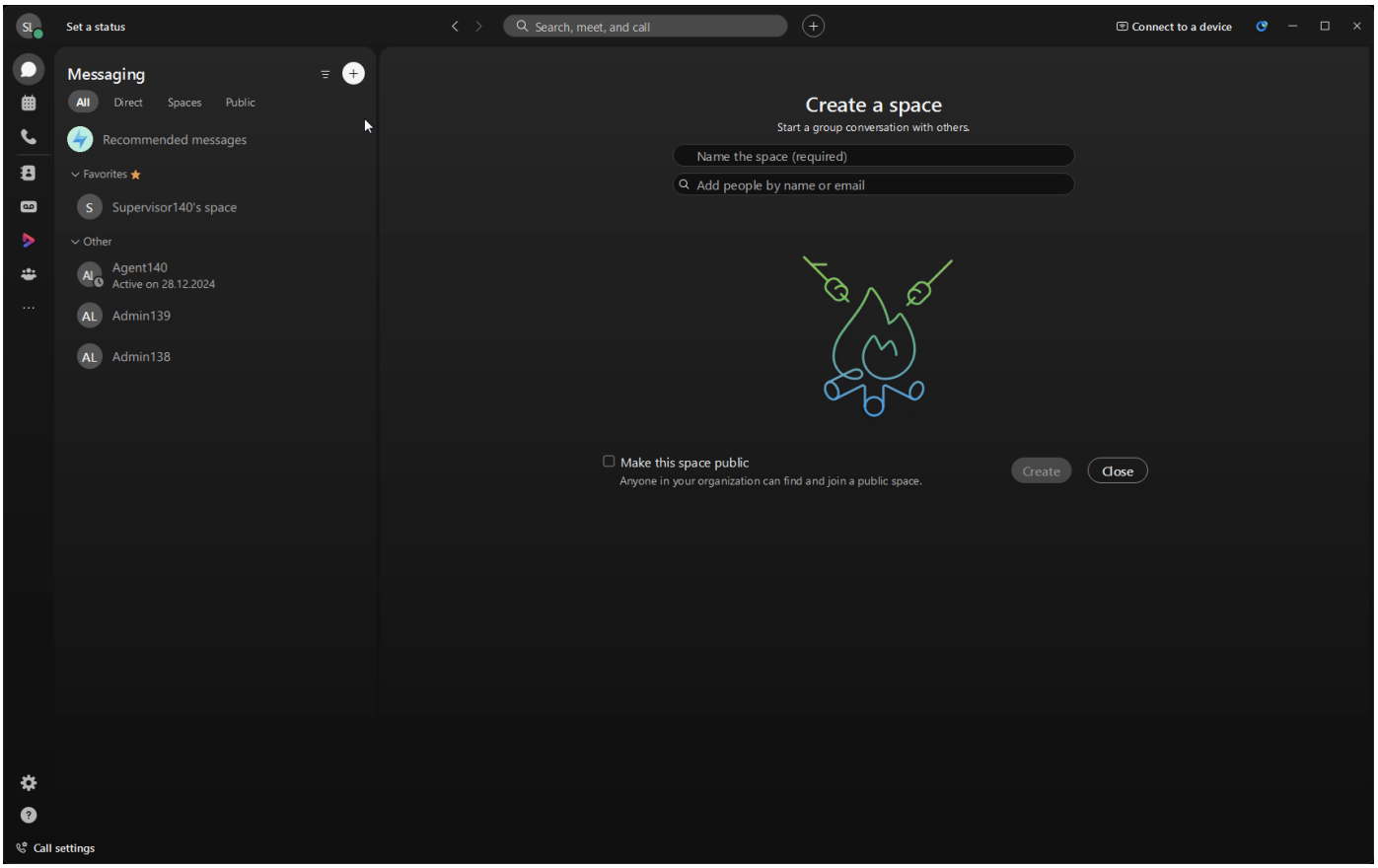


4. Validate and publish the flow:

- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.

TESTING

1. Open your Webex Desktop and make your agent is **Available**, and you're ready to make a call.
2. Open your Webex App and dial the Support Number provided to you, which is configured in your **Your_Attendee_ID_Channel** configuration.



3. Listen to the welcome prompt to confirm the TTS language and voice have changed.

Congratulations, you have succesfully completed Basci Call Routing mission! 🎉

2.1.2 Mission 2: Skill-Based Routing

Story

Skill-Based Routing (SBR) intelligently matches customer contacts with agents who have the specific expertise needed (e.g., language, product knowledge, service level), using configurable skill requirements and relaxation rules to ensure the best fit, which improves resolution and satisfaction by avoiding simple "first-available" routing.

This mission focuses on enhancing customer satisfaction by enabling Skill-Based Routing in Webex Contact Center call flow to direct the call to the qualified agent. At the same time, we will look into the effectiveness of skill relaxation rules to reduce the waiting time for callers if all more qualified agents are busy.

Call Flow Overview

1. A new call enters the flow.
2. The flow executes the logic and places the call into a skill-based queue, applying the skill requirements and relaxation rules defined for the queue.
3. Once the call is placed in the queue, the following logic applies:

- If there is an available agent with matching skills, the call is routed to that person.
- If there are no available agents with matching skills, the call waits in the queue until the skill-relaxation rule is applied.
- If the skill-relaxation rule is applied, it lowers the skill requirements for the call, and Webex Contact Center starts looking for an agent using the updated requirements.

Mission Details

Your mission is to:

1. Switch to a skill-based queue and define the skill requirement for it.
2. Configure a skill relaxation rule to lower skill requirement after a time interval.
3. Make test calls and verify the agent selection logic before and after skill relaxation.

> **Note**
>
> The Skill Definition and Skill Profile are preconfigured for you and linked to your agent account. Your task is to focus on configuring the flow and ensuring the agent is selected properly depending on current skill requirements.

PRECONFIGURED ENTITIES

- Skill Definition: **Webex CC Lab Expert**.
- Skill Profile: **Webex CC Lab Skill Profile** with **Webex CC Lab Expert** skill set as 5.
- Skill Profile is associated with your agent account **wxcclabs+agent_IDYour_Attendee_ID@gmail.com**.

Part 1 - Add Skill-Based Queue

BUILD

1. Switch to the Flow Designer tab with your **Main_Flow_Your_Attendee_ID**, make sure **Edit** toggle is **ON**.

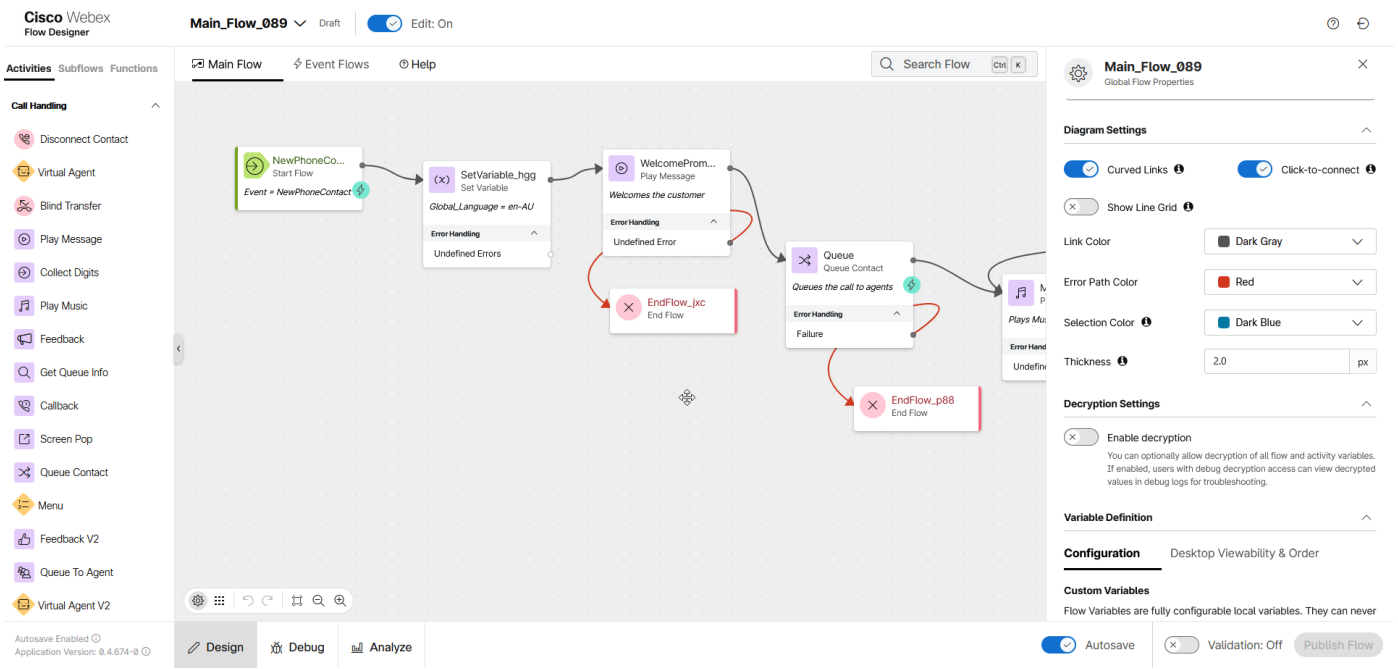
> **Note**
>
> If you closed browser tab with Flow Designer before, switch to Webex Control Hub. Look for the contact center option in the left pane under **SERVICES – Contact Center** and click on it. Then navigate to **Flows**, search for your flow **Main_Flow_Your_Attendee_ID** and click on it to open it in the flow designer.

2. Click on **Queue** node and select **Your_Attendee_ID_SBR_Queue** Then scroll down to the **Skill Requirements** section and configure the following fields in the **Skill Requirement** block:

- Select the skill name: **Static**
- Skill (drop-down list): **Webex CC Lab Expert**

- Select the condition: **Static**
- Condition (drop-down list): **>=**

- Select the value: **Static**
- Value (drop-down list): **3**



3. Validate and publish the flow:

- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.

CHECKPOINT TEST

> **Note**
>
> The skill requirement we have set for **Your_Attendee_ID_SBR_Queue** is applied to every call that lands in that queue.
>
> Since we set the requirement **Webex CC Lab Expert >= 3** and the current value of **Webex CC Lab Expert** skill configured under the skill profile **Webex CC Lab Skill Profile** assigned to your agent account is **5**, the agent meets skill requirement for the call and should be selected to take it.
>
> Let's test this.

1. Your Agent desktop session should be still active but if not, use Webex CC Desktop application and login with agent credentials you have been provided **wxcclabs+agent_IDYour_Attendee_ID@gmail.com** and become **Available**
2. Make a test call to the Support Number, ensure the call is assigned to your Agent and answer it.
3. Finish the call.

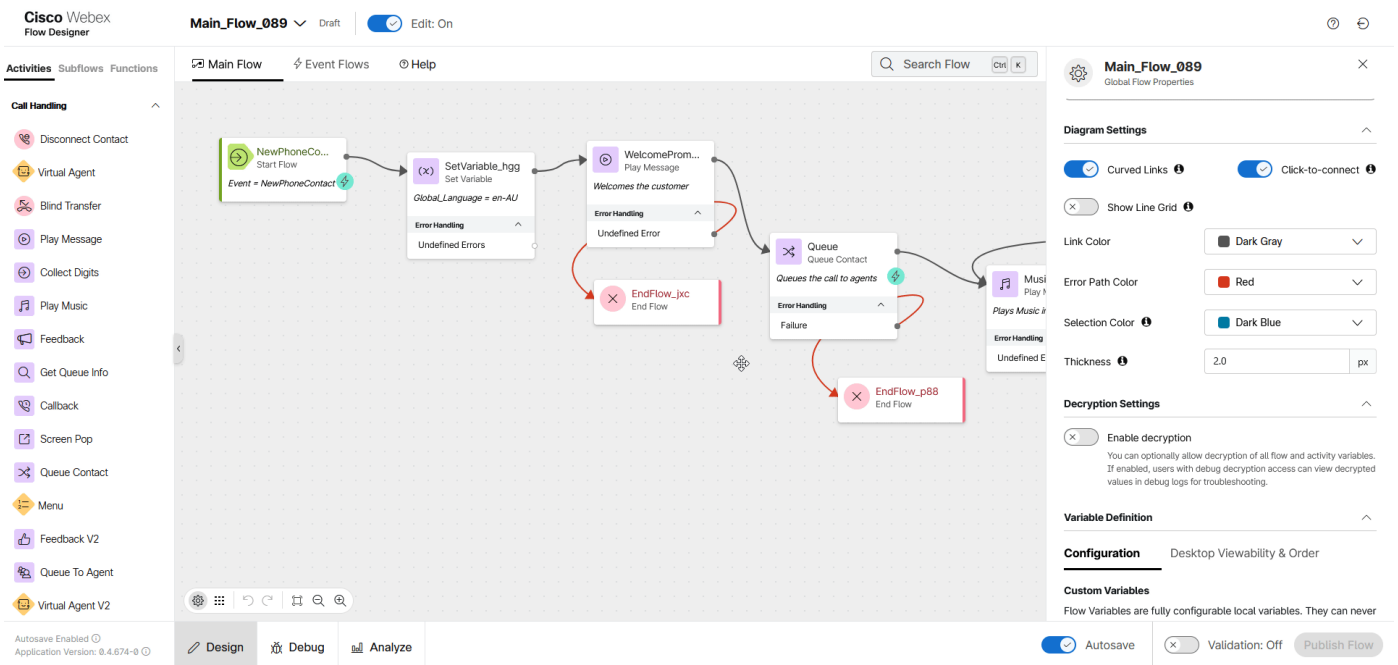
Part 2 - Raise Skill Requirement

> **Note**
>
> Let's raise skill requirement to see what happens if there are no agents with the appropriate skill level available.

BUILD

1. Switch to your flow and make the following changes to raise skill requirement:

- Click on **Queue** node - you should see **Your_Attendee_ID_SBR_Queue** skill-based queue configured there.
- Scroll down to the **Skill Requirements** section and update the required value for **Webex CC Lab Expert** skill from 3 to 7. The full requirement should be **Webex CC Lab Expert >= 7**:



2. Validate and publish the flow:

- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.

CHECKPOINT TEST #2

> **Note**
>
> Since we have raised the requirement **Webex CC Lab Expert >= 7** and the current value of **Webex CC Lab Expert** skill for your agent is still **5**, the agent no longer meets the skill requirement for the call and **should not** be selected to pick it up.
>
> Let's test this.

1. Open your Webex CC Desktop application  and make sure your agent is in the **Available** state.
2. Make a test call to the Support Number. Ensure the caller hears the welcome prompt, then the call is placed in a queue and the caller hears music in queue.
3. Make sure the call stays in the queue and not assigned to your agent even though the agent is available.
4. Finish the call.

Part 3 - Add Skill Relaxation Rule

> **Note**
>
> The skill relaxation rule allows for lowering the skill requirements for the call after a certain period of time. This helps to reduce the waiting time for callers by engaging less qualified agents when all more qualified agents are busy.
>
> Since our current skill requirement is **Webex CC Lab Expert >= 7** and our agent account has skill profiviency **Webex CC Lab Expert = 5**, let's configure the skill relaxation rule to lower the skill requirement to **Webex CC Lab Expert >= 4** after 15 seconds.

BUILD

1. Switch to your flow and click on **Queue** node - you should see **Your_Attendee_ID_SBR_Queue** skill-based queue configured there along with skill requirement **Webex CC Lab Expert >= 7**.

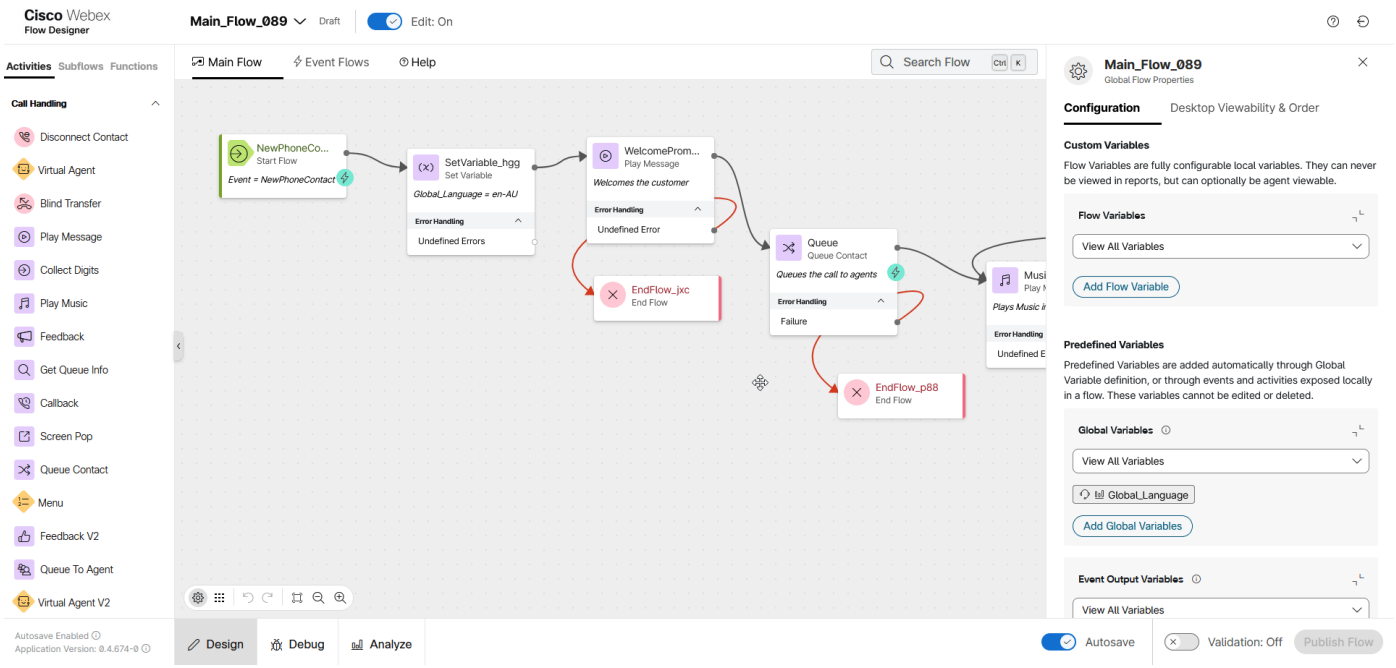
   Scroll down to the **Skill Relaxation** section, turn on **Enable Skill Relaxation** toggle and lower skill requirement **Webex CC Lab Expert** by configuring the following fields of **Skill Relaxation Step 1** block:

- After waiting in queue for: **15** seconds

- Select the skill name: **Static**
- Skill (drop-down list): **Webex CC Lab Expert**

- Select the condition: **Static**
- Condition (drop-down list): **>=**

- Select the value: **Static**
- Value (drop-down list): **4**



2. Validate and publish the flow:

- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.

FINAL MISSION TEST

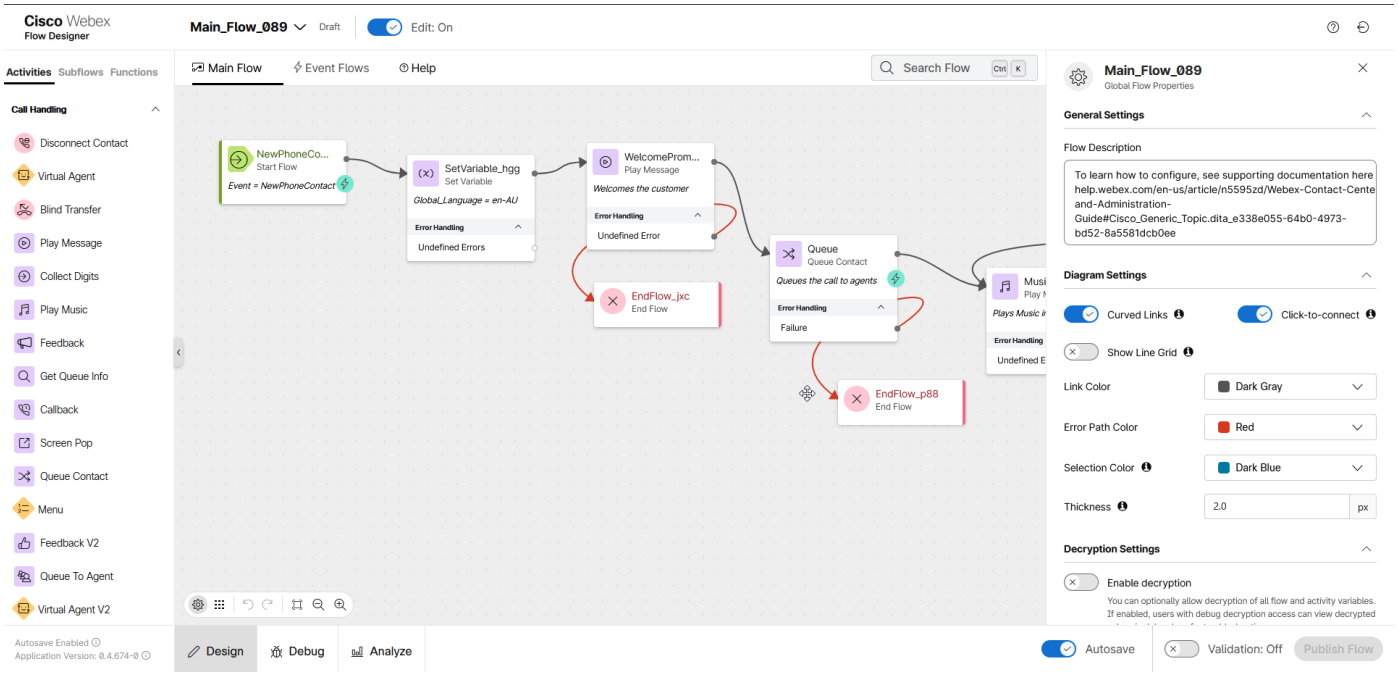
> **Note**
>
> We have lowered the skill requirement for **Webex CC Lab Expert** from **>= 7** to **>= 4** after 15 seconds in the queue.
>
> Since the current skill proficiency of **Webex CC Lab Expert** for your agent is still **5**, the agent does not meet the skill requirement for the call when it lands in the queue and **should not** be selected to answer it. Therefore, the call will remain in the queue. However, after 15 seconds, the skill requirement will be lowered to **Webex CC Lab Expert >= 4**, and the agent should be selected to answer it.
>
> Let's test this.

1. Open your Webex CC Desktop application and make sure your agent is in the **Available** state.
2. Make a test call to the Support Number. Ensure that the caller hears the welcome prompt, then the call is placed in a queue and caller hears music in queue.
3. Make sure the call stays in the queue for 15 seconds before being assigned to your agent.
4. Answer the call and verify that it works.
5. Finish the call.

Post Testing steps

1. **[IMPORTANT]** Now we need to revert the configuration of **Queue** node in your flow to the non-SBR queue **Your_Attendee_ID_Queue** as we are going to use same flow in upcoming tasks:

- Click on the **Queue** node on the flow canvas.
- Navigate to the node settings panel on the right.
- Scroll down and select **Your_Attendee_ID_Queue** from **Queue** dropdown list.



2. Validate and publish the flow:

- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.

3. Open your Webex CC Desktop application and make sure your agent is in the **Available** state.
4. Make another test call to the Support Number and confirm that the caller hears the welcome message, and then the call is instantly assigned to your agent after entering the queue.

Congratulations, you have succesfully completed Skill-Based Routing mission! 🎉

2.1.3 Mission 3: Using Business Hours

Story

Business Hours allows you to configure the operational hours of the contact center, offering an enhanced experience in routing strategy configuration and simplifying the routing flow for improved efficiency and customer satisfaction.

Call Flow Overview

1. A new call enters the flow.
2. The flow determines the caller's language preference and plays a preconfigured Text-to-Speech (TTS) prompt.
3. The flow determines whether it is currently within working hours and routes the call appropriately.
4. The call is routed to the appropriate queue.

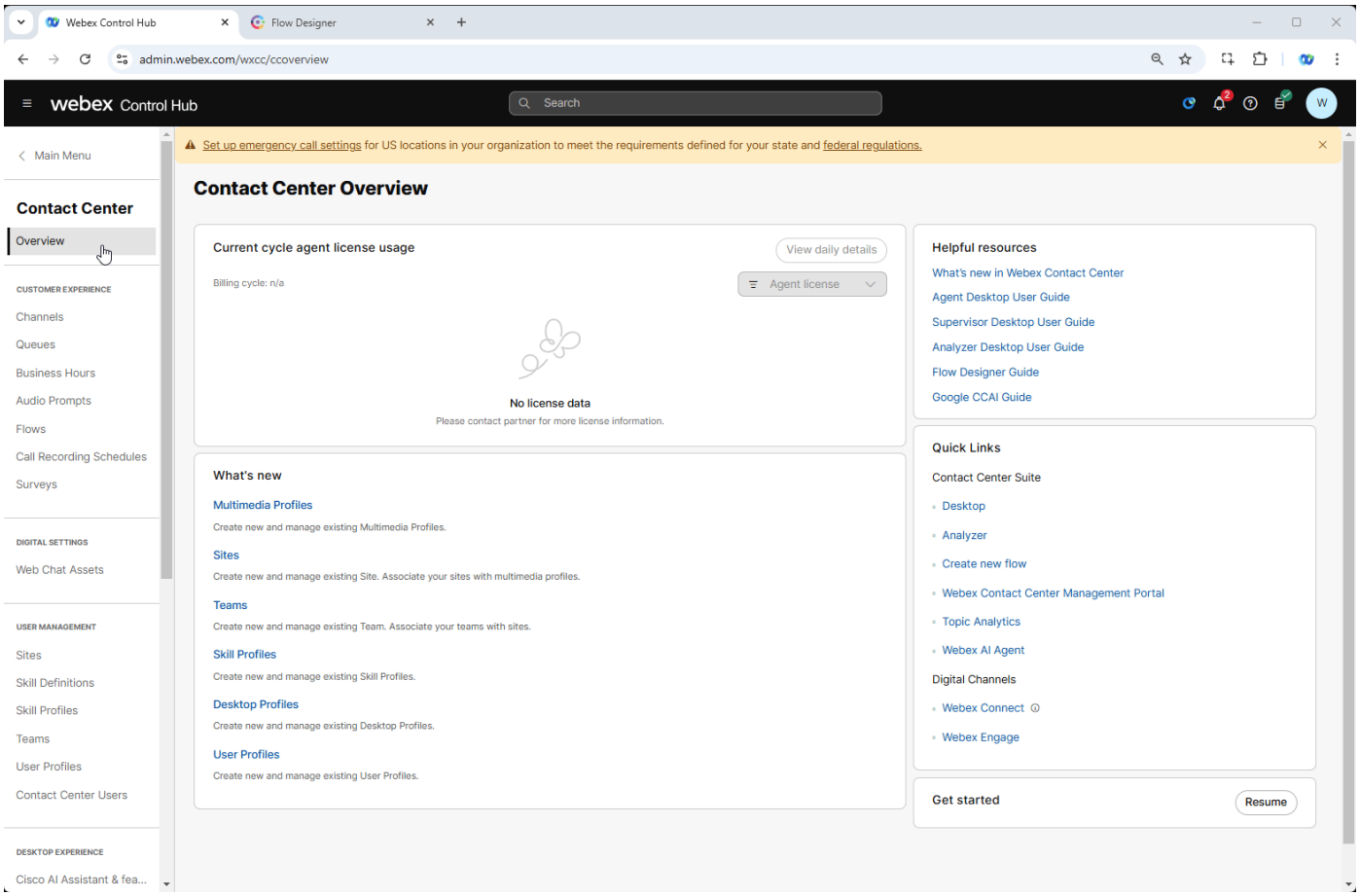
Mission Details

Your mission is to:

- Continue to use same flow **Main_Flow_Your_Attendee_ID** we created in previous Mission.
- Add Business Hours functionality **Your_Attendee_ID_Bussiness_Hours** to your flow. Business Hours entity has been configured for you and contains the following settings:
- **Working Hours** - Define the time during which the contact center will be operational. Each working hour configuration can include one or more shifts. Different schedules can be set for various time zones.
- **Holidays** - Specify a day or range of days declared as holidays. The entire 24 hours of the selected day(s) are marked as non-operational.
- **Overrides** - Configure working hours for special cases, such as emergencies or occasions like Christmas, when the contact center operates for additional hours.

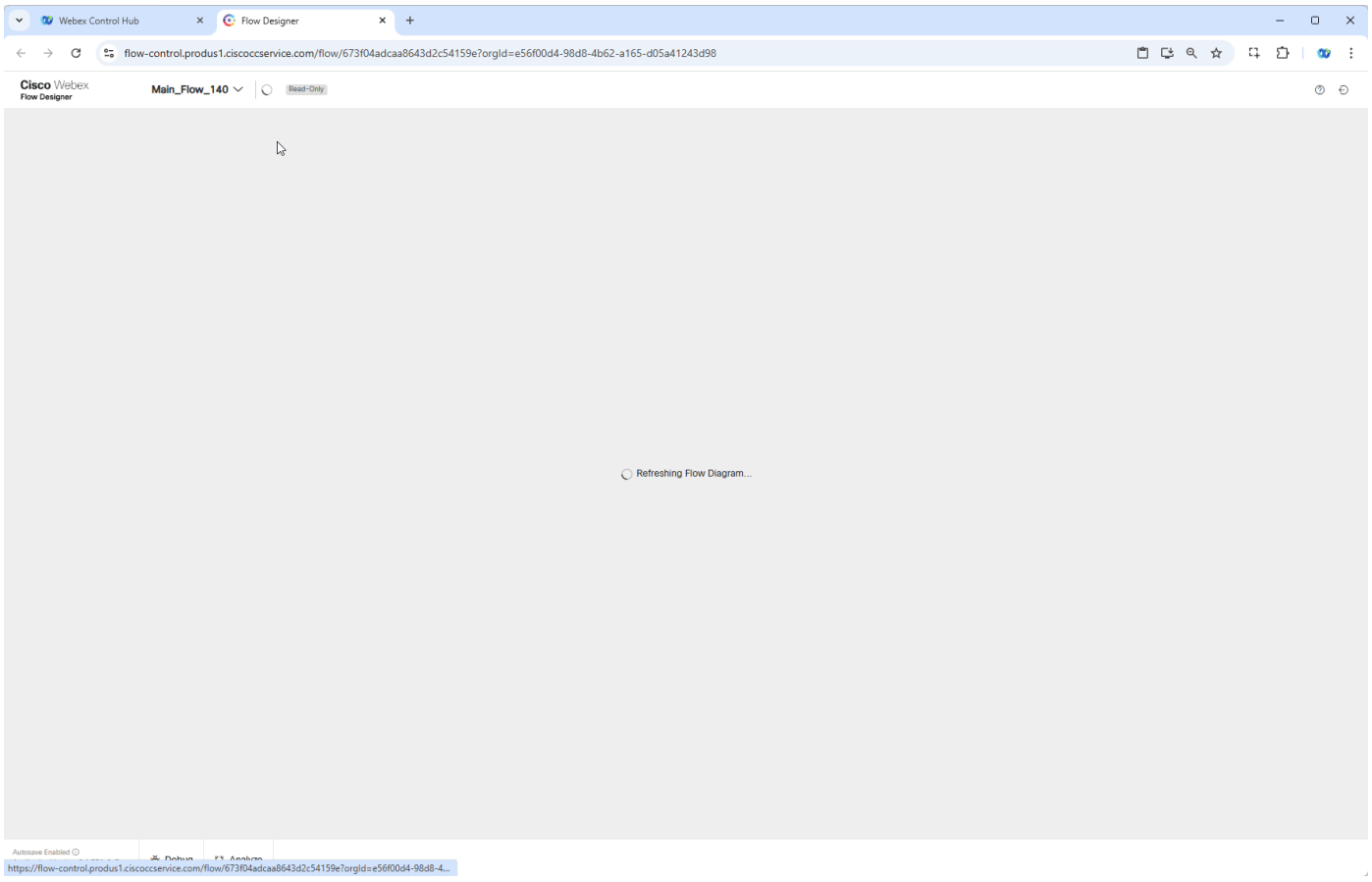
Build

1. Switch to **Control Hub** and navigate to **Business Hours** under Customer Experience section. Locate your **Your_Attendee_ID_Bussiness_Hours** You will see that currently only **Working Hours** are configured for every working day between 12:00 AM to 11:59 PM".



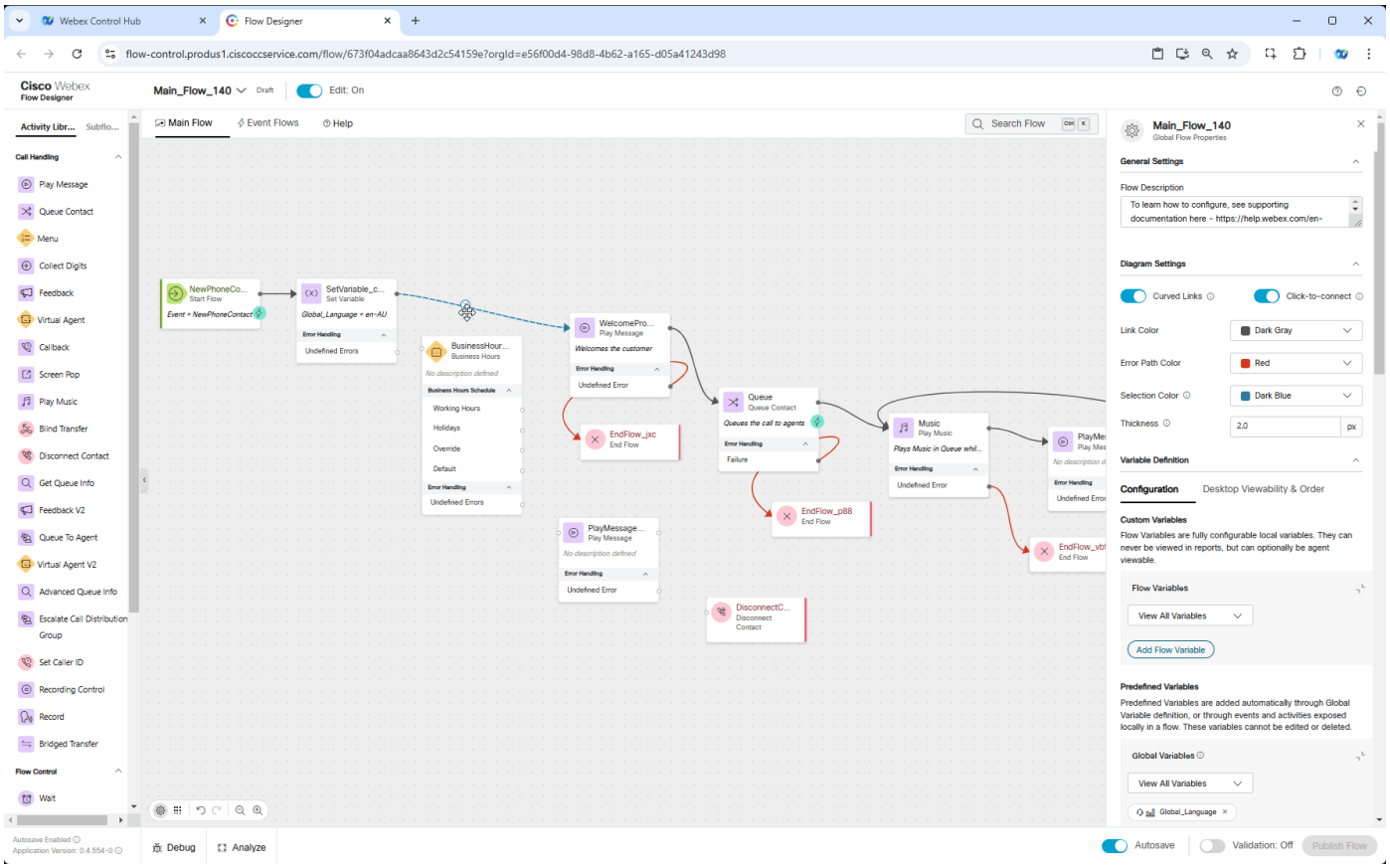
2. Switch to Flow Designer. Open your flow **Main_Flow_Your_Attendee_ID** and make sure **Edit** toggle is **ON**.
3. Drag and drop following nodes to the canvas:

- **Business Hours**
- **Play Message**
- **Disconnect Contact**



4. Connect **Set Variable** node to **Business Hours** and **Business Hours** node exits as follows:

- **Working Hours** connect to **WelcomePrompt** node.
- **Holidays**, **Overrides** and **Default** connect to new added **Play Message** node.
- New added **Play Message** connect to **Disconnect Contact** node.



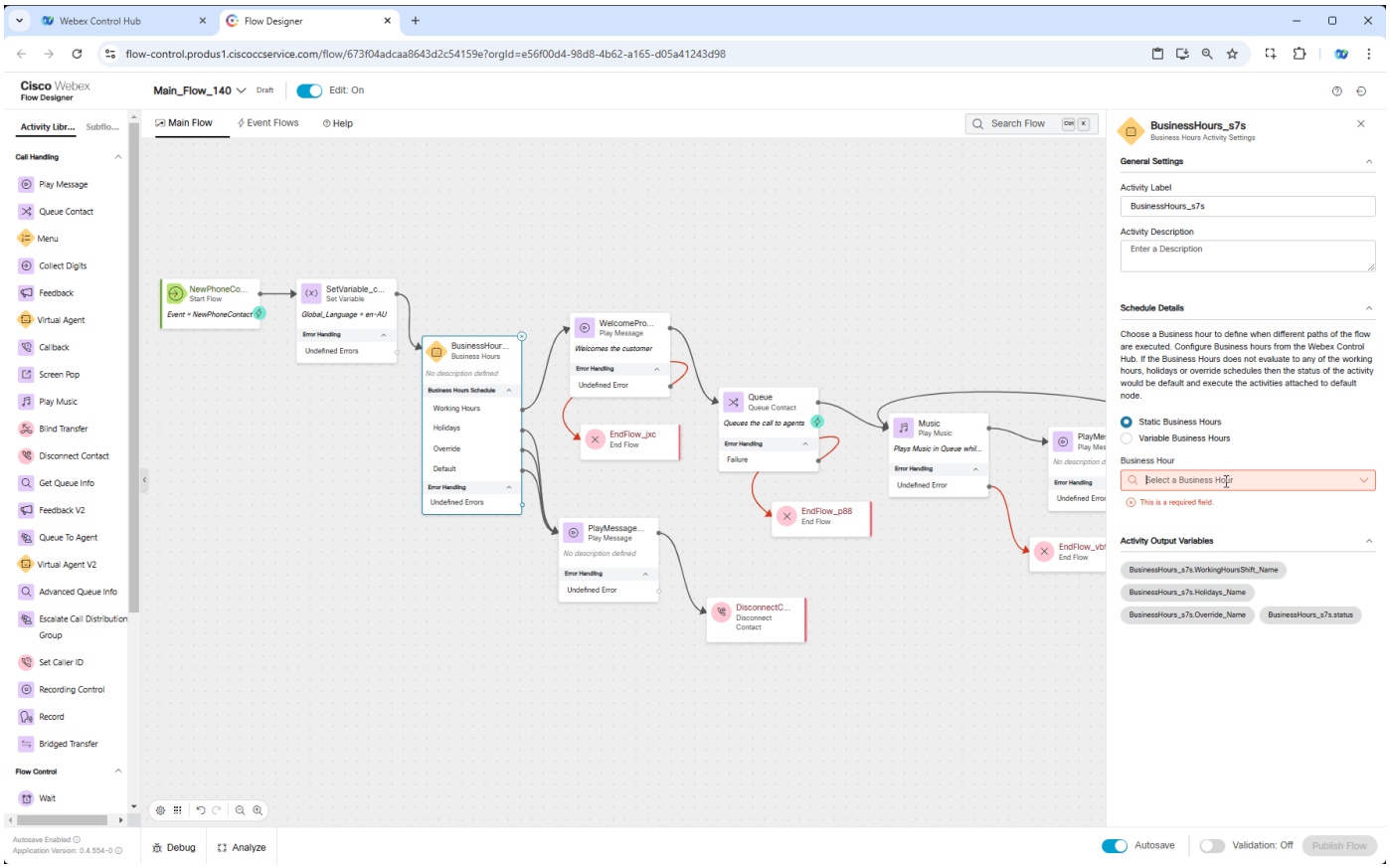
5. Click on **Business Hours** node and select preconfigured Business Hours Entity **Your_Attendee_ID_Bussiness_Hours** .
6. Configure **Play Message** node as follows:

- Turn on **Enable Text-To-Speech** toggle
- Select the Connector: **Cisco Cloud Text-to-Speech**
- Click the **Add Text-to-Speech Message** button and paste text: ***It's not working hours currently. Please call later. Goodbye.***
- Delete the selection for Audio File

7. Validate and publish the flow:

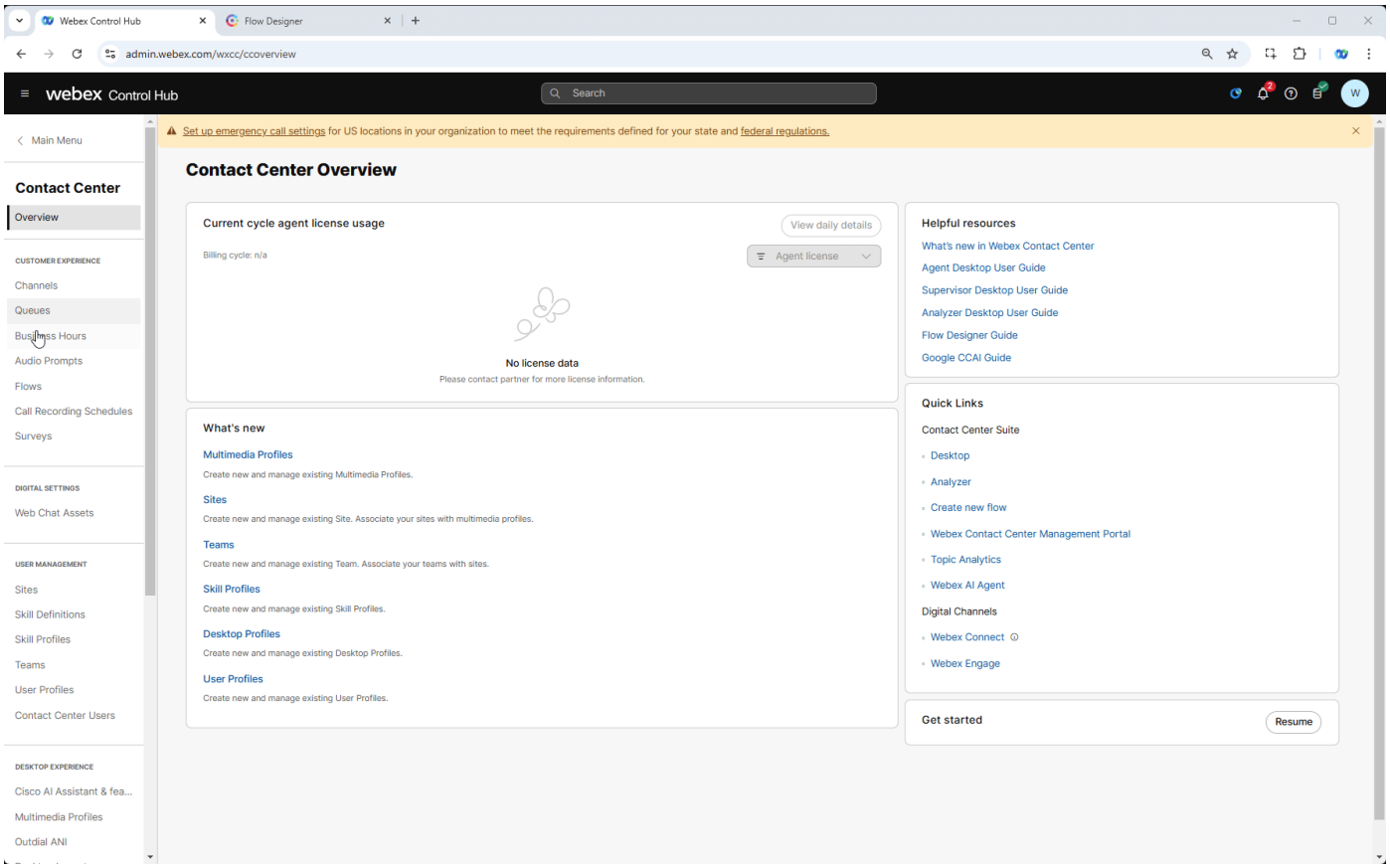
- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.

> **Note**
>
> We haven't changed the flow behavior yet as Working Hours covers the current time. You can make a call and accept it on agent desktop to verify.



8. We are going to use **Override** option to change the logic. Overrides as well as Business hours have been preconfigured for you. Now we need to apply it on your **Your_Attendee_ID_Bussiness_Hours** entity. Switch to Control Hub and open **Your_Attendee_ID_Bussiness_Hours** in **Control Hub**, scroll down to **Additional Settings** and select **Overrides_Hours** from Override dropdown list. Then click **Save**.

> **Note**
>
> Override Hours entity was configured to overwrite Working Hours and set to duration of current Cisco Live lab

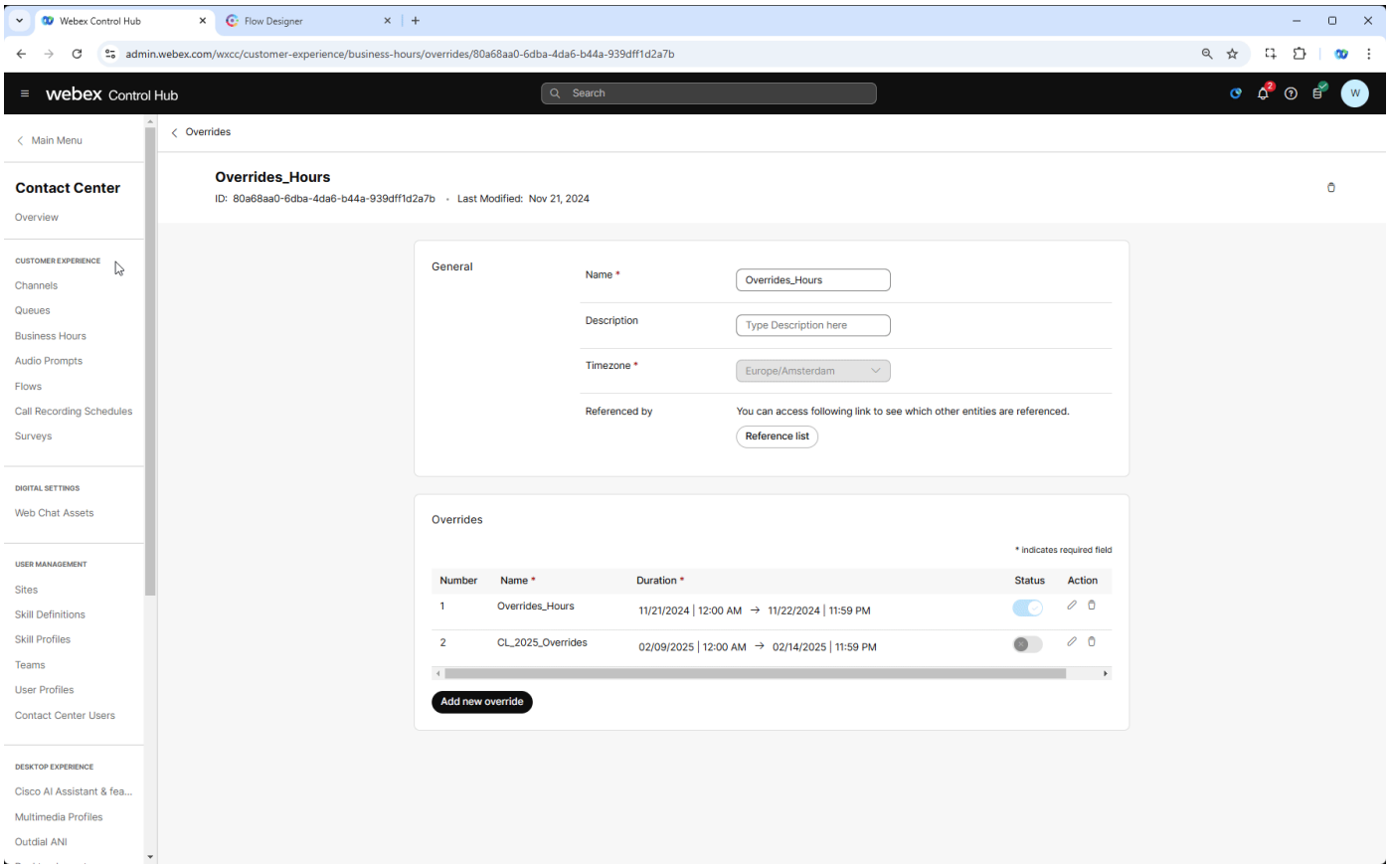


Testing

1. Open your Webex App and dial the Support Number provided to you, which is configured in your **Your_Attendee_ID_Channel** configuration. Make sure you hear the message we set in ***Step 6***.

POST TESTING STEPS

1. **[IMPORTANT]** Now we need to revert the configuration we made in ***Step 8*** as we are going to use same flow in upcoming tasks. Open **Your_Attendee_ID_Bussiness_Hours** in **Control Hub**, scroll down to Additional Settings and select **None** from **Override** dropdown list. Then click **Save**.



2. Make one more call from Webex App to make sure you hear the original Welcome message you set on first steps of previous Mission.

Congratulations, you have succesfully completed Using Business Hours mission! 🎉

2.1.4 Mission 4: Work with Event Flow

Story

An Event Flow in Webex Contact Center is a workflow triggered by specific events in the customer interaction process, such as call arrival, agent assignment, call disconnection or actions within the IVR.

Event flows enable a wide range of scenarios, with one common use case being the ability to update an external database with data collected during a call—either from the IVR or through interaction with a live agent.

Call Flow Overview

1. A new call enters the flow.
2. The flow executes the logic configured in previous steps.
3. When the agent answers the call, they receive a screen pop and can adjust call details on the interaction panel.
4. The flow triggers an event when the agent disconnects from the call.

Mission Details

1. Continue to use same flow **Main_Flow_Your_Attendee_ID**
2. Configure a screen pop in your flow.
3. Configure an API call to trigger on the AgentDisconnect event.

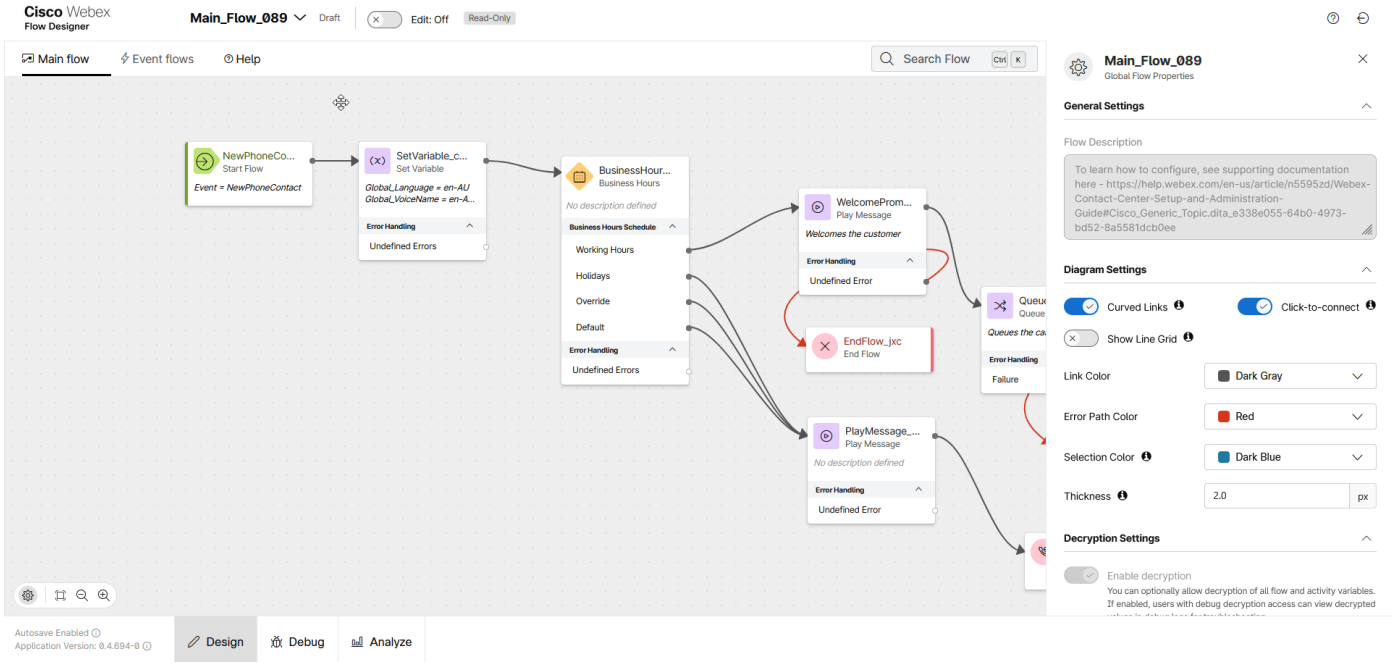
> **Note**
>
> In this mission, we'll utilize **UseWebhook site**, a free online tool that generates a temporary, unique URL for capturing and inspecting HTTP requests. It's widely used by developers and testers for debugging and testing webhooks or other HTTP-based APIs.

Build

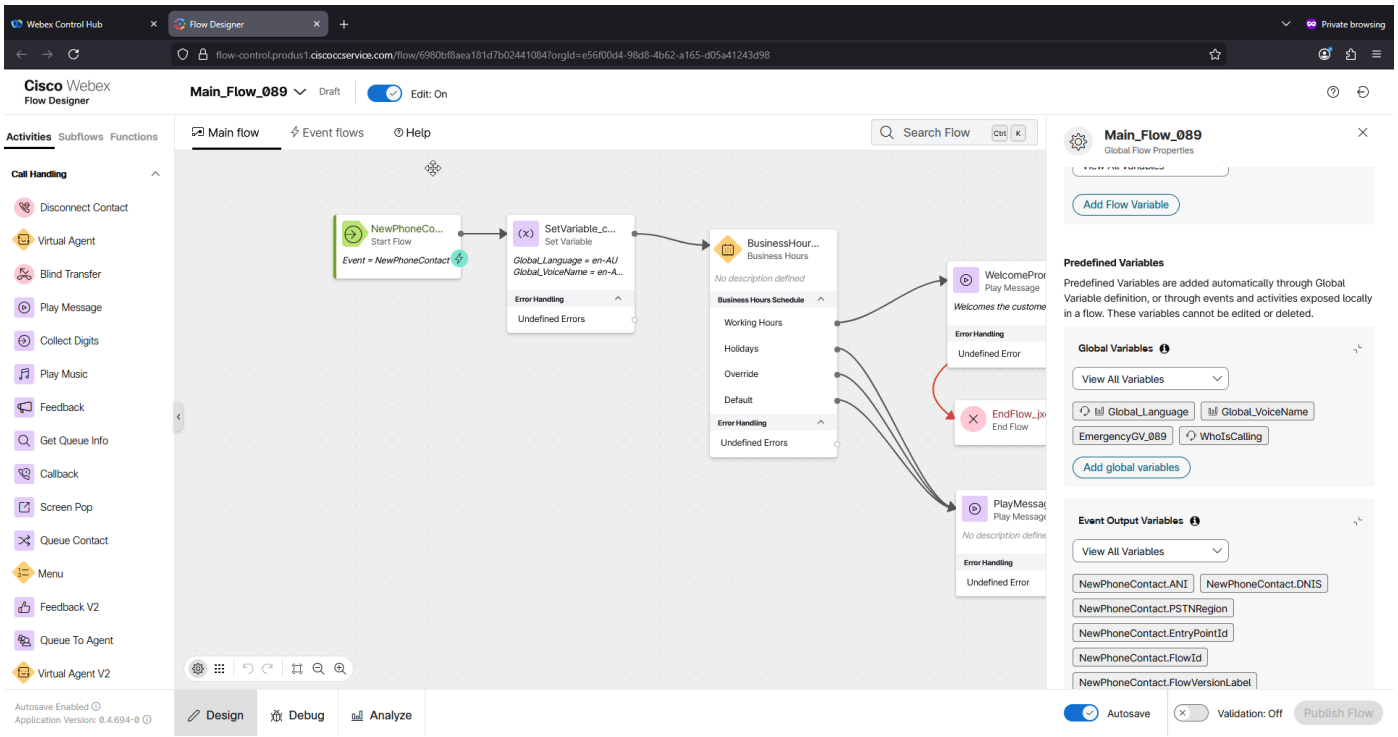
Note

The **Global Variable** with name **WhoIsCalling** that we are going to use in this mission has been already created. Switch to **Control Hub** and navigate to **Flows** under Customer Experience section. Select Global Variables on top and search for **WhoIsCalling** to observe it's configuration. Please **DO NOT** modify it here.

1. Open you your **Main_Flow_Your_Attendee_ID** or refresh the Flow Designer page to make sure new created Global Variables are being populated. Make sure **Edit** toggle is **ON**
2. Add **WhoIsCalling** Global Variable to the flow.



3. Open New Browser tab and paste the following URL **UseWebhook site**. Then click on **Copy URL** button next to the webhook URL at the top of the page to make a copy of URL. **Please keep this browser tab open to avoid changing your unique Webhook URL.**



4. Go back to your flow and navigate to **Event Flows** tab, delete **EndFlow** node connected to **AgentDisconnect** event.
5. Add **HTTPRequest** and **DisconnectContact** node in between these nodes.

- Connect **AgentDisconnect** to **HTTPRequest** node.
- Connect **HTTPRequest** to **DisconnectContact**.

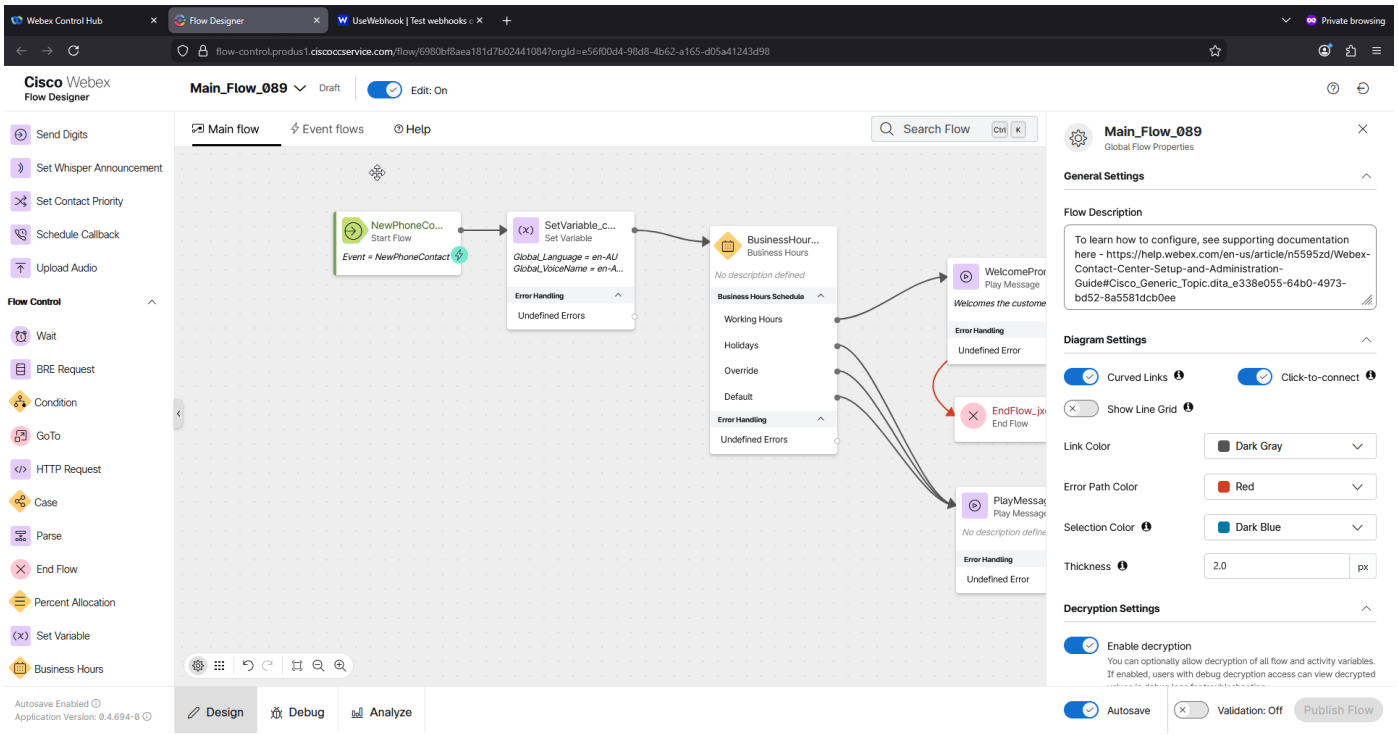
6. Modify **HTTPRequest** node settings:

- Use Authenticated Endpoint: **Off**
- Request URL: *Paste your unique URL copied on Step 3 from https://usewebhook.com/*
- Method: **POST**
- Content Type: **Application/JSON**

Request Body:

```
{
    "DNIS": "{{NewPhoneContact.DNIS}}",
    "ANI": "{{NewPhoneContact.ANI}}",
    "InteractionId": "{{NewPhoneContact.InteractionId}}",
    "Language": "{{Global_Language}}",
    "WhoCalls": "{{WhoIsCalling}}"
}
```

> **Note**
>
> We are building a dictionary with values generated by flow, language we set in main lab and also WhoIsCalling value which will be provided by agent in agent desktop.

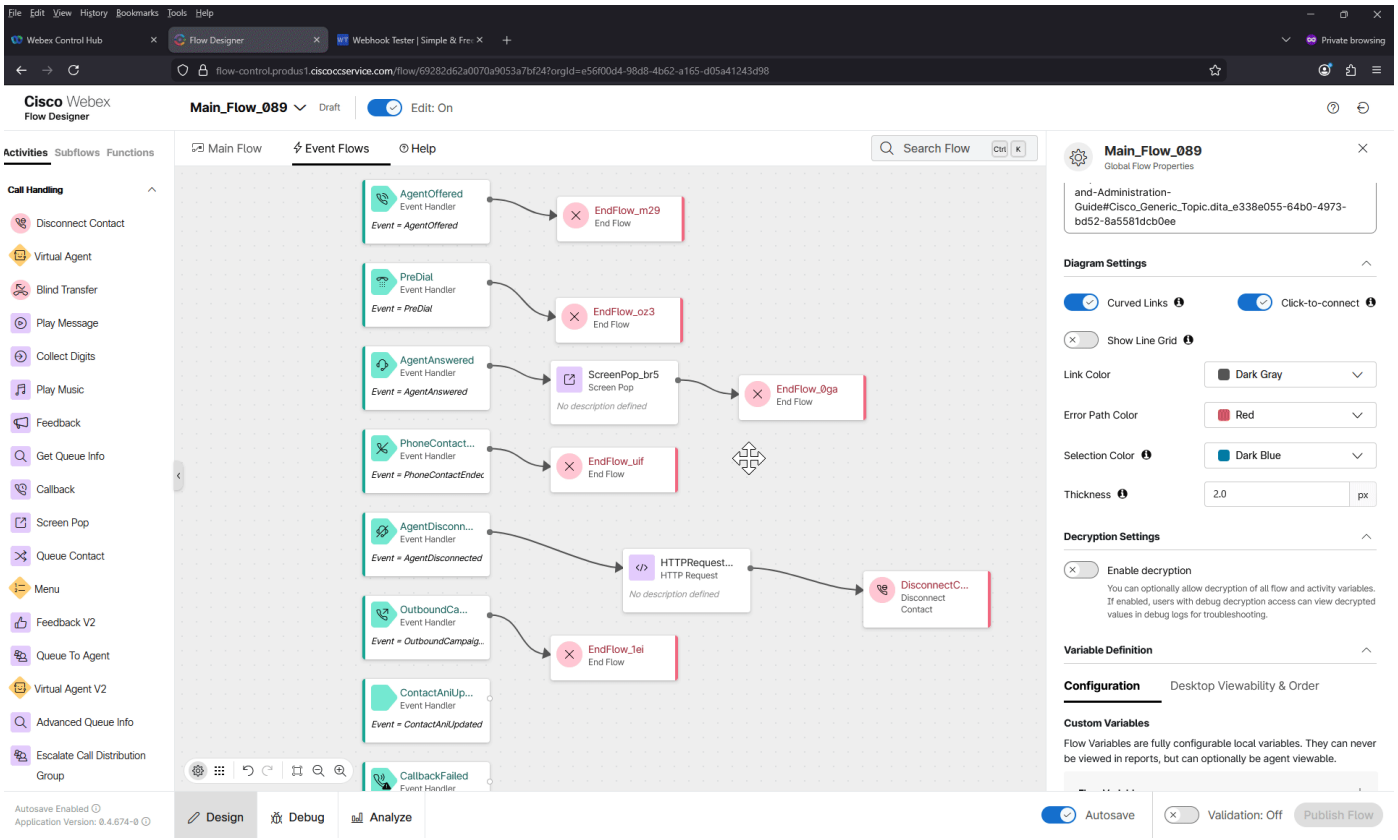


7. Modify the settings of **ScreenPop** node connected to the **AgentAnswered** event in the same flow:

- Screen Pop URL: **https://www.webex.com/us/en/products/customer-experience/contact-center.html**
- Screen Pop Desktop Label: **Webex Contact Center**
- Display Settings: **New browser tab**

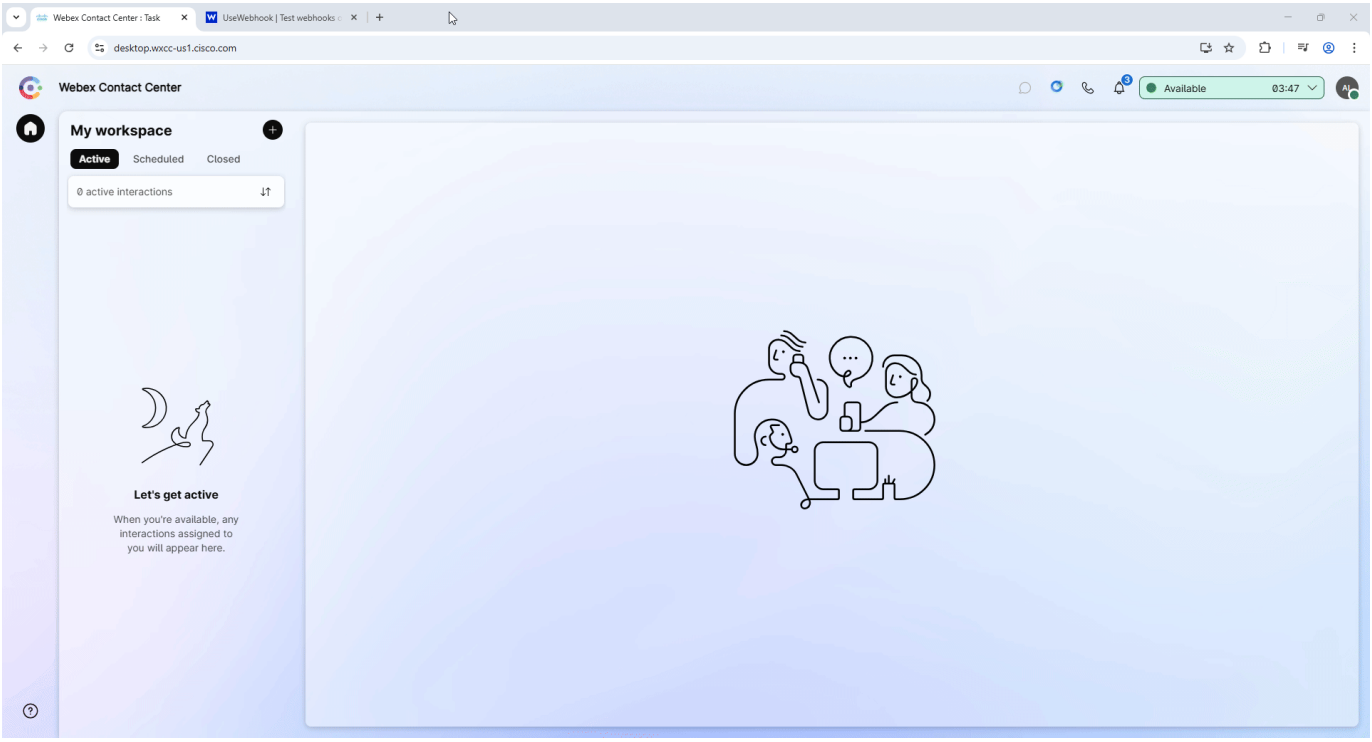
8. Validate and publish the flow:

- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.



Testing

1. Make sure you're logged into Webex CC Desktop application as Agent **wxcclabs+agent_IDYour_Attendee_ID@gmail.com** and set status to **Available**.
2. Make a call to the Support Number and if success you should hear Welcome message and then accept the call by agent.
3. Upon accepting the call, a new browser tab will be opened with the Screen Pop URL configured in **Step 4**.
4. Switch back to the Agent Desktop. In agent interaction panel change **Who Is Calling?** to any name you like then click **Save** and **end the call in the agent desktop**.
5. Switch to the **UseWebhook site** tab in your browser - you should see the request which came right after Agent dropped the call with all the needed data.

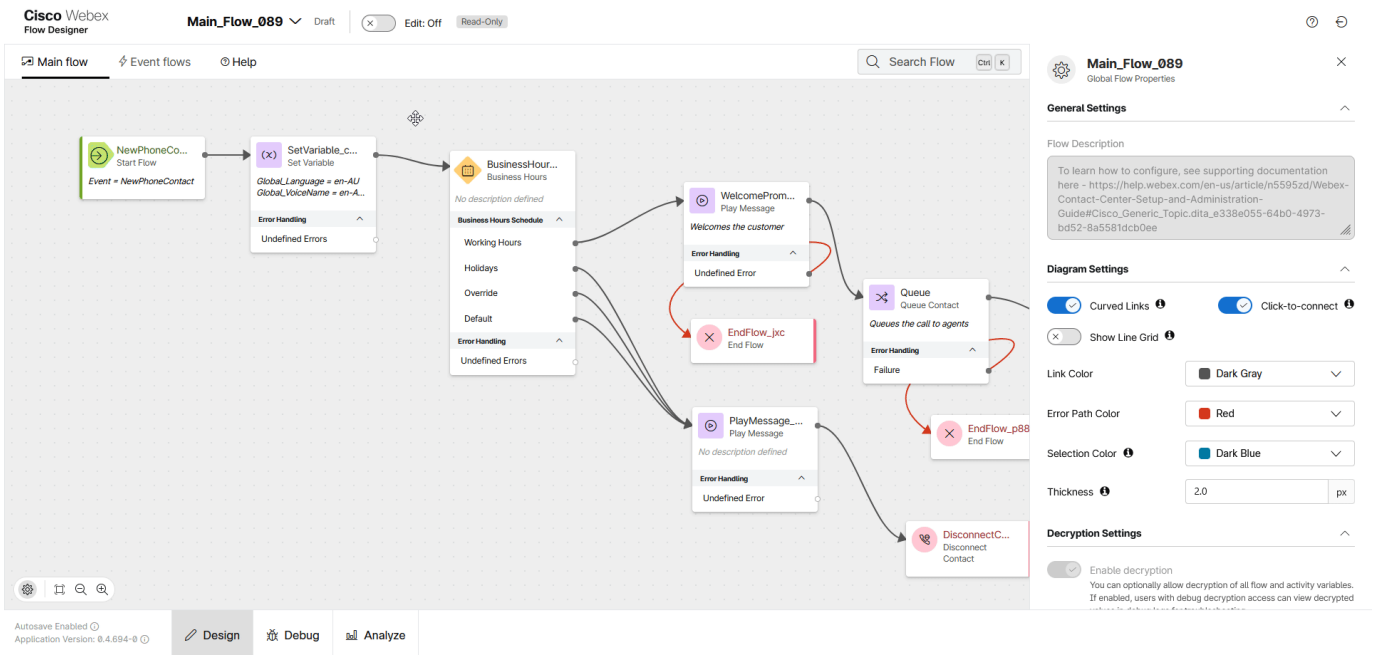


Post Testing Steps

We recommend removing the ScreenPop node after testing. Otherwise, every time you make a new call to the Main Flow, a pop-up will appear, which may be distracting.

1. Open your flow **Main_Flow_Your_Attendee_ID**. Make sure **Edit** toggle is **ON**.
2. Navigate to **Event Flows** and delete **ScreenPop** node connected to the **AgentAnswered** event node.
3. Connect **AgentAnswered** event node to the corresponding **EndFlow** node.
4. Validate and publish the flow:

- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.



Congratulations, you have succesfully completed Work with Event Flow mission! 🎉🎉

2.1.5 Mission 5: Adding Real-Time Transcript

Feature Description

You can enhance communication efficiency and quality assurance in your contact center with the real-time transcripts feature. It allows agents to access real-time transcriptions of customer interactions directly on their Agent Desktop, enabling them to follow conversations more accurately and respond effectively.

BENEFITS

- **Accurate communication**: Captures conversations precisely, aiding understanding, especially with diverse accents or non-native speakers.
- **Improved efficiency**: Eliminates manual note-taking, enabling agents to focus and resolve issues faster.
- **Better performance**: Helps agents deliver timely, accurate solutions, boosting customer trust.
- **Higher customer satisfaction**: Reduces misunderstandings, improving CSAT scores and experiences.
- **Training and quality assurance**: Provides reliable references for coaching, evaluations, and compliance checks.
- **Seamless integration**: Easily integrates with systems, optimizing queue-level management.
- **AI support**: Enhances decision-making with complementary AI Assistant features.

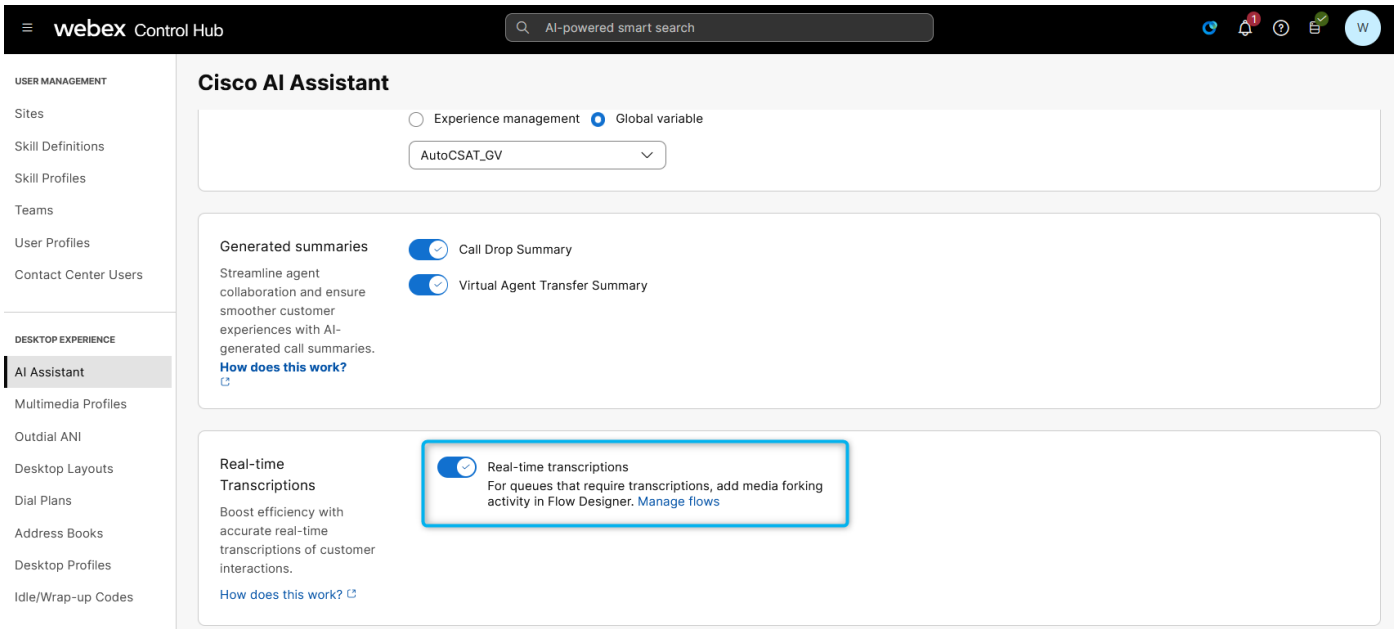
MISSION DETAILS

Your mission is to:

1. Verify Real-Time Transcript feature configuration
2. Configure flow with **Start Media Stream** functionality
3. Test Real-Time Transcript feature

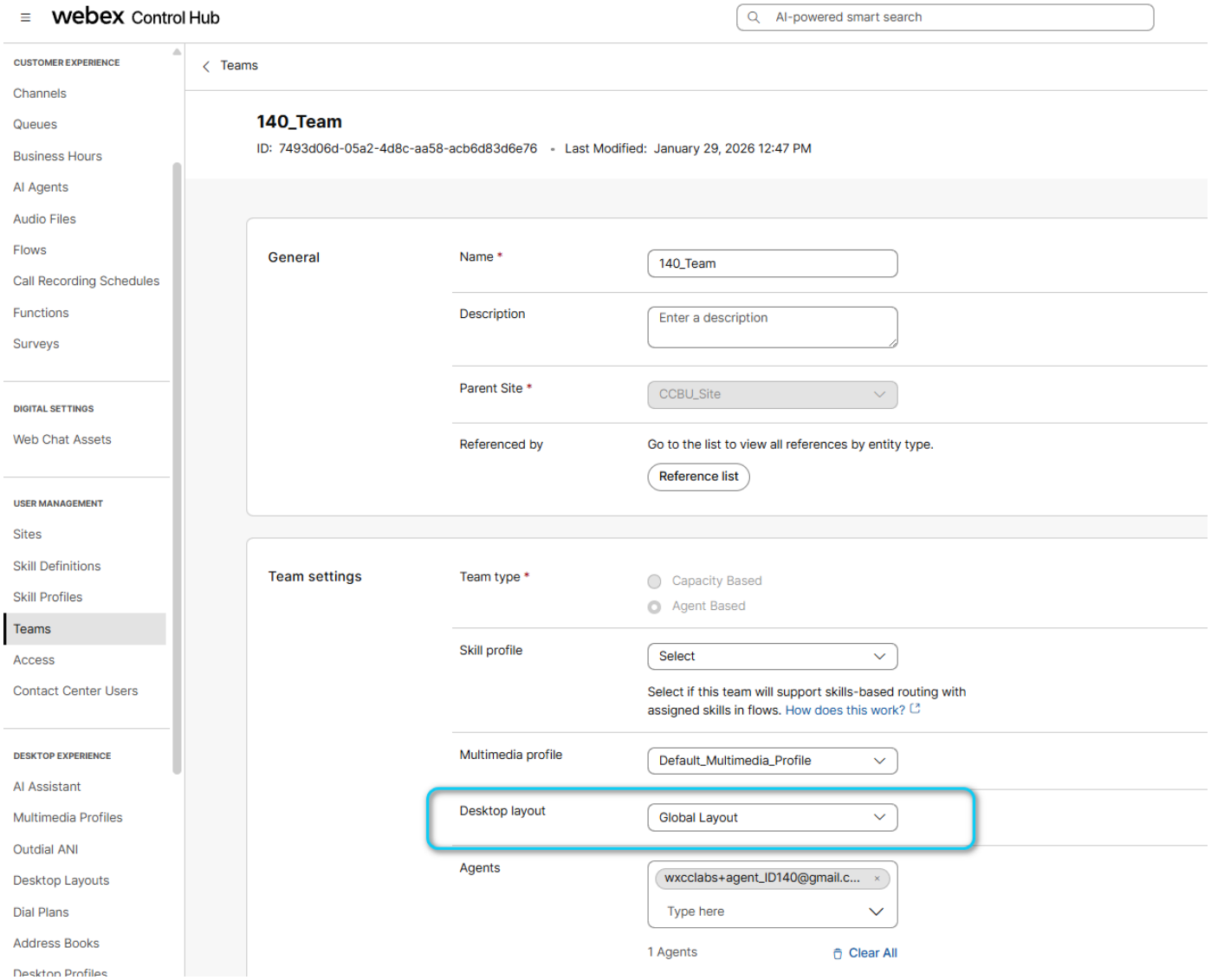
Verify feature configuration

1. Switch to Control Hub and navigate to **AI Assistant** under Desktop Experience section. Make sure **Real-time Transcriptions** feature is turned on.

