# 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.
- **Leverage AI and Automation:** Explore integrations with pre-configured AI tools such as **Webex AI Agent** or interactive customer interactions and **Cisco Text-to-Speech** for dynamic responses.
- **Optimize Routing Logic:** Implement advanced routing capabilities, such as callback handling, last agent routing and using Global ariables 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!

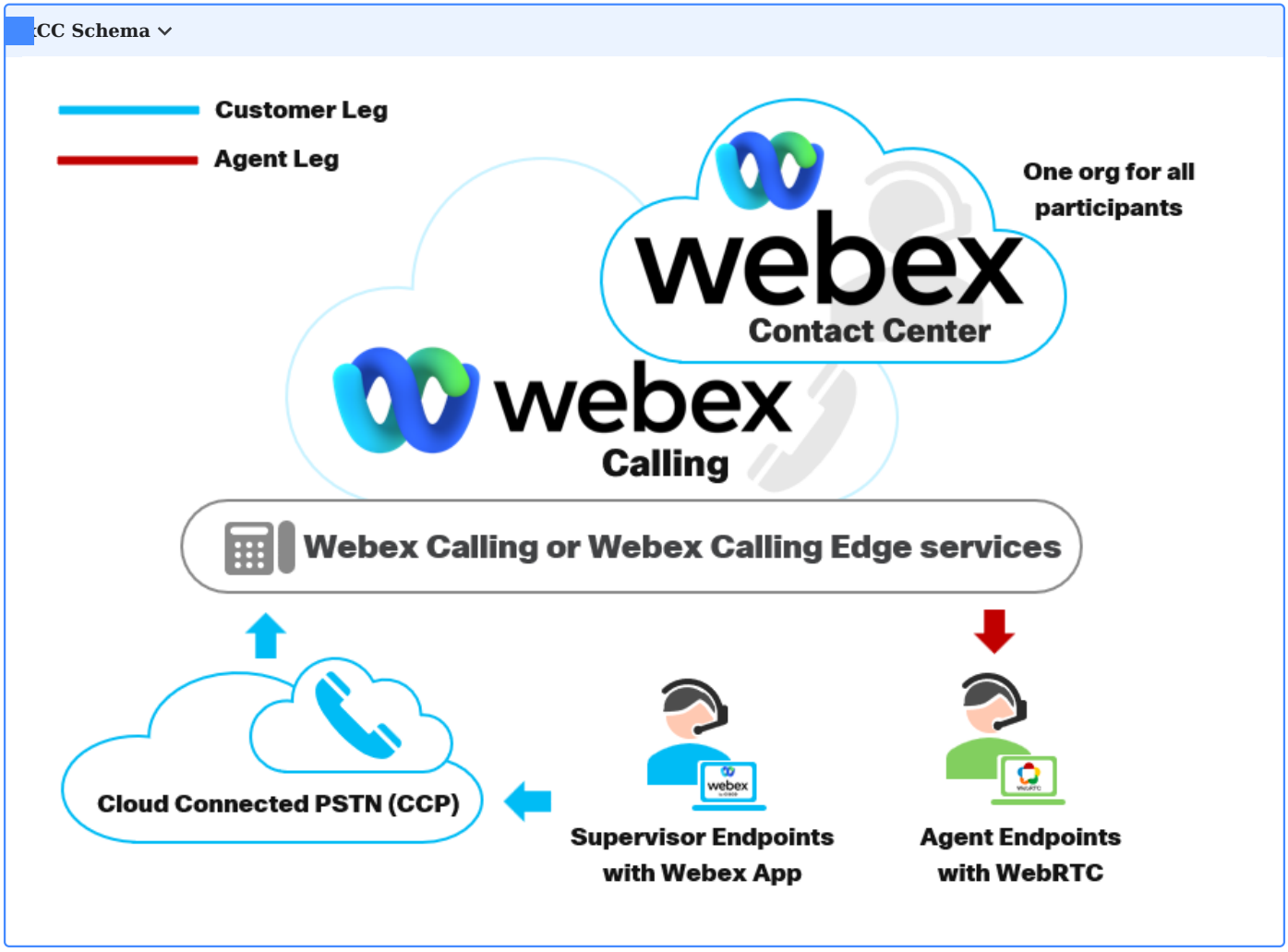
## 1.2 Getting to know your environment

### 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 agents, 1 Supervisor and 1 admin.
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.

---

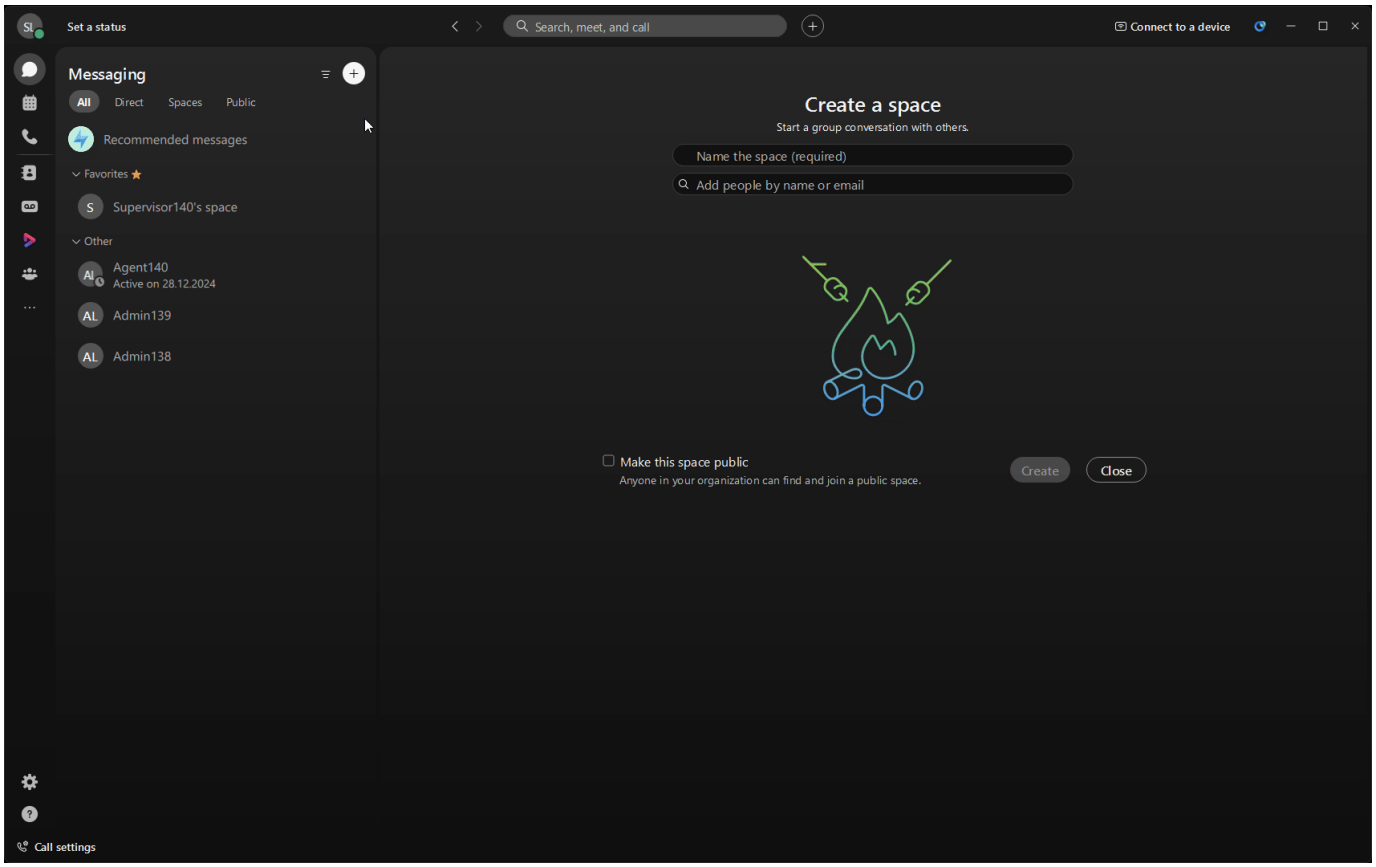
#### Testing

##### AGENT DESKTOP

Use Agent Desktop application  pre-installed on your workstation to login your agent. In addition, Desktop profile was configured in a way where you don't need to select a Telephony line. By default only Desktop Audio (WebRTC) has been enabled.

##### CALLING TO CONTACT CENTER

All call 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**

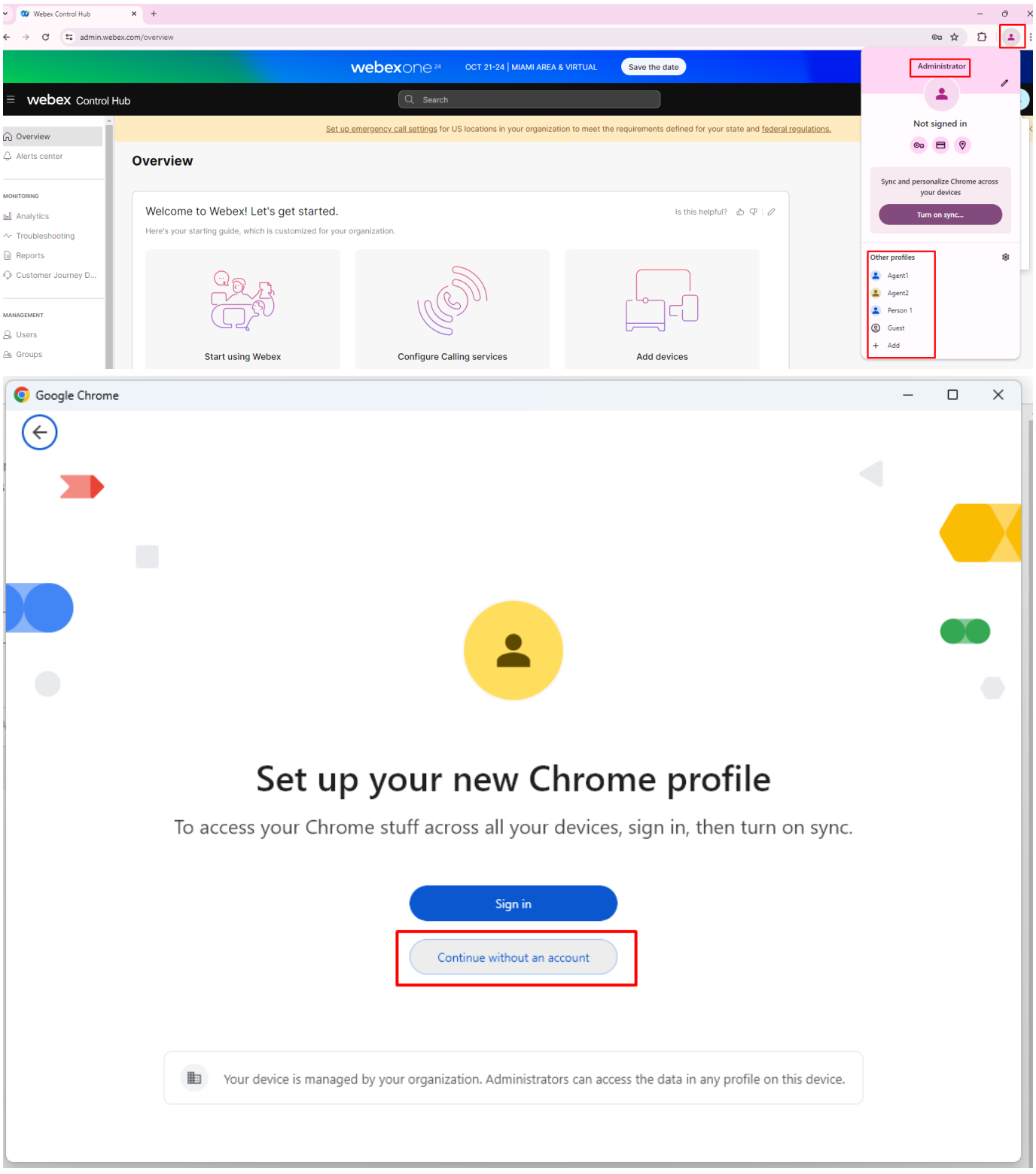
International dialing is not allowed so you won't be able to dial your cell phones unless you have a US number.

## 1.3 Chrome and Webex App Setup

#### 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

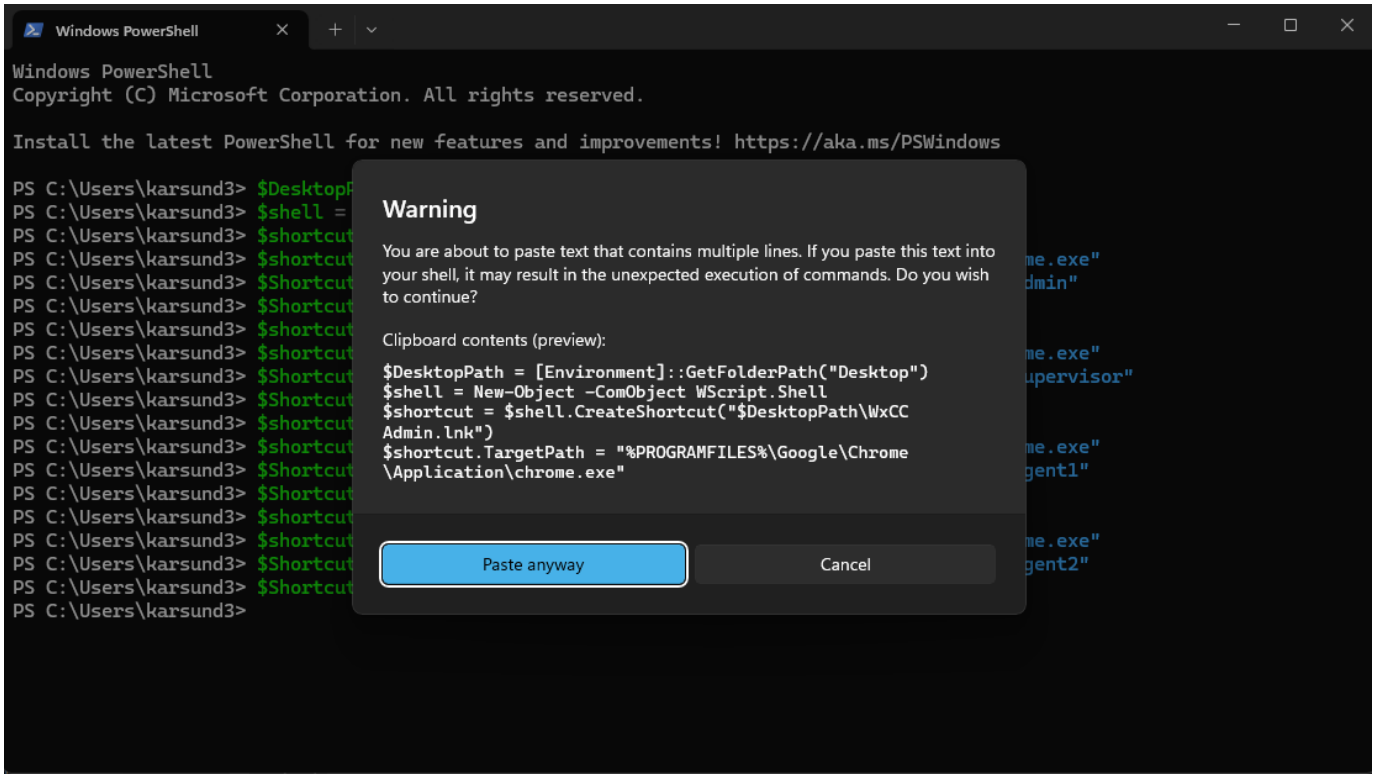



### Creating Chrome user profiles

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

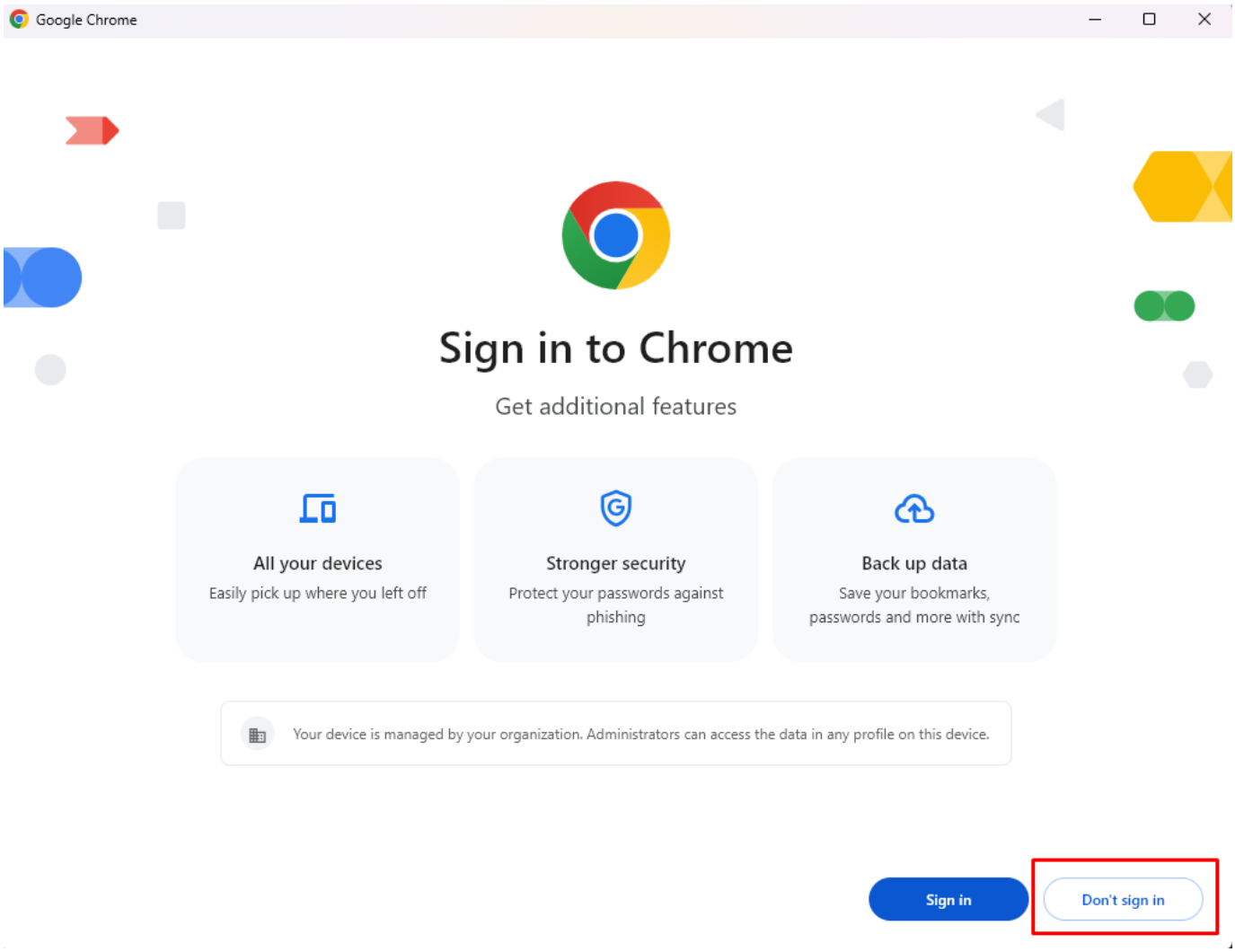
```
$DesktopPath = [Environment]::GetFolderPath("Desktop")
$shell = New-Object -ComObject WScript.Shell
$shortcut = $shell.CreateShortcut("$DesktopPath\WxCC Admin.lnk")
$shortcut.TargetPath = "%PROGRAMFILES%\Google\Chrome\Application\chrome.exe"
```

```
$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()
```

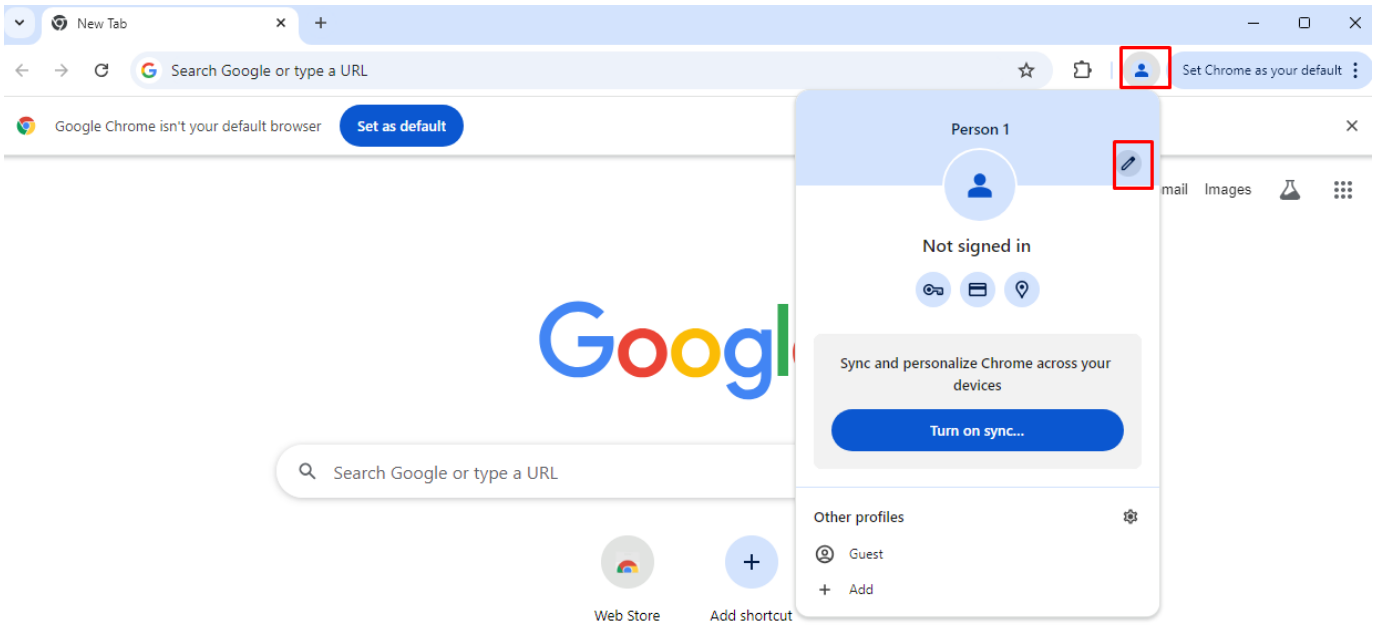



Check the desktop of your lab PC. You should find 2 Chrome shortcuts created - WxCC Admin, **WxCC Agent1 and WxCC Supervisor**

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.

# 1.4 Choose Your Adventure

## 1.4.1 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.

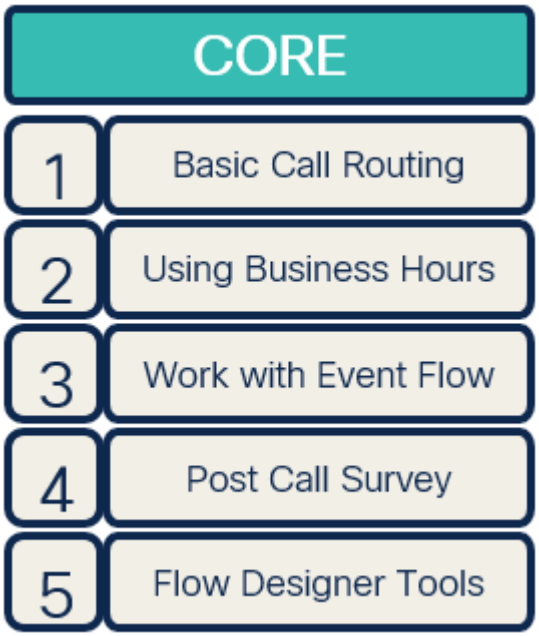
Want to push further? Finish 2 tracks to excel, 3 tracks to become an expert, or take on all 4 tracks to earn the ultimate title of Mega Superstar!

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**. One of the missions contains information you are going to use on the final troubleshooting task, which we will complete as the last mission.

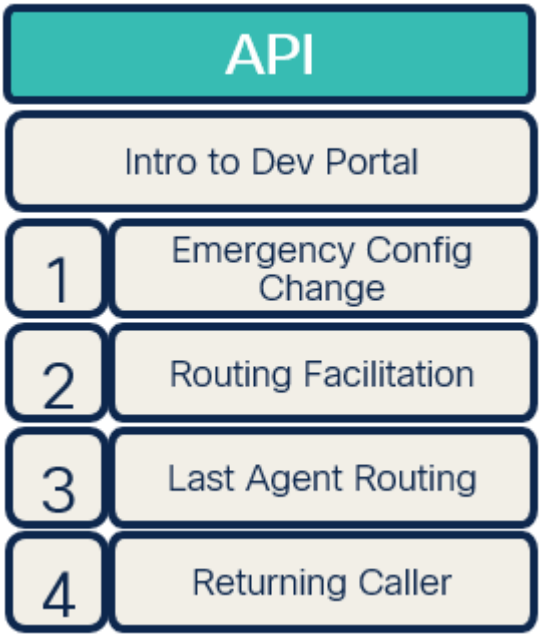
## 1.4.2 Track 1: Core Track

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.



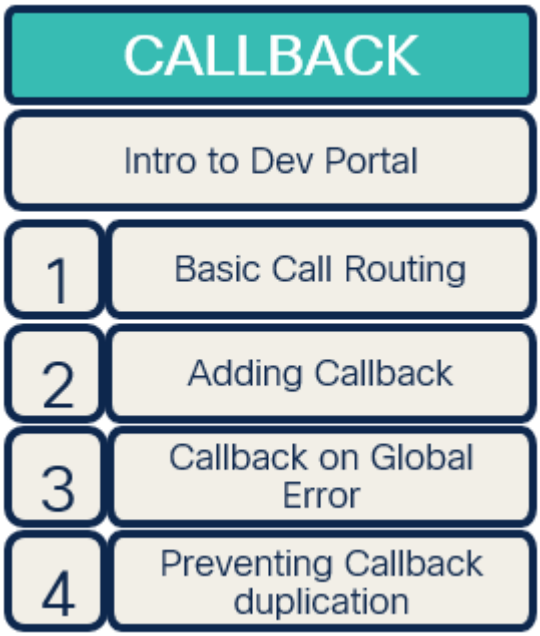

## 1.4.3 Track 2: API Track

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




### 1.4.4 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.




### 1.4.5 Track 4: AI Agent Track

Although the smallest, but challenging track. It involves configuring Cisco’s native AI agent (bot) and integrating it with Flow Designer to enable flow customization.