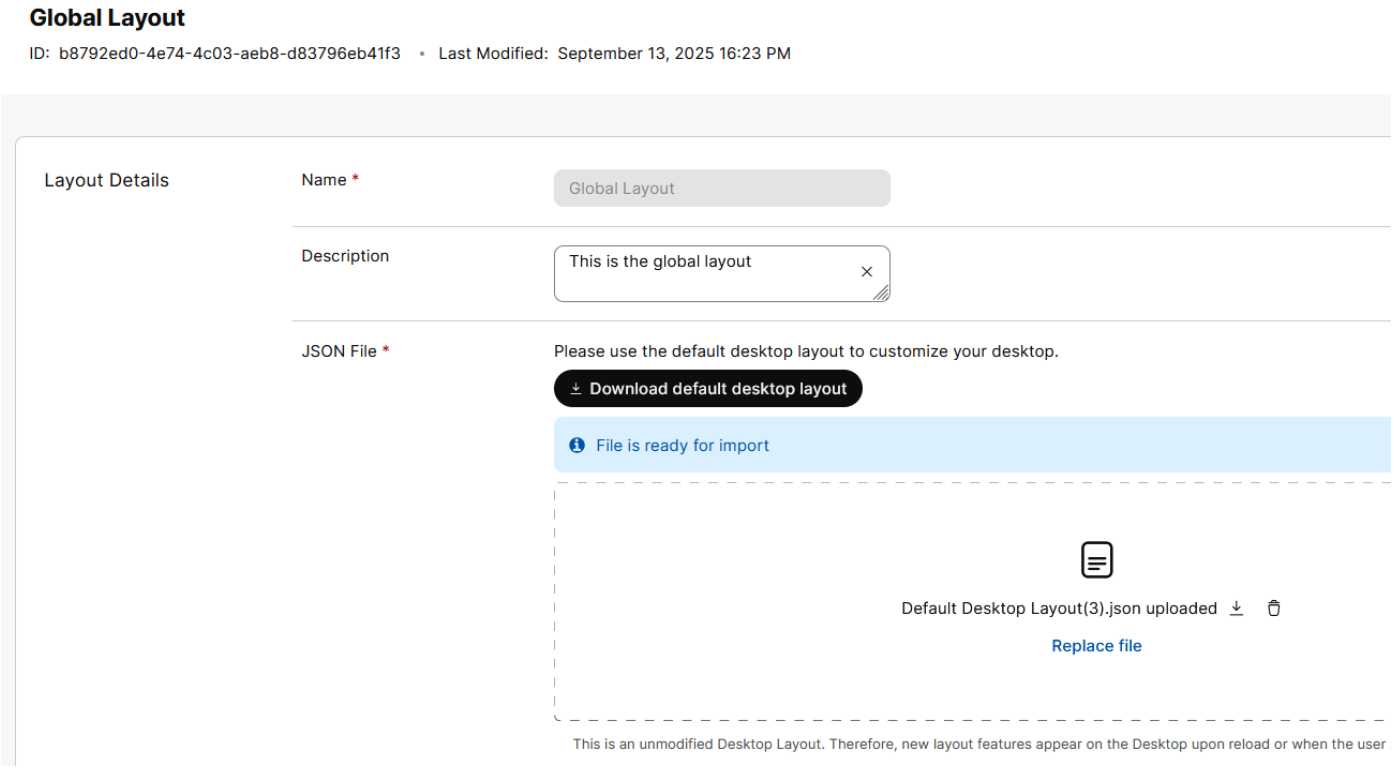


2. The agent's team has been preconfigured with the Global Layout, which already includes the AI Assistant functionality. You can verify it visiting the following configuration pages in Control Hub:

a. Contact Center -> Teams (under User Management section) -> search for your team **Your_Attendee_ID_Team**



b. Contact Center -> Desktop Layout (under Desktop Expirience section) -> search for **Global Layout**



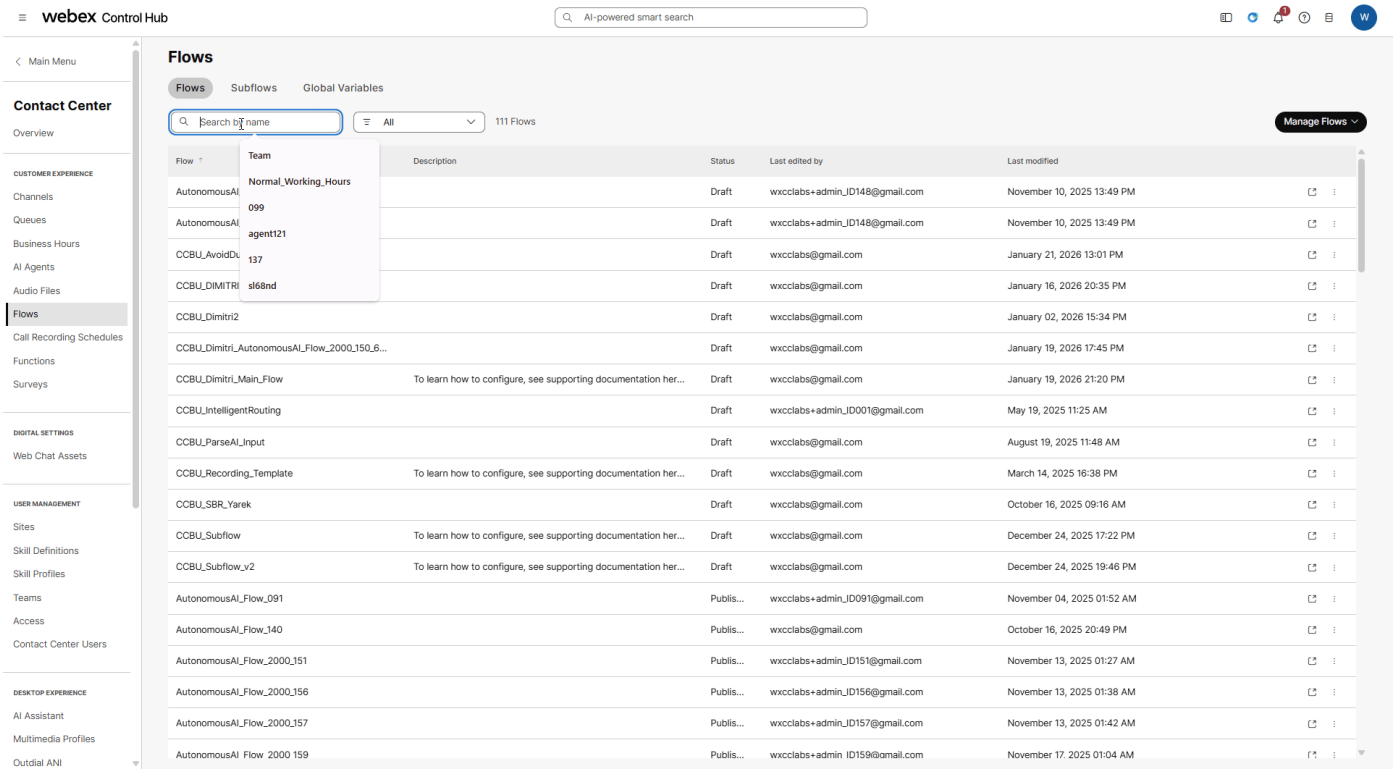
c. Click **Download default desktop layout**. Make sure **RT_TRANSCRIPT** widget is configured.

```
                "name": "CONTACT_HISTORY"
              }
            }
          ]
        },
        {
          "comp": "md-tab",
          "attributes": {
            "slot": "tab",
            "class": "widget-pane-tab"
          },
          "children": [
            {
              "comp": "slot",
              "attributes": {
                "name": "RT_TRANSCRIPT_TAB"
              }
            }
          ],
          "visibility": "RT_TRANSCRIPT"
        },
        {
          "comp": "md-tab-panel",
          "attributes": {
            "slot": "panel",
            "class": "widget-pane"
          },
          "children": [
            {
              "comp": "slot",
              "attributes": {
                "name": "RT_TRANSCRIPT"
              }
            }
          ],
          "visibility": "RT_TRANSCRIPT"
        },
        {
          "comp": "md-tab",
```

Or you can download preconfigured desktop layout here: Desktop Layout

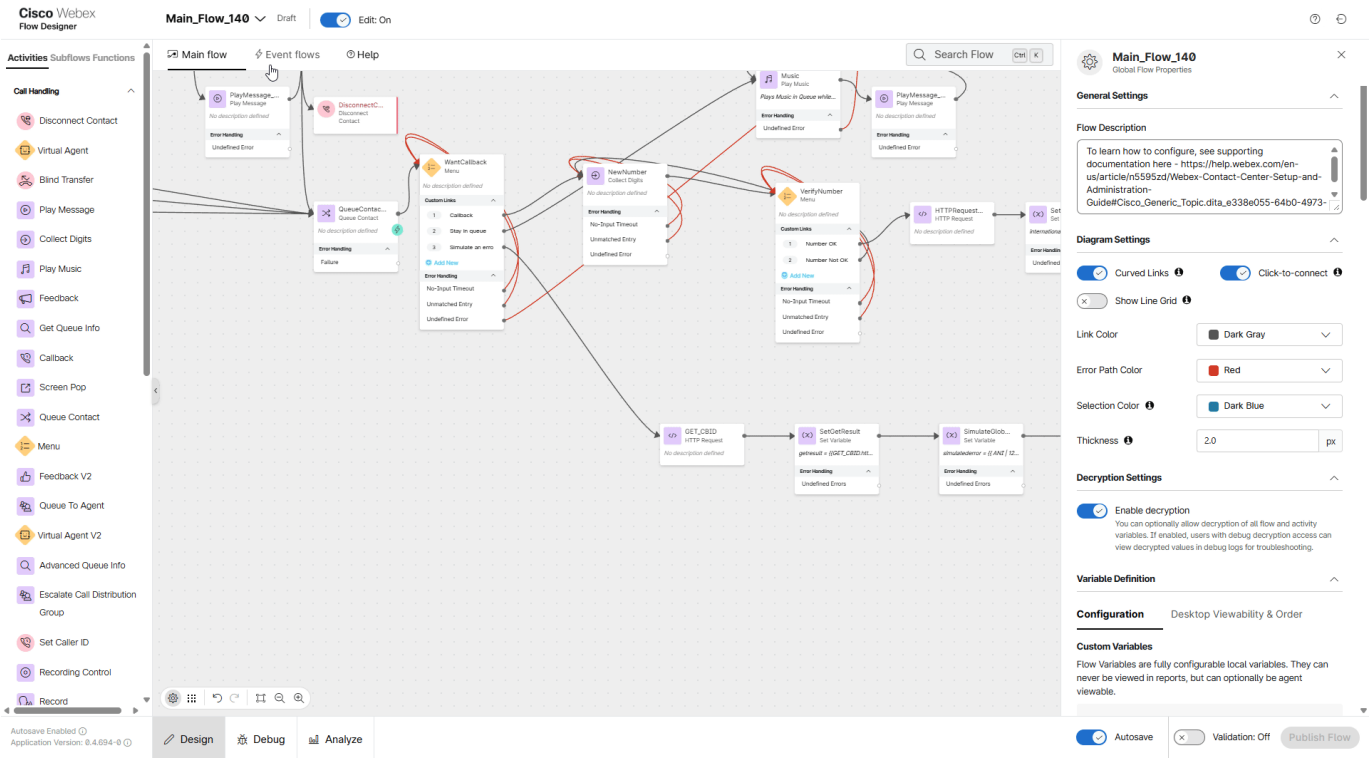
Build

1. Open up your voice flow **Main_Flow_Your_Attendee_ID** and click on **Edit**.



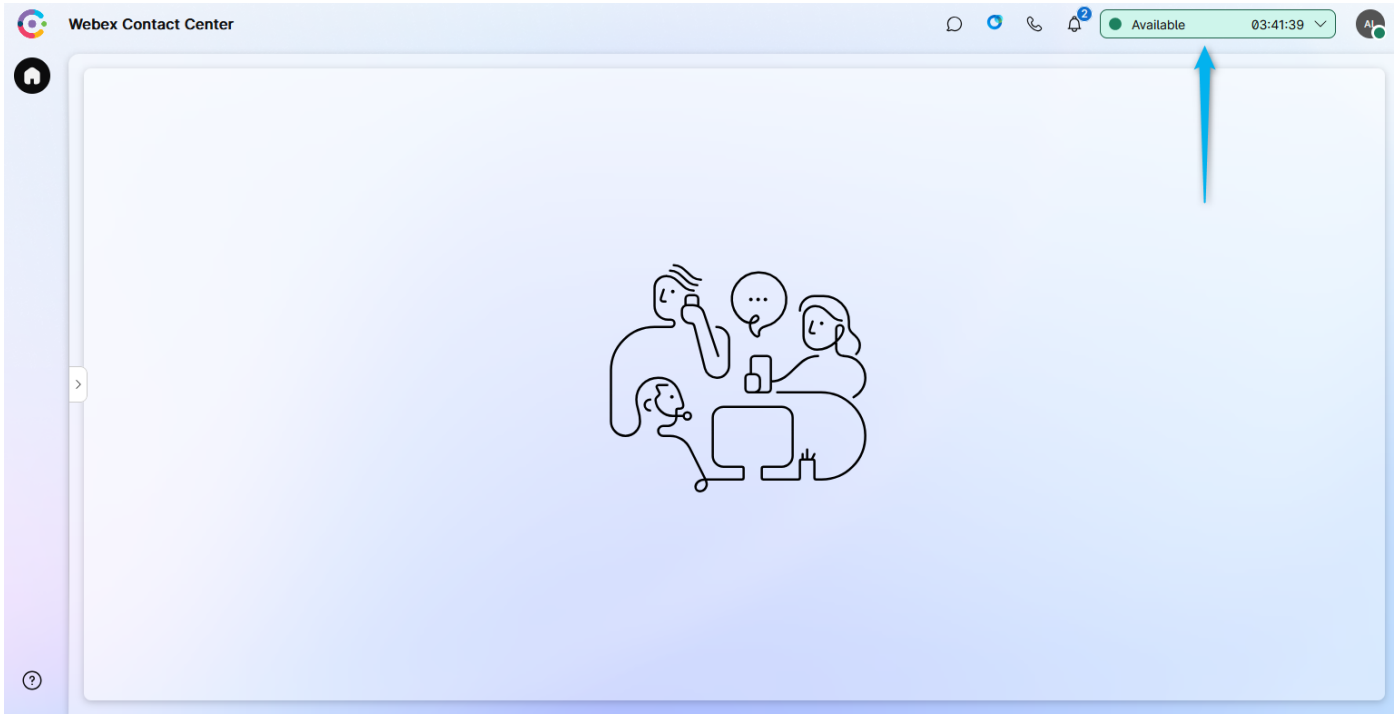
2. Click on the **Event Flow**.
3. Drag and drop **Start Media Stream** node and connect **AgentAnswer** node to the **Start Media Stream** node.
4. Drag and drop **End Flow** node and connect **Start Media Stream** to **End Flow**.
5. Validate and publish the flow:

- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.



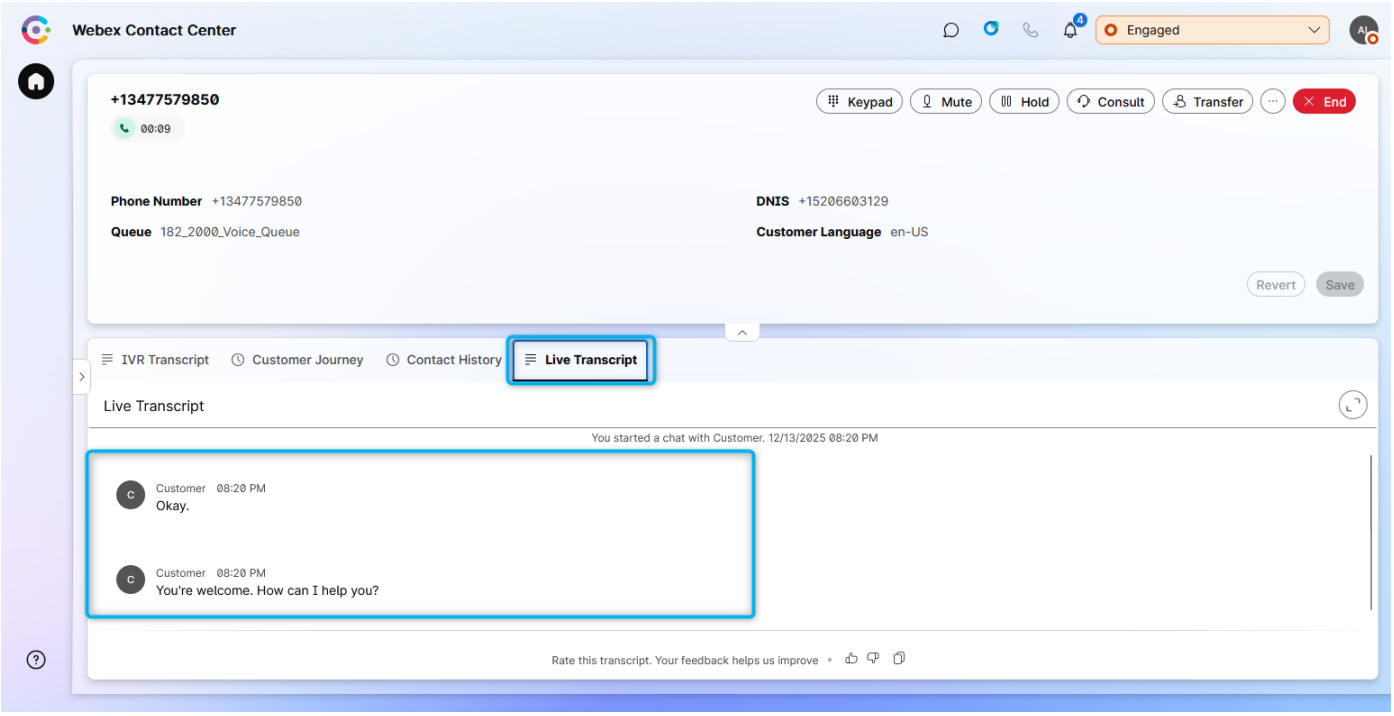
Test Real-Time Transcript feature

1. Make your agent **Available**.



2. Place the test call to the number that is associated with your **Your_Attendee_ID_Channel**, and ask to talk to an agent.

3. Answer the call and begin speaking. The Live Transcripts panel will display the most recent real-time transcript of the conversation between the caller and the agent.



Congratulations, you have succesfully completed Adding Real-Time Transcript mission! 🎉

2.1.6 Mission 6: Post Call Survey

Story

In this lab, you will complete a mission to enhance customer feedback collection by integrating a survey into the Webex Contact Center call flow. The lab is designed to be simple yet practical, focusing on minimal configuration within the Flow Designer, while leveraging a preconfigured survey template.

od to Know [Optional]

Supported Survey Question Types in Webex Contact Center

1. **Customer Satisfaction (CSAT)**:
• Purpose: Measure satisfaction with a specific interaction or service.
• Example Question: "On a scale of 1 to 5, how satisfied are you with the service you received today?"
• Use Case: Assess overall satisfaction at the end of a call or interaction.
2. **Customer Effort Score (CES)**:
• Purpose: Evaluate the ease of resolving a customer's issue or completing a task.
• Example Question: "On a scale of 1 to 5, how easy was it to complete your task today?"
• Use Case: Identify pain points in the customer journey or process efficiency.
3. **Net Promoter Score (NPS)**:
• Purpose: Measure customer loyalty and the likelihood of recommending the service.
• Example Question: "On a scale of 0 to 10, how likely are you to recommend our service to a friend or colleague?"
4. Use Case: Gauge long-term customer loyalty and brand advocacy.

Call Flow Overview

1. A new call enters the flow.
2. The flow executes the logic to enable survey functionality.
3. Agent answers the call.
4. The flow triggers an event when the agent disconnects from the call.
5. The caller remains on the line and hears the survey menu.

Mission Details

Your mission is to:

1. Integrate a preconfigured survey into the call flow using the Flow Designer.
2. Configure basic logic to determine when to route customers to the survey (e.g., after a call ends).
3. Understand how Webex Contact Center supports various survey question types, including CSAT, CES, and NPS.

Note

The survey is prebuilt and includes key questions designed to gather actionable insights from customers. Your task is to focus on configuring the flow and ensuring the survey is triggered seamlessly during the customer journey.

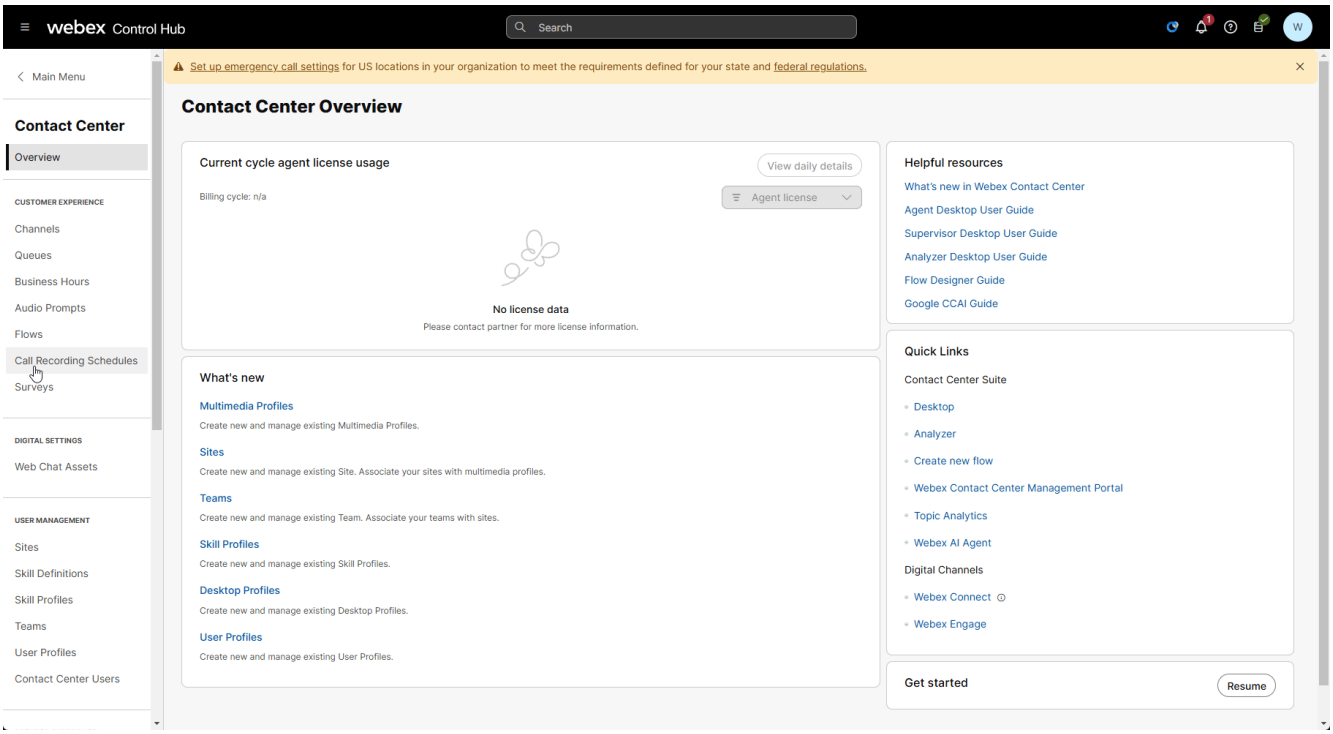
PRECONFIGURED ENTITIES

- Survey: **Webex CC 2025**
- System defined GlobalVariable: **Global_FeedbackSurveyOptIn**

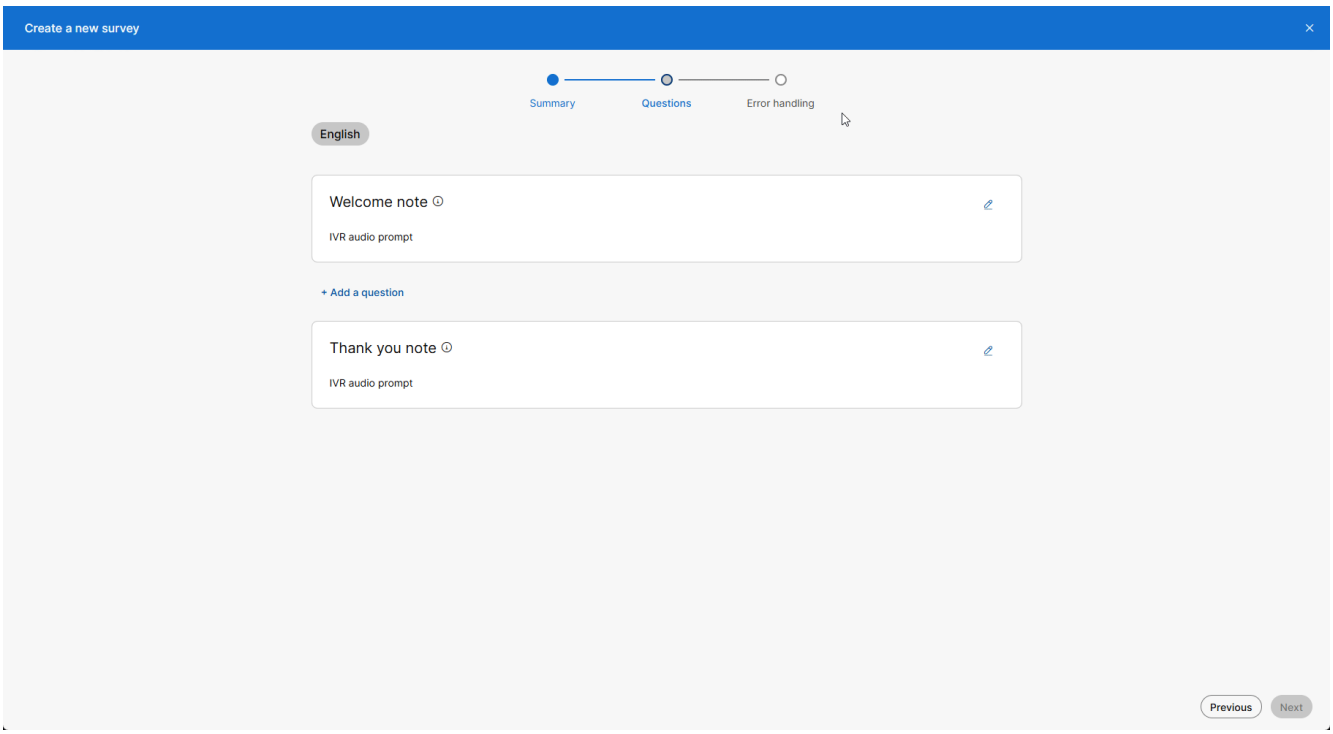
[Optional] In case you don't want to use preconfigured Survey you can configure your own. Expand below section to create your own Survey otherwise proceed to **Build** section below

eate your own Survey [Optional]

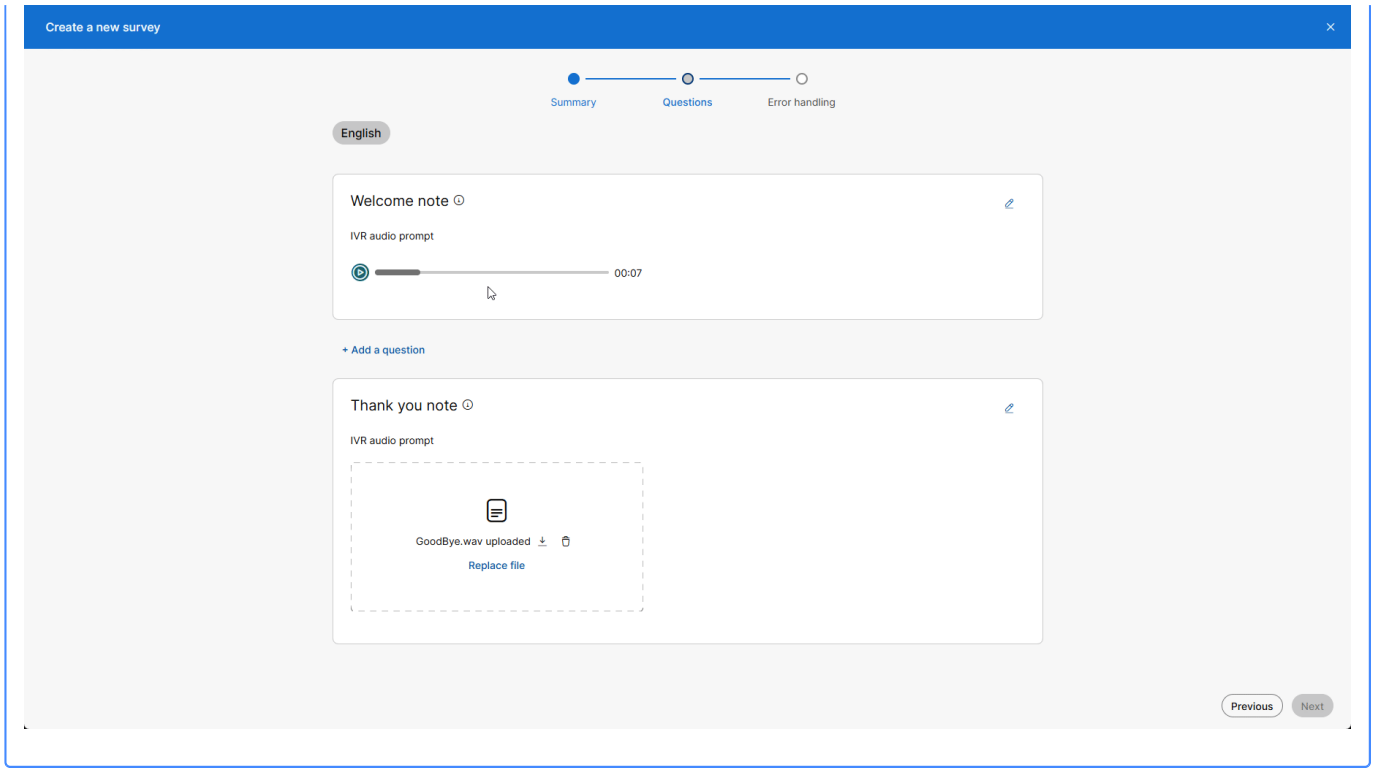
- Download audio prompts from the shared folder.
- In **Control Hub -> Contact Center** open a **Survey** configuration page under **Customer Experience**. Then click **Create new survey**.
- Enter survey name as **PCS_Your_Attendee_ID** in **Survey name** field. Make sure **IVR survey** is selected. Then click next.



- Edit **Welcome note** and **Thank you note** by uploading respective audio prompts to the survey.

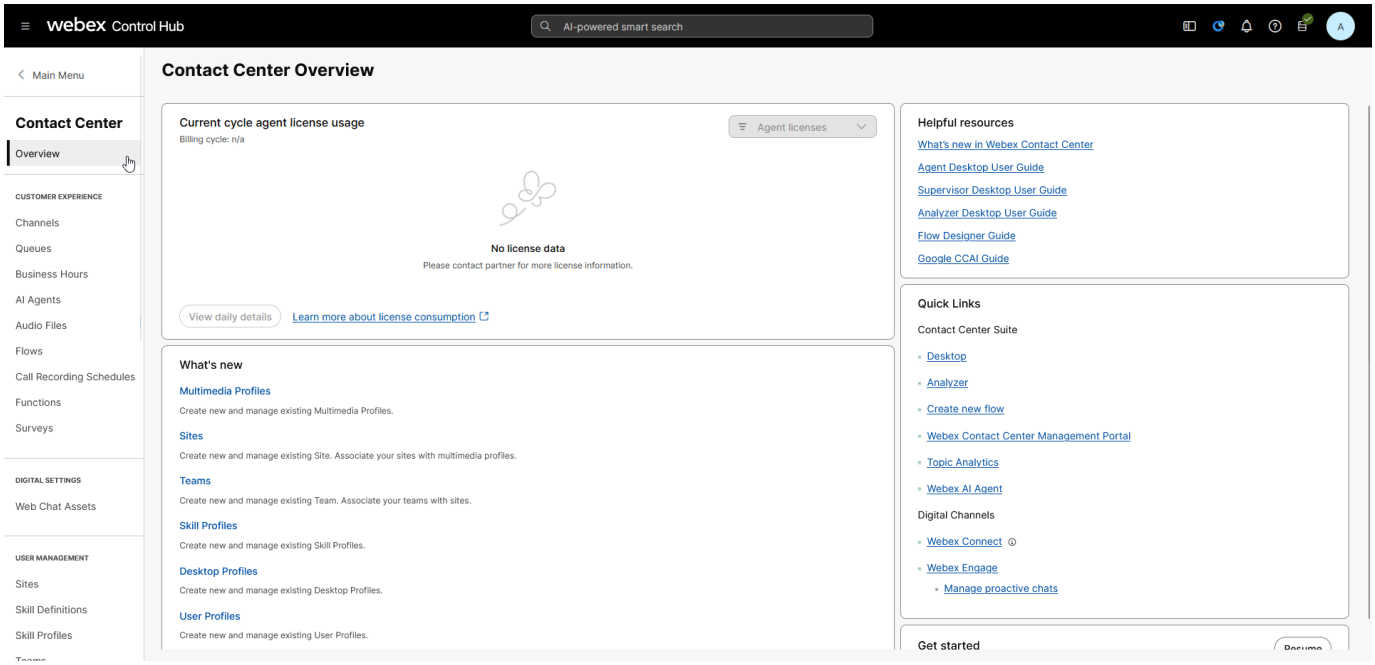


- Click on **Add a question** which is in the middle between **Welcome note** and **Thank you note**. Choose either NPS, CSAT or CES type of question and upload respective audio prompt to the survey.
- Add more questions if you want.
- Click **Next**. You can ignore **Error Handling** configuration page. Click **Save**.

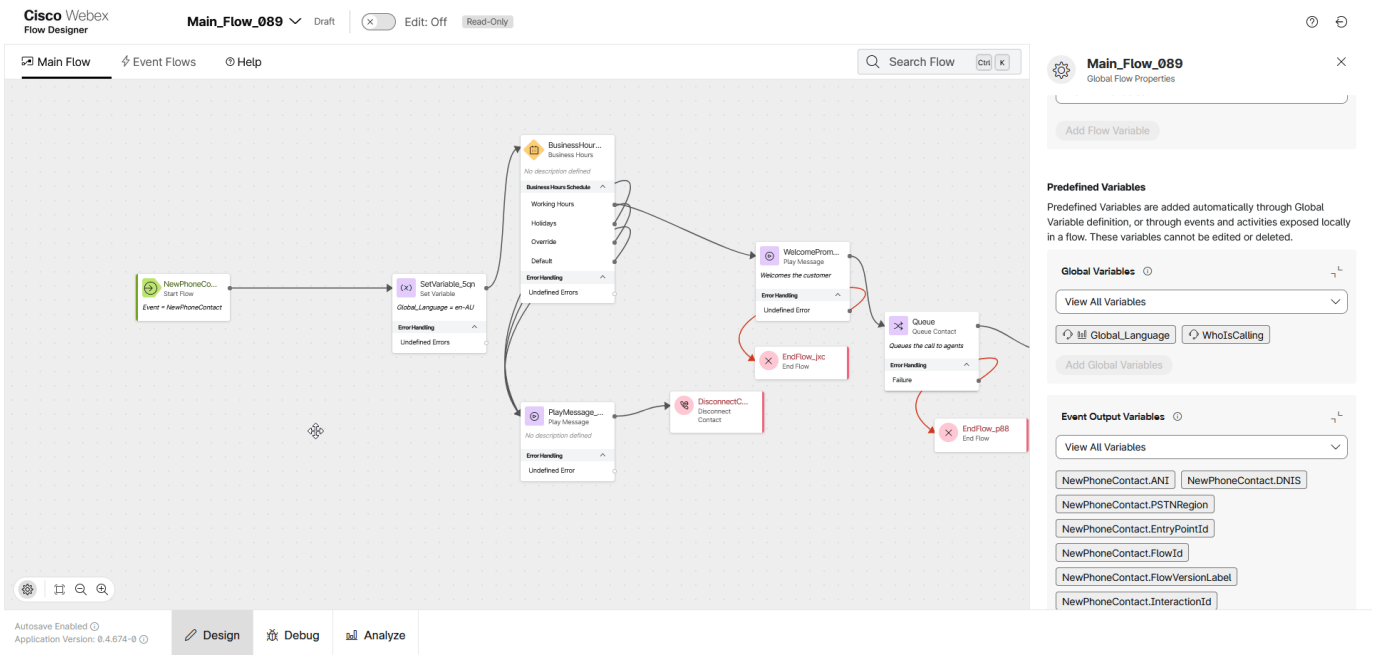


Build

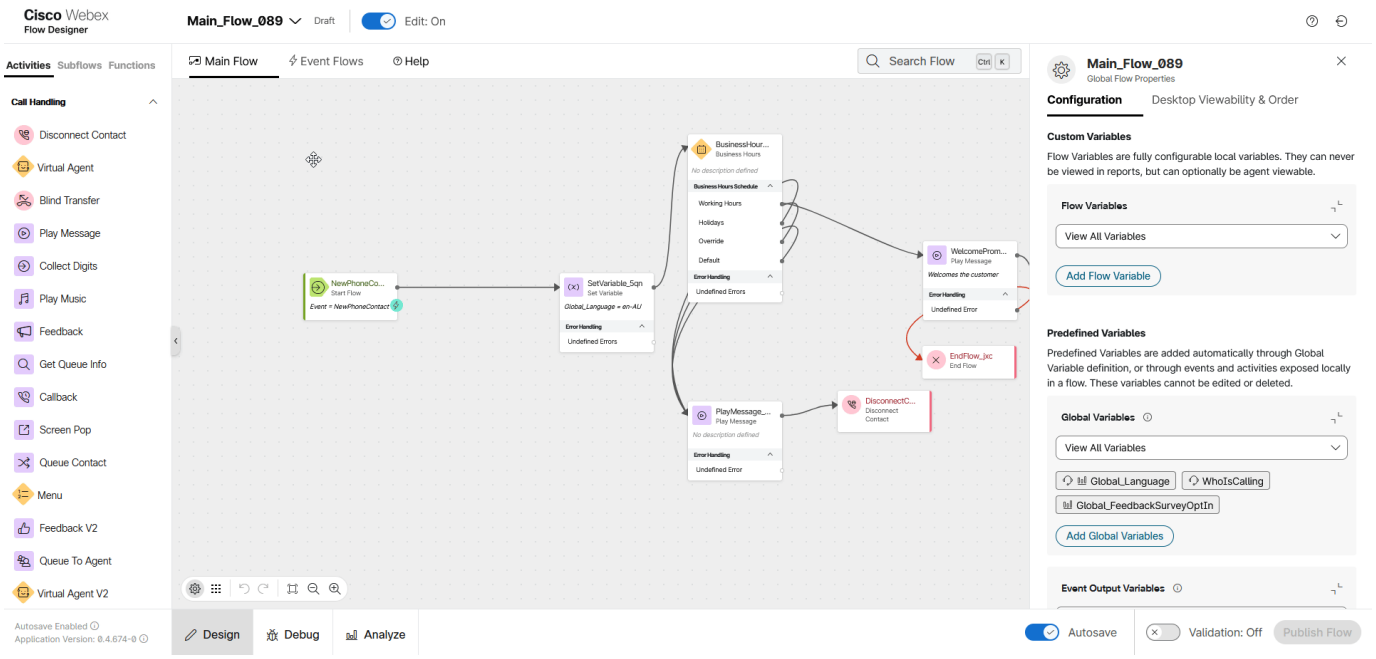
1. Switch to the Control Hub then go to **Contact Center**. Navigate to the **Surveys** under the **Customer Experience** section. Locate **Webex CC PCS** survey and click on it to familiarise yourself with its configuration.



1. Switch to the Flow Designer. Open your **Main_Flow_Your_Attendee_ID**, make sure **Edit** toggle is **ON**.
2. Add Global Variable **Global_FeedbackSurveyOptIn** to your flow.

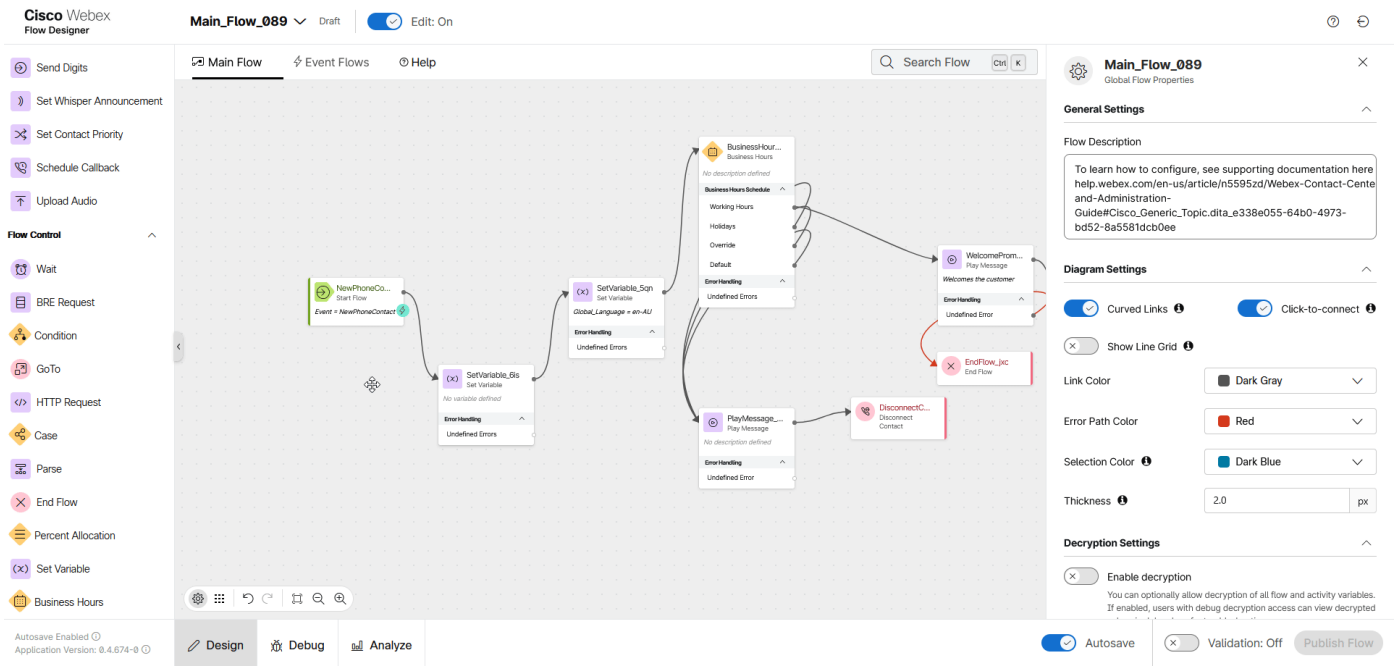


1. Delete the connection between the **NewPhoneContact** node and the first **Set Variable** node we used to set language preference. Then drag new **Set Variable** node from the activity library on the left to flow canvas, put it between **NewPhoneContact** and existing **Set Variable** nodes and connect all three nodes into a chain.



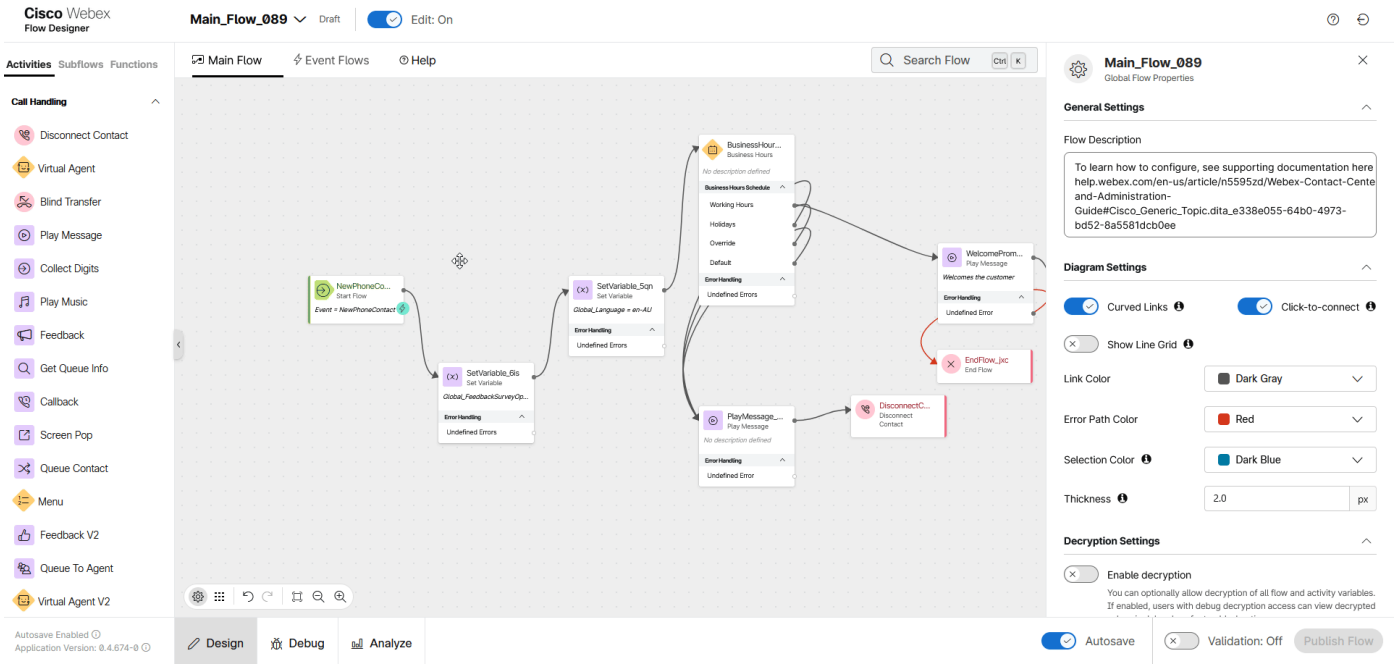
1. Click on the new **Set Variable** node you have just added and configure the following fields:

- Variable: **Global_FeedbackSurveyOptIn**
- Set Value: **true**



2. Add a Post Call Survey functionality to the flow:

- Open **Event Flows** tab and locate predefined **AgentDisconected** event node marked with light green crossed-out headset icon. If you completed previous mission you should have **HTTP Request** node connected to it.
- Delete the connection between **HTTP Request** and **DisconnectContact** nodes.
- Drag **Feedback V2** from the activity library on the left, place it between **HTTP Request** and **DisconnectContact** nodes and connect all three nodes into a chain.
- Click on **Feedback V2** node and configure Survey Method as **Voice Based** and select **Webex CC PCS** from the dropdown list.



3. Let's configure the voice message that will be played to the caller if something goes wrong with **Webex CC PCS** survey:

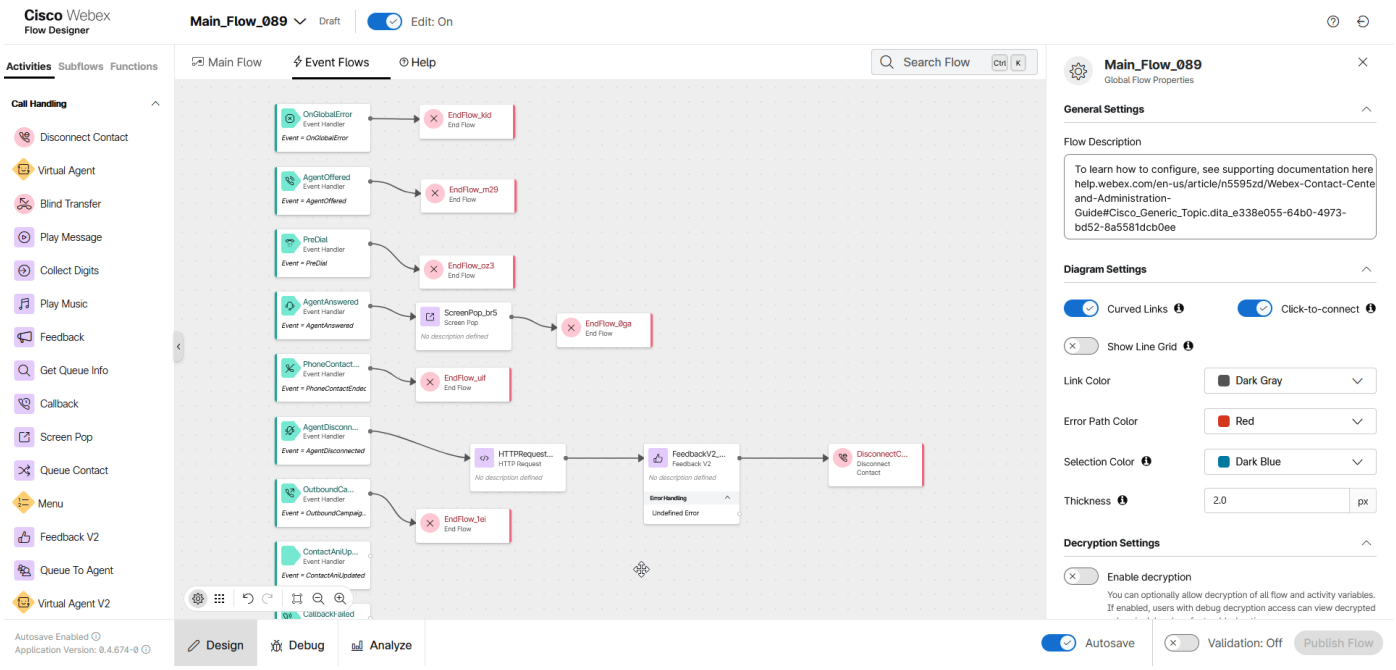
- Drag **Play Message** node from the activity library and place it below **Feedback V2** node you have just added.
- Connect **Undefined Error** output of the **Feedback V2** node to the input of the **Play Message** node
- Connect the output of the **Play Message** node to the **DisconnectContact** node.

Then click on the **Play Message** node and configure the following fields:

- Enable Text-To-Speech
- Connector: Cisco Cloud Text-to-Speech
- Click the Add Text-to-Speech Message button and paste text: ***Something went wrong on Feedback node. Please call later.***
- Delete the selection for Audio File

4. Validate and publish the flow:

- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.



Testing

1.

Your Agent desktop session should be still active but if not, use Webex CC Desktop application and login with agent credentials you have been provided **wxcclabs+agent_IDYour_Attendee_ID@gmail.com** and become **Available**

2. Make a test call to the Support Number and accept the call by Agent.
3. Finish the call by Agent, so the caller could stay on the line.
4. Now the caller should hear prompts configured in **Webex CC PCS**. Complete the survey.
5. To check survey responses, switch to the **Control Hub** and navigate to the **Surveys** under **Customer Experience** section. Locate the **Webex CC PCS** survey and click on the **Download** button on the right-hand side to download a CSV file with the provided Survey responses.

Note

If you create your own survey, as described in the Optional section of this mission, you might not see the survey responses immediately, as there is a delay in edited surveys.

Congratulations, you have succesfully completed Post Call Survey mission! 🎉

2.1.7 Mission 7: Subflow and Variable Override

Story

In Webex Contact Center, subflows modularize complex workflows by packaging reusable, independent functions (like business hour checks, queue treatment callback collection, error handling, etc.) into separate mini-flows, which are then called from main flows. Their purpose is to simplify development, reduce main flow size, improve organization, and ensure consistency by promoting code reuse for common tasks, making flows cleaner and easier to manage. In the scope of this mission you will use a subflow template to automate queue treatment in the Webex Contact Center environment. You will configure and test two ways of how to define parameters necessary to call the sublow - define them at the main flow level and override at the channel level. The main goal is to improve the customer experience while waiting in a queue by playing music, delivering messages, and looping until a predefined condition is met.

Call Flow Overview

1. A new call enters the flow.
2. The flow executes the logic and places the call into a queue.
3. Once the call is placed in the queue, the main flow calls the Subflow sending few parameters to it.
4. The Subflow executes the following logic **two times** based on the received parameters:

- Plays a music in queue. The duration is defined by the parameter.
- Plays a Text-To-Speech message. The text is fetched from the parameter.
- Plays a music in queue of the same duration again.

Mission Details

Your mission is to:

1. Create a Subflow from the template.
2. Call the Subflow from the main flow with proper parameters.
3. Override the Subflow calling parameters at the Channel level.
4. Make test calls and verify queue treatment done by the Subflow.

Part 1 - Integrate a Subflow with your Main Flow

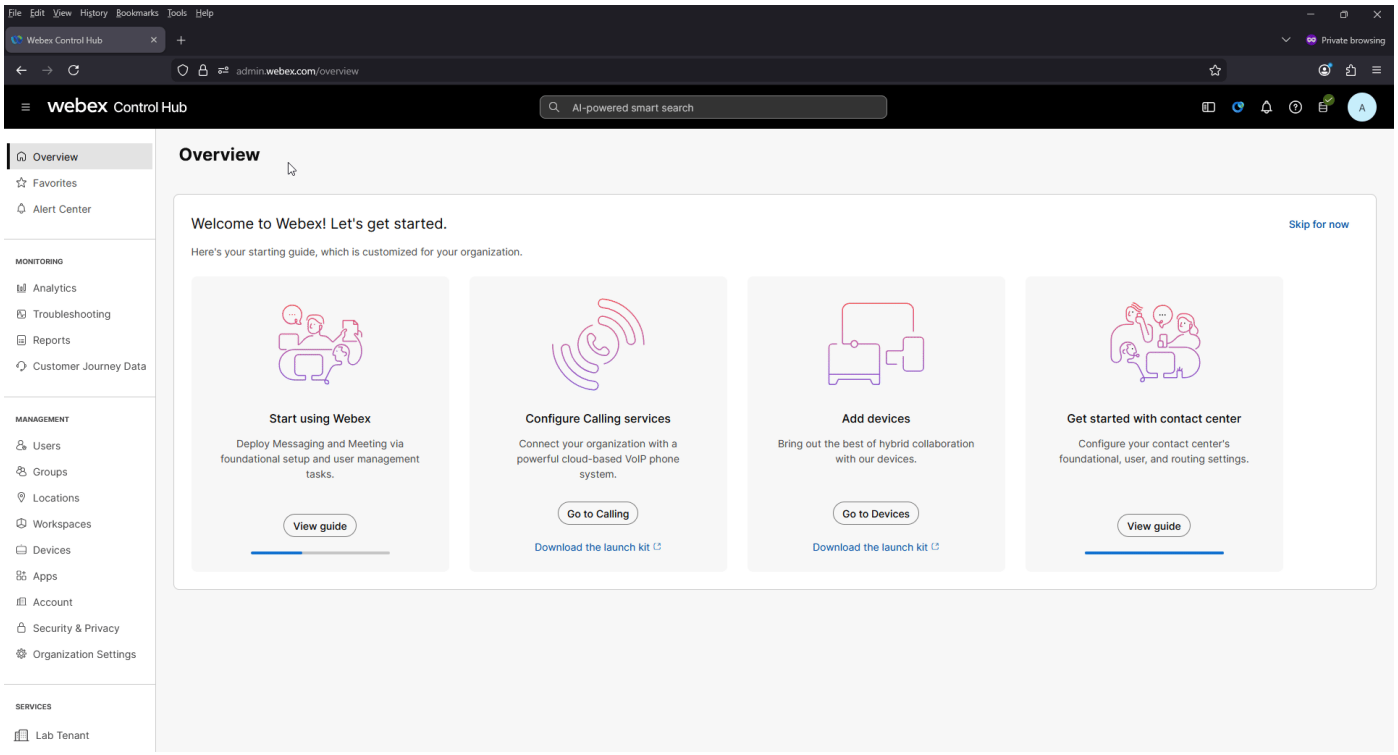
BUILD

1. Switch to Webex Control Hub. Look for the contact center option in the left pane under **SERVICES – Contact Center** and click on it.
2. Navigate to **Flows** in the left pane and click on **Subflows** tab at the top of the main window. Then click on **Manage Subflows** dropdown list at the top-right part of the main window and select **Create Subflows**.
3. **Create a new subflow** tab will be opened. Navigate to **Subflow Templates**.
4. Choose **Queue Treatment Subflow** template and click **Next**.

> **Note**
>
> You can press **View Details** link under the template name to observe flow structure and read flow description before proceeding with the template.

5. Name your subflow as **Subflow_Your_Attendee_ID** Then click on **Create Subflow**.

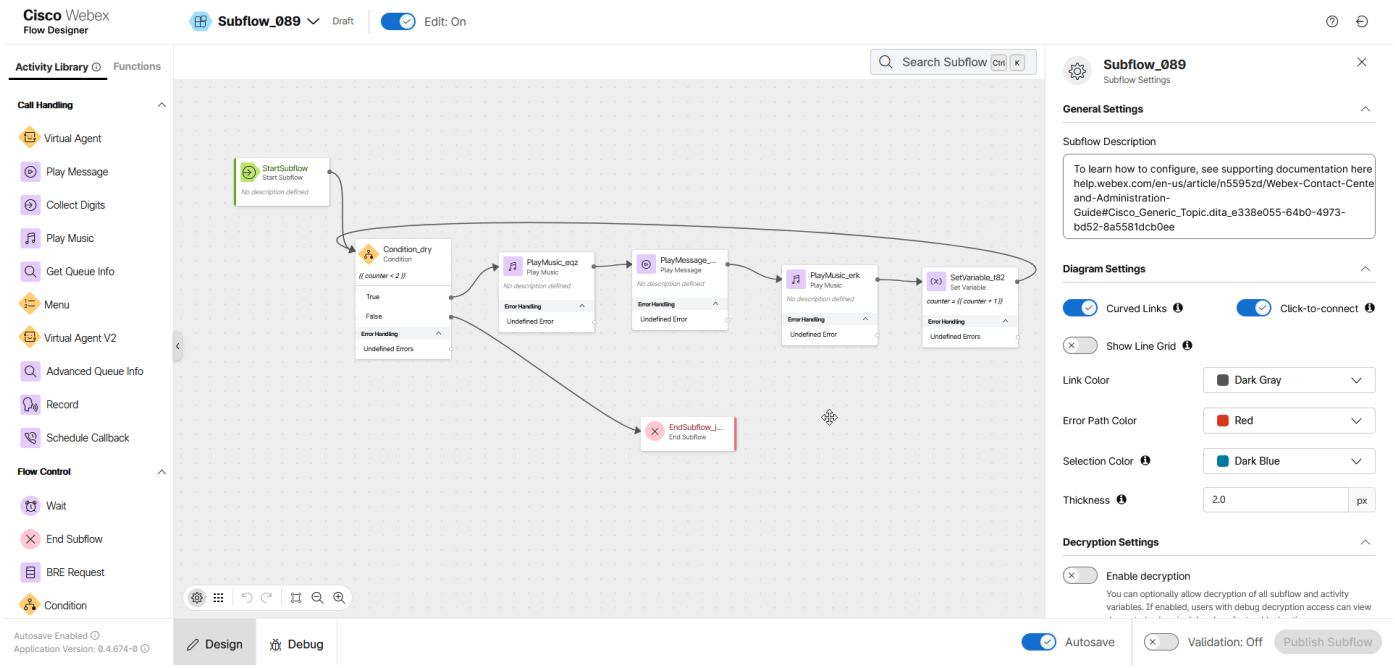
> **Note**
>
> **Edit** should be set to **On** when you create new flow, but if not switch it from **Edit: Off** mode to **Edit: On** at the top of the page.



6. In the bottom right corner toggle **Validation** from **Off** to **On** to check for any potential flow errors and recommendations.

> **Note**
>
> You can ignore recommendations but cannot skip errors.

7. Click **Publish Subflow**. In popped-up window, make sure the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Subflow**.



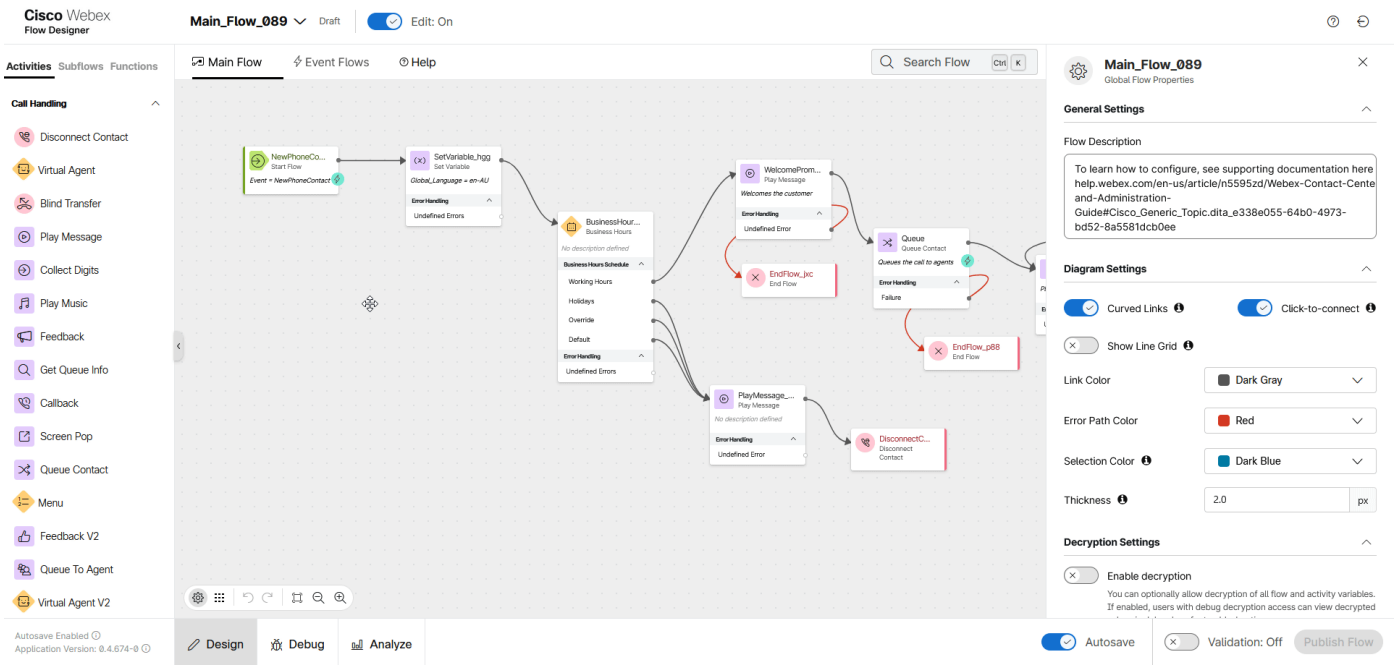
8. Switch to the Flow Designer tab with your **Main_Flow_Your_Attendee_ID** opened and make sure **Edit** toggle is **ON**.

> **Note**
>
> If you closed browser tab with Flow Designer before, switch to Webex Control Hub. Look for the contact center option in the left pane under **SERVICES – Contact Center** and click on it. Then navigate to **Flows**, search for your flow **Main_Flow_Your_Attendee_ID** and click on it to open it in the flow designer.

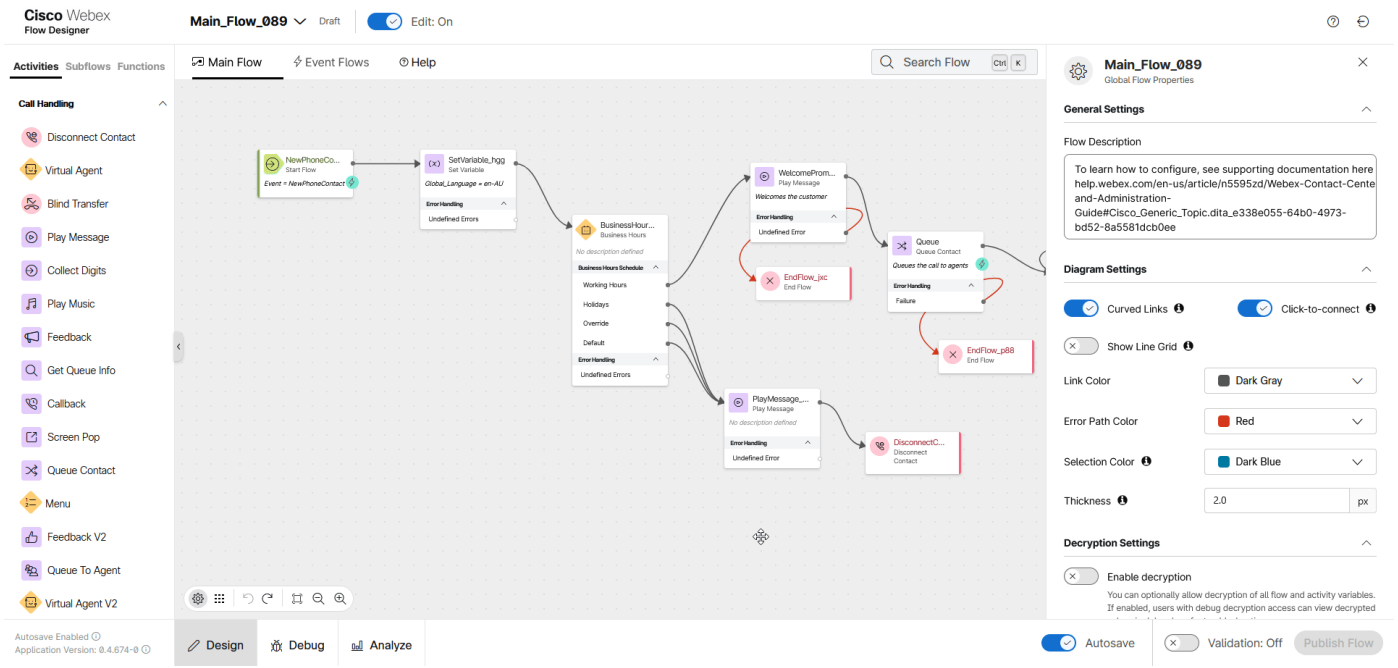
9. Click on any empty place at flow canvas and make sure you see **General Settings** appeared on the right pane. The scroll down to the **Variable Definition** section and add the following two **Flow Variables** by pressing **Add Flow Variable** button:

- Name: **subflowDuration**
- Variable Type: **Integer**
- Default Value: **5**
- Turn on the toggle **Enable External Override**. Make sure **Resource Type** is set as **None (default)**

- Name: **subflowMessage**
- Variable Type: **String**
- Default Value: **Thanks for your patience. Please hold on while we find an expert for you.**
- Turn on the toggle **Enable External Override**. Make sure **Resource Type** is set as **None (default)**



10. Delete the **Music** node connected to the **Queue** node. The **EndFlow** node which was connected to the **Music** node becomes unconnected at this step.
11. Go to the activity library pane on the left, scroll up to the top and click on the **Subflow** tab.
12. Find your subflow **Subflow_Your_Attendee_ID**, drag it to flow canvas and place right after your **Queue** node. Connect it to the other nodes in the following way:
    - The output of the **Queue** node to the input of the **Subflow_Your_Attendee_ID**.
    - The main (aka, top) output of the **Subflow_Your_Attendee_ID** node to the input of **PlayMessage** node.
    - The **Undefined Error** (aka, bottom) output of the **Subflow_Your_Attendee_ID** node to the input of unconnected **EndFlow** node.
    - The output of the **PlayMessage** node to the input of the **Subflow_Your_Attendee_ID** node.



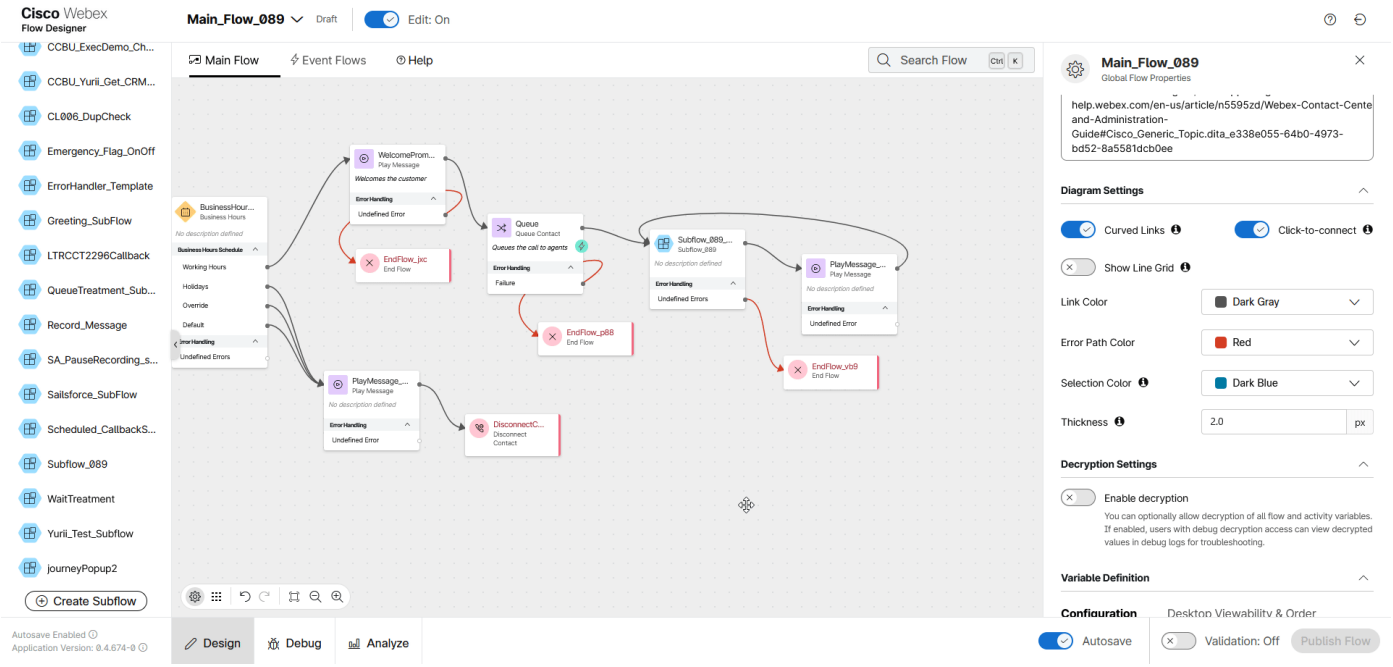
13. Click on the **Subflow_Your_Attendee_ID** node. Go to the node settings pane on the right and set the following parameters:
    - Subflow Label: **Latest**

Scroll down to the **Subflow Input Variables** section and configure the following mapping:

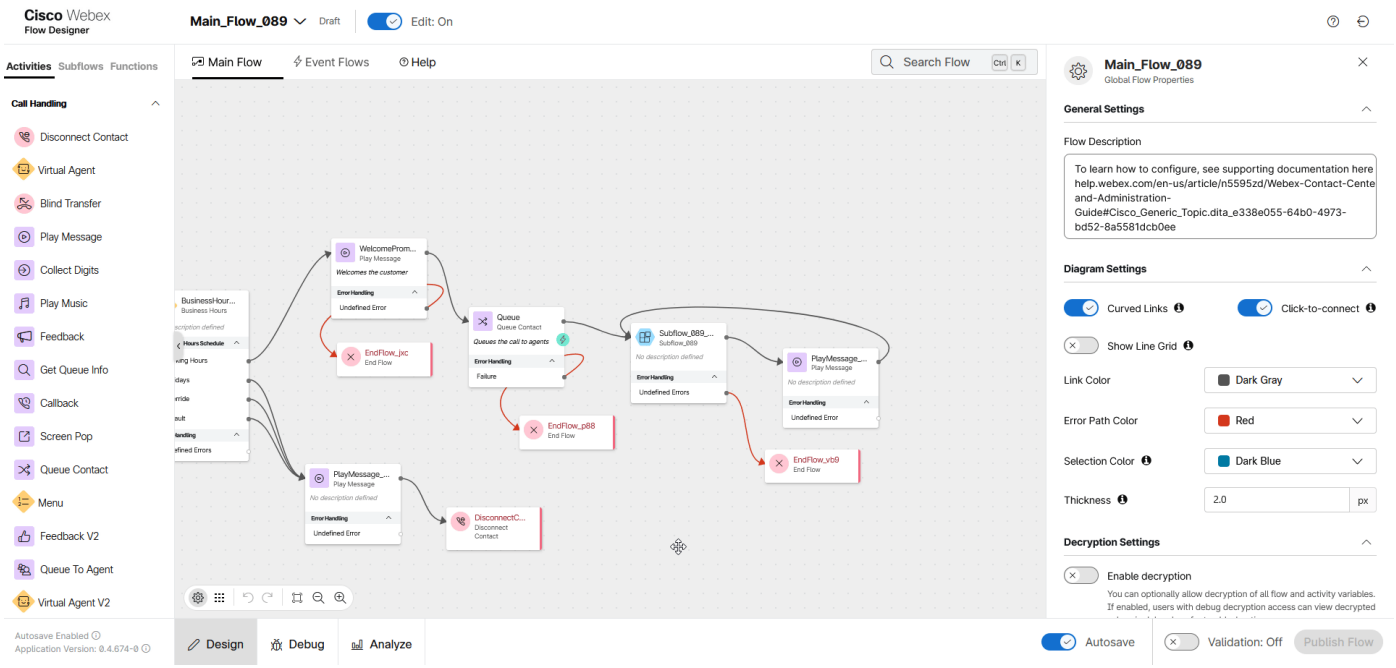
- Current flow variable: **subflowDuration**
- Subflow Input Variable: **musicDuration**

Press **Add New** button to configure the mapping for the second variable:

- Current flow variable: **subflowMessage**
- Subflow Input Variable: **queueMessage**



14. Click on the **PlayMessage** node. Go to the node settings pane on the right and paste the following message into the **Text-to-Speech Message** field:
    - ***We apologize, but all our agents are currently busy. Your waiting time may be longer than expected.***
15. Validate and publish the flow:
    - Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
    - If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
    - In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.



CHECKPOINT TEST

> **Note**
>
> Since we called the **Subflow_Your_Attendee_ID** with our custom parameters the caller should hear the following:
>
> - Repeated two times: 5 seconds of music in queue + our custom message for the subflow ***Thanks for your patience. Please hold on while we find an expert for you.*** + 5 more seconds of music in queue.
> - The message in the main flow: ***We apologize, but all our agents are currently busy. Your waiting time may be longer than expected.***
> - Then the subflow should be called again and start playing 5 seconds of music in queue again.
>
> Let's test this.

1. [IMPORTANT] Please keep in mind that the agent **does not** need to take call during this mission. Thus, if you logged in to Webex CC Desktop application, please select any Idle state there before making test calls. For example, **Busy**.
2. Make a test call to the Support Number, ensure that the caller hears all voice prompts described in the note above.
3. Finish the call.

Part 2 - Override flow variables at the Channel level

BUILD

1. Switch back to **Contact Center** settings on Webex Control Hub.
2. Navigate to **Channels** in the left pane. Search for your channel **Your_Attendee_ID_Channel** and click on it.
3. Scroll down to the **Entry point settings** section and look at **Override flow settings** part of it. You should see two variables there:

- **subDuration**
- **subMessage**

4. Modify these variables in the following way and save changes.

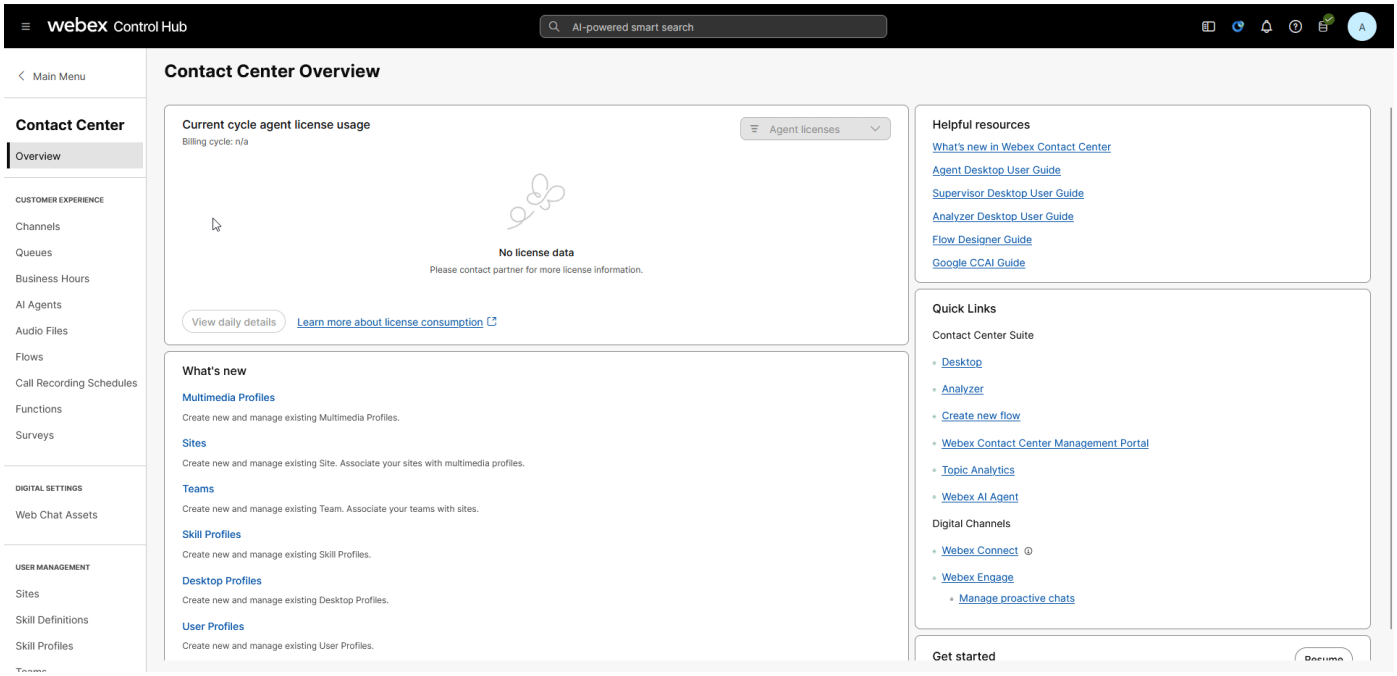
Variable **subflowDuration**:

- Turn on **Override value** toggle in the **Value** column.
- Set new value as **10**.

Variable **subflowMessage**:

- Turn on **Override value** toggle in the **Value** column.
- Set new value as ***Thanks for staying with us. Your call will be answered by the next available agent.***.

> **Note**
>
> Please keep in mind that there is **no need to validate and publish your flow** one more time after you have overridden flow variables at the channel level.



CHECKPOINT TEST

Note

Since we have overridden **subDuration** and **subMessage** variables at the channel level the **Subflow_Your_Attendee_ID** is now being called with these new values. Thus, the caller should hear updated music duration and new message:

- Repeated two times: 10 seconds of music in queue + our overrided custom message for the subflow ***Thanks for staying with us. Your call will be answered by the next available agent.*** + 10 more seconds of music in queue.
- The same message in the main flow: ***We apologize, but all our agents are currently busy. Your waiting time may be longer than expected.***
- Then the subflow should be called again and start playing 15 seconds of music in queue again.

Let's test this.

1. [IMPORTANT] Please keep in mind that the agent **does not** need to take call during this mission. Thus if you logged in to Webex CC Desktop application, please select any Idle state there before making test calls. For example, **Busy**.
2. Make a test call to the Support Number, ensure that the caller hears all voice prompts described in the note above.
3. Finish the call.

Congratulations, you have succesfully completed Subflow and Variable Override mission! 🎉🎉

2.1.8 Mission 8: Using Flow Designer Tools

Note

The current mission does not include any configuration steps, but it focuses on additional Flow Designer tools that facilitate flow troubleshooting and might provide you with ideas on how to optimize your flow logic.

Debug Overview

The Debug Tool is an essential feature in the Webex Contact Center Flow Designer, designed to simplify troubleshooting and enhance visibility into the call flow behavior. Its importance lies in its ability to provide real-time insights, enabling administrators and developers to quickly identify and resolve issues that could impact customer experience.

od to Know [Optional]

Why Debug is Important?

1. **Real-Time Analysis**: Tracks the call flow execution step by step, showing which nodes are executed and the data passed between them.
2. **Error Identification**: Quickly pinpoint errors, such as misconfigured nodes, incorrect variable usage, or unexpected call routing.
3. **Optimization**: Provides insights into flow performance, allowing you to optimize for efficiency and accuracy.

HOW TO USE THE DEBUG TOOL

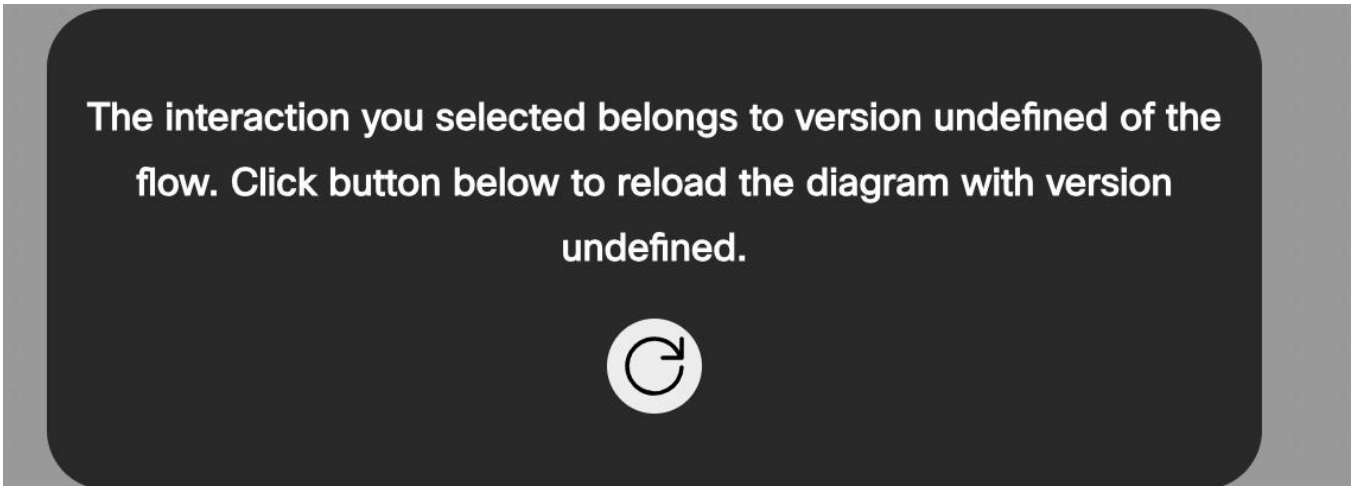
1. Switch to the Flow Designer and open your flow, **Main_Flow_Your_Attendee_ID**. Then, click the **Debug** button at the bottom of the Flow Designer.
2. You can view the calls you've made today during the previous exercises. Please click on the one at the top.

a. You can search your call by **Interaction ID**

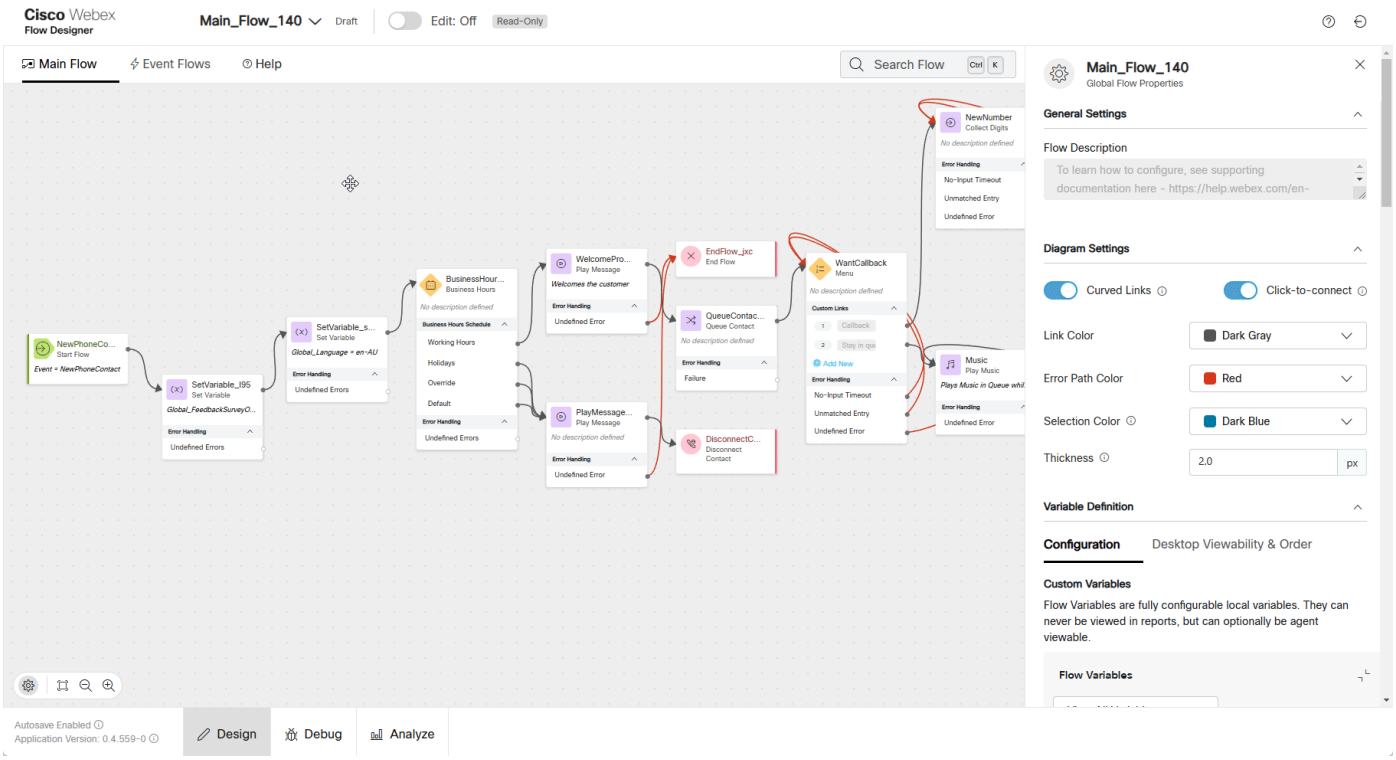
b. Filter by **Date Range** and by **Label**

Note

You might see an alert as on the following screenshot. Click refresh button to reload the diagram.



3. Observe the execution path, with visual indicators highlighting the flow nodes being executed. You can switch between Event Flows and Main Flows to see all nodes executed.
4. By clicking on each activity name you will see it's details. ***Examples: Entry point ID, Flow Label, DNIS, selected Business Hours, also TTS value and what events were triggered.***



5. Spend some time to explore the tool. Identify bottlenecks, loops, or errors if any.
6. As an option, you can break something in your flow and see how Debug tool shows that error.

By leveraging the **Debug Tool** effectively, you can ensure your call flows function as intended, providing a seamless experience for both customers and agents.

Flow Analytics Overview

Flow Analytics feature is designed to provide flow developer, administrators and supervisors with a comprehensive, graphical view of how Flow paths are being utilized across all customer interactions. This feature enables better analysis of IVR flow operations, helping to identify areas for improvement and increase self-service containment. The feature provides an aggregated view that allows users to:

- Analyze traces aggregated over a period of time.
- Visualize the aggregated data in a flow diagram, with various metrics like, average call duration, error percentage, along with some activities level metrics.
- Show interaction traces for a selected activity.
- Switch between multiple versions of analytics views.
- Color-coded links between activities based on the number of activity executions, and status.

od to Know [Optional]

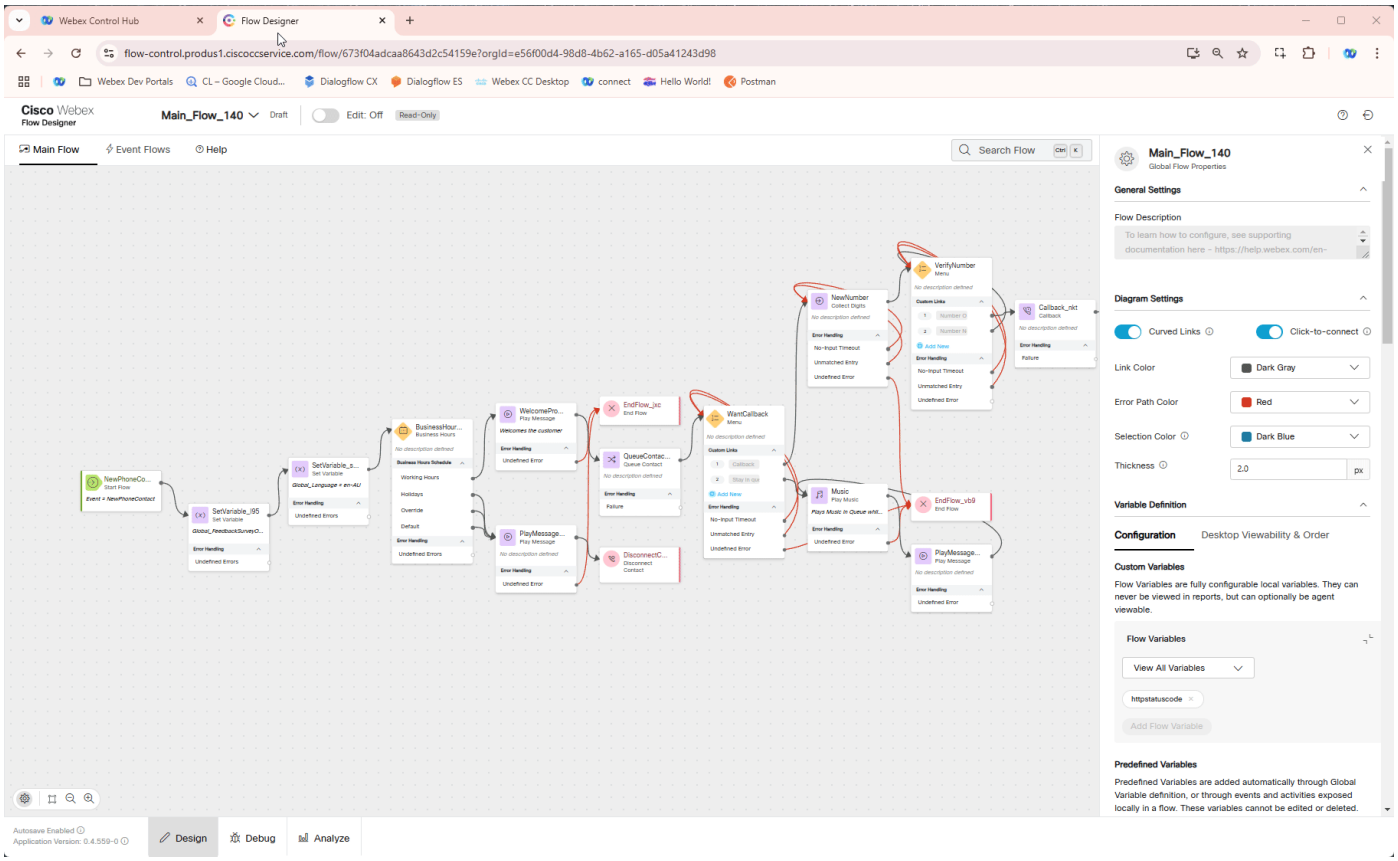
Why Flow Analytics is Important?

1. **Performance Monitoring**: Tracks key metrics, such as flow usage, execution frequency, and processing times, helping you assess flow efficiency.
2. **Behavior Analysis**: Identifies patterns in customer interactions and highlights potential issues, such as abandoned calls or potential loops.
3. **Proactive Optimization**: Offers data-driven insights to fine-tune flow configurations, ensuring optimal performance and alignment with business objectives.

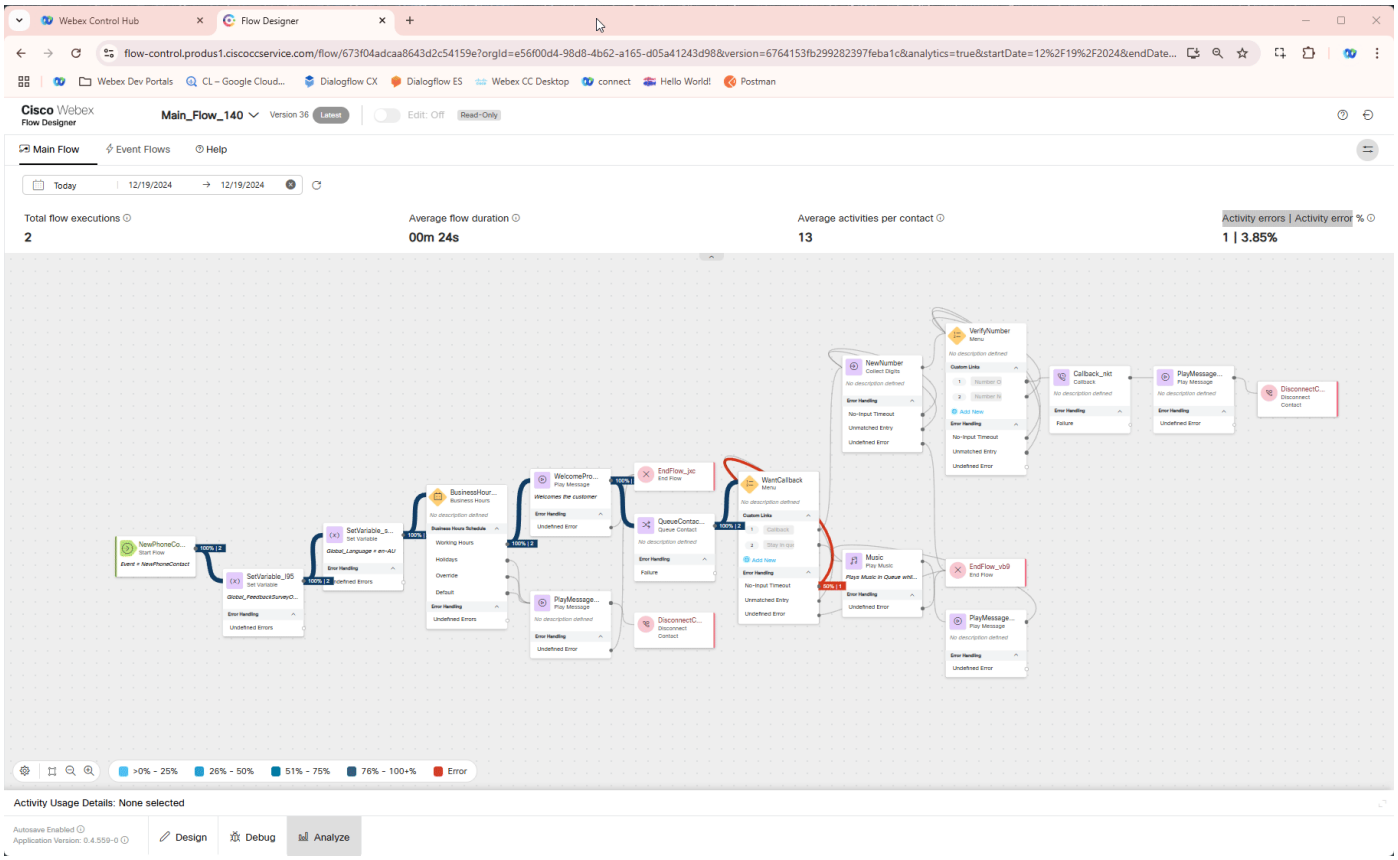
HOW TO USE THE FLOW ANALYTICS TOOL

1. On the Flow Designer, click on the **Analyze** tab at the bottom of the Flow Designer to open the Flow Analytics Tool
2. Specify a DateTime range for the report. All calls we made happened today hence select **Today** option.
3. Review visualizations and reports showing flow metrics, such as:

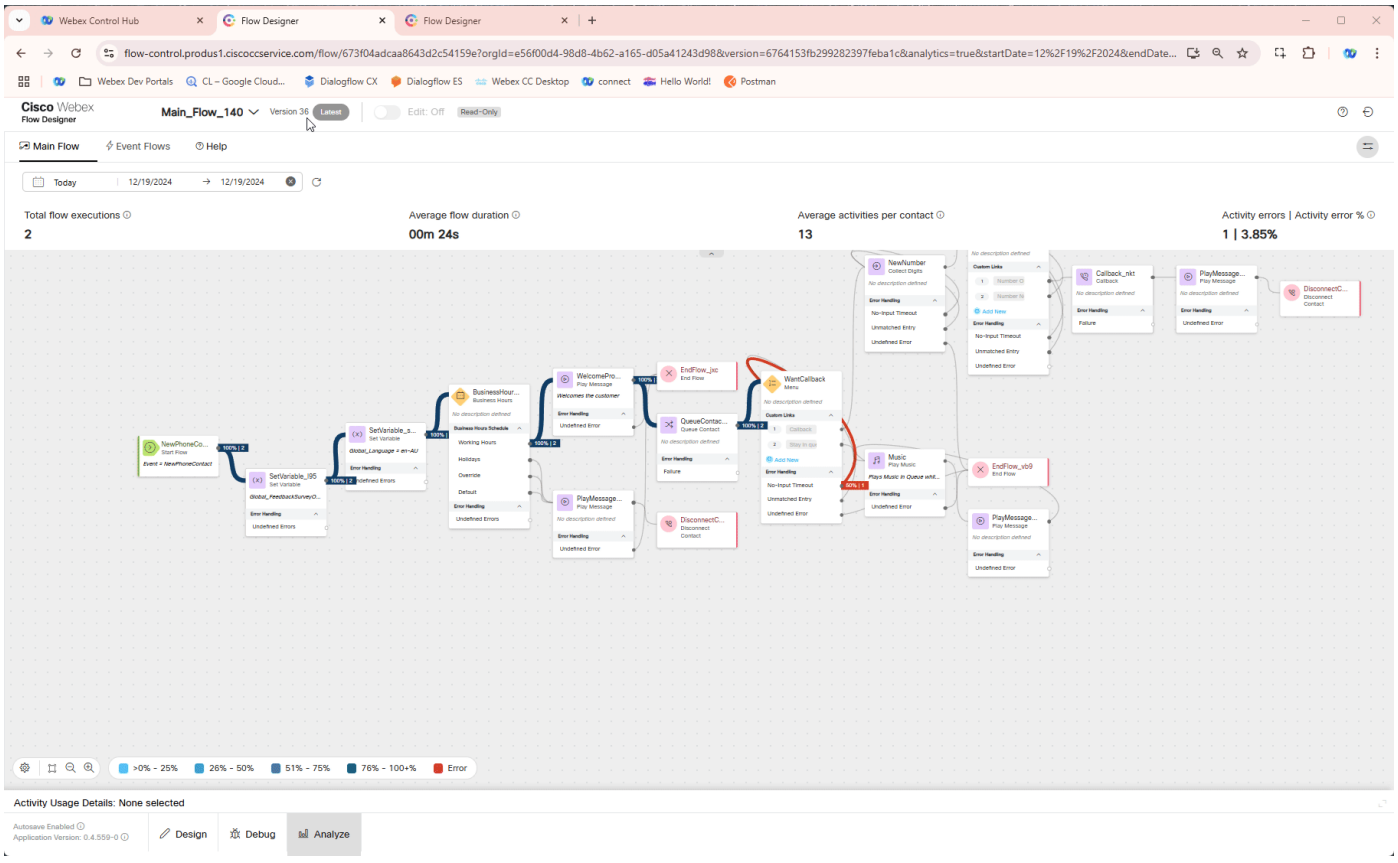
- Total flow Executions
- Execution paths and their frequency
- Average flow duration
- Average activities per contact
- Activity errors | Activity error %



4. Drill down into specific interactions by clicking on desired node.
5. If you spot any errors, click on that node. In the popped-up Activity Usage Details window, you can find call details, including Interaction ID, Start and End time, Duration, and a cross-launch link to the Debugger.



6. Observe older flow versions by selecting **Version History**. Then expand **Other Versions**. Choose anyone you like and click **View**. You might need to specify DateTime range again if the selected flow version was never executed within the chosen range in **Step 2**.



By leveraging the Flow Analytics Tool, you gain a comprehensive understanding of how your flows perform and interact with customers, enabling you to make data-backed decisions to improve both efficiency and user satisfaction.

Congratulations, you have succesfully completed Using Flow Designer Tools mission! 🎉🎉

2.2 Conclusion

We hope you enjoyed the hands-on experience of the CORE Track of Webex Contact Center lab. Throughout this lab, you've gained valuable insights into building and configuring essential elements of Webex Contact Center workflows. Starting with the Basic Call Routing mission, you explored foundational features like flow templates, Text-to-Speech (TTS) capabilities, language and voice options.

From there, you enhanced your skills with missions such as:

- **Using Business Hours** to introduce flexibility and adaptability to your flows.
- **Implementing skill-based routing** to select proper agent based on their skills and proficiency level.
- **Leveraging Event Flows** to dynamically handle real-time events.
- **Designing and implementing a Post Call Survey** to gather actionable customer feedback.
- **Embedding a subflow into the main flow** to simplify flow development, reduce main flow size making it cleaner and easier to manage.
- You also worked extensively with the Flow Designer Tools, gaining confidence in navigating, troubleshooting, and optimizing flows.

By completing these missions, you've not only developed a deep understanding of Webex Contact Center features but also acquired practical skills to harness its capabilities in real-world scenarios.

Should you need further assistance or have questions, feel free to reach out or join discussions in the Webex community. We're here to support your success as you continue to explore and implement these powerful tools in your future projects.

Thank you for participating in the CORE Track, and we encourage you to continue exploring and building on these foundational skills as you progress through the next stages of the Webex Contact Center Flow Designer Lab!

3. API TRACK

3.1 Lab Guide

3.1.1 Using Webex Contact Center Developer Portal

Story

Webex Contact Center APIs enable automation, customization, and integration with external applications. By leveraging these APIs, administrators can streamline processes, enhance agent efficiency, and improve customer interactions. In this introduction mission, we will explore how to interact with the Developer Portal and execute different types of API calls.

Mission Details

In this mission, attendees will learn how to interact with Webex Contact Center APIs by performing API calls via the **Developer Portal**. Specifically, we will work with the Address Book feature.

od to Know [Optional]

Understanding API Calls with Real-Life Comparisons

APIs (Application Programming Interfaces) allow different systems to communicate by sending and receiving structured requests. Here are the most common API call types, explained with real-world analogies:

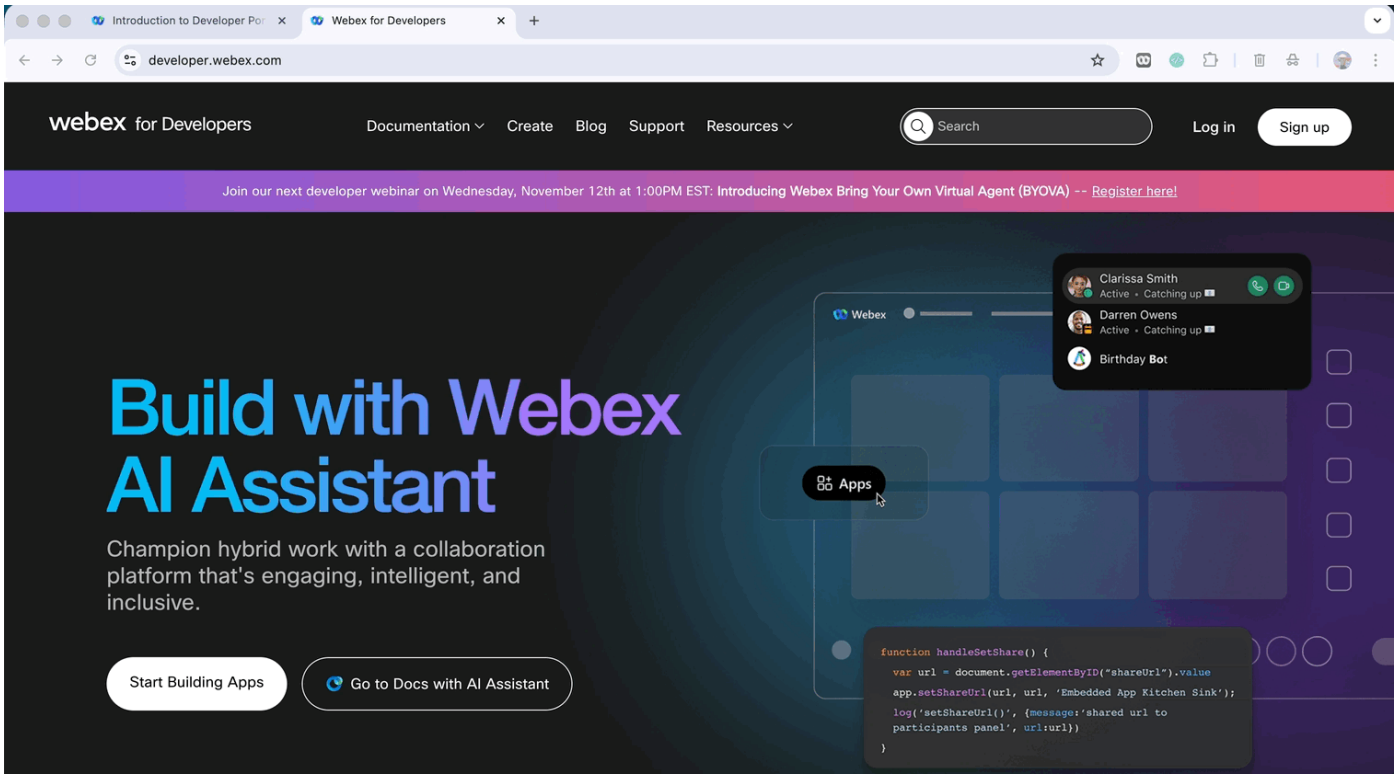
1. **GET – Retrieving Information**
   Analogy: Checking your bank balance at an ATM. You request information, and the system provides it without making any changes.
   Example Use Case: Retrieving a customer's interaction history in Webex Contact Center before routing their call.
2. **POST – Creating New Data**
   Analogy: Ordering a new item online. You submit details, and a new order (or record) is created in the system.
   Example Use Case: Creating a new customer support ticket when an issue is reported during a call.
3. **PUT – Updating Existing Data**
   Analogy: Changing your home address in an online banking system. Instead of adding a new address, the existing one is replaced.
   Example Use Case: Updating a customer's preferred contact method in a CRM system.
4. **PATCH – Modifying Partial Data**
   Analogy: Updating your phone number on a social media profile without changing other details like your name or email.
   Example Use Case: Changing only the priority level of an existing support ticket.
5. **DELETE – Removing Data**
   Analogy: Canceling a hotel reservation. The record is removed, preventing further use.
   Example Use Case: Deleting a scheduled callback request if the customer no longer needs assistance.
6. **Webhooks – Automated Notifications**
   Analogy: Receiving an SMS alert when your package is out for delivery. Instead of requesting updates repeatedly, you get notified when something happens.
   Example Use Case: Notifying an agent when a VIP customer joins the queue.
7. **SEARCH API (GraphQL Queries) – Retrieving Specific Data Efficiently**
   Analogy: Using a restaurant menu app to filter only "vegan dishes under $10" instead of browsing the entire menu. Unlike traditional GET requests that return all data, GraphQL allows users to request exactly what they need.
   Example Use Case: Searching for all unresolved support tickets assigned to a specific agent without loading unnecessary ticket details.

APIs streamline operations by automating tasks, integrating systems, and enhancing customer experiences. Understanding these core calls helps optimize workflows in platforms like Webex Contact Center.

Build

CREATE A NEW ADDRESS BOOK ENTITY BY USING POST

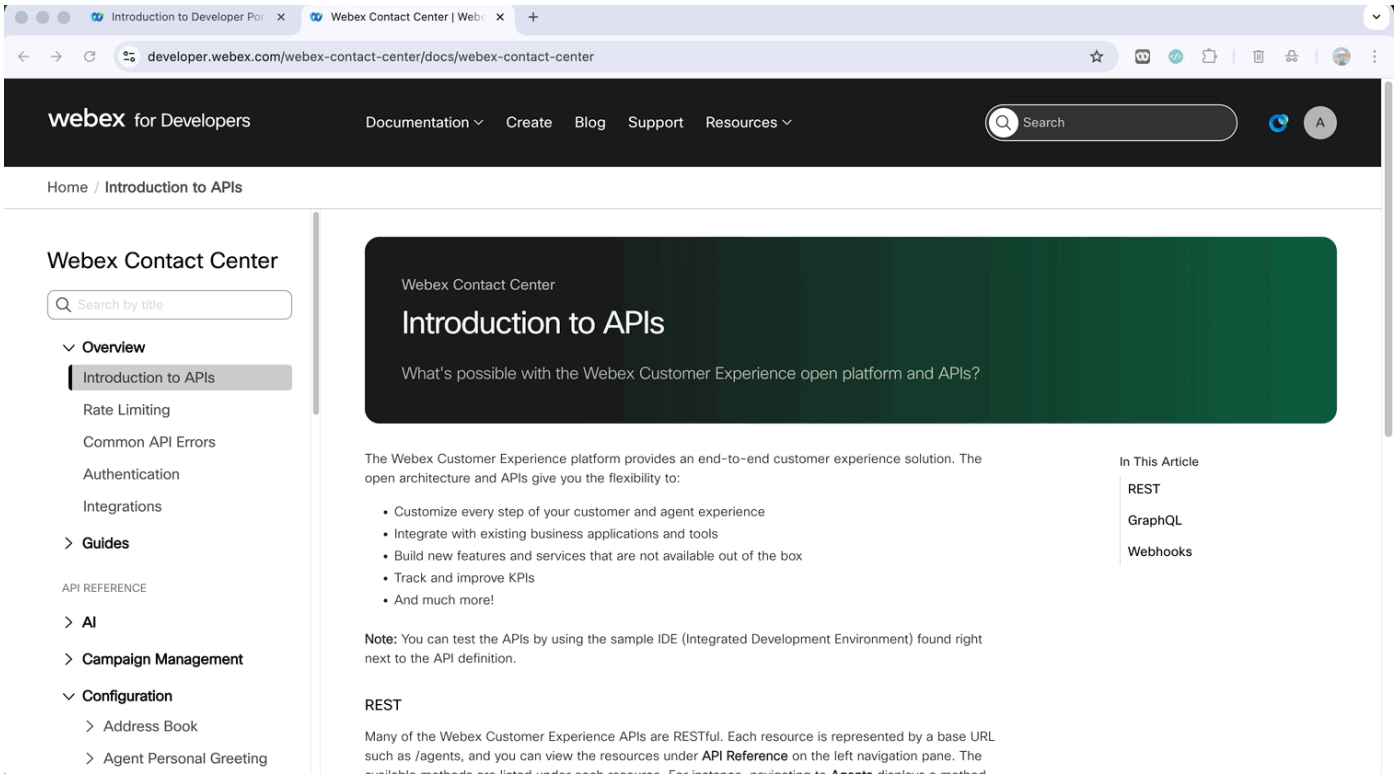
1. Open **Developer Portal** and click on **Sign In**. Your login will be of the format **wxcclabs+admin_IDYour_Attendee_ID@gmail.com**. You will see another login screen where you may need to enter the email address again and the password provided to you.
2. Click on the little arrow next to **Documentation**, choose **Webex Contact Center** under **Customer Experience** section.



3. On Menu panel on the left, scroll down to **API Reference** section and click on **Configuration** then **Adress Book**. Look at the list of available API calls.

> **Note**
>
> **Address Book Overview**
> Address Book is available in the Webex Contact Center Agent Desktop. Agents can make outbound calls using Address Books, selecting numbers from pre-configured lists instead of entering them manually in the **Start a New Call** field. Administrators can configure and manage Address Books via the Webex Contact Center APIs.

4. Scroll down and click on **Create a new Address Book**, then click on **Request Body (JSON)** on the right.



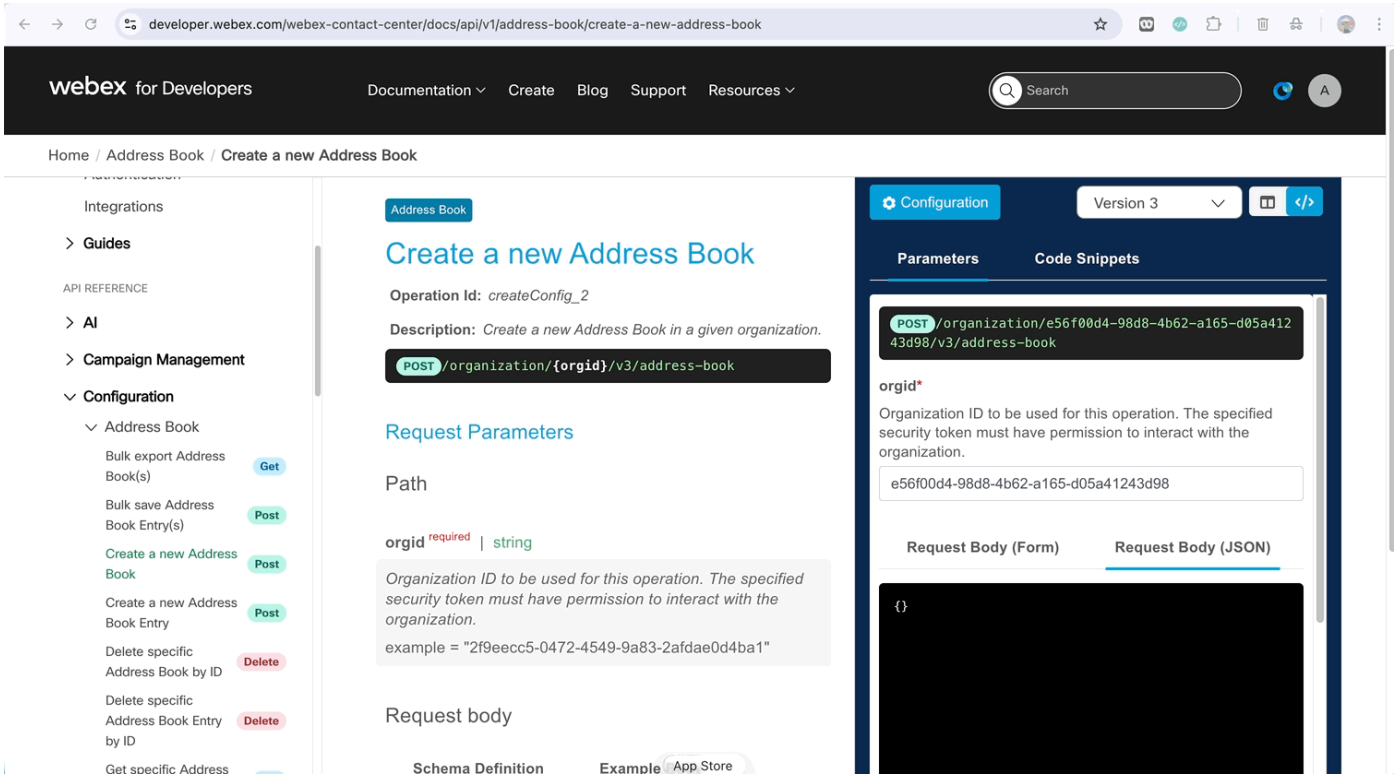
5. Clear **Request Body** content. Paste the following body and replace the **\<Your_Attendee_ID>** with your 3-digit **Attendee ID**. Then scroll down and click on **Run**.

Request Body:

```
{
    "name": "AddressBook_<Your_Attendee_ID>",
    "parentType": "ORGANIZATION"
}
```

Example of the expected response: **201 Response**

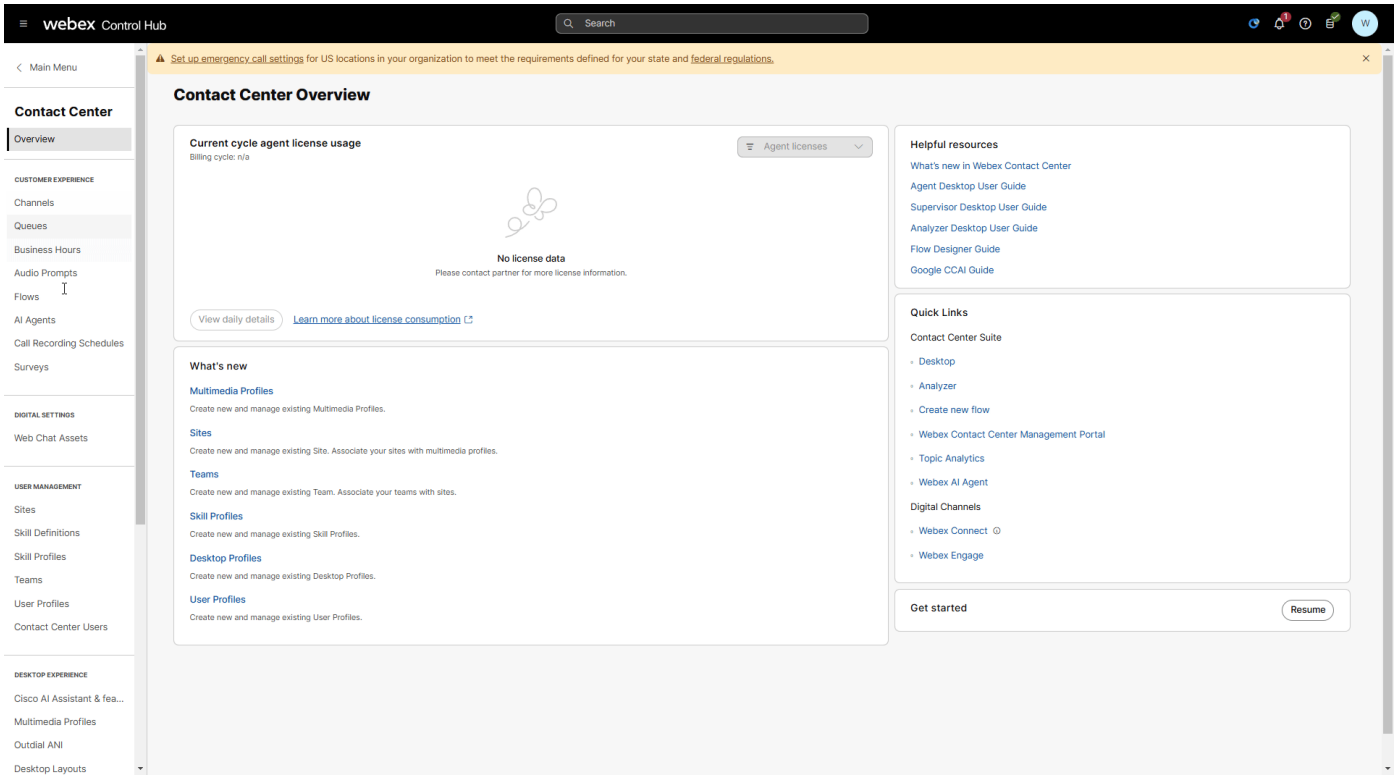
```
{
  "organizationId": "e56f00d4-98d8-4b62-a165-d05a41243d98",
  "id": "4aa50a6b-a520-4221-bc9d-a050c111061f",
  "version": 0,
  "name": "AddressBook_140",
  "parentType": "ORGANIZATION",
  "createdTime": 1738585491913,
  "lastUpdatedTime": 1738585491913
}
```



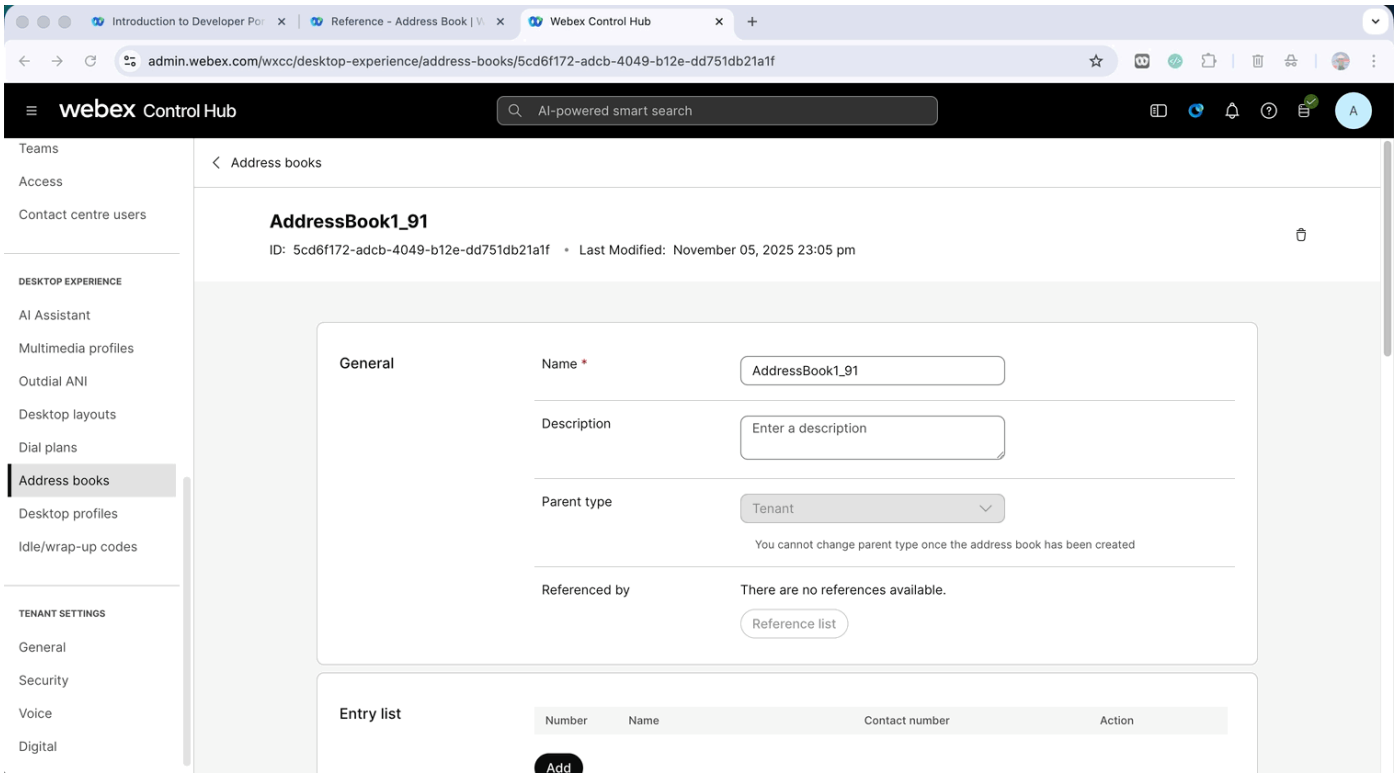
6. Switch to Webex Control Hub and navigate to **Address Book** under **Desktop Experience Section**. Locate your new created **AddressBook_Your_Attendee_ID**

7. You should see your new created **AddressBook_Your_Attendee_ID**. There are still no Address Book entries so let's add them.

8. On the same **Address Book** configuration page, copy the **AddressBook_AddressBook_ ID** into notepad.



9. Switch to **Developer Portal** and select **Address Book** again from left menu pane.



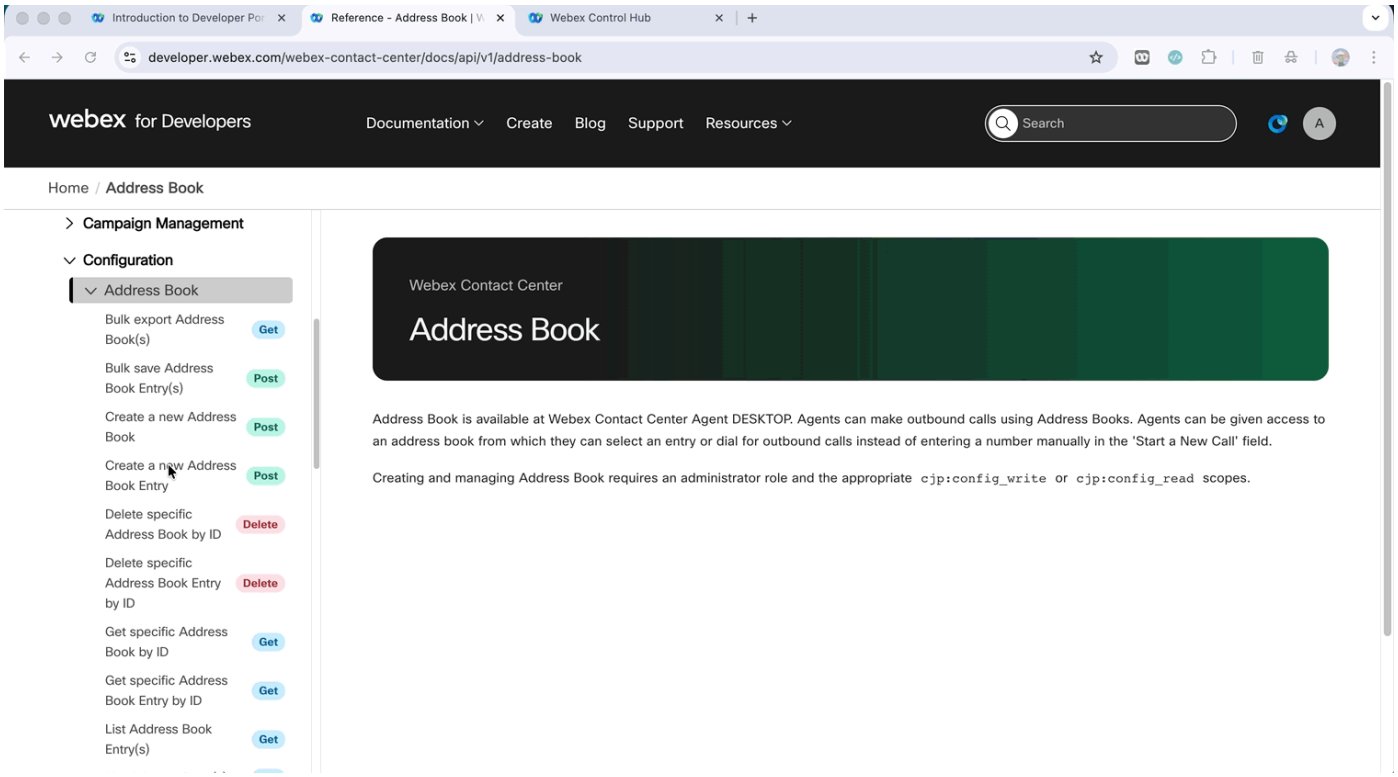
10. Click on **Create a new Address Book Entry**, then click on **Request Body (JSON)** on the right.
11. In the **Parameters** section paste **addressBookId** you copied on **Step 8** of the current mission.
12. Clear **Request Body** content and paste the following body, then scroll down click on **Run** button.

    Request Body:

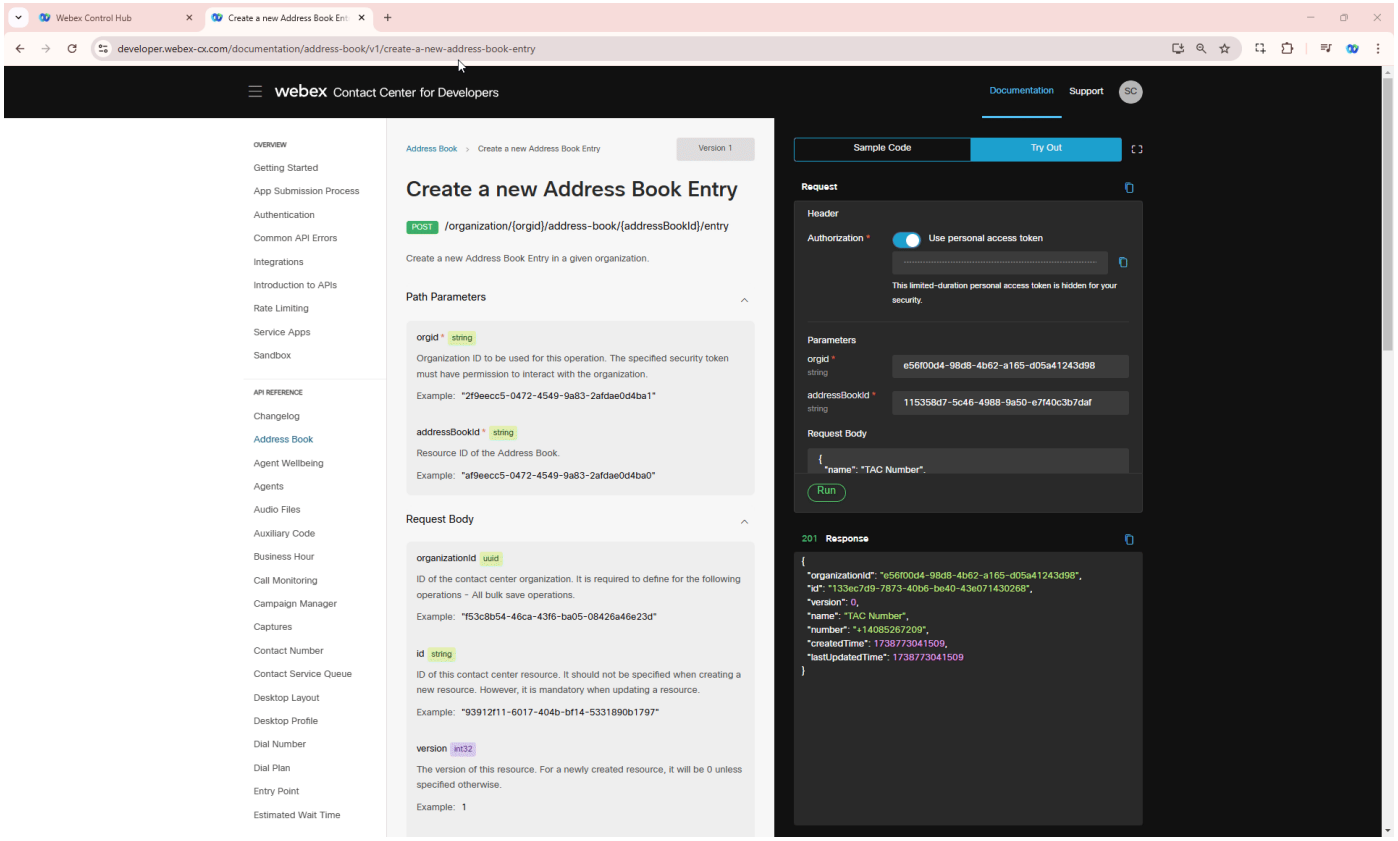
```
{
  "name": "TAC Number",
  "number": "+14085267209"
}
```

    Example of the expected Response: **201 Response**

```
{
  "organizationId": "e56f00d4-98d8-4b62-a165-d05a41243d98",
  "id": "133ec7d9-7873-40b6-be40-43e071430268",
  "version": 0,
  "name": "TAC Number",
  "number": "+14085267209",
  "createdTime": 1738773041509,
  "lastUpdatedTime": 1738773041509
}
```



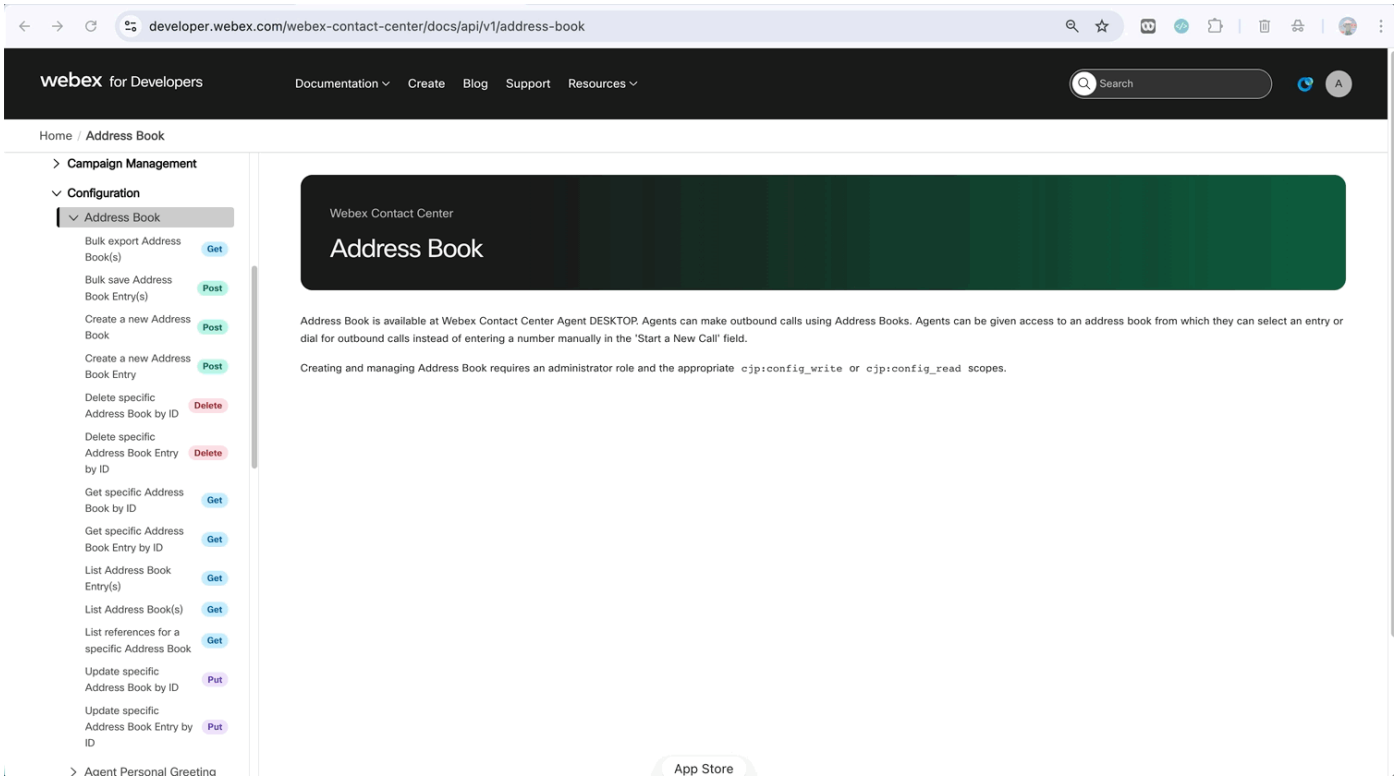
13. Switch to Webex Control Hub. Open the configuration page of your **AddressBook_Your_Attendee_ID** and **refresh** the page.
14. You should see your new created **Entry List** with Name **TAC Number** and Contact Number **+14085267209**.



RETRIEVE ADDRESS BOOK ENTRY BY ID (GET)

We will retrieve information about your newly created address book using a GET API call.

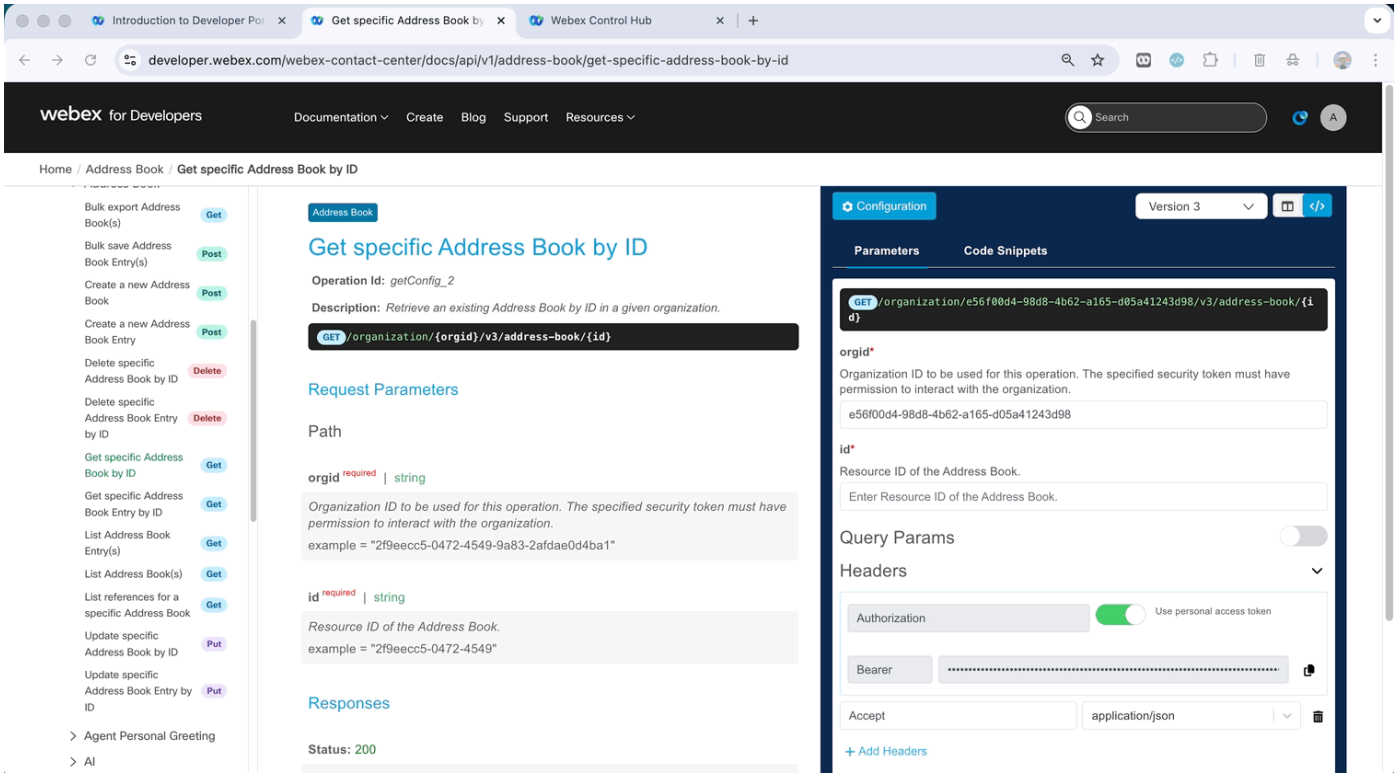
1. Switch to **Developer Portal** and select **Address Book** again from left menu pane.
2. Locate and open **Get specific Address Book by ID**, then switch to **Parameters** section on the right.



3. Paste the same **AddressBook_AddressBook_ ID** into **id** field of the **Parameters** section. You can quickly copy it by switching back to Control Hub. Then click **Run** at the bottom.

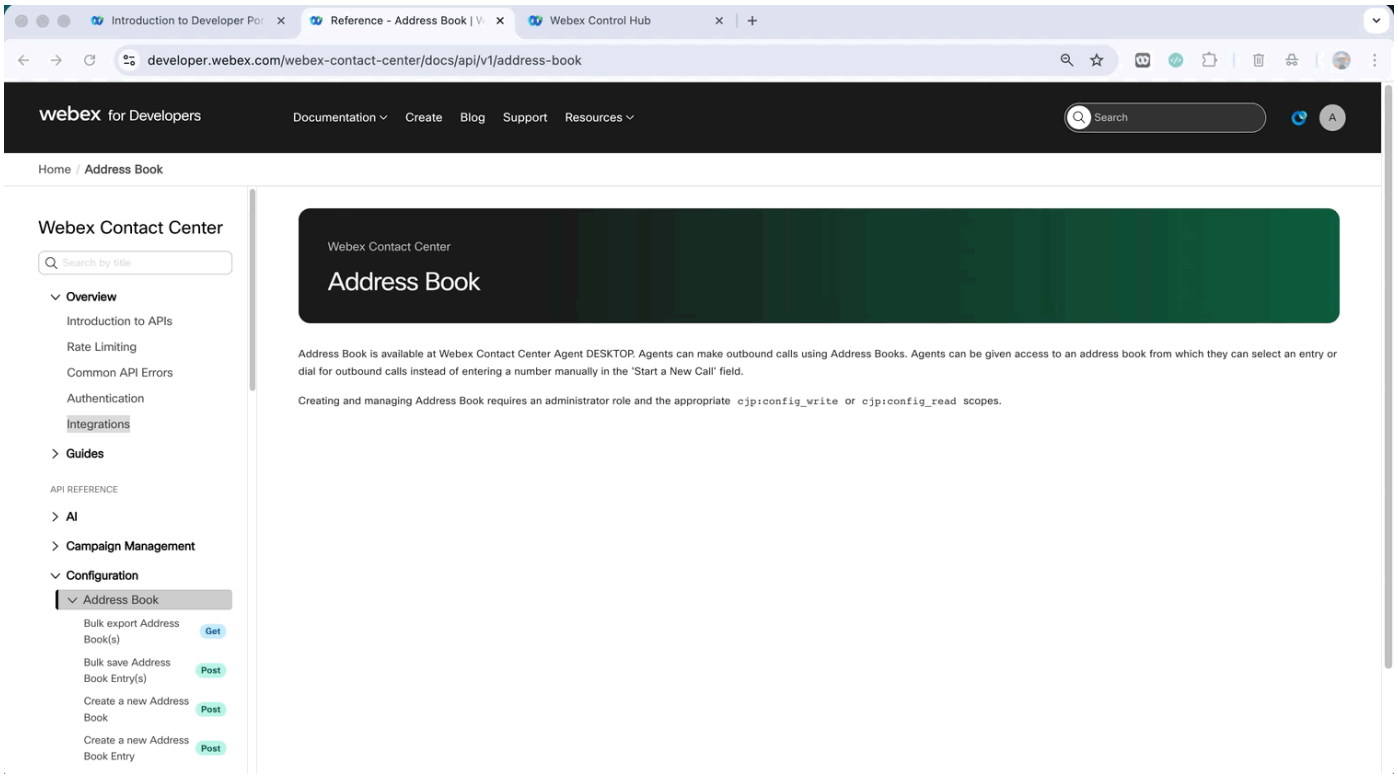
   Example of the expected Response: **200 Response**


```
{
  "id": "115358d7-5c46-4988-9a50-e7f40c3b7daf",
  "name": "AddressBook_140",
  "description": "",
  "parentType": "ORGANIZATION",
  "createdTime": 1738771074000,
  "lastUpdatedTime": 1738773007000
}
```

UPDATE ADDRESS BOOK DESCRIPTION

1. Switch to **Developer Portal** and select **Address Book** again from left menu pane.
2. Locate and open **Update specific Address Book by ID**, then switch to **Parameters** section on the right.



3. Paste the same ID of your **AddressBook_AddressBook_** into **id** cell of **Parameters** section. You can quickly copy **ID** of your address book by switching back to Control Hub.
4. Clear **Request Body** content and paste the following body. Then click **Run**.

   Replace value ***YourAddressBook_Name*** to your **AddressBook_Your_Attendee_ID**

   Replace ***YouAddressBook_ID*** to actual **ID** of your AddressBook_AddressBook_

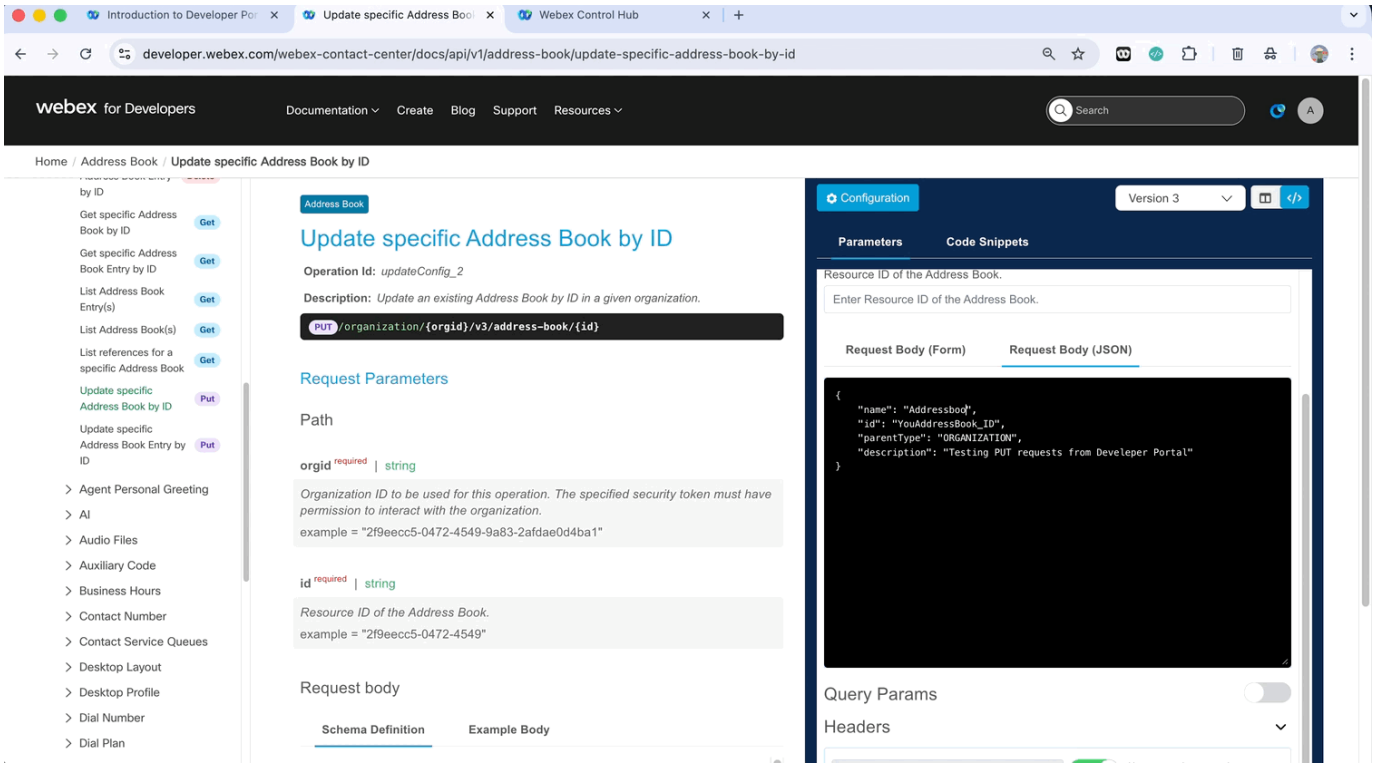
   Request Body:

```
{
  "name": "YourAddressBook_Name",
  "id": "YouAddressBook_ID",
  "parentType": "ORGANIZATION",
  "description": "Testing PUT requests from Developer Portal"
}
```

   Example of the expected Response: **200 Response**

```
{
  "organizationId": "e56f00d4-98d8-4b62-a165-d05a41243d98",
  "id": "115358d7-5c46-4988-9a50-e7f40c3b7daf",
  "version": 5,
  "name": "AddressBook_140",
  "description": "Testing PUT requests from Developer Portal",
  "parentType": "ORGANIZATION",
  "createdTime": 1738771074000,
  "lastUpdatedTime": 1738775594832
}
```

5. Switch to Webex Control Hub. Open the configuration page of your **AddressBook_Your_Attendee_ID** and **refresh** the page to validate updated **Description** field.

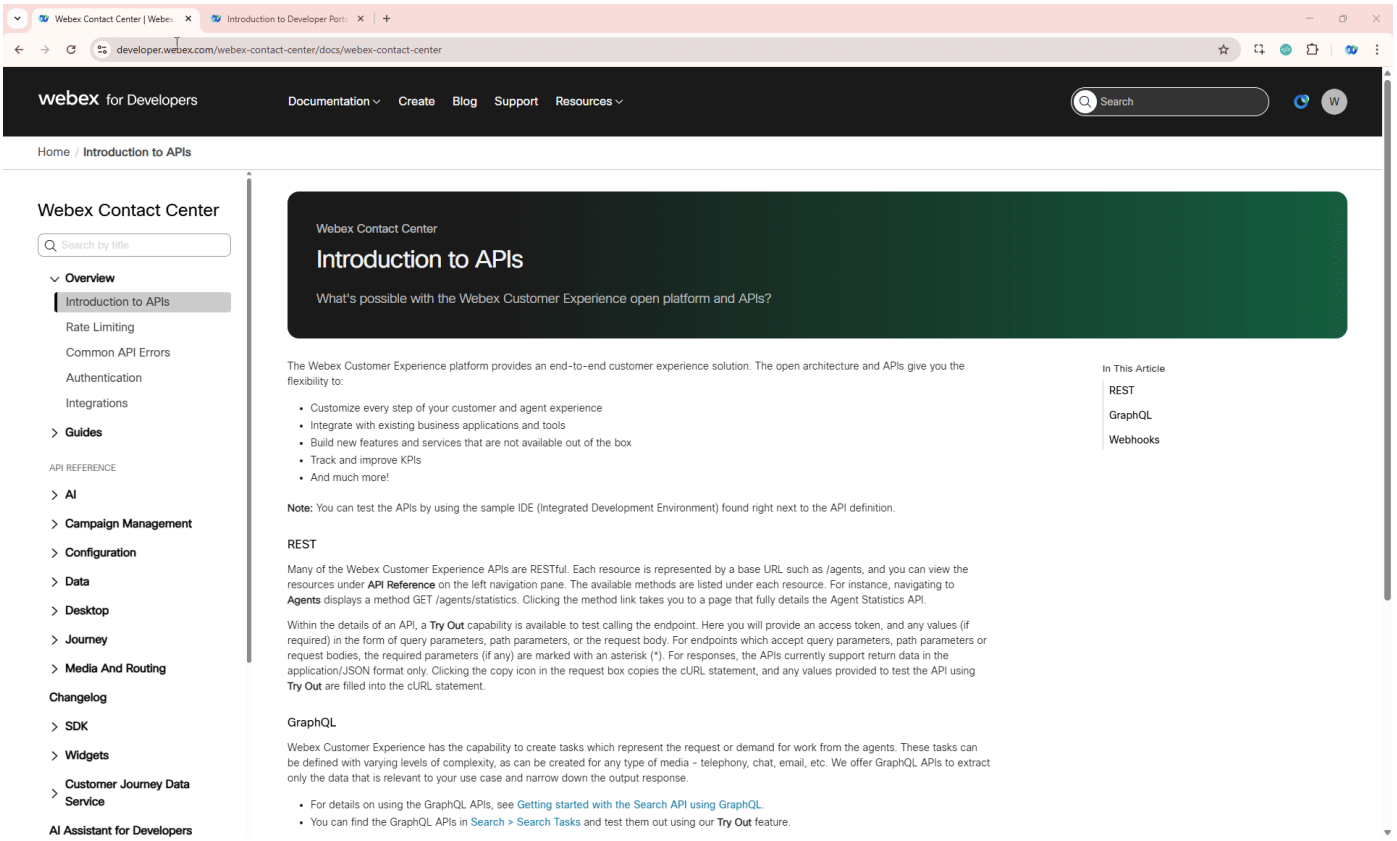


USE SEARCH API TO RETRIEVE DATA FROM ANALYZER DB.

Note

When working with Webex Contact Center (Webex CC) GraphQL queries, timestamps are represented in **Epoch time (Unix timestamp)** format. This format counts the number of seconds (or milliseconds) that have elapsed since **January 1, 1970 (UTC)**.
If you need to convert a regular date/time into Epoch format or vice versa, you can use this online converter: https://www.epochconverter.com/
Ensure that your queries and filters use the correct time format to retrieve accurate results.

1. Switch to **Developer Portal** then locate and select **Data** then select **Search** from **API REFERENCE** menu
2. Click on **Search tasks** and then switch to **Try Out** tab
3. Make sure **Authorization** toggle to *Use personal access token* is **ON**



4. Click on **Maximize Screen**, clear the text from **GraphQL** query window. Then paste the following query.

Request Body:

```
{
  #Global CAD Variables: Usage of taskDetails Object to retrieve the Value of Global Variables
  taskDetails(
    # NOTE: from and to are mandatory arguments that take the Epoch timestamp in milliseconds
    from: 1767864600000 #This can be set to Date.now() - (days * 24 * 60 * 60 * 1000) for lookback in days
    to: 1767868200000 #This can be set to Date.now() in millis
    filter: {
      #Filter the type of Task
      and: [
        { channelType: { equals: telephony } } #Telephony calls only
        { origin: { equals: "+31204854646" } } #Customer ANI
        { status: { equals: "ended" } } #Final Disposition
        { direction: { equals: "inbound" } } #Inbound call only
        { isActive: { equals: false } } #Resolved call only
        { owner: { notequals: { id: null } } } #Only calls that had an Owner
      ]
    }
  ) {
    tasks {
      id #TaskId-SessionId-CallId
      status #Status
      totalDuration #CallTime
      origin #ANI
      destination #DNIS
      lastAgent {
        #Agent
        id
        name
      }
      stringGlobalVariables(name: "Global_Language") {
        #GlobalCADVariable
        name
        value
      }
    }
  }
}
```

Note

Current query is configured to search calls with following details from Analyzer database:

a. Time range set to 1 hour: From **Thursday, January 8, 2026 10:30:00 AM** to **11:30:00 AM**.
b. Telephony inbound calls only.
c. Calls only from **+31204854646**.
d. Ended calls only.
e. Calls that were assigned to an owner (agent).

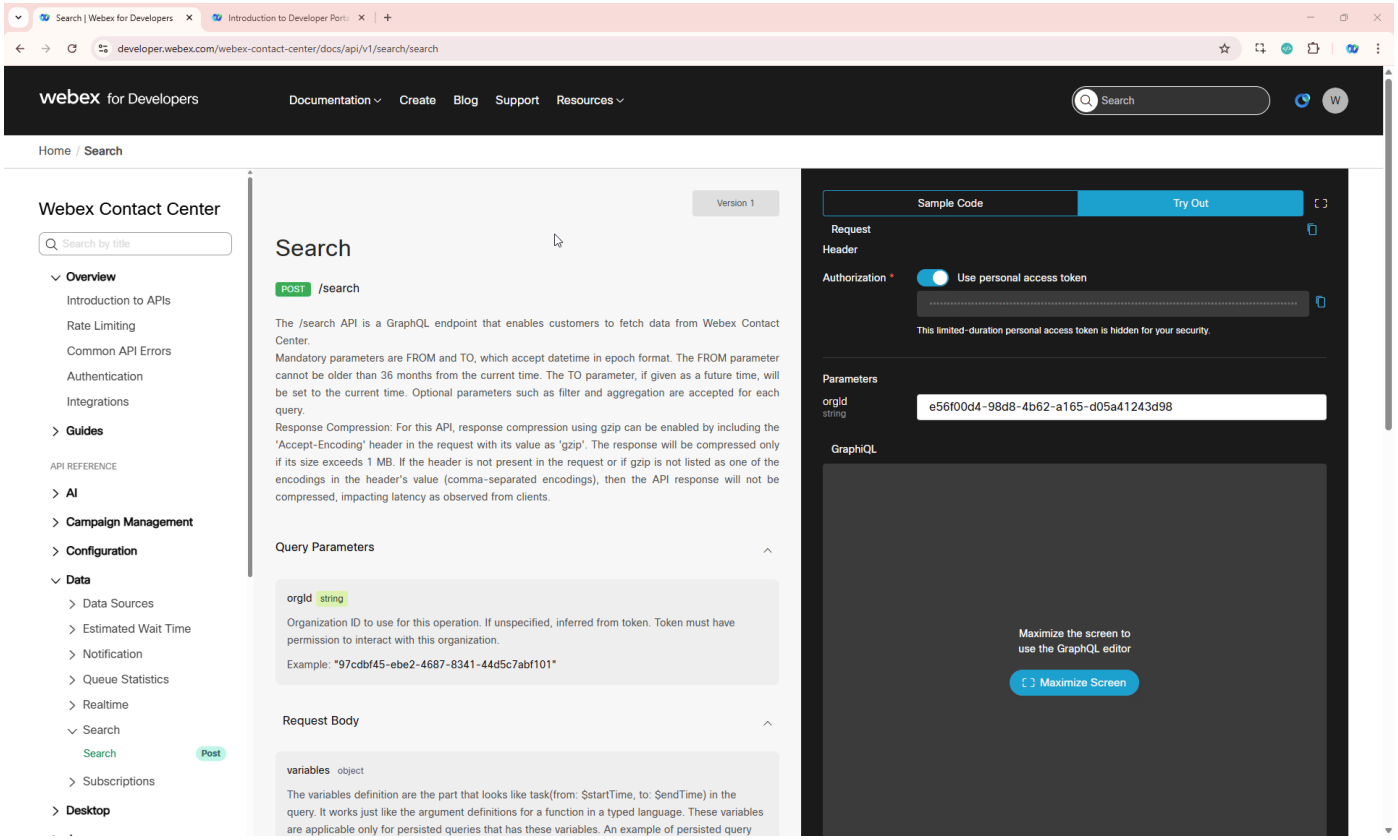
Example of the expected response: **200 Response**


```
{
    "data": {
        "taskDetails": {
            "tasks": [
                {
                    "id": "34e12bbc-4888-474e-b6c8-b0afbe8d442f",
                    "status": "ended",
                    "totalDuration": 209402,
                    "origin": "+31204854646",
                    "destination": "+441256231731",
                    "lastAgent": {
                        "id": "b9b45479-756f-4c55-8663-8ae7800a9a18",
                        "name": "Agent140 Lab"
                    },
                    "stringGlobalVariables": {
                        "name": "Global_Language",
                        "value": "en-AU"
                    }
                }
            ]
        }
    }
}
```

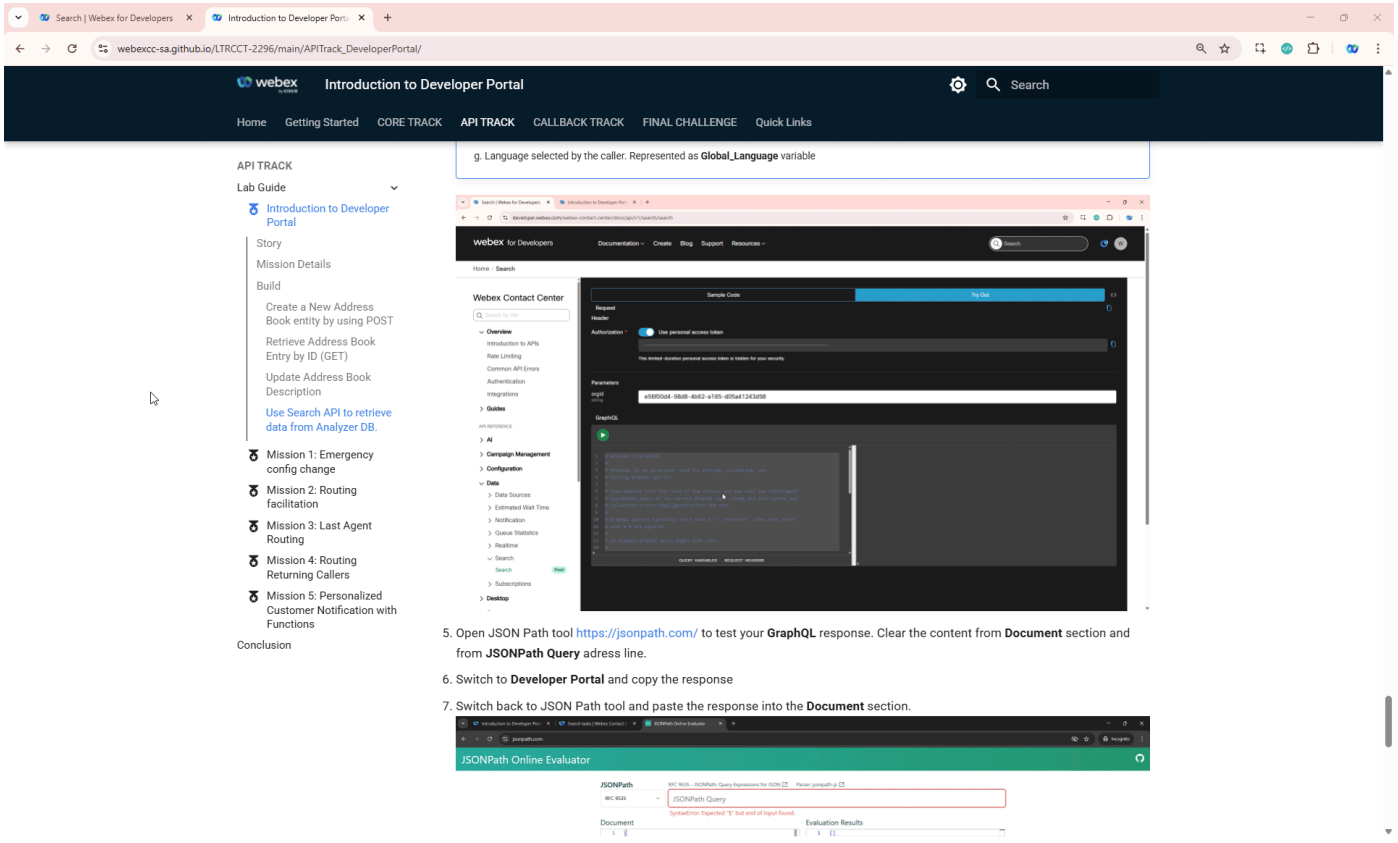

Note

Output of the query is configured to represent the following information

a. Unique ID of the call
b. Status of the call
c. Total duration of the call
d. Origin of the call. Who called.
e. Destination of the call. Entry Point number.
f. Agent, who accepted the call: ID and Name
g. Language selected by the caller. Represented as **Global_Language** variable



5. Open JSON Path tool https://jsonpath.com/ to test your **GraphQL** response. Clear the content from **Document** section and from **JSONPath Query** adress line.
6. Switch to **Developer Portal** and copy the response
7. Switch back to JSON Path tool and paste the response into the **Document** section.



8. Test the following paths by pasting them into **JSONPath Query** adress line one by one:

`$.data.taskDetails.tasks[0].id` - Interaction ID of the call.

`$.data.taskDetails.tasks[0].status` - Status of the call.

`$.data.taskDetails.tasks[0].totalDuration` - Total Duration of the call.

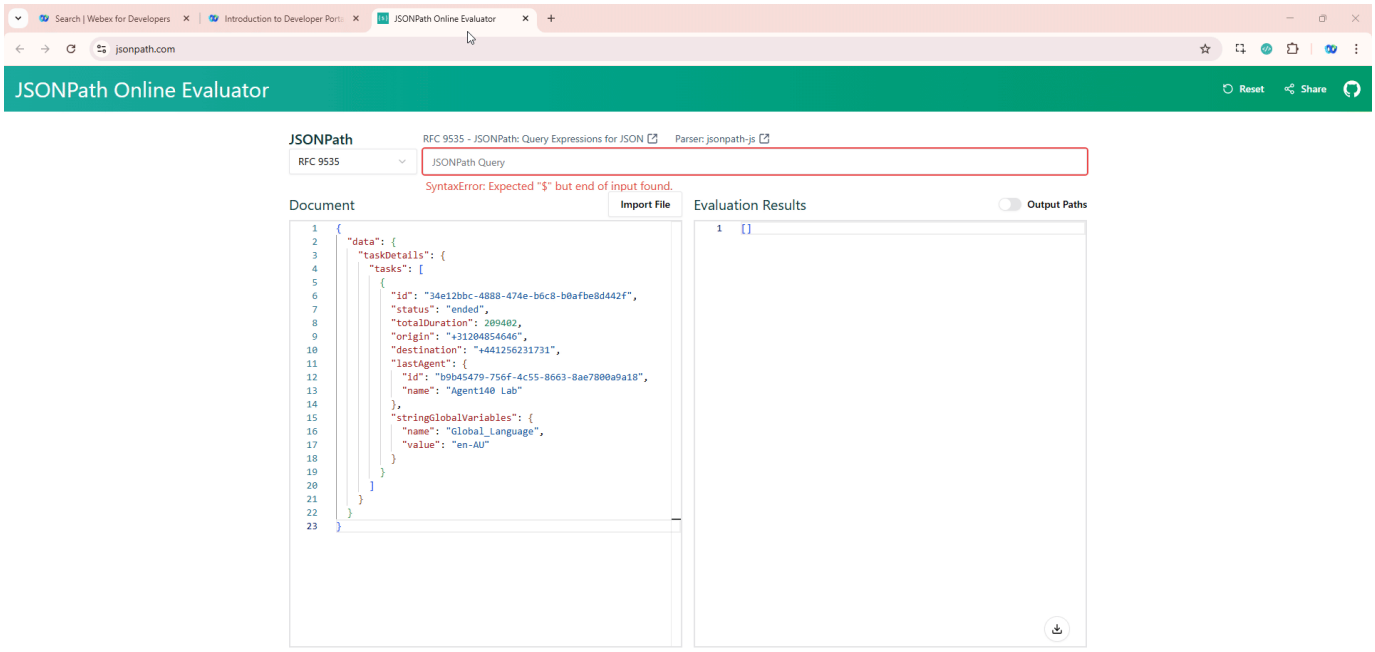
`$.data.taskDetails.tasks[0].destination` - Call destination. This is the number essigned to Entry Point.

`$.data.taskDetails.tasks[0].lastAgent.id` - Agent ID who accepted the call.

`$.data.taskDetails.tasks[0].lastAgent.name` - Agent name who accepted the call.

`$.data.taskDetails.tasks[0].stringGlobalVariables.name` - Language Global Variable that was used in the flow.

`$.data.taskDetails.tasks[0].stringGlobalVariables.value` - Language selected by a caller.



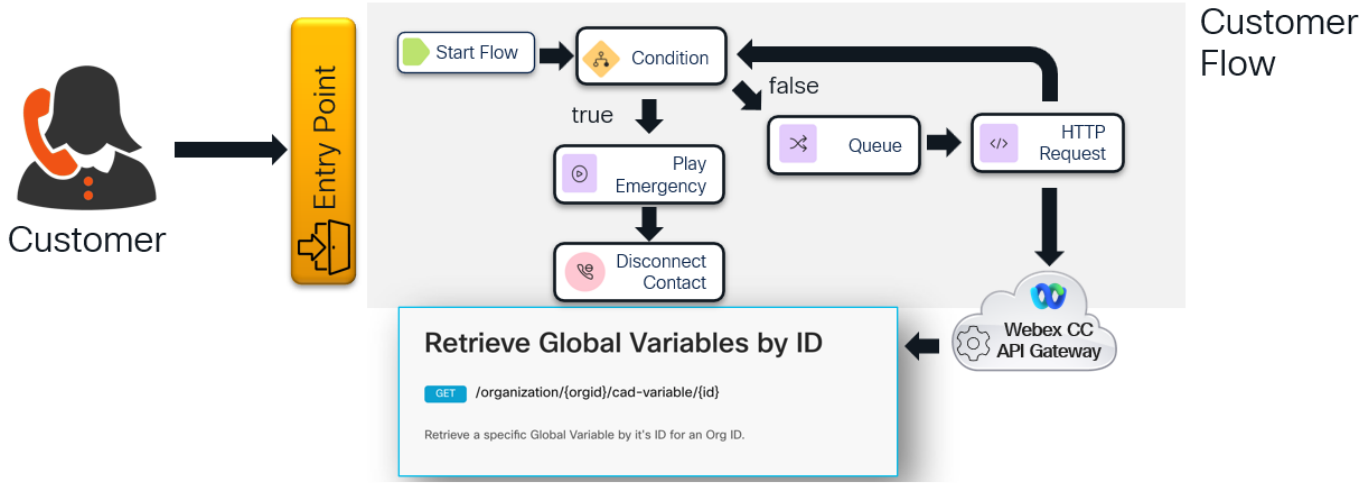
ashphy - Kazuki Hamasaki

Congratulations, you have succesfully completed Introduction to Developer Portal mission! 🎉🎉

3.1.2 Mission 1: HTTP API POST to Control Hub (Emergency config change)

Story

Consider a scenario where a supervisor needs ability to change routing decision during an emergency without accessing admin portal. It can be done by changing the **Default Value** of a Global Variable via API PUT call from False to True and use Condition in main IVR script to do routing decision. In this mission we are going to create a control script for Supervisors that changes default value of Global Variable from **True** to **False**



Prerequisites:

- In this mission, we will modify the behavior of the main flow by adjusting the value of a Global Variable through a separate management flow.
- The management flow will be created within the scope of this mission below.
- [Important!] The creation of the main flow **Main_Flow_Your_Attendee_ID** is outlined in the first part of the **Mission 1** of **CORE TRACK** (Steps #1-12). Please complete that part to continue with this mission.

Call Flow Overview

1. Supervisor calls to management flow and provide its PIN code.
2. If the PIN correct, a PUT API request will be triggered to change a Global Variable default setting from **False** to **True**.
3. A caller makes a call to contact center where **Main_Flow_Your_Attendee_ID** checks the global variable and transfer the call further based on settings.

Mission Details

Your mission is to:

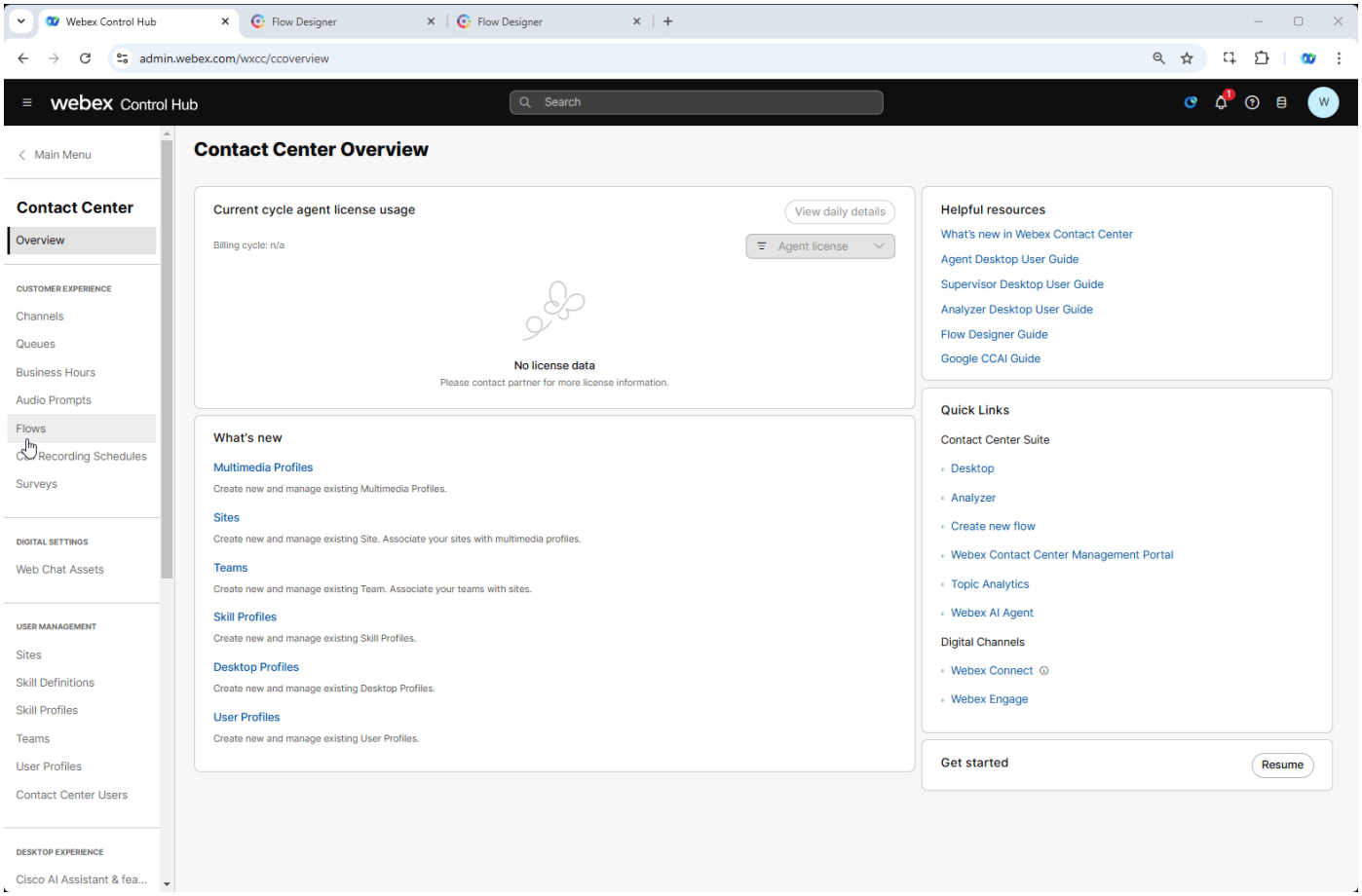
1. Create a management flow which will trigger a global variable default value change.
2. Modify your **Main_Flow_Your_Attendee_ID** to check the global variable default value.