# 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 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.

**ıy Flow Templates? [Optional]**

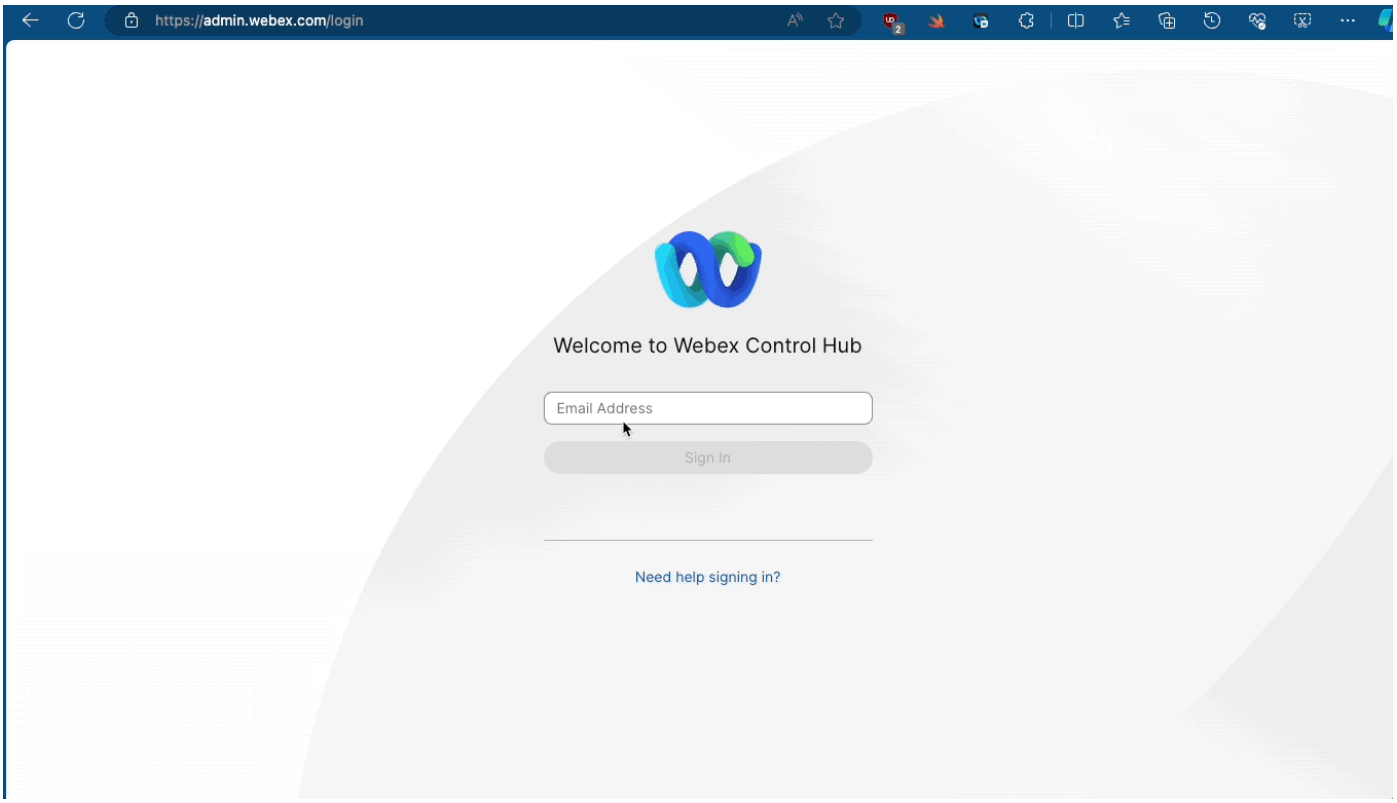
Flow Templates in Webex Contact Center are an essential feature for flow developers, offering a range of benefits that streamline the development process and enhance the efficiency and consistency of flow creation. Here’s what they bring to the table:

- **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

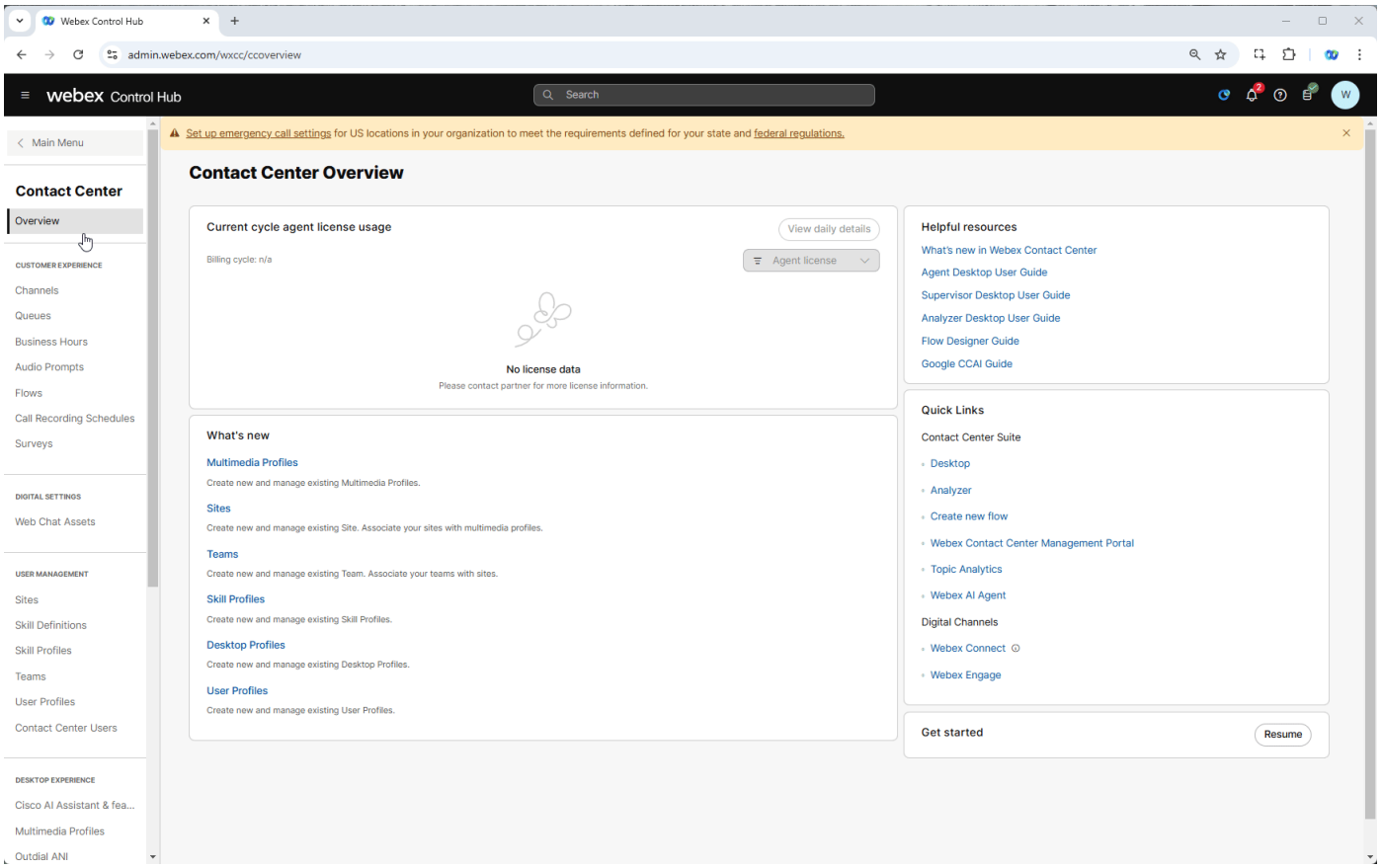
1. Login into Webex Control Hub by using your Admin profile. Your login will be of the format **wxcclabs+admin_IDYour_Attendee_ID@gmail.com**. You will see another login screen with OKTA on it where you may need to enter the email address again and the password provided to you.




**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 it
3. Navigate to **Flows**, click on **Manage Flows** dropdown list and select **Create Flows**
4. New Tab will be opened. Navigate to **Flow Templates**
5. Choose **Simple Inbound Call to Queue** template and click **Next**. You can open View Details and to see observe flow structure and read flow description
6. Name your flow as **Main_Flow_Your_Attendee_ID**. Then click on Create Flow



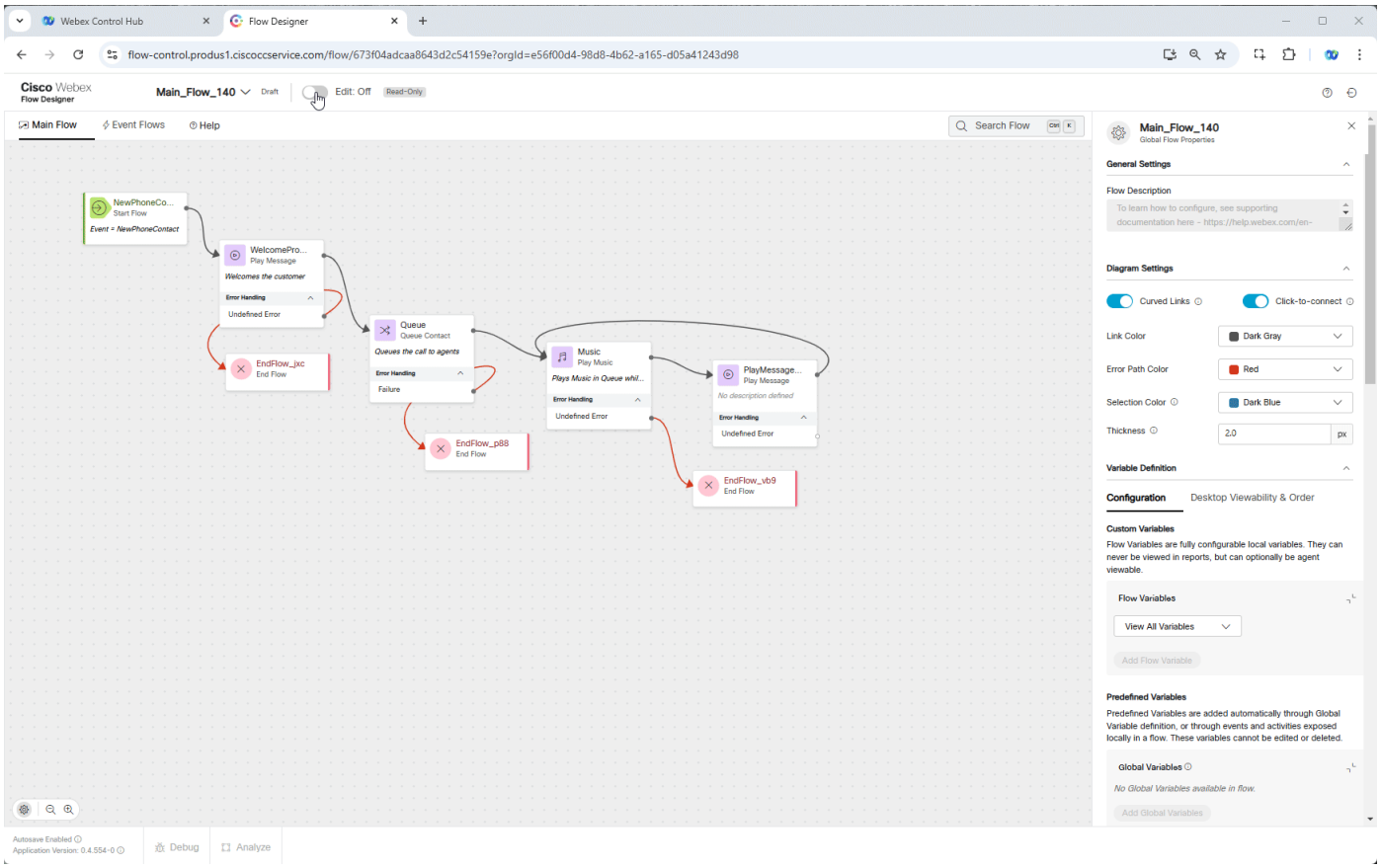

7. **Edit** should be set to **On** when you create new flow, but if not switch it from **Edit: Off** mode to **Edit: On**. Select **Play Message** node with label **WelcomePrompt** and on the node settings modify **Text-to-Speech Message** to any greetings you like. This message will be the first message you hear while calling to your script.
8. Select **Queue** node. On the **General settings** keep Static Queue checked and select queue **Your_Attendee_ID_Queue** from the drop down list

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

9. [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.
10. On 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.

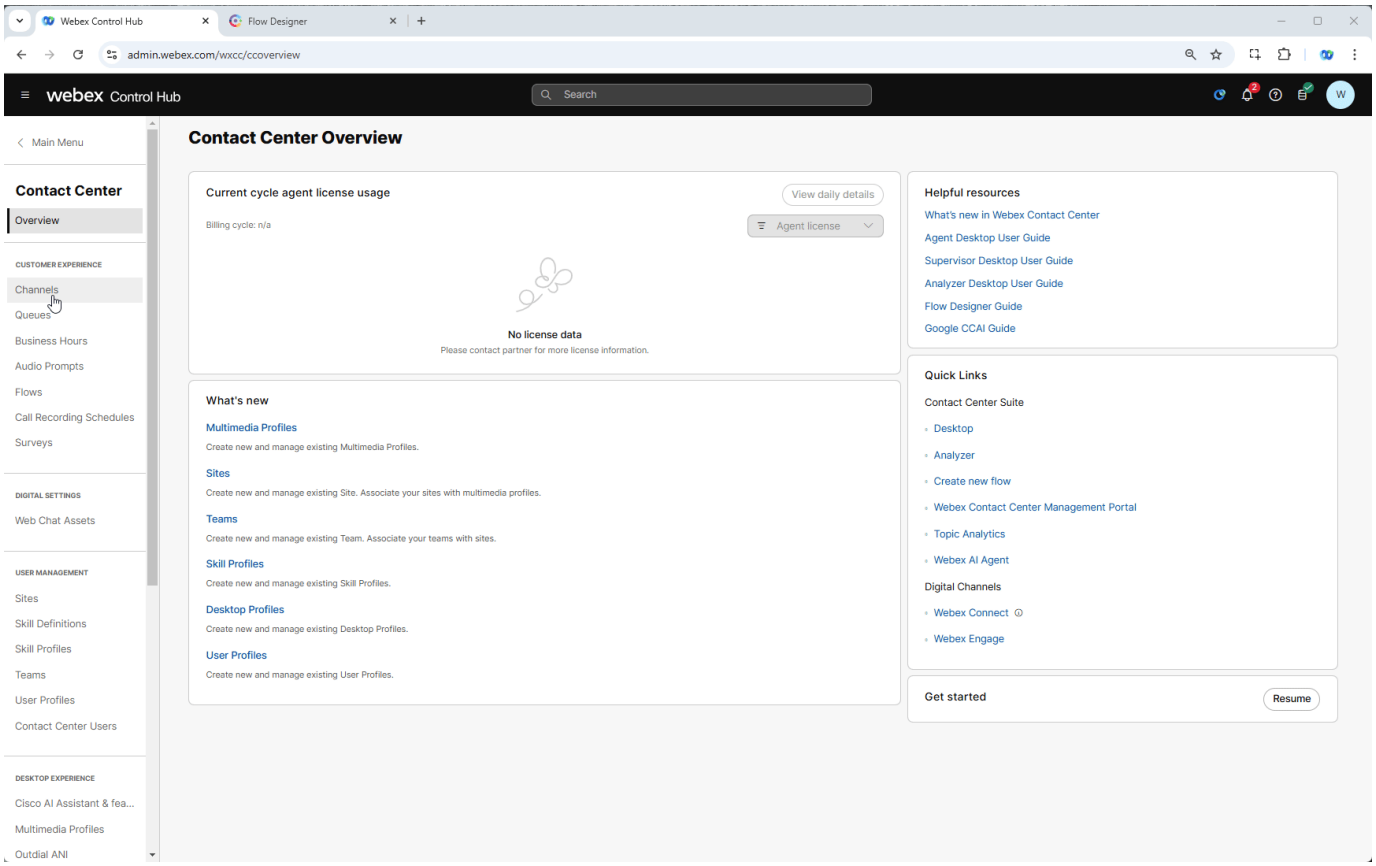
11. Click **Publish** Flow




12. In popped-up window, click on dropdown menu to select **Latest** label, then click **Publish**.
13. 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** .
14. Click on **Your_Attendee_ID_Channel**
15. In **Entry Point** settings section change the following, then click **Save** button:

    Routing Flow: **Main_Flow_Your_Attendee_ID**

    Version Label: **Latest**

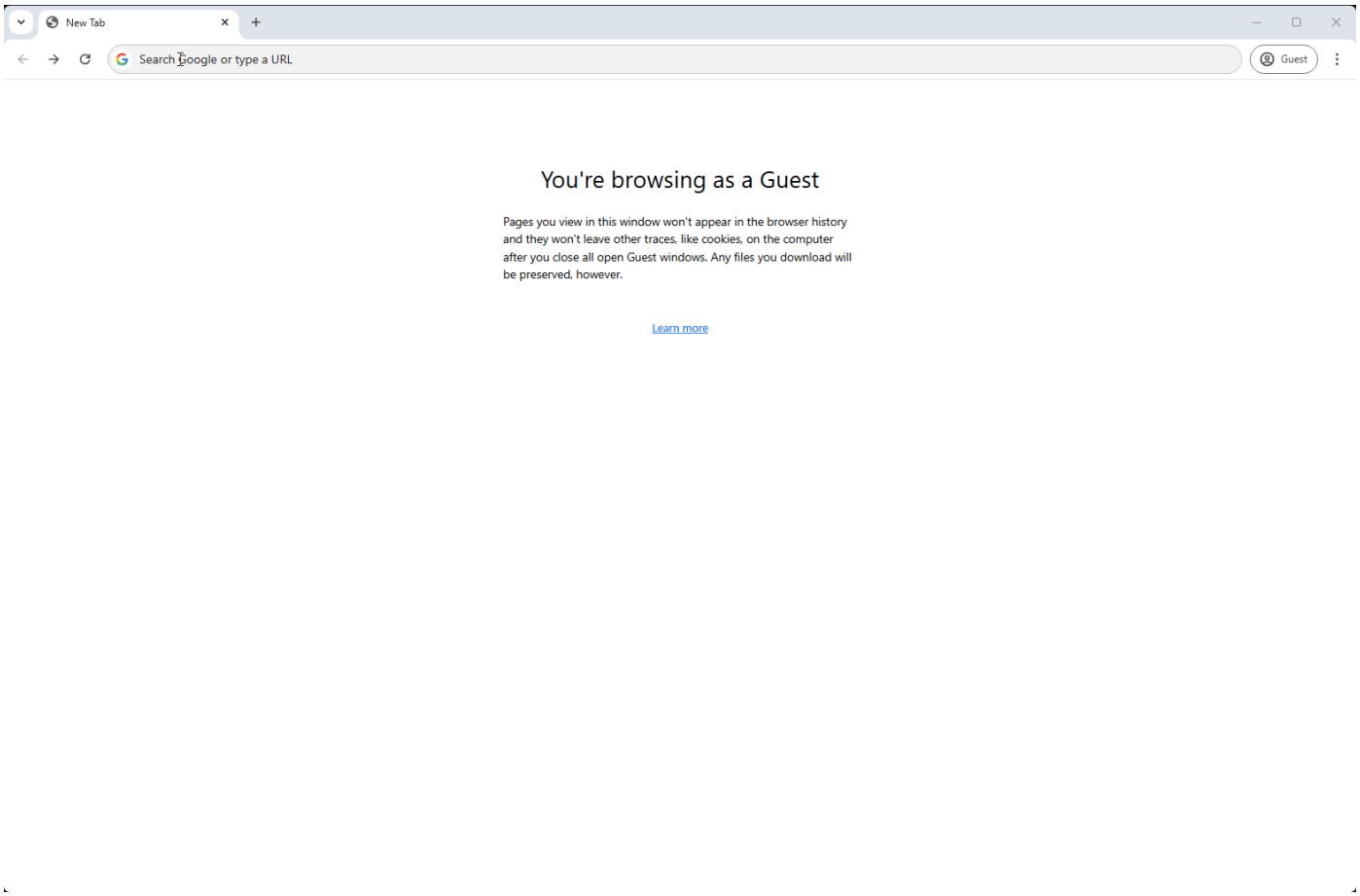



---

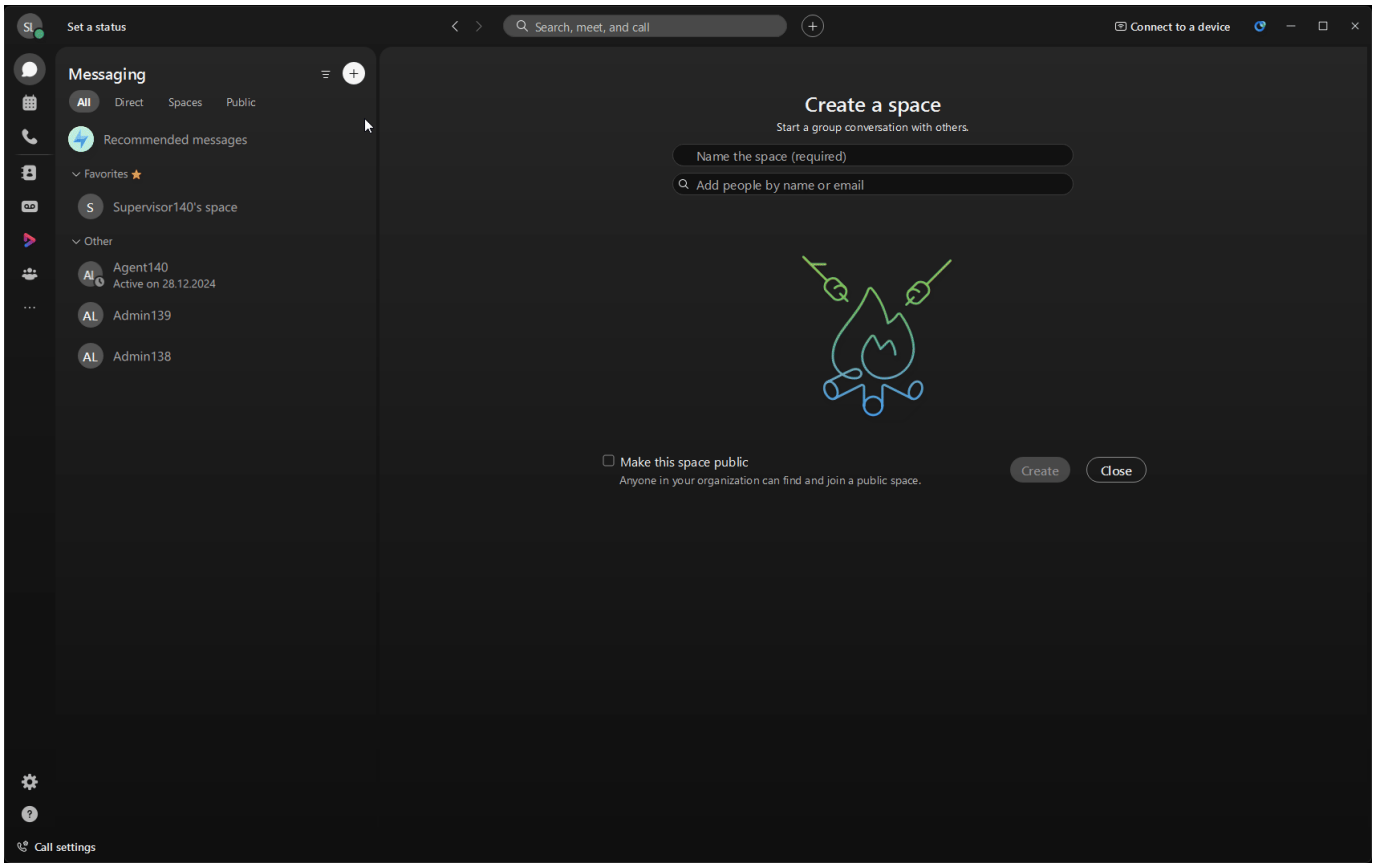
CHECKPOINT TEST

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 OKTA on it where you may need to enter the email address again and the password provided to you.
2. Select your Team **Your_Attendee_ID_Team**. Click **Submit**. Allow browser to access Microphone by clicking **Allow** on every visit.
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.




---

### Enhance Your Flow by adding Language

MISSION DETAILS

Your mission is to:

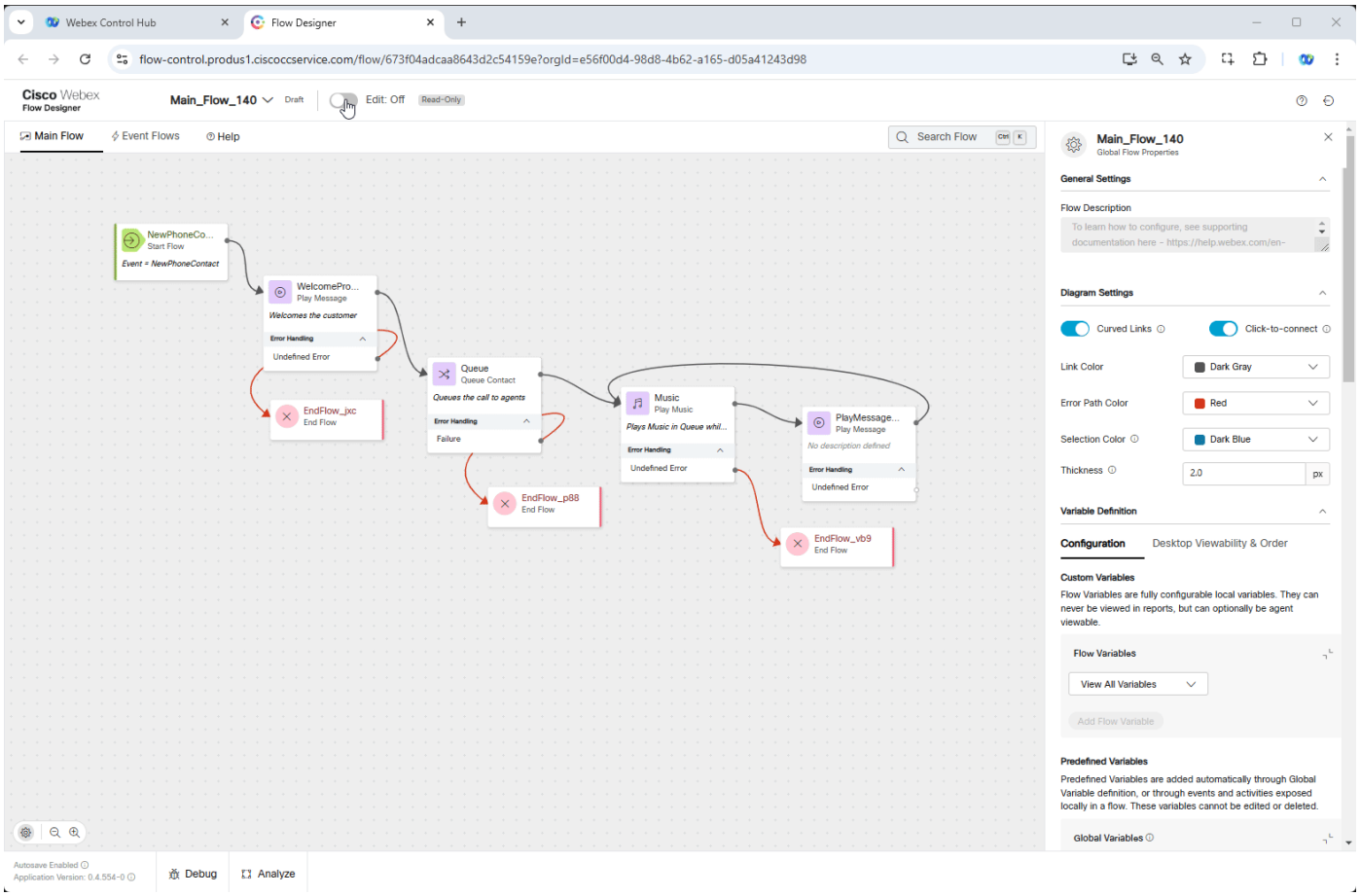
- Use the same flow created in the previous section.
- Modify the TTS section to use **en-AU** (English - Australia) and connect the Set Variable node as illustrated below.
- Place a call to verify and validate the speech functionality.

**xt-to-Speech (TTS) in Webex Contact Center[Optional]**

All supported languages can be found 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. On the right hand side you will see the **Global Flow Properties** Panel. 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.



3. Add a **Set Variable** with following configuration.

   Delete connection between **NewPhoneContact** and **WelcomePrompt**

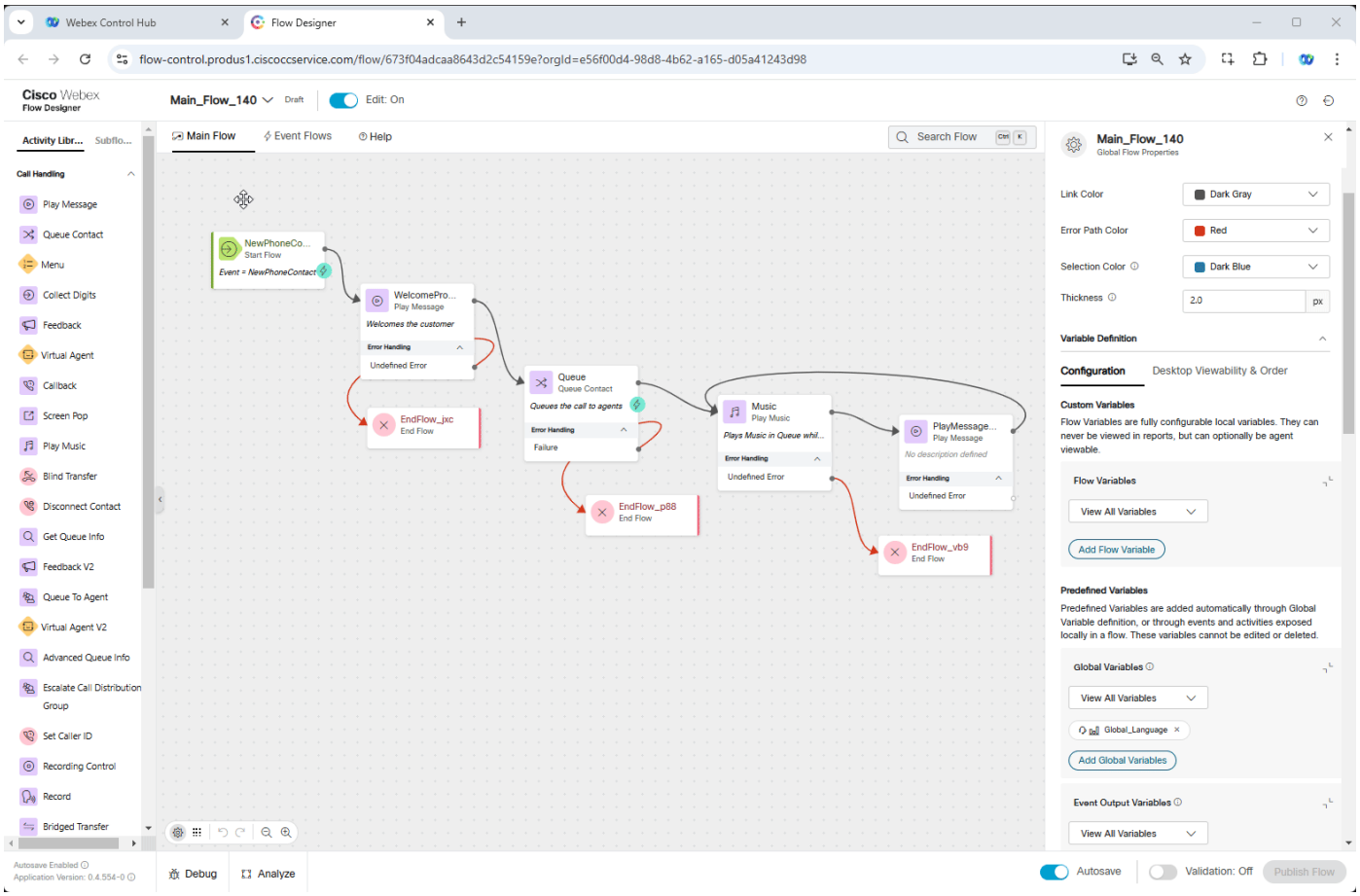
   Connect **NewPhoneContact** to **Set Variable**

   Connect **Set Variable** to **WelcomePrompt**

   Variable: **Global_Language**

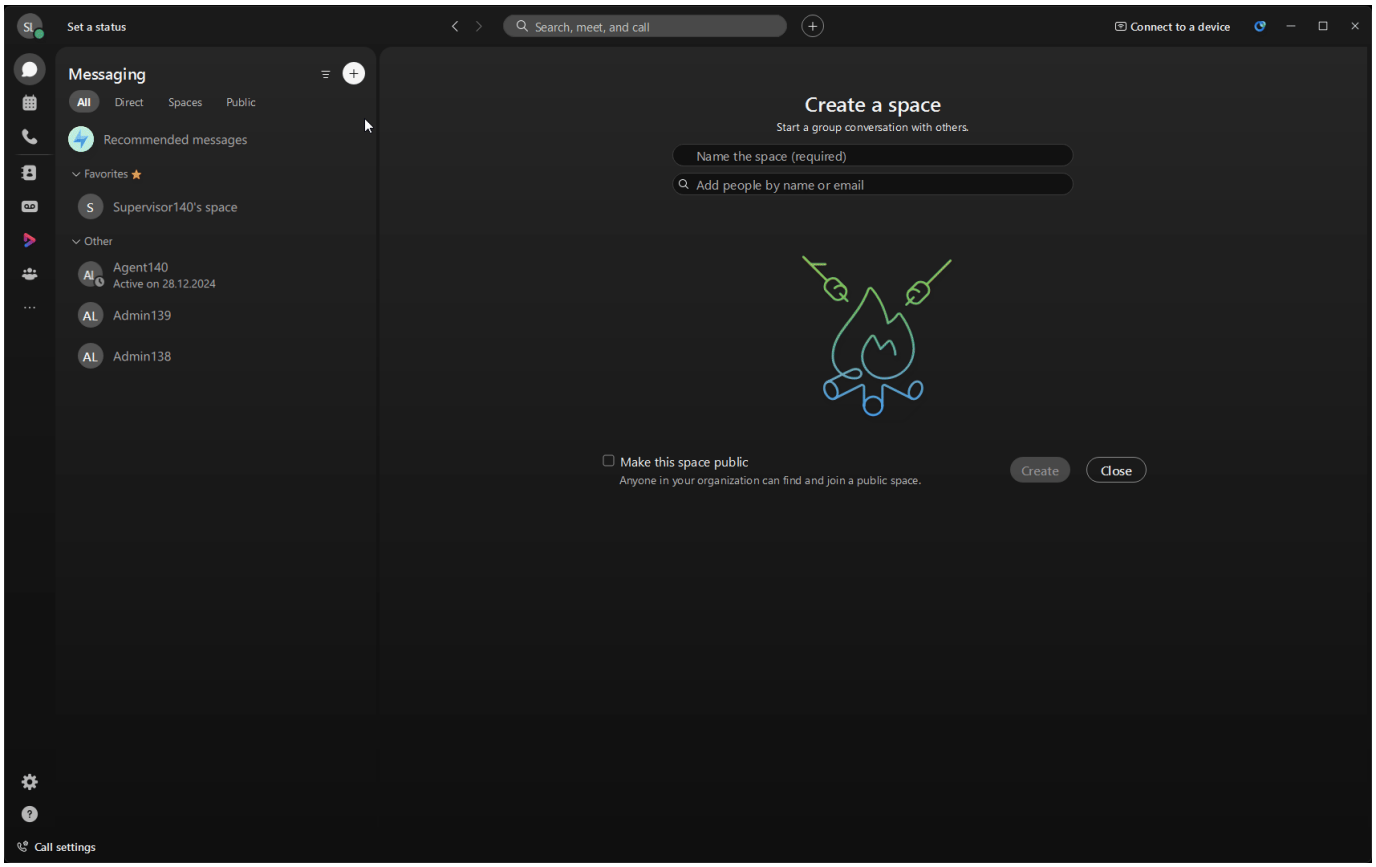
   Set Value: **en-AU**

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




TESTING

1. Open your Webex Desktop and make your agent **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.




1. Verify if the TTS language changed

**Congratulations on completing another mission.**

## 2.1.2 Mission 2: Using Business Hours to add flexibility

#### 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 pre-configured 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.

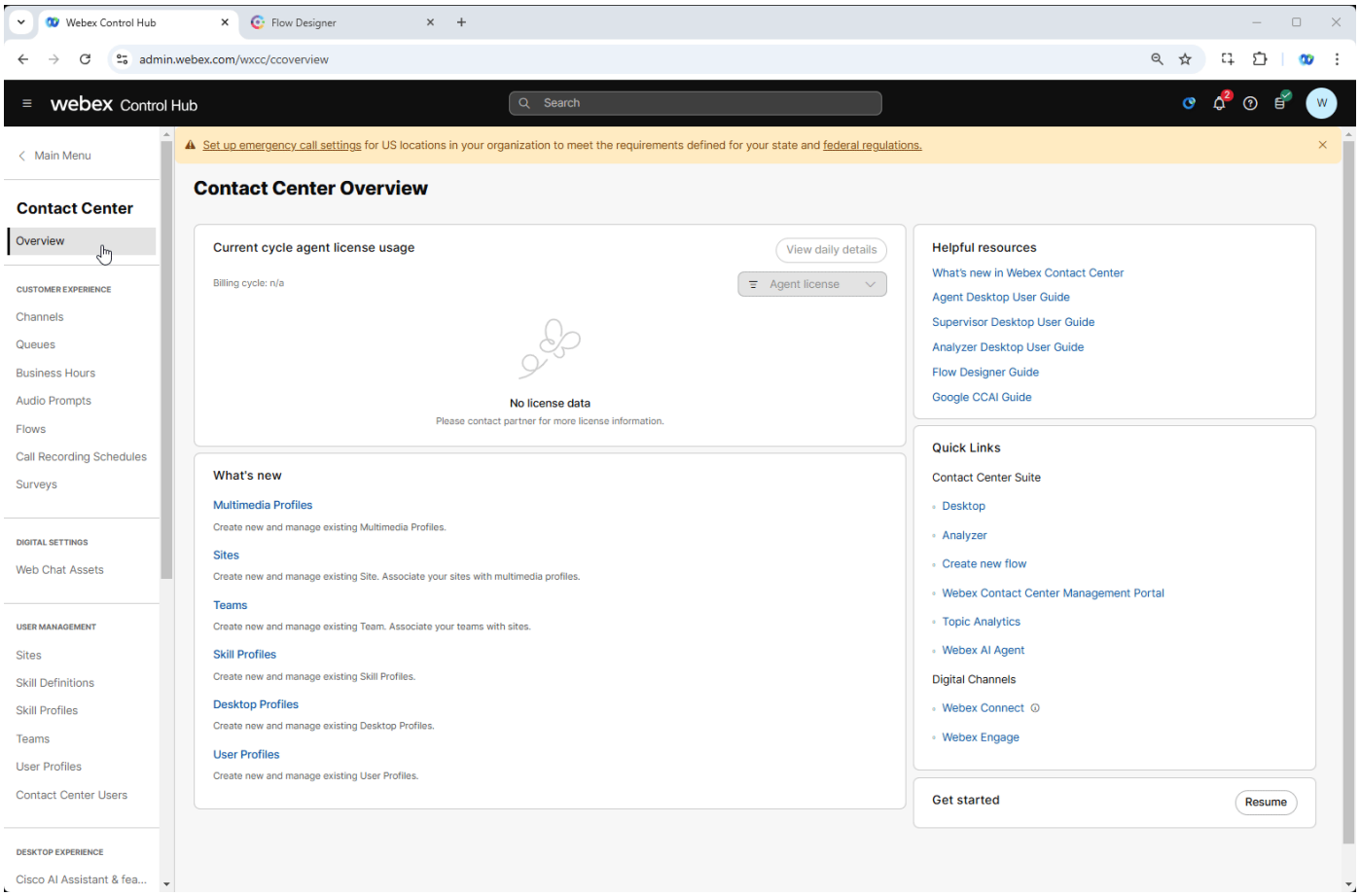
#### Mission Details

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.

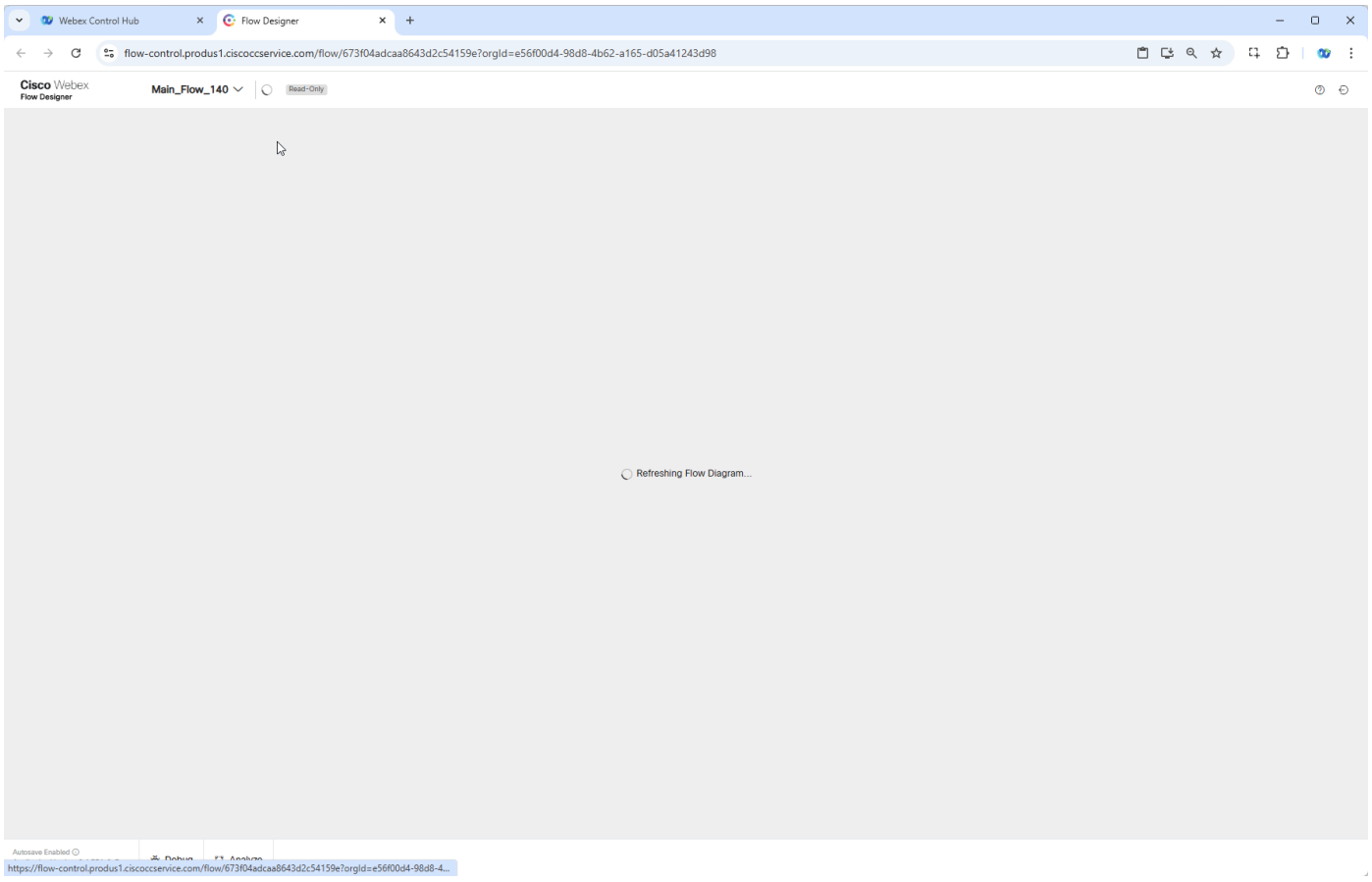
#### Build

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".



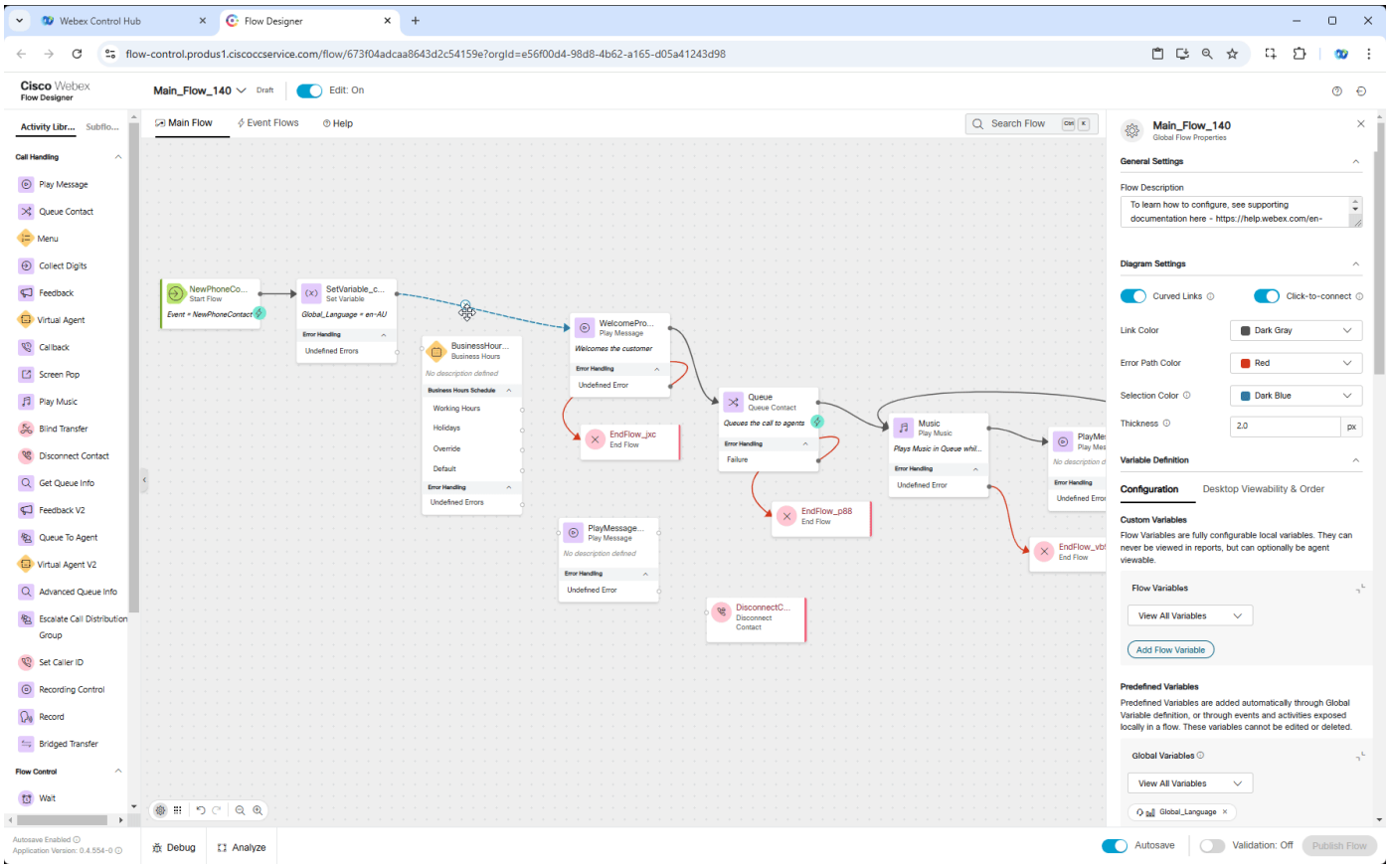

2. Switch to Flow Designer. Open your flow **Main_Flow_Your_Attendee_ID** and make sure **Edit** toggle is **ON**.
3. Drag and drop following nodes to the canvas:

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




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

- **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




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

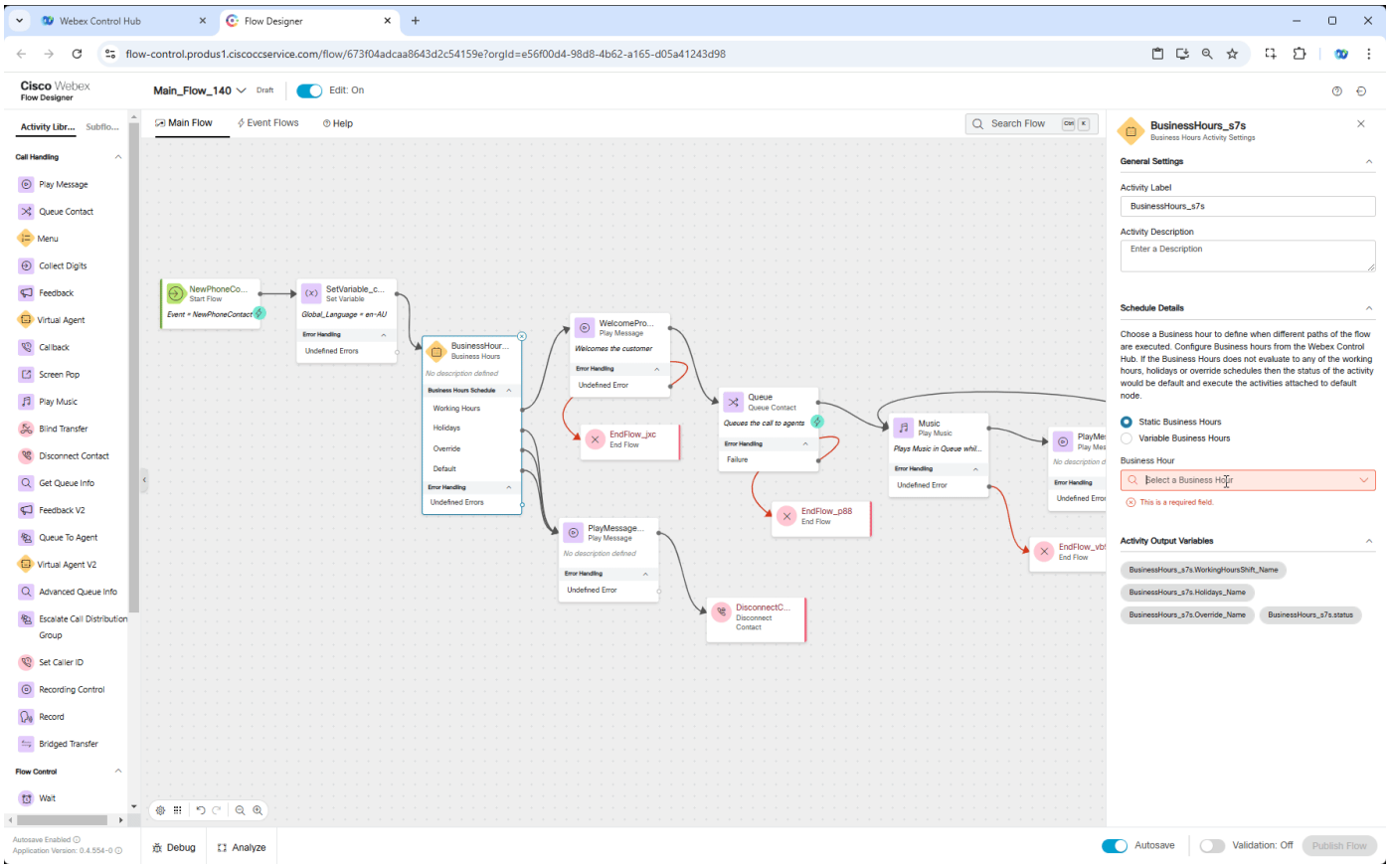
   Enable Text-To-Speech

   Select the Connector: Cisco Cloud Text-to-Speech

   Click the Add Text-to-Speech Message button and paste text: ***It's not working hours currently. Please call later. Goodbye.***

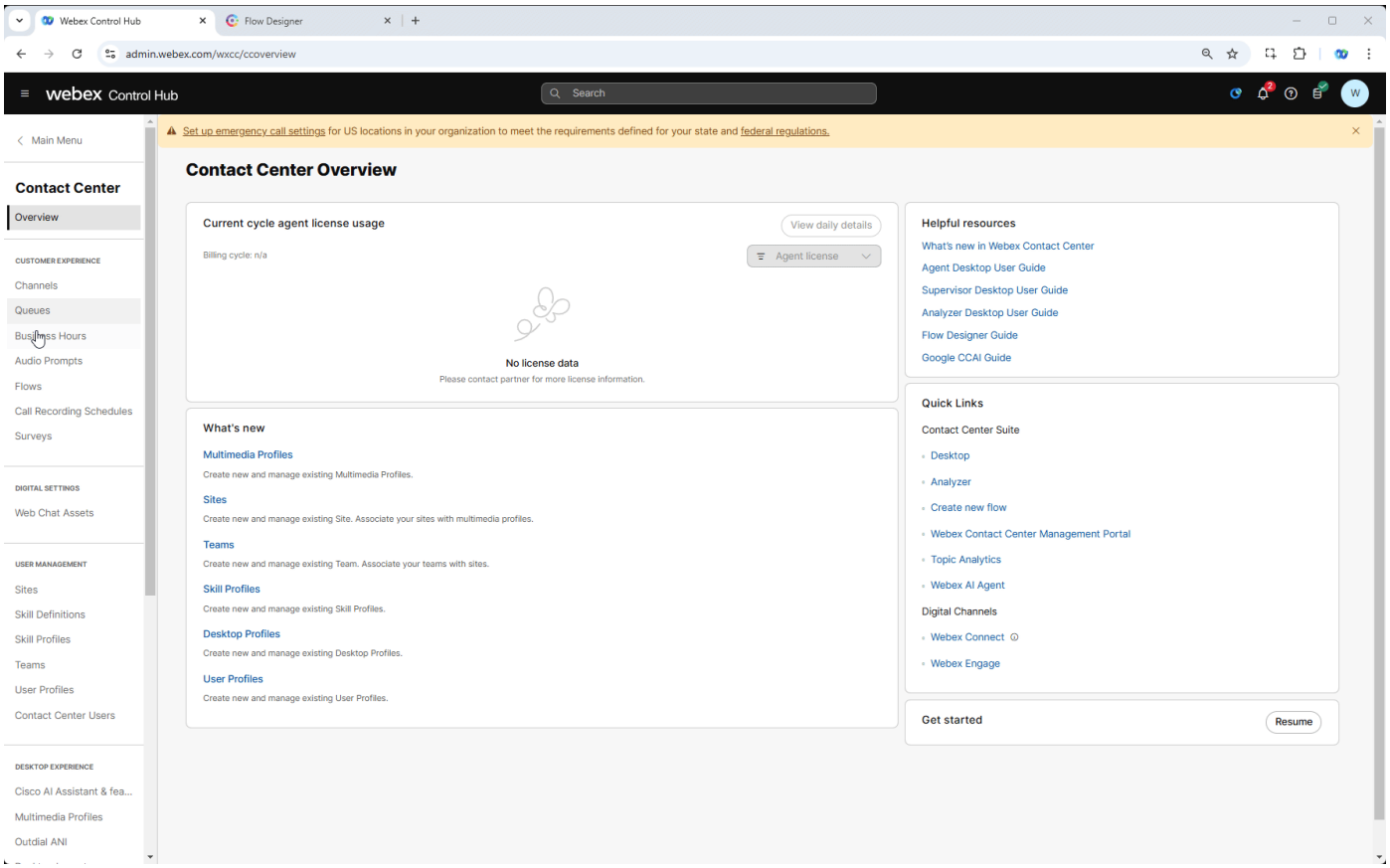
   Delete the Selection for Audio File
7. Validate the flow by clicking **Validate**, **Publish** and select the Latest version of the flow