Build

1. In Control Hub **Flows** page open **Global Variables** tab at the top and create new Global Variable:

- Name: **EmergencyGV_Your_Attendee_ID**
- Type: **Boolean**
- Default Value: **False**

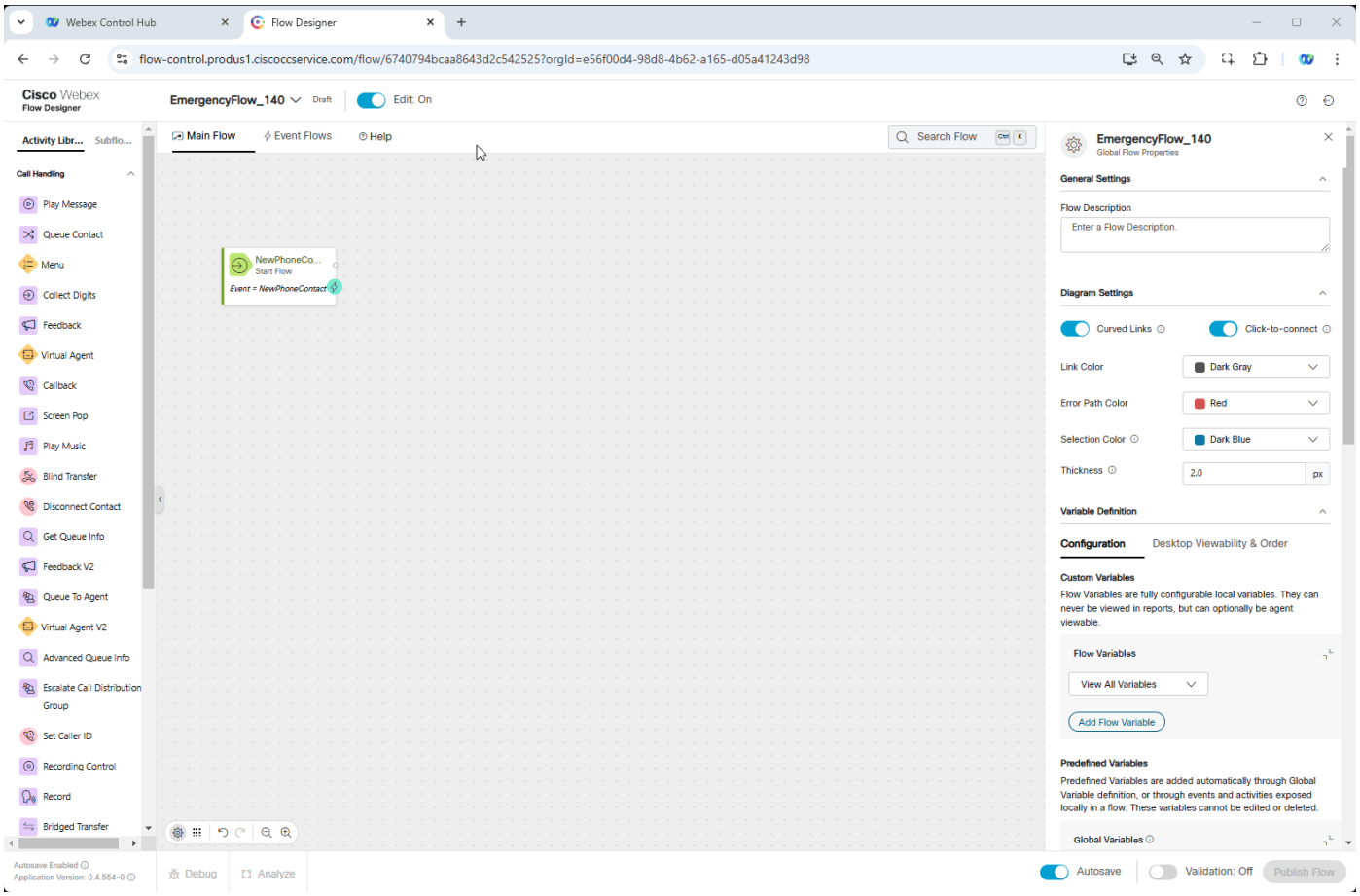
[Important] Copy your new created **Global Variable ID** and save to a notepad. We are going to use them in API request in further steps.



2. Create a new flow by navigating to **Flows**, click on **Manage Flows** dropdown list and select **Create Flows**.
3. Select **Start Fresh** and give it a name **EmergencyGV_Your_Attendee_ID**. Then click **Create Flow**.

   The flow canvas with the initial **NewPhoneContact** node will load when a flow is created.
4. Add a **Collect Digits** node:

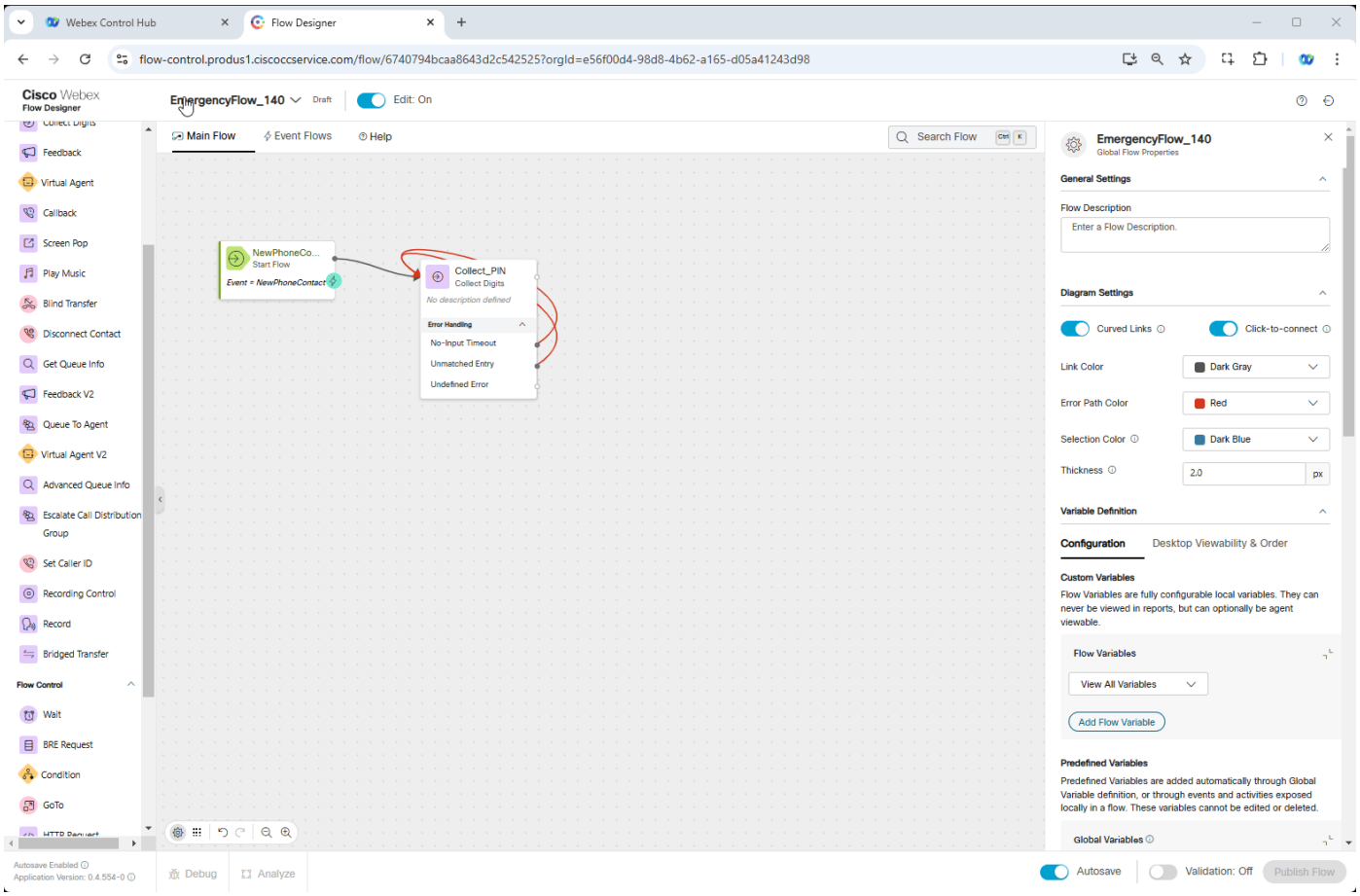
- Click on it, navigate to the node settings on the right and rename the **Activity Label** of the node to **CollectPIN**
- Connect the **New Phone Contact** output node edge to this **Collect Digits** node
- Loop **No-Input Timeout** and **Unmatched Entry** to the input of the **Collect Digits** node
- Turn on **Enable Text-To-Speech** toggle
- Select the Connector: **Cisco Cloud Text-to-Speech**
- Click the **Add Text-to-Speech Message** button
- Text-to-Speech Message: ***Please enter 4 digits pin code to activate emergency flow.***
- Delete the **Audio File** element above **Text-to-Speech Message** field
- Set checkbox in **Make Prompt Interruptible**



5. Add **Condition** node:

- Activity Label: **PIN_Check**
- Connect the output node edge from the **Collect Digits** node to this node
- In the Expression section write an expression ***{{CollectPIN.DigitsEntered == '1111'}}***

  [Optional] You can verify the expression result by clicking on **Test Expression** icon in the Expression section



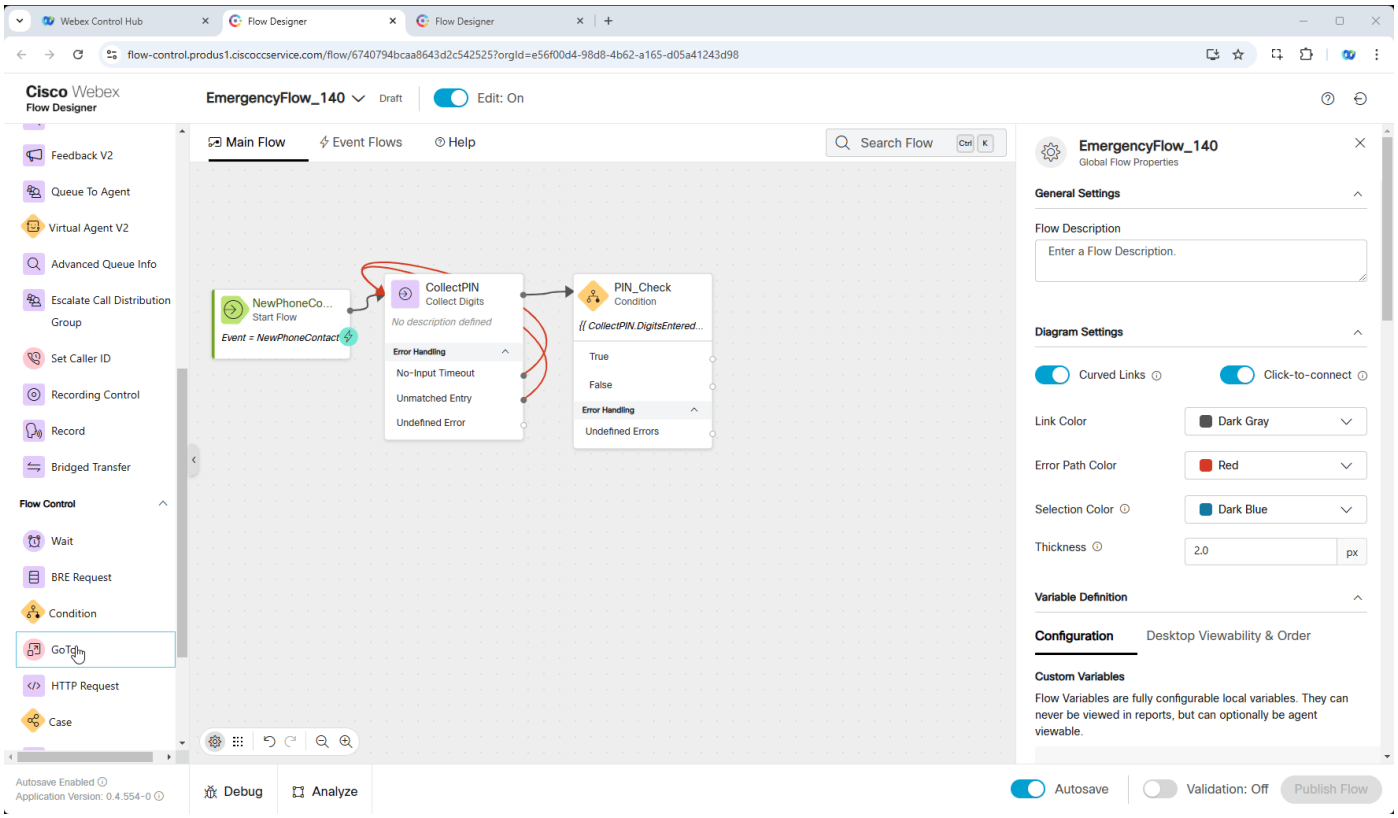
6. Add **HTTP Request** node. We are going to use **Update Global Variable API PUT** request in the node configuration.

- Activity Label: **HTTP_PUT**
- Connect the **TRUE** output edge from the **PIN_Check** node to this node
- Connector: **WxCC_API**
- Request Path: **/organization/e56f00d4-98d8-4b62-a165-d05a41243d98/cad-variable/*{ID}*** - change ***{ID}*** with Global Variable ID you created in **Step 1** of this mission.
- Method: **PUT**
- Content Type: **Application/JSON**
- Request Body ([Important!] Replace the value of **id** and **name** parameters below by the proper ones created in step 1 above):

```
{
    "active": true,
    "agentEditable": false,
    "agentViewable": false,
    "variableType": "Boolean",
    "defaultValue": "true",
    "desktopLabel": "",
    "id": "Your Global Variable ID created in step 1",
    "name": "Your Global Variable Name created in step 1",
    "organizationId": "e56f00d4-98d8-4b62-a165-d05a41243d98",
    "reportable": false,
    "version": 1
}
```

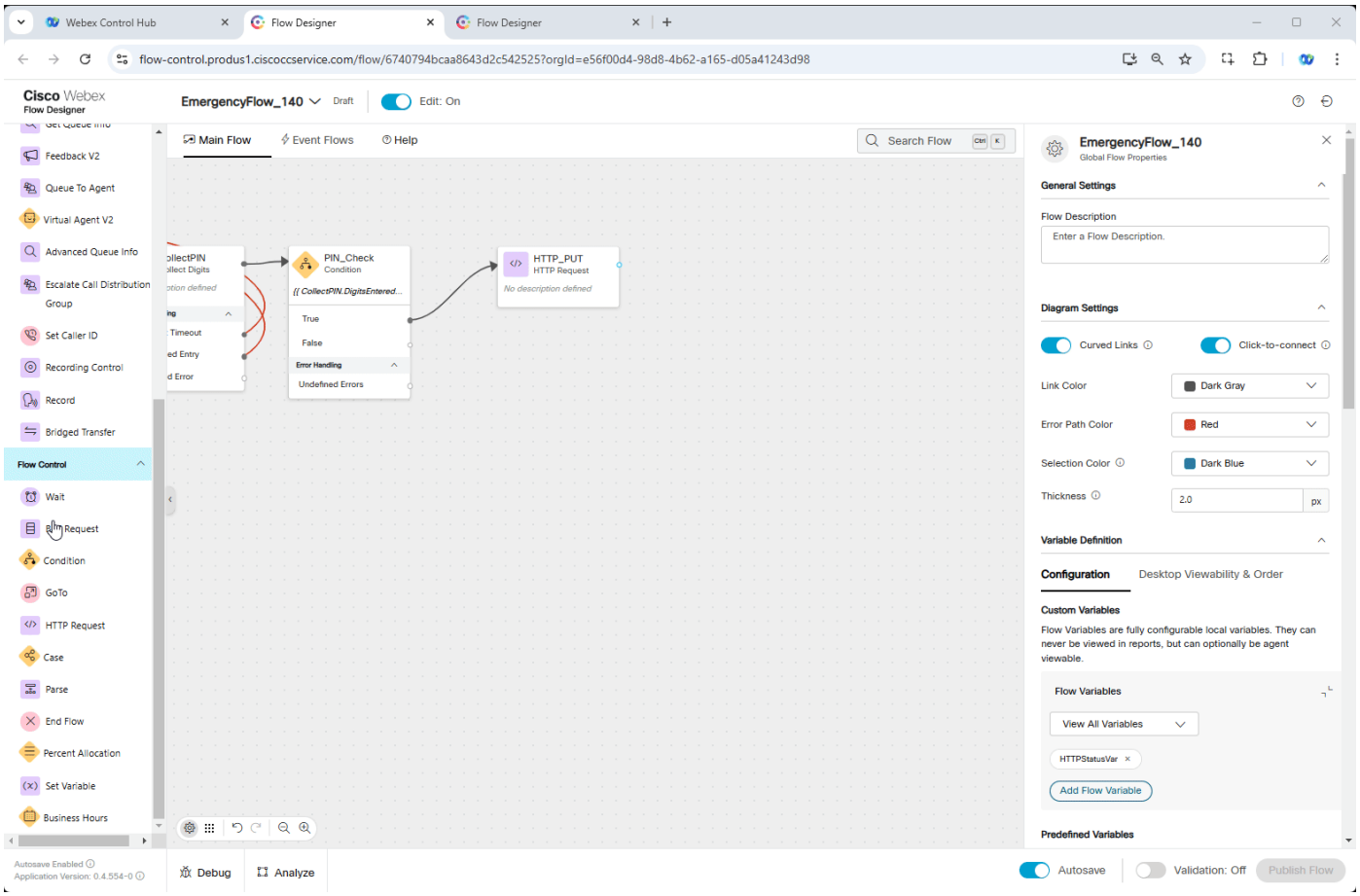
Note

In Request body we are going to change Default Value of Global Variable **EmergencyGV_Your_Attendee_ID** from ***false*** to ***true***



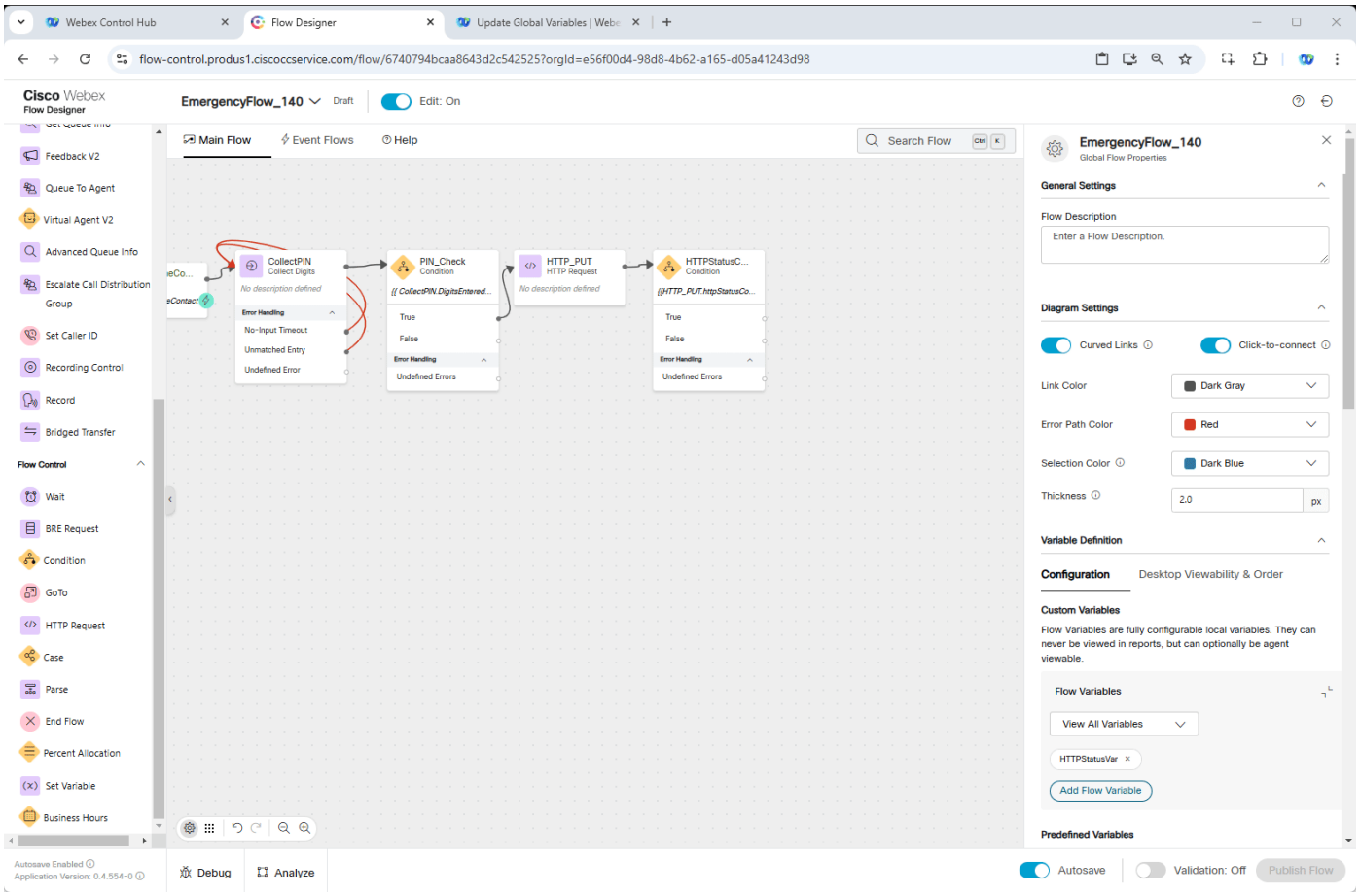
7. Add one more **Condition Node**. In this node we are going to check the status of our API PUT request. If it is **200 OK** the output will be **True** and if other than **200** then **False**.

- Activity Label: **HTTPStatusCode**
- Connect the output node edge from the **HTTP_PUT** node to this node
- In the Expression section write an expression ***{{HTTP_PUT.httpStatusCode == 200}}***



8. Add a **Play Message** node

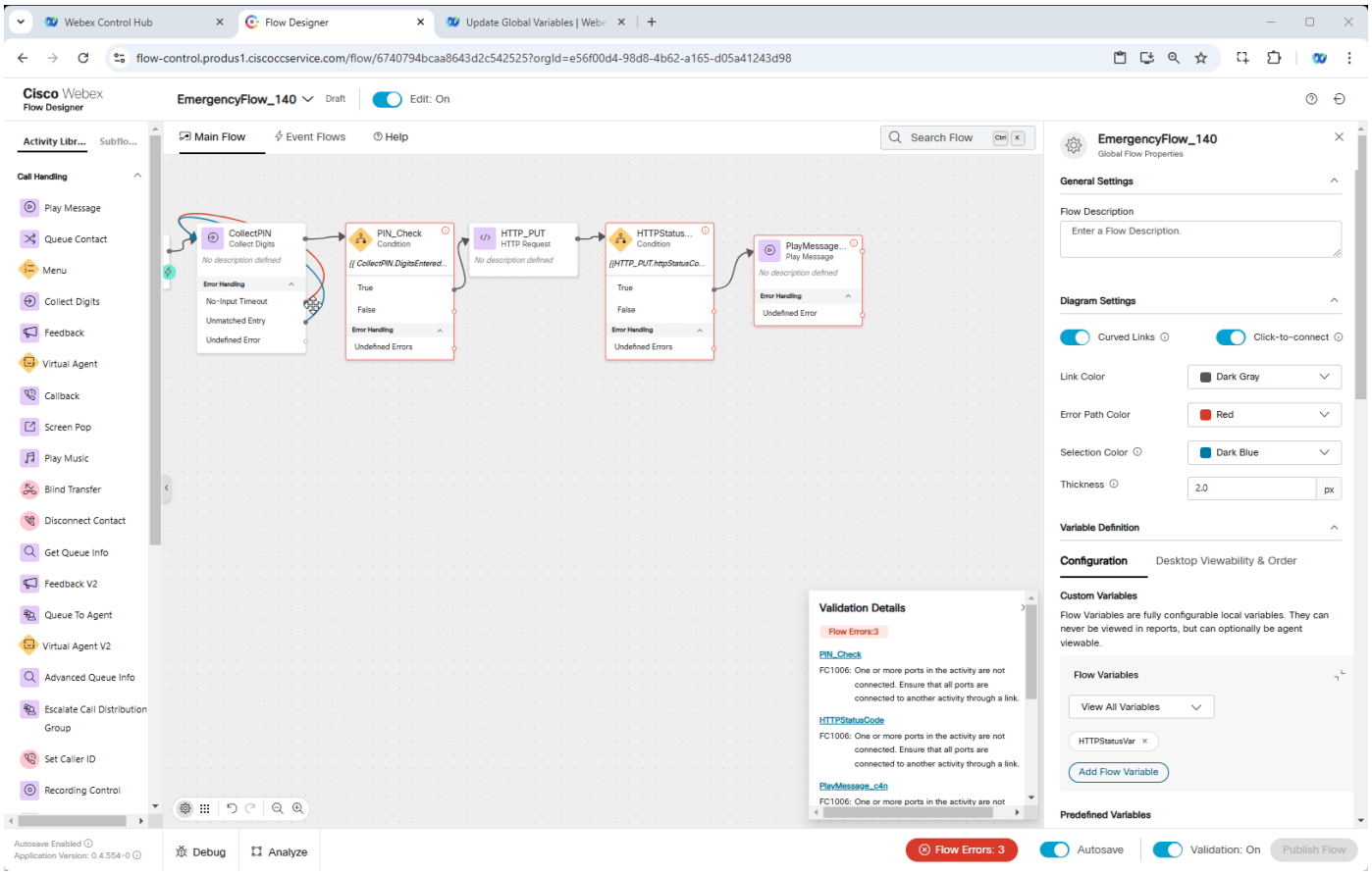
- Connect the **HTTPStatusCode** TRUE output node edge to this **Play Message** node
- Turn on **Enable Text-To-Speech** toggle
- Select the Connector: **Cisco Cloud Text-to-Speech**
- Click the **Add Text-to-Speech Message** button
- Delete the selection for Audio File
- Text-to-Speech Message: ***You have successfully modified your emergency configuration.***



9. Add another **Play Message** node

- Connect the **HTTPStatusCode** FALSE output node edge to this **Play Message** node
- Connect the **PIN_Check** FALSE output node edge you created in **Step 5** to this **Play Message** node
- Enable Text-To-Speech
- Select the Connector: **Cisco Cloud Text-to-Speech**
- Click the **Add Text-to-Speech Message** button
- Delete the selection for Audio File
- Text-to-Speech Message: ***Something went wrong. Please check your configuration and try again.***

10. Add **Disconnect Contact** node at the end of the flow and connect both **Play Message** nodes created in **Steps 8** and **9** to that node.



11. Validate and publish the flow:
    - Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
    - If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
    - In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.
12. Map your flow to your inbound channel
    - Navigate to Control Hub > Contact Center > Channels
    - Locate your Inbound Channel (you can use the search): **Your_Attendee_ID_Channel**
    - Select the Routing Flow: **EmergencyGV_Your_Attendee_ID**
    - Select the Version Label: **Latest**
    - Click **Save** in the lower right corner of the screen

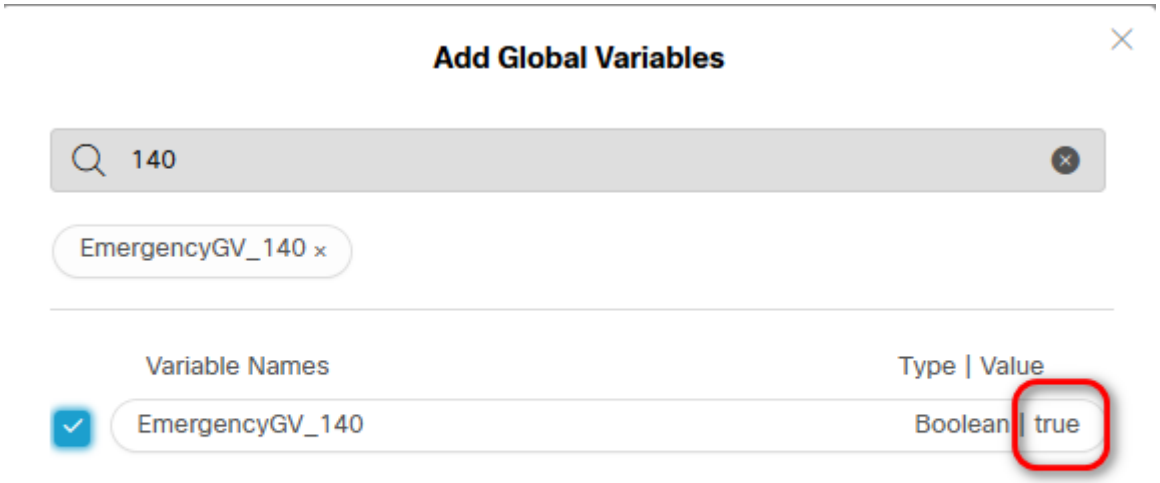
Testing

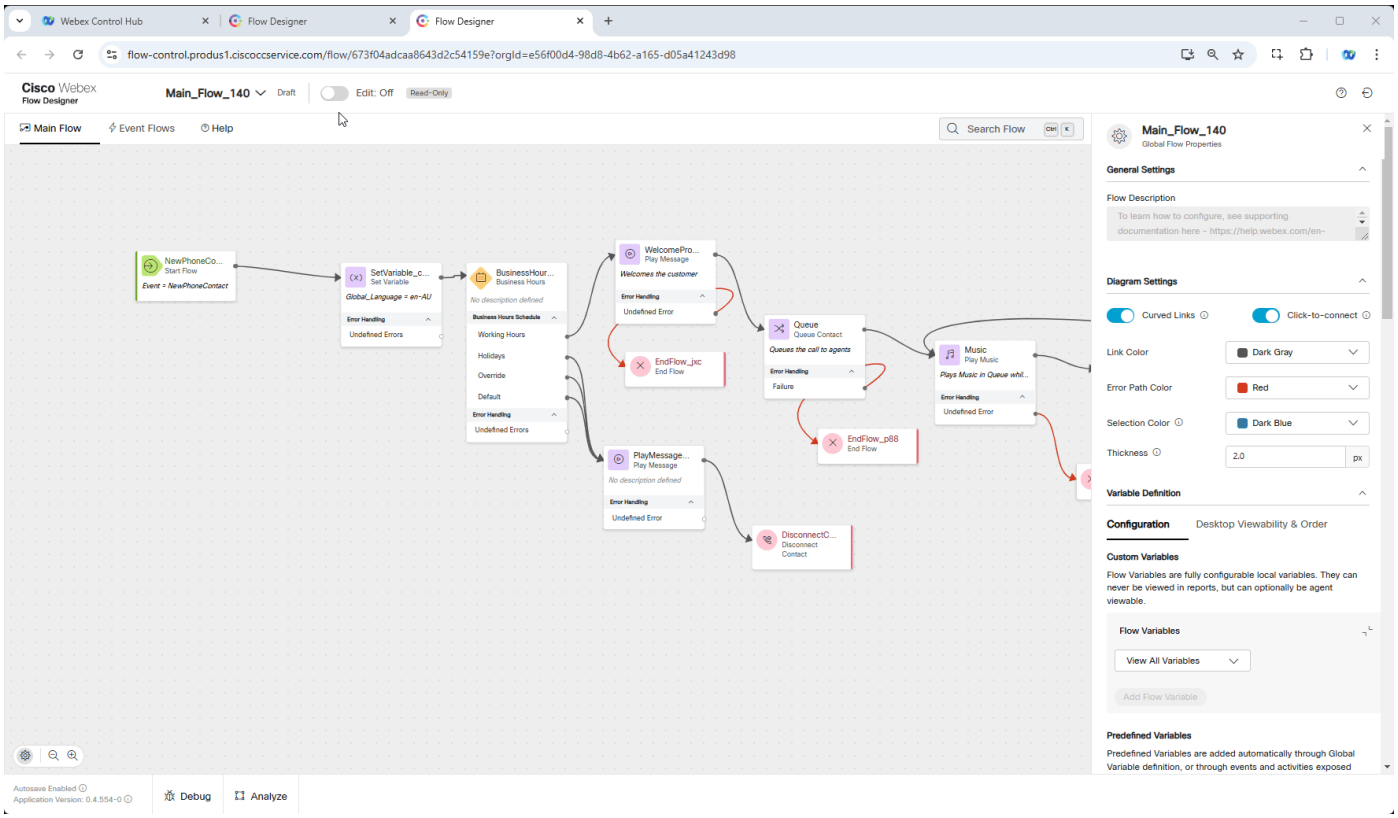
1. Open your Global Variable **EmergencyGV_Your_Attendee_ID** and make sure Default Value is set to **False**
2. Make a call to your Support Number, when asked provide a pin code 1111# and listen the next message:

   a. If "***You have successfully modified your emergency configuration.***" you're good to proceed with step 3.

   b. If "***Something went wrong. Please check your configuration and try again.***" then before proceeding you need to fix your flow. Call the instructor for assistance.
3. Open your Global Variable **EmergencyGV_Your_Attendee_ID** again, refresh the page if it was opened and make sure **Default Value** is now set to True.

4. Now, let's get to the fun part. Open the **Main_Flow_Your_Attendee_ID** we created in **Mission 1** of **Core track**, make sure **Edit** toggle is **ON**.
5. Add Global Variable **EmergencyGV_Your_Attendee_ID** and make sure Default Value is set to **True** in General Settings of the flow as shown on the following picture.





6. Add **Condition** node:

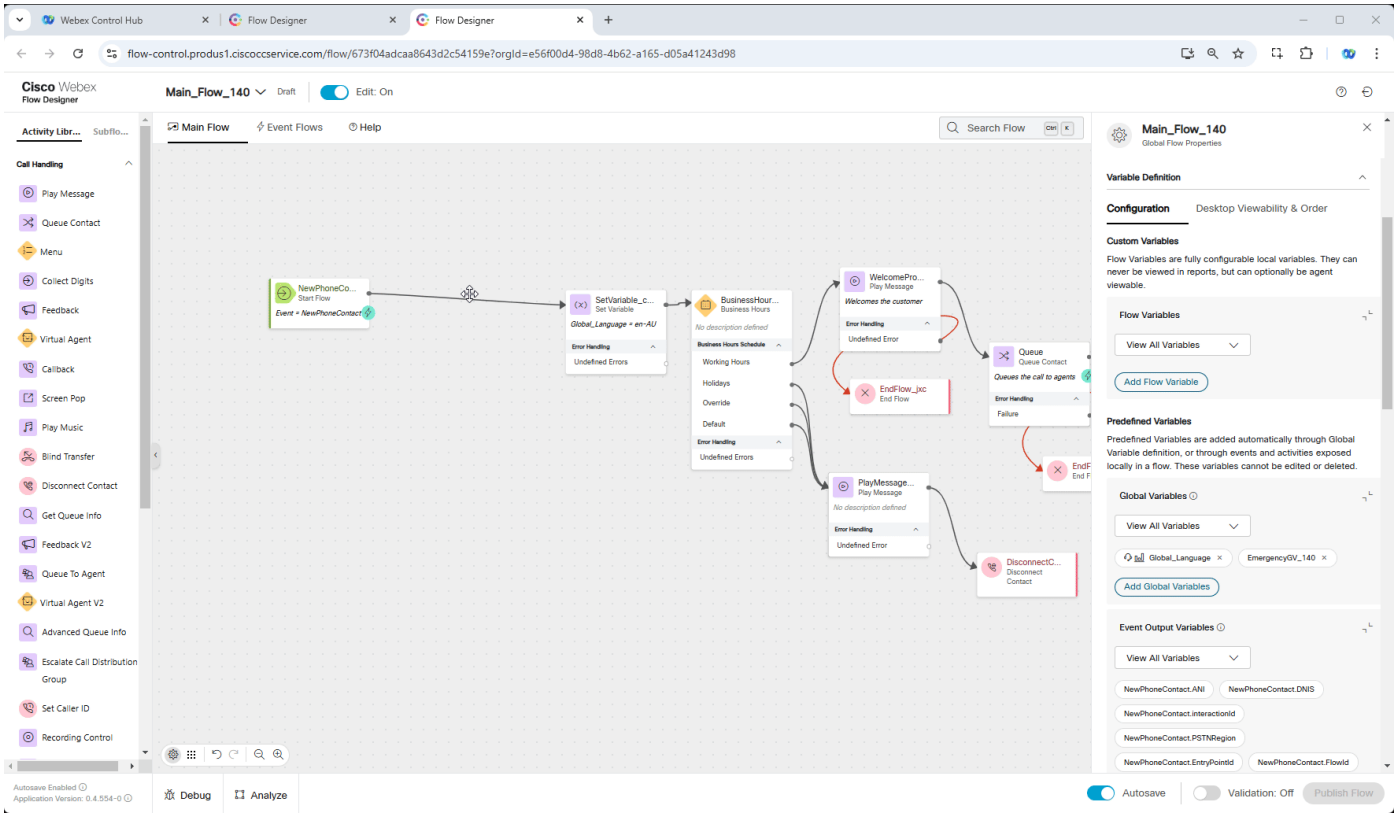
- Delete the connection between **NewPhoneContact** node and **Set Variable** node at the beginning of the flow
- Connect the output node edge of the **NewPhoneContact** node to **Condition** node
- Connect the output False node edge from the **Condition** node to **Set Variable** node
- In the Expression section write an expression ***{{EmergencyGV_Your_Attendee_ID == true}}***

tional

You can Verify the expresion result by Clicking on **Test Expression** icon in the Expresion section.

Note

Depending on which Track you have followed after the Core Track, you may have **NewPhoneContact** connected either to **FeedbackSet** node or to **SetVariable** node. Remove this connection and add a **Condition** node in between.

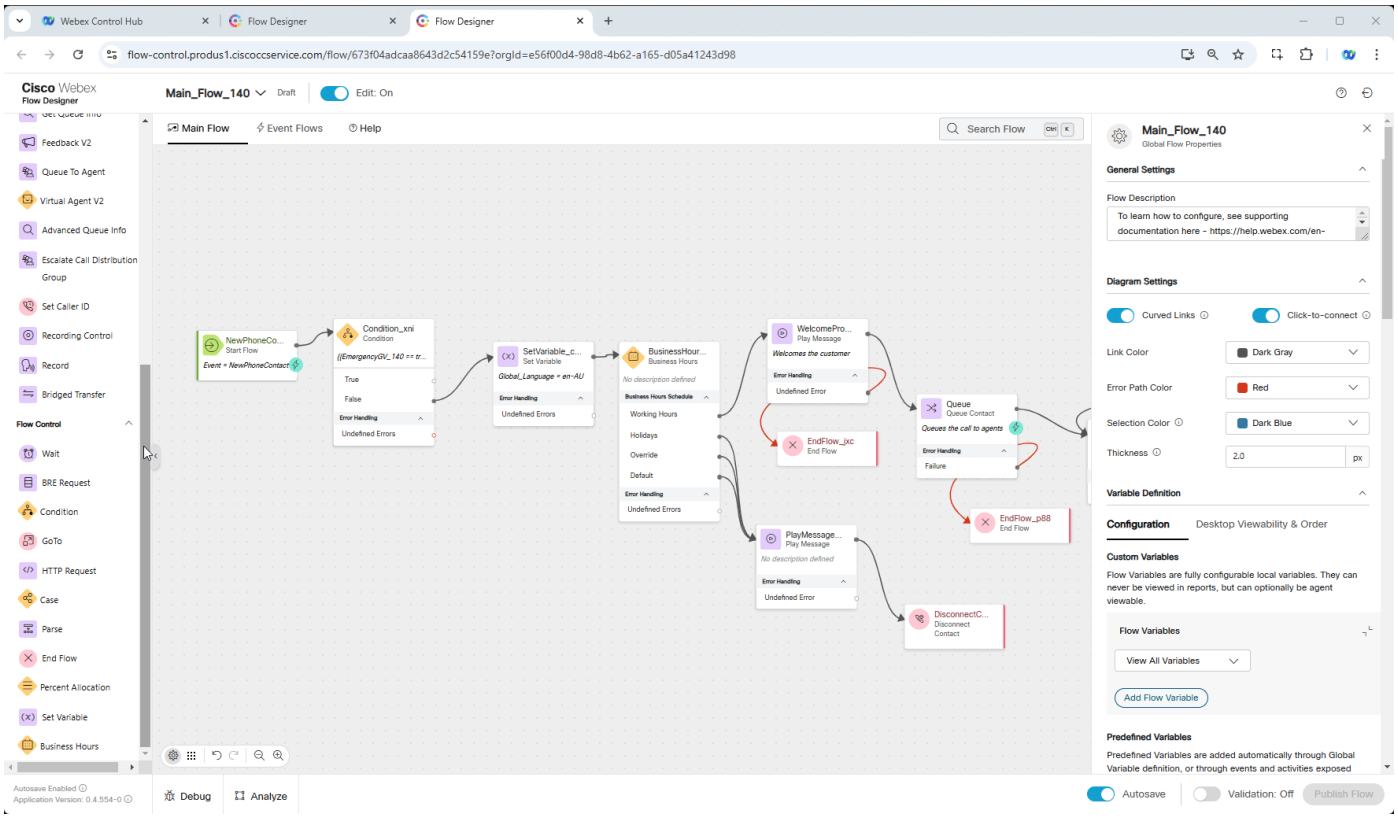


7. Add a **Play Message** node and **Disconnect Contact** node.

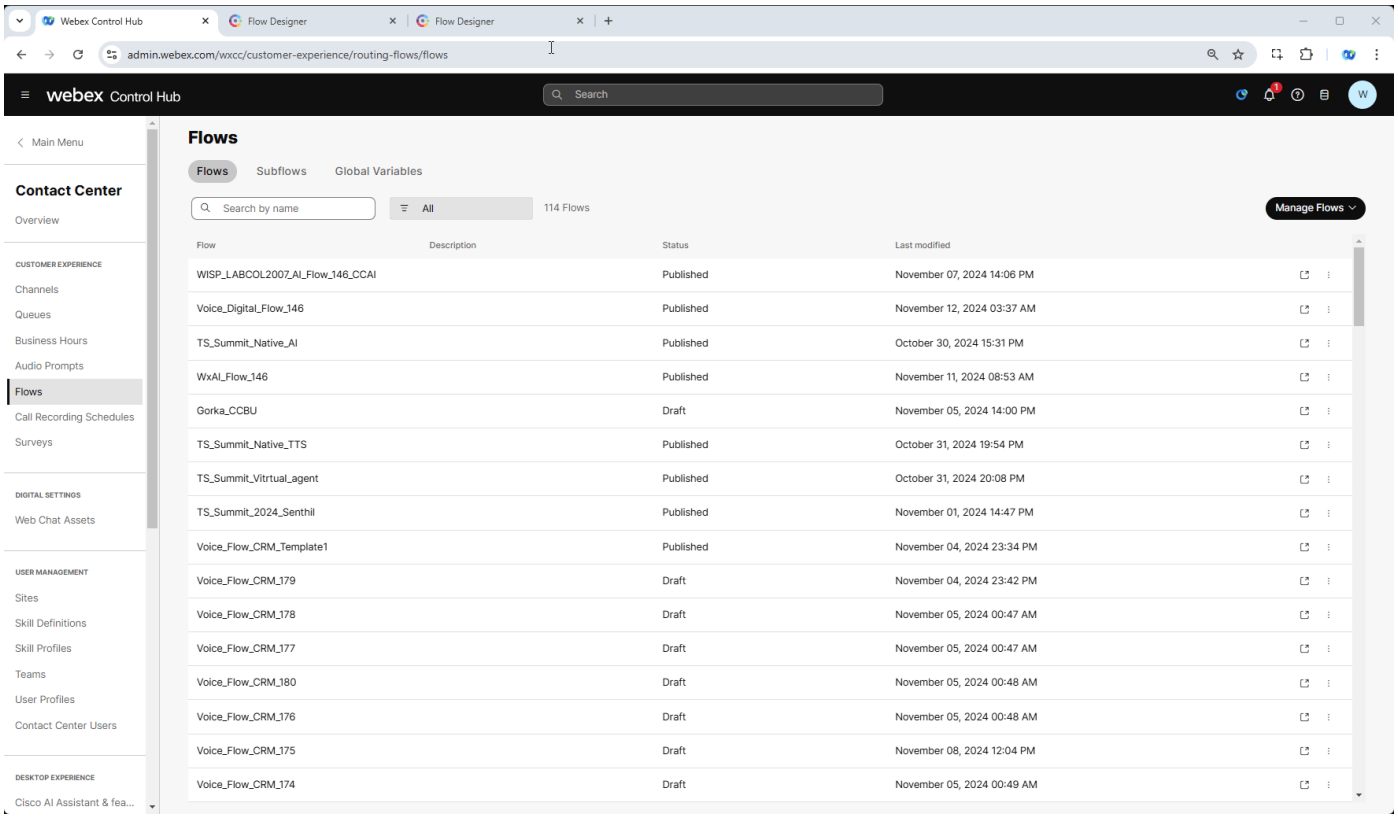
- Connect the **TRUE** output node edge of the **Condition** node to **Play Message** node
- Connect the output node edge of **Play Message** node to **Disconnect Contact** node.

Click on **Play Message** node and configure it in the following way:

- Turn on **Enable Text-To-Speech** toggle
- Select the Connector: **Cisco Cloud Text-to-Speech**
- Click the **Add Text-to-Speech Message** button
- Delete the selection for Audio File
- Text-to-Speech Message: ***Sorry, Emergency flow has been enabled. All operators have been evacuated. Please call later.***



8. Validate and publish the flow:
   - Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
   - If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
   - In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.
9. Because we are using only one number to make calls we need to map your **Your_Attendee_ID_Channel** back to the **Main_Flow_Your_Attendee_ID**
   - Navigate to Control Hub > Contact Center > Channels
   - Locate your Inbound Channel (you can use the search): **Your_Attendee_ID_Channel**
   - Select the Routing Flow: **Main_Flow_Your_Attendee_ID**
   - Select the Version Label: **Latest**
   - Click **Save** in the lower right corner of the screen
10. Make a call and you should hear the message we configured on **Step 7**.
11. Revert the Global Variable value from **True** to **False** in Control Hub and click **Save**.
    - Name: **EmergencyGV_Your_Attendee_ID**
    - Type: **Boolean**
    - Default Value: **False**



12. Make a test call again and you should hear the Welcome Prompt.

Congratulations, you have succesfully completed Emergency Config mission! 🎉🎉

3.1.3 Mission 2: Routing facilitation

Story

The primary objective of this new feature is to enhance nodes activities to include a dynamic variable-based selection option to make your flow smaller and simpler to adjust. You will learn how to use **Dynamic Variables** in multiple nodes including **GoTo**, **Business Hours**, **Queue** and other nodes.

Call Flow Overview

1. When call arrives fetch the data from **MockAPI** based on your Dialed Number
2. Write the data into respective preconfigured flow variables. These variables are being used in all consequent nodes.
3. Business Hours entity configured to cover EMEA timezone. Call should go through **Working Hours** exit edge in normal behavior.
4. Play Message nodes have been configured to play messages received from API call

Mission Details

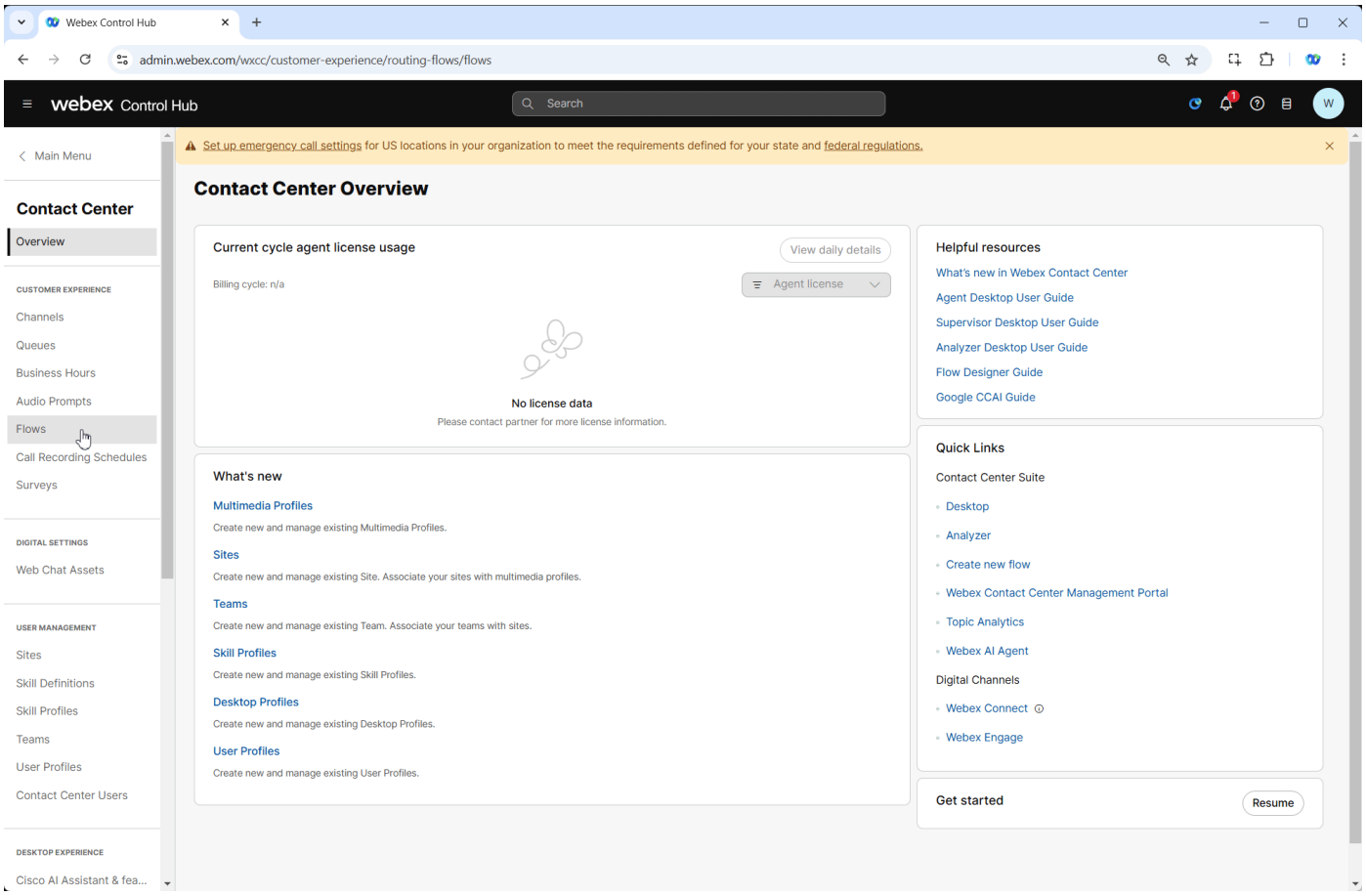
Your mission is to:

1. Create a new flow by using pre-defined flow template.
2. Request the data from external database and parse it into flow variables which are coming with a flow template.
3. You do not need to create Business Hours, Channels and additional Flows as they have been preconfigured for you.

> **od to Know [Optional]**
>
> We are going to imitate a real API server by providing realistic responses to requests. For that we chose Server **MockAPI**.
>
> For more information of how you can use MockAPI please watch these Vidcasts:
>
> - **[ADVANCED] Use MockAPI to enhance your Demos - PART 1**
> - **[ADVANCED] Use MockAPI to enhance your Demos - PART 2**

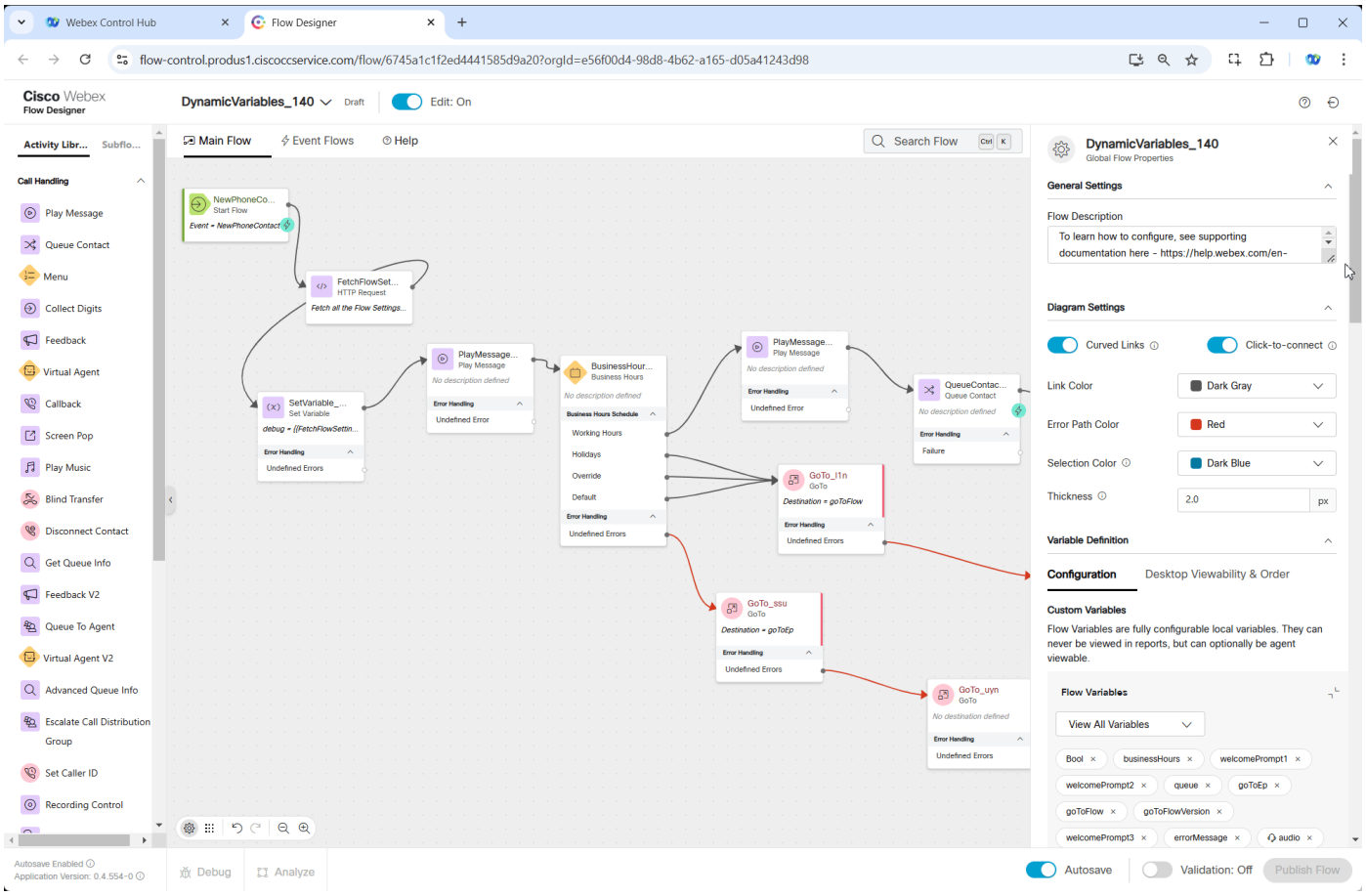
Steps

1. Switch to Control Hub, then navigate to **Flows**, click on **Manage Flows** dropdown list and select **Create Flows**.
2. New Tab will be opened. Navigate to **Flow Templates**.
3. Choose **Dynamic Variable Support** and click **Next**. You can open **View Details** and to see observe flow structure and read flow description.
4. Name you flow as **DynamicVariables_Your_Attendee_ID** . Then click on Create Flow.



5. Observe preconfigured nodes and flow variables. If you have questions please reach out to lab proctor.

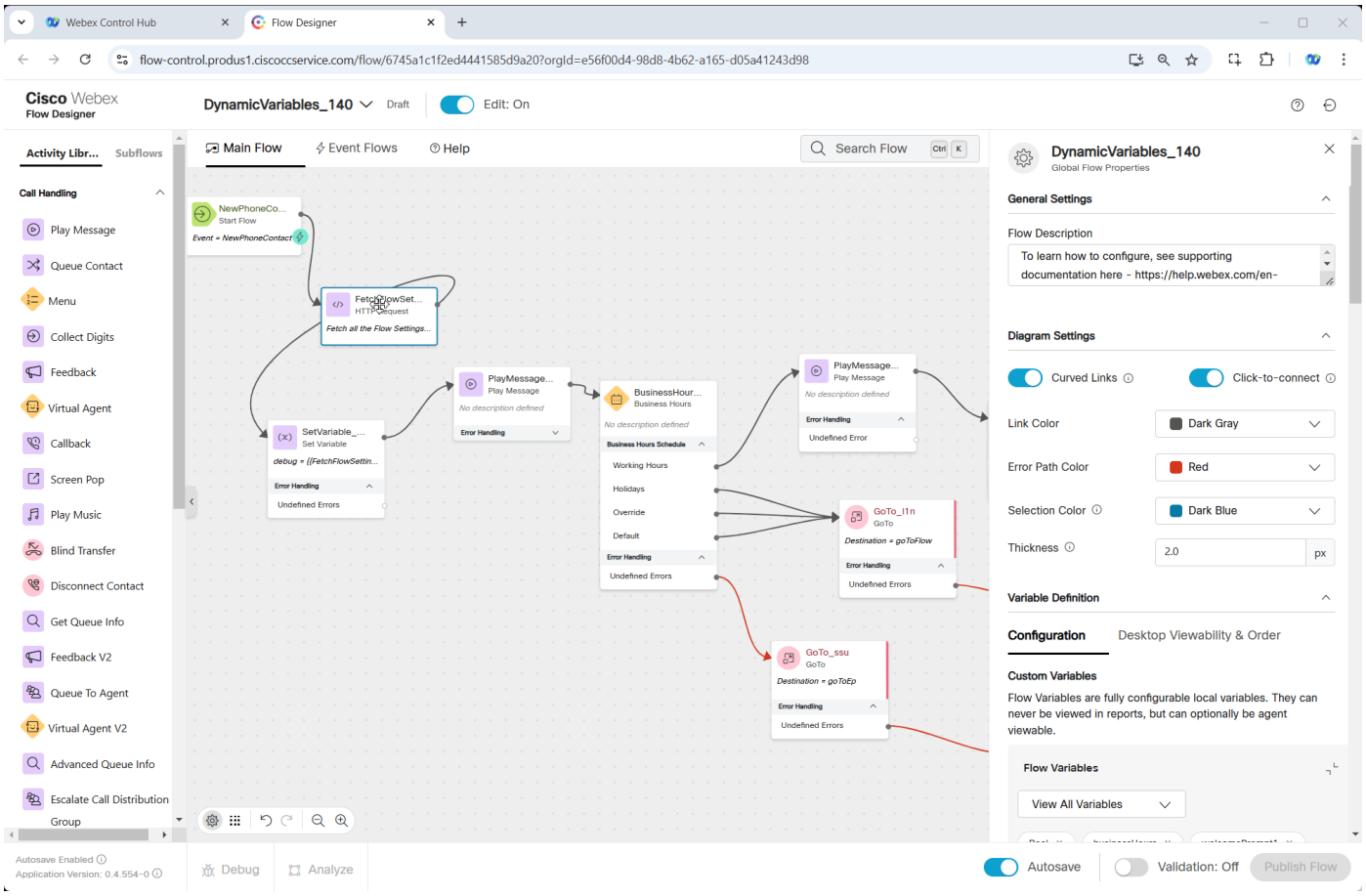
- **FetchFlowSettings** node is used to access external database over API and parse the result by writing response result into respective Flow Variables which have been preconfigured for you already.
- **SetVariable_mwn** node writes complete API response into debug variable so you could see the complete API call result in Debug tool. It's been taken from **FetchFlowSettings.httpResponseBody** output variable of **FetchFlowSettings** node
- All **Play Message** and **Play Music** nodes have been preconfigured to play TTS messages taken from respective API response
- **BusinessHours_os2** node set to **bussinessHours** variable which is your business hour entity **Your_Attendee_ID_Bussiness_Hours**
- **QueueContact_a62** node set to **queue** variable which is your queue entity **Your_Attendee_ID_Queue**
- Some **GoTo** nodes are configured to use variables and some have static values. We will adjust them while going through further steps.



6. Select **FetchFlowSettings** HTTP Node and paste the following GET request in Request URL field by replacing a templated one:

   https://674481b1b4e2e04abea27c6e.mockapi.io/flowdesigner/Lab/DynVars?dn={{NewPhoneContact.DNIS | slice(2) }}

7. In the same node, under Parsing Settings add **[0]** after **$** sign to the path expression of each output variable. This needs to be done due to output structure of API response.

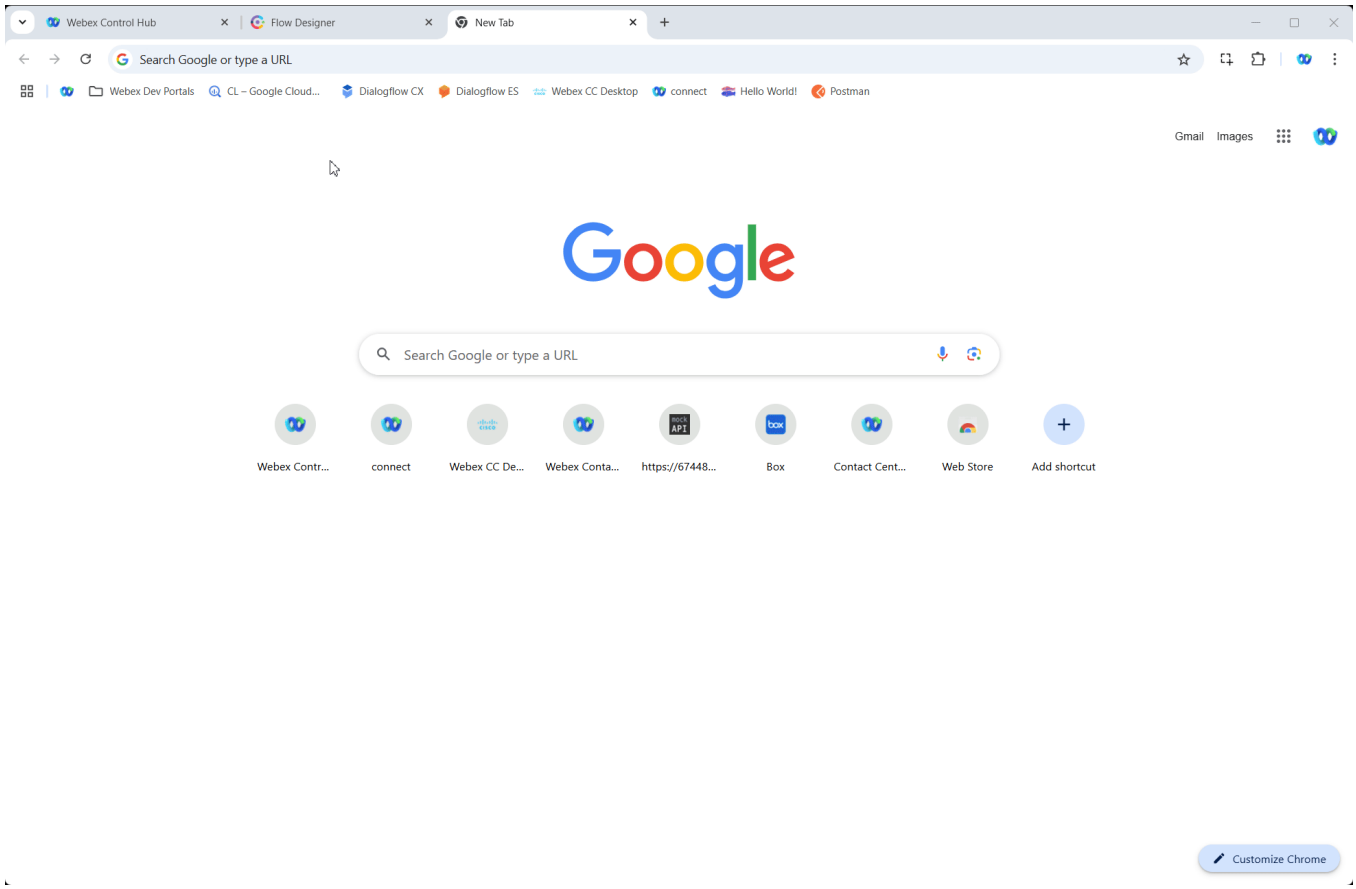


st your API Source [Optional]

a. Test your API resource. **https://674481b1b4e2e04abea27c6e.mockapi.io/flowdesigner/Lab/DynVars?dn={DNIS}**

b. Replace DNIS with the provided DNIS number stripping **+1**. [For example:] If your number **+14694096861**, then your GET Query should be ***https://674481b1b4e2e04abea27c6e.mockapi.io/flowdesigner/Lab/DynVars?dn=4694096861***

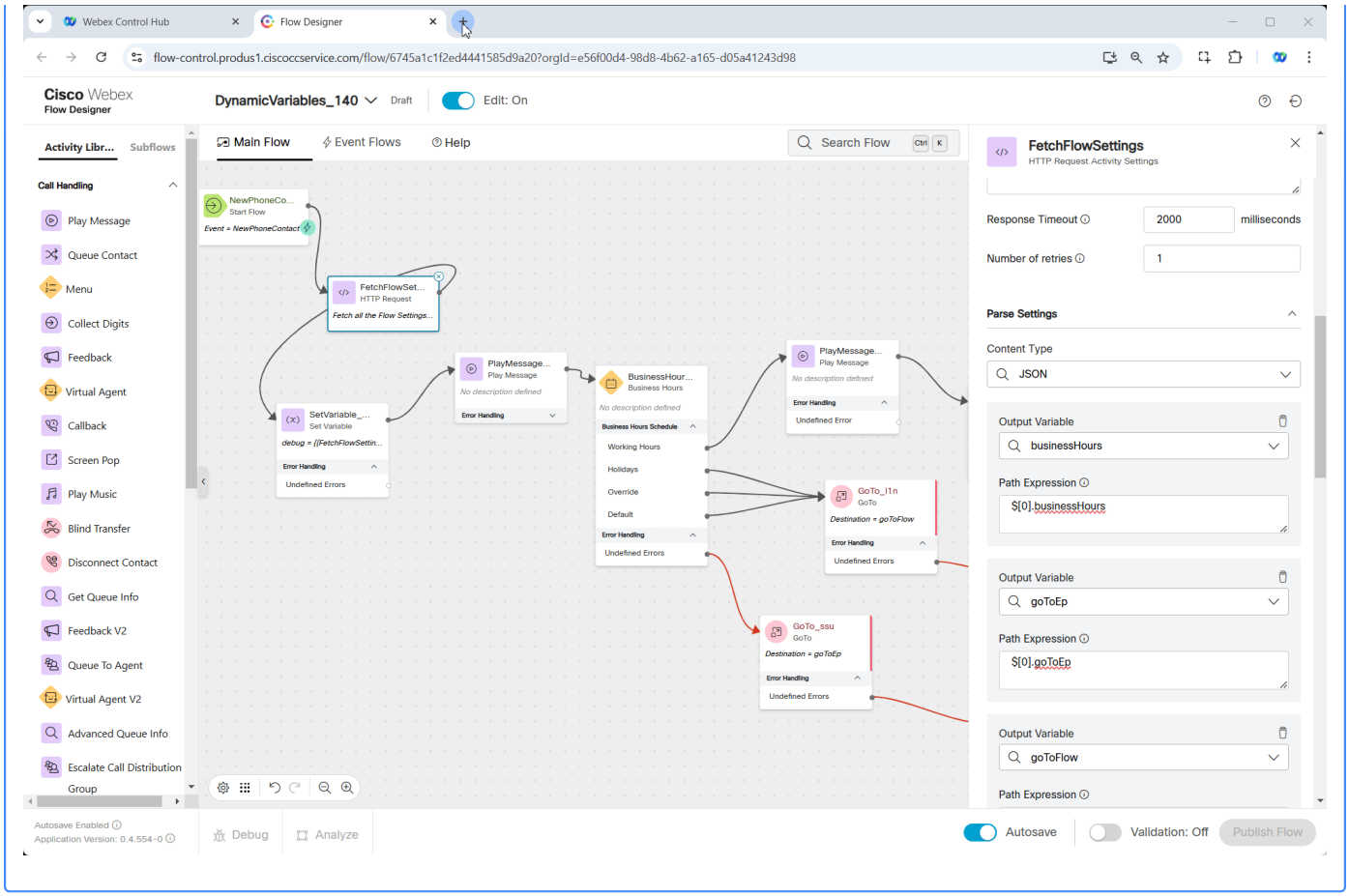
c. Open Chrome browser, paste your Get query URL into the Browser address line and press Enter. You should get the JSON response like this:



d. Open the new browser tab and navigate to JSONPath Online Evaluator

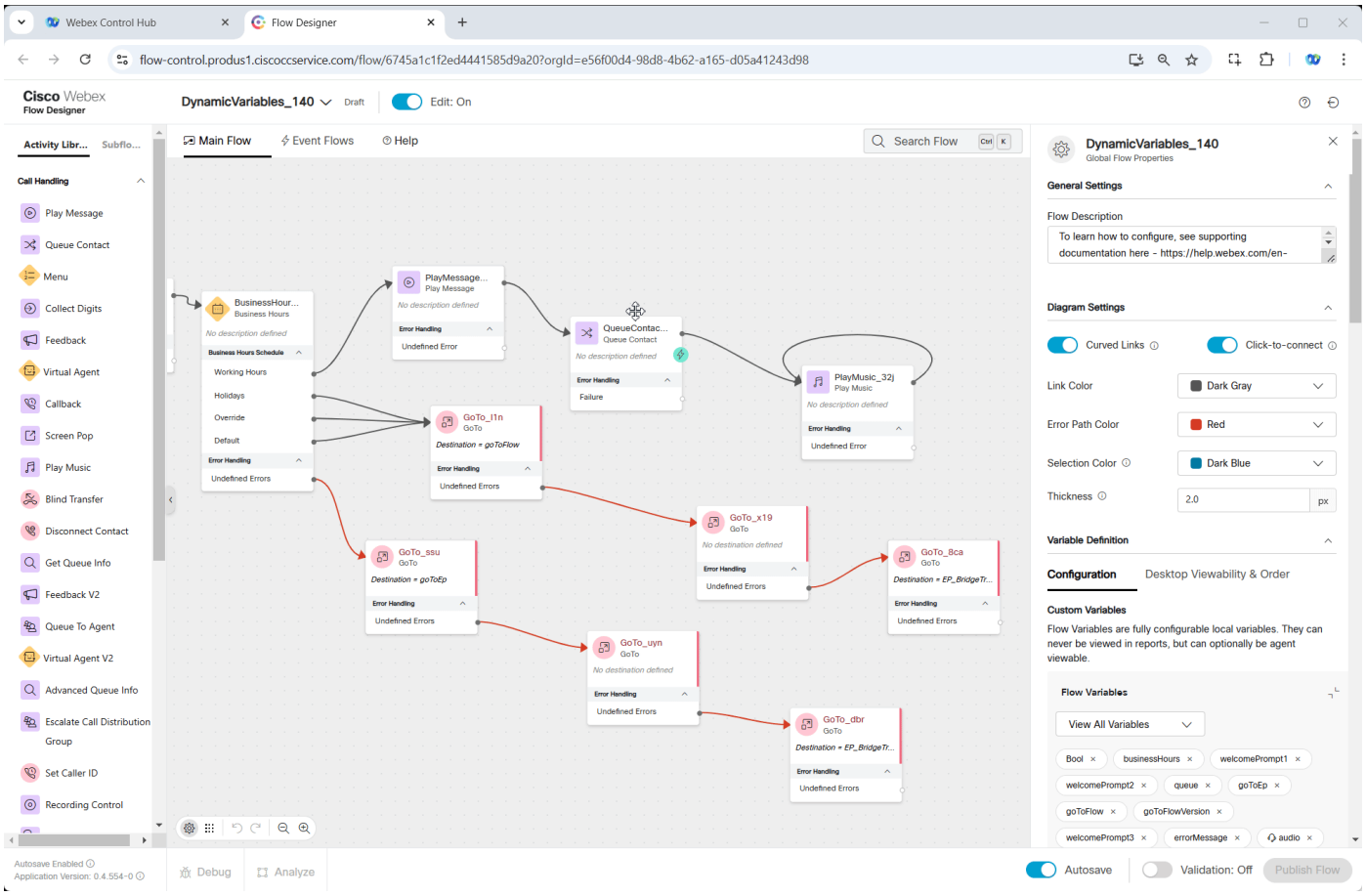
e. Copy the JSON response (including square brackets) you obtained above and paste it into **Document** window of **JSONPath Online Evaluator**.

f. In **JSONPath** box copy and paste one of the path expression from **FetchFlowSettings** to verify your results. For example, **$[0].businessHours**



8. Open a **Queue** Node and set **Fallback Queue** to **CCBU_Fallback_Queue**. That is needed to make sure the call will find an end queue in case API GET call fails.
9. Open **GoTo_x19** node and set:
   - Destination Type: **Flow**
   - **Static Flow**
   - Flow: **CLTS_ErrorHandling_Flow**

   Choose Version Label: **Latest**
10. Open **GoTo_8ca** and set:
    - Destination Type: **Entry Point**
    - **Static Entry Point**
    - Entry Point: **CLTS_ErrorHandling_Channel**
11. Repeat node settings in **Step 9** for **GoTo_uyn**
12. Repeat node settings in **Step 10** for **GoTo_dbr**

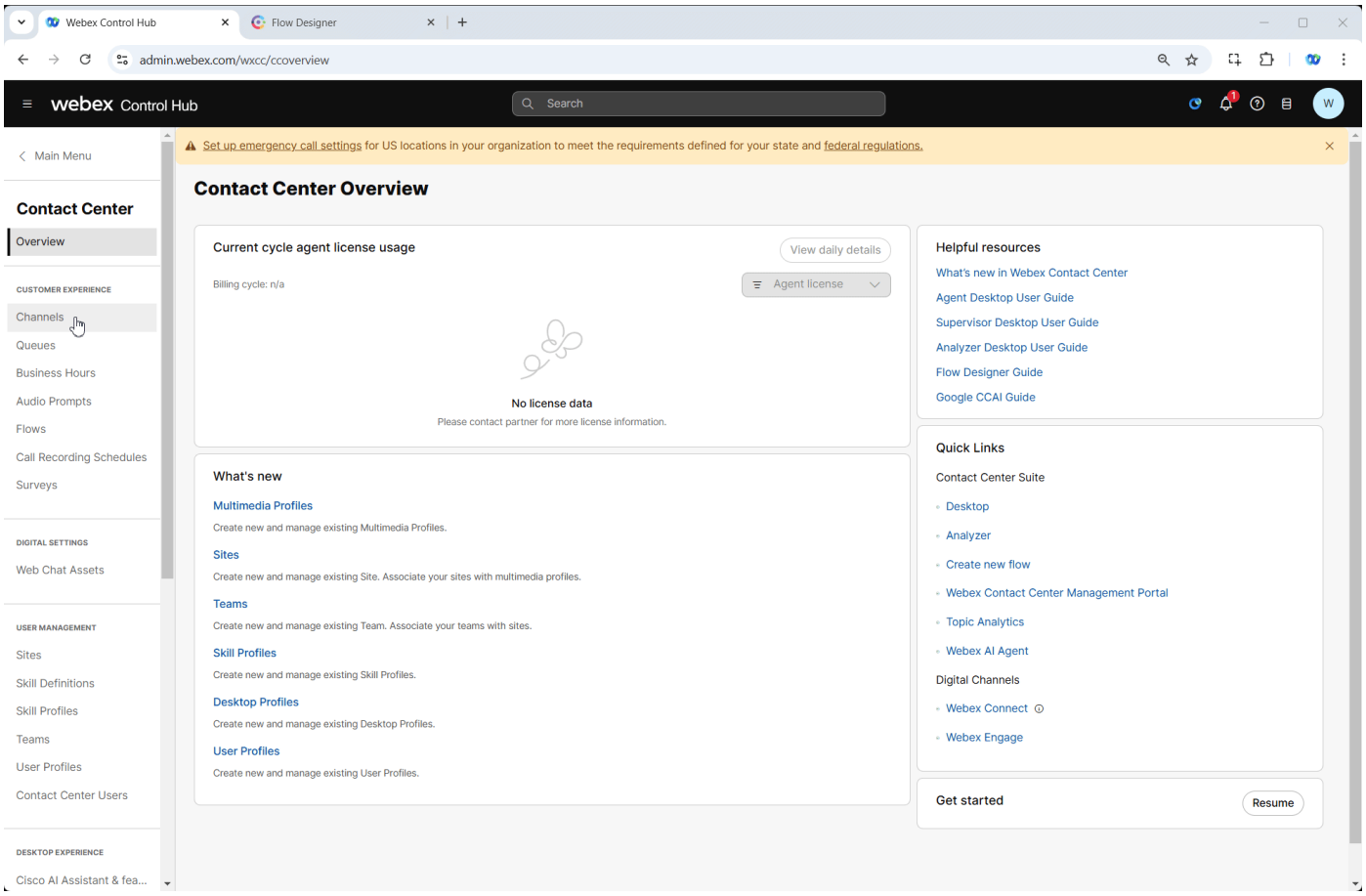


13. Validate and publish the flow:

- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.

14. Switch to Control Hub and navigate to **Channels** under Customer Experience Section

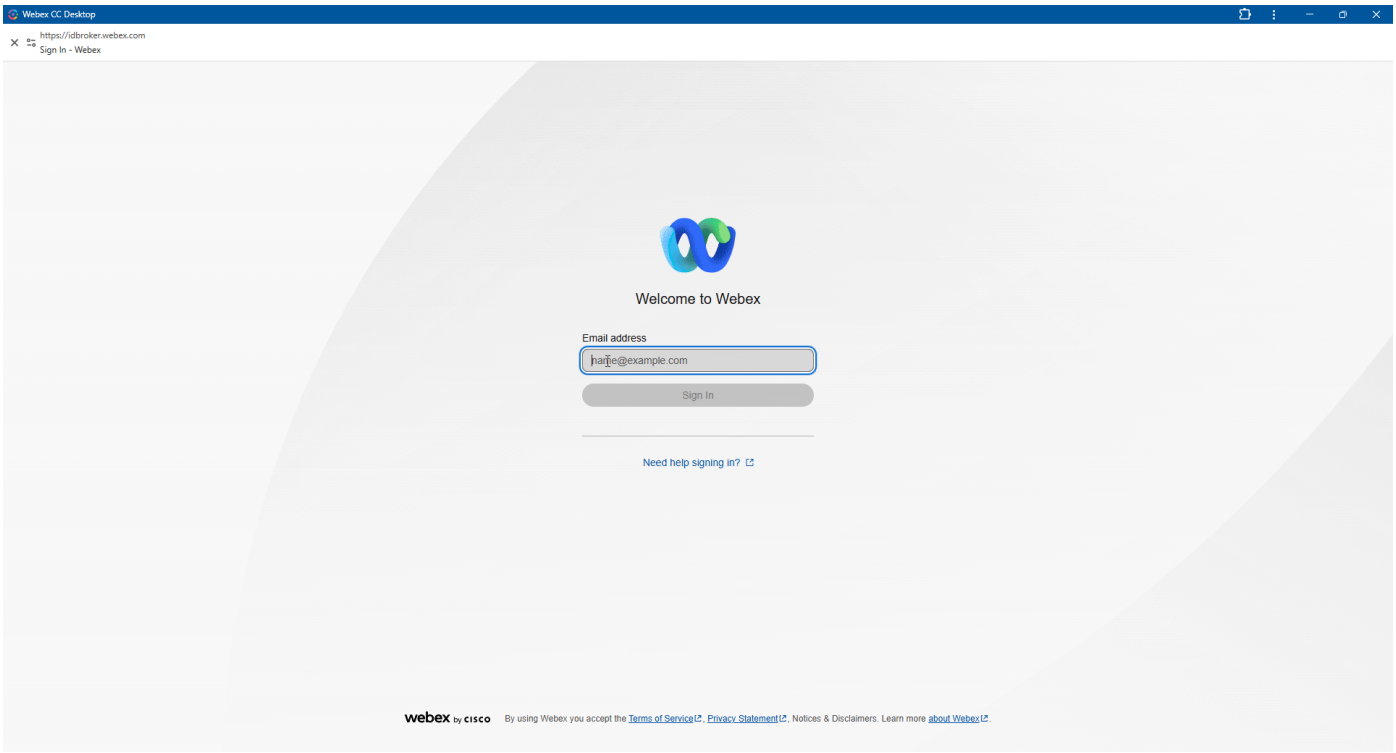
- Locate your Inbound Channel (you can use the search): **Your_Attendee_ID_Channel**
- Select the Routing Flow: **DynamicVariables_Your_Attendee_ID**
- Select the Version Label: **Latest**
- Click **Save** in the lower right corner of the screen



Testing

1.

   Your Agent desktop session should be still active but if not, use Webex CC Desktop application and login with agent credentials you have been provided **wxcclabs+agent_IDYour_Attendee_ID@gmail.com**. You will see another login screen with Webex icon on it where you may need to enter the email address again and the password provided to you.
2. Select Team **Your_Attendee_ID_Team**. Click **Submit**. Allow browser to access Microphone by clicking **Allow** on ever visit.
3. Make your agent **Available** and you're ready to make a call.