> **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

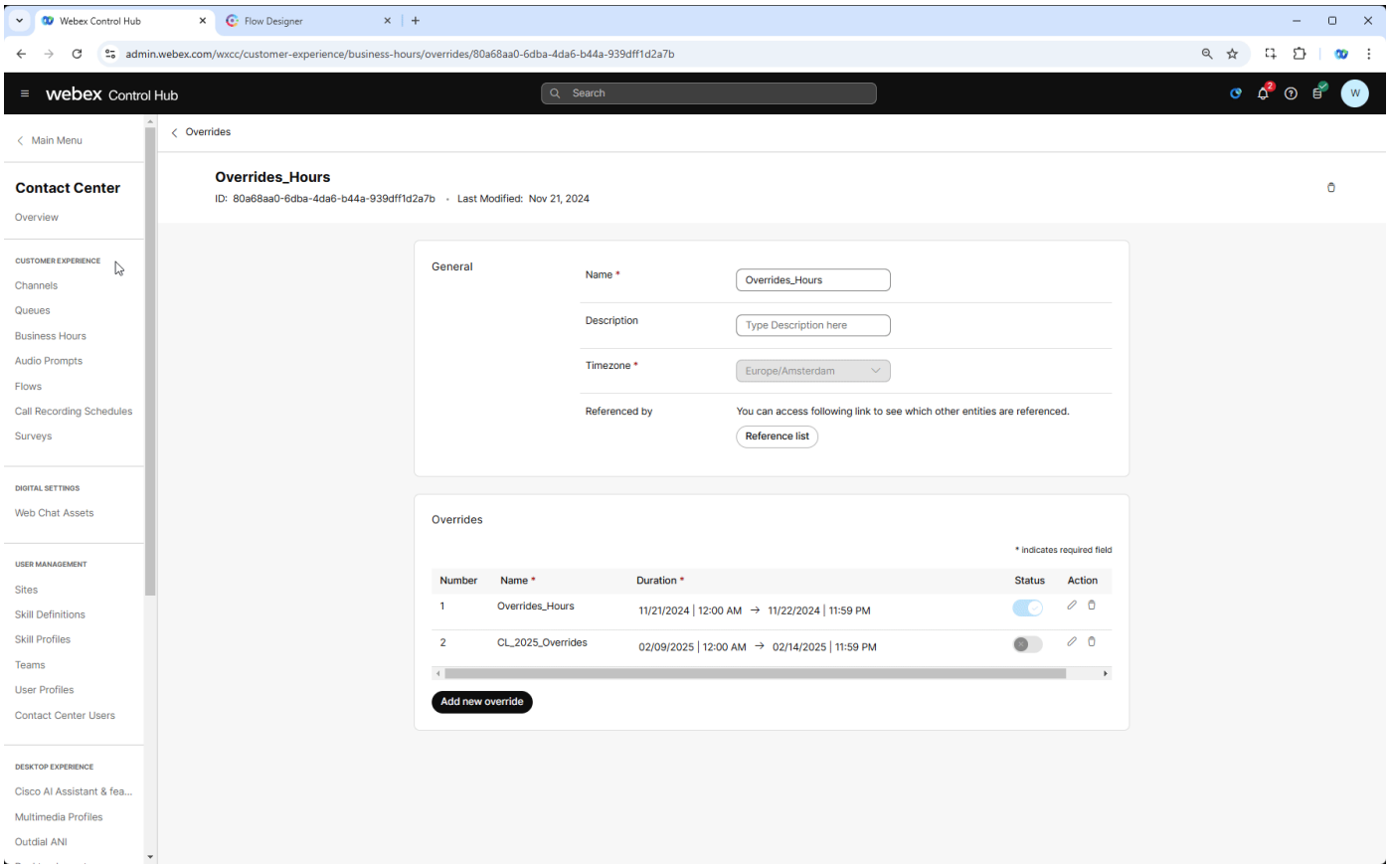



### Testing

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 on completing another mission.**

## 2.1.3 Mission 3: 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.

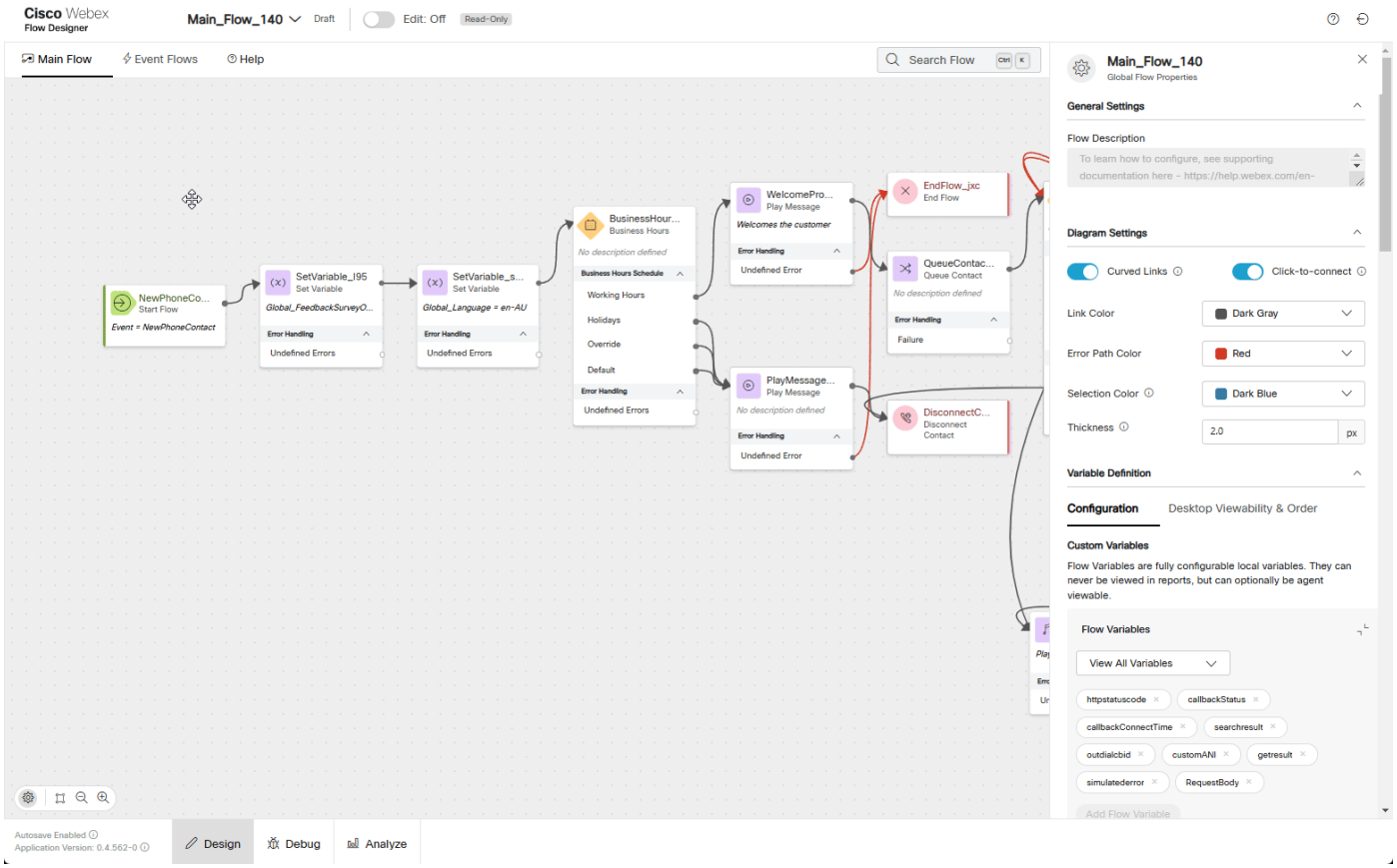
> **Note**
>
> In this mission, we'll utilize **Webhook.site**, a free online tool that generates a temporary, unique URL for capturing and inspecting HTTP requests. It's widely used by developers and testers for debugging and testing webhooks or other HTTP-based APIs.

**Build**

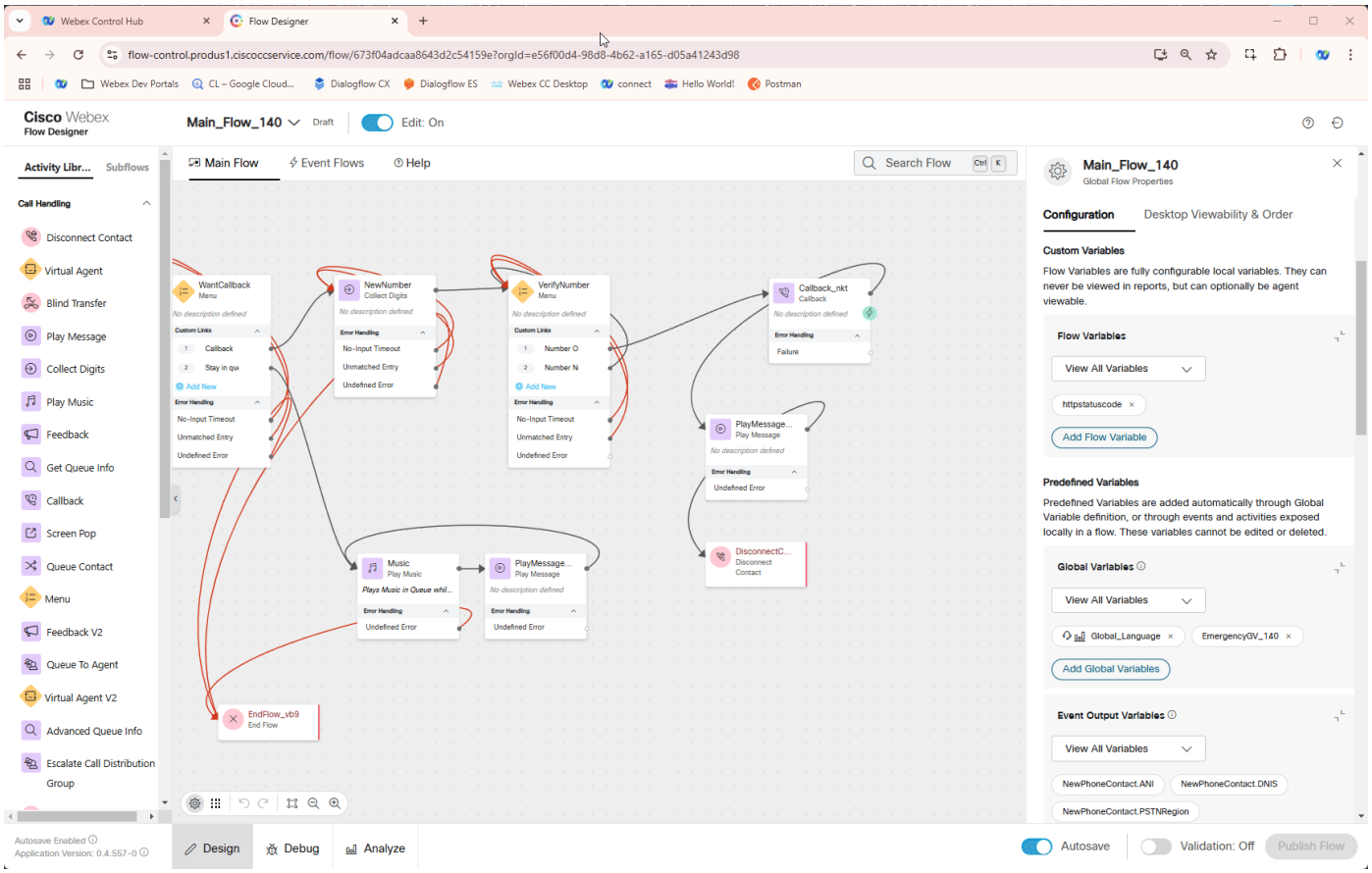
**Note**

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. You **DO NOT** need to 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 **Webhook.site**. Then click on **Your unique URL** to make a copy of URL. **DO NOT close this Tab**




4. Go back to your flow and navigate to **Even Flows** tab, delete **EndFlow_xkf** node which is connected to **AgentDisconnect**
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://webhook.site/.*

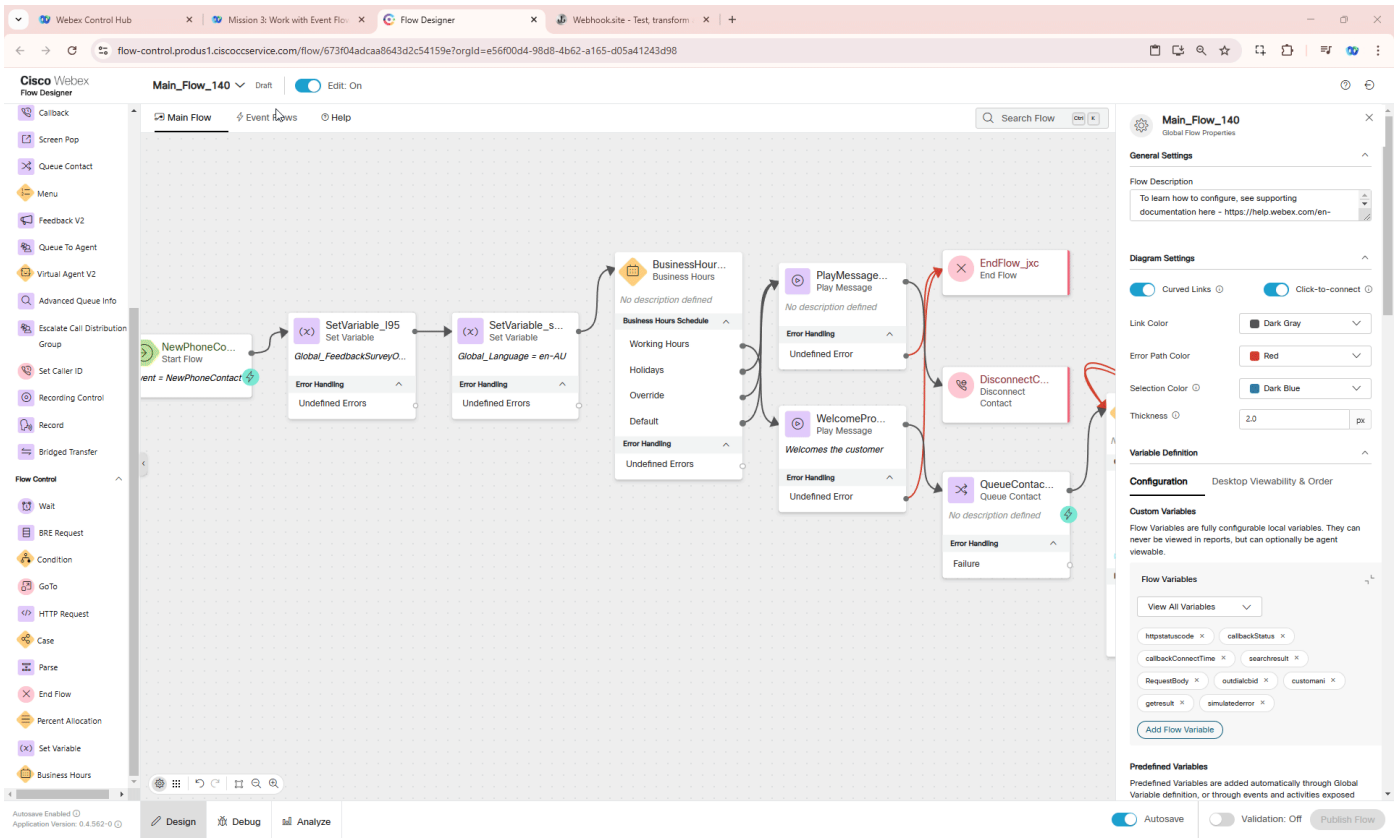
   Method: **POST**

   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.

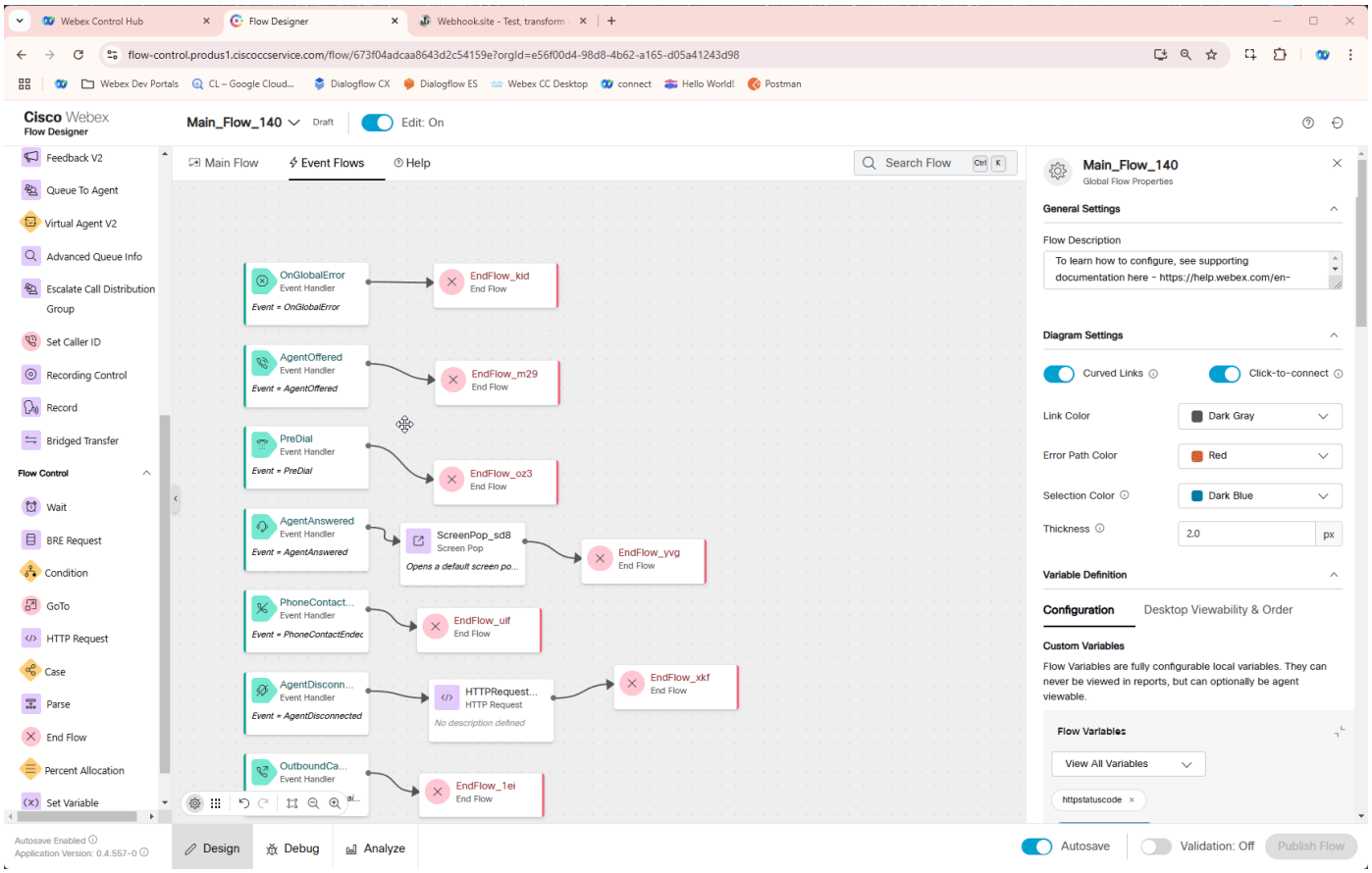



7. Modify **Screenpop** configuration in the same flow

   Screen Pop URL: **https://www.ciscolive.com/emea/faqs.html**

   Screen Pop Desktop Label: **Cisco Live Amsterdam 2025 FAQ**

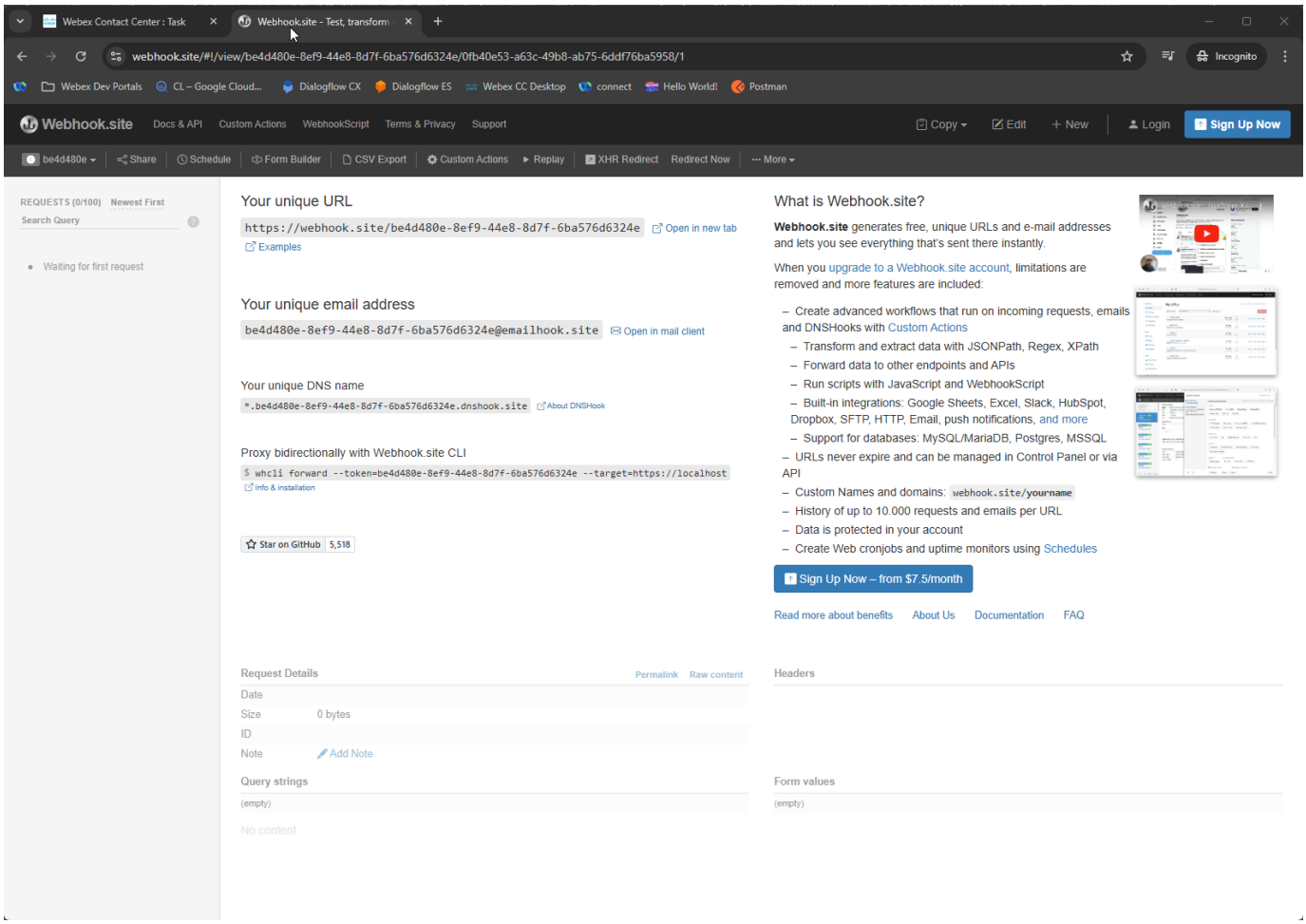
   Display Settings: **Inside Desktop**




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

### Testing

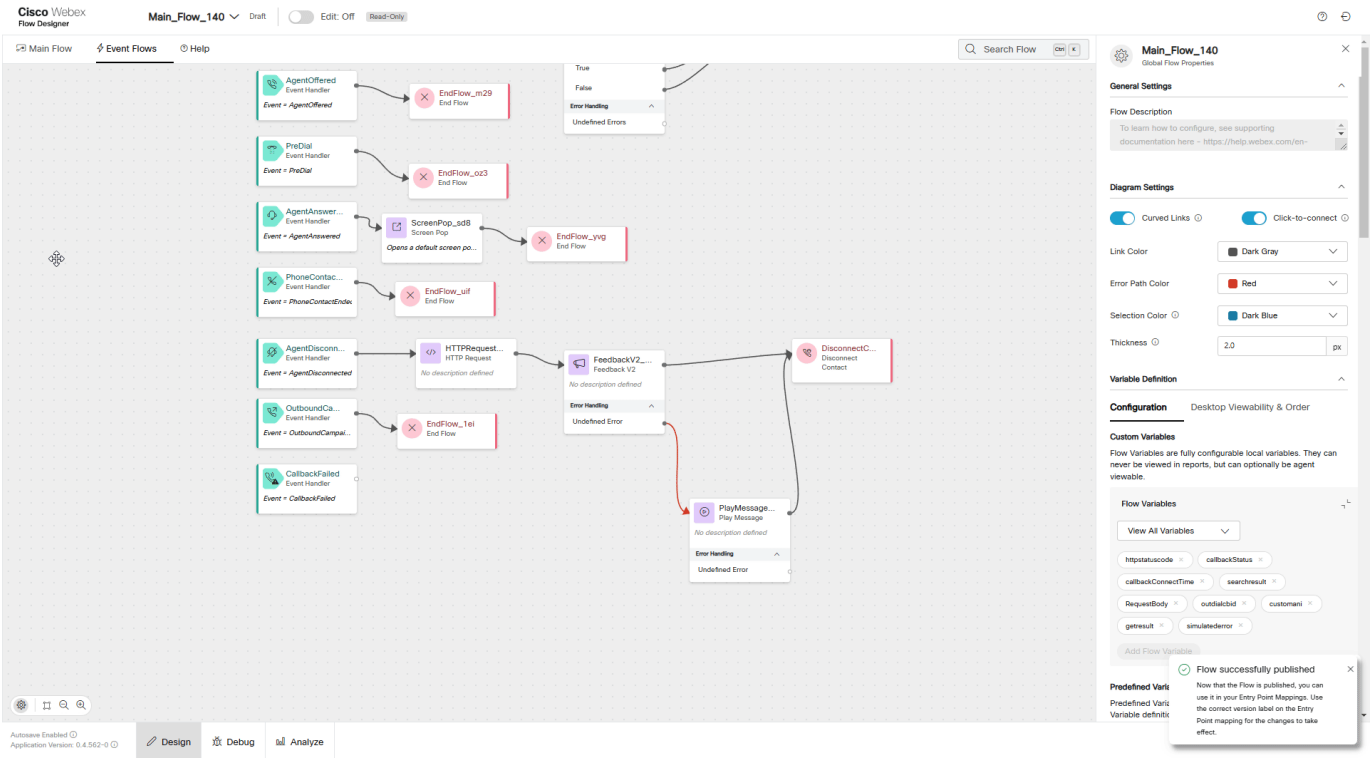
1. Make sure you're logged into Webex CC Desktop application as Agent **wxcclabs+agent_IDYour_Attendee_ID@gmail.com** and set status to **Available**.
2. Make a call to the Support Number and if success you should hear Welcome message and then accept the call by agent.
3. Upon accepting the call, a new 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 text you like then click **Save** and End the call.
5. Switch to the **Webhook.site** 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 **Even Flows** and delete **Screenpop** node
3. Validate the flow by clicking **Validate**, **Publish** and select the Latest version of the flow