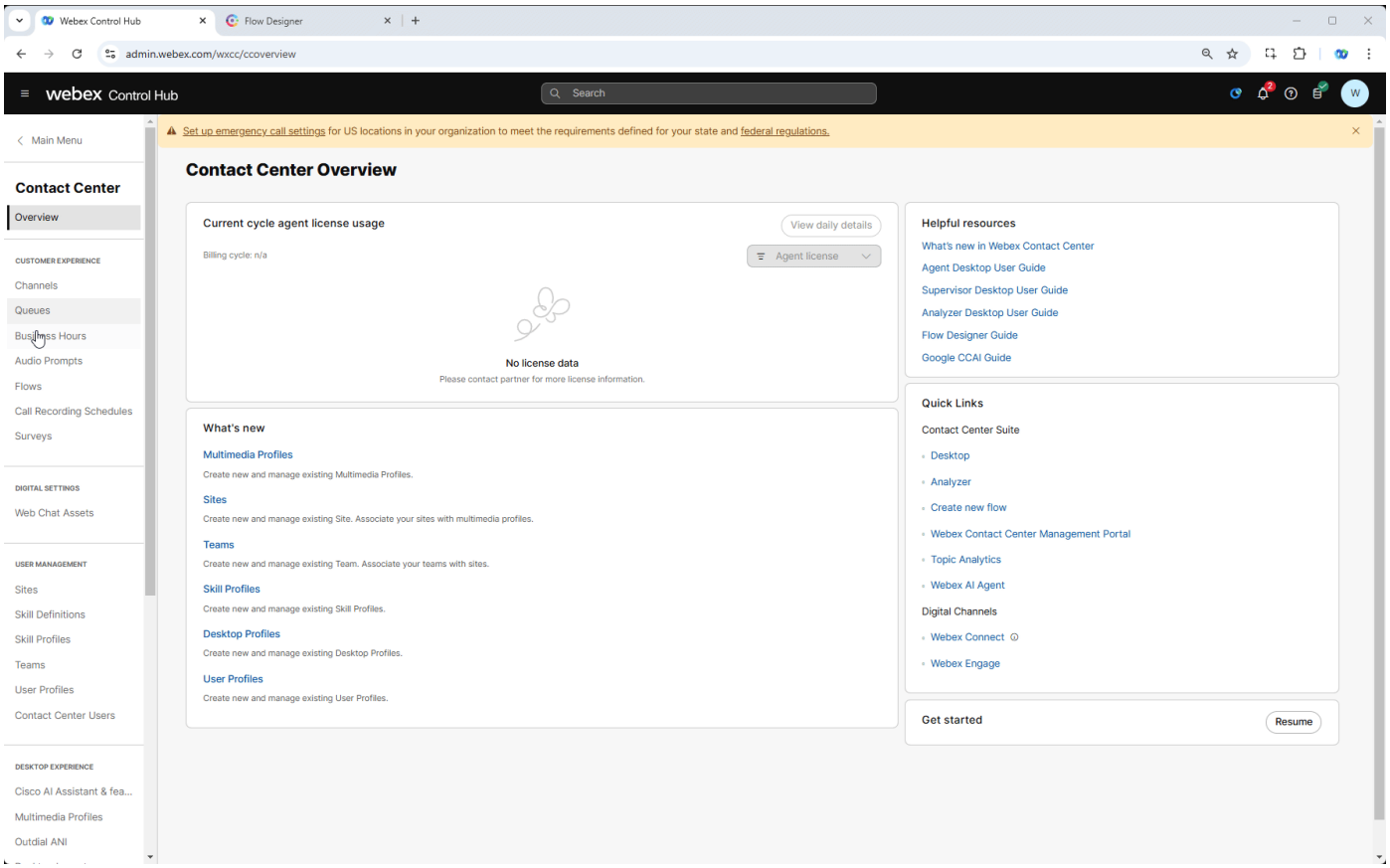


4. Make a call to test you flow. If everything configured as per instructions you should hear a **welcome1** message that is a value of ***$[0].welcomePrompt1*** and then ***$[0].welcomePrompt2***. Finally, the call should land on the ***$[0].queue***

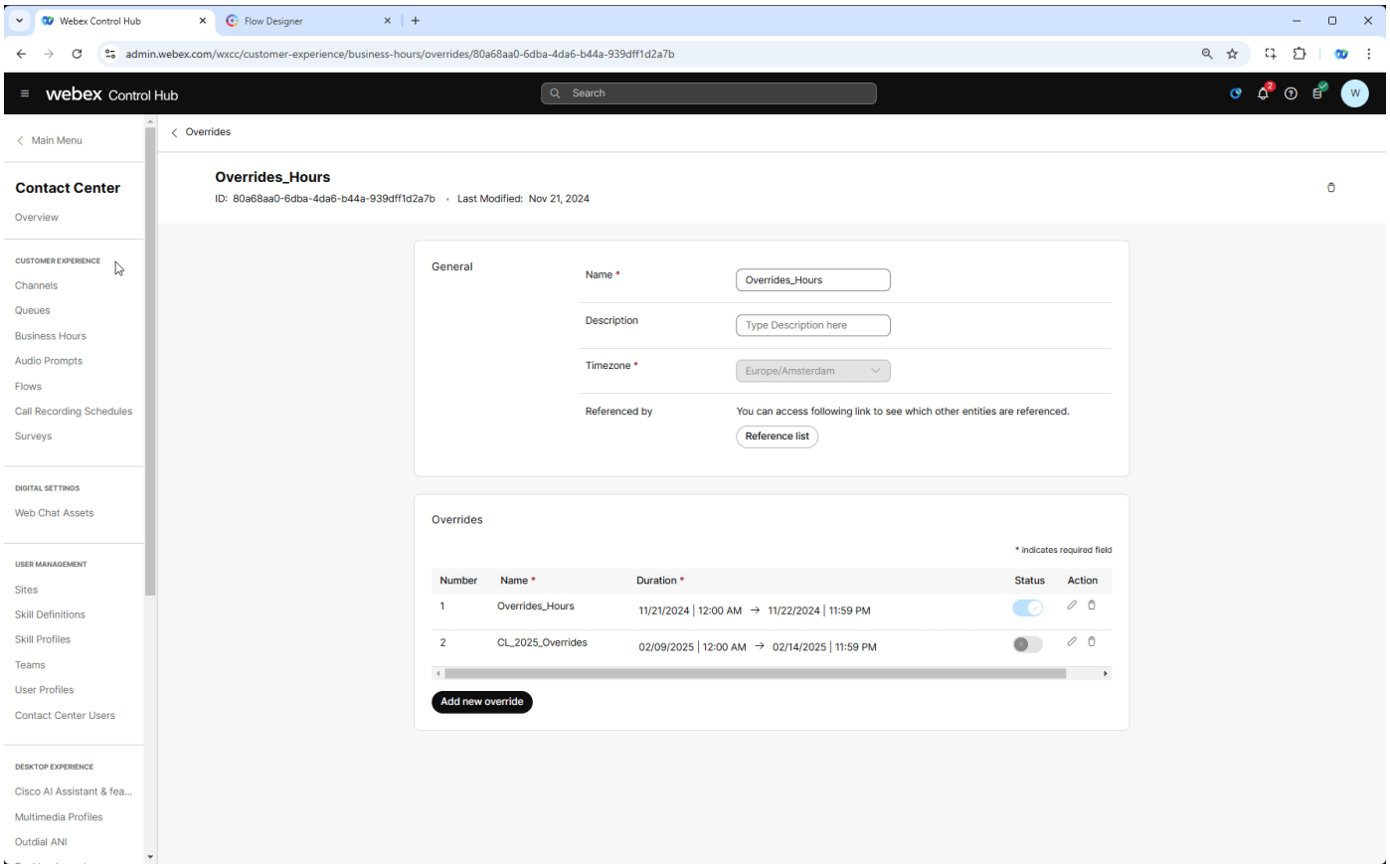
[OPTIONAL] TEST OTHER VARIABLES

1. You can do the same trick we did in Mission 2 of Core Track and use **Override** option to change the logic. Overrides as well as Business hours have been preconfigured for you. Now we need to apply it on your **Your_Attendee_ID_Bussiness_Hours** entity. Open **Your_Attendee_ID_Bussiness_Hours** in **Control Hub**, scroll down to Additional Settings and select **Overrides_Hours** from Override dropdown list. Then click Save.

> **Note**
>
> Override Hours entity overwrites Working Hours and set to duration of current Cisco Live lab



2. Make a new call to be redirected to flow ***$[0].goToFlow*** where the following message can be heard: **"Thanks you for calling. You are now on Error Handling flow and will be redirected to Global Support line in a moment. Goodbye."**
3. Now we need to revert the configuration we made in Step 1. Open **Your_Attendee_ID_Bussiness_Hours** in **Control Hub**, scroll down to **Additional Settings** and select **None** from **Override** dropdown list. Then click **Save**.



Congratulations, you have succesfully completed Routing Facilitation mission! 🎉

3.1.4 Mission 3: Last Agent Routing

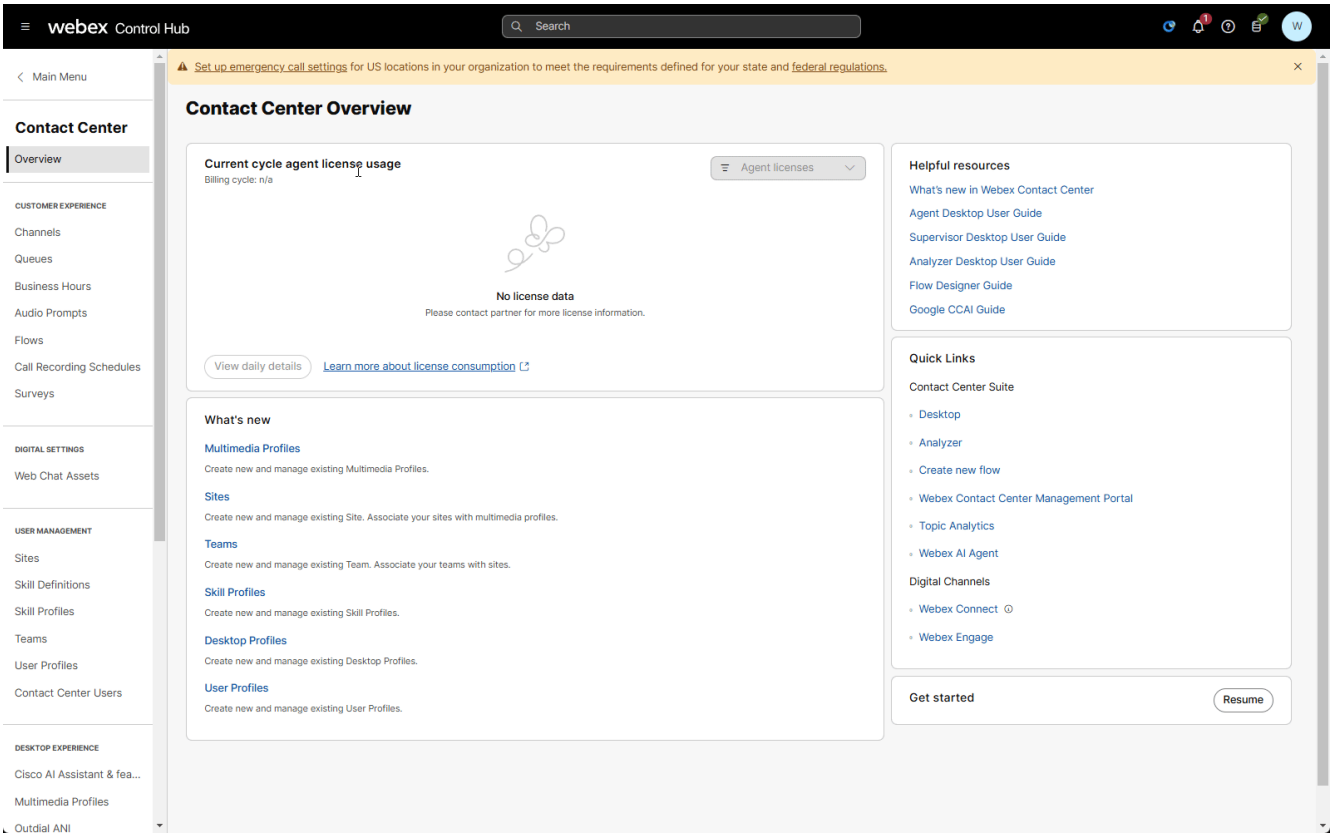
Story

A common request for returning customers calling into a contact center is to work with the last person with which they had a good experience. This may be because they are already familiar with what the customer needs, or it may just be that the customer is familiar with the agent and enjoyed their last interaction. With the new Auto CSAT feature in the Webex Contact Center we can automatically account for this request and route to the last agent which had a high Auto CSAT with the customer.

[Important!] Since this is a lab environment where you will act as both the customer and the agent, accurately scoring a call will be challenging. Additionally, Auto CSAT has not been taught due to the insufficient number of calls required for AI to learn and generate proper scoring. In this lab, we will use a Global Variable to store the score, which is also used for Auto CSAT teaching. With a sufficient number of provided scores, Auto CSAT will eventually be able to score calls automatically.

Note

[Important!] Auto CSAT is preconfigured for you on Control Hub. Please avoid changing the configuration.



Call Flow Overview

1. New call comes into the flow.
2. Call the Search API to find the last agent with which they had a good Auto CSAT.
3. If the Auto CSAT is greater or equal **4** and agent is available, we will route the call to that agent.
4. If the agent is not available or if no recent good AutoCSAT scores exist for the caller, we will route the call to the queue for the next available agent.

Mission Details

Your mission is to:

1. Create a new flow from the scratch.
2. Build a Search API query to request information from Analyzer database and parse it into flow variables.
3. Prioritize the call if conditions match and route the call to agent.

Preconfigured elements

1. Wait treatment Subflow which will provide Music in Queue and Queue Messages.
2. Auto CSAT flow **CCBU_PostCallSurvey_AutoCSAT** has been created to help contact centers efficiently gather customer feedback through a simple automated post-call survey using DTMF tones.

Build

1. Create a new flow from scratch named **LastAgentRouting_Your_Attendee_ID** and add these flow variables:

   Agent ID variable:

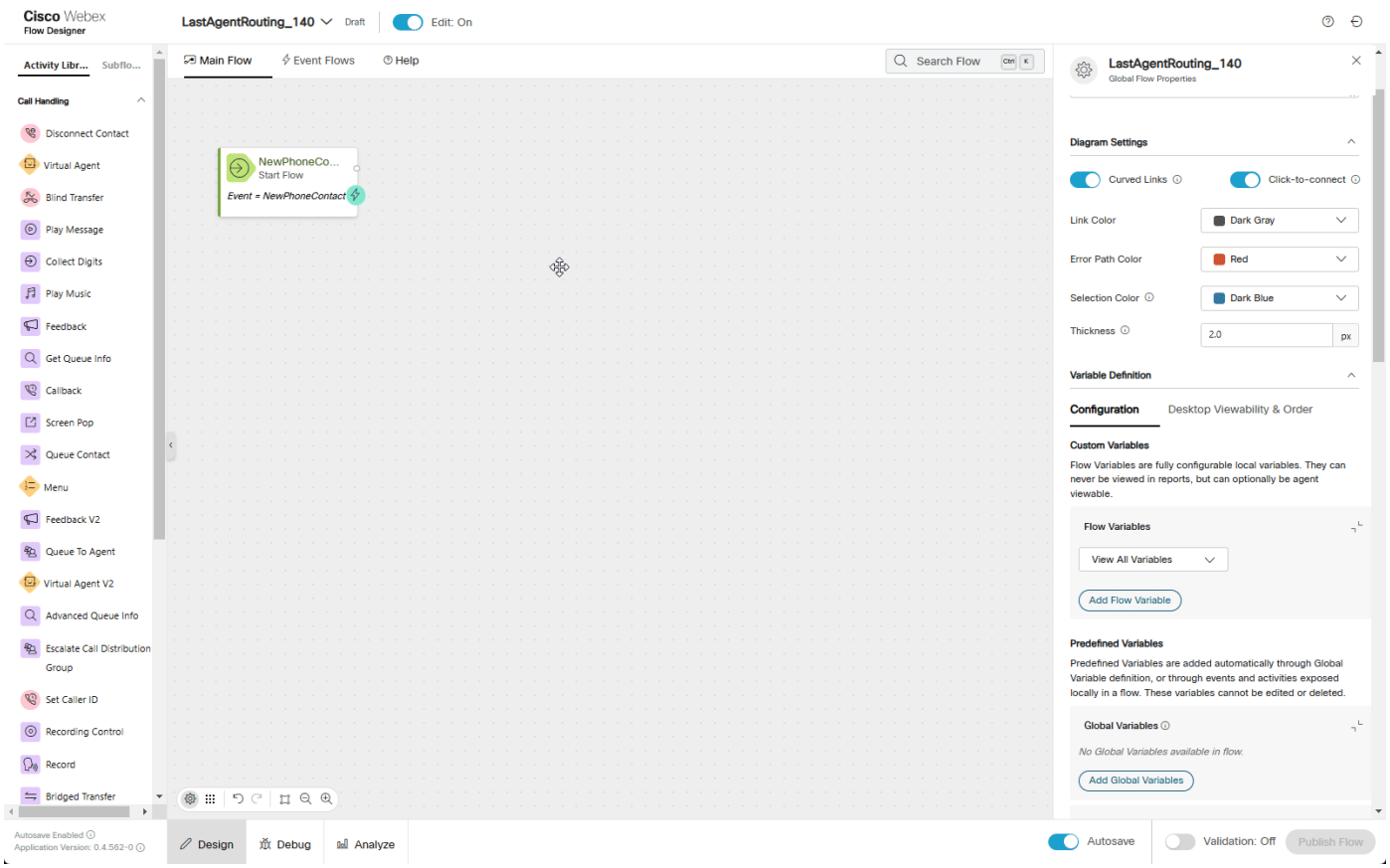
   - Name: **agentID**
   - Type: **String**
   - Default Value: leave it empty

   Variable to write HTTP Response into it:

   - Name: **JSONResponse**
   - Type: **String**
   - Default Value: leave it empty

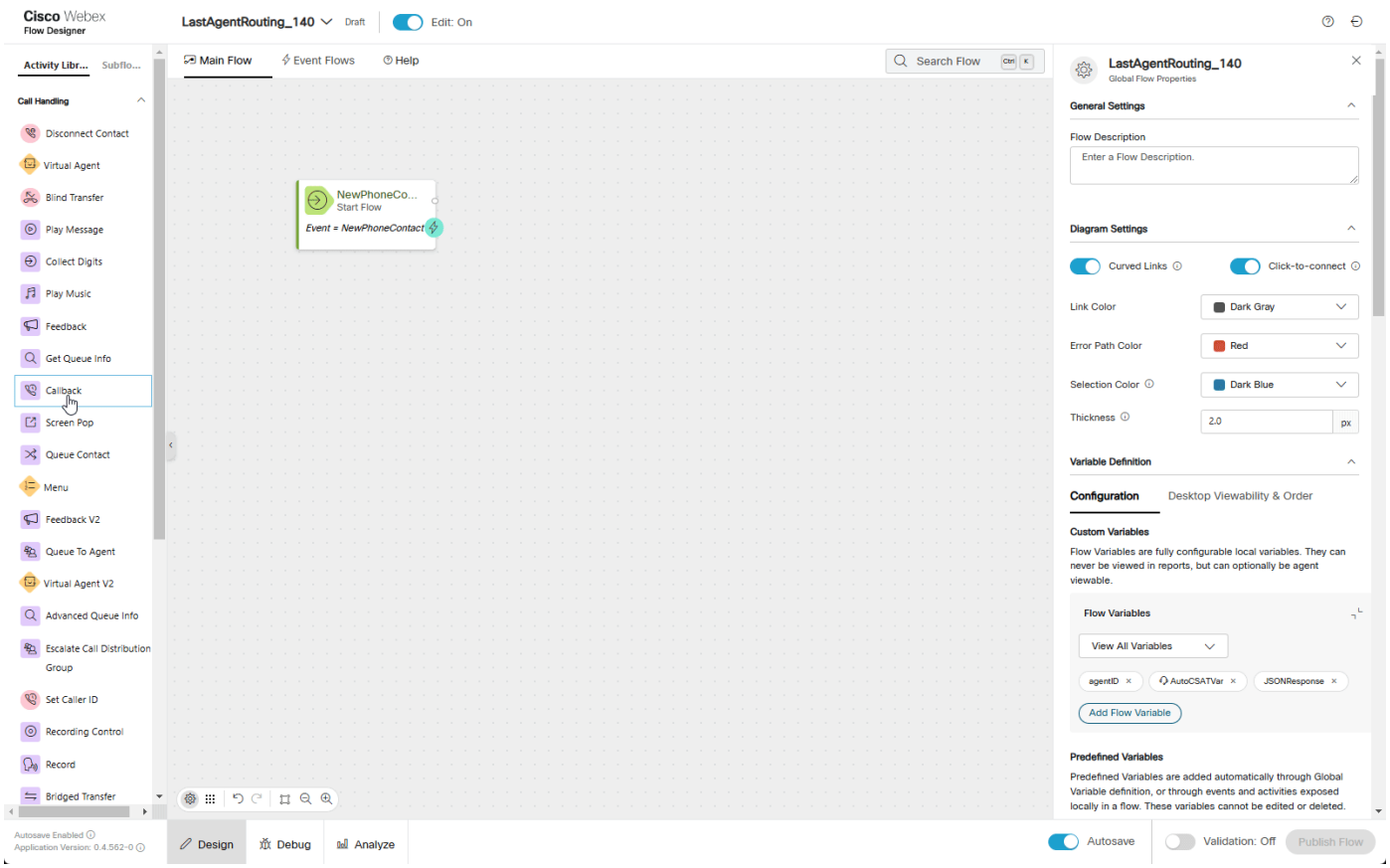
   String type AutoCSAT variable:

   - Name: **AutoCSATVar**
   - Type: **Decimal**
   - Default Value: **0.0**
   - Turn on **Agent Viewable** toggle
   - Desktop Label: **Auto CSAT**



2. Add a **Play Message** node

- Connect the **New Phone Contact** node edge to this **Play Message** node
- Turn on **Enable Text-To-Speech** toggle
- Select the Connector: **Cisco Cloud Text-to-Speech**
- Click the **Add Text-to-Speech Message** button
- Delete the selection for Audio File
- Text-to-Speech Message: ***Welcome to the last agent routing mission.***



3. Add an **HTTPRequest** node for our query

- Activity Label: **GraphQL_Query**
- Connect the output node edge from the **Play Message** node to this node
- Select Use Authenticated Endpoint
- Connector: **WxCC_API**
- Request Path: **/search**
- Method: **POST**
- Content Type: **GraphQL**
- Copy this GraphQL query into the Request Body:

```
query lastagentSearch($from: Long!, $to: Long!, $filter: TaskDetailsFilters) {
taskDetails(from: $from, to: $to, filter: $filter) {
    tasks {
      csatScore
      autoCsat
      origin
      owner {
        id
        name
      }
      doubleGlobalVariables(name: "AutoCSAT_GV"){
        name
        value
      }
    }
  }
}
```

- Copy the following variables into the GraphQL Variables:

```
{
  "from": "{{now() | epoch(inMillis=true) - 15000000}}",
  "to": "{{now() | epoch(inMillis=true)}}",
  "filter": {
    "and": [
      {"lastEntryPoint": {"id": {"equals": "{{NewPhoneContact.EntryPointId}}" }}},
      { "status": { "equals": "ended" } },
      { "origin": { "equals": "{{NewPhoneContact.ANI}}" }},
      {
        "doubleGlobalVariables": {
          "name":   { "equals": "AutoCSAT_GV" },
          "value":  { "gte": 4 }
        }
      }
    ]
  }
}
```

Parse Settings:

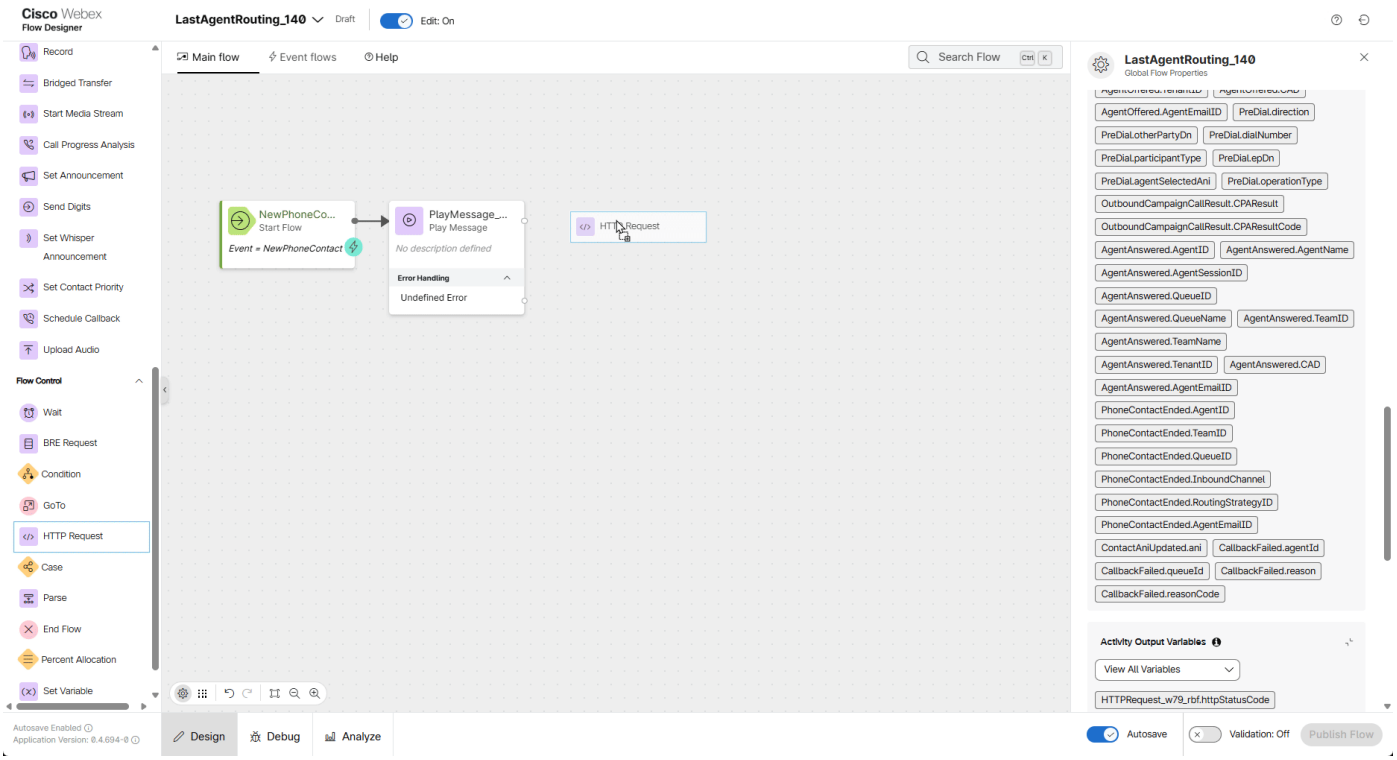
- Content Type: **JSON**
- Output Variable: `agentID`
- Path Expression: `$.data.taskDetails.tasks[0].owner.id`

Click on **+ Add New** to add new output variable

- Output Variable: `AutoCSATVar`
- Path Expression: `$.data.taskDetails.tasks[0].doubleGlobalVariables.value`

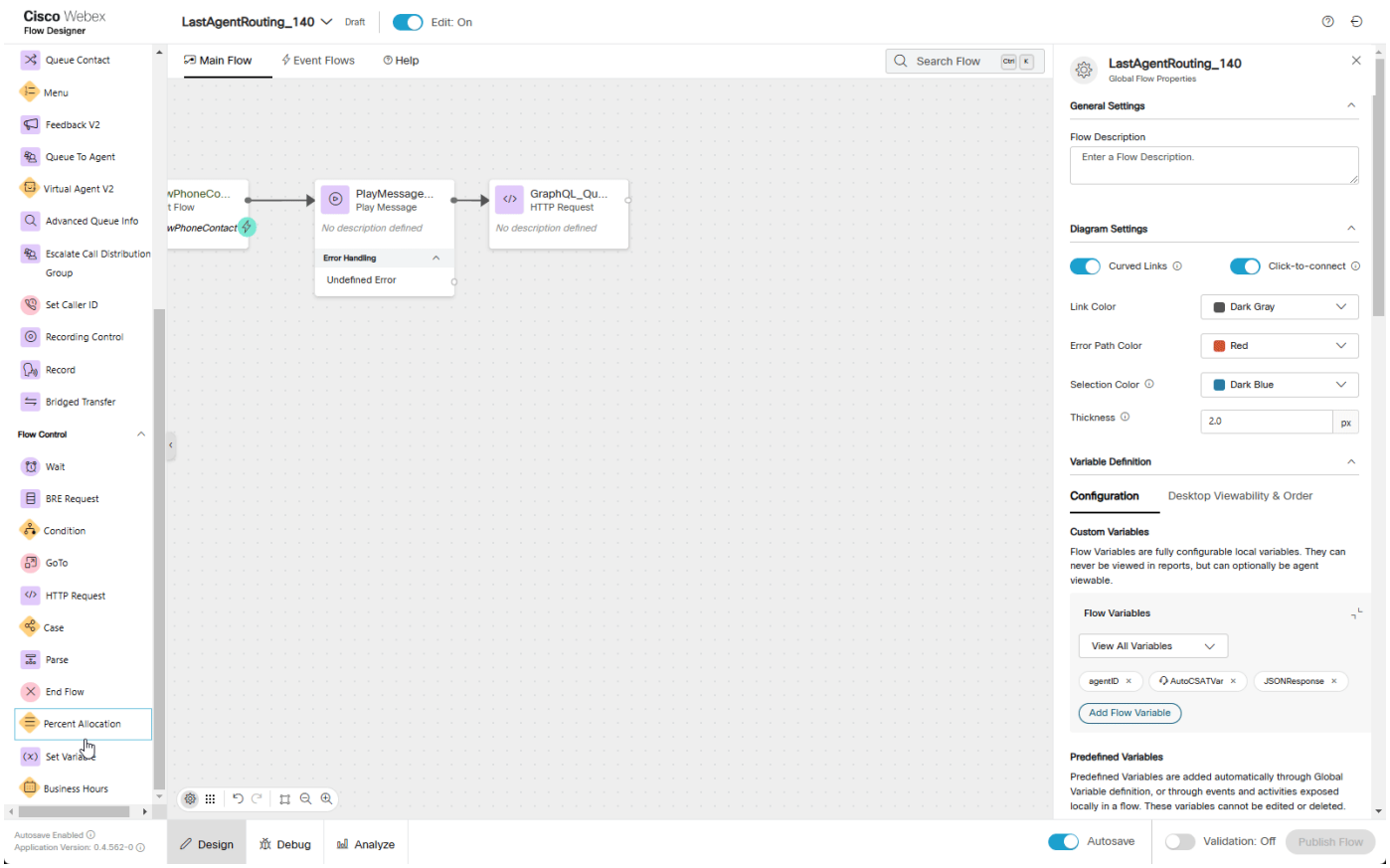
ample of expected response [Optional]


```
{
  "data": {
      "taskDetails": {
          "tasks": [
              {
                  "csatScore": 0,
                  "autoCsat": null,
                  "owner": {
                      "id": "b9b45479-756f-4c55-8663-8ae7800a9a18",
                      "name": "Agent140 Lab"
                  },
                  "doubleGlobalVariables": {
                      "name": "AutoCSAT_GV",
                      "value": 4.0
                  }
              }
          ]
      }
  }
}
```

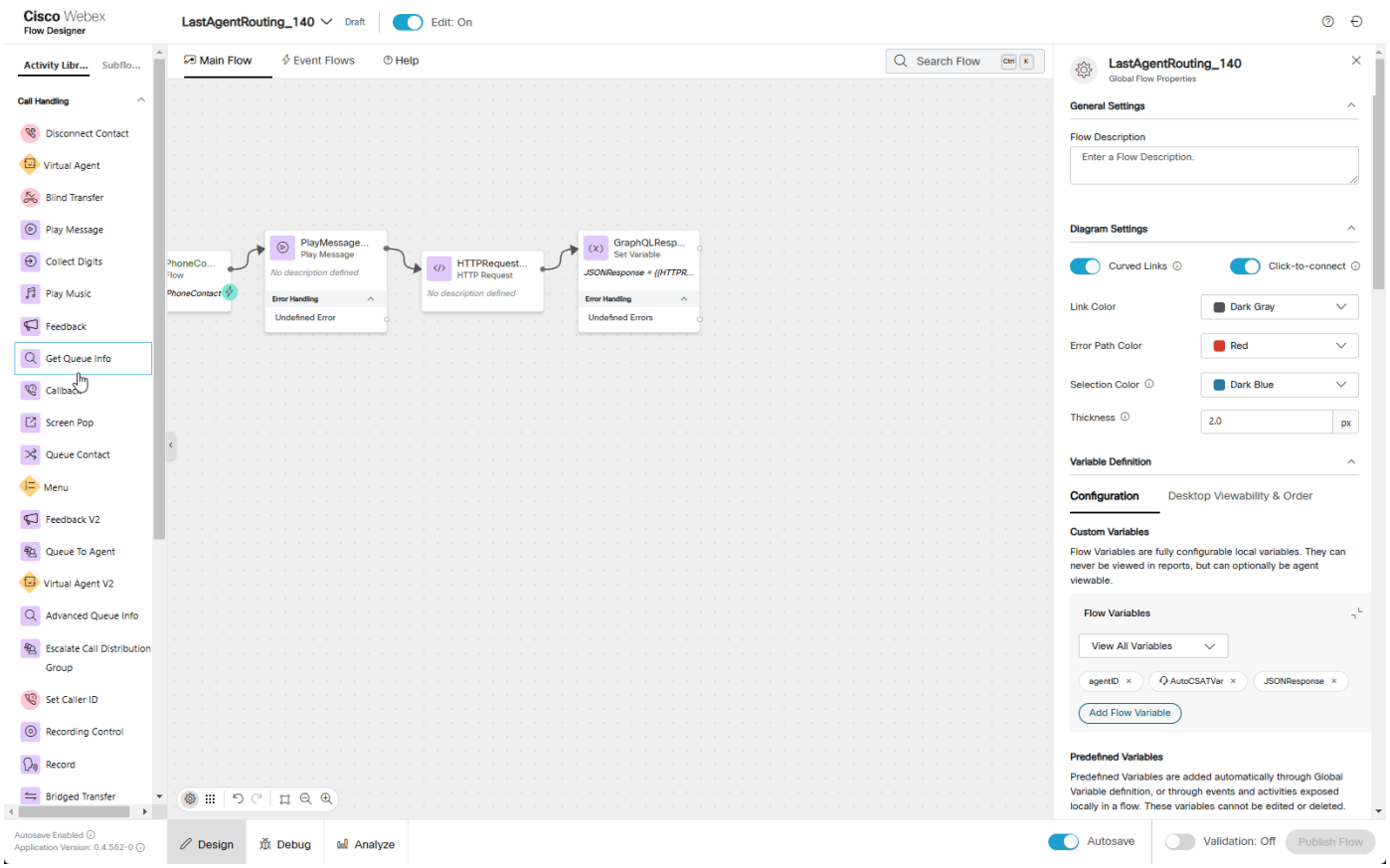
4. Add **Set Variable** node

- Activity Label: **GraphQL_Response**
- Connect **GraphQL_Query** to this node
- We will connect **Set Variable** node in next step
- Variable: **JSONResponse**
- Set To Variable: **GraphQL_Query.httpResponseBody**



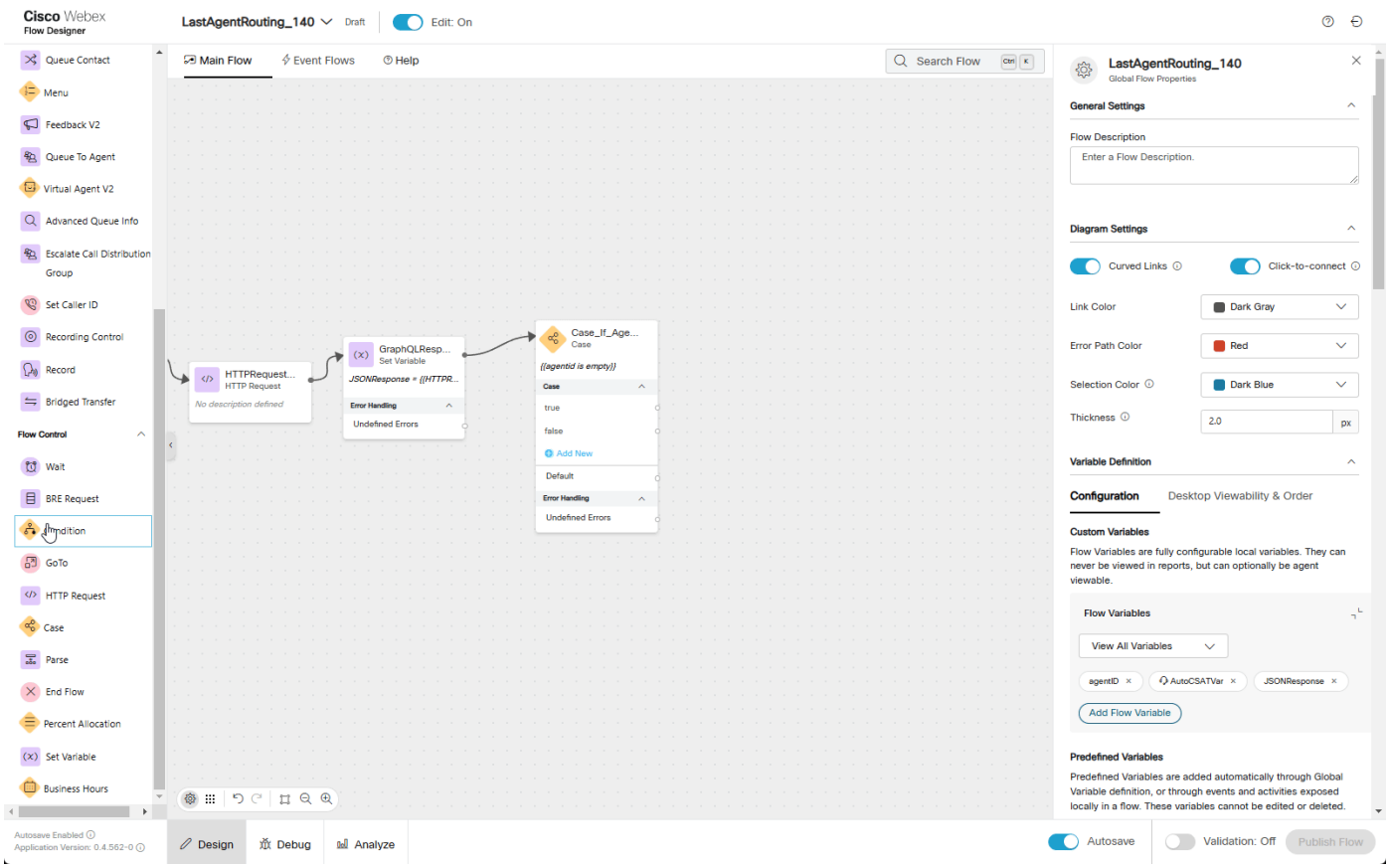
5. Add a **Case** node

- Activity Label: **Case_If_AgentIDEmpty**
- Connect the output node edge from teh **GraphQL_Response** node to this node
- Select **Build Expression**
- Expression: `{{agentID is empty}}`
- Change **Case 0** to **true**
- Change **Case 1** to **false**
- We will connect the **true** and **false** in future steps.



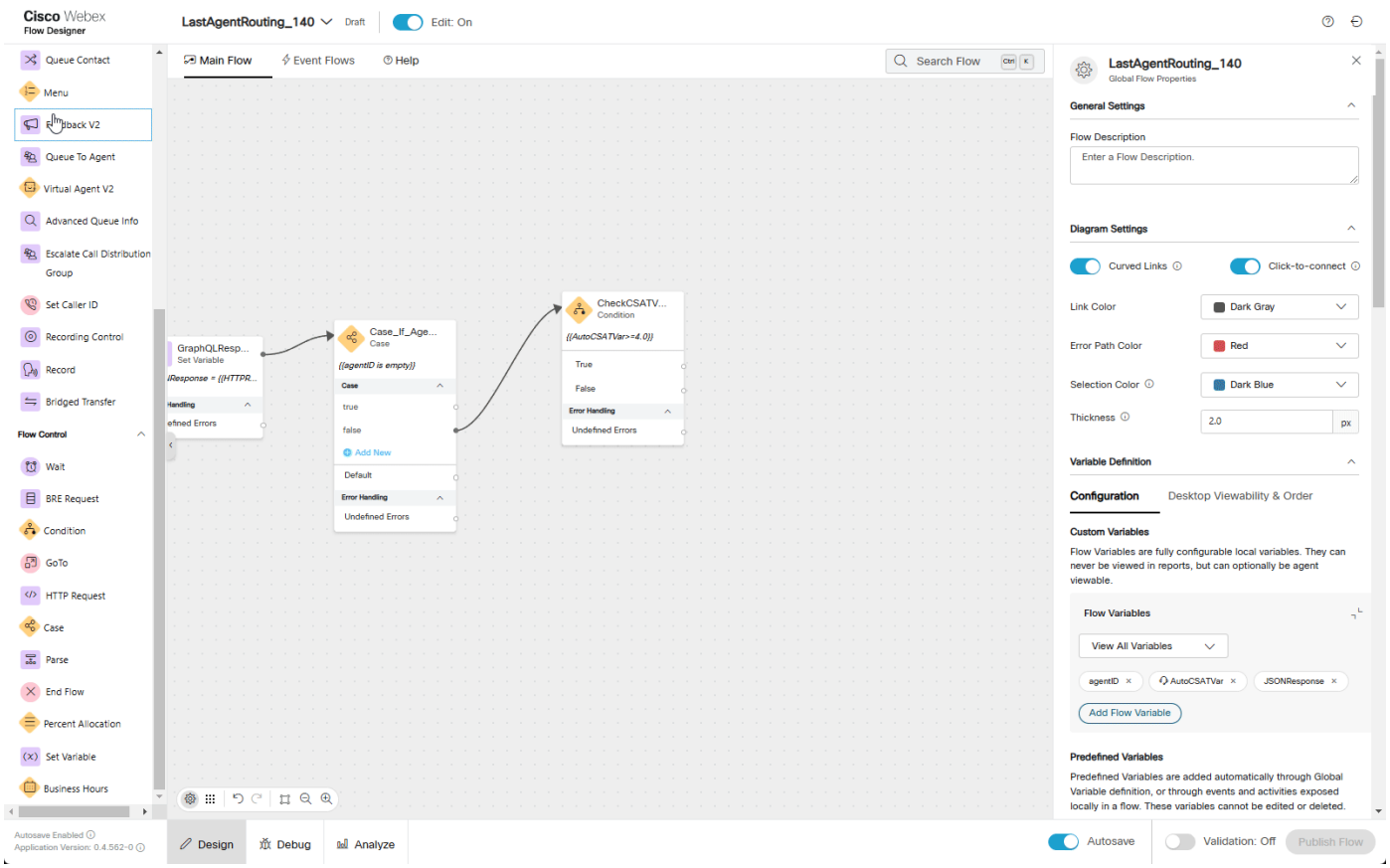
6. Add a **Condition** node

- Activity Label: **CheckCSATValue**
- Connect **false** exit of **Case** node to this node
- We will connect the **True** and **False** output edges in future steps.
- Expression: **{{AutoCSATVar>=4.0}}**



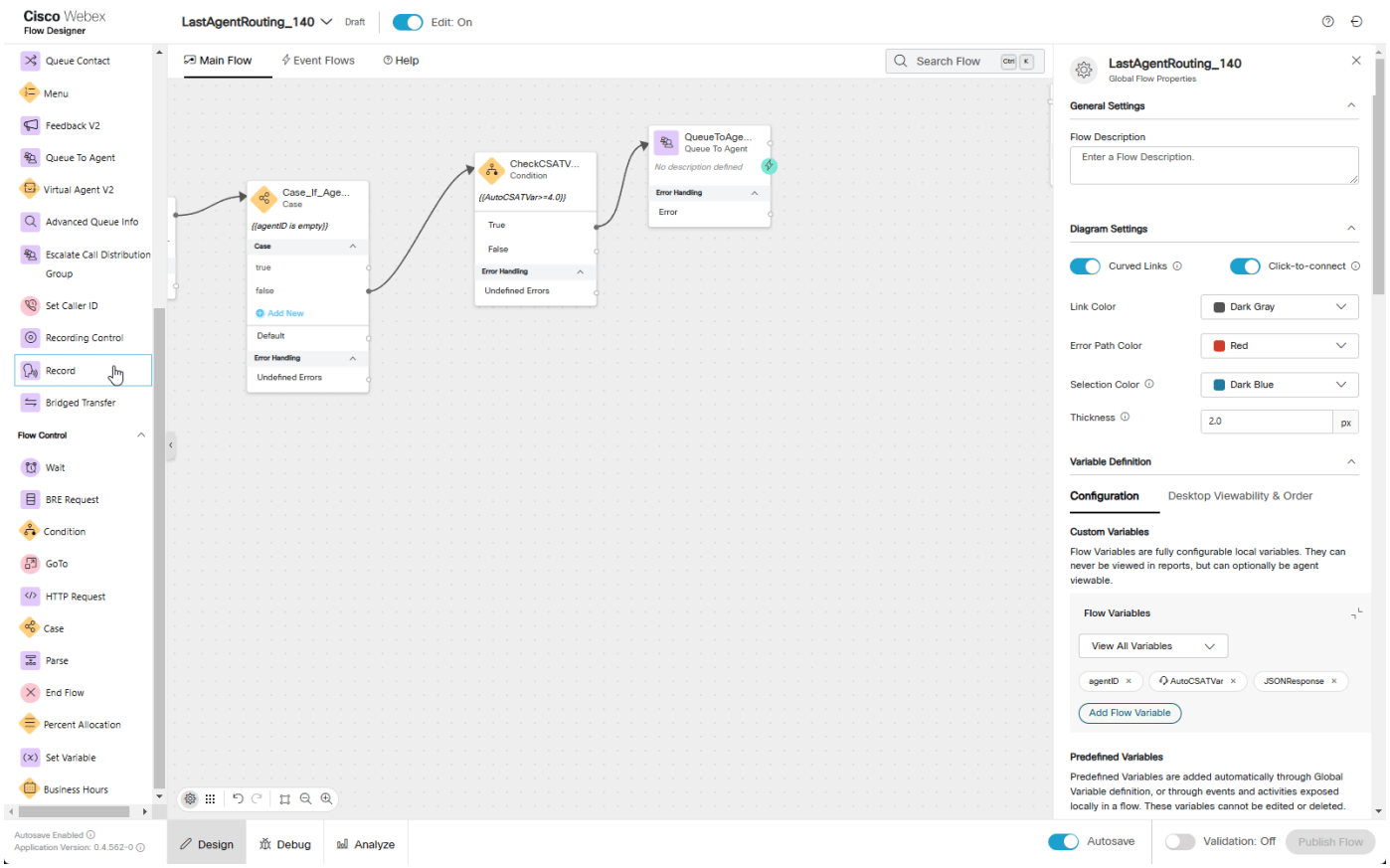
7. Add a **Queue To Agent** node

- Connect the **True** node edge of the **CheckCSATValue** condition node created in previous step to this **Queue To Agent** node.
- Agent Variable: **agentID**
- Agent Lookup Type: **ID**
- Turn on **Set Contact Priority**
- Select **Static Priority**
- Static Priority Value: **P1**
- Reporting Queue: **Your_Attendee_ID_Queue**
- **Park Contact if Agent Unavailable** toggle should remain off
- Static Recovery Queue: **Your_Attendee_ID_Queue**



8. Add a **Queue Contact** node

- Connect the **False** node edge from the **CheckCSATValue** condition node created in **Step 6** to this node
- Connect **true** node edge of **Case_If_AgentIDEmpty** node created in **Step 5** to this node
- Connect **Default** node edge of **Case_If_AgentIDEmpty** node created in **Step 5** to this node
- Connect **Queue To Agent** Output and Error node edges created in previous step to this **Queue Contact**
- Select **Static Queue**
- Queue: **Your_Attendee_ID_Queue**

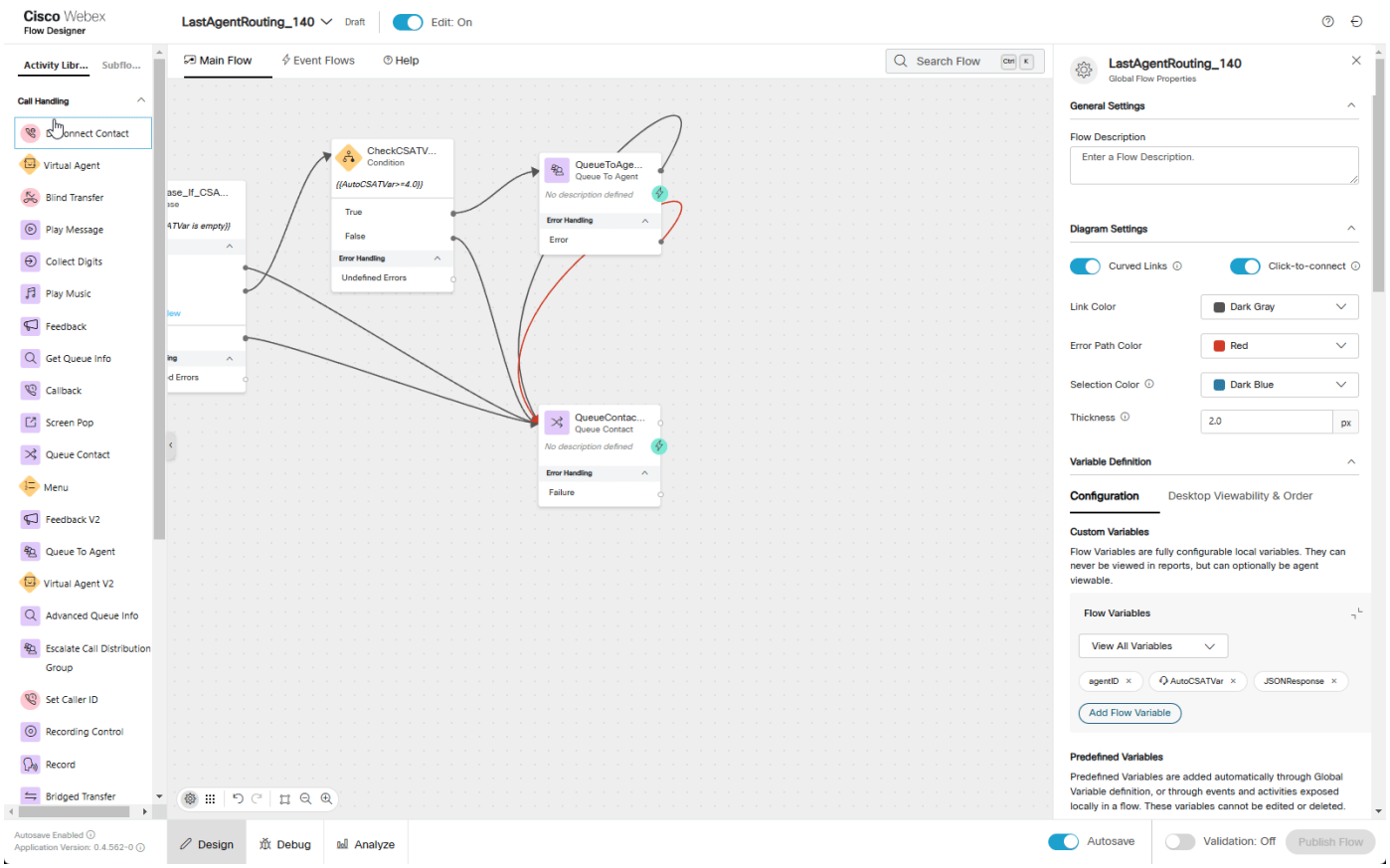


9. Add a **Subflow** node and **DisconnectContact** node

- In the Activity Library pane on the left side of the screen, click **Subflows**
- Find the **Subflow** names **WaitTreatment** and drag it onto the flow canvas like you would any other node.
- Add a **DisconnectContact** node
- Connect the output node edge from this node to the **DisconnectContact** node.
- Connect the **Queue Contact** node edge that we created in previous step to the **WaitTreatment** subflow node

Click on **WaitTreatment** subflow node and configure the following settings:

- Subflow Label: **Latest**
- Make sure **Enable automatic updates** is turned on

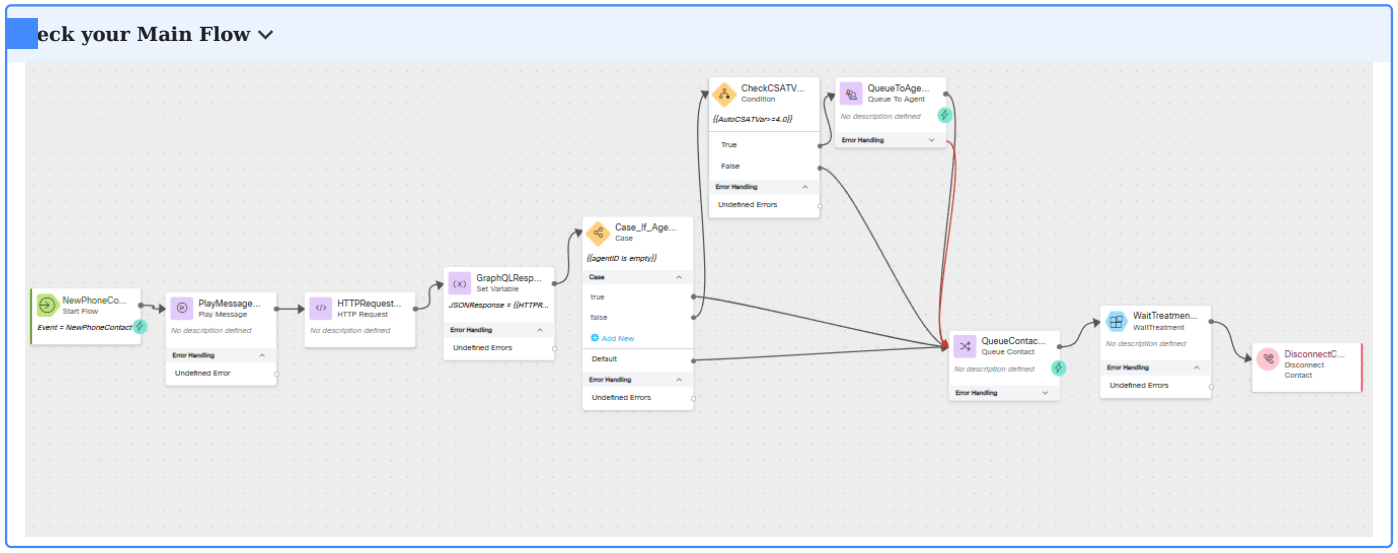


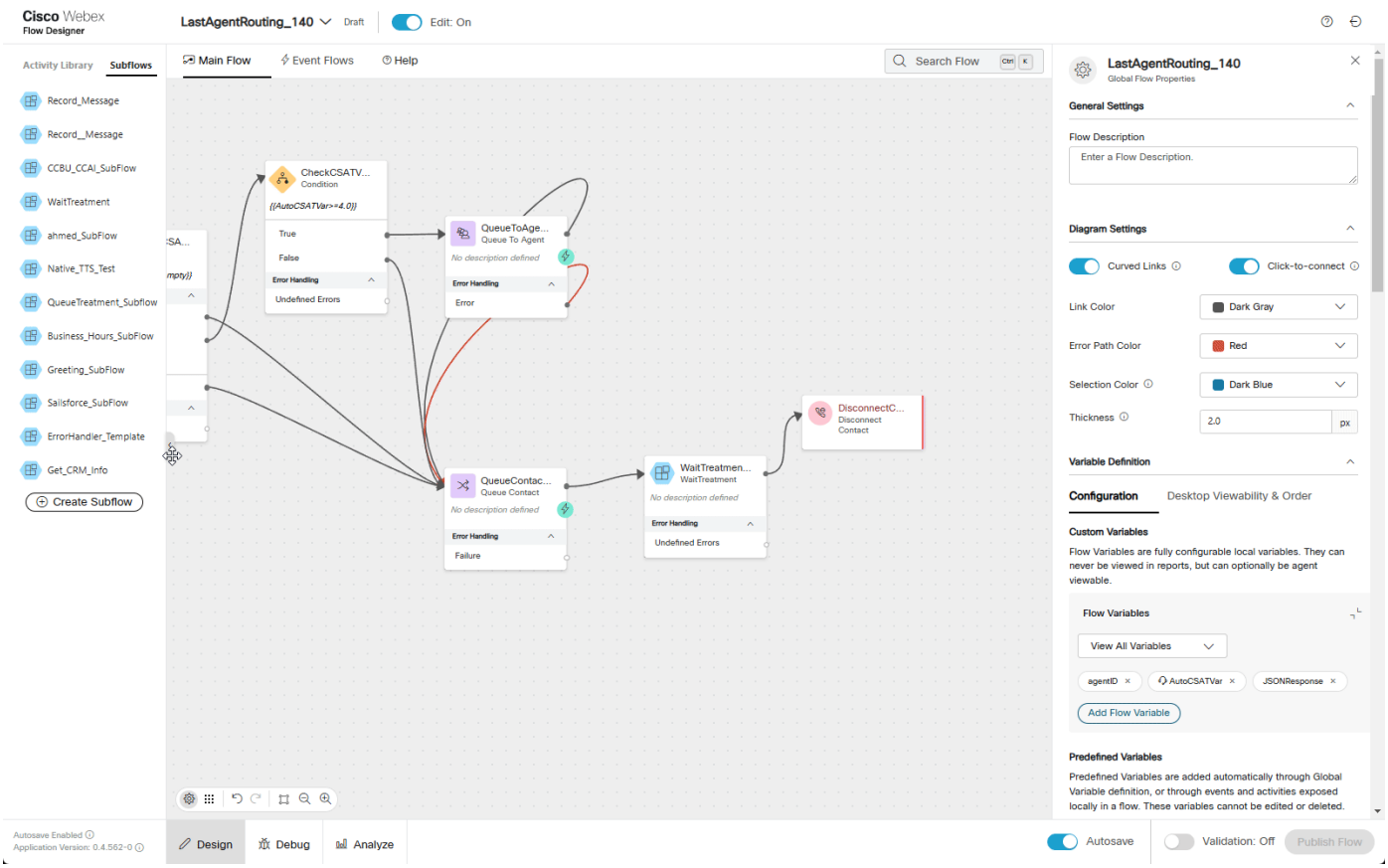
10. Navigate to **Event Flows** and add **GoTo** node to the canvas.

- Connect **AgentDisconnect** event node edge to this **GoTo** node

Click on the **GoTo** node and configure the following settings:

- Destination Type: **Flow**
- Static Flow: **CCBU_PostCallSurvey_AutoCSAT**
- Choose Version Label: **Latest**



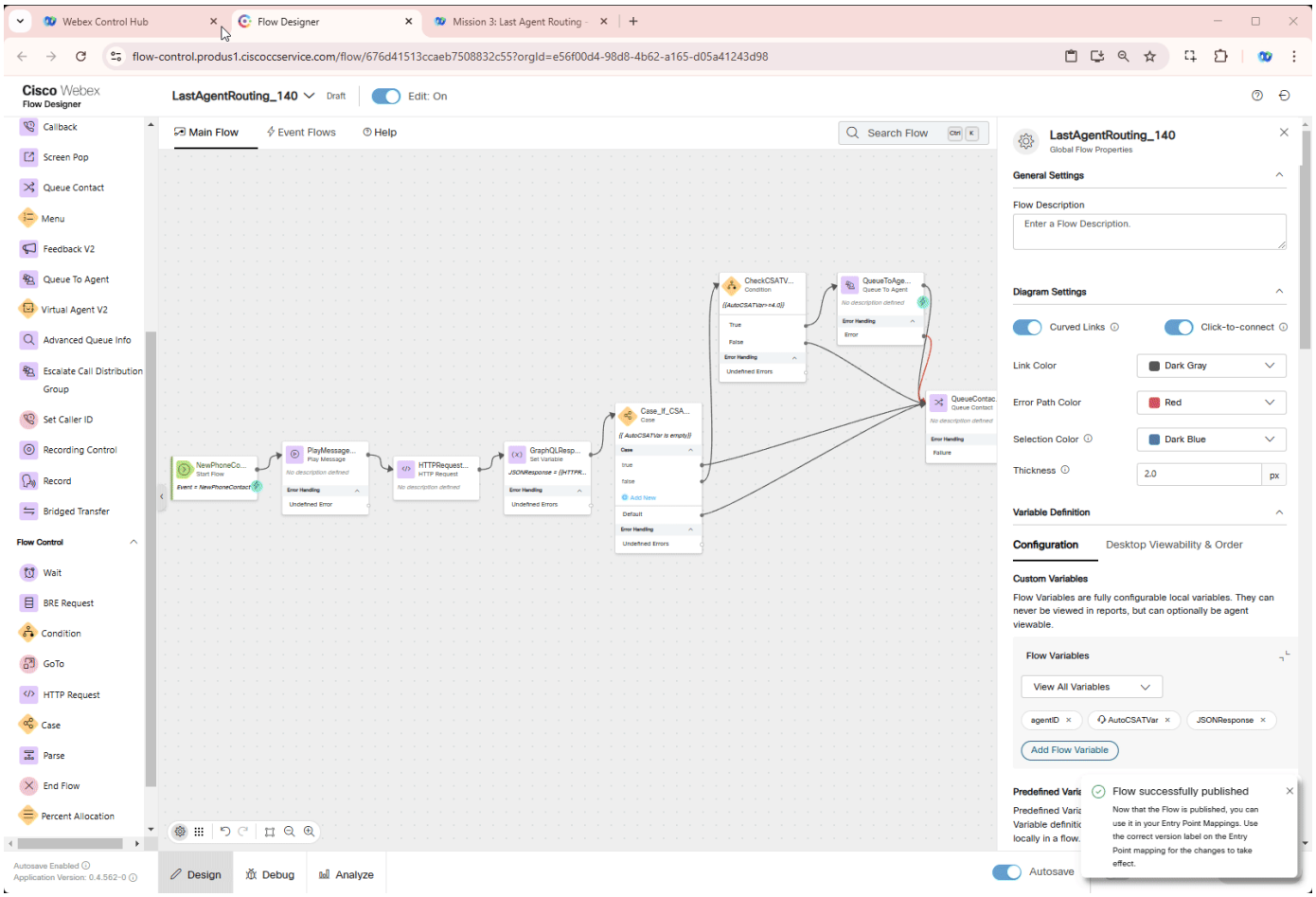


11. Validate and publish the flow:

- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.

12. Map your flow to your inbound channel

- Navigate to Control Hub > Contact Center > Channels
- Locate your Inbound Channel (you can use the search): **Your_Attendee_ID_Channel**
- Select the Routing Flow: **LastAgentRouting_Your_Attendee_ID**
- Select the Version Label: **Latest**
- Click Save in the lower right corner of the screen



Testing

1. Your Agent desktop session should be still active but if not, use Webex CC Desktop application and login with agent credentials you have been provided **wxcclabs+agent_IDYour_Attendee_ID@gmail.com**. You will see another login screen where you may need to enter the email address again and the password provided to you.
2. On your Agent Desktop, set your status to **Available**.

a. Using Webex App, place a call to your Inbound Channel number **Your_Attendee_ID_Channel**

b. You should be offered a call, click on the **Answer** button. (You may want to mute the mic on both Webex App and the Agent Desktop).

c. End the call from Agent Desktop and you should hear an invitation to rate your experience with us on a scale of 1 to 5.

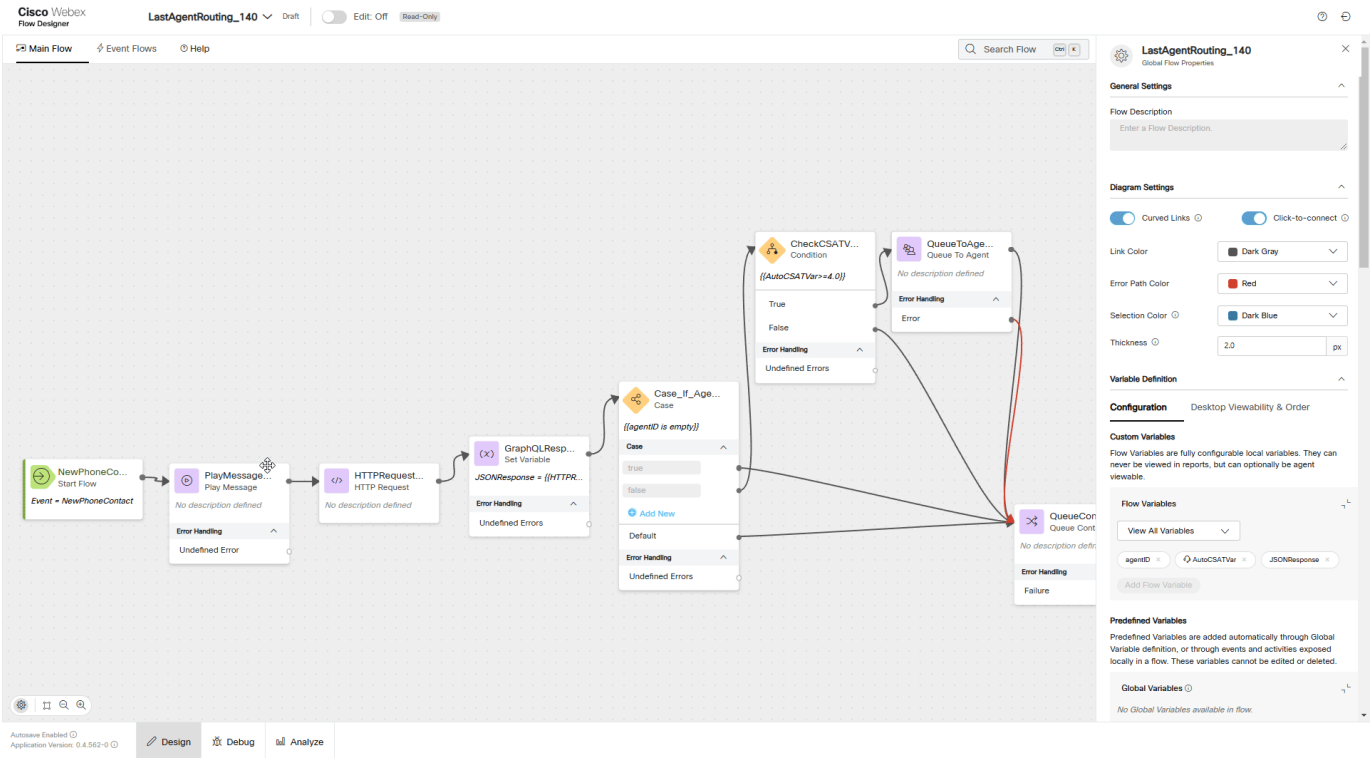
d. Select **5** on Webex App keypad.

3. In your flow, open the flow debugger and select the latest call from the list (on top of the list).

a. Trace the steps taken in the flow.

b. Select **GraphQL_Query** and scroll down the details panel on the right-hand side to **Modified Variables**. They should be empty since there are no CSAT scores at the moment you made the first call.

c. **Case_If_AgentIDEmpty** should exit via **true** node edge as the **GraphQL_Query** had no response, hence the call arrived to your agent via **Your_Attendee_ID_Queue** and not via **QueueToAgent** node.



1. Make sure your agent status is set to **Available**
2. Using Webex App, place another call to your Inbound Channel number **Your_Attendee_ID_Channel**

a. You should be offered the call, click on the **Answer** button.

b. If everything set correctly you should see Auto CSAT set to **5.0**

c. End the call and select any wrap up reason if asked.

3. In your flow, open the flow debugger and select the latest call from the list (on top of the list).

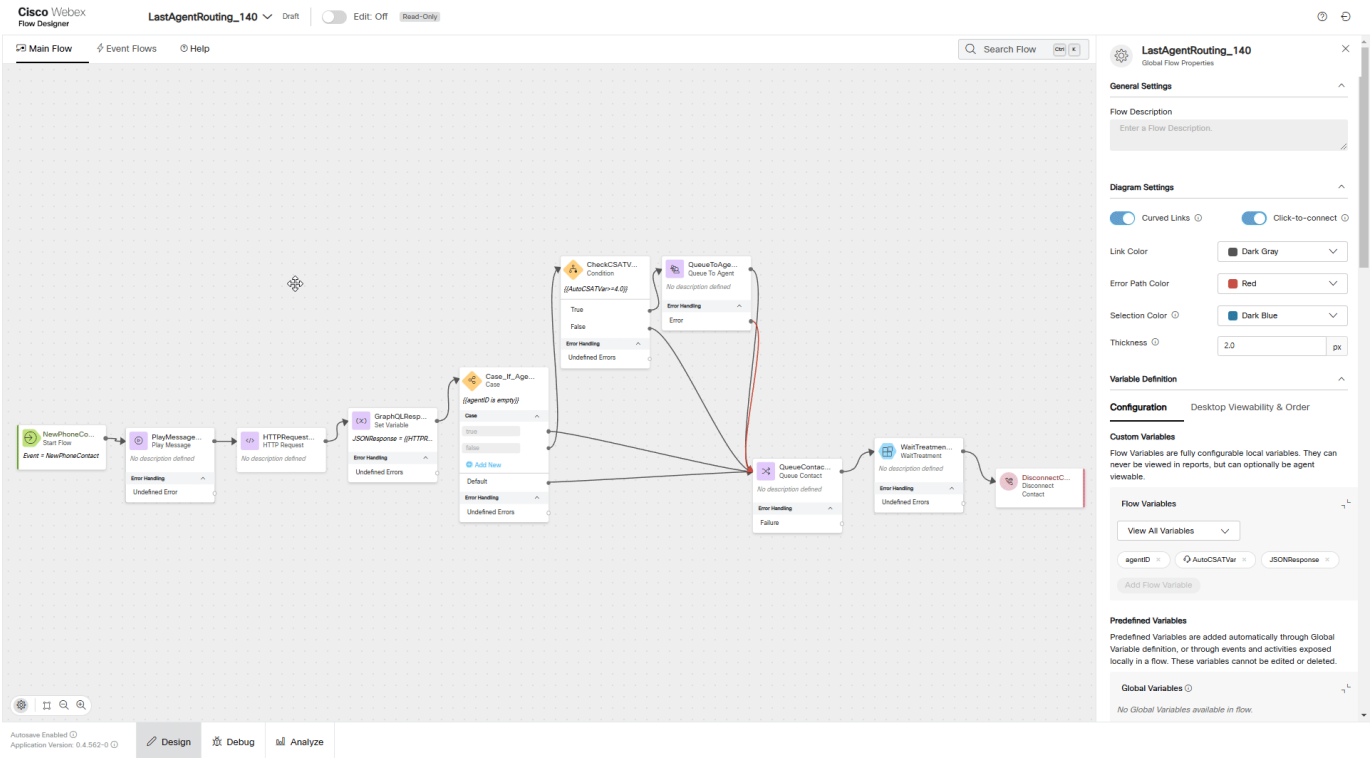
a. Trace the steps taken in the flow

b. Select **GraphQL_Query** and scroll down the details panel on the right-hand side to **Modified Variables**. You should see that now **agentID** and **AutoCSATVar** have assigned values.

c. Select **GraphQLResponse**. In details panel on the right-hand side you should see **Modified Variables** has a JSON response.

d. **Case_If_AgentIDEmpty** should exit via **false** node edge as the **GraphQL_Query** is not empty.

e. **CheckCSATValue** is now either equals **5** which matches the condition hence the call arrived to your agent via **QueueToAgent** node.



Congratulations, you have succesfully completed Last Agent Routing mission! 🎉

3.1.5 Mission 4: Routing Returning Callers

> **Note**
>
> We are intentionally adding a bit of complexity to this lab by removing GIFs and screenshots. This approach will help you gain a deeper understanding of how to build and configure Webex Contact Center logic. If you encounter any difficulties while configuring steps in this mission, feel free to ask one of the instructors for assistance.

Story

When a customer calls back into the contact center within ten minutes of their last call ending, we can assume there was a dropped call, missed callback, or they need additional assistance from their last interaction. We are going to prioritize their call in the queue so that they can finish their business.

Call Flow Overview

1. New call comes into the flow.
2. Call the Search API to check if the ANI (caller's number) had a call which ended in the last 10 minutes.
3. If the caller had a connected call which ended within the last 10 minutes, we will play a message and will queue the call with a higher priority so they will get assigned to the next available agent.
4. If the caller did not end a call with the contact center in the previous 10 minutes, we will queue the call normally.

Mission Details

Your mission is to:

1. Create a new flow from scratch.
2. Build a Search API query to request information from Analyzer database and parse it into flow variables.
3. Build a condition that matches use case scenario and route the call to agent.

> **Note**
>
> We are going to touch Subflow which is the feature that enables easier management of complex flows by breaking down commonly used and repeated portions into reusable subflows. This improves readability of flows, increases reusability of repeated functionality in the subflow, as well as improves development time since there is no redundant design of the same flows.
>
> Subflows also introduce the ability to share commonly used subroutines between developers, between customers and will help unlock a library of subflows available in the marketplace.

PRECONFIGURED ELEMENTS

1. **WaitTreatment** subflow which will provide Music in Queue and Queue Messages.
2. **WxCC_API** connector for calling Webex Contact Center API.

Build

1. Create a fresh flow named **ReturningCaller_Your_Attendee_ID** .

2. Create a flow variable:

- Name: **previousID**
- Type: **String**
- Default Value: leave it empty

3. Add a **Play Message** node for our welcome message:

- Connect the **New Phone Contact** output node edge to this **Play Message** node
- Turn on **Enable Text-To-Speech** toggle
- Select the Connector: **Cisco Cloud Text-to-Speech**
- Click the **Add Text-to-Speech Message** button
- Delete the selection for Audio File
- Text-to-Speech Message: **Welcome to the advanced routing and API integrations lab.**

4. Add an **HTTP Request** node for our query:

- Connect the output node edge from the **Play message** node to this node
- Select **Use Authenticated Endpoint**
- Connector: **WxCC_API**
- Path: **/search**
- Method: **POST**
- Content Type: **GraphQL**
- Copy this GraphQL query and paste it into the **Query** filed of the **Request Body**:

```
query returnAgent(
  $from: Long!
  $to: Long!
  $filter: TaskDetailsFilters
) {
  taskDetails(from: $from, to: $to, filter: $filter) {
    tasks {
      id
      status
      channelType
      createdTime
      endedTime
      origin
      destination
      direction
      terminationType
      isActive
      isCallback
      lastWrapupCodeName
    }
  }
}
```

- Copy the following variables into the **GraphQL variables**:

```
{
  "from": "{{now() | epoch(inMillis=true) - 900000}}",
  "to": "{{now() | epoch(inMillis=true)}}",
  "filter": {
    "and": [
    {
        "ivrScriptId": {
            "equals": "{{NewPhoneContact.FlowId}}"
            }
    },
    {
       "status": {
        "equals": "ended"
        }
    },
    {
        "origin": {
          "equals": "{{NewPhoneContact.ANI}}"
          }
    },
    {
        "connectedCount": {
            "gte": 1
            }
    }
    ]
  }
}
```

Parse Settings:

- Content Type: `JSON`
- Output Variable: **previousID**
- Path Expression: `$.data.taskDetails.tasks[0].id`

5. Add a **Condition** node:

- Connect the output from the **HTTP Request** node to this node
- Expression: `{{previousID is empty}}`
- We will connect the **True** node in a future step.
- Connect the **False** node edge to the **Play Message** node created in the next step.

6. Add a **Play Message** node:

- Connect the **False** node edge from the **Condition** node created at previous step to this node
- Turn on **Enable Text-To-Speech** toggle
- Select the Connector: **Cisco Cloud Text-to-Speech**
- Click the **Add Text-to-Speech Message** button
- Delete the selection for Audio File
- Text-to-Speech Message: **It looks like you were just working with an agent and had to call back in. We are prioritizing this call for the next available agent.**

7. Add a **Queue Contact** node:

- Connect the output node edge from the **Play Message** node added in the last step to this node
- Select **Static Queue**
- Queue: **Your_Attendee_ID_Queue**
- Turn on **Set Contact Priority** toggle
- Select **Static Priority**
- Static Priority Value: **P1**

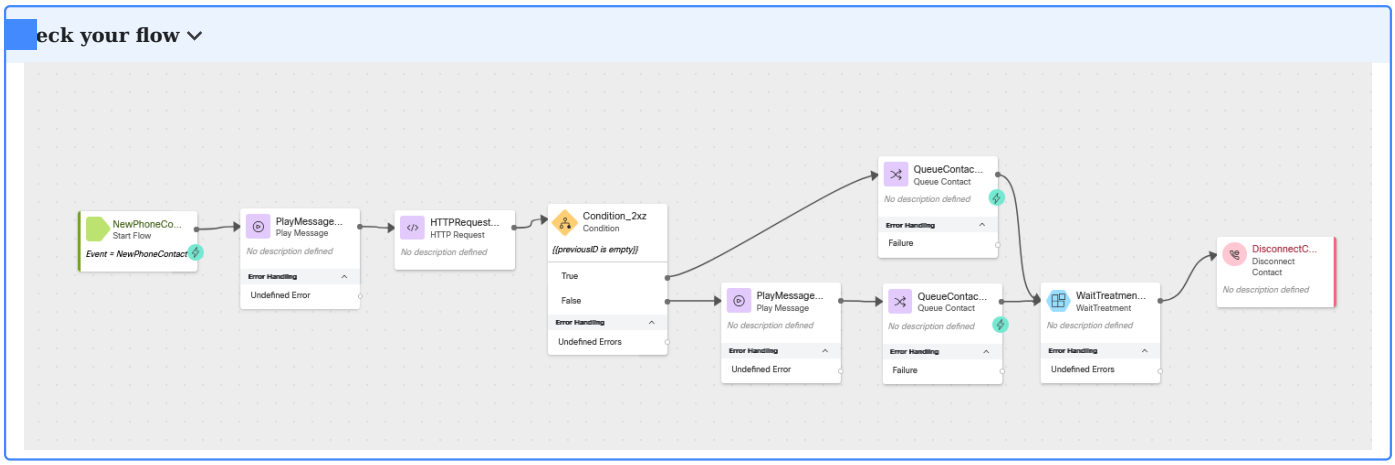
8. Add a **Subflow** node:

- In the Activity Library pane on the left side of the screen, click **Subflows**
- Find the subflow **WaitTreatment** and drag it onto the flow canvas like you would any other node.
- Connect the output node edge from the **Queue Contact** node added in the previous step to this node.

  Click on the **WaitTreatment** subflow node and configure the following settings:
- Subflow Label: **Latest**
- Make sure **Enable automatic updates** is turned on

9. Add a **Disconnect Contact** node and connect the output node edge from **WaitTreatment** subflow node added at the previous step to this **Disconnect Contact** node.
10. Add one more **Queue Contact** node:

- Connect the **True** node edge from the **Condition** node to this node
- Select **Static Queue**
- Queue: **Your_Attendee_ID_Queue**
- Connect the **Output** node edge from this node to the **WaitTreatment** subflow node



11. Validate and publish the flow:
    - Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
    - If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
    - In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.
12. Switch to Control Hub and navigate to **Channels** under Customer Experience Section
    - Locate your Inbound Channel (you can use the search): **Your_Attendee_ID_Channel**
    - Select the Routing Flow: **ReturningCaller_Your_Attendee_ID**
    - Select the Version Label: **Latest**
    - Click Save in the lower right corner of the screen

Testing

1. Your Agent desktop session should be still active but if not, use Webex CC Desktop application and login with agent credentials you have been provided **wxcclabs+agent_IDYour_Attendee_ID@gmail.com** You will see another login screen where you may need to enter the email address again and the password provided to you.
2. On your Agent Desktop, make sure your status is set to **Available**.
3. Using Webex, place another call to your Inbound Channel number **Your_Attendee_ID_Channel**
4. On your Agent Desktop, you should be offered a call, click on the **Answer** button. (You may want to mute the mic on both Webex and the Agent Desktop). After a few moments end the call and select any wrap up reason.
5. In your Flow:

a. Open Flow Debugger.

b. Select the last interaction (at the top of the list).

c. Trace the steps taken in the flow.

6. Close Flow Debugger.
7. Using Webex, place another call to your Inbound Channel number **Your_Attendee_ID_Channel**
8. On your Agent Desktop, make sure your status is **Available**.

a. You should hear a message indicating that you recently placed a call and that your new call will be prioritized. (You may want to mute the mic on both Webex and the Agent Desktop).

b. On your Agent Desktop, you should be offered a call, click on the **Answer** button.

c. After a few moments end the call and select any wrap up reason.

9. In your Flow:
   a. Open Flow debugger.
   b. Select the last interaction (at the top of the list).
   c. Trace the steps taken in the flow.
10. Answer these questions:
    a. Did the call choose another path?
    b. Why or why not?

Congratulations, you have succesfully completed Routing Returning Callers mission! 🎉

3.1.6 Mission 5: Personalized Customer Notification with Functions

Story

In this mission, you will use a **Function** node in Webex Contact Center to retrieve customer data from an external database and generate a personalized message using JavaScript. This ensures callers receive real-time updates on their last purchase, outstanding balance, and service requests without agent intervention.

Call Flow Overview

1. When call arrives fetch the data from **MockAPI** based on your Dialed Number.
2. Write the data into respective preconfigured flow variables. These variables will be passed to a **Function**.
3. **Function** will generate a message based on inputs and pass it back to your flow.
4. **Play Message** node plays the message received from **Function**.

Mission Details

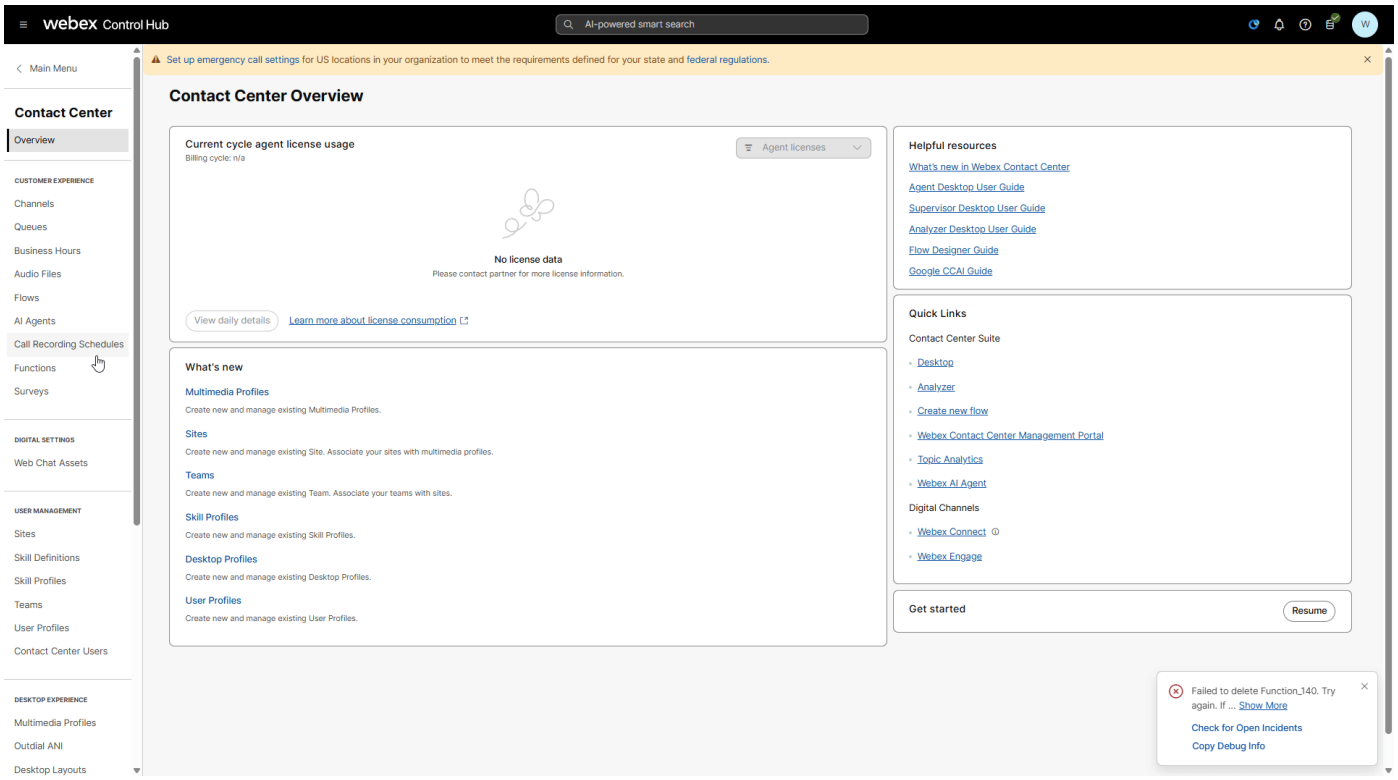
Your mission is to:

1. Create a function that retrieves input variables, processes them, and generates a customer-friendly message.
2. Create a new flow with the **Function** element.
3. Request data from an external database, parse it into flow variables, and pass them to your **Function**.
4. Play the message generated by the **Function** to the caller.

> **od to Know [Optional]**
>
> We are going to imitate a real API server by providing realistic responses to requests. For that we chose Server **MockAPI**.
>
> For more information of how you can use MockAPI please watch these Vidcasts:
>
> - **[ADVANCED] Use MockAPI to enhance your Demos - PART 1**
> - **[ADVANCED] Use MockAPI to enhance your Demos - PART 2**

Building Function

1. Switch to Control Hub, then navigate to **Functions**, click on **Create a function**.
2. New Tab will be opened. Select **Start Fresh** and provide name **Function_Your_Attendee_ID** . Leave **Function Language** as **JavaScript**, then click **Create Function**.



3. Add following variables by clicking **Add Input Variables** on the right-hand side:

- Last purchase date variable:

  Name: **lastPurchase**

  Type: **String**

- Customer pending request variable:

  Name: **pendingServiceRequest**

  Type: **String**

- Expected resolution date variable:

  Name: **resolutionDate**

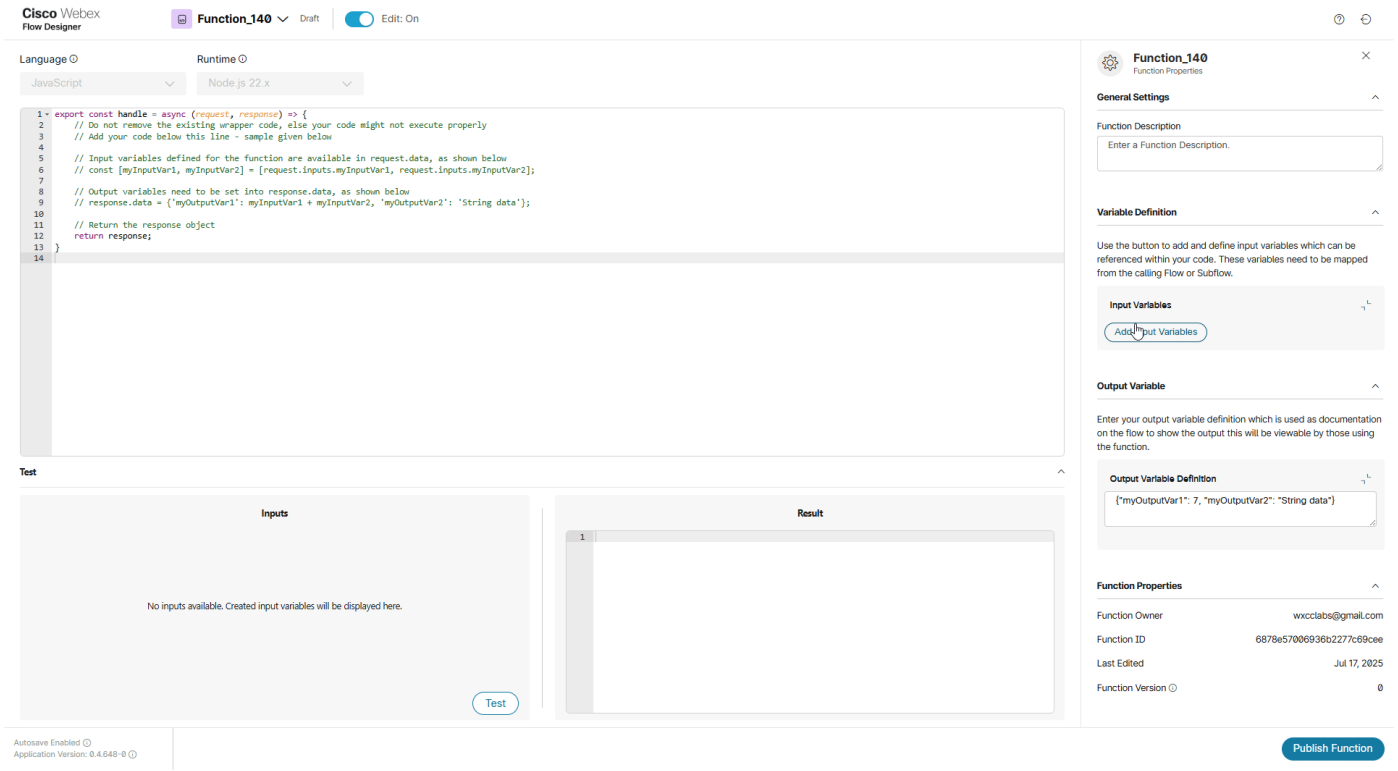
  Type: **String**

- Outstanding account balance variable:

  Name: **outstandingBalance**

  Type: **String**

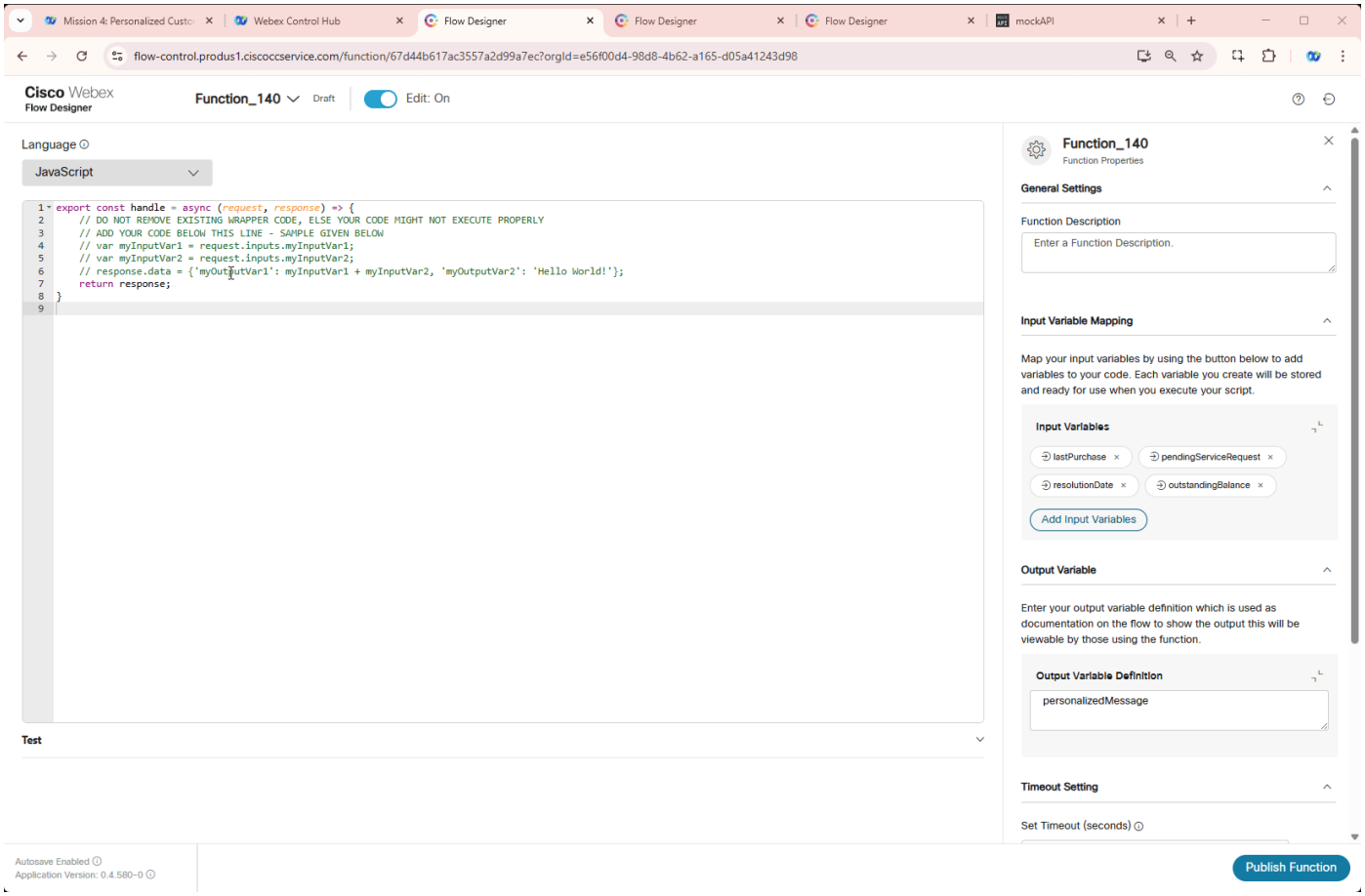
4. Add output variable **personalizedMessage** in **Output Variable Definition** field.



5. Clear the editor from the default code and paste the following JavaScript code into Function editor.

```
export const handle = async (request, response) => {
   // Retrieve input variables (all as strings)
   var lastPurchase = request.inputs.lastPurchase;  // e.g., '17-03-2025'
   var outstandingBalance = request.inputs.outstandingBalance;  // e.g., '120.50'
   var pendingServiceRequest = request.inputs.pendingServiceRequest;  // e.g., 'Network issue'
   var resolutionDate = request.inputs.resolutionDate;  // e.g., '20-03-2025'

   // Convert outstandingBalance to a number safely
   let balanceAmount = parseFloat(outstandingBalance);
   if (isNaN(balanceAmount)) {
       balanceAmount = 0;  // Default to 0 if conversion fails
   }

   // Function to convert 'DD-MM-YYYY' to a proper Date object
   function parseDate(dateString) {
       if (!dateString) return null;
       const parts = dateString.split('-'); // Split "17-03-2025" into ["17", "03", "2025"]
       if (parts.length !== 3) return null; // Invalid format

       const [day, month, year] = parts.map(num => parseInt(num, 10));
       return new Date(year, month - 1, day); // Month is 0-based in JS Dates
   }

   // Format the date to a readable string (e.g., "March 17, 2025")
   function formatDate(date) {
       return date ? date.toLocaleDateString('en-US', { year: 'numeric', month: 'long', day: 'numeric' }) : null;
   }

   // Parse and format dates
   let formattedLastPurchase = formatDate(parseDate(lastPurchase));
   let formattedResolutionDate = formatDate(parseDate(resolutionDate));

   // Generate the personalized message
   let message = 'Hello, we have some information about your account:';

   if (formattedLastPurchase) {
       message += ` Your last purchase was on ${formattedLastPurchase}.`;
   }
   if (balanceAmount > 0) {
       message += ` You have an outstanding balance of $${balanceAmount.toFixed(2)}.`;
   }
   if (pendingServiceRequest) {
       message += ` We also have a pending service request: ${pendingServiceRequest}`;
       if (formattedResolutionDate) {
           message += `, resolution by ${formattedResolutionDate}.`;
       } else {
           message += `.`;
       }
   } else {
       message += ' There are no pending service requests at the moment.';
   }
```

```
    // Return the generated message
    response.data = { 'personalizedMessage': message };

    return response;
    };
```



od to know: Script Breakdown [Optional]

This script dynamically generates a personalized message for customers based on their account details. In our lab it is a Channel Support Number assigned to you. It is used within Webex Contact Center Flow Designer to enhance customer interactions.

a. How It Works:
Retrieves input variables from the Flow:

- `lastPurchase` → Date of the last purchase (e.g., `"17-03-2025"`).
- `outstandingBalance` → The amount owed by the customer (e.g., `"120.50"`).
- `pendingServiceRequest` → Details of any unresolved service request (e.g., `"Network issue"`).
- `resolutionDate` → Expected resolution date of the service request (e.g., `"20-03-2025"`).

b. Processes and formats the data:

- Converts `lastPurchase` and `resolutionDate` from `"DD-MM-YYYY"` format into a more readable format (`March 17, 2025`).
- Ensures `outstandingBalance` is a valid number and avoids errors if it's missing or invalid.

c. Generates a customer-friendly message:

- If the customer made a purchase, the date is included.
- If an outstanding balance exists, the amount is shown.
- If there's a pending service request, it's mentioned, along with a resolution date (if available).
- If no pending issues exist, it confirms that everything is fine.

d. Returns the message to be used in the Flow:

- The script outputs `personalizedMessage`, which can be read out in an IVR or displayed to an agent.

- Perform a test to see how you script preforms by navigating to **Test** panel below the editor. Provide the following data into respective fields, then click **Test** button.

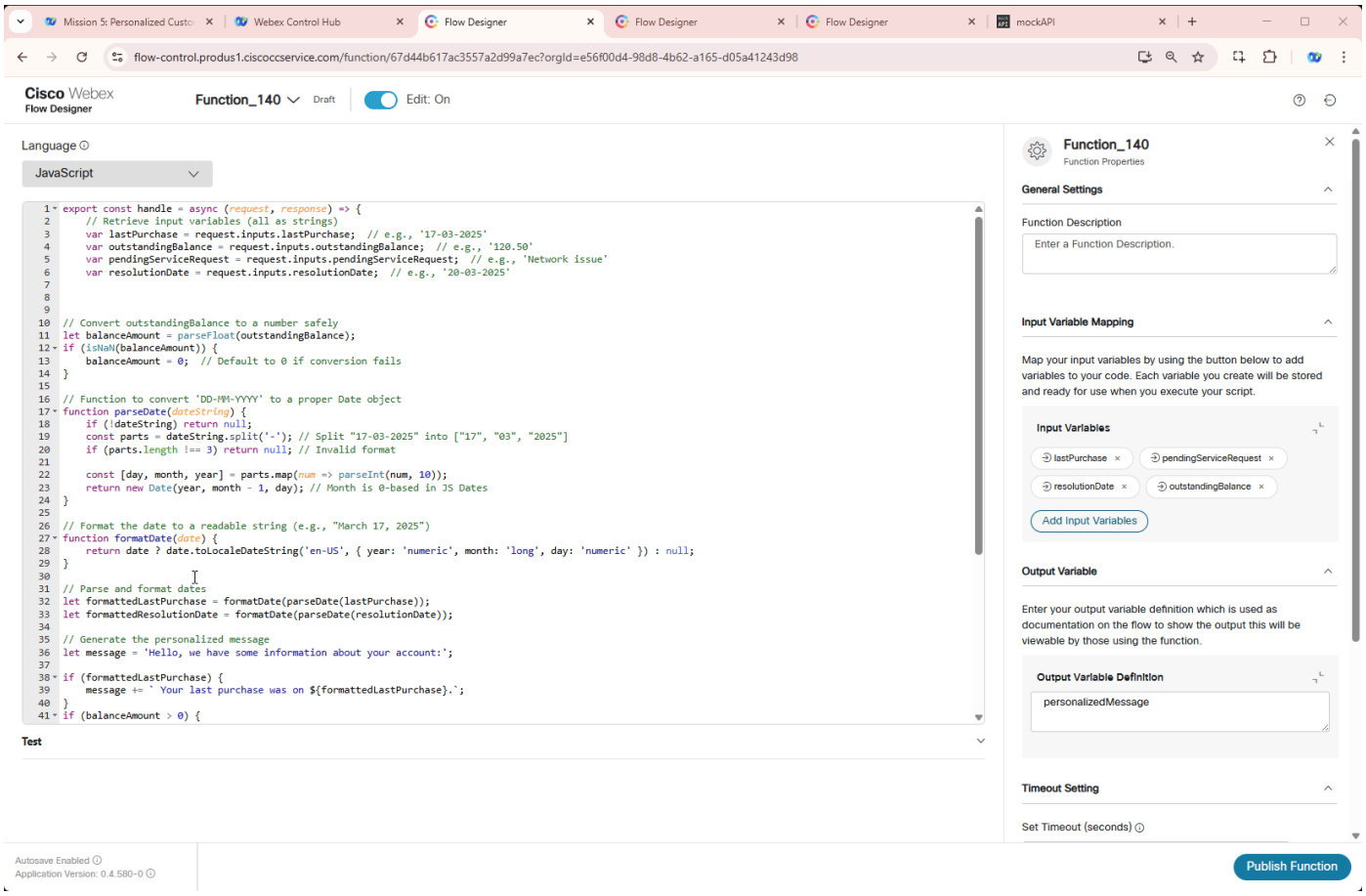
`lastPurchase` : **17-03-2025**

`pendingServiceRequest` : **Network issue on Amsterdam location**

`resolutionDate` : **20-03-2025**

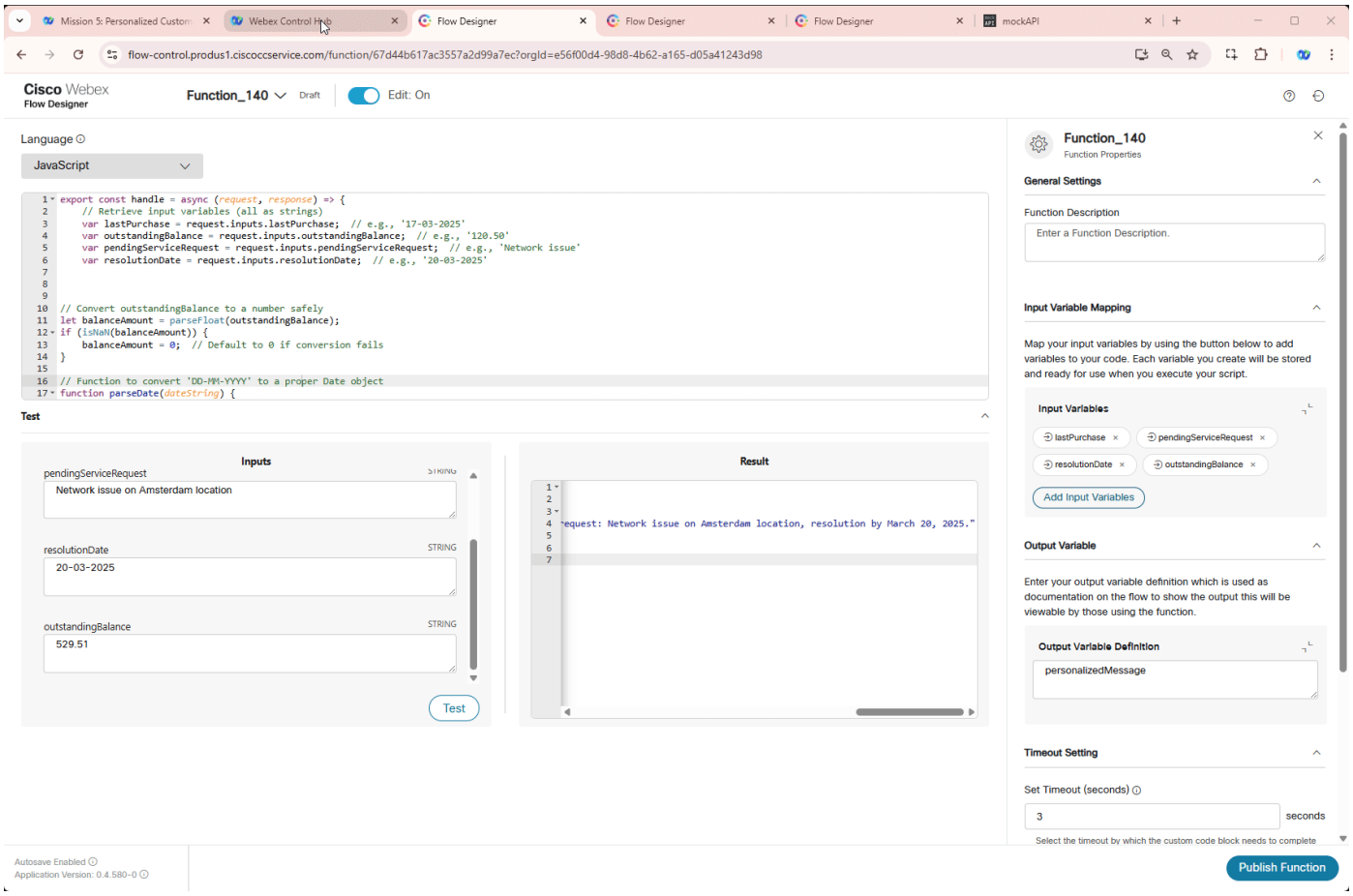
`outstandingBalance` : **529.51**

- Click on **Publish Function** in the bottom right corner of the page. Then click **Publish Function** in pop up window.



Building Flow

1. Switch to Control Hub, then navigate to **Flows**, click on **Manage Flows** then select **Create Flows** from drop down list.
2. Select **Start Fresh** then enter flow name **FunctionFlow_Your_Attendee_ID** and click on **Create Flow**.



3. Add these flow variables:

- Last purchase date variable:

  Name: **lastPurchase**

  Type: **String**

  Default Value: leave it empty

- Customer pending request variable:

  Name: **pendingServiceRequest**

  Type: **String**

  Default Value: leave it empty

- Expected resolution date variable:

  Name: **resolutionDate**

  Type: **String**

  Default Value: leave it empty

- Outstanding account balance variable:

  Name: **outstandingBalance**

  Type: **String**

  Default Value: leave it empty

- HTTP Response variable:

  Name: **HTTPResponse**

  Type: **String**

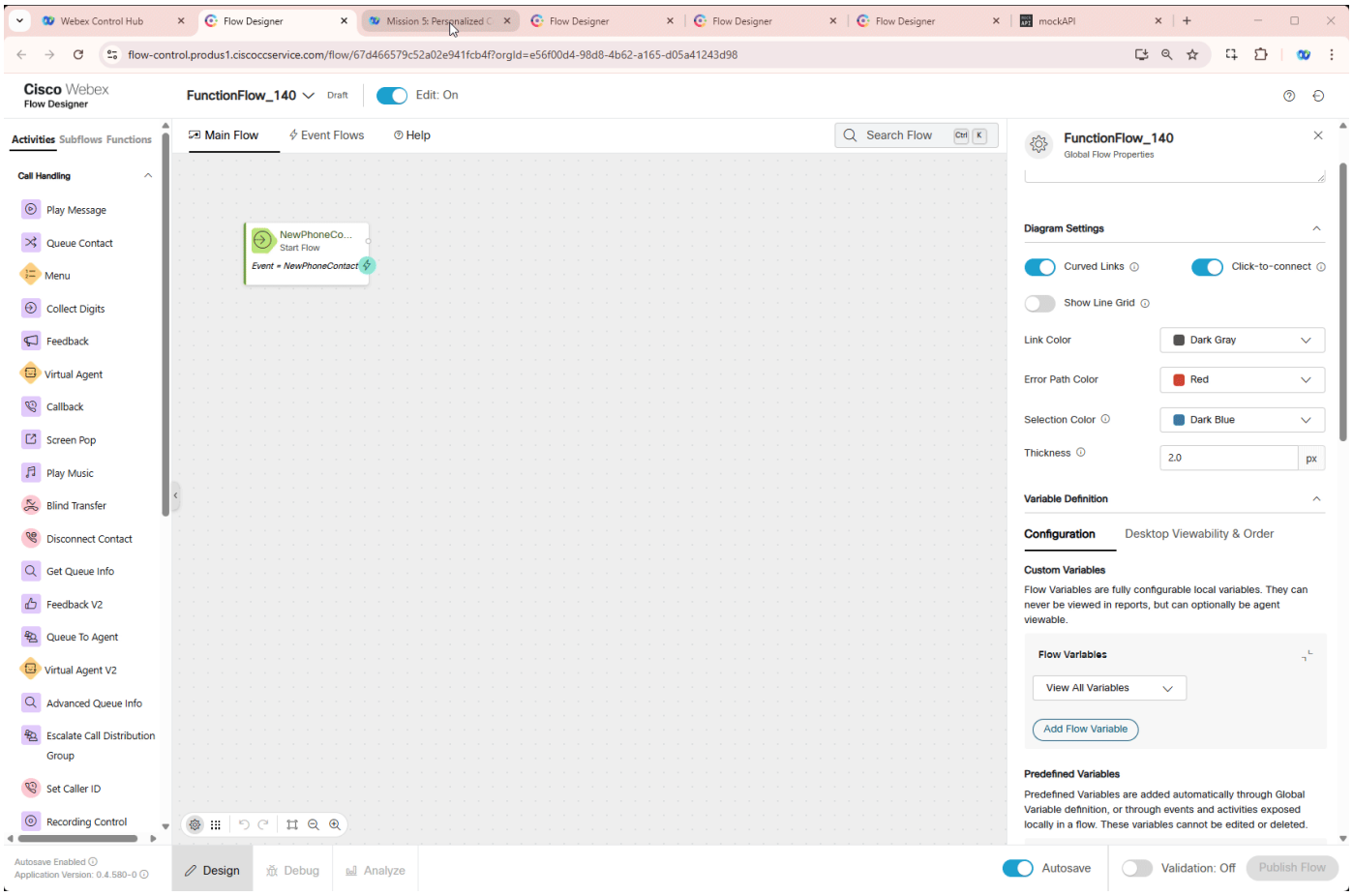
  Default Value: leave it empty

- Personalized message variable:

  Name: **personalizedMessage**

  Type: **String**

  Default Value: leave it empty

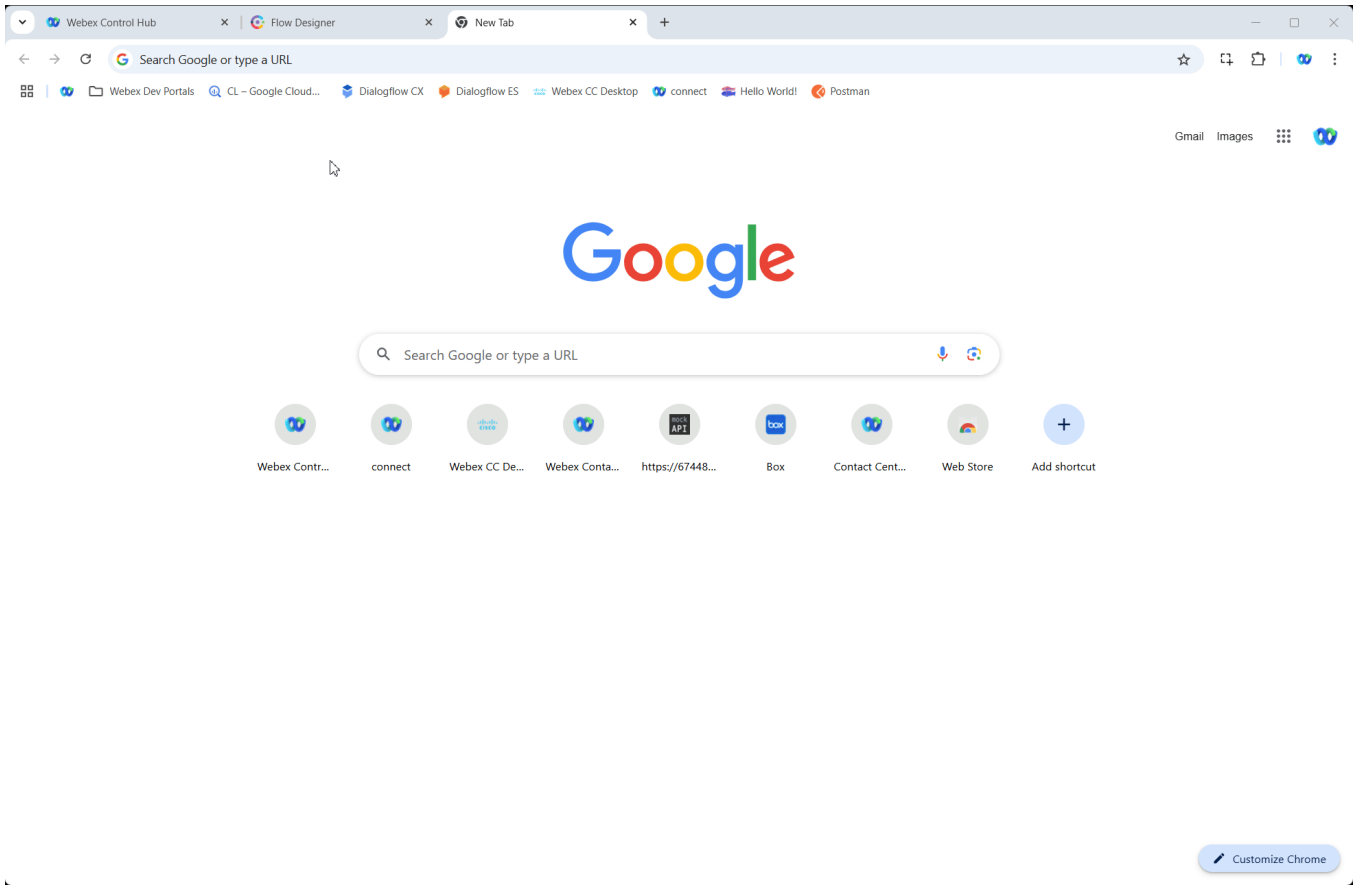


st your API Source [Optional]

a. Test your API resource. **https://674481b1b4e2e04abea27c6e.mockapi.io/flowdesigner/Lab/DynVars?dn={DNIS}**

b. Replace DNIS with the provided DNIS number stripping **+1**. [For example:] If your number **+14694096861**, then your GET Query should be ***https://674481b1b4e2e04abea27c6e.mockapi.io/flowdesigner/Lab/DynVars?dn=4694096861***

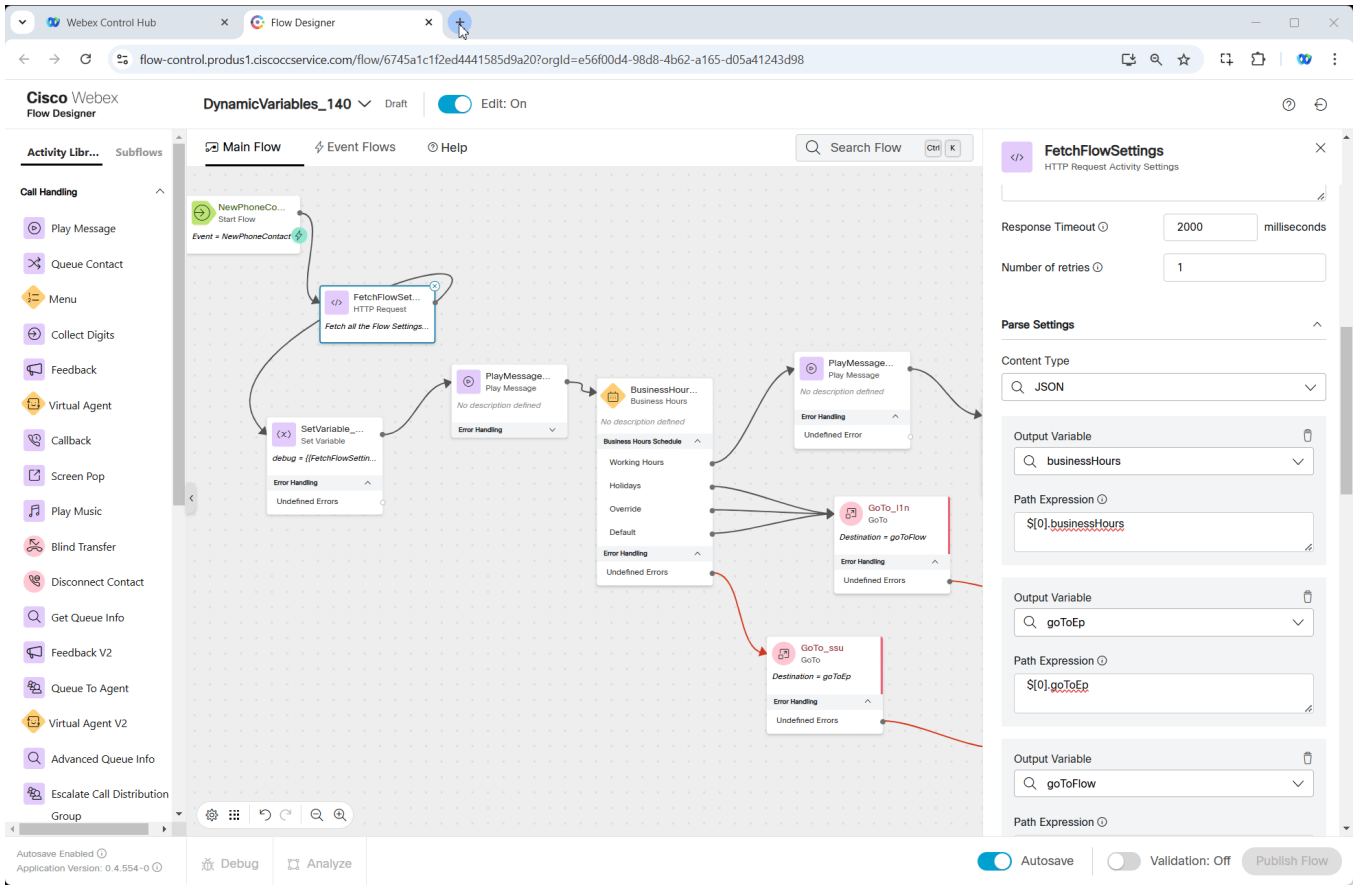
c. Open Chrome browser, paste your Get query URL into the Browser address line and press Enter. You should get the JSON response like this:



d. Open the new browser tab and navigate to JSONPath Online Evaluator

e. Copy the JSON response (including square brackets) you obtained above and paste it into **Document** window of **JSONPath Online Evaluator**.

f. In **JSONPath** box copy and paste one of the path expression from **FetchFlowSettings** to verify your results. For example, **$[0].businessHours**



4. Add an **HTTP Request** node. We are going to fetch data and write it into our new created variables `lastPurchase`, `pendingServiceRequest`, `resolutionDate` and `outstandingBalance`.

   Connect **NewPhoneContact** to this HTTP node

   We will connect **HTTP Request** node in next step

   Activity Name: **GET_HTTPRequest**

   Use Authenticated Endpoint: **Off**

   Request URL: ***https://674481b1b4e2e04abea27c6e.mockapi.io/flowdesigner/Lab/DynVars?dn={{NewPhoneContact.DNIS | slice(2) }}***

   Method: **GET**

   Content Type: **Application/JSON**

   Parsing Settings:

   Content Type: **JSON**

   Output Variable: `pendingServiceRequest`

   Path Expression: `$[0].pendingServiceRequest`

   Click **+ Add New**

   Output Variable: `lastPurchase`

   Path Expression: `$[0].lastPurchase`

   Click **+ Add New**

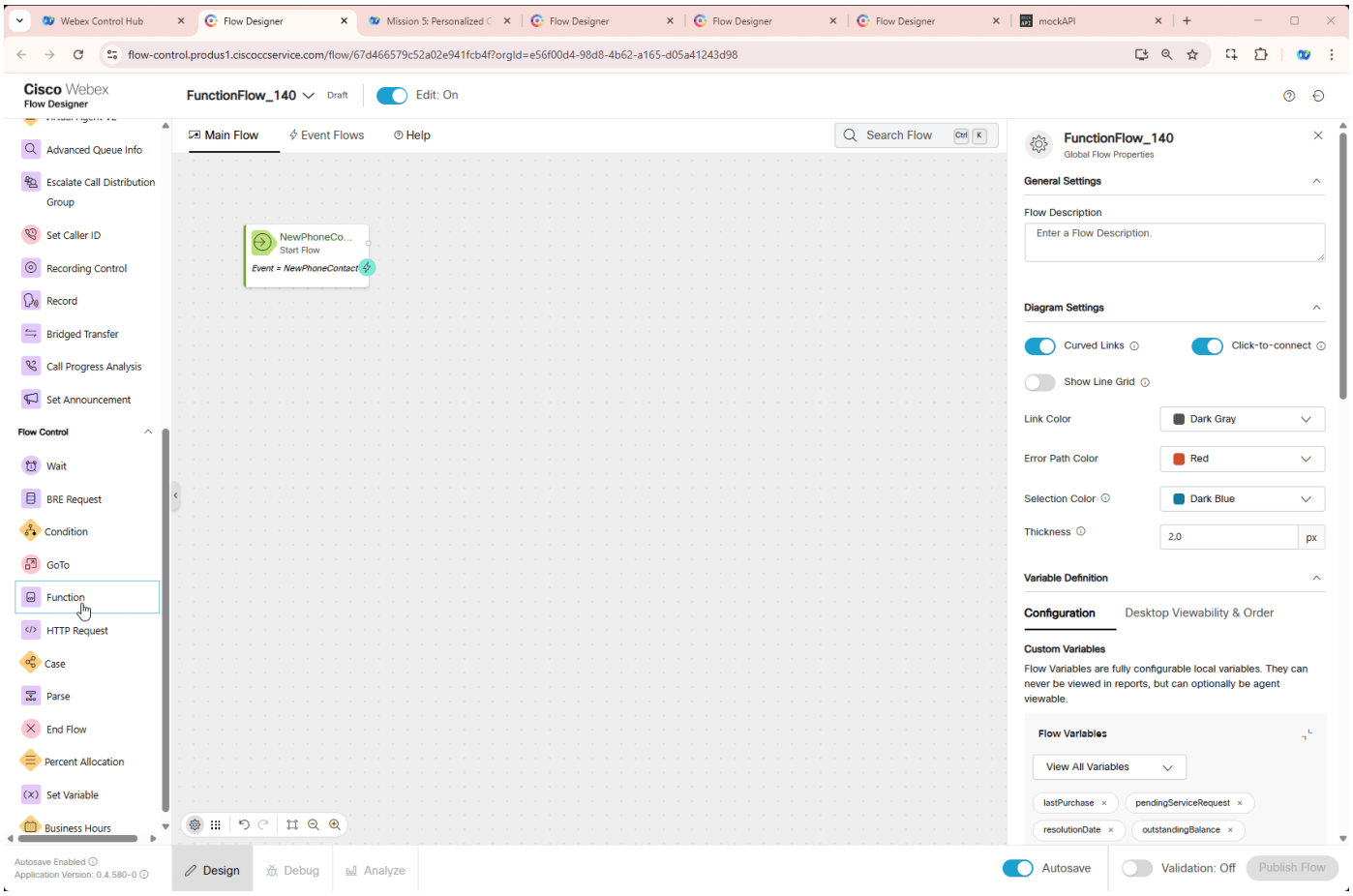
   Output Variable: `outstandingBalance`

   Path Expression: `$[0].outstandingBalance`

   Click **+ Add New**

   Output Variable: `resolutionDate`

   Path Expression: `$[0].resolutionDate`



5. Add **Set Variable** node

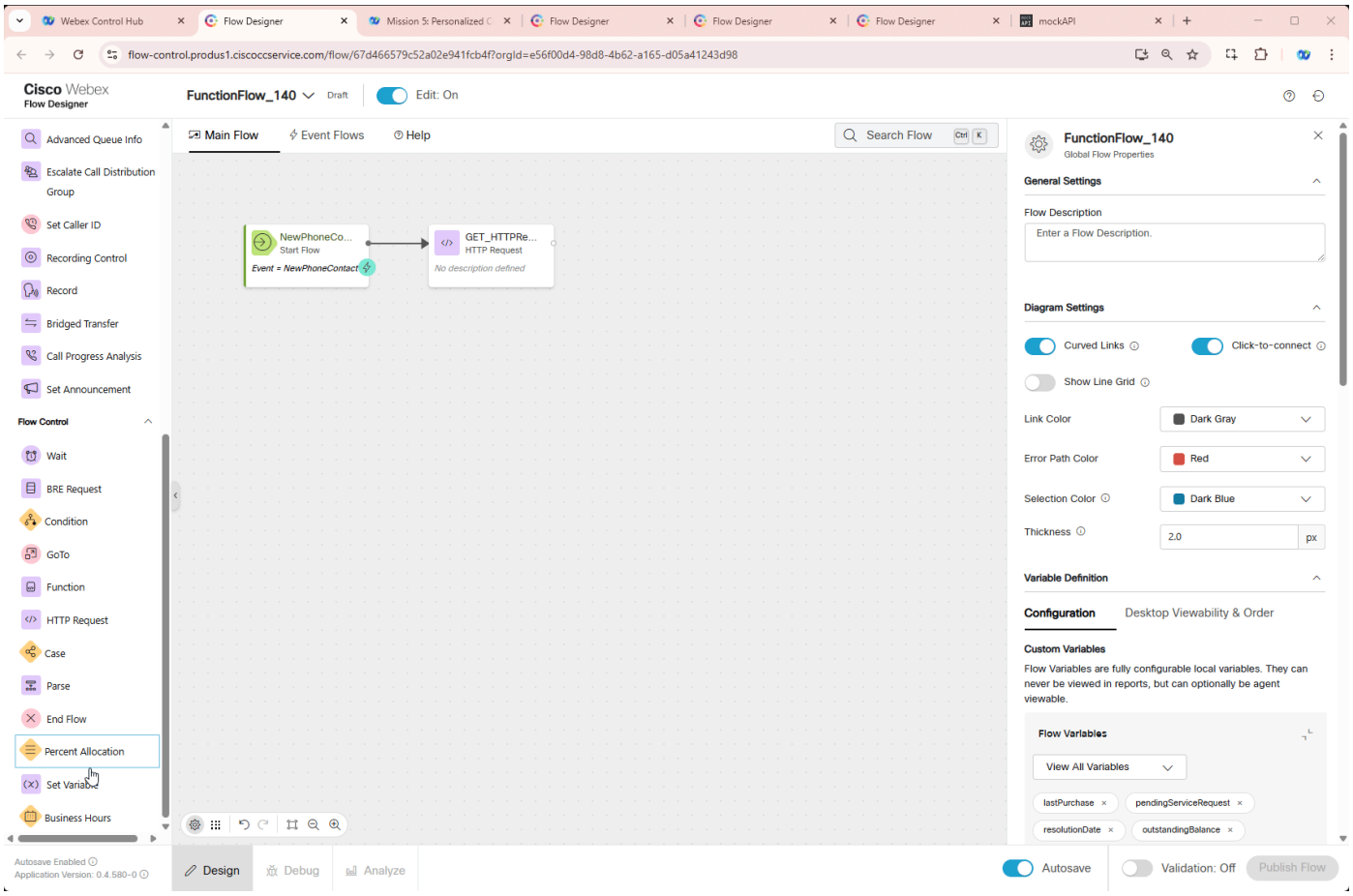
   Activity Label: **HTTP_Response**

   Connect **GET_HTTPRequest** you added at the previous step to this node

   We will connect this **Set Variable** node in next step

   Variable: `HTTPResponse`

   Set Variable: `GET_HTTPRequest.httpResponseBody`



6. Switch to **Functions** tab at the panel on the left-hand side. Then drag **Function_Your_Attendee_ID** node to the canvas.

   Connect **HTTP_Response** node you added at the previous step to this node

   We will connect this node in next step

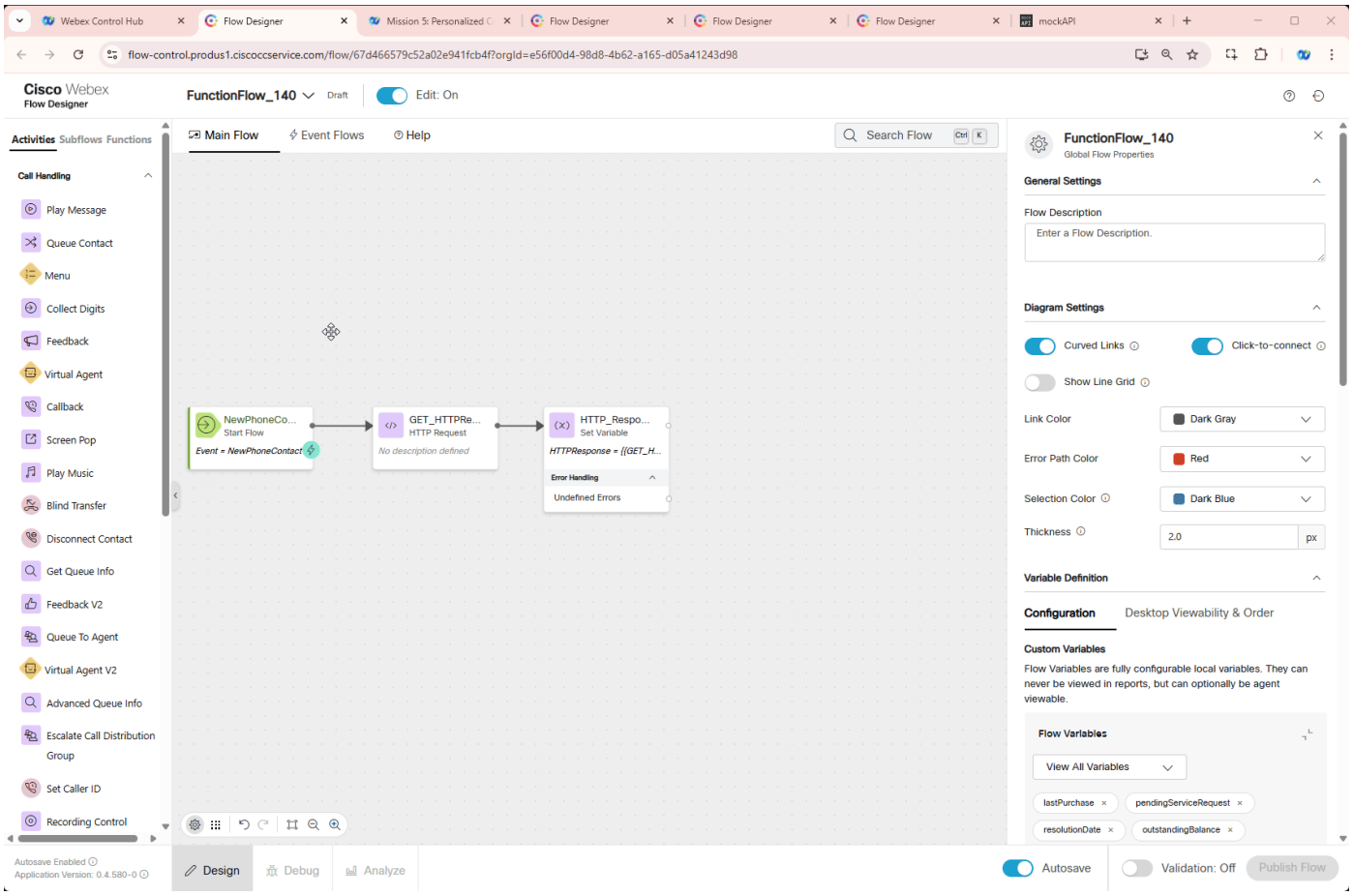
   Function Version Label: **Latest**

   Function Input Variables should auto populate

   Output Settings

   Output Variable: `personalizedMessage`

   Path Expression `$.personalizedMessage`



7. Switch to **Activity** tab in the left menu. Add **Play Message** and **Disconnect Contact** nodes

   Connect the **Function** node edge to the **Play Message** node

   Enable Text-To-Speech

   Select the Connector: **Cisco Cloud Text-to-Speech**

   Click **Add Text-to-Speech Message** button

   Delete the selection for Audio File

   Text-to-Speech Message: **{{personalizedMessage}}**

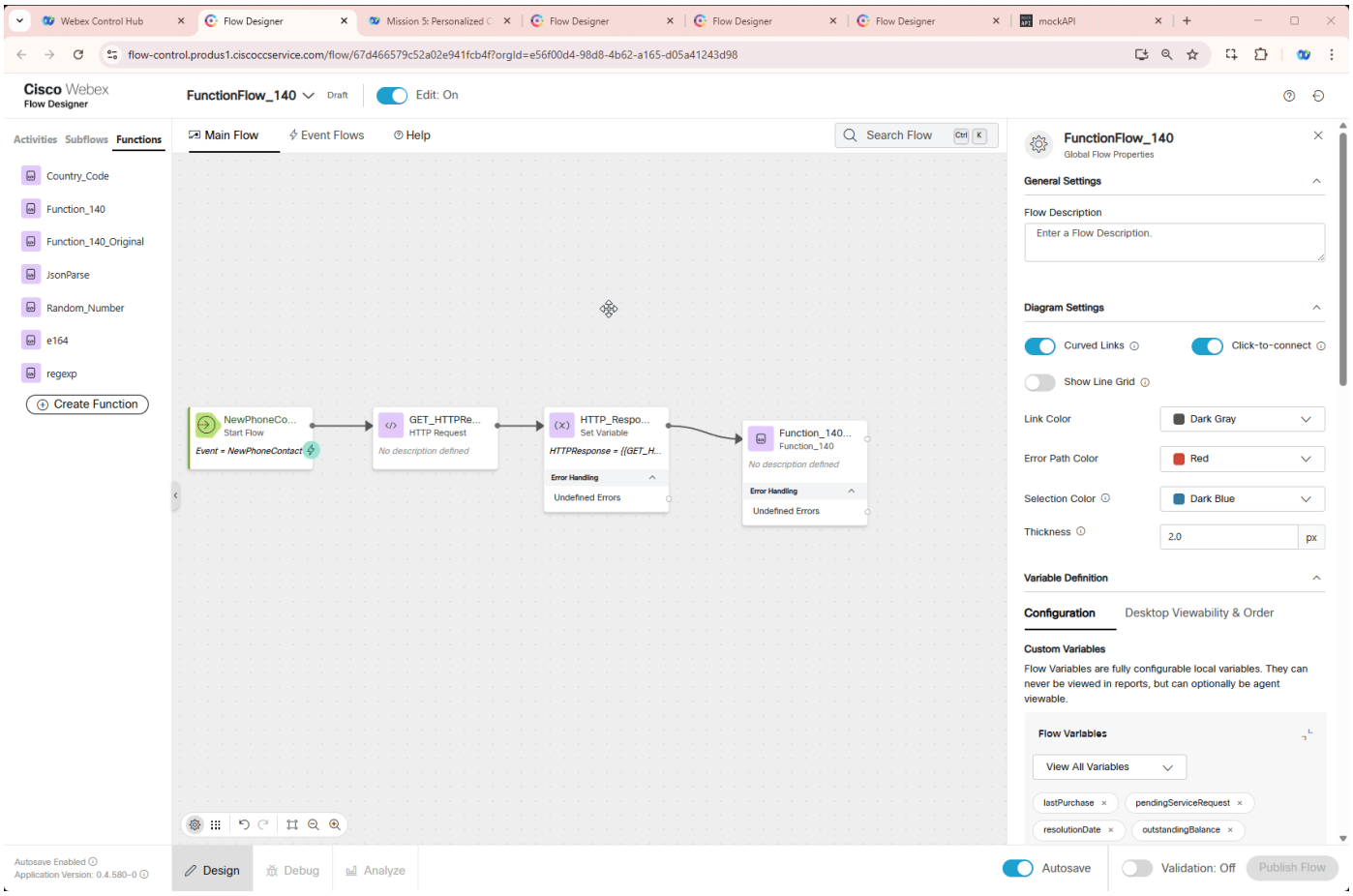
   Connect the **Play Message** node edge to this **Disconnect Contact** node

8. Validate and publish the flow:

   Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.

   If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.

   In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.



9. Map your flow to your inbound channel

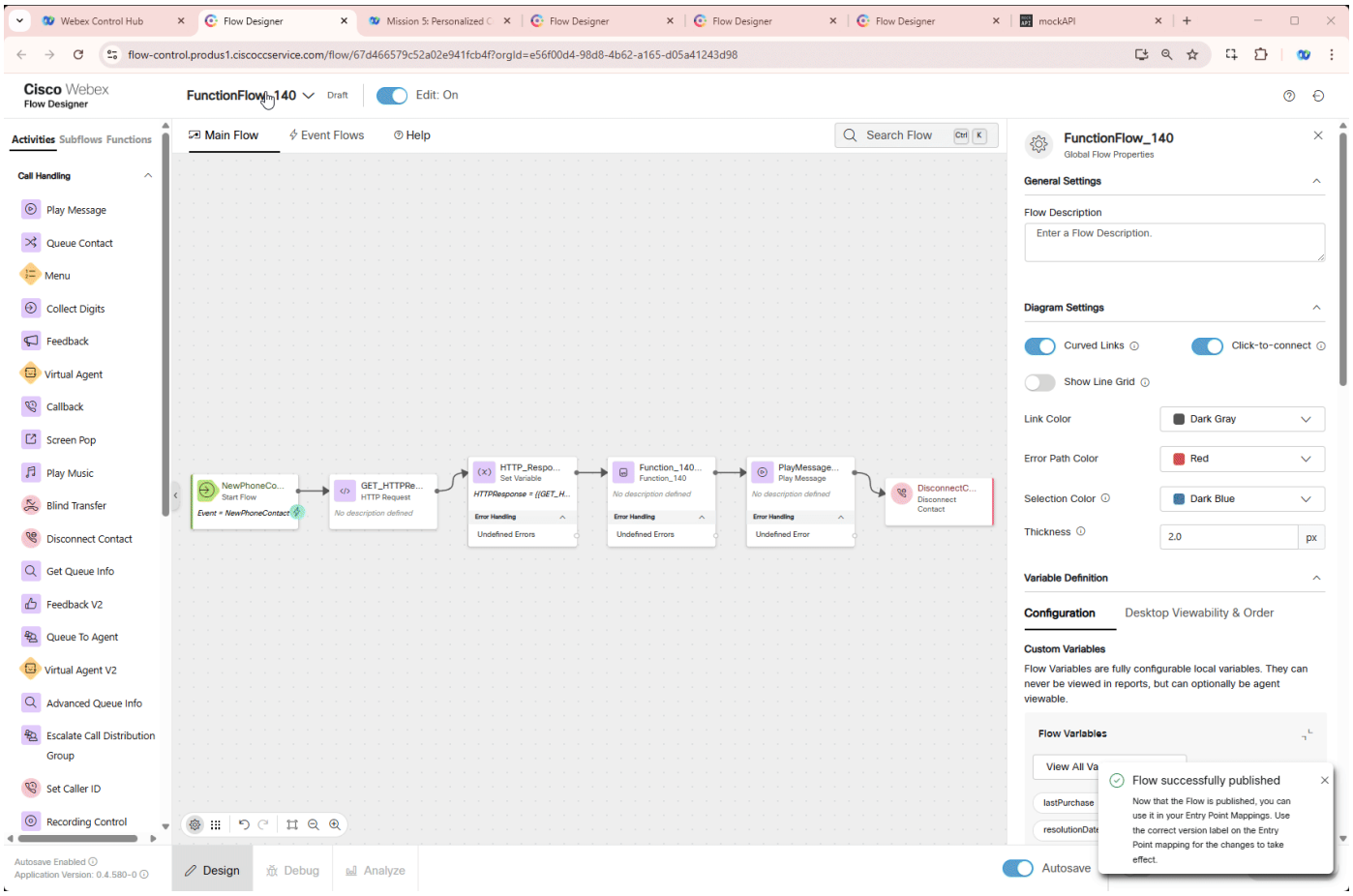
   Navigate to Control Hub > Contact Center > Channels

   Locate your Inbound Channel (you can use the search): **Your_Attendee_ID_Channel**

   Select the Routing Flow: **FunctionFlow_Your_Attendee_ID**

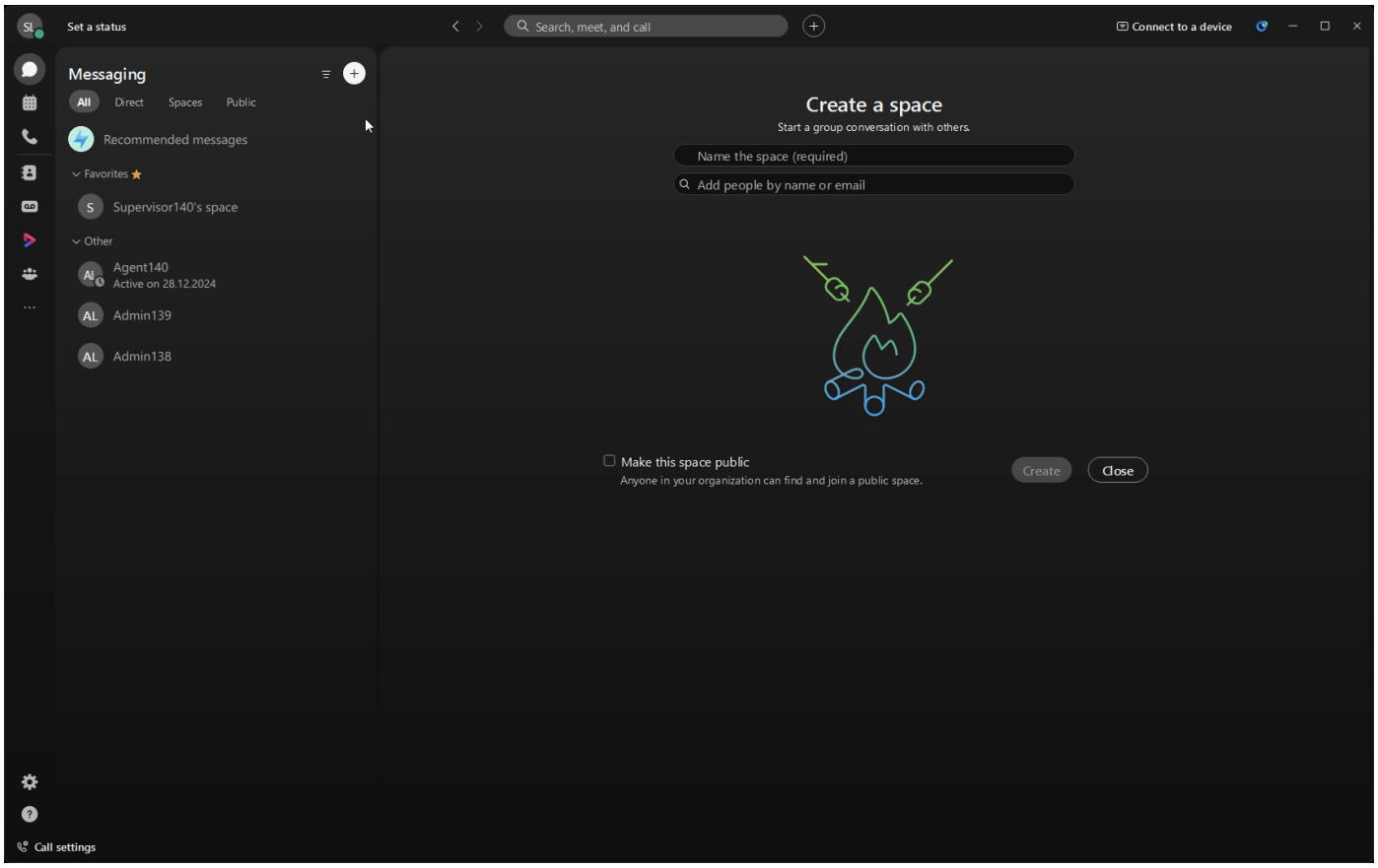
   Select the Version Label: **Latest**

   Click **Save** in the lower right corner of the screen

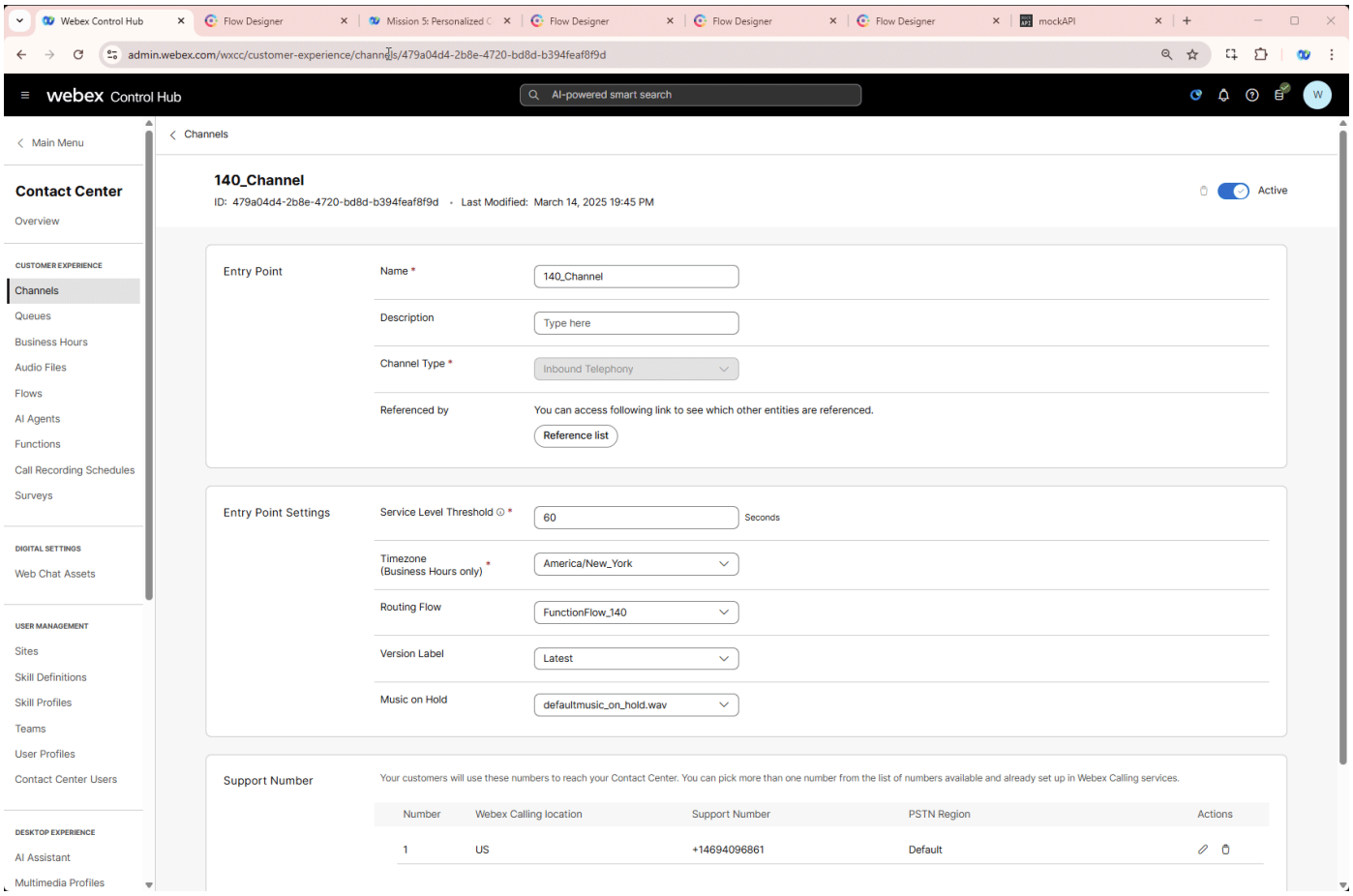


Testing

1. Open your Webex App and dial the Support Number provided to you, which is configured in your **Your_Attendee_ID_Channel** configuration.



2. Verify if you hear a message built inside your function.
3. Switch to Flow Designer and click **Debug** tab at the bottom and select last interaction (the first in the list).
4. Click on **GET_HTTPRequest** and on the right hand side window scroll to the bottom. Check if **Modified Variables** have values assigned.
5. Click on **HTTP_Response** to see full HTTP response body in **Modified Variables**.
6. Click on **Function** step and check if Modified Variables contains `personalizedMessage` built by your function.



Congratulations, you have succesfully completed Personalized Customer Notification with Functions mission! 🎉🎉

3.2 Conclusion

We hope you found the API Track both insightful and rewarding as you expanded your expertise in leveraging APIs for dynamic and intelligent call routing in Webex Contact Center. This session provided hands-on experience with key techniques to enhance flexibility, efficiency, and personalization in customer interactions.

From there, you enhanced your skills with missions such as:

- **Introduction to Developer Portal** – Learning how to navigate and use the Webex Developer Portal to discover APIs, test endpoints, and integrate services efficiently.
- **Emergency Configuration Change** – Using API requests to instantly modify system settings for real-time adaptability.
- **Routing Facilitation with Variables** – Enhancing precision in call routing by dynamically adjusting logic based on real-time data.
- **Last Agent Routing** – Ensuring returning customers are connected with the same agent for a seamless experience.
- **Reconnecting with the same Agent** – Allowing customers to reach the same agent if they call back within 10 minutes after their last call, maintaining conversation continuity.
- **Personalized Customer Notification with Functions** – Using Functions inside Flow Designer to generate tailored customer messages based on dynamic inputs, enabling fully personalized experiences.

By mastering these API-driven techniques, you are now equipped to design smarter, more responsive workflows that enhance both operational efficiency and customer satisfaction.

If you have any questions or need further guidance, feel free to reach out or join the Webex discussion forums. We look forward to seeing how you apply these advanced skills in your future projects!

Thank you for completing the API Track, and we look forward to your continued innovation with Webex Contact Center.

4. CALLBACK TRACK

4.1 Lab Guide

4.1.1 Mission 1: Basic Call Routing (Flow Template, TTS, Language)

> **Note**
>
> The input in the images that follow are only examples. They do not reflect the input you need to use in the lab exercises. In some cases, the input in the images may not follow the same attendee or pod ID from previous images. They are for representation only

Story

Imagine calling a contact center, seeking quick, personalized help. Behind the scenes, a flow smoothly routes your call based on your needs.

CALL FLOW OVERVIEW

1. A new call enters the flow. *(This initiates the interaction and triggers the defined call-handling process.)*
2. The flow determines the caller's language preference and plays a pre-configured Text-to-Speech (TTS) prompt. *(This ensures the caller receives information in their preferred language.)*
3. The call is routed to the appropriate queue. *(This directs the caller to the right team on the flow logic.)*

MISSION DETAILS

Your mission is to:

1. Configure key flow elements for efficient caller journeys.
2. Explore Flow Templates to streamline flow creation.
3. Set up routing with conditions, such as language preference.
4. Gain the skills to design flows for real-world scenarios.

y Flow Templates? [Optional]

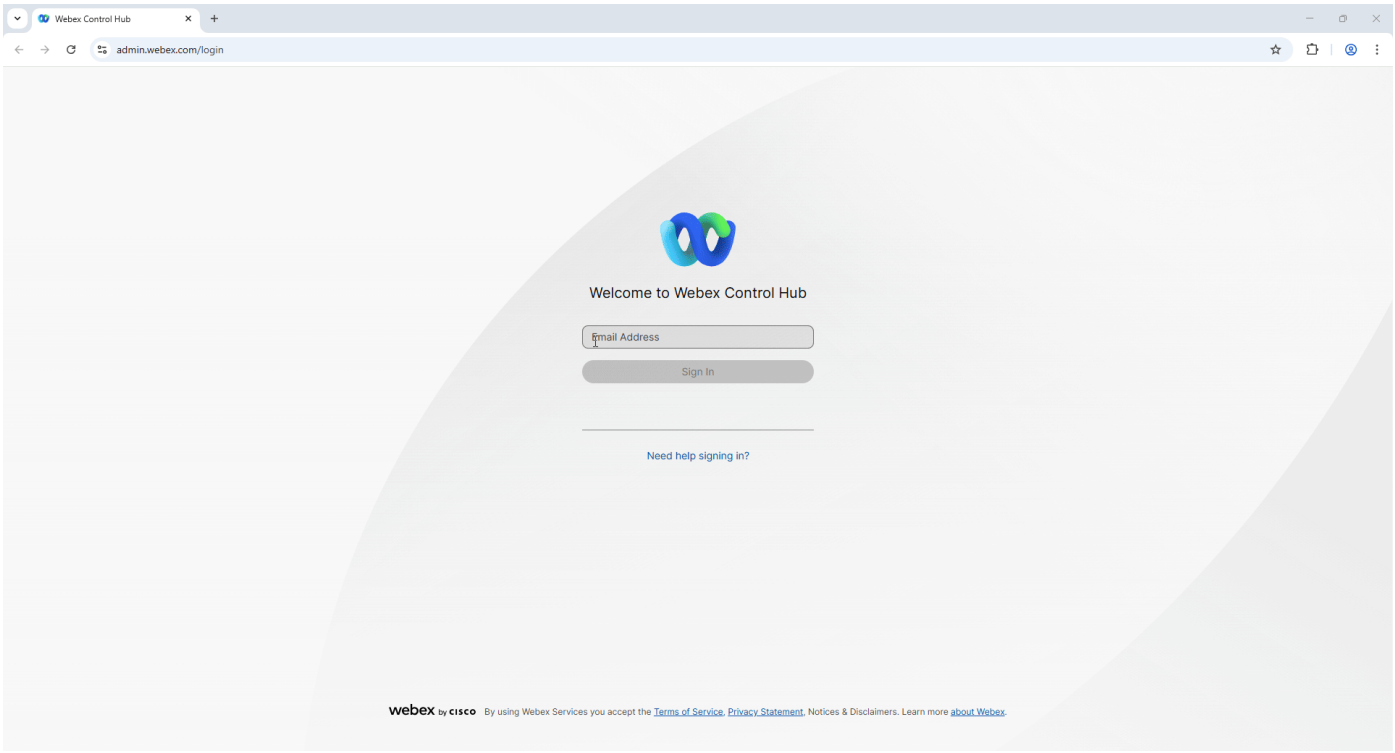
Flow Templates in Webex Contact Center are an essential feature for flow developers, offering a range of benefits that streamline the development process and enhance the efficiency and consistency of flow creation. Here's what they bring to the table:

- **Consistency and Standards**: Templates ensure that flows adhere to best practices, creating consistent experiences across multiple projects.
- **Time Savings**: Pre-built structures reduce the need to start from scratch, enabling faster setup and allowing more focus on customization.
- **Reduced Errors**: Using tested templates lowers the risk of mistakes and minimizes troubleshooting.
- **Easy Onboarding**: New developers or partners can learn quickly by using templates as guides.
- **Scalability**: Templates allow developers to replicate and adapt solutions efficiently across different flows or deployments.
- **Innovation**: Developers can spend more time on unique features and integrations rather than reconfiguring basics.

Flow Templates are designed to empower developers, speed up the development lifecycle, and maintain high-quality standards across flows, making them a core asset in Webex Contact Center flow design.

BUILD

1. Login into Webex Control Hub by using your Admin profile. Your login will be of the format **wxcclabs+admin_IDYour_Attendee_ID@gmail.com**. You will see another login screen with Webex logo on it where you may need to enter the email address again and the password provided to you.



Note

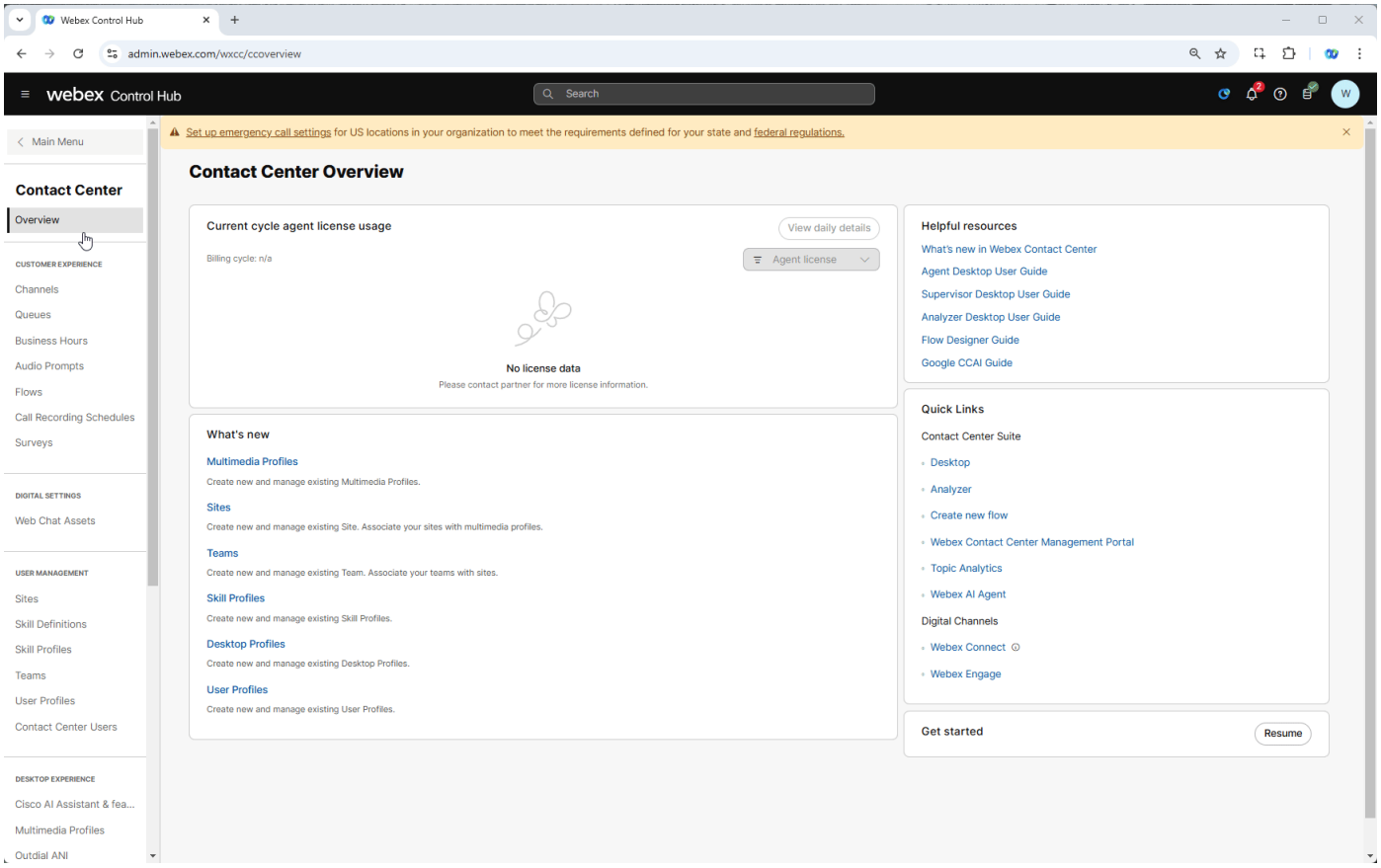
Remember to take up the offer from Chrome to save your password

2. This is the **Administration interface** for Webex Contact Center and is also known as the Control Hub. Look for the contact center option in the left pane under **SERVICES – Contact Center** and click on it.
3. Navigate to **Flows**, click on **Manage Flows** dropdown list and select **Create Flows**.

4. **Create a new flow** tab will be opened. Navigate to **Flow Templates**.
5. Choose **Simple Inbound Call to Queue** template and click **Next**.

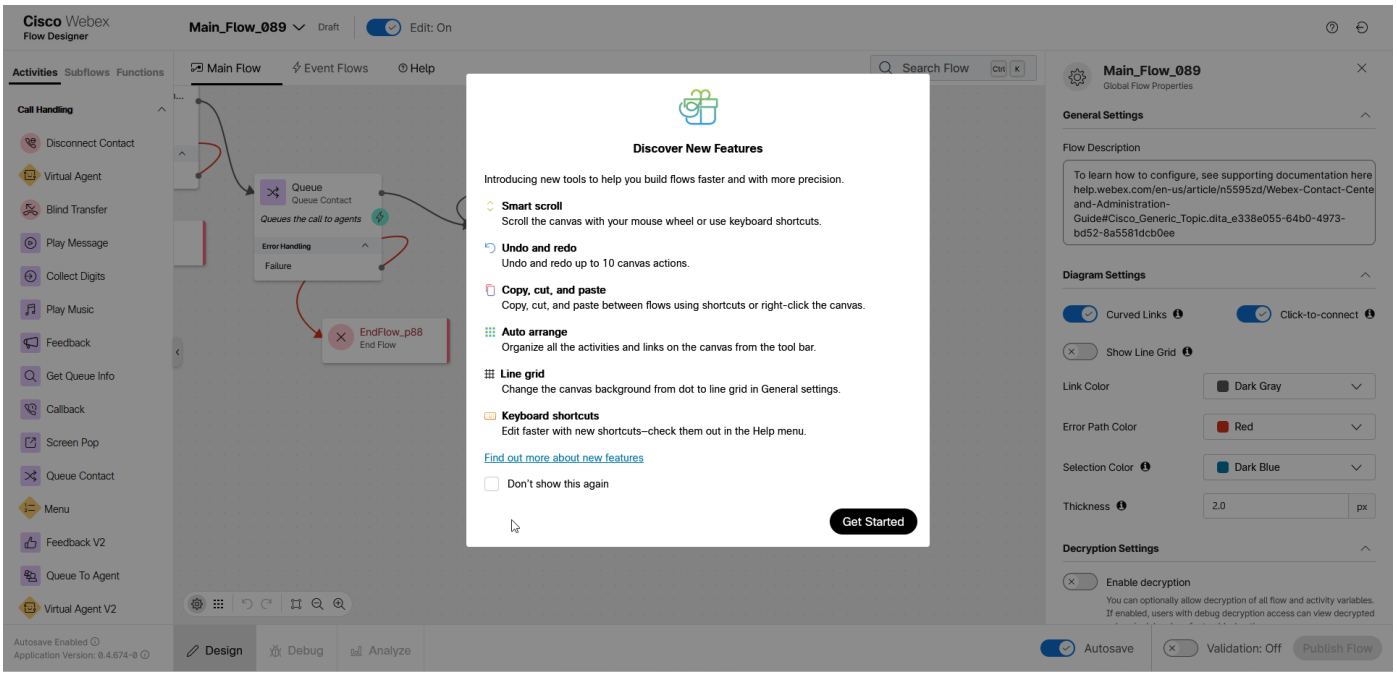
> **Note**
>
> You can press **View Details** link under the template name to observe flow structure and read flow description before proceeding with the template.

6. Name your flow as **Main_Flow_Your_Attendee_ID** Then click on **Create Flow**.



7. If **Discover New Features** popup window appears in front of flow canvas, set the checkbox **Don't show this again** at the bottom and press **Get Started** button to close it and and go to the flow designer.

> **Note**
>
> **Edit** should be set to **On** when you create new flow, but if not switch it from **Edit: Off** mode to **Edit: On** at the top of the page.

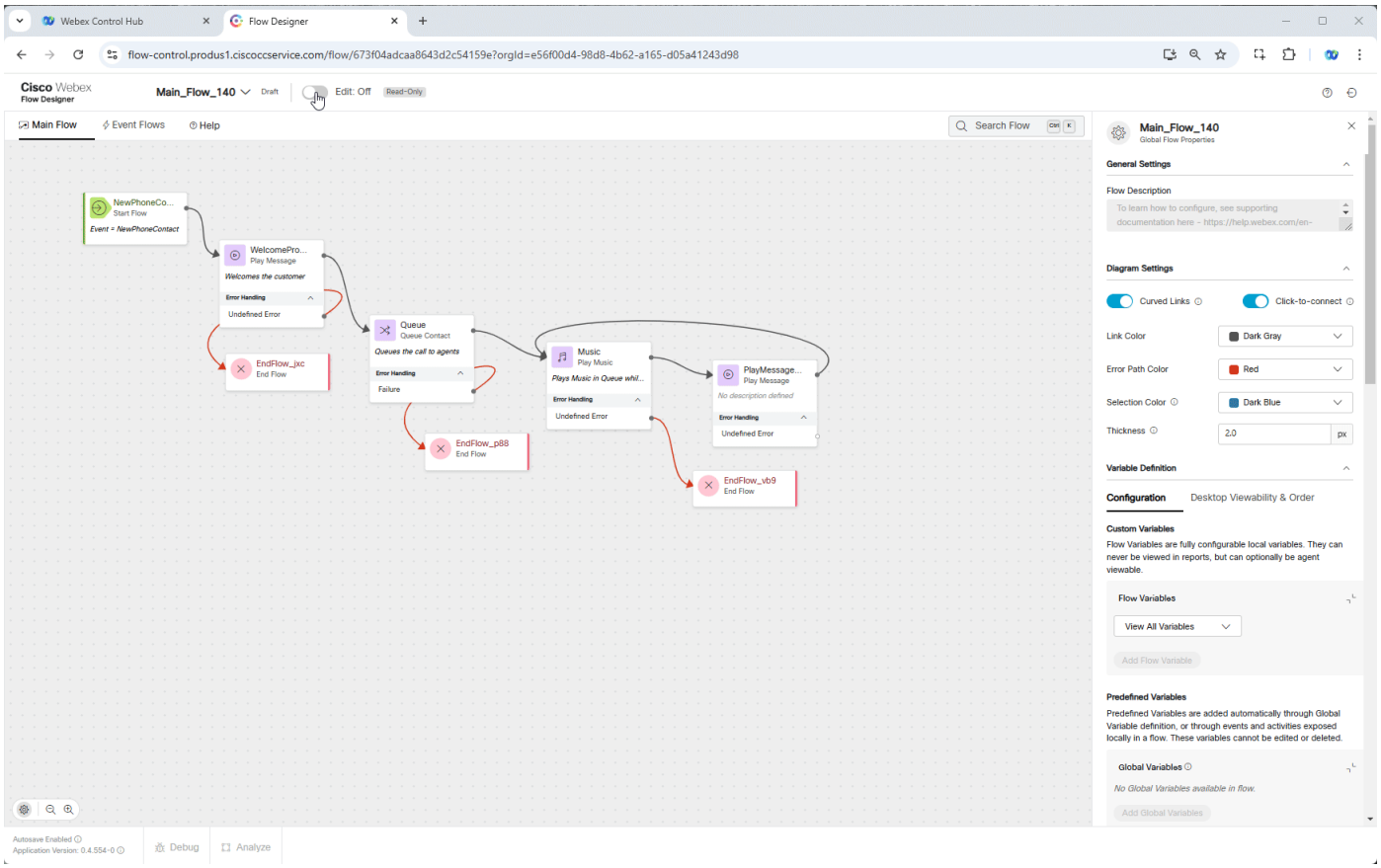


8. Select **Play Message** node with label **WelcomePrompt** and on the node settings modify **Text-to-Speech Message** to any greetings you like. This message will be the first message you hear while calling to your script.
9. Select **Queue** node. On the **General settings** keep Static Queue checked and select queue **Your_Attendee_ID_Queue** from the dropdown list

> **Note**
>
> As mentioned in **Getting Started**, all queues have been preconfigured so you don't need to change them at current step.

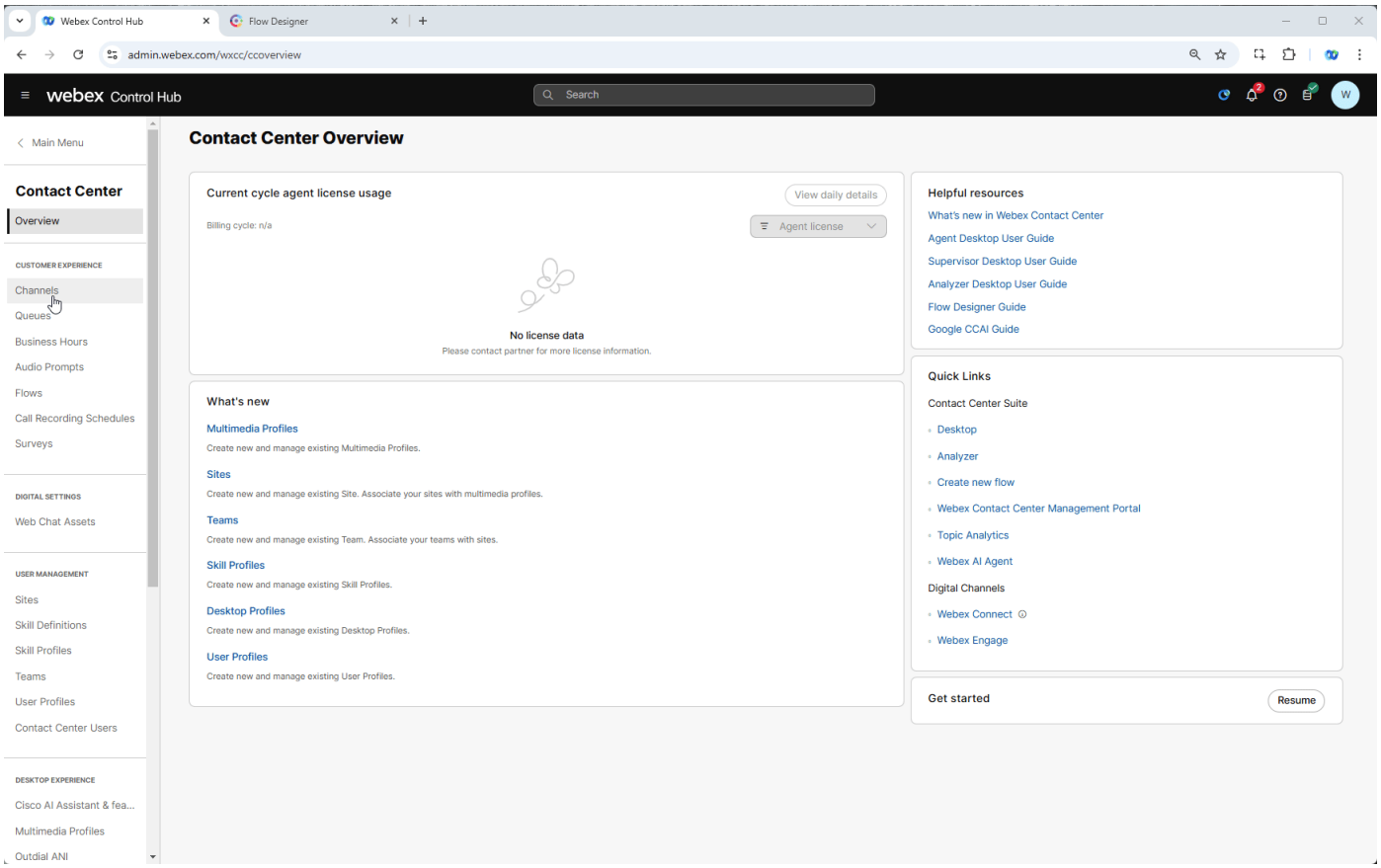
10. [Optional] Select **Play Message** node (the one which goes after **Queue** and **Play Music** nodes) and on the **Node settings** modify **Text-to-Speech Message** to any message you like. This message will be played while the caller is waiting in the queue.
11. Validate and publish the flow:

- Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.
- If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.
- In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.