**Congratulations on completing another mission where you have learnt how to use events in your flows.**

## 2.1.4 Mission 4: 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.

#### PRE-CONFIGURED ENTITIES

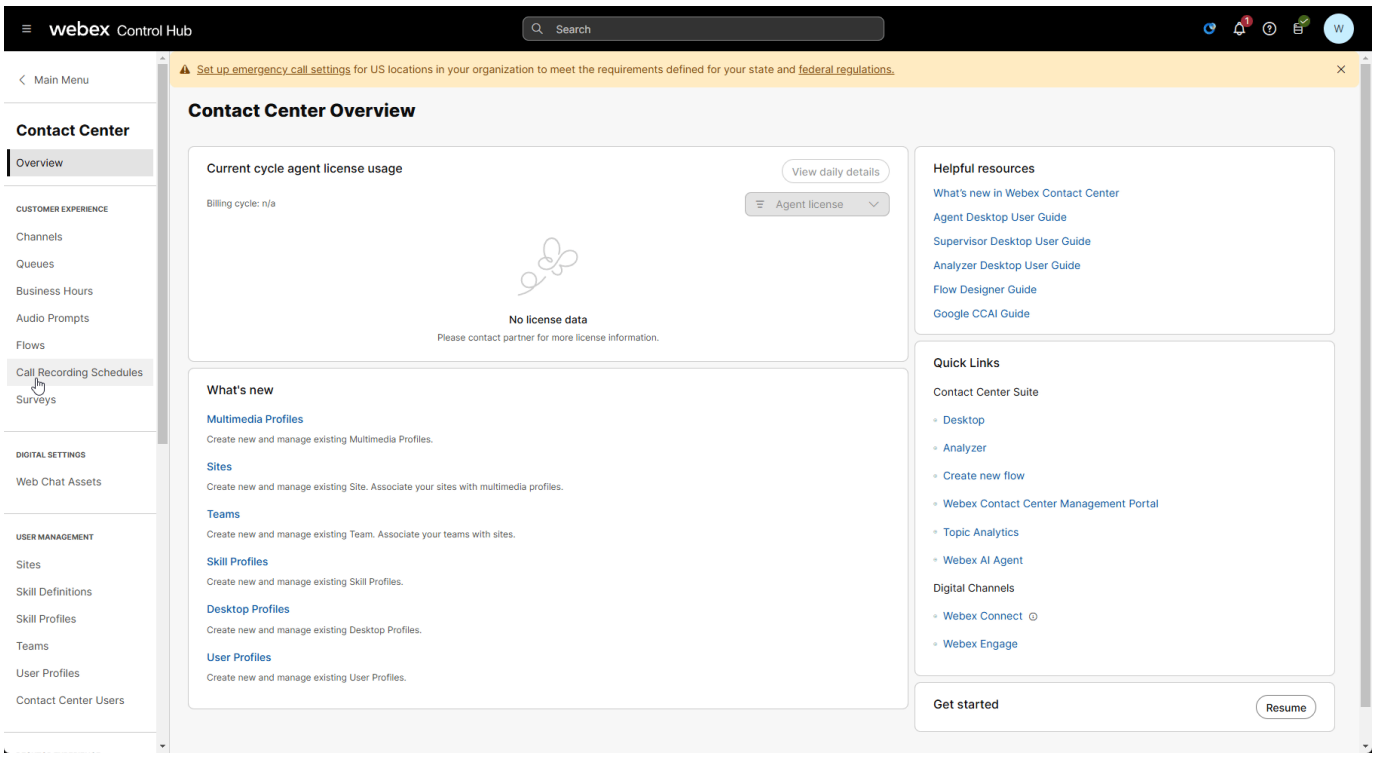
Survey: **PCS-2025**

System defined GlobalVariable: **Global_FeedbackSurveyOptIn**.

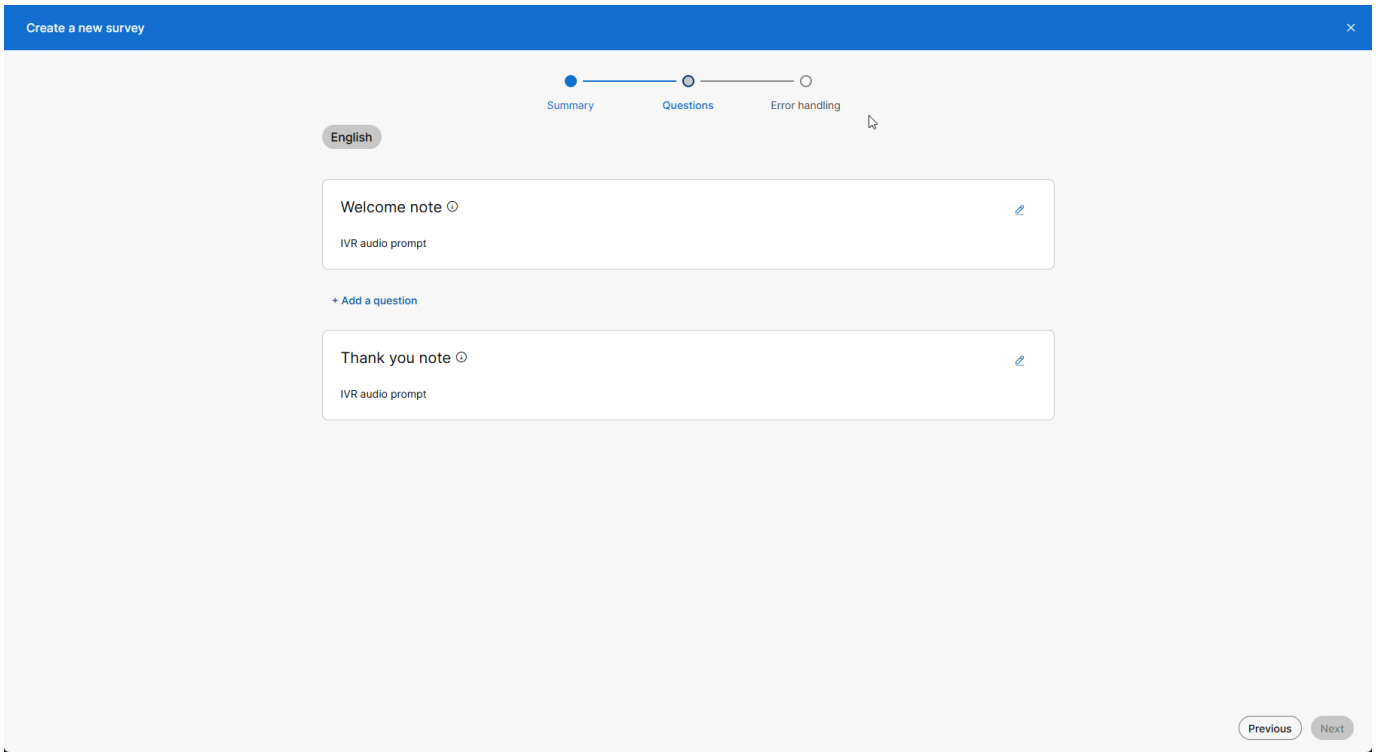
[Optional] In case you don't want to use pre-configured Survey you can configure your own. Expand below section to create your own Survey otherwise proceed to **Build** section below

**eate your own Survey [Optional]**

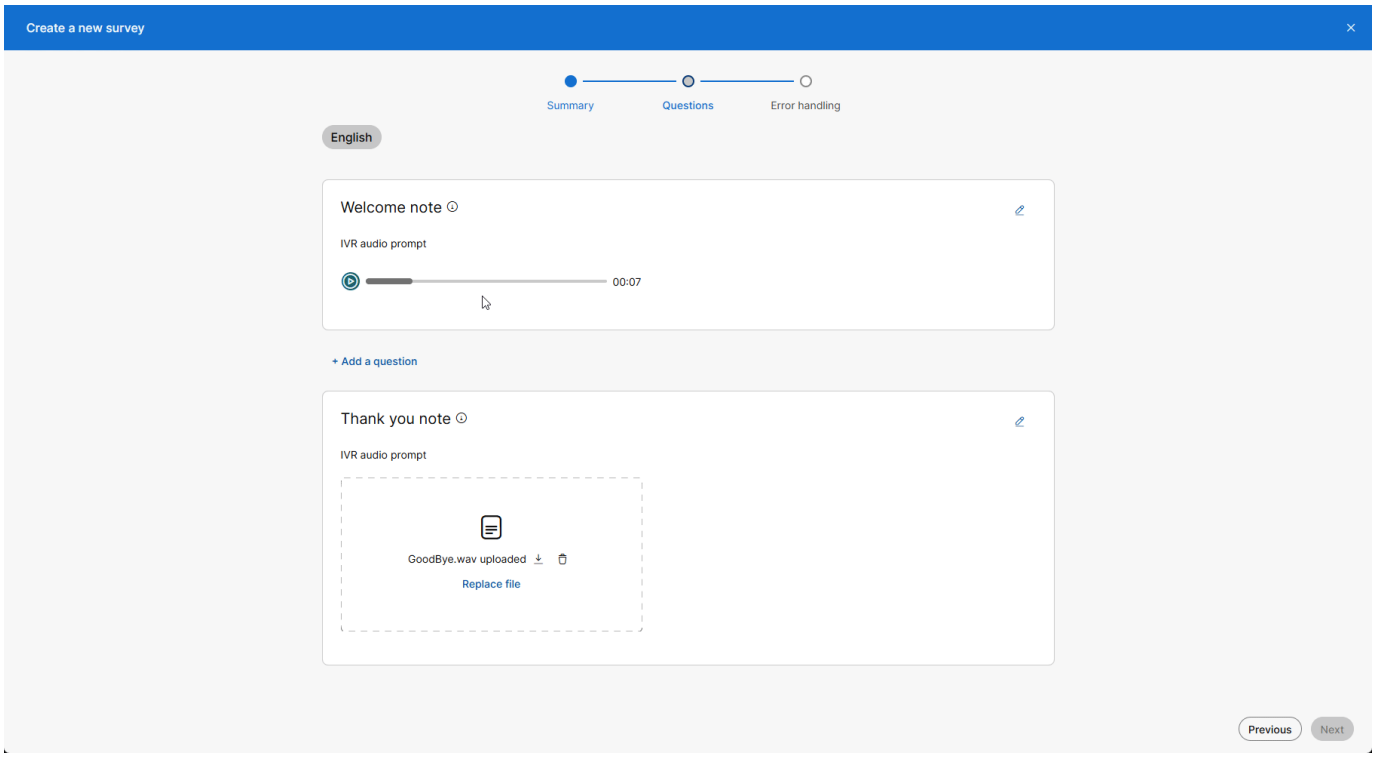
- In **Control Hub -> Contact Center** open a **Survey** configuration page under **Customer Expirience**. Then click **Create new survey**.
- Enter survey name in **Survey name** field. Make sure **IVR survey** is selected. Then click next




- Edit **Welcome note** and **Thank you note** by uploading the following files. Download files to your desktop prior uploading to survey.

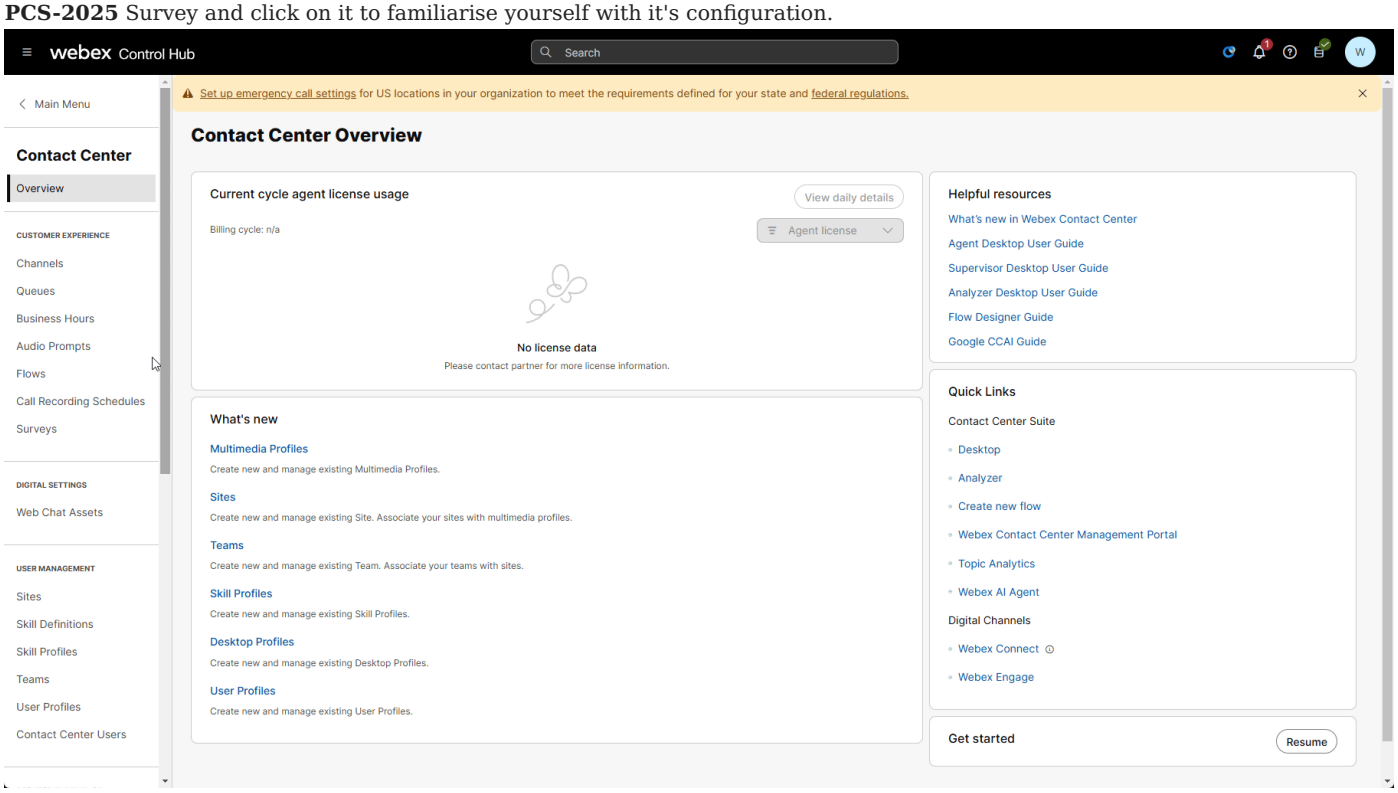



- Click on **Add a question** which is in the middle between **Welcome note** and **Thank you note**. Choose either NPS, CSAT or CES type of question.
- Upload respective audio prompts. Prompts can be downloaded from shared folder.
- Click *Next. You can ignore Error Handling configuration page. Click Save**

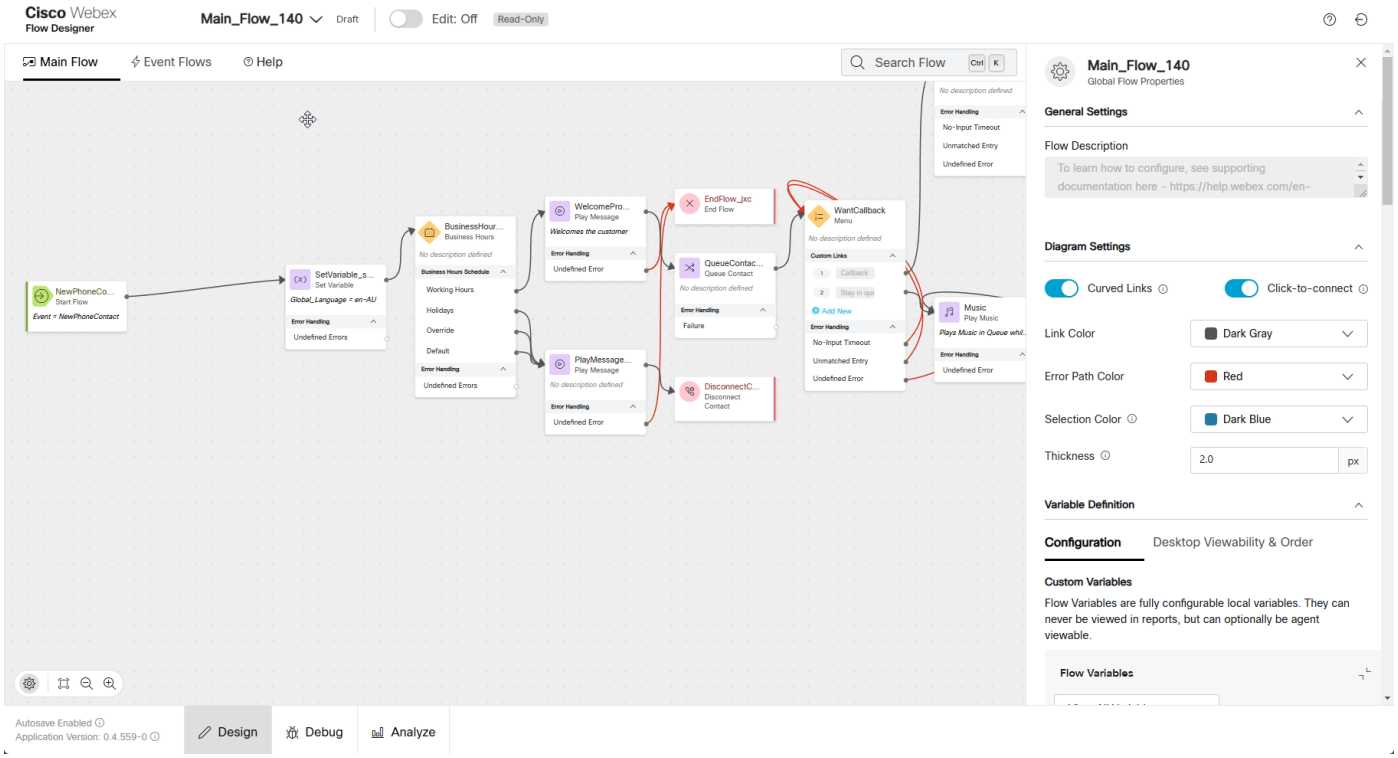


### Build

1. Switch to the Control Hub then go to **Contact Center**. Navigate to the **Surveys** under the **Customer Experience** section. Locate **PCS-2025** Survey and click on it to familiarise yourself with it's configuration.




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



4. Drag **Set Variable** node to canvas:

   Activity Name: **FeedbackSet**

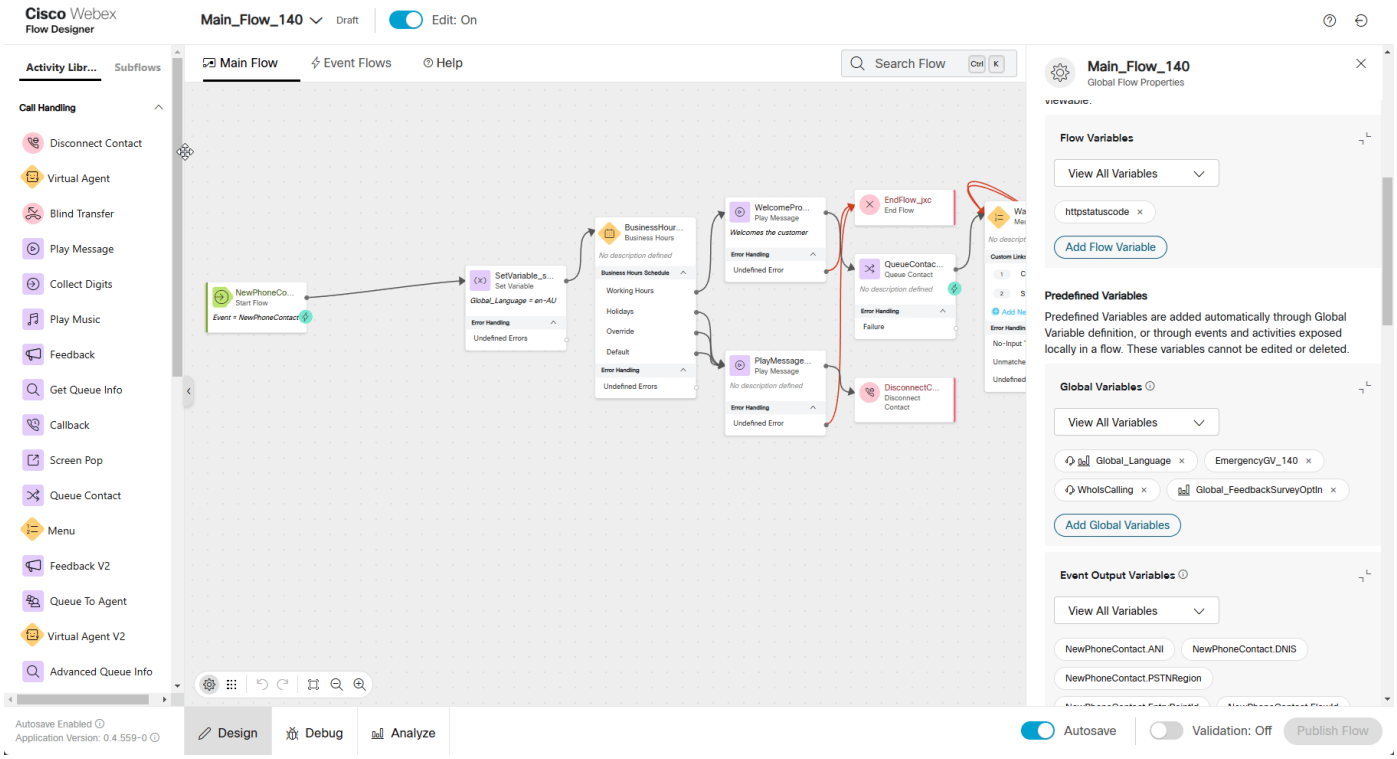
   Variable: **Global_FeedbackSurveyOptIn**

   Set Value: **true**

   Delete connection between **NewPhoneContact** and **Set Variable** on which we configured Language while doing the Main Lab.

   Connect **NewPhoneContact** to the front of the **FeedbackSet** node

   Connect **FeedbackSet** to the front of the **Set Variable** node




5. Open **Event FLows** tab and locate **AgentDisconected** node. If you completed previous mission you should have **HTTPRequest** node connected to it. Delete the connection between **HTTPRequest** node and **DisconnectContact**.

6. Drag **FeedbackV2** and **Play Message**

   **FeedbacV2**

   SurveyMethod -> VoiceBased: **PCS-2025**

   Connect **HTTPRequest** to **FeedbackV2** node

   Connect **FeedbackV2** node to **Disconnect** node

   Connect **FeedbackV2** Undefined Error to **Play Message** node

   **Play Message**

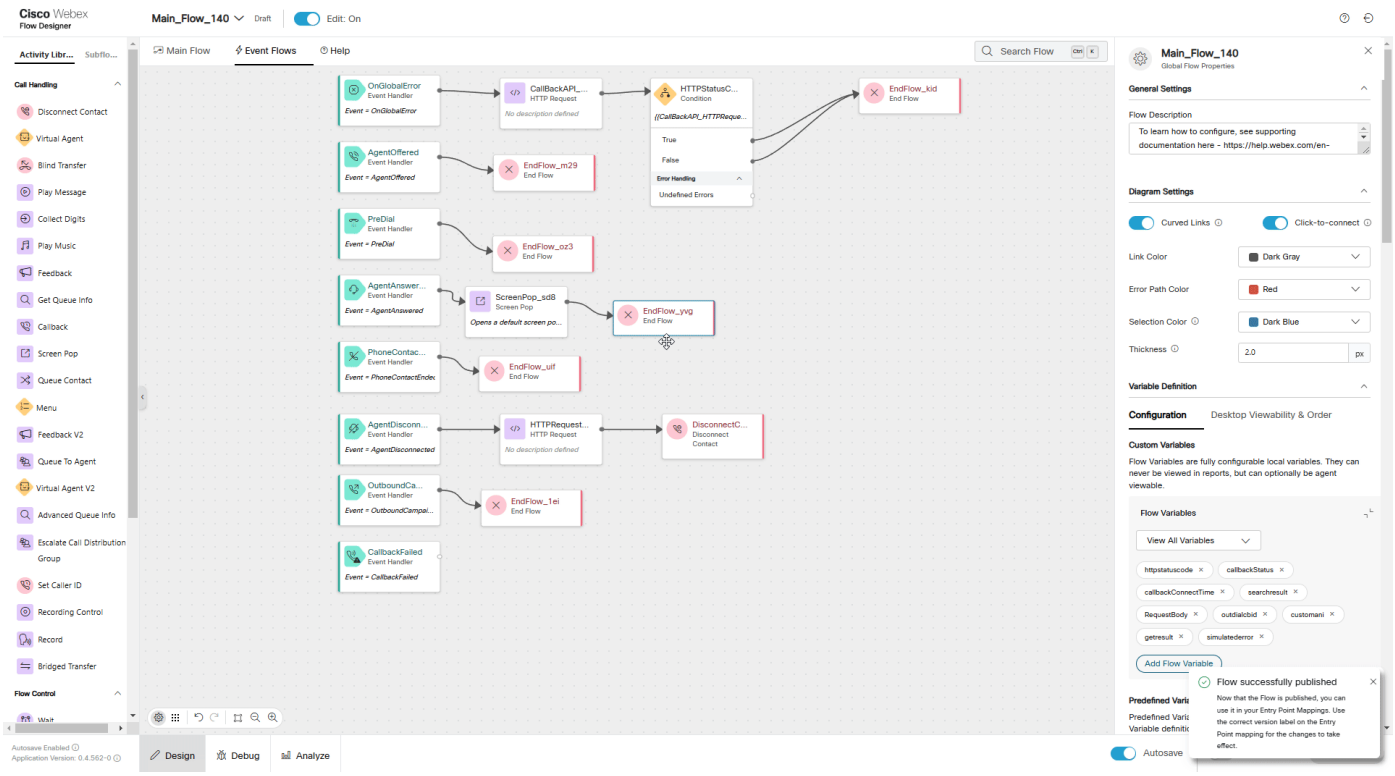
   Enable Text-To-Speech

   Select the 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

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




7. Validate the flow by clicking **Validate**, **Publish** and select the **Latest** version of the 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 **PCS-2025**. Complete the survey.
5. To check survey responses, switch to the **Control Hub** and navigate to the **Surveys** under **Customer Experience** section. Locate the **PCS-2025** 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 on completing another mission where we have learnt how Post Call Survey can be implemented.**

## 2.1.5 Mission 5: 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

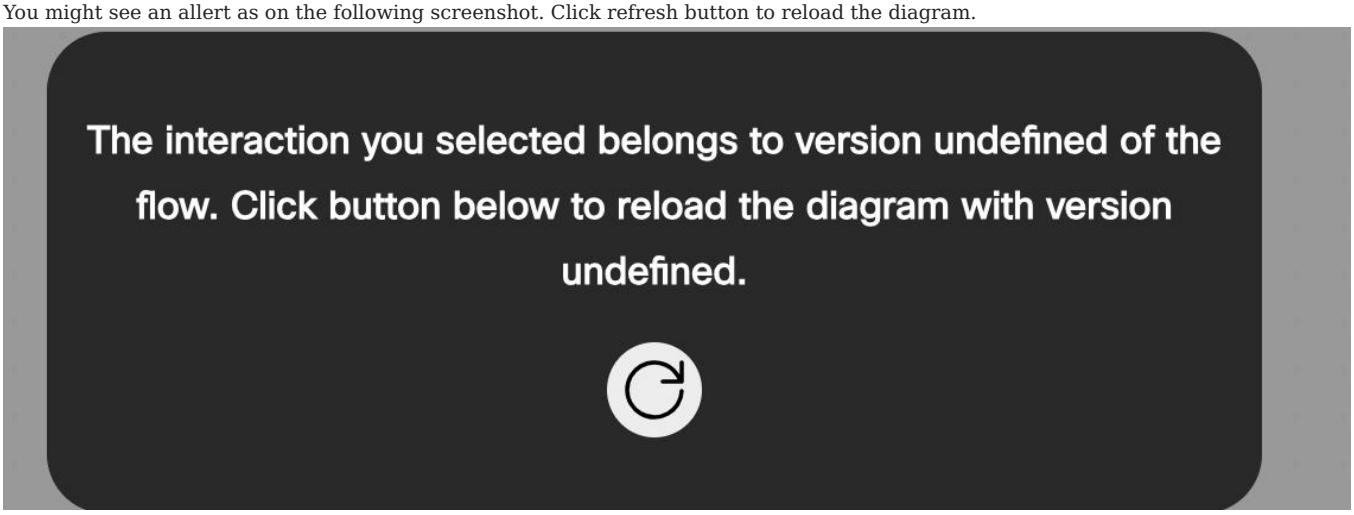
1. Switch to the Flow Designer and open your flow, **Main_Flow_Your_Attendee_ID**. Then, click the **Debug** button at the bottom of the Flow Designer.
2. You can view the calls you've made today during the previous exercises. Please click on the one at the top.

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

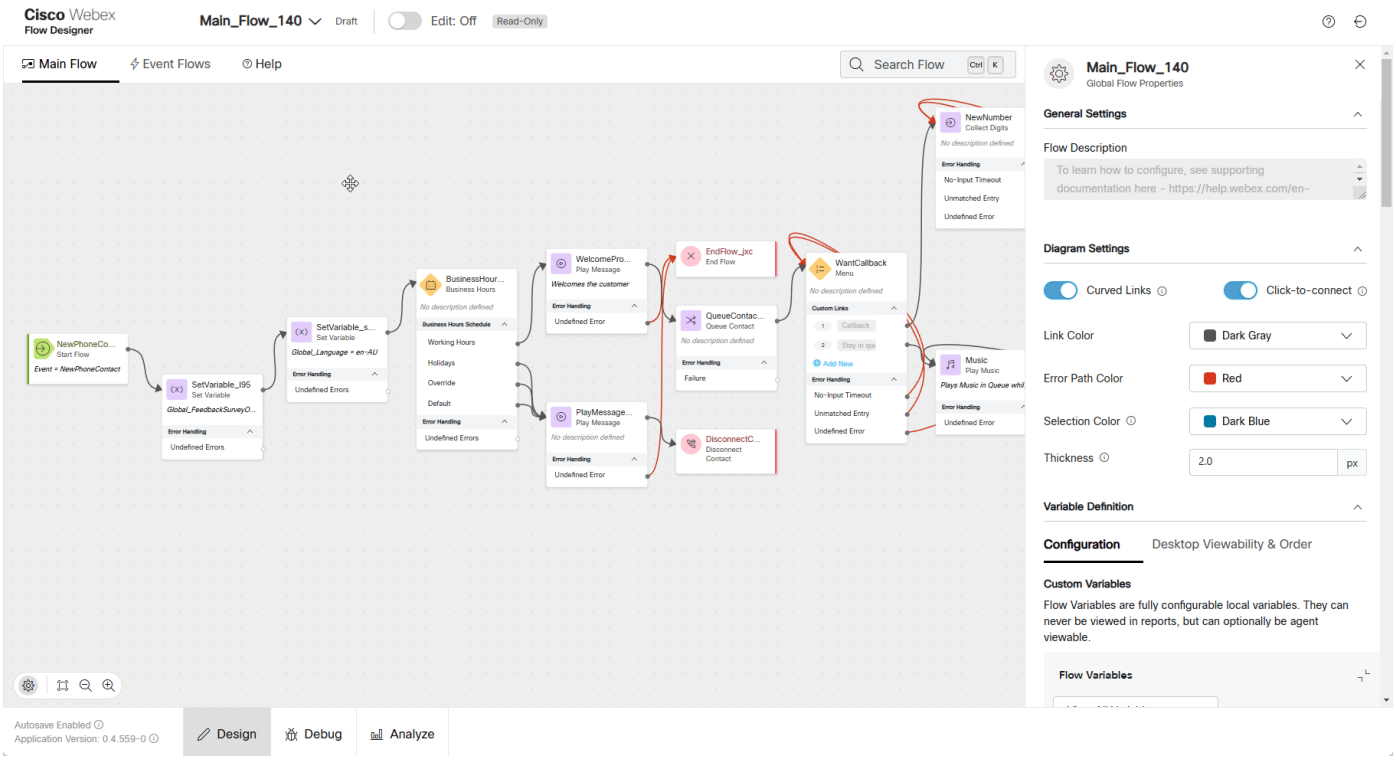
b. Filter by **Date Range** and by **Label**

**Note**

You might see an allert 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]**

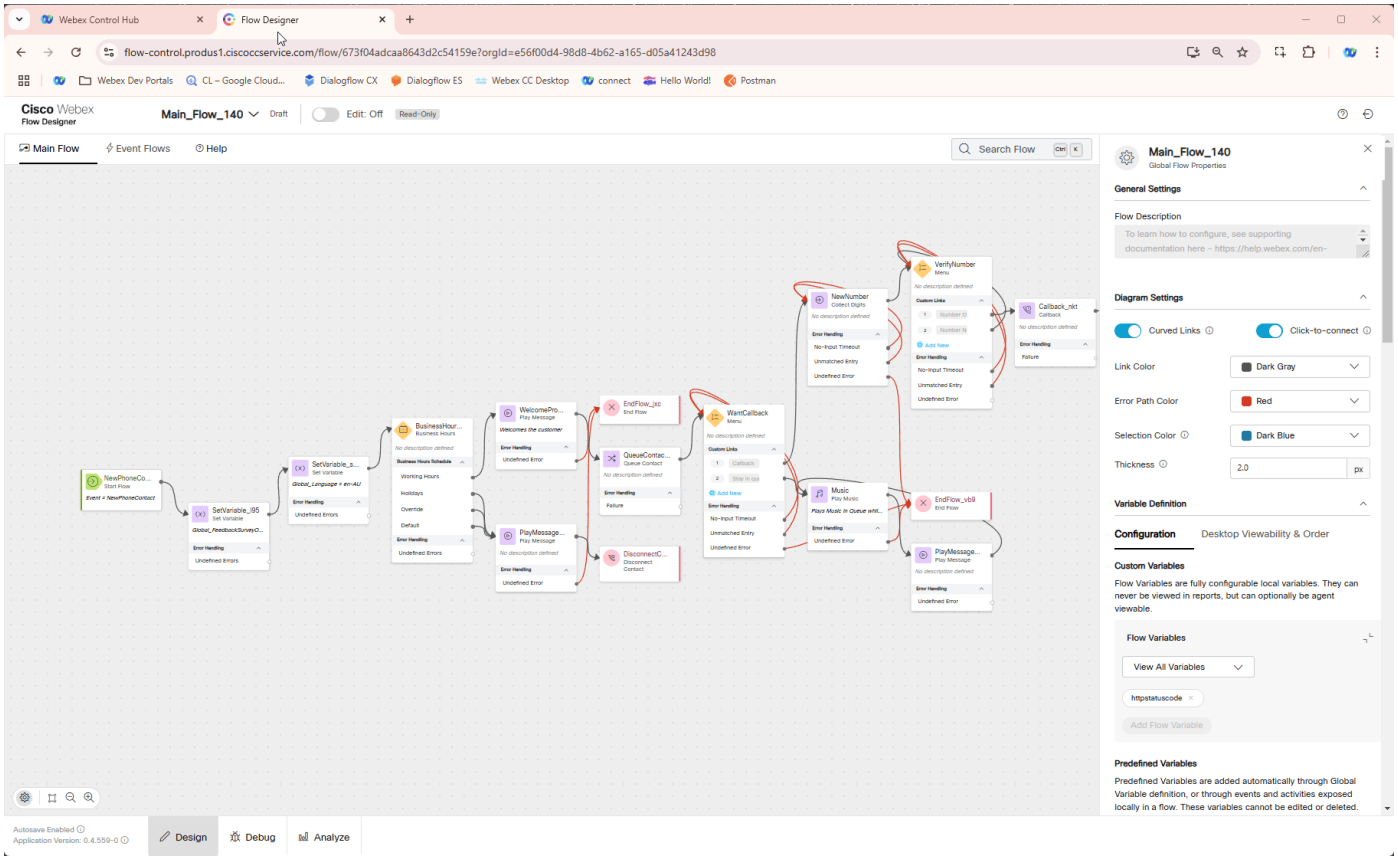
### Why Flow Analytics is Important?

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.

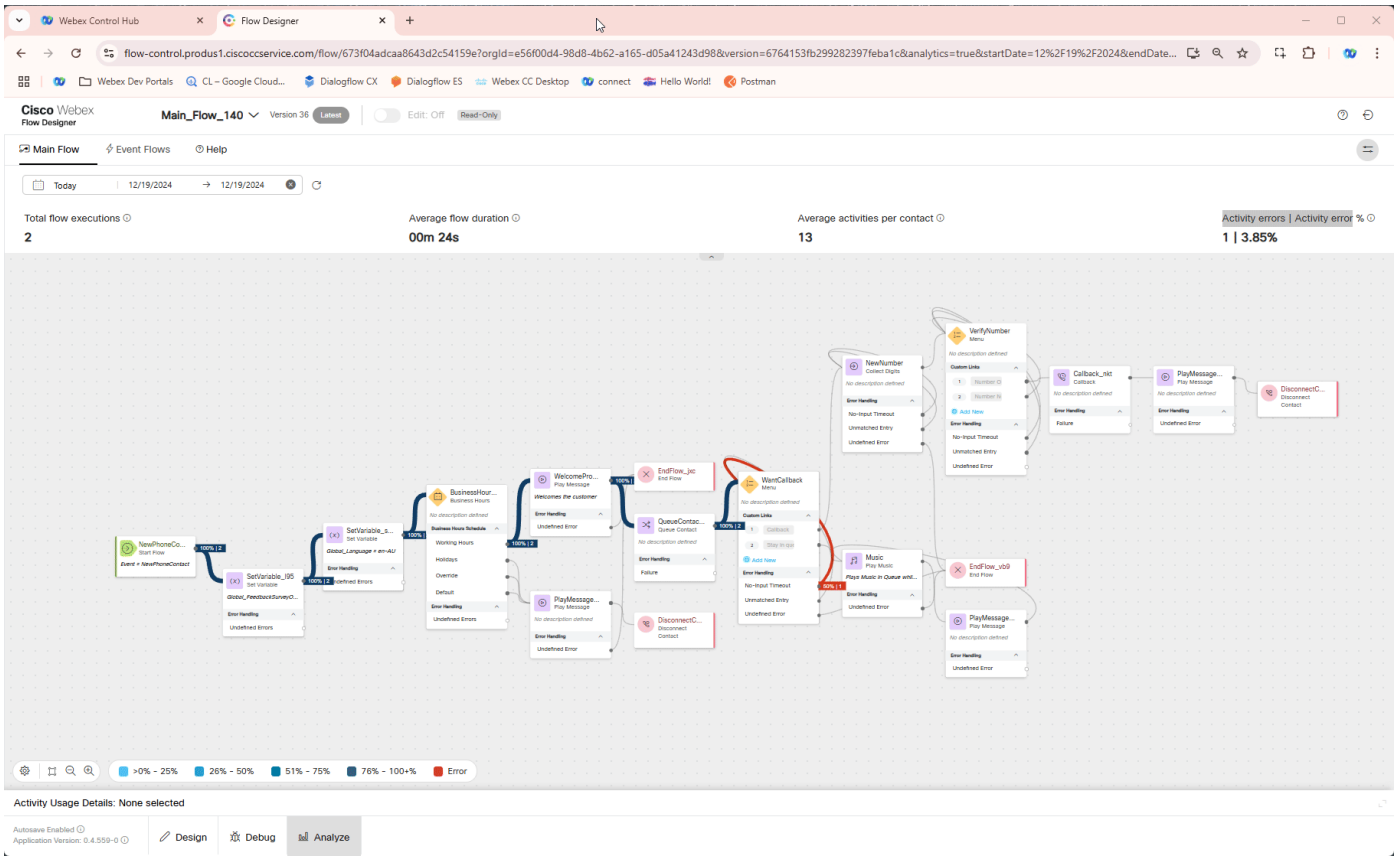
### HOW TO USE THE FLOW ANALYTICS TOOL

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:

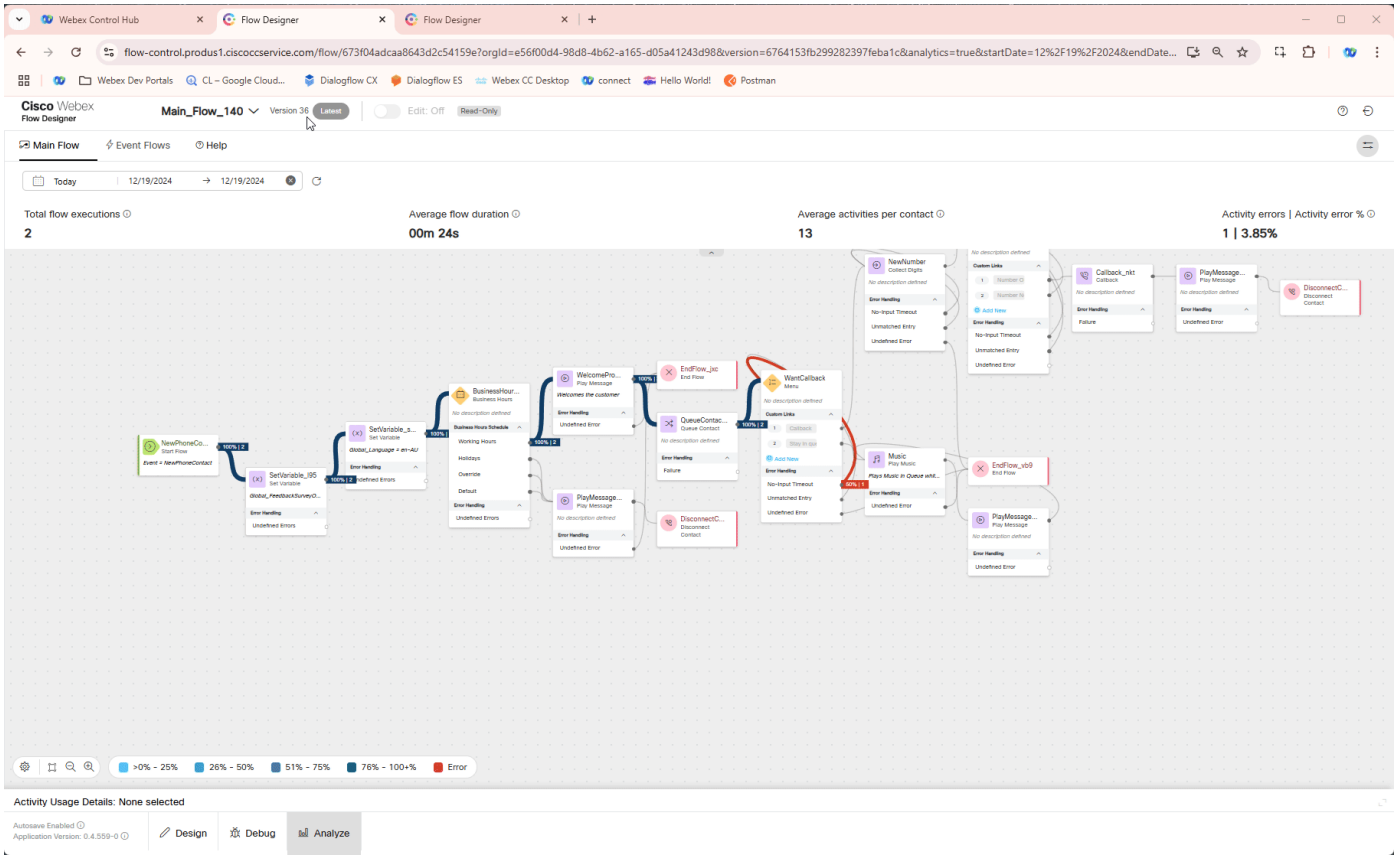
- Total flow Executions
- Execution paths and their frequency
- Avarage flow duraion
- Average activities per contact
- Activity errors | Activity error %




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 withing 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.

## 2.2 Conclusion

We hope you enjoyed the hands-on experience of the 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, and language-based routing.

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

- **Using Business Hours** to introduce flexibility and adaptability to your flows.
- **Leveraging Event Flows** to dynamically handle real-time events.
- **Designing and implementing a Post Call Survey** to gather actionable customer feedback.
- 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 Part 1 of the lab, 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:

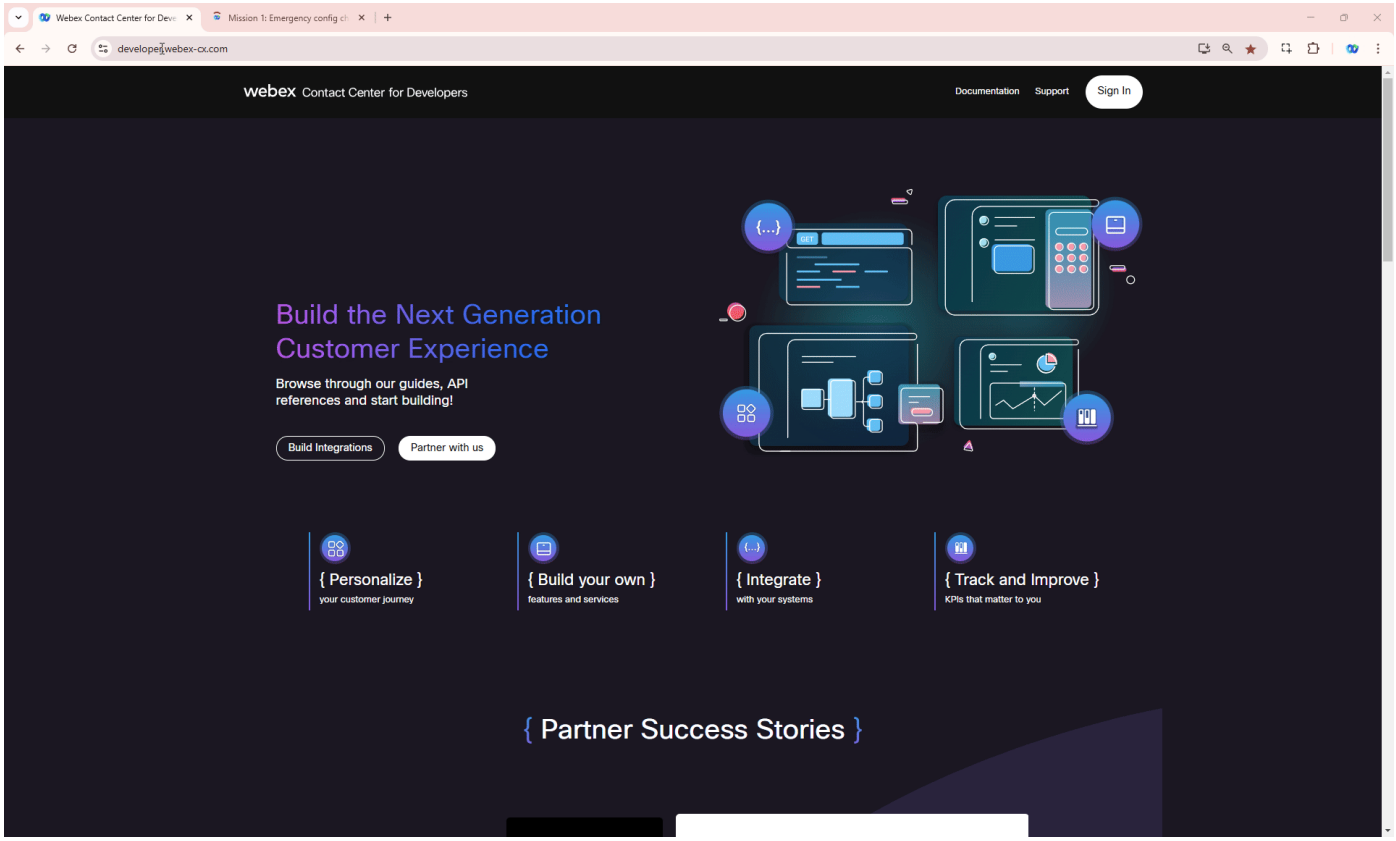
1. **GET – Retrieving Information Analogy**: Checking your bank balance at an ATM. You request information, and the system provides it without making any changes. **Example Use Case**: Retrieving a customer's interaction history in Webex Contact Center before routing their call.
2. **POST – Creating 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

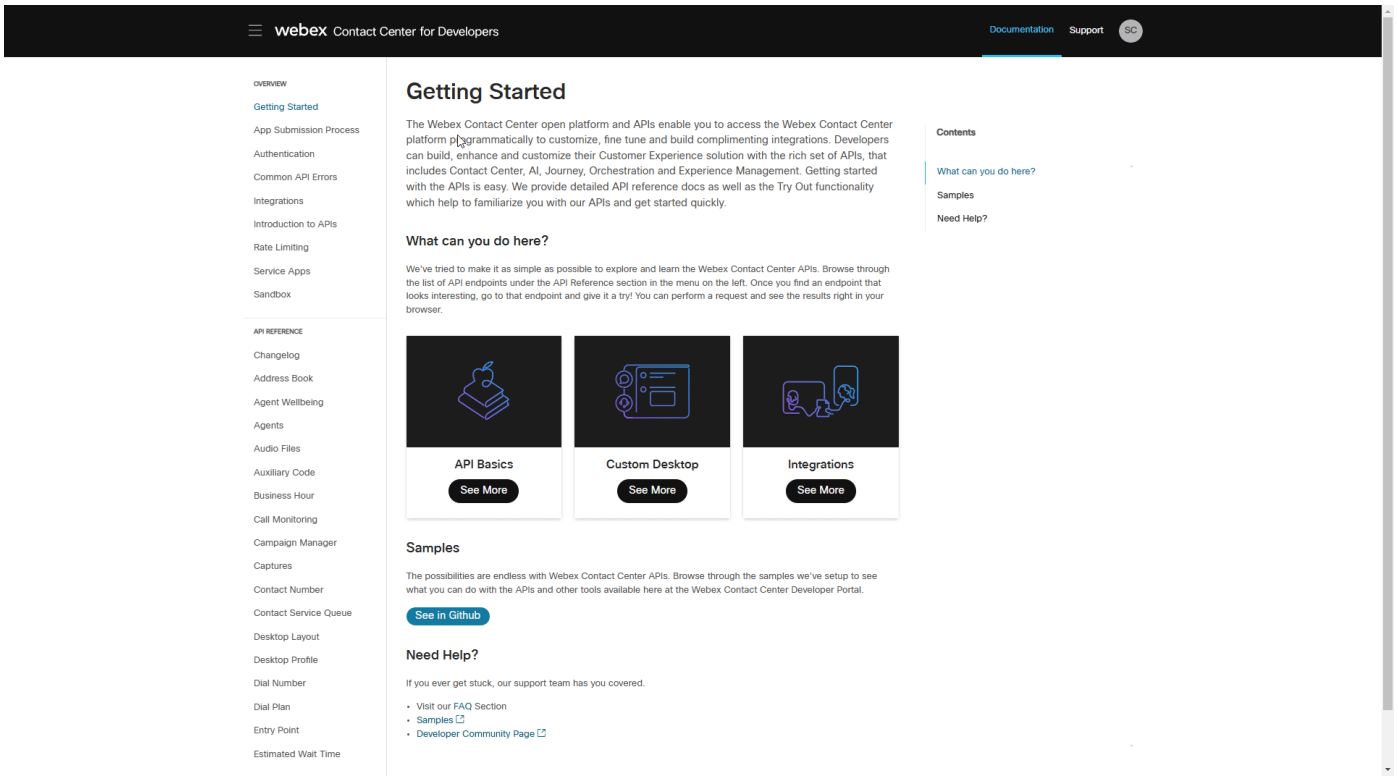
1. Open **Developer Portal** and click on **Sign In**. Your login will be of the format **wxcclabs+admin_IDYour_Attendee_ID@gmail.com**. You will see another login screen with OKTA on it where you may need to enter the email address again and the password provided to you.
2. Click on **Documentation** which is on top right corner of the portal page.




3. On Menu pannel on the left, scroll down to **API Reference** section and click on **Adress Book**. Observe 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 **Try Out**.