12. Return to Control Hub to assign the Flow to your **Channel (Entry Point)**. Go to **Channels**, search for your channel **Your_Attendee_ID_Channel**
13. Click on **Your_Attendee_ID_Channel**
14. In **Entry Point** settings section change the following, then click **Save** button:

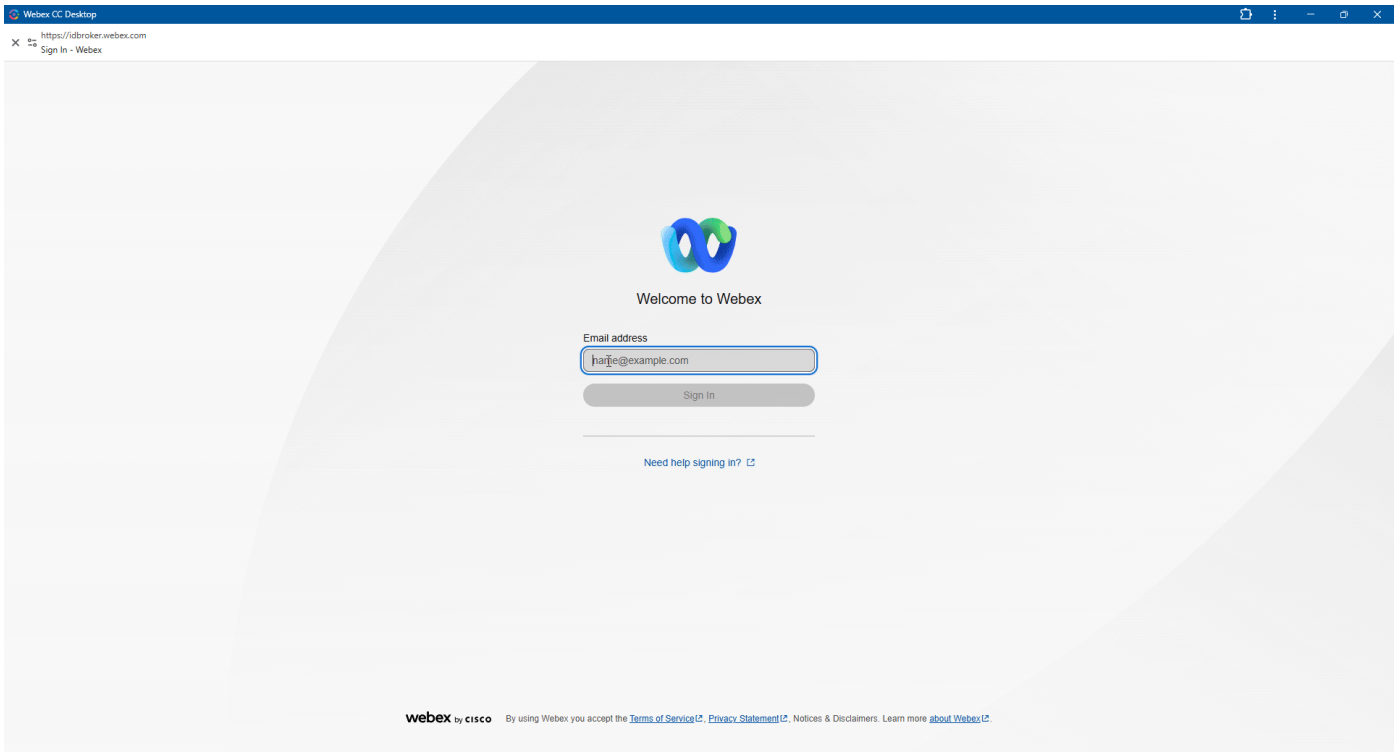
    - Routing flow: **Main_Flow_Your_Attendee_ID**
    - Music on hold: **defaultmusic_on_hold.wav**
    - Version label: **Latest**



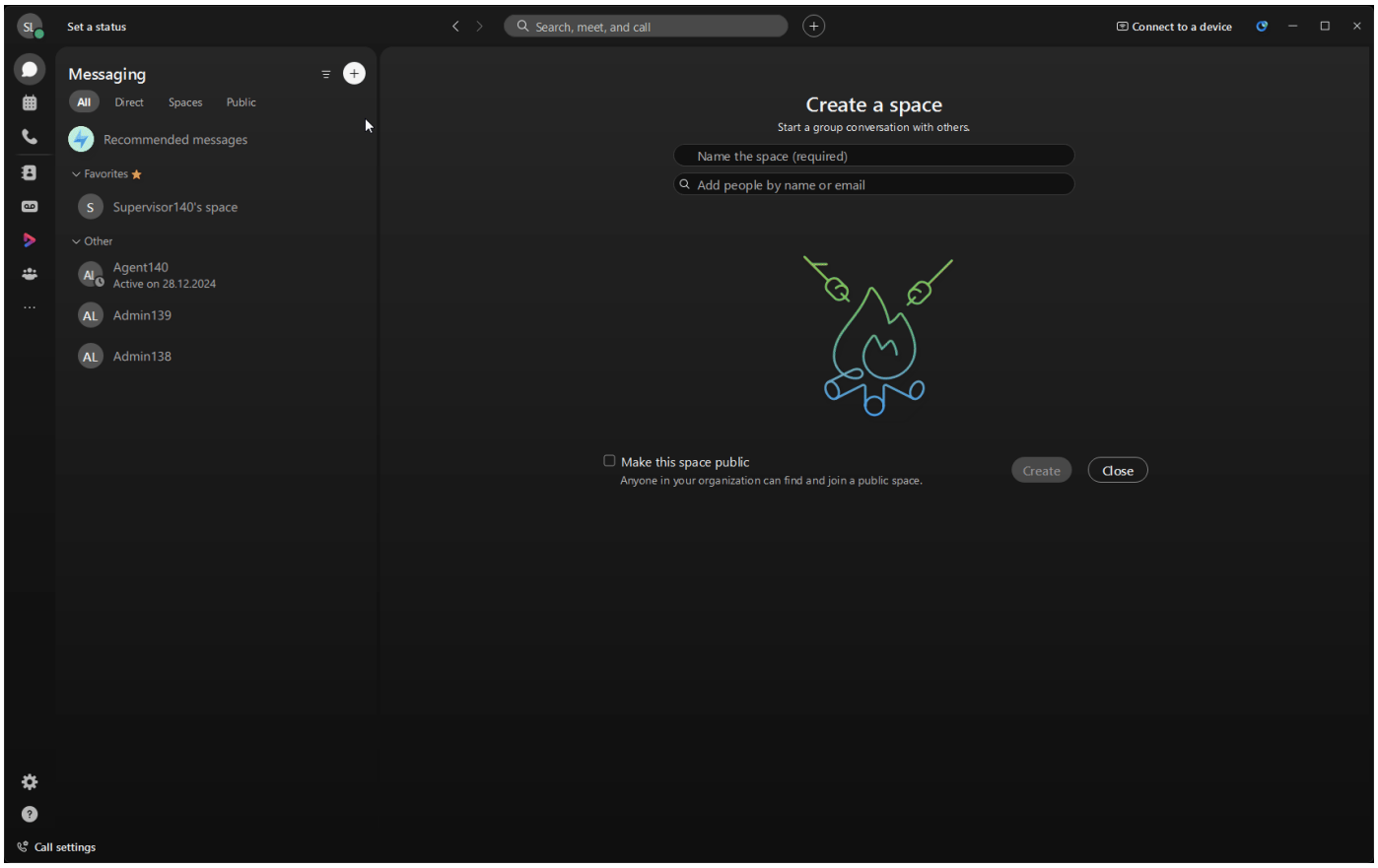
CHECKPOINT TEST

1. Launch Webex CC Desktop application and login with agent credentials you have been provided **wxcclabs+agent_IDYour_Attendee_ID@gmail.com** You will see another login screen with Webex icon on it where you may need to enter the email address again and the password provided to you.
2. Allow notifications and browser to access Microphone by clicking **Allow** in the relevant pop-up prompts.
3. Make your agent **Available** and you're ready to make a call.

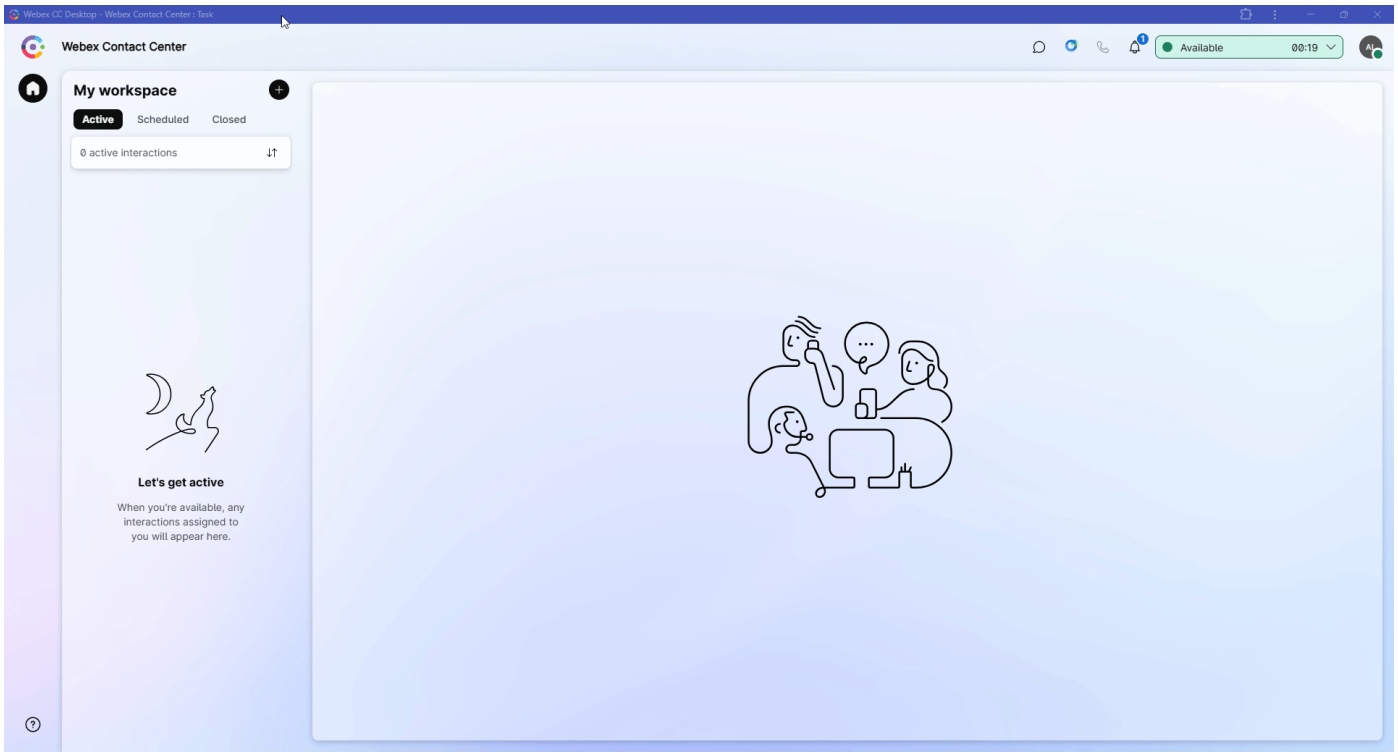
> **Note**
>
> This is the only time during the lab when you need to log in to the Webex CC Desktop application. It has been configured to keep your agent logged into the application for the entire duration of the lab. If, for any reason, you are logged out manually or due to a network error, please log in again as explained above.



4. Open your Webex App and dial the Support Number provided to you, which is configured in your **Your_Attendee_ID_Channel** configuration.



5. Ensure that you hear a welcome prompt configured in your flow and receive an incoming call in your agent desktop. Answer the call in the agent desktop and confirm that it has been successfully established. Then end the call, select any value from the **Wrap Up Reasons** dropdown list on the agent desktop, and press the **Submit Wrap Up** button.



Congratulations, you have succesfully completed Basci Call Routing mission! 🎉

4.1.2 Mission 2: Adding Callback Functionality

Story

Callback functionality is an essential feature in a modern contact center, providing a solution that enhances both customer satisfaction and operational efficiency.

Imagine a customer calls to upgrade their service but faces a 20-minute wait, they can request a callback instead of staying on hold. If no agents are available, they'll be offered the choice to remain in the queue or opt for a callback. Upon choosing the callback, they provide their number, which is validated, and the system schedules the call. Once an agent is free, the system connects with the customer. This ensures businesses retain leads while providing a seamless customer experience.

Call Flow Overview

1. A new call enters the flow.
2. The flow executes the logic configured in previous steps.
3. The call is routed to the appropriate queue, but no agents are available.
4. Since no agents are available, a callback option is offered to the caller.
5. Once an agent becomes available, the callback is initiated to the provided number.

Mission Details

Your mission is to:

1. Continue to use same flow **Main_Flow_Your_Attendee_ID**
2. Add aditional callback functionality to your **Main_Flow_Your_Attendee_ID**.

Build

1. Switch to the Flow Designer. Open your flow **Main_Flow_Your_Attendee_ID**. Make sure **Edit** toggle is **ON**.
2. Delete connection from **Queue** node to **Music**.
3. Add **Menu** node:

   Rename Activity Label to **WantCallback**

   Enable Text-To-Speech

   Select the Connector: **Cisco Cloud Text-to-Speech**

   Click the **Add Text-to-Speech Message** button and paste text: ***All agents are busy. Please press 1 if you want to schedule a callback. Press 2 if you want to wait in queue.***

   Delete the selection for Audio File

   Set the checkbox **Make Prompt Interruptible**

   Under Custom Menu Links:

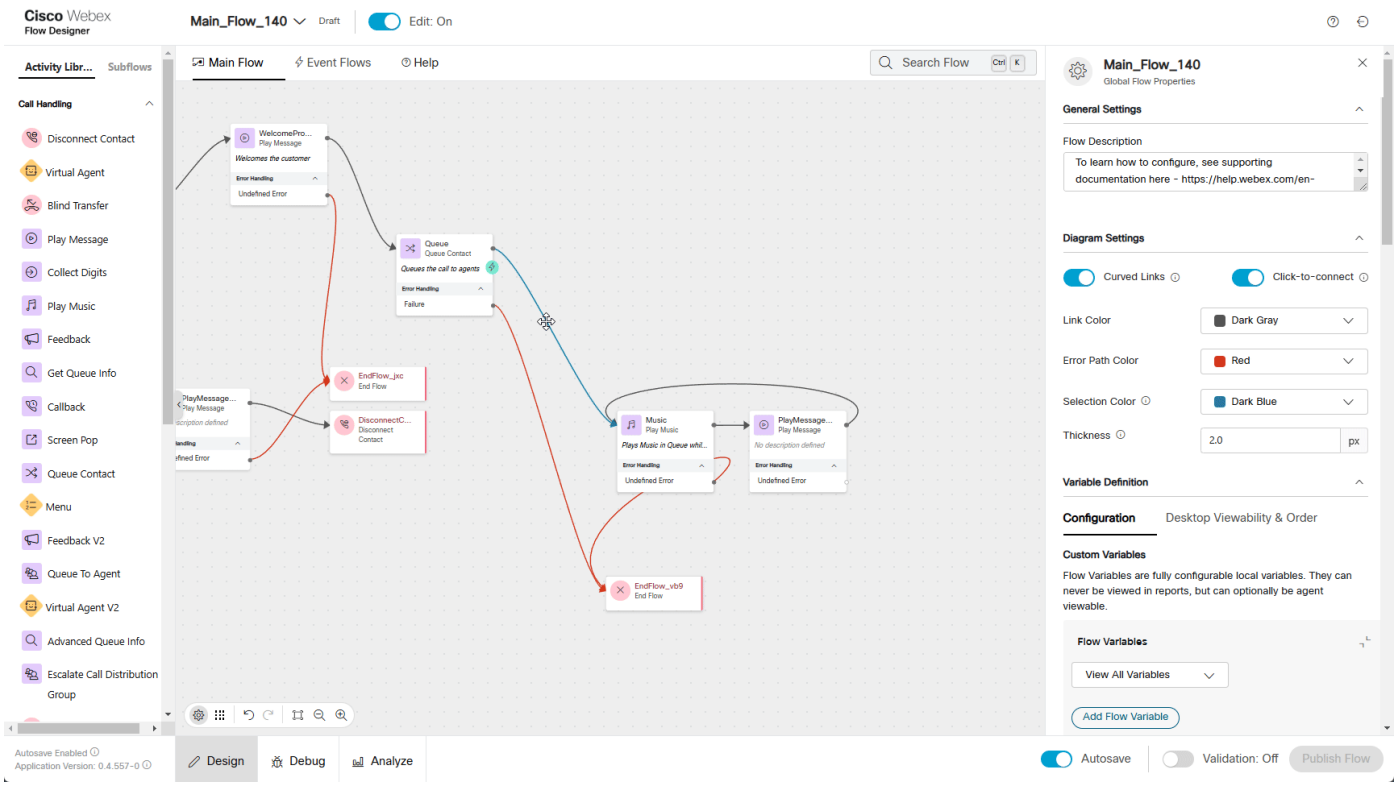
   Change first Digit Number **0** to **1**, add Link Description as **Callback**

   Add New Digit Number as **2** with Link Description **Stay in queue**

   Connect existing **Queue** node to **WantCallBack** node

   Connect **No-Input Timeout** to the front of the **WantCallBack** node

   Connect **Unmatched Entry** to the front of the **WantCallBack** node



4. Add **Collect Digits** node:

   Rename Activity Label to **NewNumber**

   Enable Text-To-Speech

   Select the Connector: **Cisco Cloud Text-to-Speech**

   Click the **Add Text-to-Speech Message** button and paste text: ***Please enter your 11 digits phone number to which we should call you back.***

   Delete the Selection for Audio File

   Set the checkbox **Make Prompt Interruptible**

   Advanced Settings:

   No-Input Timeout: **5**

   Minimum Digits: **9**

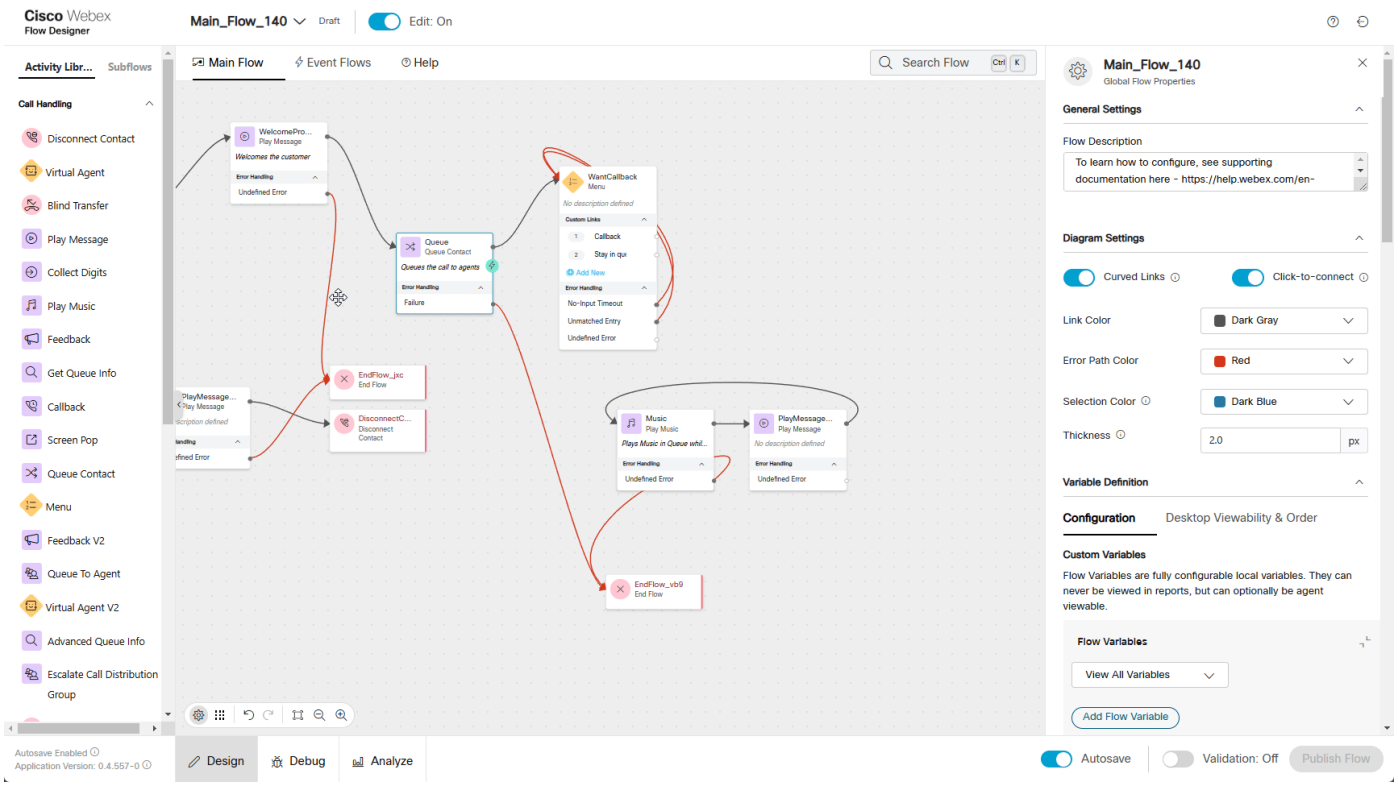
   Maximum Digits: **15**

   Connect **No-Input Timeout** to the front of the **NewNumber** node

   Connect **Unmatched Entry** to the front of the **NewNumber** node

   Connect **Callback** from **WantCallback** node created in step 3 to **NewNumber** node

   Connect **Stay in queue** from **WantCallback** node created in step 3 to **Music** node



5. Add one more **Menu** node:

   Rename Activity Label to **VerifyNumber**

   Enable Text-To-Speech

   Select the Connector: **Cisco Cloud Text-to-Speech**

   Click the **Add Text-to-Speech Message** button and paste the following:

```
<speak>
You entered <say-as interpret-as="telephone">{{NewNumber.DigitsEntered}}</say-as>.
Press 1 if the number is correct.
Press 2 if you want to re-enter the number.
</speak>
```

   Delete the selection for Audio File

   Set the checkbox **Make Prompt Interruptible**

   Custom Menu Links:

   Change first Digit Number from **0** to **1**, add Link Description as **Number OK**

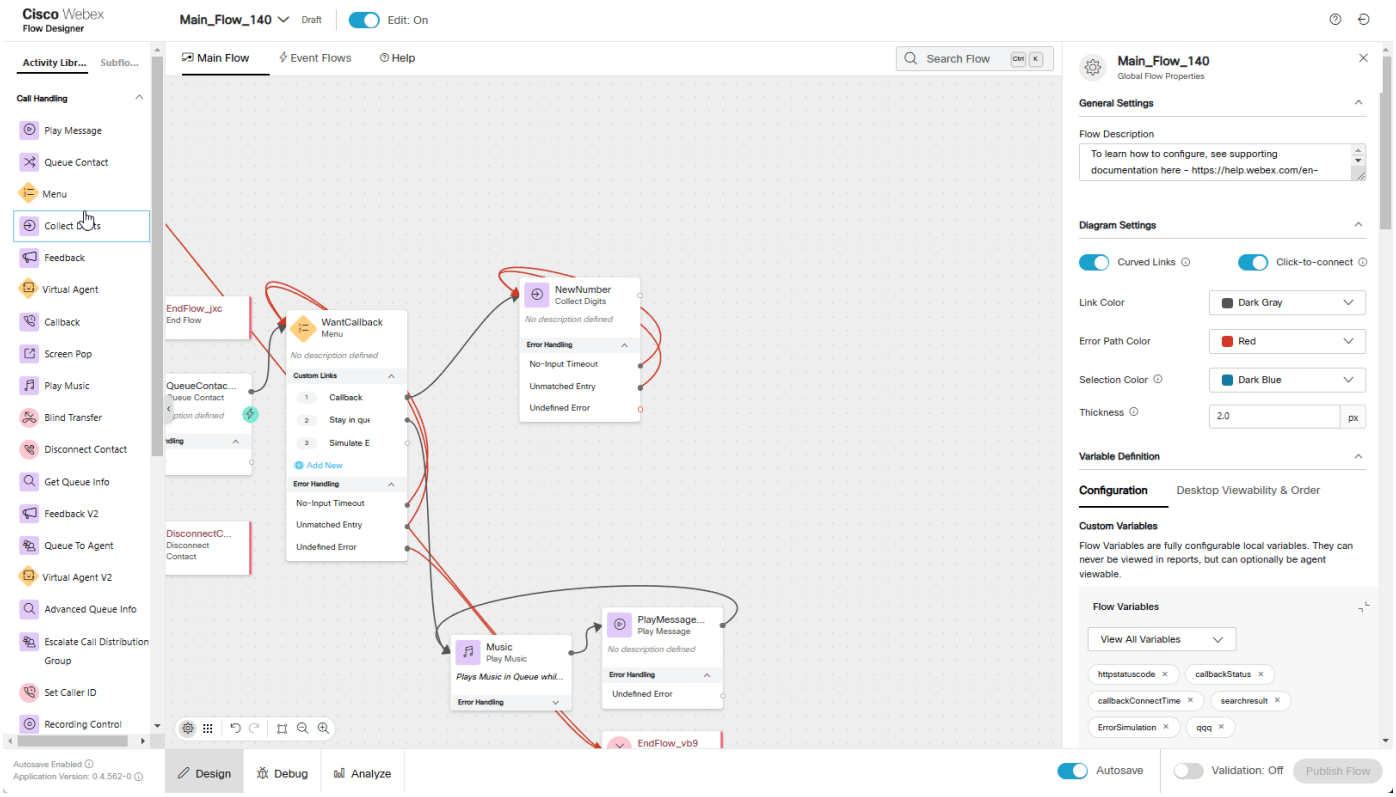
   Add New Digit Number as **2** with Link Description **Number Not OK**

   Connect **No-Input Timeout** to the front of the **VerifyNumber** node

   Connect **Unmatched Entry** to the front of the **VerifyNumber** node

   Connect **NewNumber** created in step 4 to **VerifyNumber** node

   Connect **Number Not OK** from **VerifyNumber** node to **NewNumber** node (aka, **Collect Digits** node) created in Step 4



6. Add **Callback** node:

   Callback Dial Number select ***NewNumber.DigitsEntered*** from dropdown list

   Callback Queue:

   Static Queue: **Your_Attendee_ID_Queue**

   Callback ANI: Choose any number from dropdown list.

   Connect **Number OK** from **VerifyNumber** node created in step 5 to this **CallBack** node

7. Add **Play Message** node as follows:

   Enable Text-To-Speech

   Select the Connector: **Cisco Cloud Text-to-Speech**

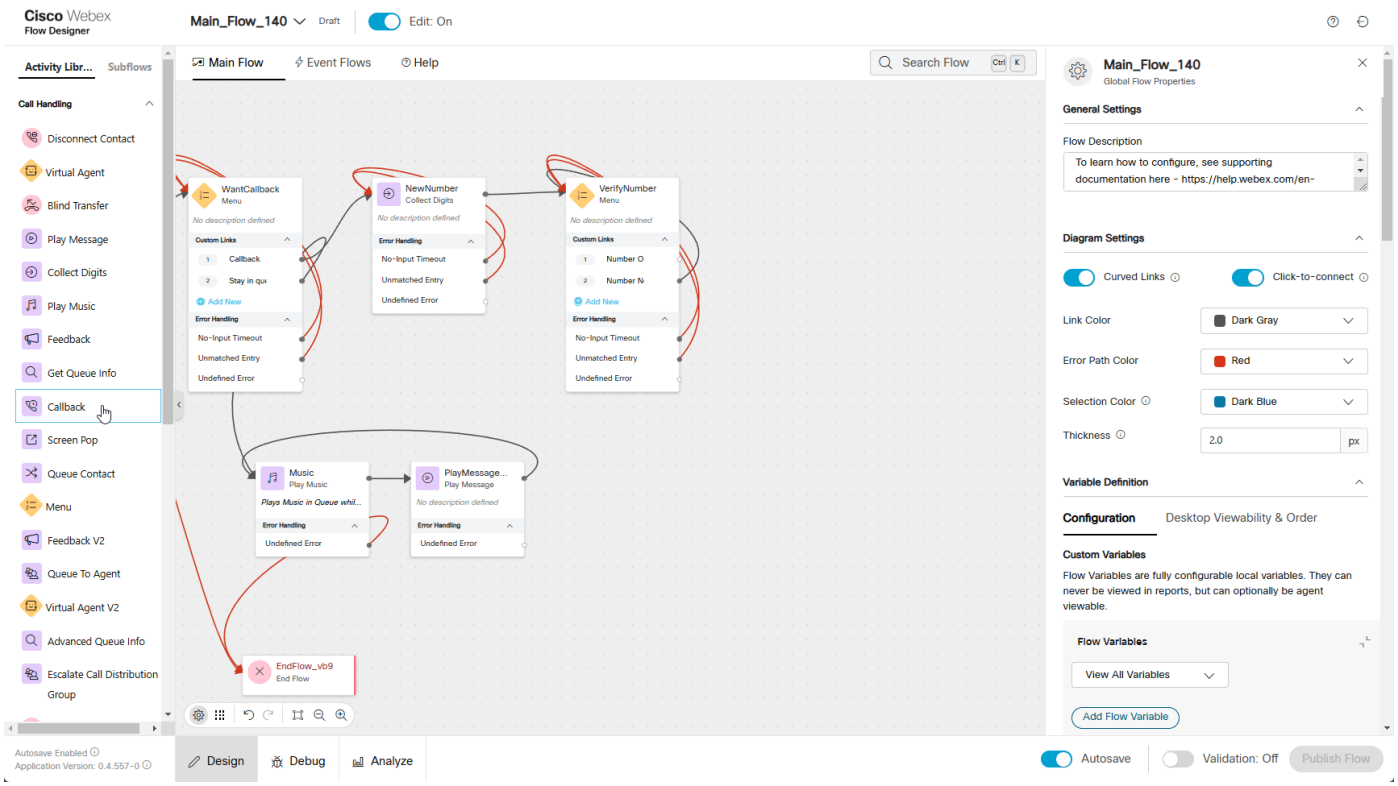
   Click the **Add Text-to-Speech Message** button and paste text: **Your call has been successfully scheduled for a callback. Good Bye.**

   Delete the selection for Audio File

   Connect **CallBack** node created in step 6 to this **Play Message** node

   Connect the output of this **Play Message** node to **Disconnect Contact** node

8. Add **Disconnect Contact** and connect the output of the **Play Message** node you created at step #7 to this **Disconnect Contact** node



9. Validate and publish the flow:

   Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.

   If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.

   In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.

Testing

1. Make sure you're logged into the Webex CC Desktop application as an Agent and set the status to **Not Available**. In this case, the call will not be assigned to an agent, and a callback will be proposed to the caller.
2. Make a call to the Support Number provided to you. If your flow is set up correctly, you should hear a message that you configured, offering you the option to schedule a callback.
3. When callback is proposed, press 1 on Webex App Keypad to request a callback.
4. When asked, provide a new number for a callback. Because in the current lab we have number limitations, we are going to provide a well-known Cisco Worldwide Technical Support contact number **1 408 526 7209** as a callback number. Use the Keypad in Webex app to provide the Cisco Technical Support number, then confirm when asked.
5. Once done, another message about successful scheduling should play.
6. Make your agent **Available**. Webex Contact Center will reserve you right away and propose to answer a callback call.
7. Answer the call and wait until you are connected to a Cisco Technical Support IVR and hear a welcome prompt. Then disconnect the call in agent desktop.

Congratulations, you have succesfully completed Adding Callback Functionality mission! 🎉

4.1.3 Mission 3: Callback on Global Error

Story

Imagine a caller is navigating an IVR menu when, suddenly, the call drops due to an unexpected error in the flow. This unplanned interruption leaves the customer disconnected without completing their request. In this scenario we are going to configure our flow to schedule a callback to the caller when such failure scenario occurs.

Call Flow Overview

1. A new call enters the flow.
2. The flow executes the logic by querying external database for Outbound Channel and ANI.
3. The call is routed to the appropriate queue, but no agents are available.
4. On a callback offering a new option should be selected to simulate an error and drop the call.
5. Once an agent becomes available, the callback is initiated to the number.

Mission Details

Your mission is to:

1. Simulate a global error scenario to trigger a Global Error Event and initiate a workflow to reconnect with a caller whose call was disconnected due to an undefined error.
2. Configure an API POST request to schedule a callback when global error happens. You cannot rely on the Callback node in Main Flow because the call leg is no longer active after termination. Instead, you must design a custom solution to address this limitation.
3. You do not need to configure Outdial Channel and Ourdial Queue as they have been preconfigured for you:

- **Outdial_Your_Attendee_ID_Channel**
- Outdial queue **Outdial_Your_Attendee_ID_Queue** to which your **Your_Attendee_ID_Team** has been assigned.

4. Simulate a real API server. You will use **MockAPI** to retrieve the Outdial channel ID and the target callback number. The retrieved Outdial channel ID will then be used in the Callback API POST request.

od to Know [Optional]

We are starting to use Webex Contact Center APIs in this mission. More information can be found in the **Webex Contact Center for Developers** portal.

- **[ADVANCED] Use MockAPI to enhance your Demos - PART 1**
- **[ADVANCED] Use MockAPI to enhance your Demos - PART 2**

Build

> **Note**
>
> **We are going to extend the same flow by adding additional functionality to simulate a global error scenario which will trigger a callback to a caller.**

1. Switch to the Flow Designer. Open your flow **Main_Flow_Your_Attendee_ID** and make sure **Edit** toggle is **ON**.
2. On the right-hand side, in the **Global Flow Properties** panel, scroll down to locate the **Flow Variables** section under **Custom Variables**. Click the **Add Flow Variable** button and add the following 4 flow variables:

- Outdial Entry Point Variable :

  Name: **outdialcbid**

  Type: **String**

  Default Value: leave it empty

- Custom ANI variable:

  Name: **customani**

  Type: **String**

  Default Value: leave it empty

- HTTP GET Result variable:

  Name: **getresult**

  Type: **String**

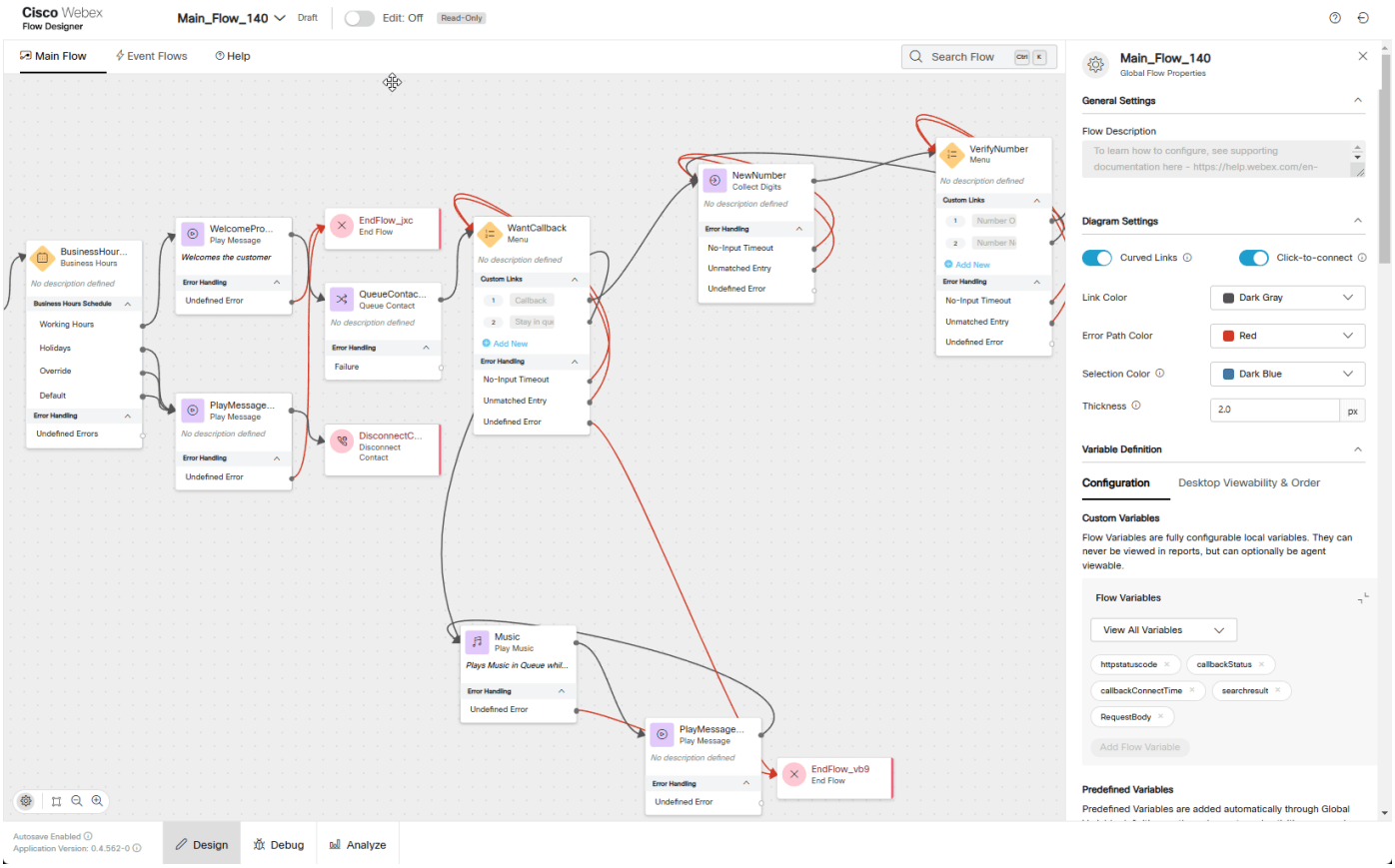
  Default Value: leave it empty

- Simulated Error variable:

  Name: **simulatederror**

  Type: **String**

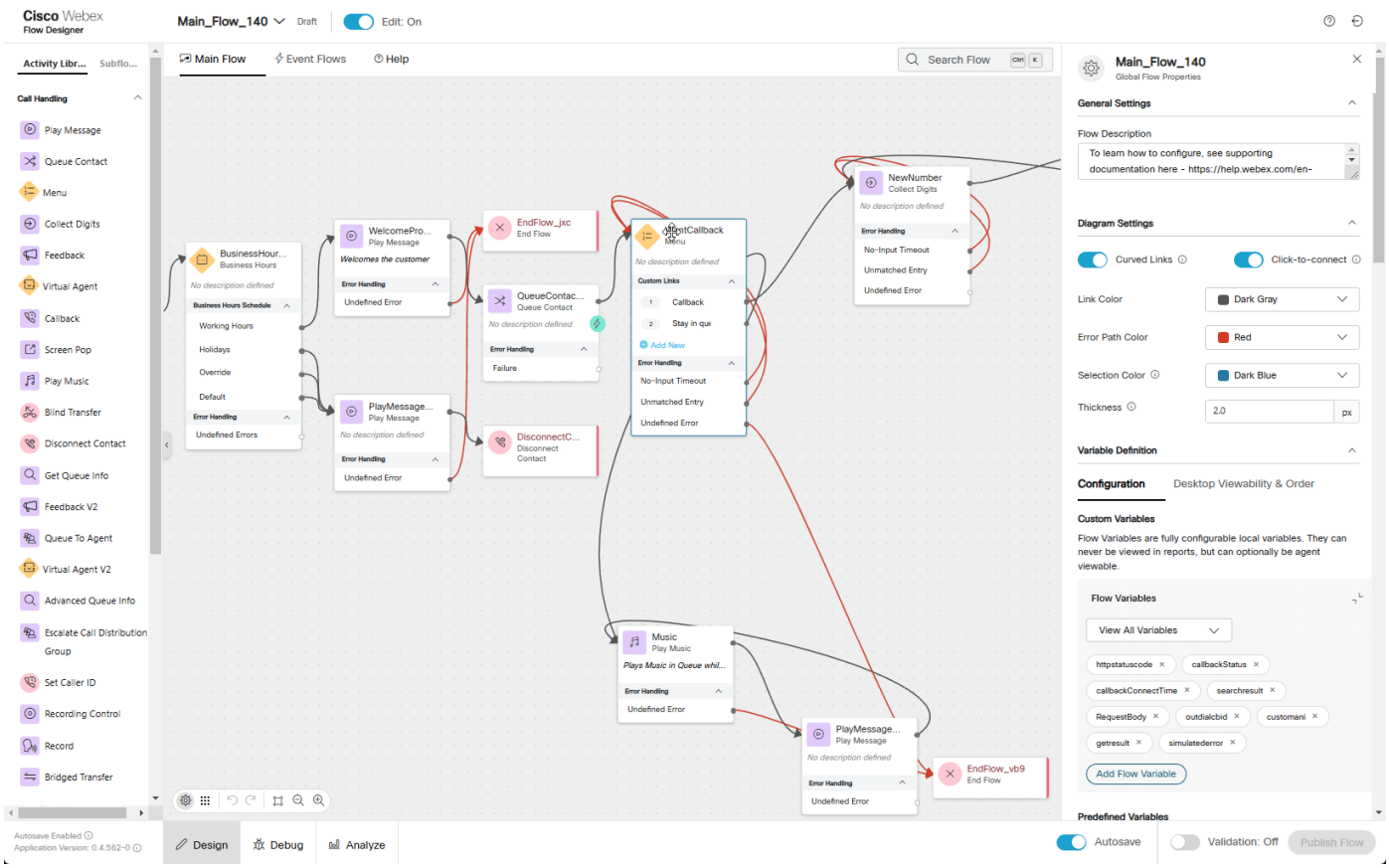
  Default Value: leave it empty



3. Click on **WantCallback** node

   Add Option 3. Name it as **Simulate an error**

   Text-to-Speech Message: ***All agents are busy. Please press 1 if you want to schedule a callback. Press 2 if you want to wait in queue. Press 3 to simulate global error.*** . We are extending the existing message by adding Option 3.



4. Add an **HTTP Request** node for our query. We are going to fetch Outbound Channel/Entry Point ID and custom ANI. Remember we used the same Cisco Worldwide Technical Support contact number in Mission 2 of a Callback Track - "Adding Callback Functionality".

   Connect **WantCallback** Option 3 to this HTTP node

   We will connct **HTTP Request** node in next step

   Activity Name: **GET_CBID**

   Use Authenticated Endpoint: **Off**

   Requestt URL: ***https://674481b1b4e2e04abea27c6e.mockapi.io/flowdesigner/Lab/DynVars?dn={{NewPhoneContact.DNIS | slice(2) }}***

   Method: **GET**

   Content Type: **Application/JSON**

   Parsing Settings:

   Content Type: **JSON**

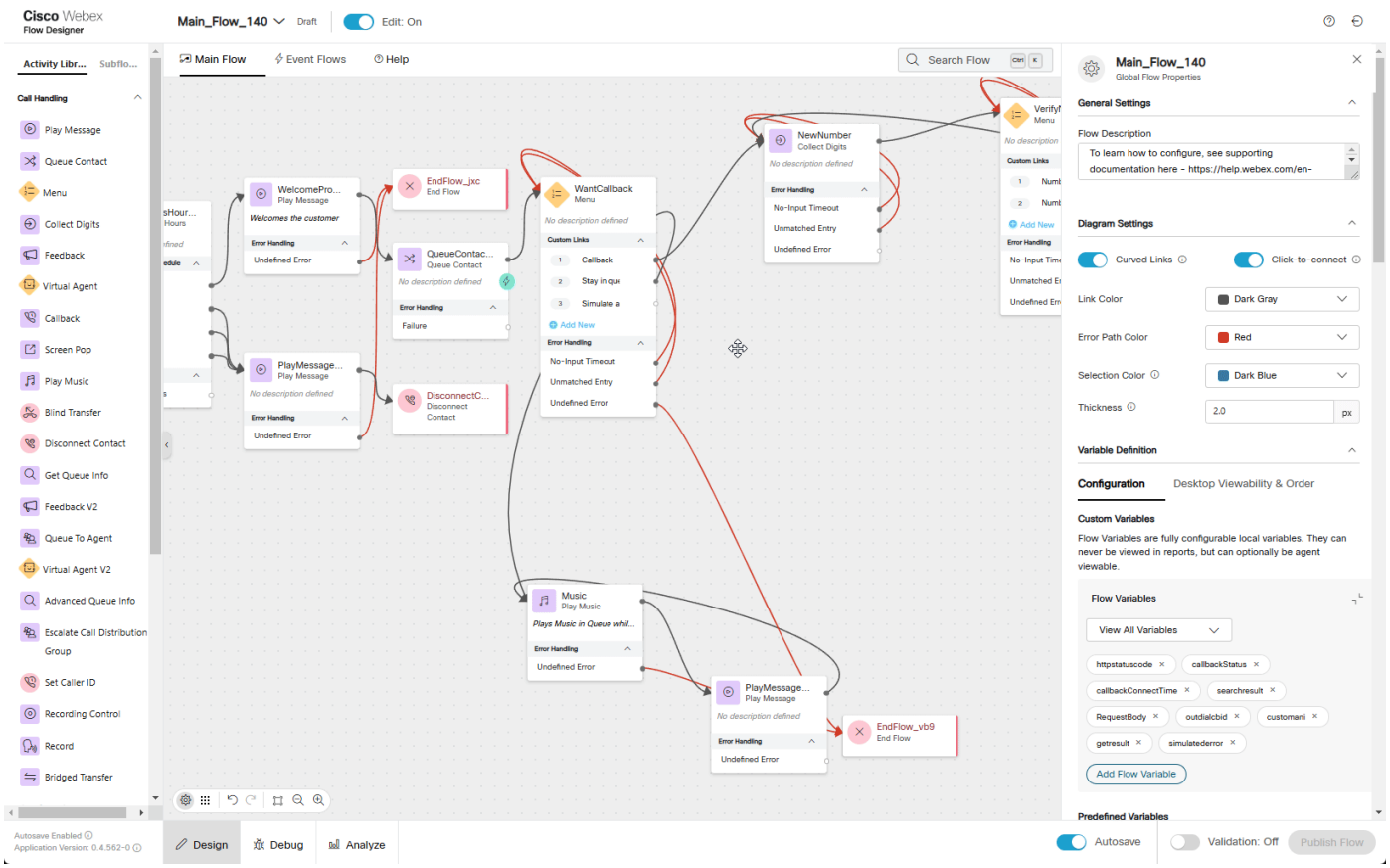
   Output Variable: **outdialcbid**

   Path Expression: **$[0].outboundcallbackep**

   Click **Add New**

   Output Variable: **customani**

   Path Expression: **$[0].tacnumber**

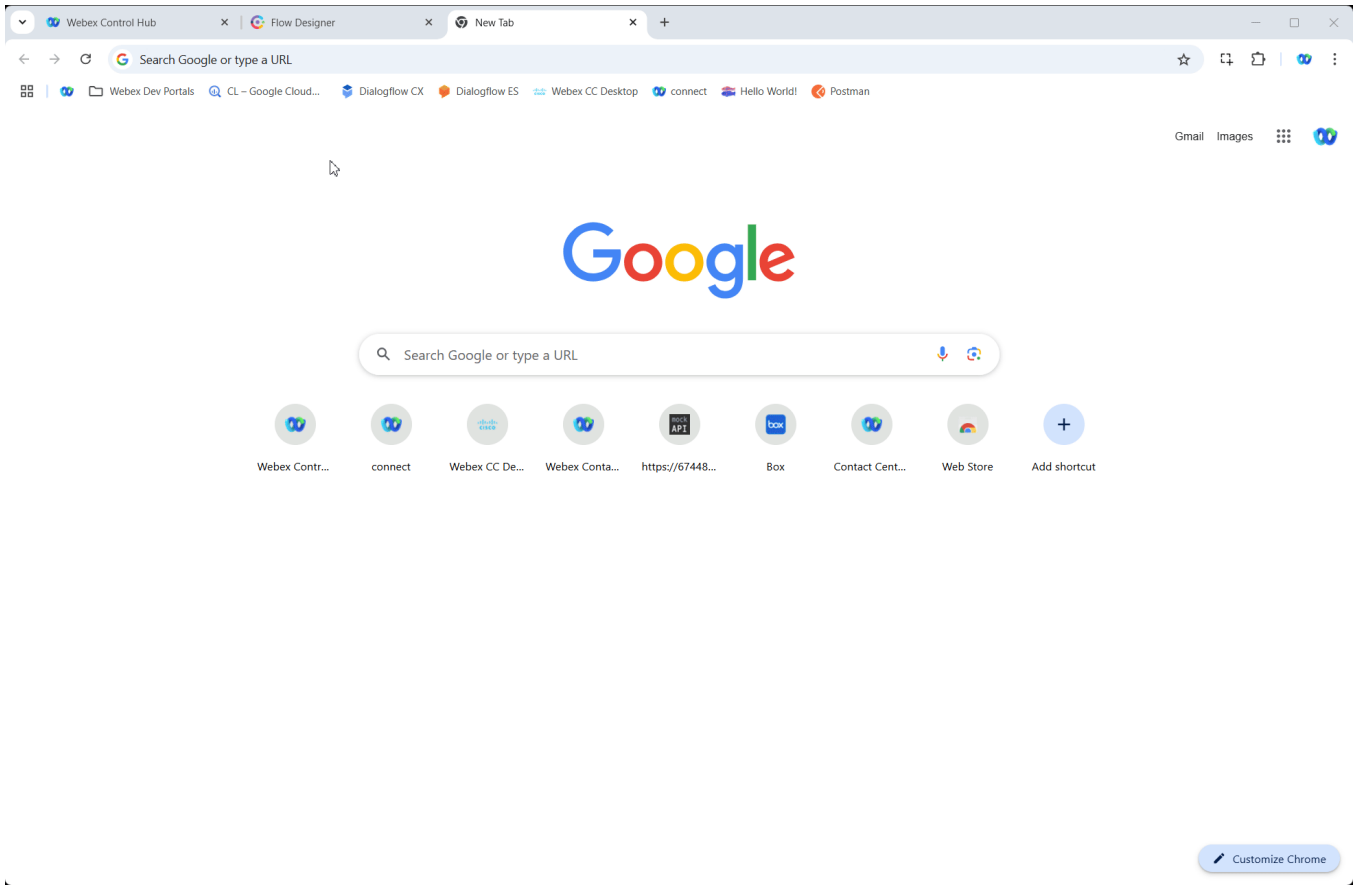


st your API Source [Optional]

a. Test your API resource. **https://674481b1b4e2e04abea27c6e.mockapi.io/flowdesigner/Lab/DynVars?dn={DNIS}**

b. Replace DNIS with the provided DNIS number stripping **+1**. [For example:] If your number **+14694096861**, then your GET Query should be ***https://674481b1b4e2e04abea27c6e.mockapi.io/flowdesigner/Lab/DynVars?dn=4694096861***

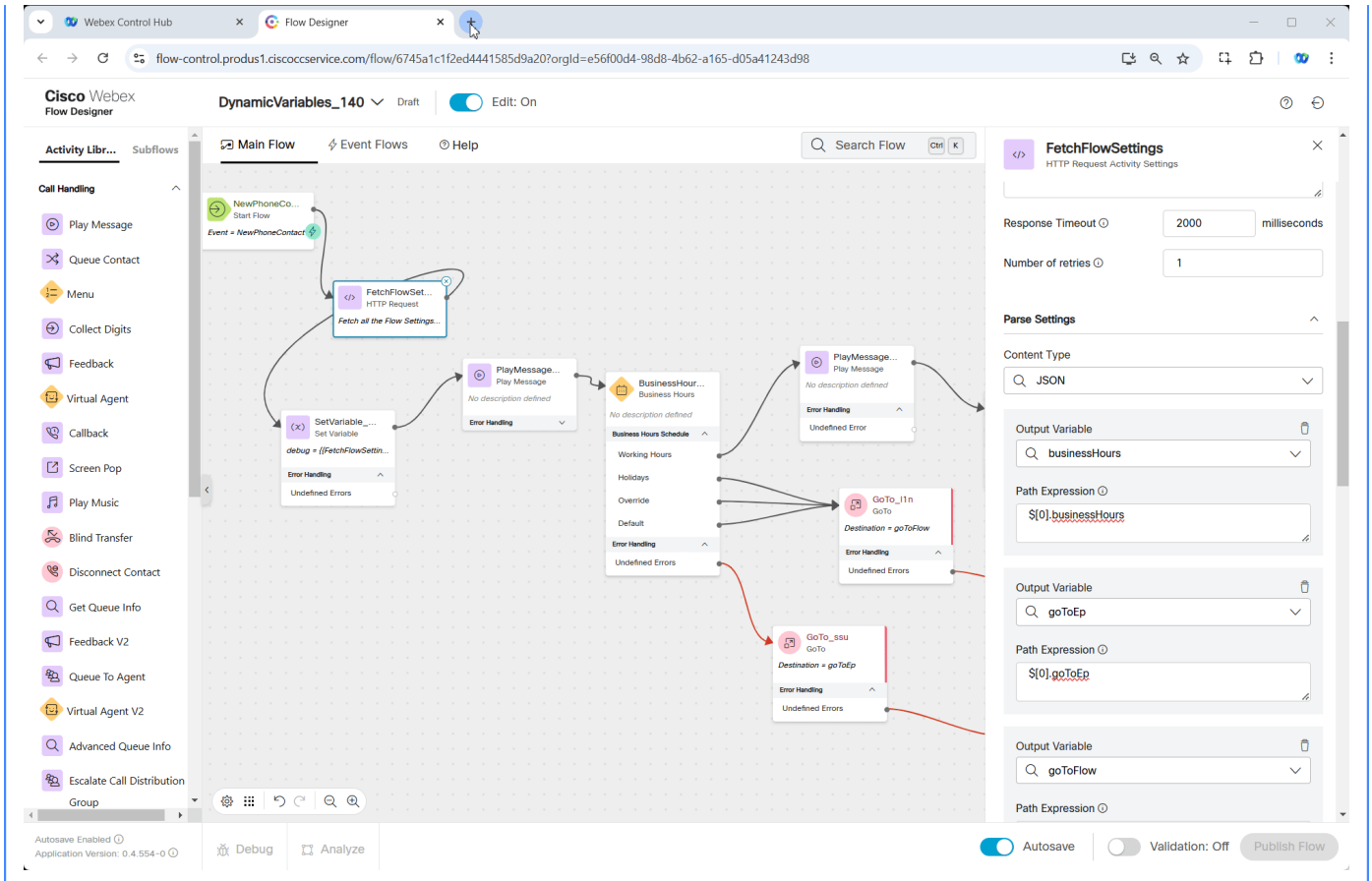
c. Open Chrome browser, paste your Get query URL into the Browser address line and press Enter. You should get the JSON response like this:



d. Open the new browser tab and navigate to JSONPath Online Evaluator

e. Copy the JSON response (including square brackets) you obtained above and paste it into **Document** window of **JSONPath Online Evaluator**.

f. In **JSONPath** box copy and paste one of the path expression from **FetchFlowSettings** to verify your results. For example, **$[0].businessHours**



5. Add **Set Variable** node

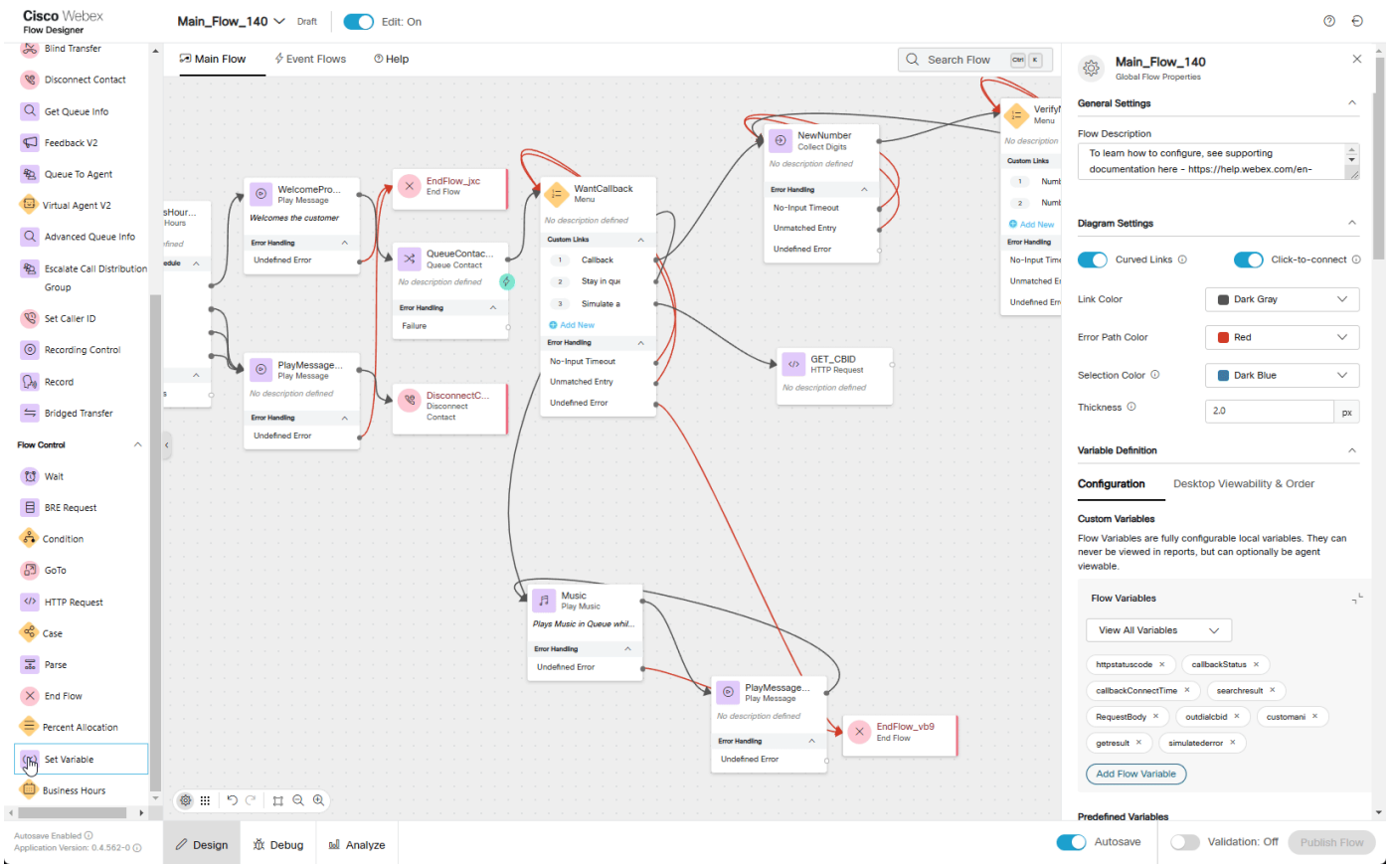
   Activity Label: **SetGetResult**

   Connect **GET_CBID** to this node

   We will connct **Set Variable** node in next step

   Variable: **getresult**

   Set Variable: **GET_CBID.httpResponseBody**



6. Add one more **Set Variable** and **Disconnect Contact** nodes. We are going to intentionally configure an incorrect value in the **Set Variable** node to forcibly trigger a Global Error.

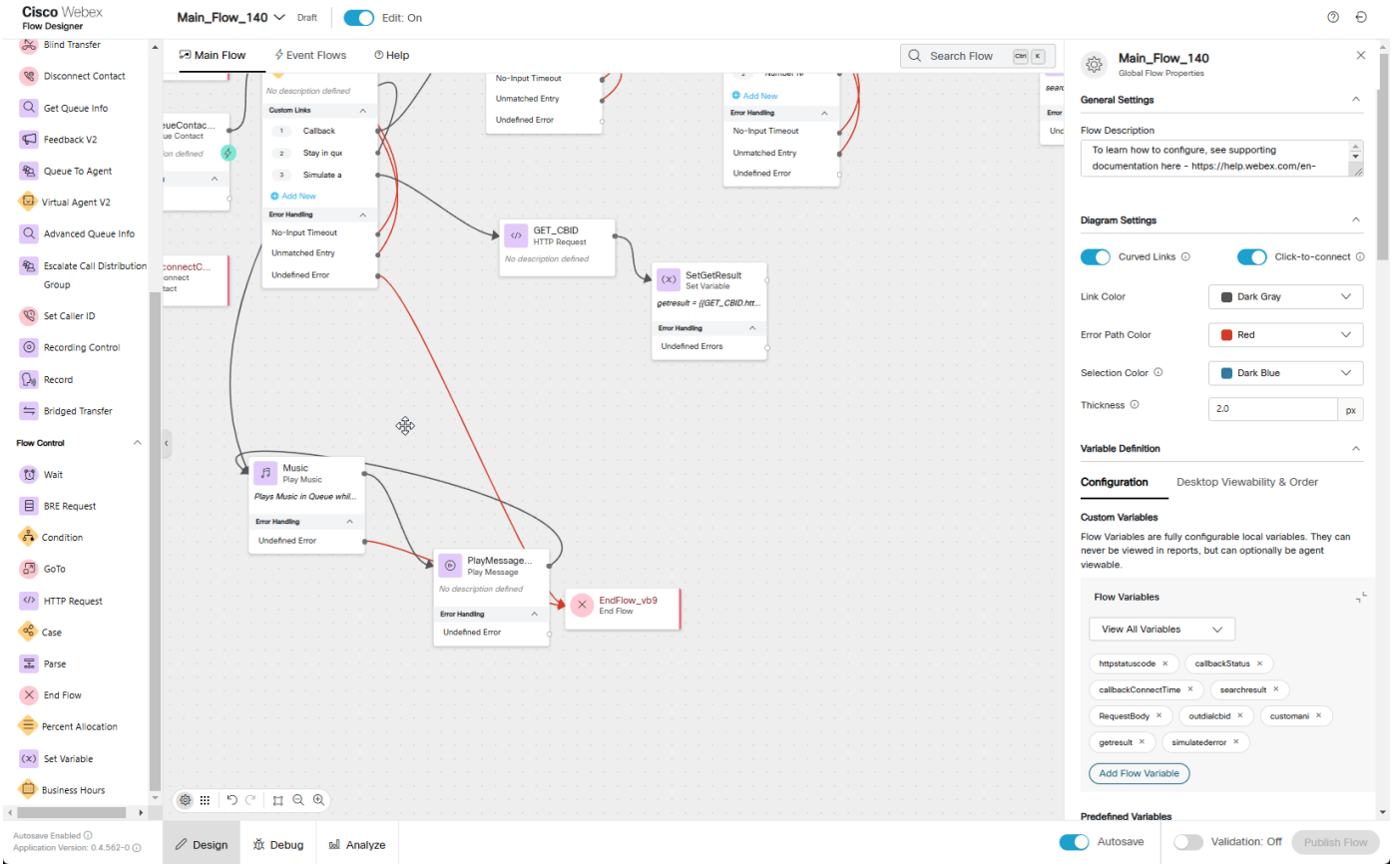
   Activity Label: **SimulateGlobalError**

   Connect **SetGetResult** to this node

   Connect this node to **Disconnect Contact** node

   Variable: **simulatederror**

   Set Value: ***{{ ANI | 123}}***



7. Navigate to **Event Flows** and delete connection from **OnGlobalError** to **EndFlow**.

8. Add **HTTP Request** node to the flow. In this step we are going to build a **Create Task** API POST request. See **Create Task API** for details.

   Activity Label: **CallBackAPI_HTTPRequest**

   Connect the **OnGlobalError** output edge node to this node

   Use Authentication Endpoint: **On**

   Connector: **WxCC_API**

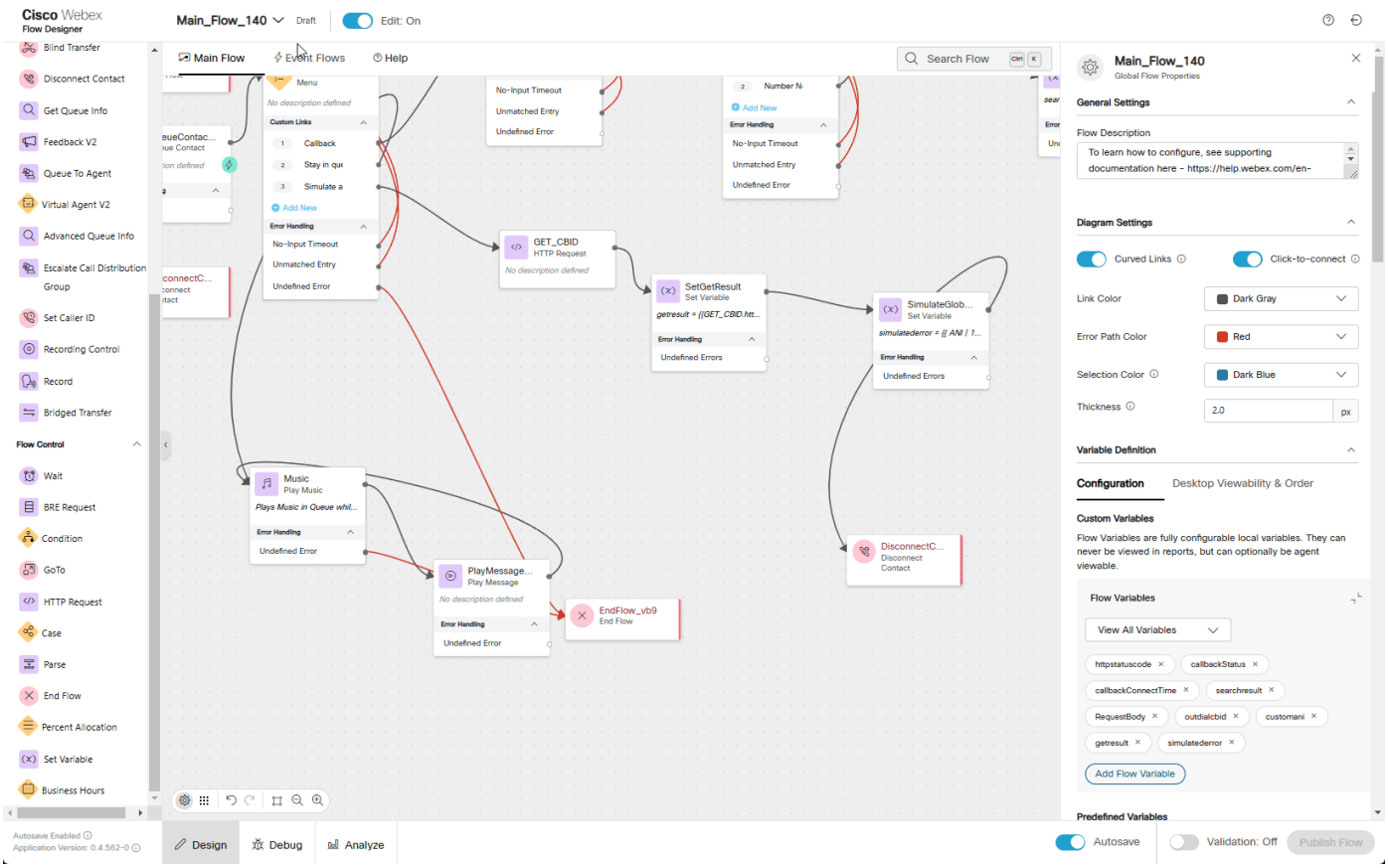
   Request Path: **/v1/tasks**

   Method: **POST**

   Content Type: **Application/JSON**

   Request Body:

```
{
    "entryPointId": "{{outdialcbid}}",
    "destination": "{{customani}}",
    "attributes": {"Message":"tester","To Queue":"sales"},
    "outboundType": "CALLBACK",
    "mediaType": "telephony",
    "callback": {
    "callbackOrigin": "web",
    "callbackType": "immediate"
    }
}
```



9. Add **Condition** node. In this node we are going to check the status of our API POST request. If HTTP response is **201 Created** the output will be **True** and if other than **201** then **False**.

   Activity Label: **HTTPStatusCode**

   Connect the output node edge from the **CallBackAPI_HTTPRequest** node to this node

   Connect both **True** and **False** exists to **EndFlow** node. We will be able to see in Debug tool whether request was succsesful or not.

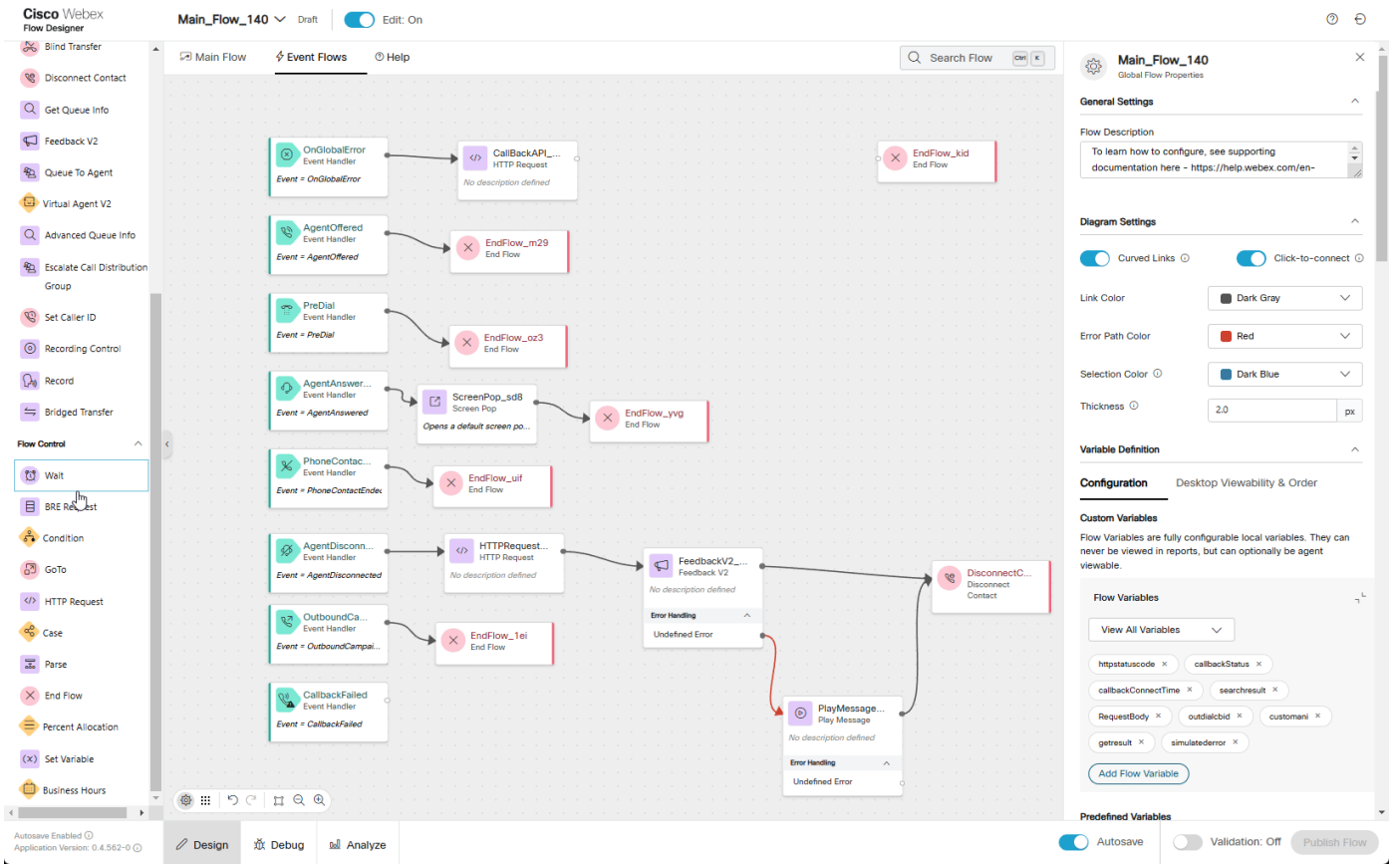
   In the Expression section write an expresion ***{{CallBackAPI_HTTPRequest.httpStatusCode == 201}}***

10. Validate and publish the flow:

    Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.

    If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.

    In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.



11. Switch to **Control Hub**. Navigate to **Channels** under **Customer Experience Section**, locate your channel **Your_Attendee_ID_Channel**.
12. Click on **Your_Attendee_ID_Channel**
13. In **Entry Point** settings section change the following, then click **Save** button:

    Routing Flow: **Main_Flow_Your_Attendee_ID**

    Version Label: **Latest**

Testing

1. Make sure you're logged into Webex CC Desktop application as Agent and set status to **Not Available** (select any Idle state). In this case call will not be assigned to an agent and callback will be proposed to a caller.
2. Make a call to the Support Number and if success you should hear configured messages.
3. Next message will propose you options to request callback, stay in queue or simulate an error. Press **3** on Webex App Keypad to simulate an error.
4. If everything configured correctly your call should be disconnected.
5. Open Debug tool in your **Main_Flow_Your_Attendee_ID** and click on first call in the list which should be the last call you made. Look for **WantCallback** in Activity Name column and make sure the call left **WantCallback** out of Option 3 and continue through **GET_CBID**.

6. Click on either **GET_CBID** node of the flow or on Activity Name **GET_CBID** in the Debug tool and scroll to the bottom the right-hand side section of Debug tool. Under **Modified Variables** you should see values assigned to **outdialcbid** and **customani** flow variables. Where **outdialcbid** is ID of your **Outdial_Your_Attendee_ID_Channel** and **customani** is a well known Cisco Worldwide Support contact number **1 408 526 7209**. The same number we used in previous exercise. This time we used an external database as well as GET API call to extract that number.
7. While still on Debug tool, click on **SetGetResult** to see full response from HTTP request that we wrote into **getresult** flow variable.
8. Make sure **SimulateGlobalError** activity name has an **Error** next to it in **Outcome** column. That mean you successfully simulated **Global Error** event.
9. Click on next activity name **GlobalErrorHandling** which goes after **SimulateGlobalError** activity name. Flow Designer automatically will open **Event Flows** tab.
10. Observe **Condition** node to make sure exit went out via **True** exit. This tells you that HTTP response is **201 Created** and callback has been scheduled successfully.
11. On Webex Desktop, make your agent **Available**. Webex Contact Center will reserve your agent right away and propose to answer a callback call.
12. Answer the call and wait until you are connected to a Cisco Technical Support IVR and hear a welcome prompt. Then disconnect the call in agent desktop.

Congratulations, you have succesfully completed Callback on Global Error mission! 🎉

4.1.4 Mission 4: Preventing Callback duplication

> **Note**
>
> This task relies on completing Mission 3 of the Callback Track. Ensure that mission is completed to have a fully functional callback feature in your flow.

Story

If a caller who already has a scheduled callback contacts the contact center again to request another callback, our system can recognize this. It will then notify the caller that a callback is already scheduled and will be completed as soon as the next agent becomes available.

Call Flow Overview

1. A new call from a caller, who already has a scheduled callback, enters the flow.
2. The flow executes the logic configured in previous missions.
3. The call is routed to the appropriate queue, but no agents are available.
4. Since no agents are available, a callback option is offered to the caller.
5. Once the caller requests for a callback, IVR replies that callback has been scheduled already.

Mission Details

Your mission is to:

1. Enhance the functionality of the **Main_Flow_Your_Attendee_ID** by introducing an advanced feature to check if a callback already exists for a specific tested number.
2. Use **Search API** request to fetch the data from Analyzer database. For more details see **Search API** for details.