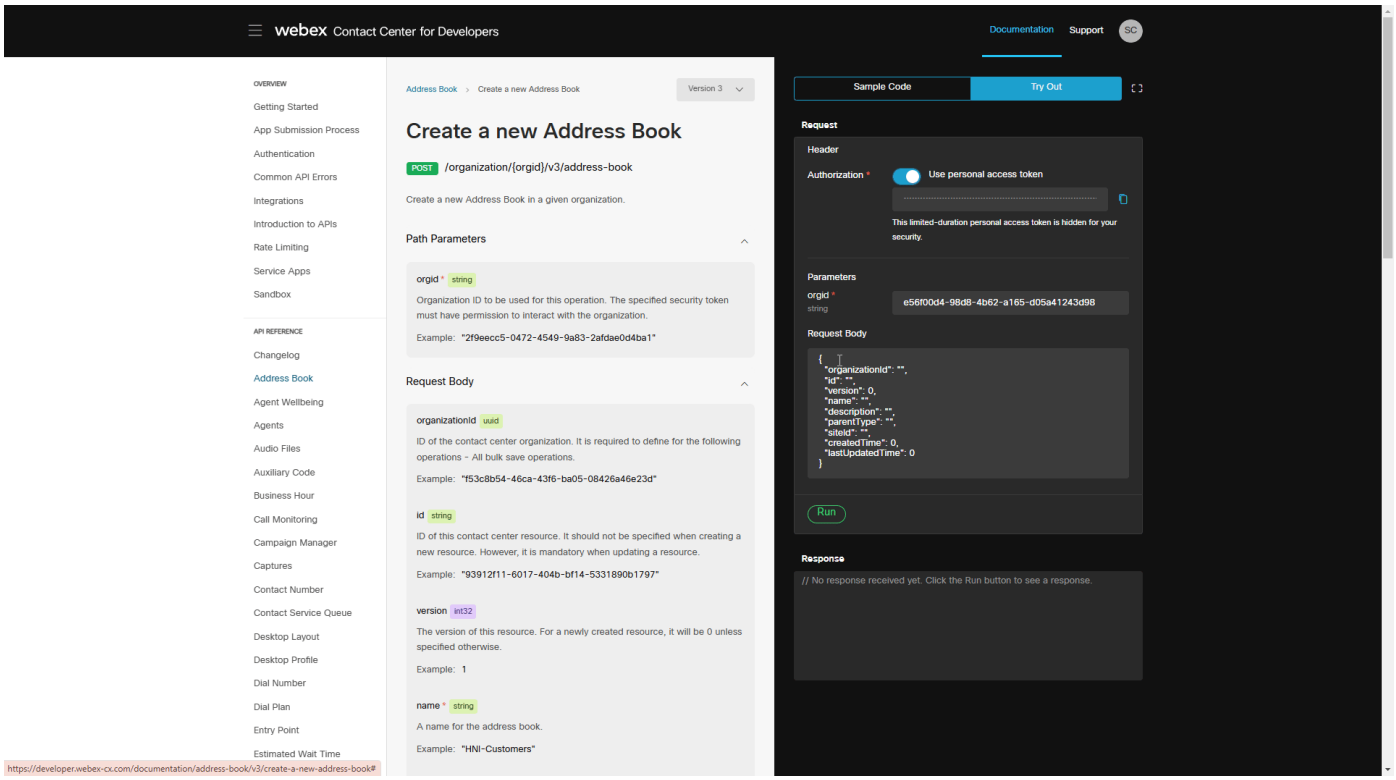

5. Clear **Request Body** content. Paste the following body and replace the with your **attendee ID**. Click on **Run**.

Request Body:

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

Expected Response: **201 Response**

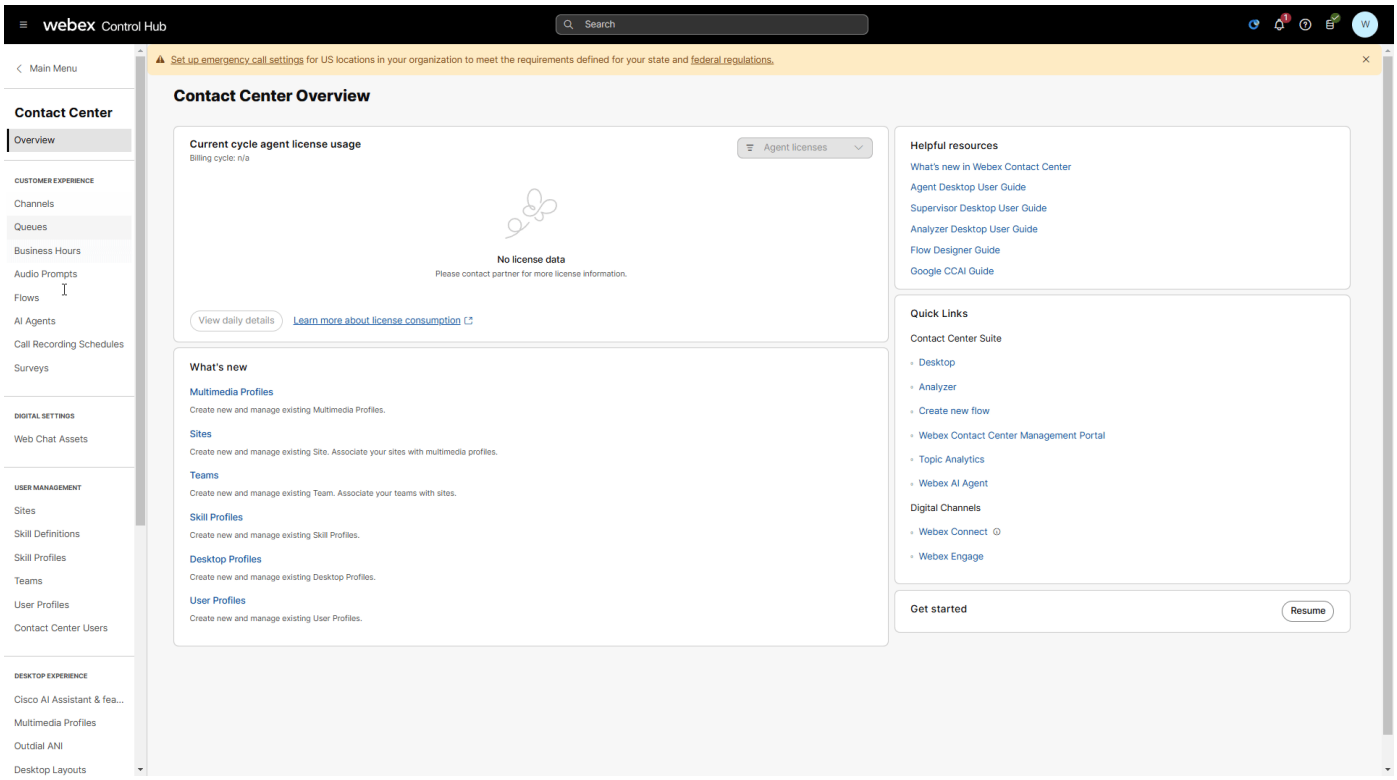
```
{
  "organizationId": "e56f00d4-98d8-4b62-a165-d05a41243d98",
  "id": "4aa50a6b-a520-4221-bc9d-a050c111061f",
  "version": 0,
  "name": "AddressBook_140",
  "parentType": "ORGANIZATION",
  "createdTime": 1738585491913,
  "lastUpdatedTime": 1738585491913
}
```



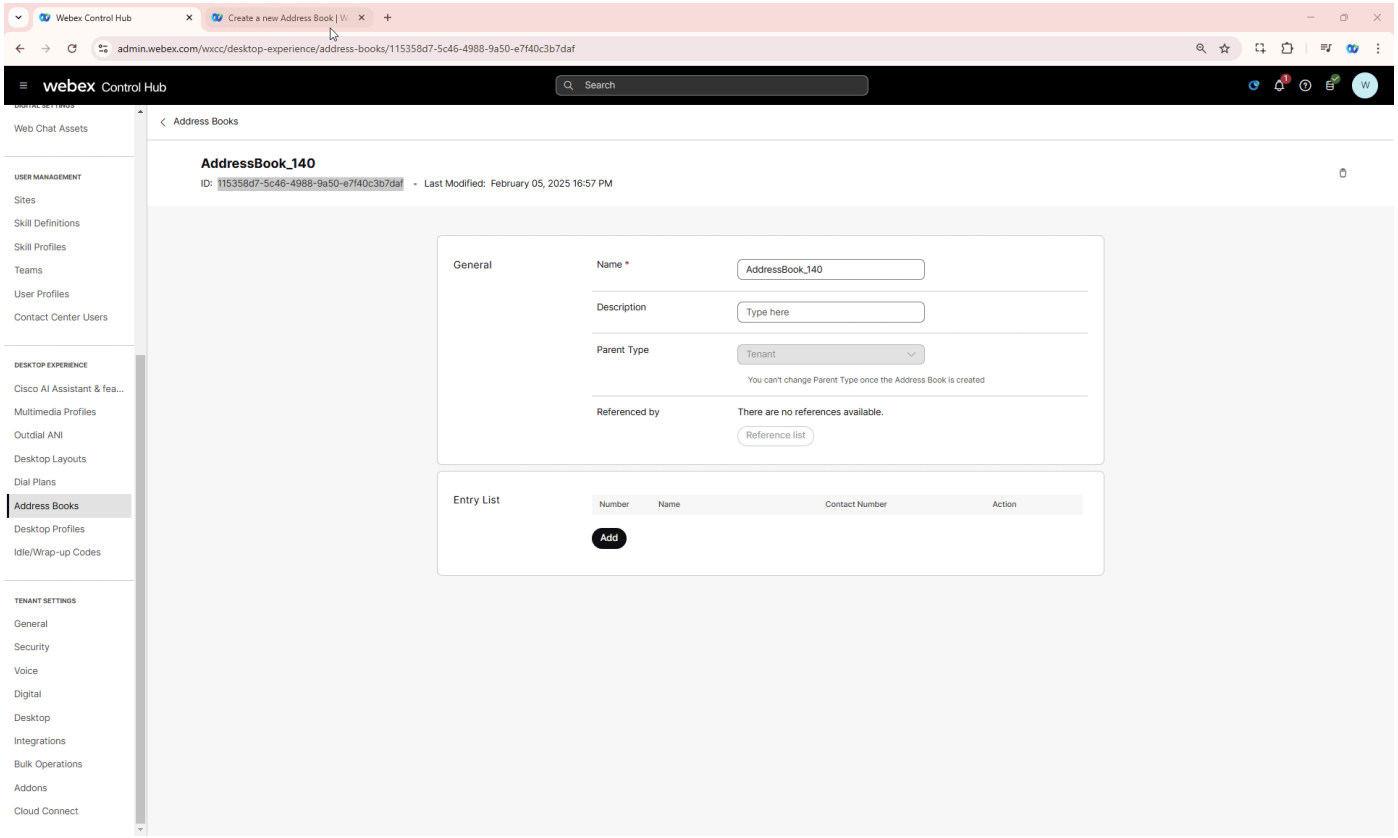

6. Switch to Webex Control Hub and navigate to **Address Book** under **Desktop Experience Section**. Locate your new created **AddressBook_Your_Attendee_ID**

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.



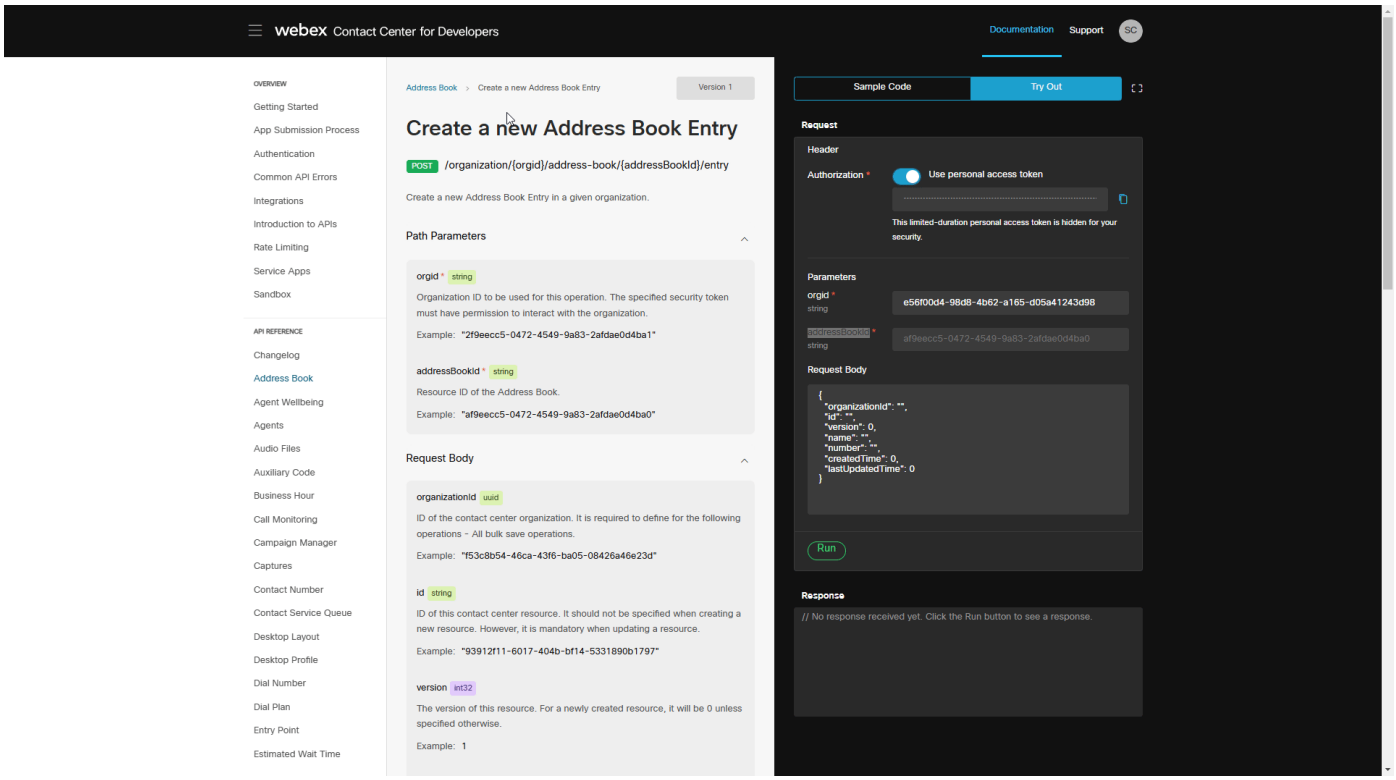

10. Click on **Create a new Address Book Entry**, then switch to **Try Out** tab within the same page.
11. In the **Parameters** section paste **ID** you copied on **Step 8** of the current mission.
12. Clear **Request Body** content and paste the following body, then click on **Run** button.

    Request Body:

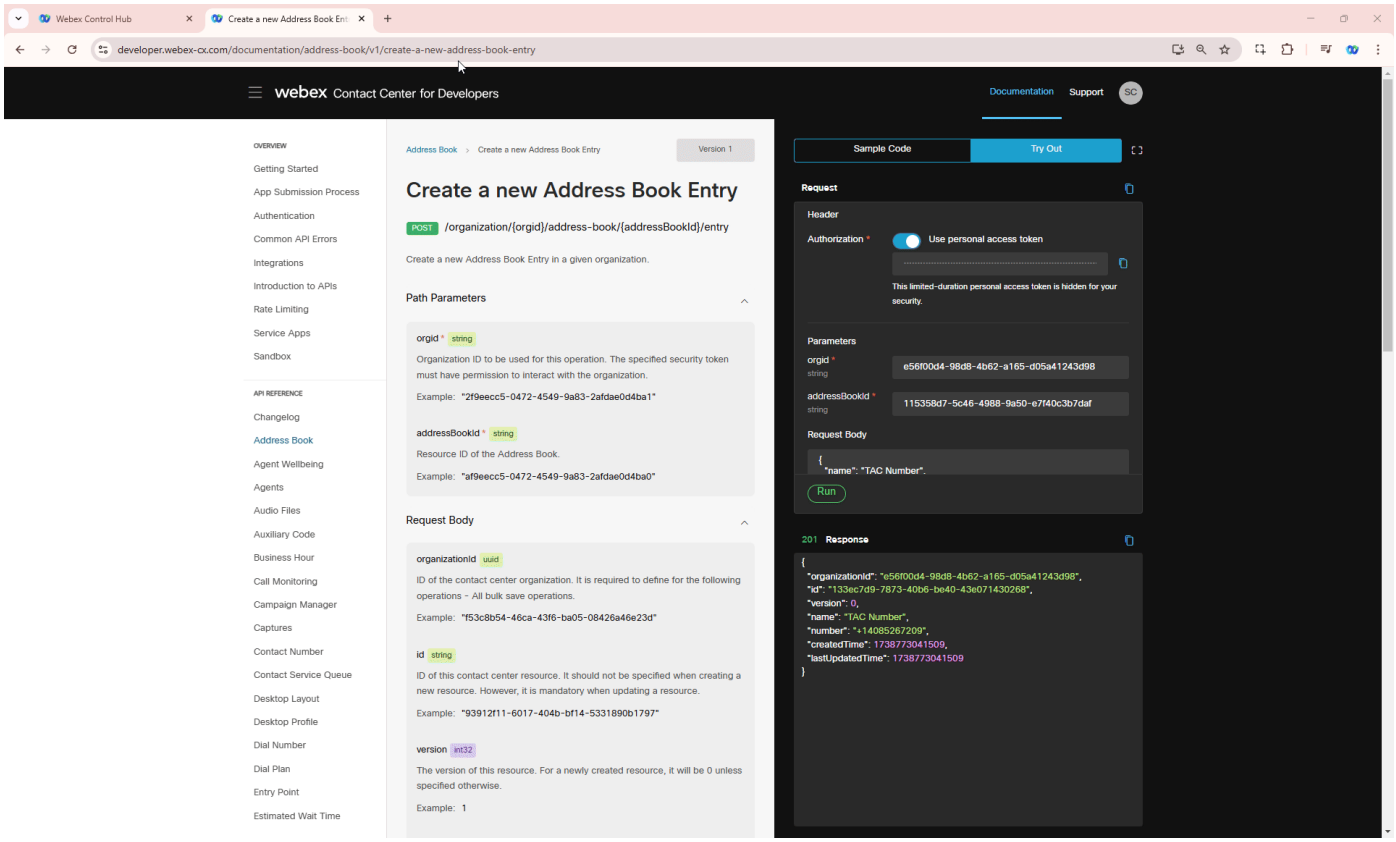
```
{
  "name": "TAC Number",
  "number": "+14085267209"
}
```

    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
}
```



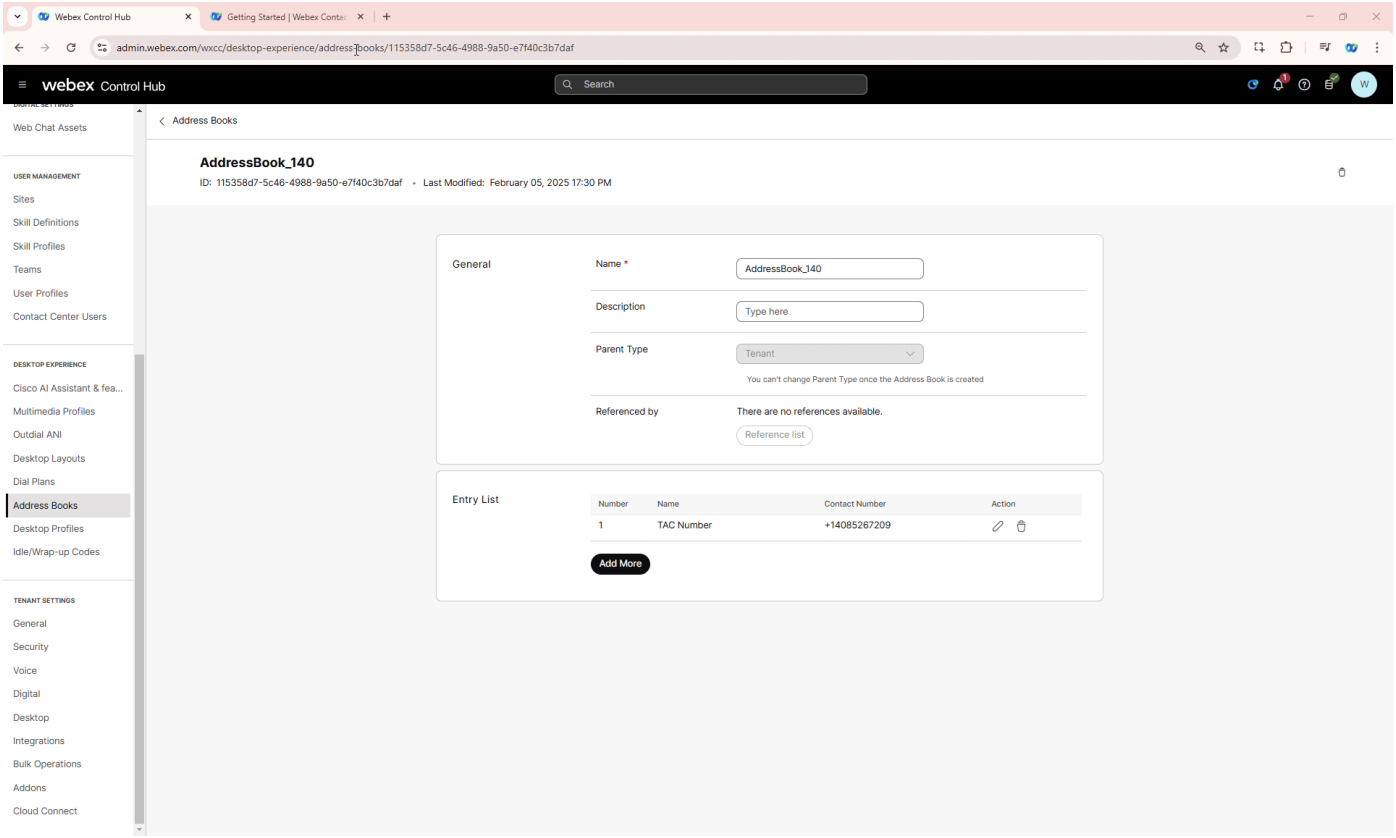

13. Switch to Webex Control Hub. Your **Addres Book** configuration page should still be open. **Refresh** the page.
14. But if not open, locate and open your **AddressBook_Your_Attendee_ID**
15. You should see your new created **Entry List** with Name **Tac Number** and Contact Number **+14085267209**.



#### RETRIEVE ADDRESS BOOK ENTRY BY ID (GET)

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

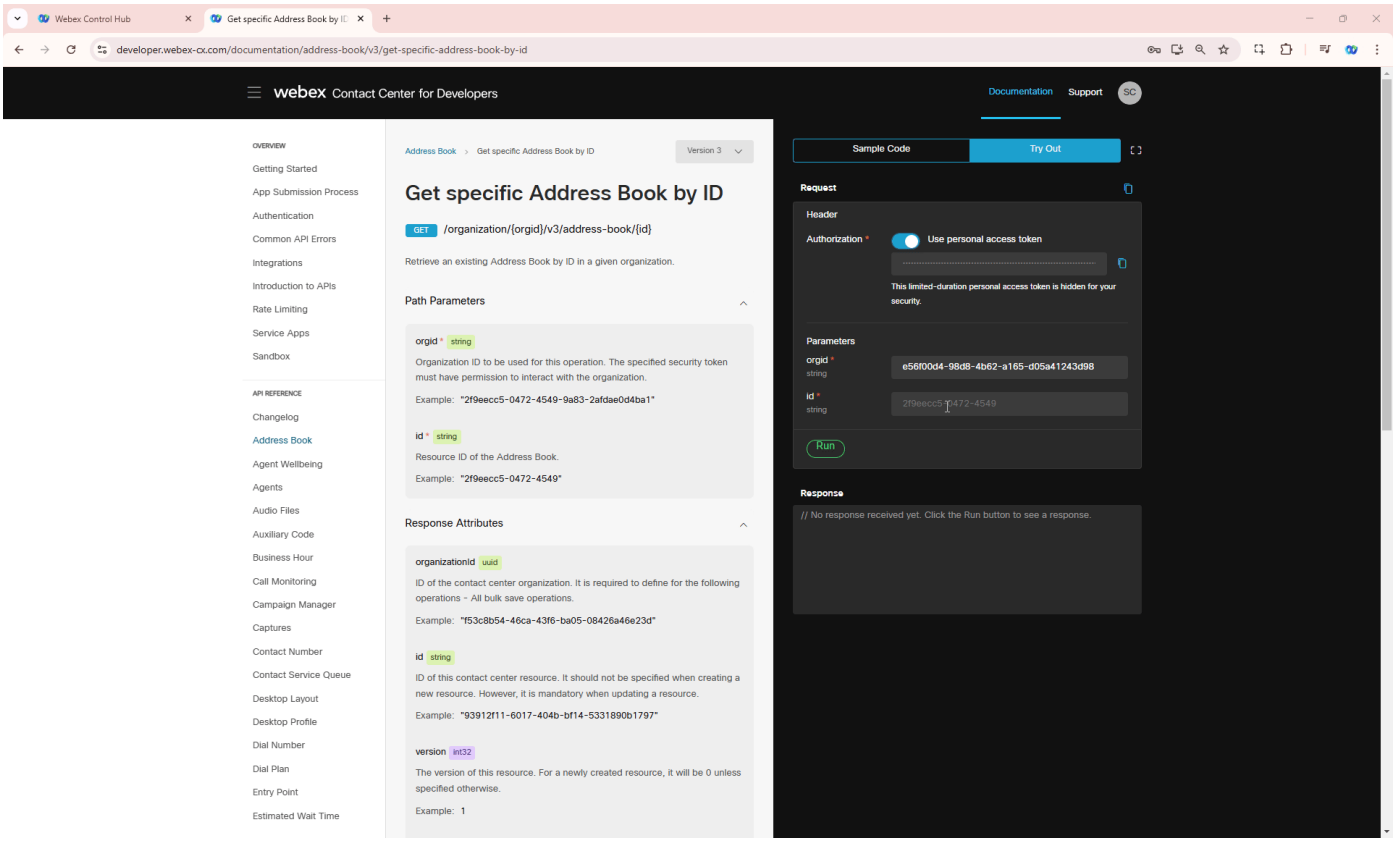
1. Switch to **Developer Portal** and select **Address Book** again from left menu pane.
2. Locate and open **Get specific Address Book by ID**, then switch to **Try Out** tab.



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

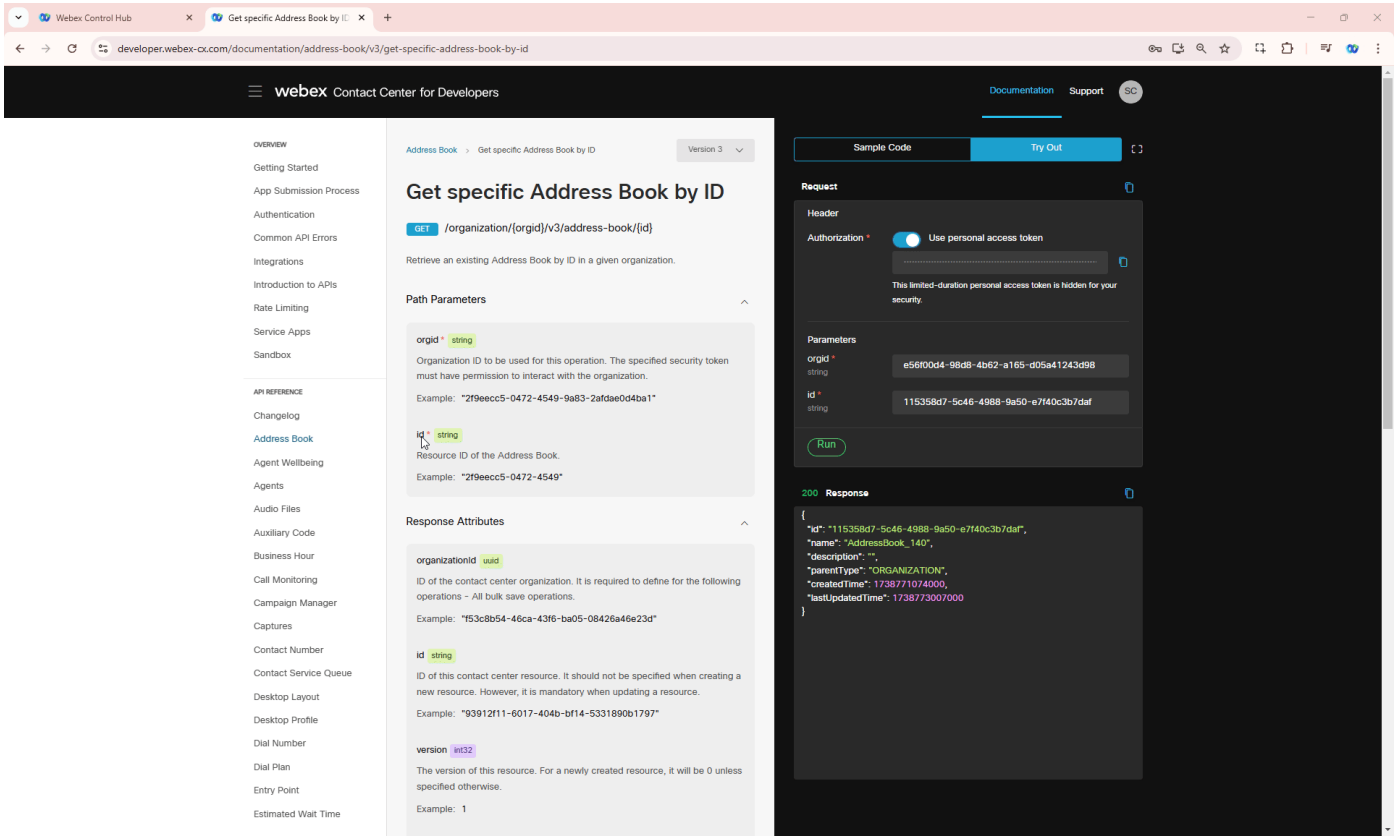
   Expected Response: **200 Response**


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


UPDATE ADDRESS BOOK DESCRIPTION BY USING PUT

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 **Try Out** tab.




3. Paste the same **AddressBook_AddressBook_** ID into **id** cell of **Parameters** section. You can quickly copy **name** and **id** 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 AddressBook_AddressBook_ **ID**

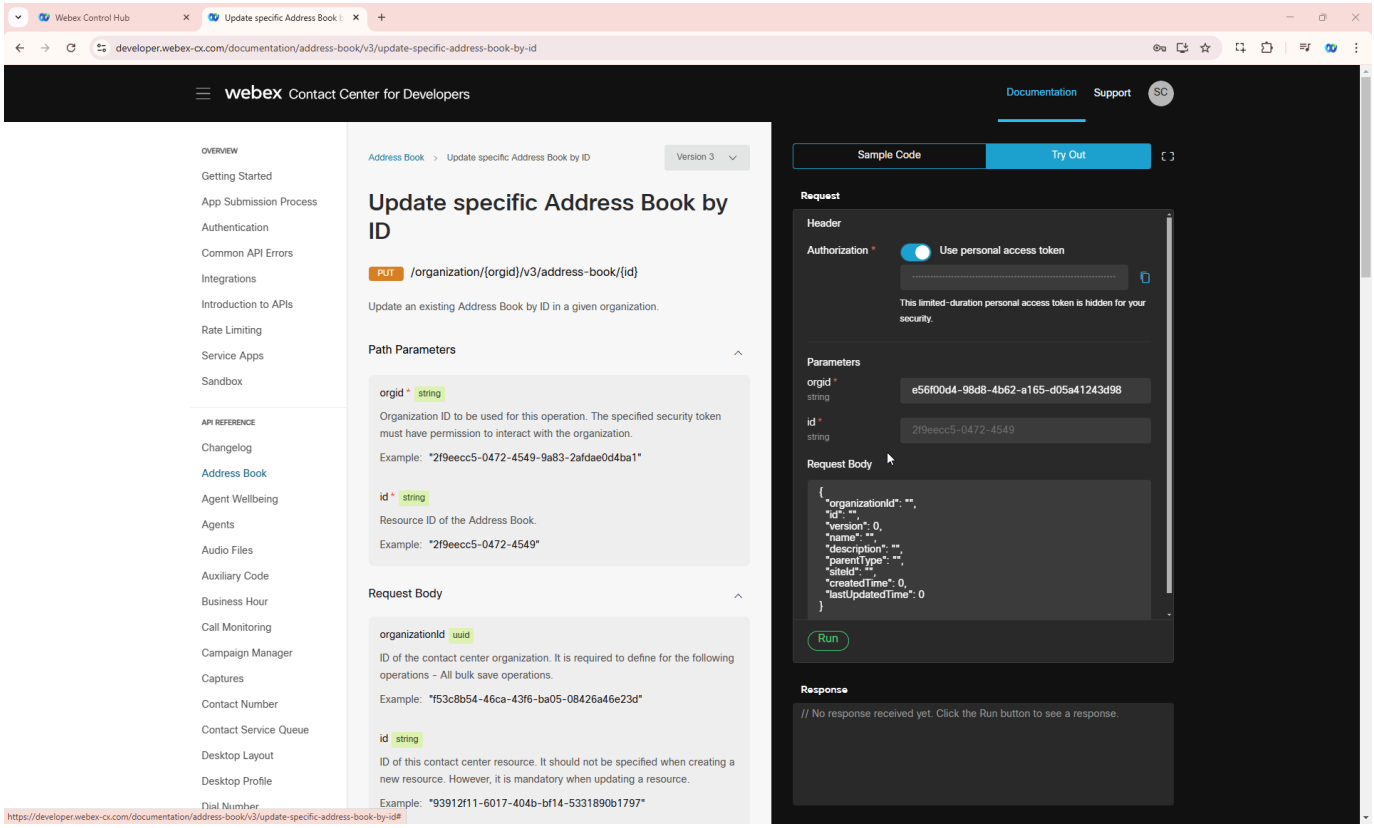
   Request Body:

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

   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. You **Addres Book** configuration page should still be open. **Refresh** the page to validate Description change.

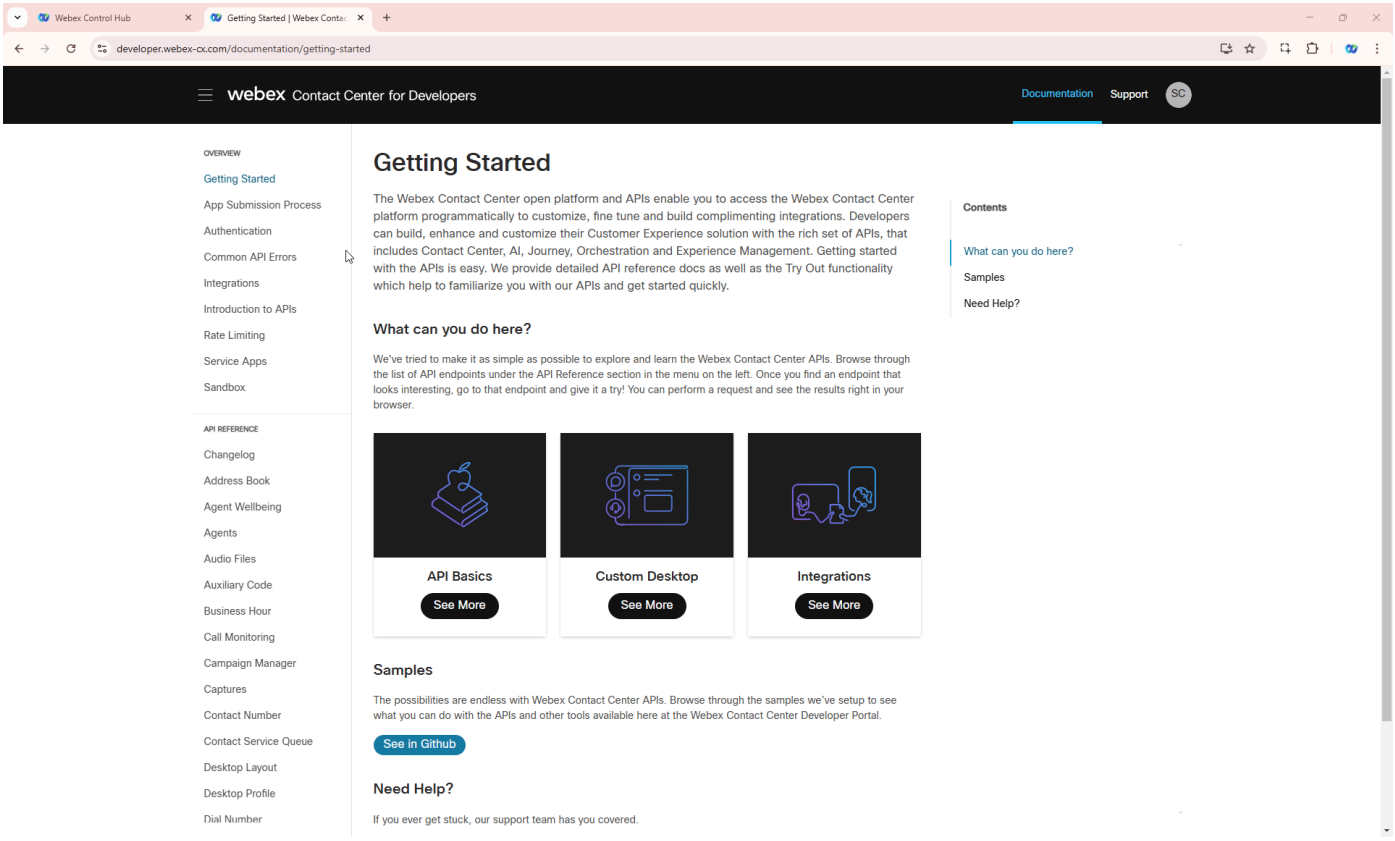


USE SEARCH API TO RETRIEVE DATA FROM ANALYZER DB.

**Note**

When working with Webex Contact Center (WxCC) 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 **Search** from **API REFERENCE** menu
2. Click on **Search tasks** and then switch to **Try Out** tab




3. 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: 1738833921000 #This can be set to Date.now() - (days * 24 * 60 * 60 * 1000) for lookback in days
    to: 1738834701000 #This can be set to Date.now() in millis
    filter: {
      #Filter the type of Task
      and: [
        { channelType: { equals: telephony } } #Telephony calls only
        { origin: { equals: "+14694097607" } } #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: From **Thursday, February 6, 2025 9:25:21 AM** to **Thursday, February 6, 2025 10:38:21 AM GMT+01:00**.

b. Telephony inbound calls only.

c. Calls only from **+14694097607**.

d. Ended calls only.

e. Calls that were assigned to an owner (agent).

Expected Response: **200 Response**