Build

1. Switch to the Flow Designer. Open your flow **Main_Flow_Your_Attendee_ID** and make sure **Edit** toggle is **ON**.
2. On the right-hand side, in the **Global Flow Properties** panel, scroll down to locate the **Flow Variables** section under **Custom Variables**. Click the **Add Flow Variable** button and add the following 3 flow variables:

- Callback Status variable:

  Name: **callbackStatus**

  Type: **String**

  Default Value: leave it empty

- Callback Connect Time variable:

  Name: **callbackConnectTime**

  Type: **String**

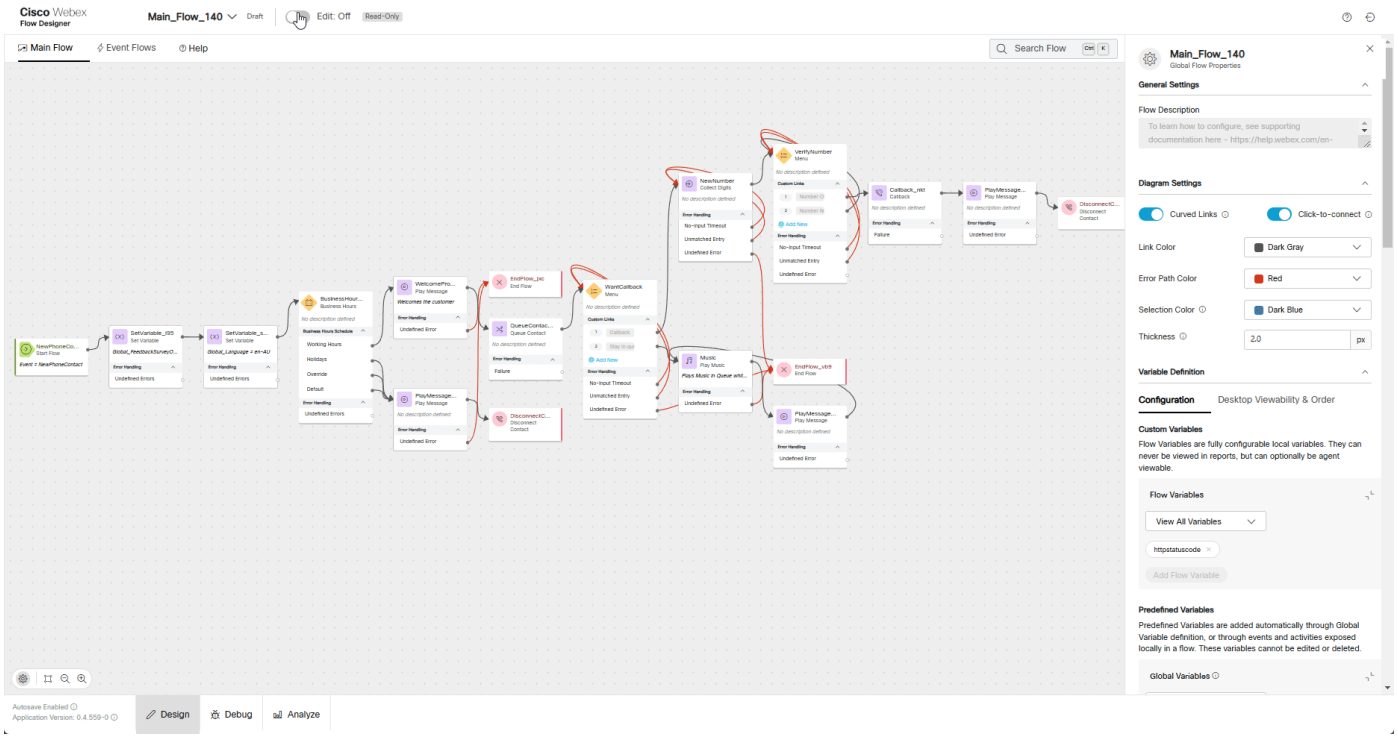
  Default Value: leave it empty

- Search Result variable:

  Name: **searchresult**

  Type: **String**

  Default Value: leave it empty



3. Add an **HTTP Request** node for the Search API query by following this order of steps:

   Remove the existing connection between **VerifyNumber** Option 1 ("Number OK") and **Callback** node

   Add an **HTTP Request** node and connect **VerifyNumber** Option 1 ("Number OK") to that node

   We will connect this **HTTP Request** node in next step

   Activity Label: **HTTPRequest_CallBackSearch**

   Select **Use Authenticated Endpoint**

   Connector: **WxCC_API**

   Path: **/search**

   Method: **POST**

   Content Type: **GraphQL**

   Copy this GraphQL query into the **Query** field of the **Request Body**:

```
query callbackSearch($from: Long!, $to: Long!, $filter: TaskDetailsFilters) {
taskDetails(from: $from, to: $to, filter: $filter) {
  tasks {
    callbackData {
      callbackRequestTime
      callbackConnectTime
      callbackNumber
      callbackStatus
      callbackOrigin
      callbackType
    }
    lastEntryPoint {
      id
      name
      }
    }
  }
}
```

   Copy the following variables JSON into the **GraphQL variables** of the **Request Body**:

```
{
  "from": "{{now() | epoch(inMillis=true) - 15000000}}",
  "to": "{{now() | epoch(inMillis=true)}}",
  "filter": {
    "and": [
      {
        "callbackData": {
          "equals": {
            "callbackNumber": "{{NewNumber.DigitsEntered}}"
```

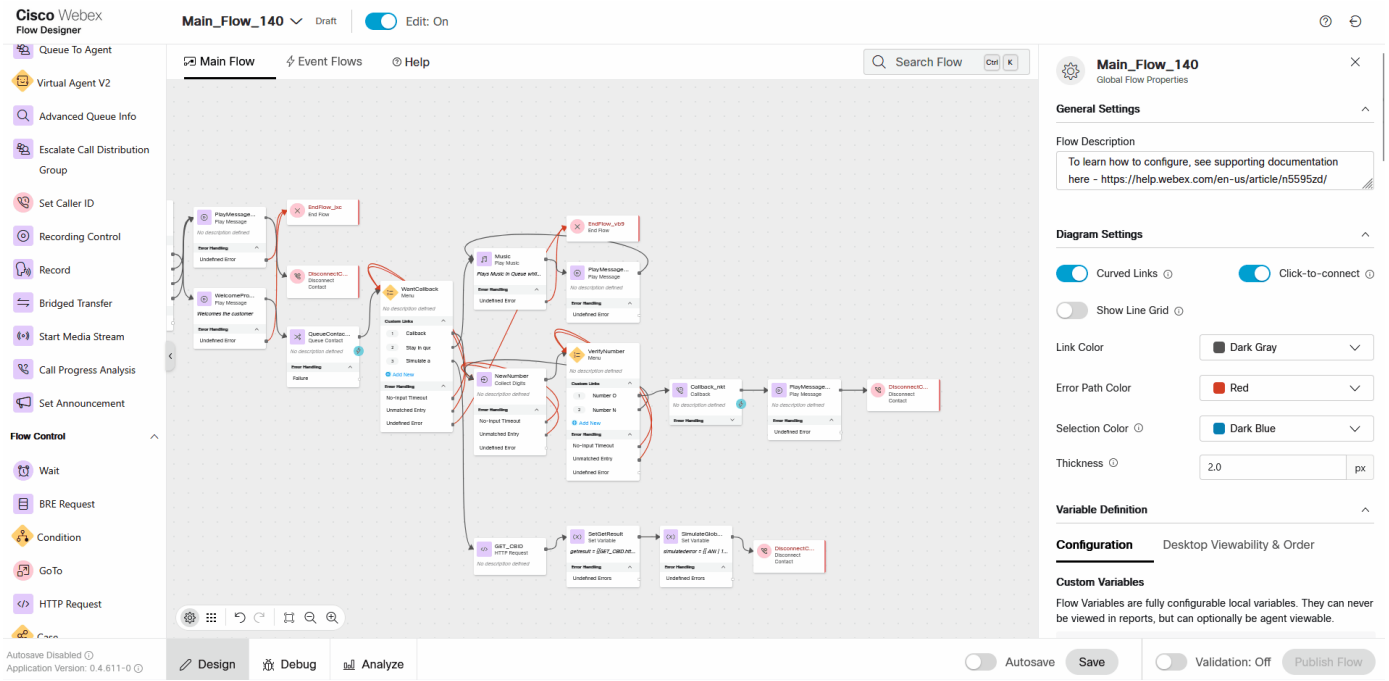
```
          }
        }
      },
      {
        "lastEntryPoint": {
          "id": {
            "equals": "{{NewPhoneContact.EntryPointId}}"
          }
        }
      }
    ]
  }
}
```

Parse Settings:

- Content Type: JSON
- Output Variable: `callbackStatus`
- Path Expression: `$.data.taskDetails.tasks[0].callbackData.callbackStatus`

Click **+ Add New**

- Output Variable: `callbackConnectTime`
- Path Expression: `$.data.taskDetails.tasks[0].callbackData.callbackConnectTime`



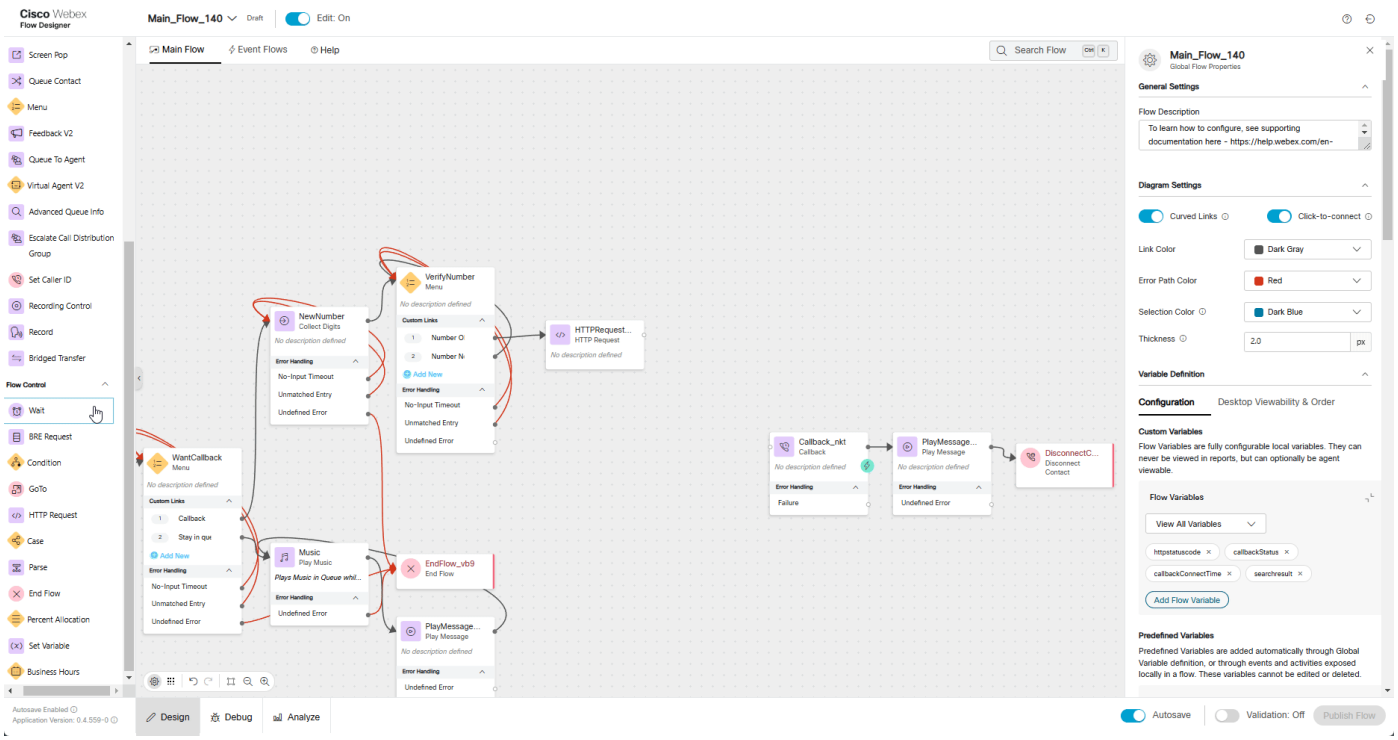
4. Add **Set Veriable** node

   Connect **HTTPRequest_CallBackSearch** to this node

   We will connct **Set Variable** node in next step

   Variable: **searchresult**

   Set To Variable: **HTTPRequest_CallBackSearch.httpResponseBody**



5. Add a **Condition** node

   Connect **Set Variable** created in previous step to this node

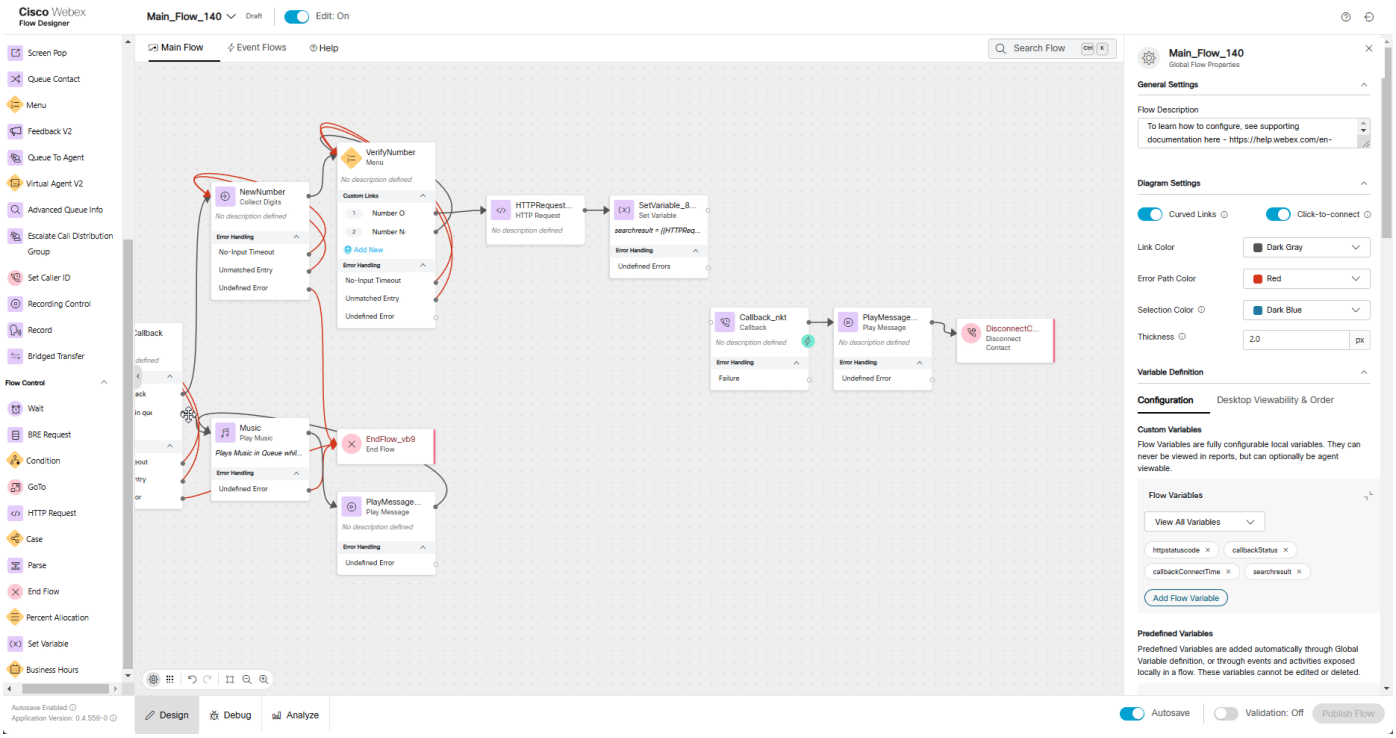
   Connect **False** exit path to existing CallBack node

   We will connect **True** exit path in next step

   Expression:

```
{{ callbackConnectTime == "-1" ? (callbackStatus == "Not Processed" ? (HTTPRequest_CallBackSearch.httpStatusCode == 200 ? "true" : "false") : "false") :
"false" }}
```

> **Note**
>
> Above expression uses nested ternary logic to combine the checks. This evaluates the first condition and then evaluates the second condition if the first is true and so on. In our case the expression returns True only when httpStatusCode equals **200**, callbackStatus is **Not Processed** and callbackConnectTime is **-1**



6. Add **Play Message** nodes:

   Enable Text-To-Speech

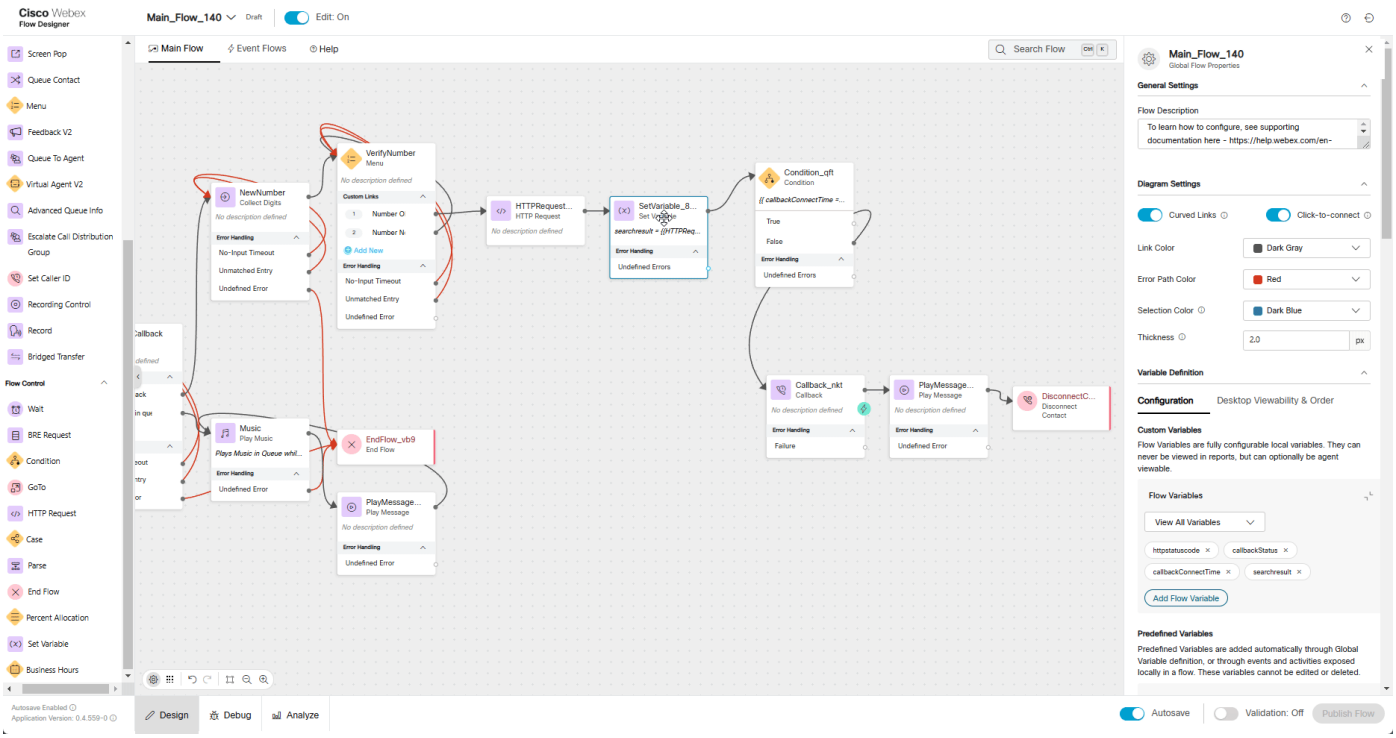
   Select the Connector: **Cisco Cloud Text-to-Speech**

   Click the **Add Text-to-Speech Message** button and paste text: **The callback for provided number has been scheduled already. Please await for a callback once next agent becomes available. Thank you for your patience.**

   Delete the selection for Audio File

   Connect **True** exit path of **Condition** node created in previous step to this **Play Message** node

7. Add **Disconnect Contact** and connect the exit path of the **Play Message** created in previous step to this node.



8. Validate and publish the flow:

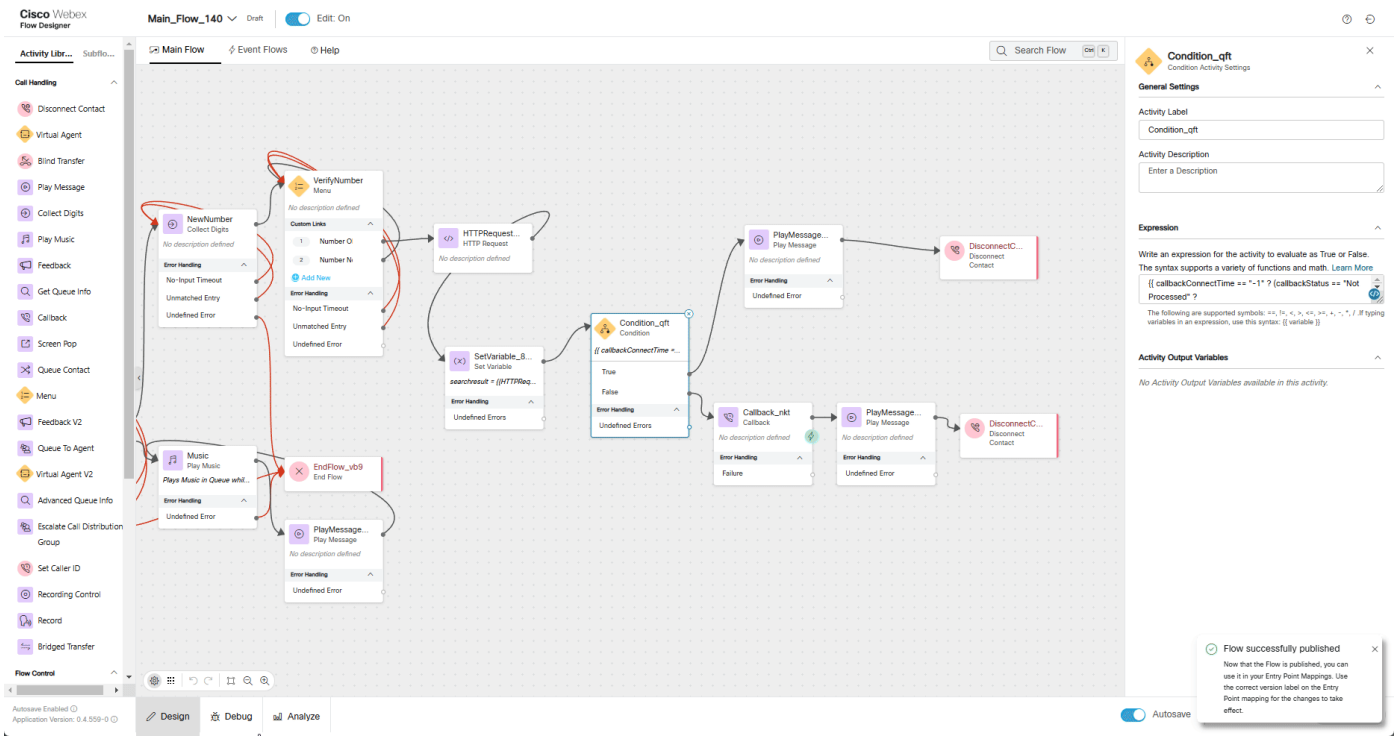
   Enable the **Validation** toggle in the bottom right corner of the flow designer window to check for any potential flow errors and recommendations.

   If there are no **Flow Errors** after validation is complete, click on **Publish Flow** next to it.

   In the pop-up window, ensure that the **Latest** label is selected in the **Add Version Label(s)** list, then click **Publish Flow**.

Testing

1. Make sure your Agent in **Not Available** state (select any Idle state). In this case call will not be assigned to an agent and callback will be proposed to a caller.
2. Make a call to your Support Number and if success you should hear configured messages and ask to press "1", "2" or "3". Press "1" to schedule a callback and provide a number for a callback. Because in current lab we are having number limitations we are going to provide a well known Cisco Worldwide Technical Support contact number **1 408 526 7209** .
3. While keeping your agent **Not Available**, make another test call to your flow, press "1" and request for another callback to the same number **1 408 526 7209** .
4. You should hear a message configured in **Step 6** of the current mission.
5. Click on **Analyze** to visually observe the call flow. Make sure you're viewing latest Published Version.
6. Review the flow and click on **HTTPRequest_CallBackSearch**. You will see the list of interactions at the bottom of the **Analyze** window. Find the last interaction (the top in the list) and click on **View interaction in debugger** link at the right-hand side of the interaction line. This will cross-launch Debugger to that particular call. Flow Debugger will be opened in a new browser tab.
7. Navigate to **HTTPRequest_CallBackSearch** Activity Name to see **Modified Variables** at the bottom of right-hand side of the debugger.
8. Click on **Set Variable**, which is the next step after **HTTPRequest_CallBackSearch**, to see full Search API response which we wrote to **searchresult** flow variable on the **Step 6** of the current mission configuration.



9. On Webex Desktop, make your agent **Available**. Webex Contact Center will reserve your agent right away and propose to answer a callback call.
10. Answer the call and wait until you are connected to a Cisco Technical Support IVR and hear a welcome prompt. Then disconnect the call in agent desktop to clean up pending callback requests.

Congratulations, you have succesfully completed Preventing Callback Duplication mission! 🎉

4.1.5 Mission 5: Adding Scheduled IVR Callback [To Verify]

Story

You are designing a customer-friendly callback experience for a busy customer contact center. During peak hours, callers often face long wait times and prefer not to stay on hold. Instead, they want control over when they receive a return call.

In this lab, you will build a Scheduled IVR Callback flow that allows callers to request a callback at a date and time of their choosing. Rather than waiting in queue, the caller interacts with the IVR, selects a preferred callback slot, and disconnects—confident that the system will call them back as requested.

Call Flow Overview

1. A new call enters the flow.
2. The flow executes the logic configured in previous steps.
3. The call is routed to the appropriate queue, but no agents are available.
4. Since no agents are available, a callback option is offered to the caller.
5. Caller provides prefered date and time for a Callback /br>
6. When time comes and agent becomes available, the callback is initiated to the provided number.

Mission Details

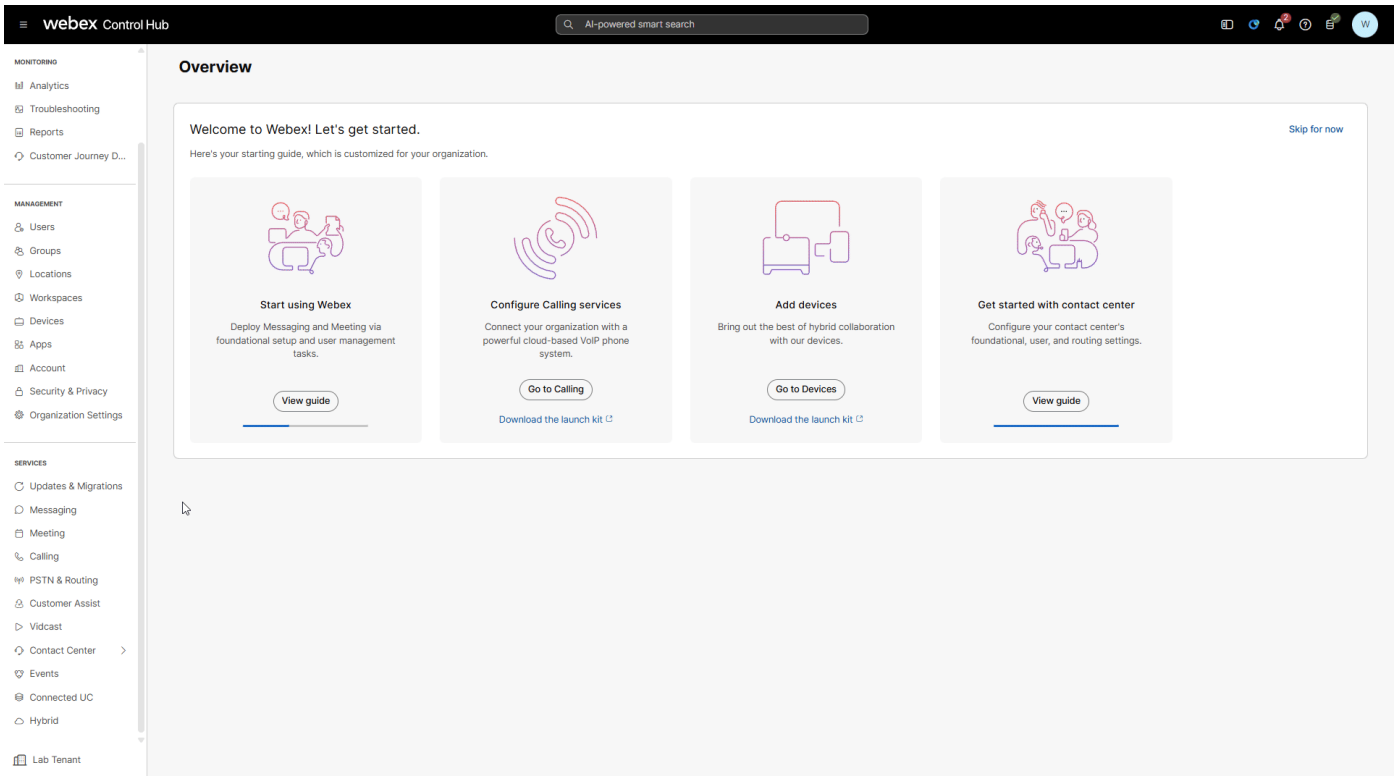
Your mission is to:

1. At this stage, if you have completed either of the Core or Callback tracks so far, your Main Flow has likely grown significantly. To simplify navigation, we will create a new flow dedicated specifically to the scheduled IVR callback.
2. Use preconfigured subflow **Scheduled_CallbackSubflow** . Existing **Scheduled_CallbackSubflow** must be used without any modifications. This subflow is shared across all lab attendees and should remain unchanged.

> **heduled Callback Subflow details. [Optional]**
>
> **Scheduled_CallbackSubflow** has already been preconfigured for you. However, the steps below explain how this subflow can be created and configured manually for reference.
>
> 1. Switch to Control Hub, then navigate to **Flows**, open the **Manage Flows** drop-down list, and select **Create Flows**.
> 2. A new tab will open. Navigate to **Flow Templates**.
> 3. Switch to **Subflow**, then click on **Subflow Templates**.
> 4. Select **Register Scheduled Callback Subflow** and click **View Details**.
> 5. The template details window provides detailed information about the subflow structure, its description, and other useful insights into the subflow logic.

Build

1. Switch to Control Hub, then navigate to **Flows**, click on **Manage Flows** dropdown list and select **Create Flows**
2. New tab will be opened. Navigate to **Flow Templates**
3. Choose **Simple Inbound Call to Queue** and click **Next**. You can open **View Details** and to see observe flow structure and read flow description.
4. Name you flow as **Scheduled_IVR_CallBack_Your_Attendee_ID** . Then click on Create Flow.



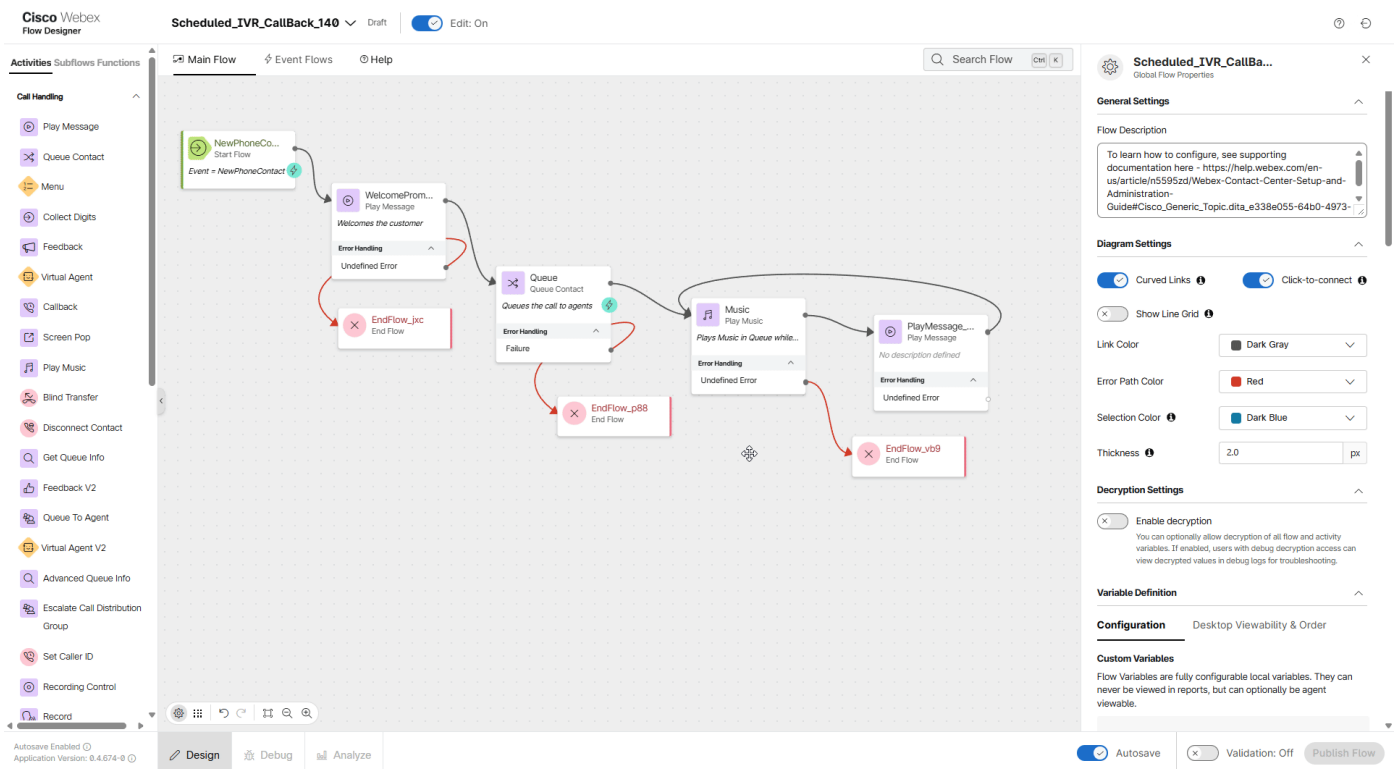
5. On the right-hand side, in the **Global Flow Properties** panel, scroll down to locate the **Flow Variables** section under **Custom Variables**. Click the **Add Flow Variable** button and add the following 3 flow variables:

- Callback Number variable:

  Name: **callbackNumber**

  Type: **String**

  Default Value: **empty**



6. Select **Queue** node. On the **General settings** keep Static Queue checked and select queue **Your_Attendee_ID_Queue** from the drop down list.
7. Remove the **Music** node, then move the **PlayMessage** and **End** nodes to the right to create space for additional nodes that will be added in the next steps.
8. Drag **Collect Digits** nodes

   Rename Activity Label to **NewCallBackNumber**

   Enable Text-To-Speech

   Select the Connector: **Cisco Cloud Text-to-Speech**

   Click the Add Text-to-Speech Message button and paste text: ***All agents are currently busy. Please enter your 11 digits phone number to which we should call you back.***

   Delete the selection for Audio File

   Advanced Settings:

   No-Input Timeout: **5**

   Make Prompt Interruptible: **True**

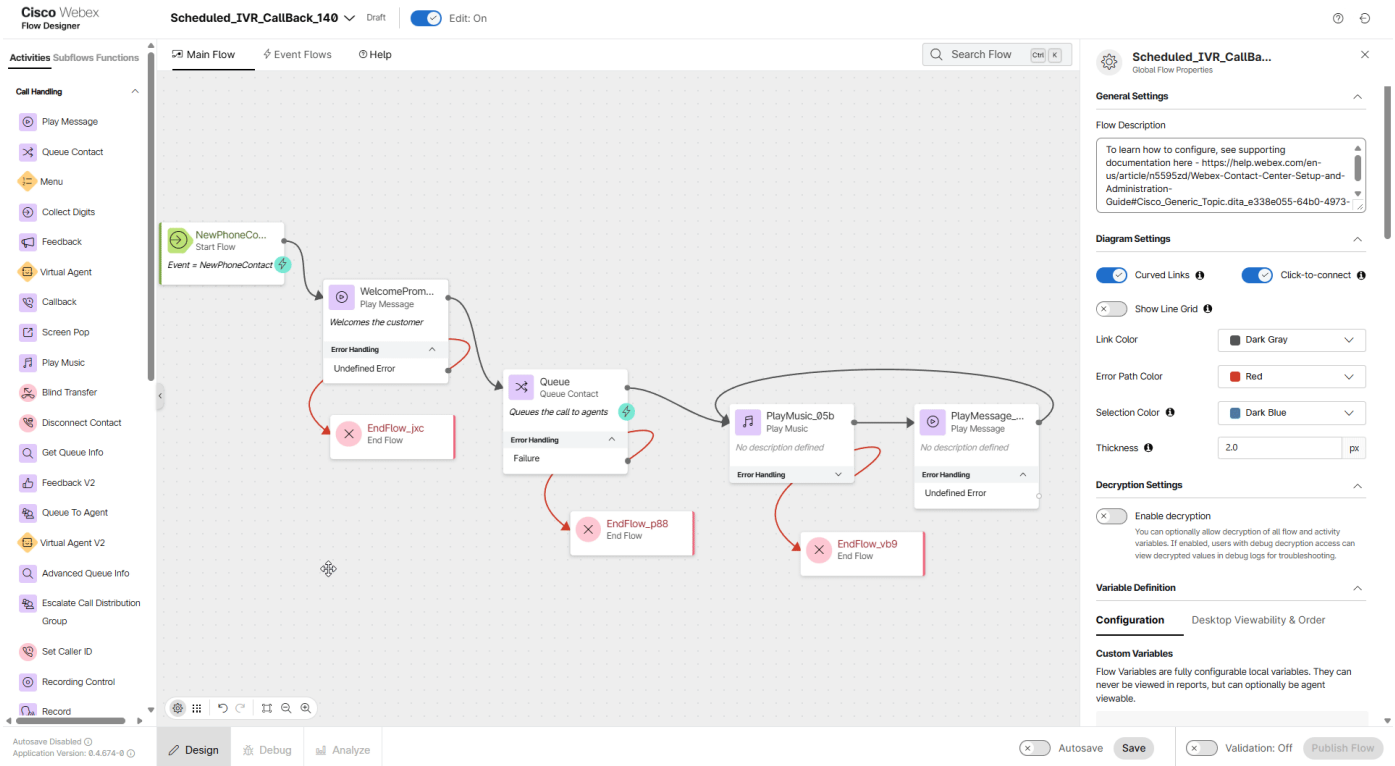
   Minimum Digits: **11**

   Maximum Digits: **11**

   Connect **No-Input Timeout** to the front of the **NewCallBackNumber** node

   Connect **Unmatched Entry** to the front of the **NewCallBackNumber** node

   Connect **Undefined Error** to the front of the **NewCallBackNumber** node



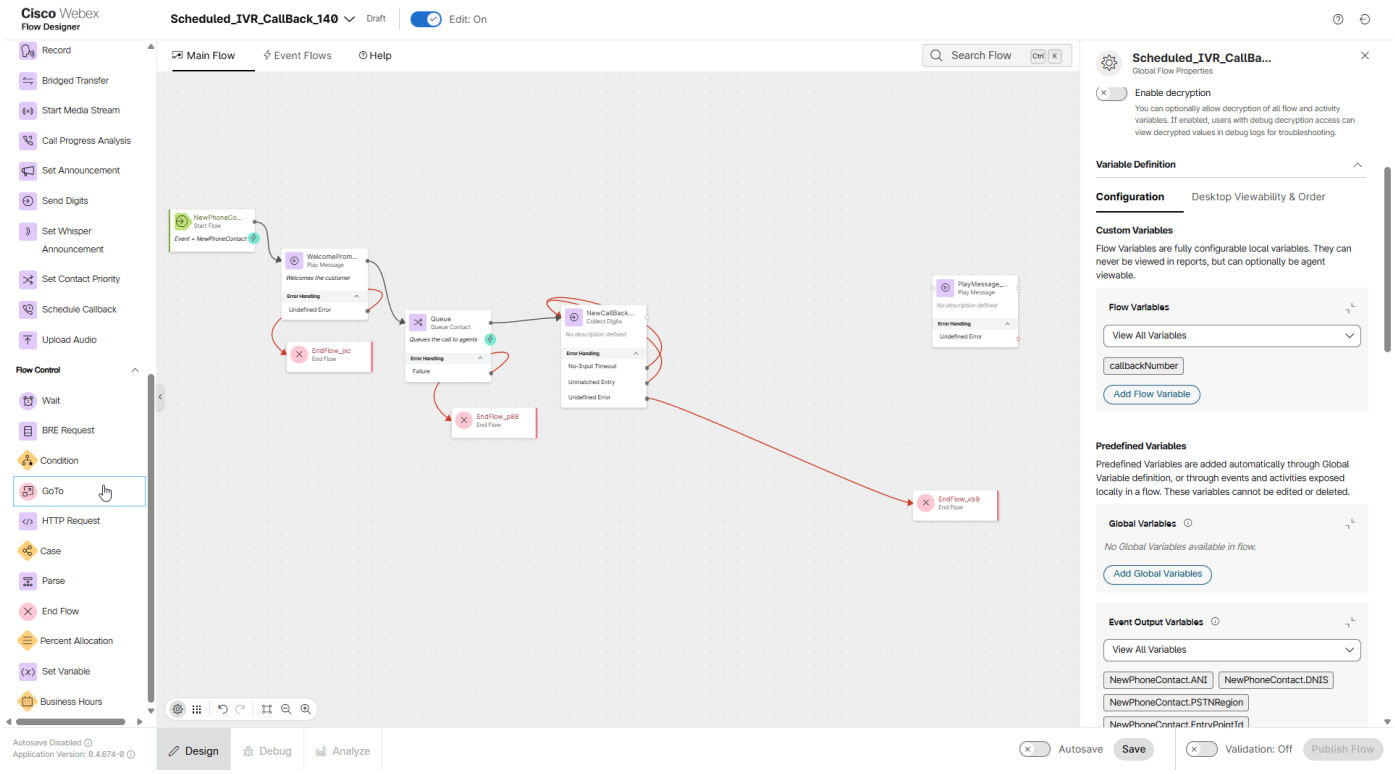
9. Add **Set Veriable** node.

   Connect **NewCallBackNumber** to this node

   We will connct **Set Variable** node in next step

   Variable: **callbackNumber**

   Set To Variable: **+{{NewCallBackNumber.DigitsEntered}}**



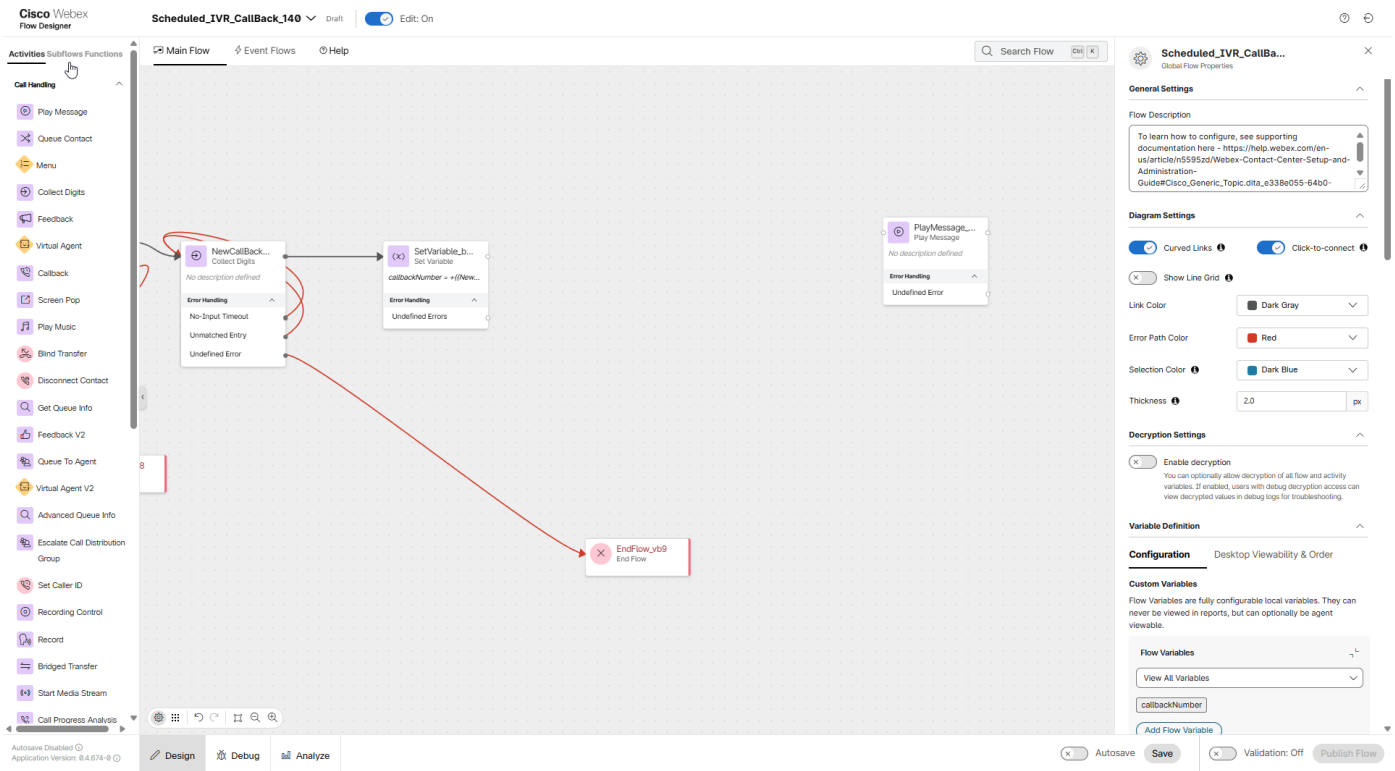
10. Switch to **Subflows** tab in the left menu. Then drag **Scheduled_CallBack_Subflow** node to the canvas.

    Connect **Set Veriable** to this node

    Connect this node to **Play Message** node

    Connect **Undefined Error** to any of available **EndFlow** nodes.

    Optionally, you can view this preconfigured subflow by clicking the **View** button in the top-right corner of the node settings, next to the node name. **This subflow is preconfigured and must not be modified. Close the subflow tab after viewing it.**



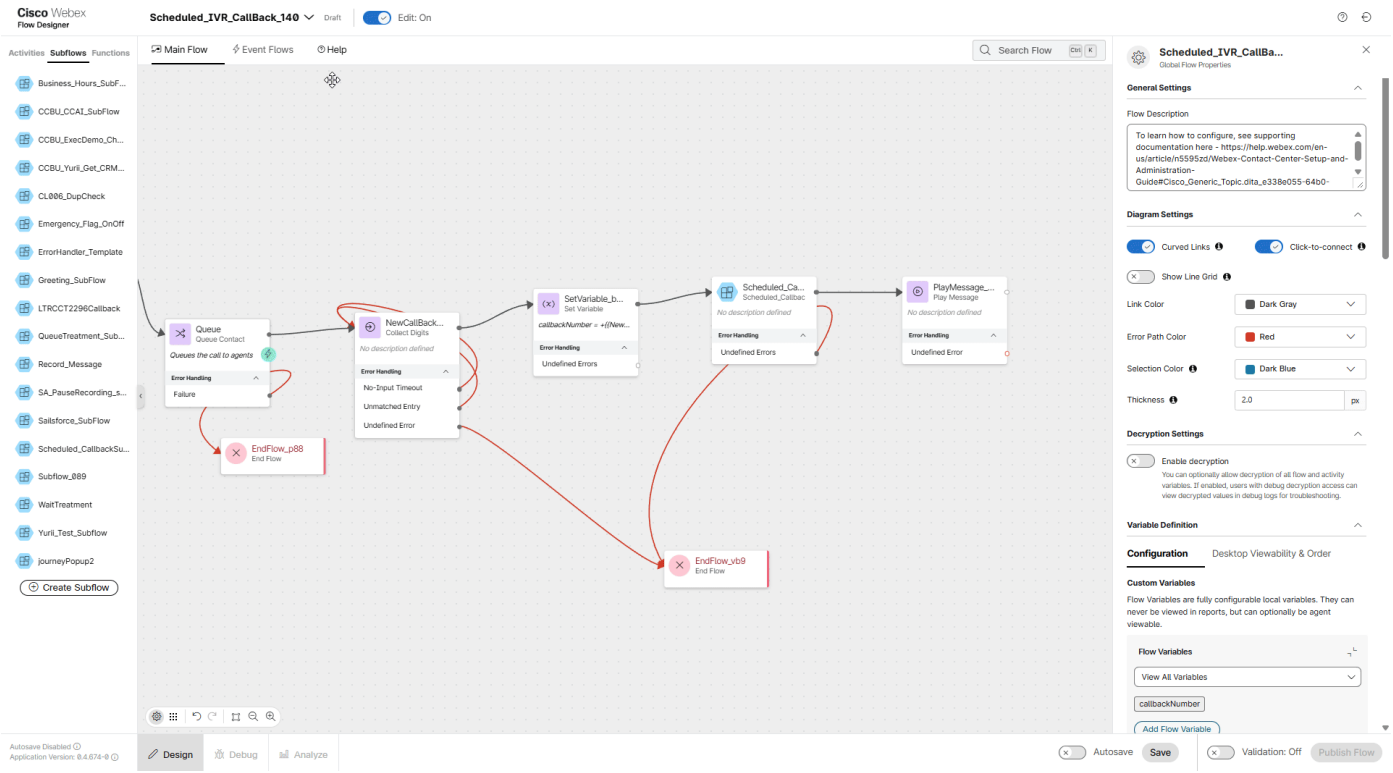
11. Click on the **Scheduled_CallBack_Subflow** node. Go to the node settings pane on the right and set the following parameters:

    Subflow Label: **Latest**

    Scroll down to the **Subflow Input Variables** section and configure the following mapping by pressing **Add New** button for every variable:

- Current flow variable: **callbackNumber**
- Subflow Input Variable: **CallbackNumber**

- Current flow variable: **Queue.QueueId**
- Subflow Input Variable: **CallbackQueue**

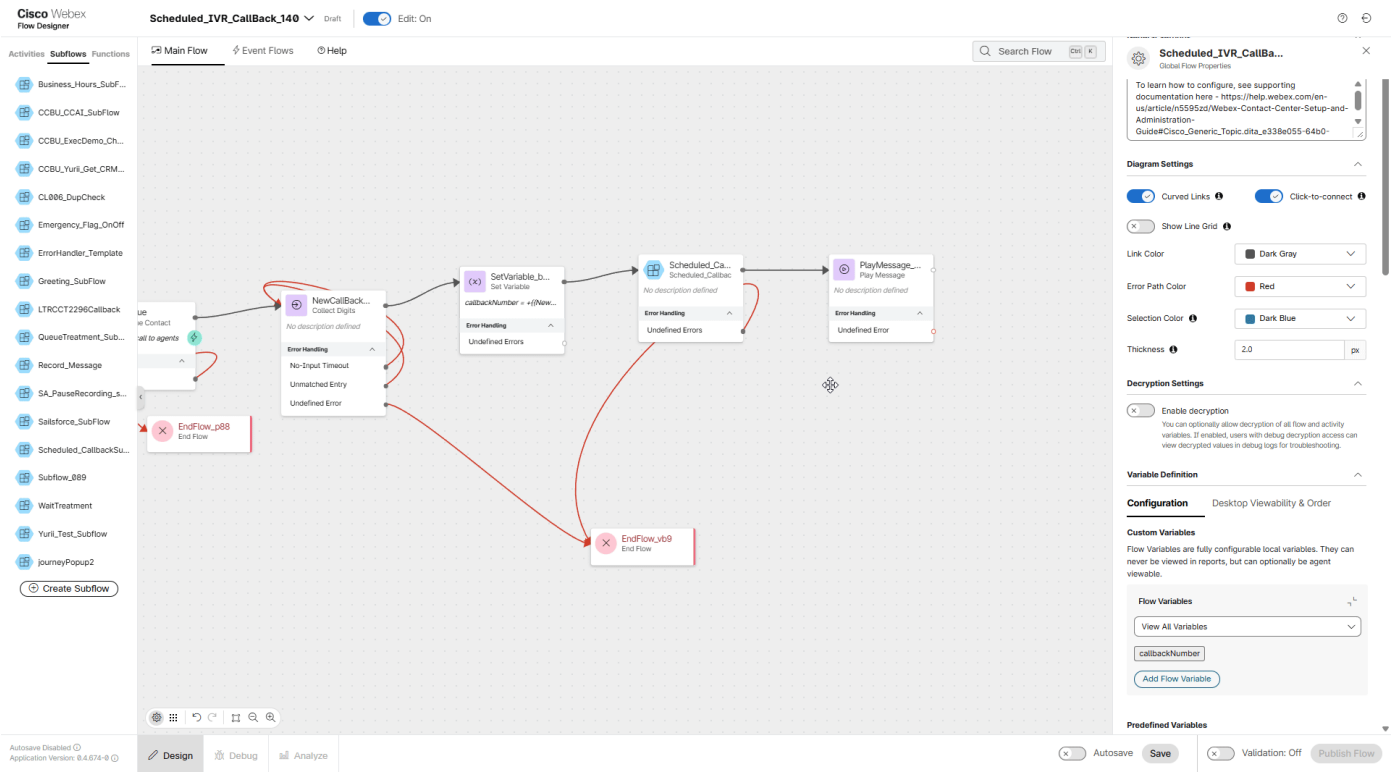


12. Click on **Play Message** modify Text-to-Speech Message by replacing existing text to the following: ***Your callback has been successfully scheduled.***

13. Add **Disconnect Contact** and node:

    Connect **Play Message** to **Disconnect Contact** node

14. Validate the flow by clicking **Validate**, **Publish** and select the Latest version of the flow.



15. Return back to Control Hub to assign the Flow to your **Channel (Entry Point)**. Go to **Channels**, search for your channel **Your_Attendee_ID_Channel**
16. Click on **Your_Attendee_ID_Channel**
17. In **Entry Point** settings section change the following, then click **Save** button:

- Routing flow: **Scheduled_IVR_CallBack_Your_Attendee_ID**
- Version label: **Latest**

Testing

1. Ensure that your agent is either **Logged Out** or set to **Not Available**. In this state, the call will not be assigned to an agent, and the system will offer the callback option to the caller.
2. With your agent remaining in **Not Available** place a test call to your support number.
3. To successfully schedule a callback, you must provide the following information when prompted using your phone's dial pad:

   a. **An 11 digit phone number.** This can be your personal mobile number or a known Cisco Worldwide Support number, such as **1 408 526 7209**. Use the dial pad to enter the Cisco TAC number.

   b. Preferred date in ***YYYYMMDD*** format. Example: Enter DialPad enter ***20260212***.

   c. Preferred start time for your callback in ***HHMM*** format. Selected time should be at least 30 minutes from now..

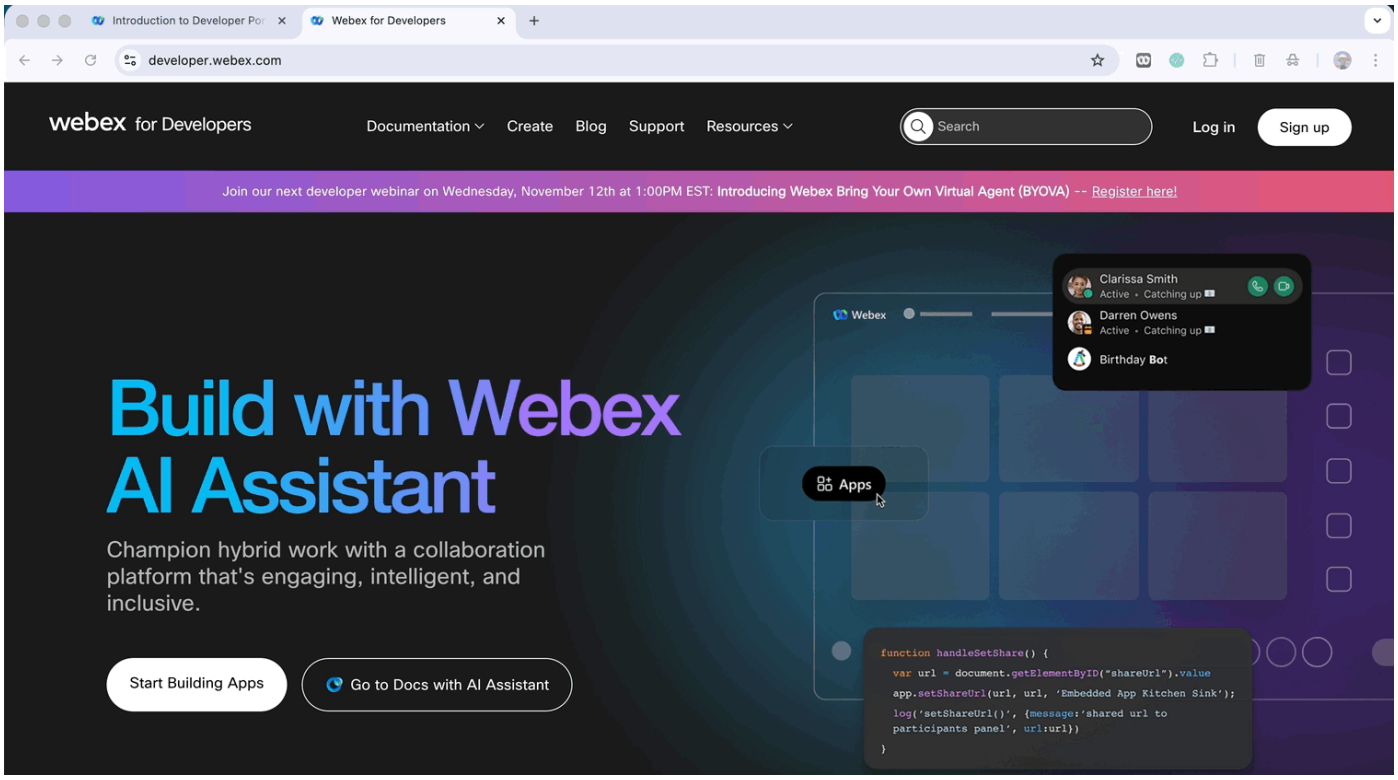
   d. Preferred end time for your callback in ***HHMM*** format. The call between window must be at least 30 minutes and no more than 8 hours.

   e. **Callback timezone.**. For EMEA press **1**.
4. After providing all inputs, you will hear a confirmation message indicating that your callback has been successfully scheduled.
5. To receive the callback, ensure you set your agent desktop to **Available** during the scheduled time window.

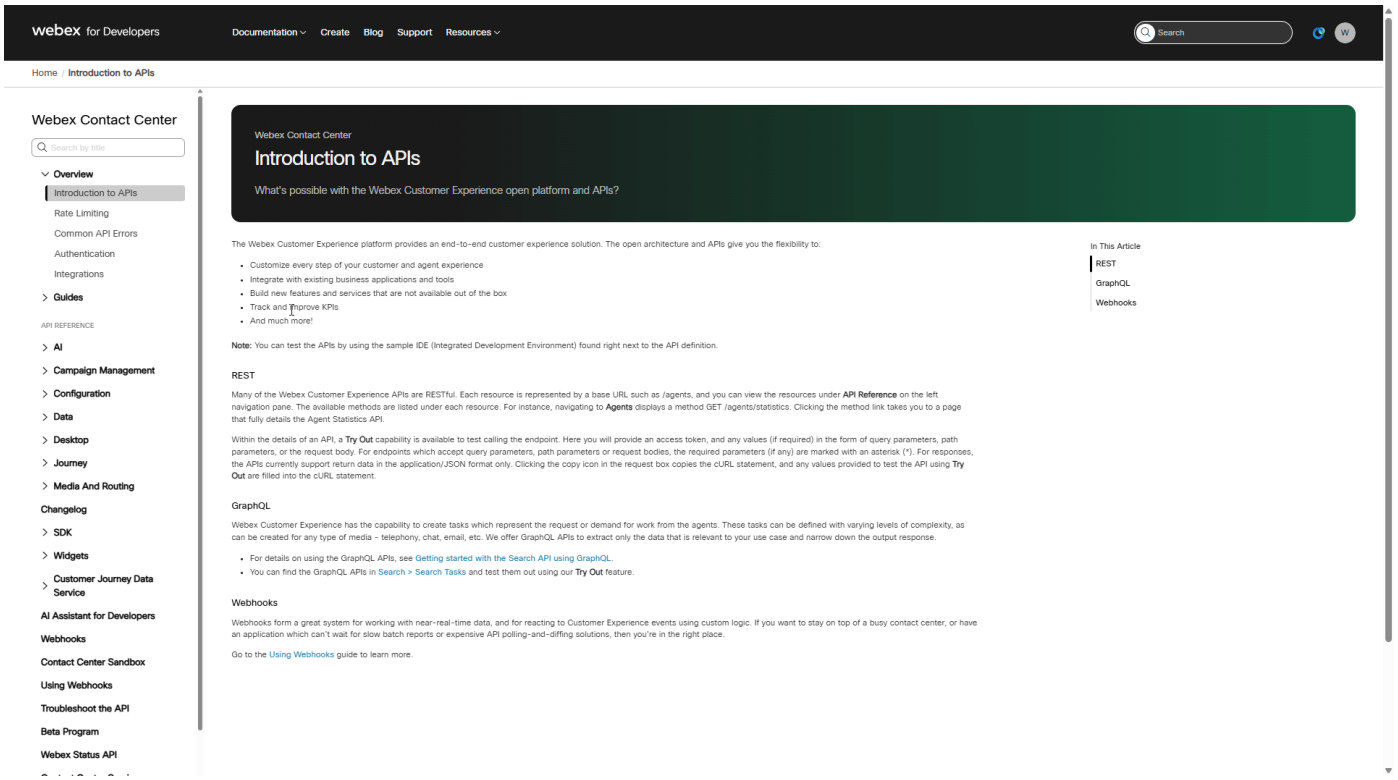
> **Note**
>
> You may proceed with other tasks without waiting for the callback time. When the time comes, please remember to make yourself available to accept the call.

Callback status verification.

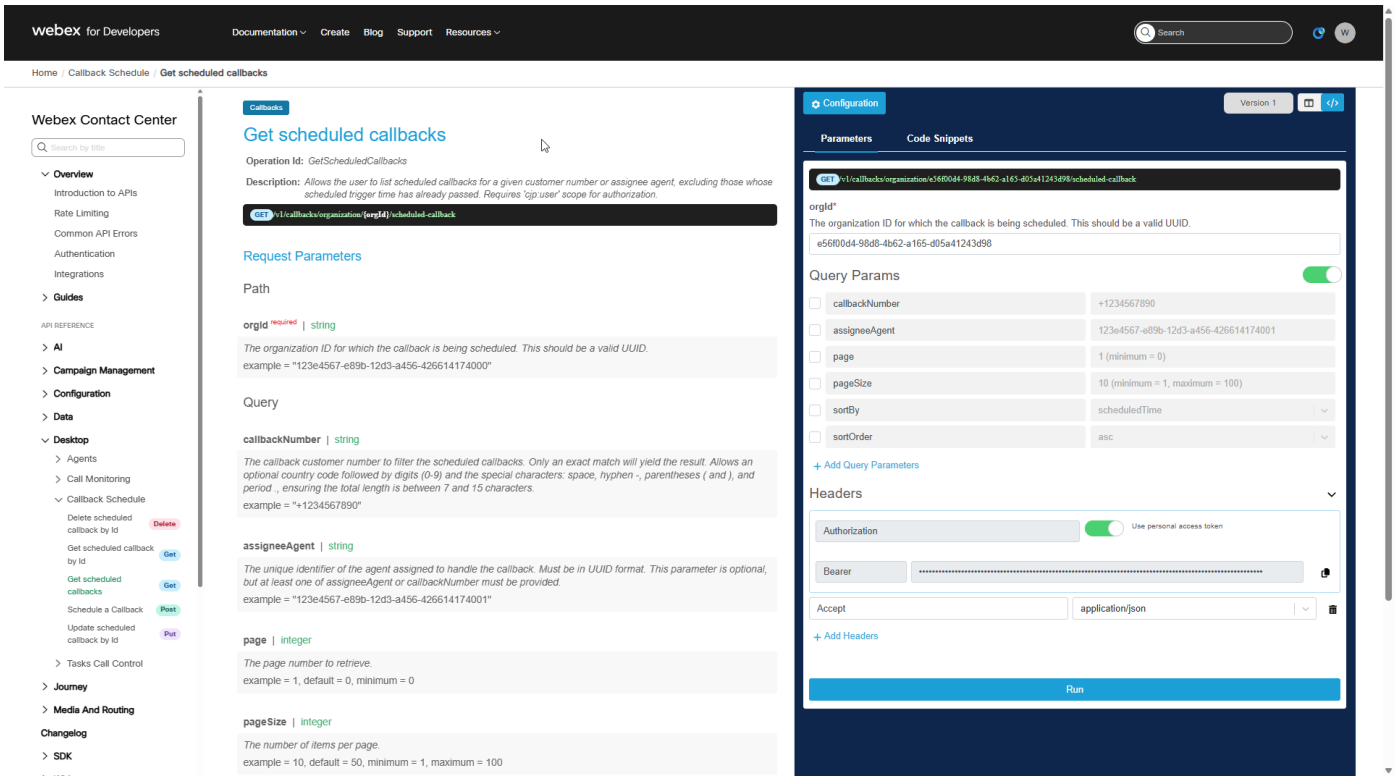
1. Open **Developer Portal** and click on **Log In**. Your login will be of the format **wxcclabs+admin_IDYour_Attendee_ID@gmail.com**. You will see another login screen with Webex icon on it where you may need to enter the email address again and the password provided to you.
2. Click on the little arrow next to **Documentation**, choose **Webex Contact Center** under **Customer Experience** section.



3. On Menu pannel on the left, scroll down to **API Reference** section, expand **Desktop** and then expand **CallBack Schedule**
4. Click on **Get scheduled callbacks** to open an endpand description page.



5. On the right hand side under **Query Params** set a checkbox next to ***callbackNumber***, then type 11 digit number you provided while were doing the test call.
6. Then click **Run**.



7. Verify output of the executed API call. Observe the important keys:

```
{
    "id": "3824bcea-03c7-41b8-957d-5d62ecda3b82",          // Unique identifier for the scheduled callback.
    "customerName": "+48575638602",                        // Customer Name. Uses Callback Number if not provided specifically
    "callbackNumber": "+48575638602",                      // Phone number provided for the callback.
    "timezone": "Europe/Amsterdam",                        // Timezone in which the callback is scheduled.
```

```
    "scheduleDate": "2026-01-19",                       // Date for the callback in ISO format (YYYY-MM-DD).
    "startTime": "18:45:00",                            // Scheduled start time in ISO 8601 format (HH:mm:ss), local to the specified timezone.
    "endTime": "19:20:00",                              // Scheduled end time in ISO 8601 format (HH:mm:ss), local to the specified timezone.
    "queueId": "ee46583c-8d0d-4c09-8829-8c0b79c11a79"   // Identifier for the queue to route the callback request.
    //<ommitted>
}
```

Full Schema Definition can be found in the **API Reference** for this API call.

Congratulations on completing another mission.

4.1.6 Mission 6: Adding Personalized Callback **[To Verify]**

> **Note**
>
> This mission does not involve Flow Designer configuration or inbound call handling. While it is not a traditional flow-building task, it is included to ensure that attendees completing the Callback track are fully aware of all callback options available in Webex Contact Center—including agent-initiated scheduled callbacks from the Desktop.

Story

In many real-world scenarios, agents agree with customers on a specific date and time for a follow-up call—for example, after investigating a case, waiting for external confirmation, or aligning with the customer's availability.

Instead of asking customers to call back or keeping reminders outside the platform, agents can schedule a callback directly from the Webex Contact Center Desktop. The system takes care of placing the outbound call at the agreed time, ensuring consistency, reliability, and a better customer experience.

In this lab, you will learn how to schedule a callback from the agent desktop, without any inbound call or flow changes, demonstrating a powerful but often overlooked capability of Webex Contact Center.

Call Flow Overview

> **Note**
>
> An agent can schedule a call with a customer at any time—whether they are on an active call or in a Not Available state.

1. An agent is working in the Webex Contact Center Desktop.
2. The agent agrees with a customer on a specific date and time for a follow-up call.
3. Using the Desktop interface, the agent schedules a callback for the selected time.
4. The customer disconnects, confident that the callback will be handled automatically. *(If on active call with a customer)*
5. At the scheduled time, when an agent becomes available, Webex Contact Center initiates the outbound callback to the specified number.

Mission Details

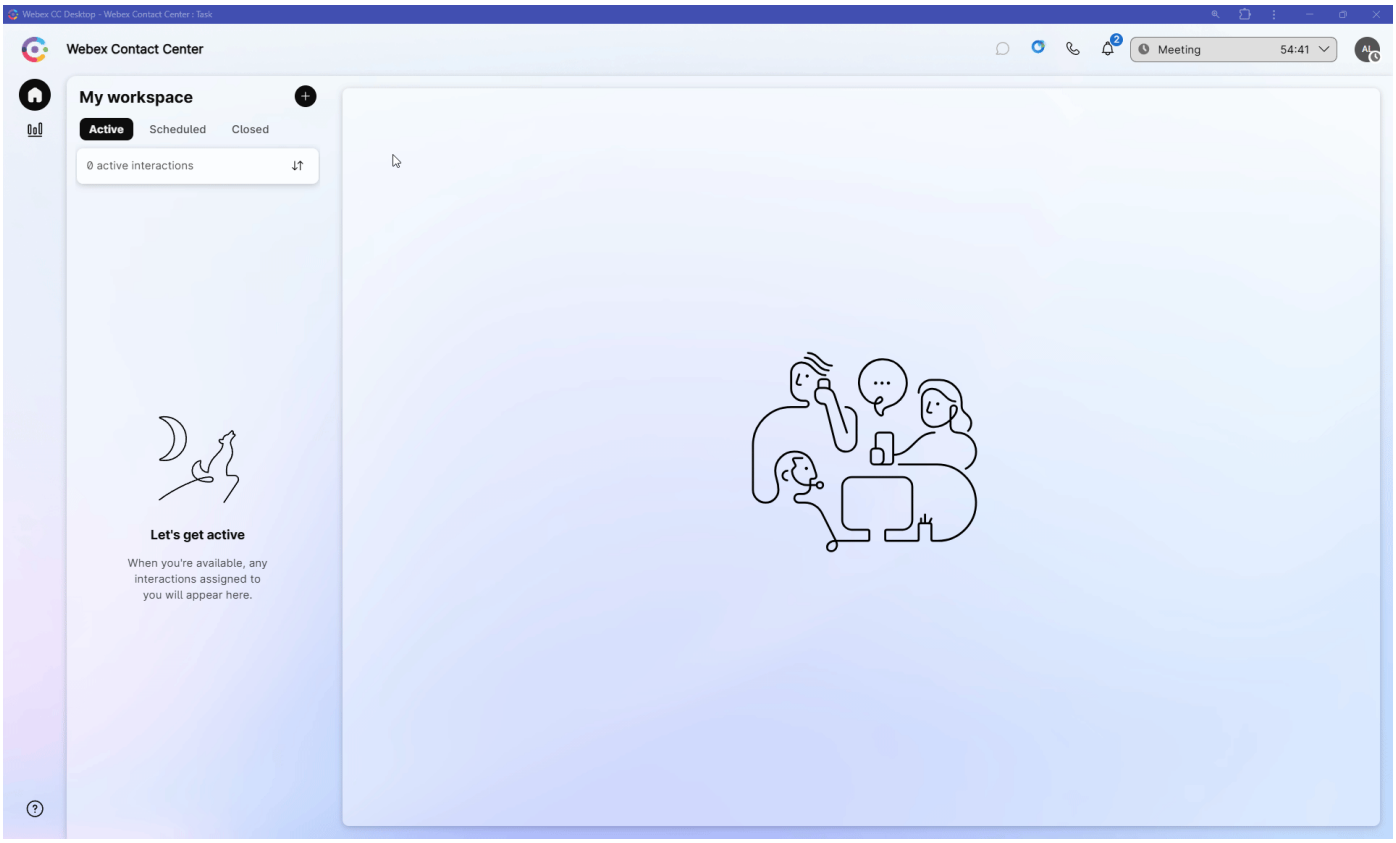
Your mission is to:

1. Understand how scheduled callbacks can be created directly from the Webex Contact Center Desktop, without relying on inbound calls or IVR flows.
2. Learn the agent-side workflow for scheduling a callback at a specific date and time.

Testing

Your Agent desktop session should be still active but if not, use Webex CC Desktop application  and login with agent credentials you have been provided **wxcclabs+agent_IDYour_Attendee_ID@gmail.com** . You will see another login screen where you may need to enter the email address again and the password provided to you.

1. With your agent remaining in **Not Available** place a test call to your support n
2. Click on the little plus sign next to **My Workstation** on top left corner, choose **Schedule a callback** from drop down menu.



3. To successfully schedule a callback, you must provide the following information on pop up window:

   a. **Phone number (required)**. An 11 digit phone number. This can be your personal mobile number or a known Cisco Worldwide Support number, such as **1 408 526 7209**. Use the dial pad to enter the Cisco TAC number.

   b. **Customer name (required)**. Type **Attendee_IDYour_Attendee_ID**

   c. **Date (required)**. Choose todays date. *(You can schedule a callback up to 31 days in advance. The call between window must be at least 40 minutes and no more than 8 hours.)*

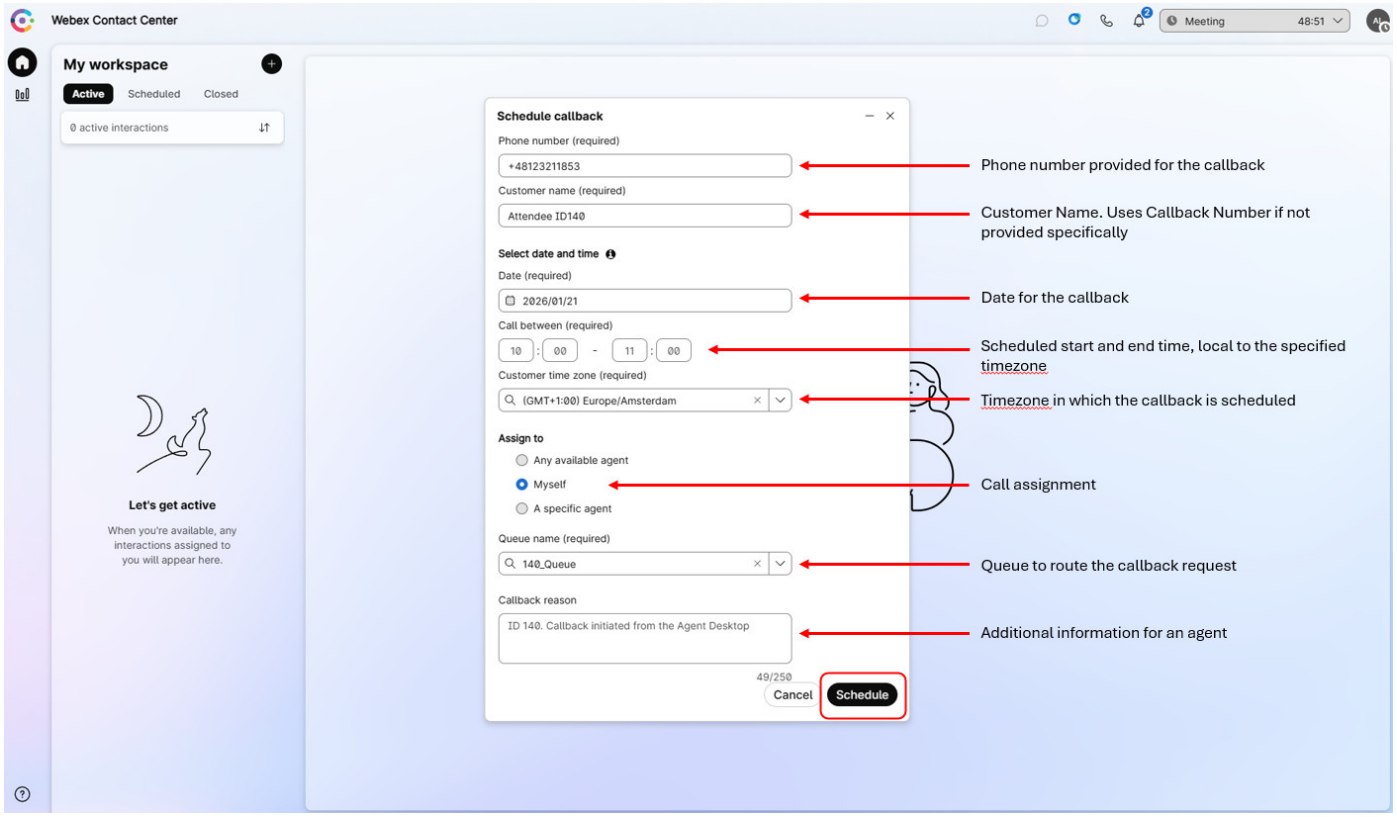
   c. **Call between (required)**. Selected time should be at least 30 minutes from now..

   d. **Customer time zone (required)**. Select **Europe/Amsterdam**

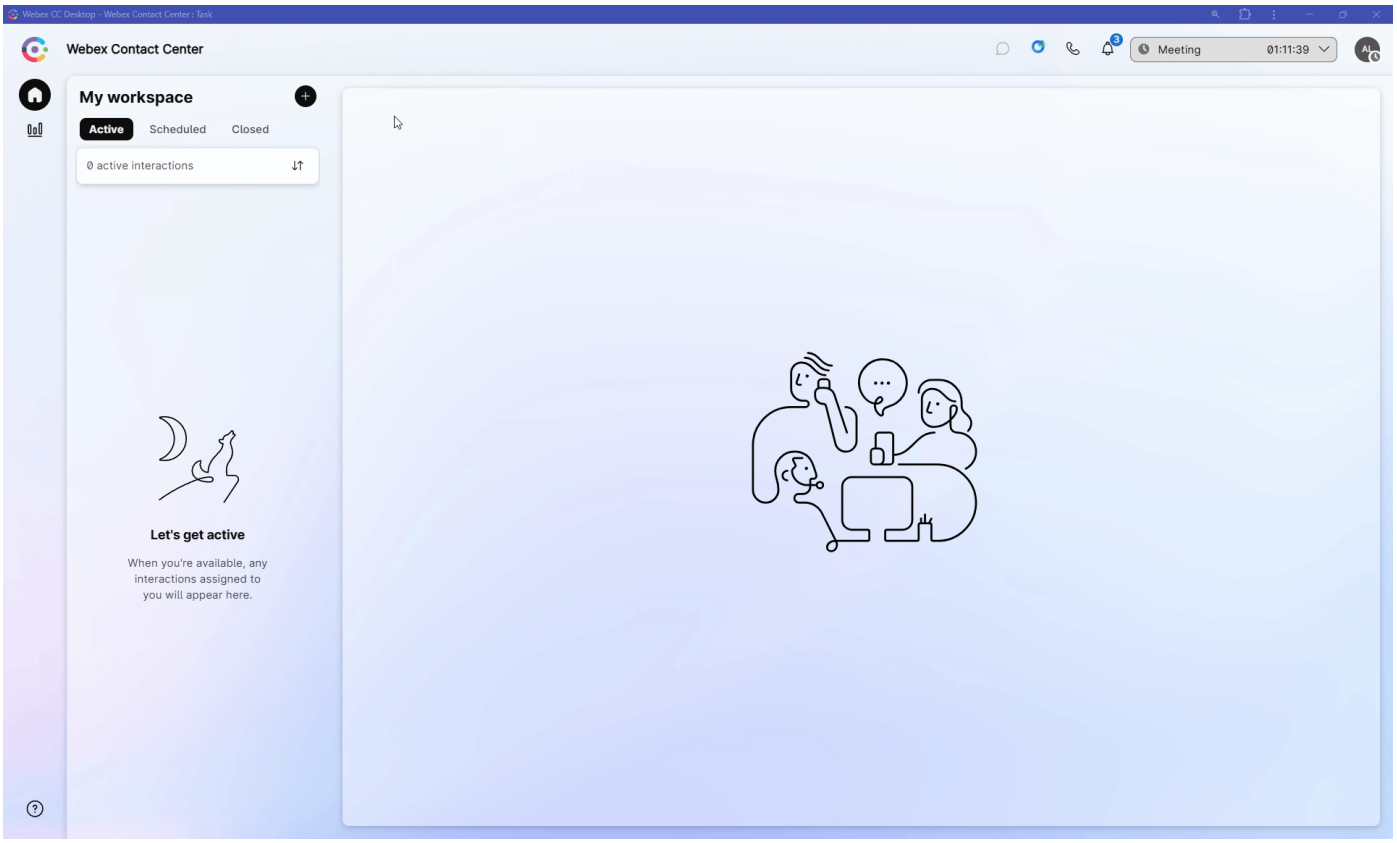
   e. **Assign to**. Choose **Myself**.

   f. **Queue name**. Provide your **Your_Attendee_ID_Queue**

   g. **Callback Reason**. Type **"*Attendee ID 140. Callback initiated from the Agent Desktop.*"**



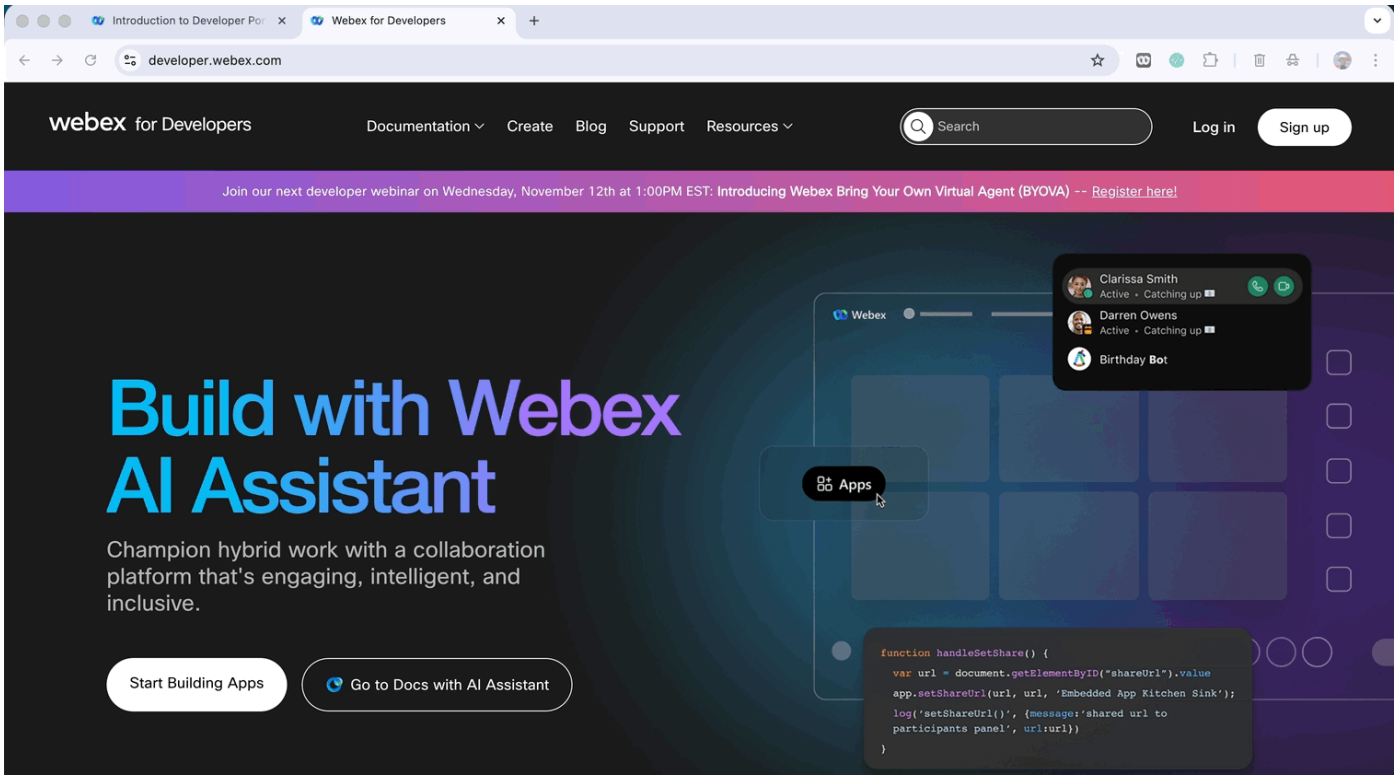
4. After providing all inputs, press **Schedule** button. You should see a pop up notification tha **Callback scheduled successfully**.
5. To receive the callback, ensure you set your agent desktop to **Available** during the scheduled time window.
6. You can verify, delete or edit your scheduled callbacks by clicking on Schedule tab on agent desktop.



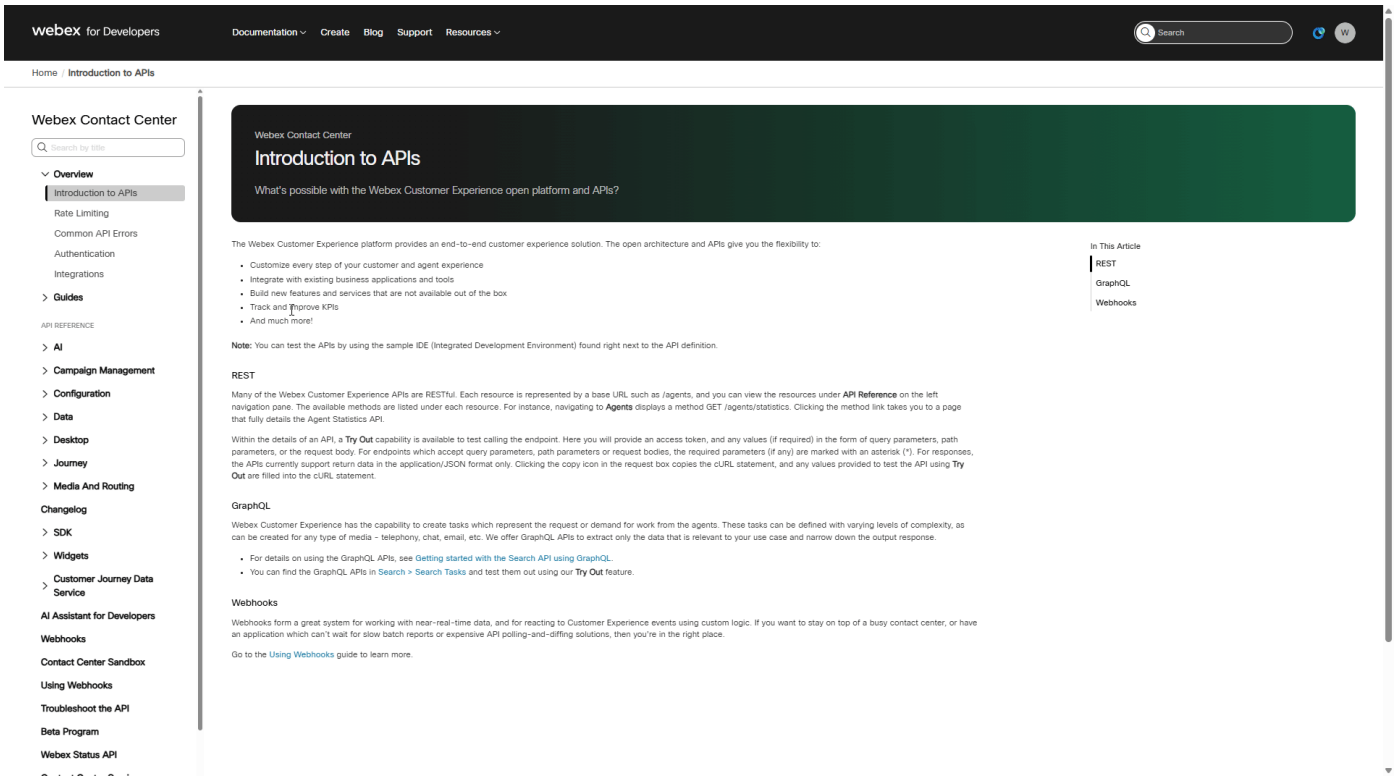
> **Note**
>
> You may proceed with other tasks without waiting for the callback time. When the time comes, please remember to make yourself available to accept the call.

Callback status verification over API.

1. Open **Developer Portal** and click on **Log In**. Your login will be of the format **wxcclabs+admin_IDYour_Attendee_ID@gmail.com**. You will see another login screen with Webex icon on it where you may need to enter the email address again and the password provided to you.
2. Click on the little arrow next to **Documentation**, choose **Webex Contact Center** under **Customer Experience** section.

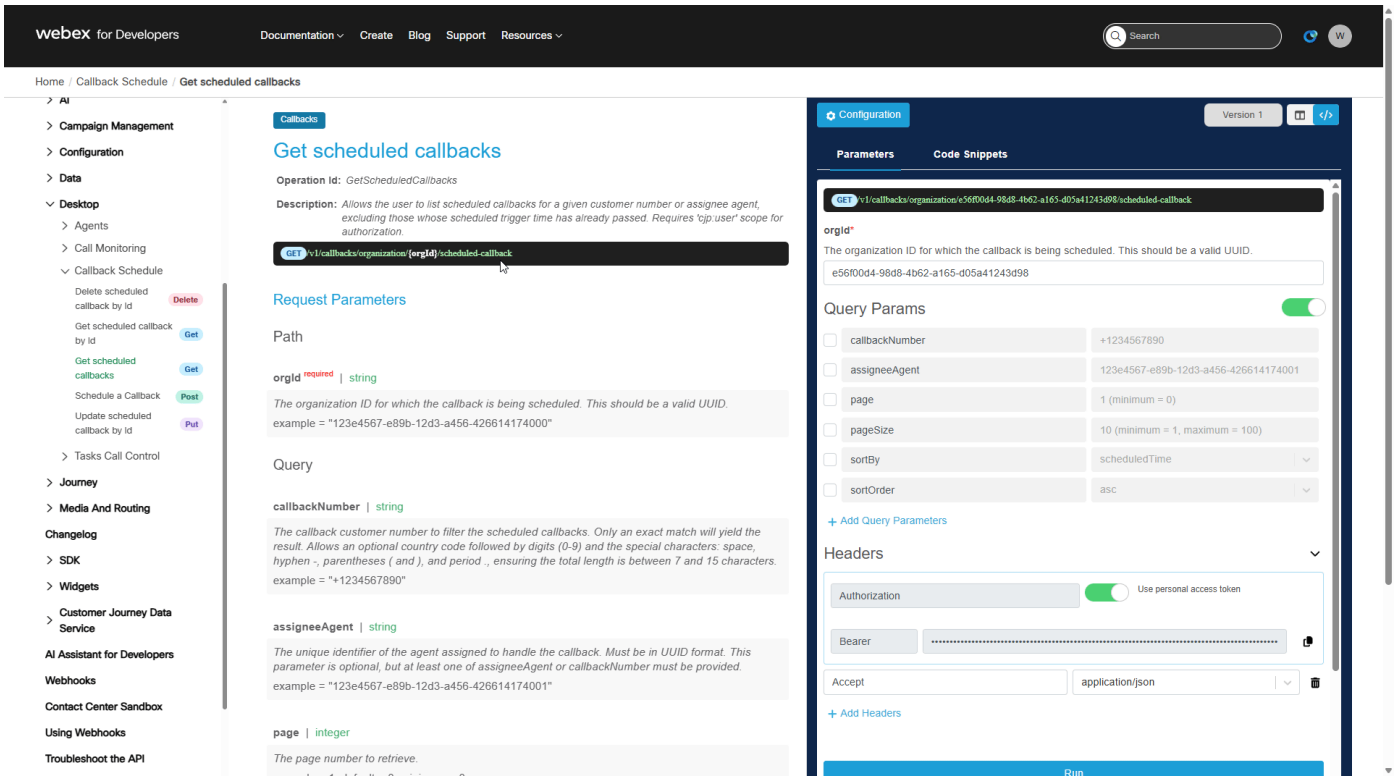


3. On Menu pannel on the left, scroll down to **API Reference** section, expand **Desktop** and then expand **CallBack Schedule**
4. Click on **Get scheduled callbacks** to open an endpand description page.



5. On the right hand side under **Query Params** set a checkbox next to ***callbackNumber***, then type 11 digit number you provided while were doing the test call.

6. Then click **Run**.



7. Verify output of the executed API call. Observe the important keys:

```
{
    "id": "3824bcea-03c7-41b8-957d-5d62ecda3b82",          // Unique identifier for the scheduled callback.
    "customerName": "+48575638602",                        // Customer Name. Uses Callback Number if not provided specifically
    "callbackNumber": "+48575638602",                      // Phone number provided for the callback.
    "timezone": "Europe/Amsterdam",                        // Timezone in which the callback is scheduled.
```

```
    "scheduleDate": "2026-01-19",                       // Date for the callback in ISO format (YYYY-MM-DD).
    "startTime": "18:45:00",                            // Scheduled start time in ISO 8601 format (HH:mm:ss), local to the specified timezone.
    "endTime": "19:20:00",                              // Scheduled end time in ISO 8601 format (HH:mm:ss), local to the specified timezone.
    "queueId": "ee46583c-8d0d-4c09-8829-8c0b79c11a79",     // Identifier for the queue to route the callback request.
    "callbackReason": "Attendee ID 140. Callback initiated from the Agent Desktop."    // Text provided while schedule a call back.
    //<ommitted>,

}
```

Full Schema Definition can be found in the **API Reference** for this API call.

Congratulations on completing another mission.

4.2 Conclusion

We hope you found the CallBack track both challenging and rewarding as you deepened your expertise with Webex Contact Center. In this session, you explored key strategies for implementing and refining the Callback feature to enhance customer experience and operational efficiency.

Key missions included:

- **Adding Basic Callback** – Ensuring customers have the option to request a return call instead of waiting in the queue.
- **Scheduling a Callback on Errors** – Handling unexpected issues by automatically scheduling a callback when an error occurs.
- **Preventing Duplicate Callbacks** – Implementing safeguards to avoid redundant callback requests, improving efficiency and customer satisfaction.

By mastering these techniques, you are now equipped to design more efficient and customer-friendly callback workflows within Webex Contact Center.

Should you have any questions or need further assistance, feel free to reach out or join the Webex discussion forums. We’re excited to see how you apply these skills in your future projects!

Thank you for completing the Callback track, and we look forward to your continued growth with Webex Contact Center.

5. FINAL CHALLENGE

5.1 Troubleshooting Mission

5.1.1 **Ooops! Something went wrong...**

```
> Searching for a challenge guide...
> System error - The challenge guide has been deleted.
> Please ask for help via the chat in the bottom-right corner.
```

6. Quick Links

Administrator Login	wxcclabs+admin_IDYour_Attendee_ID@gmail.com
Agent Login	wxcclabs+agent_IDYour_Attendee_ID@gmail.com
Supervisor Login	wxcclabs+supvr_IDYour_Attendee_ID@gmail.com
EntryPoint/Channel Name	Your_Attendee_ID_Channel
Team	Your_Attendee_ID_Team
Standard Queue	Your_Attendee_ID_Queue
Skill-Based Queue	Your_Attendee_ID_SBR_Queue
Bussiness Hours	Your_Attendee_ID_Bussiness_Hours
Control Hub	https://admin.webex.com/
Agent Desktop	https://desktop.wxcc-us1.cisco.com/