```
{
  "data": {
    "taskDetails": {
      "tasks": [
        {
          "id": "d1364618-49a4-41f5-8b5f-a8da4d12e56c",
          "status": "ended",
          "totalDuration": 35562,
          "origin": "+14694097607",
          "destination": "+14694096861",
          "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. **ID** of the call

b. Status of the call

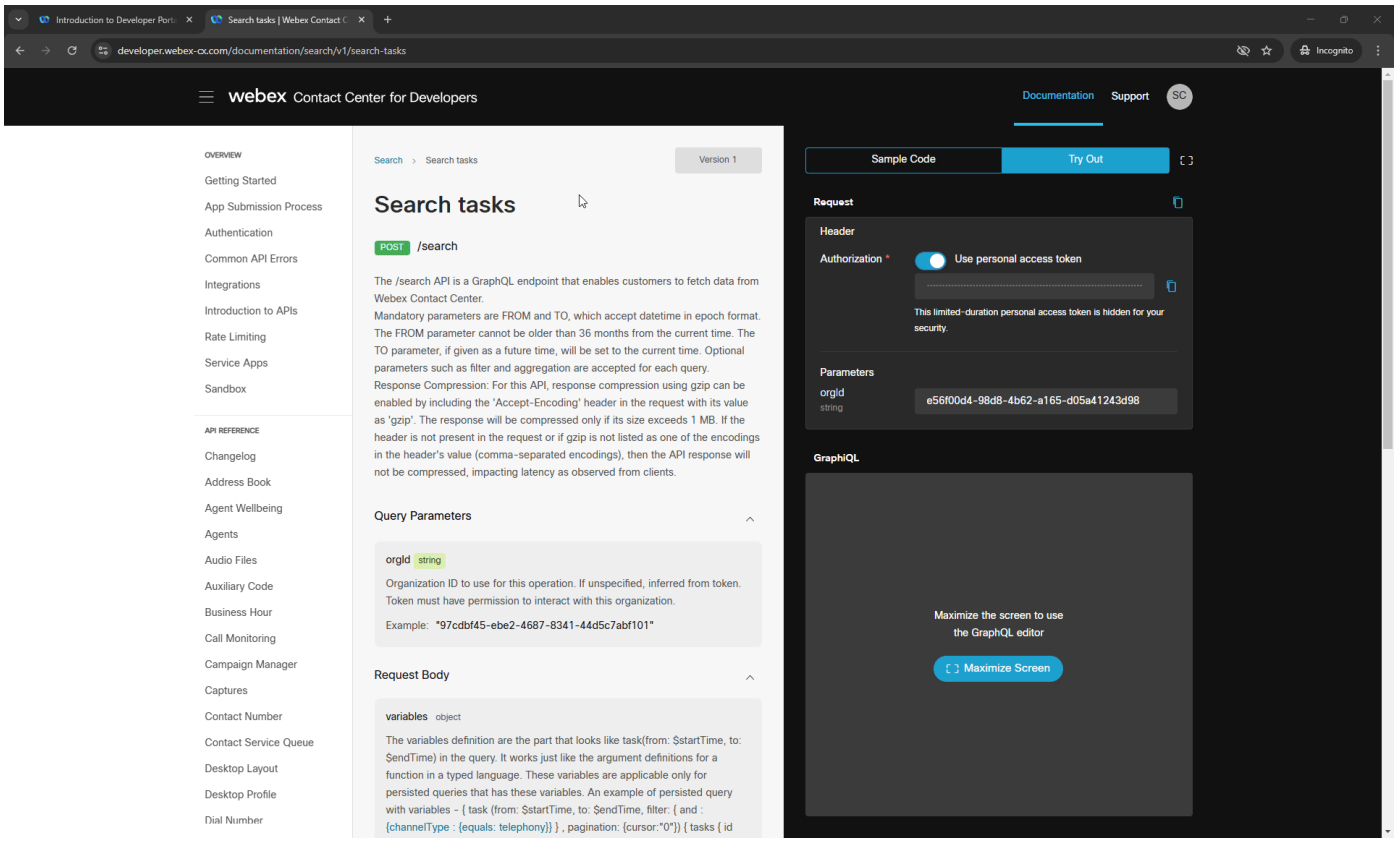
c. Total duration of the call

d. Origin of the call. Who called.

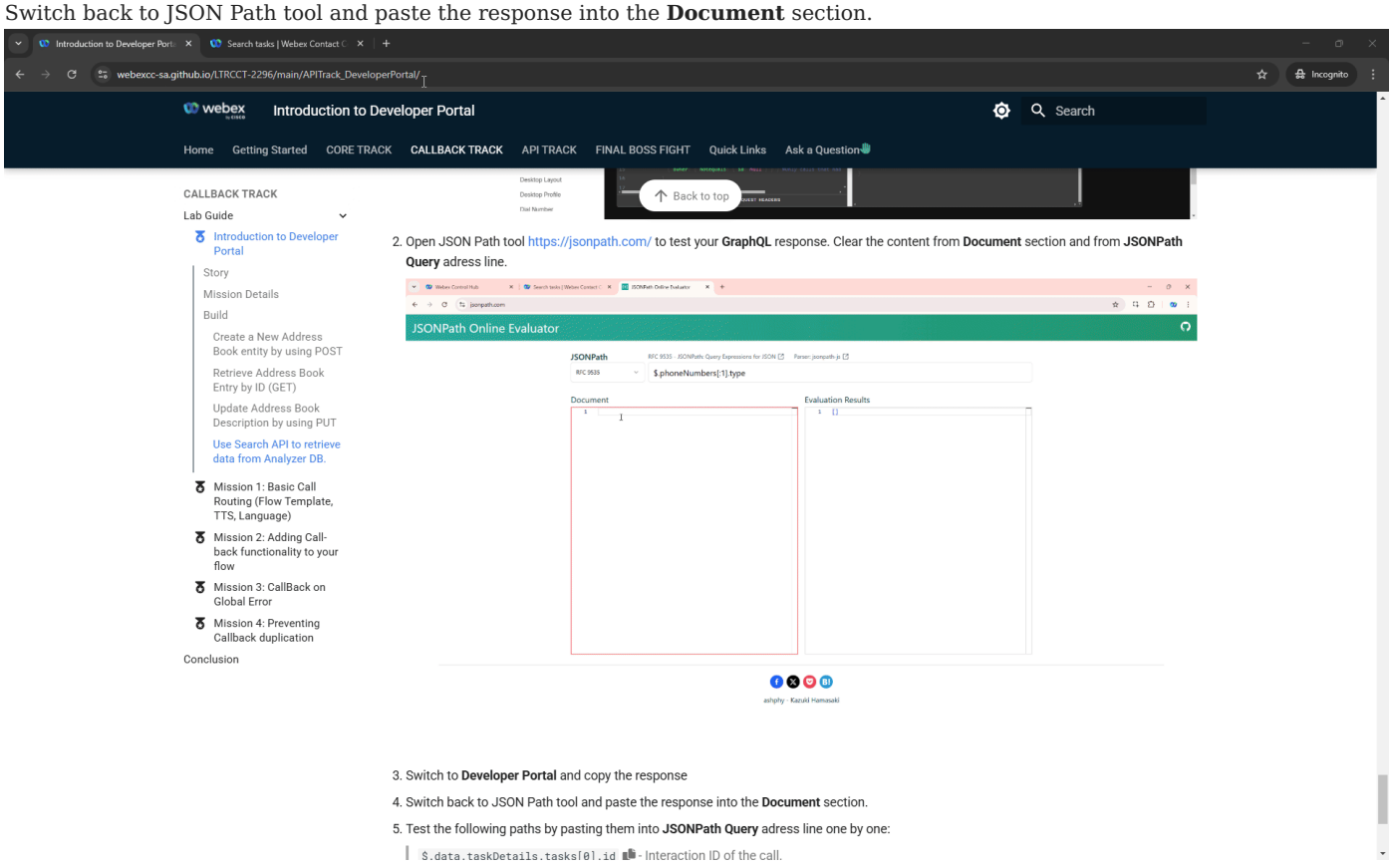
e. Destination of the call. Entry Point number.

f. Agent, whoc accepted the call: ID and Name

g. Language selected by the caller. Represented as **Global_Language** variable




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



7. 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.

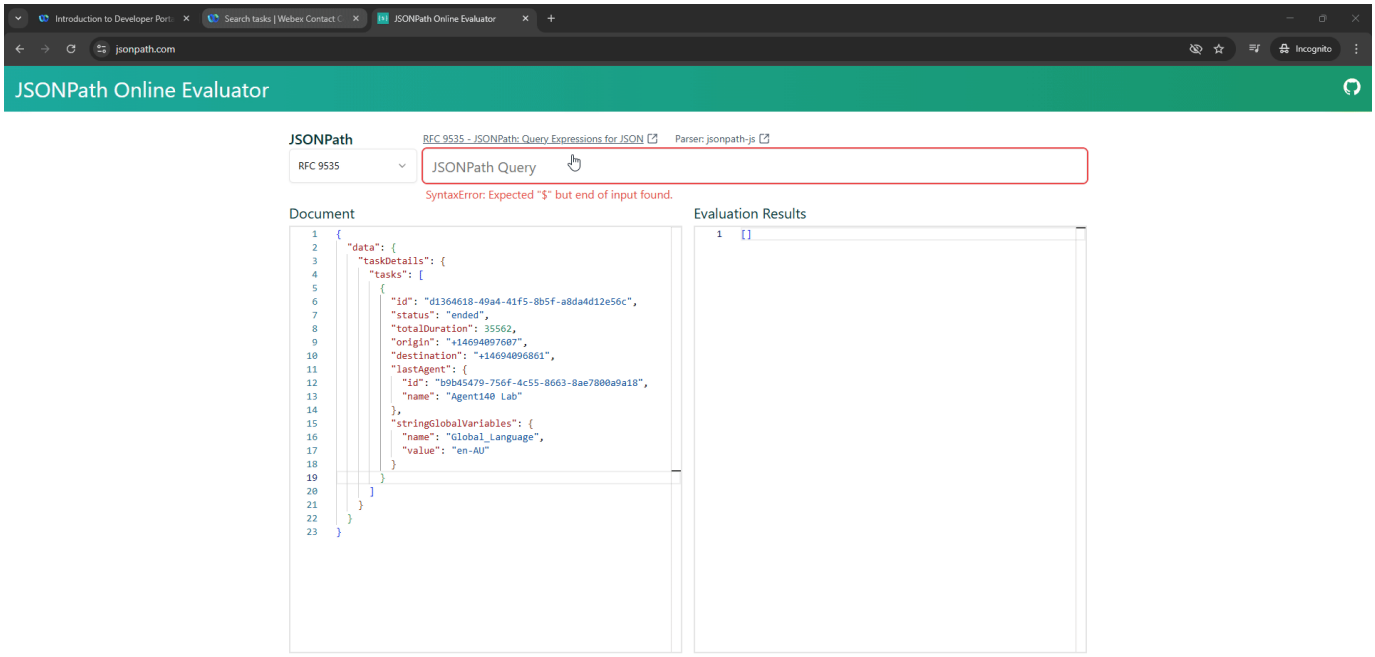
   `$.data.taskDetails.tasks[0].destination` - Call destination. This is the number essigned to Entry Point.

   `$.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.Global_Language` - Language Global Variable that was used in the flow.

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



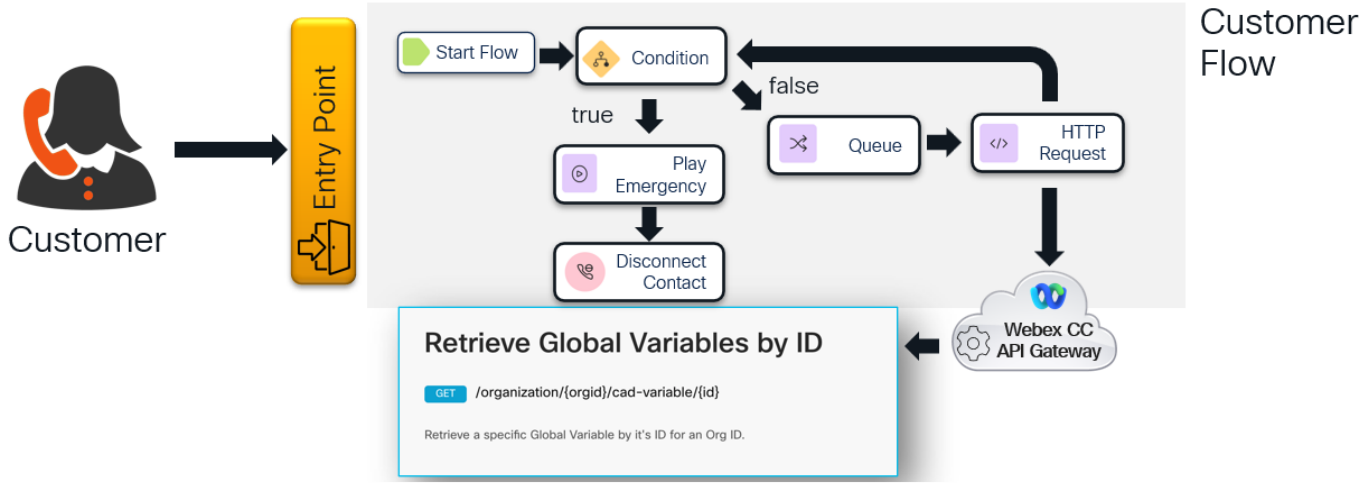

ashphy - Kazuki Hamasaki

**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 GlobalVariable 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**




### Call Flow Overview

1. Supervisor calls to management flow and provide it's 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

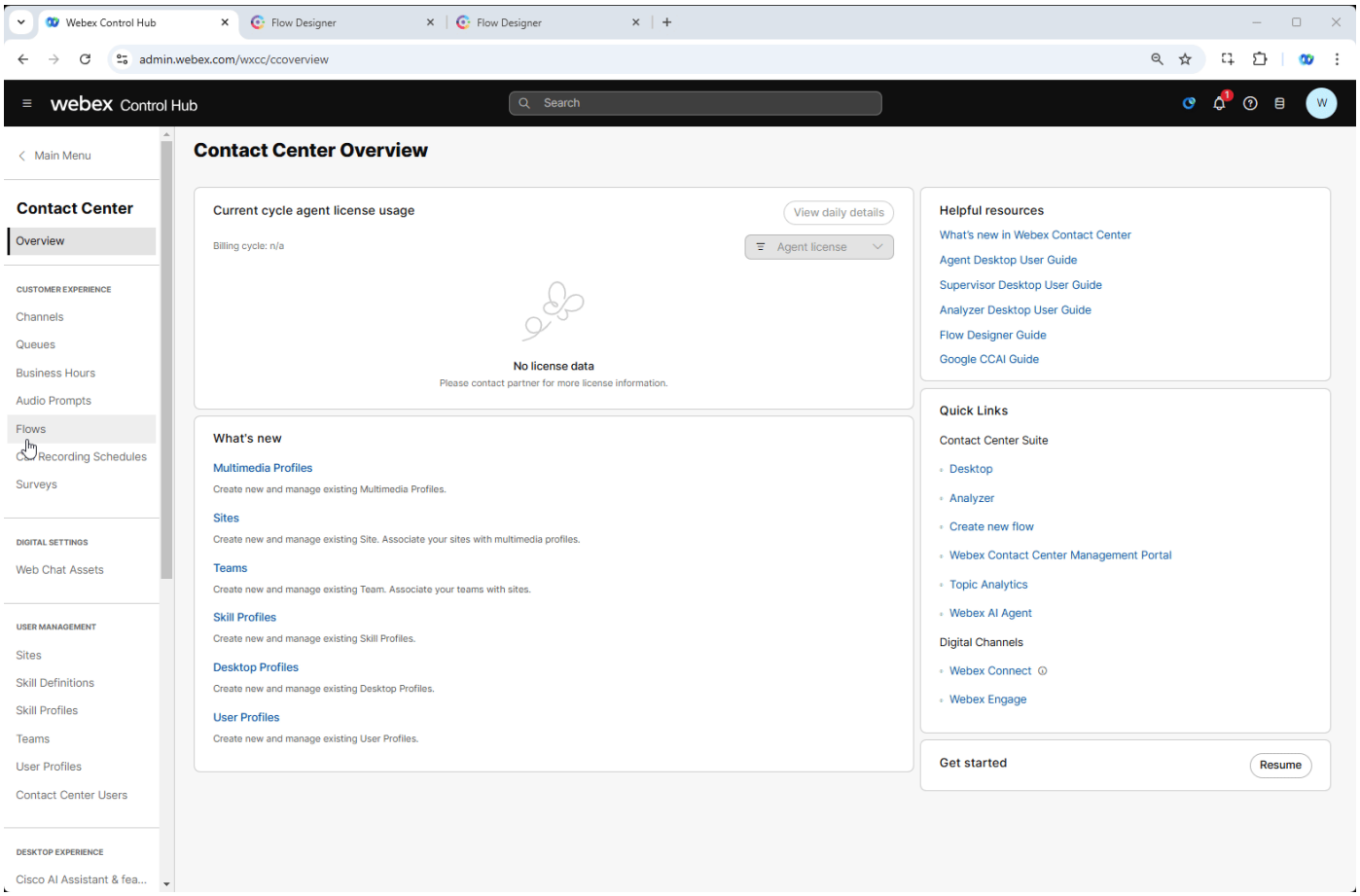
1. In Control Hub Flows page open **Global Variables** tab and create new Global Variable:

   Name: **EmergencyGV_Your_Attendee_ID**

   Type: **Boolean**

   Default Value: **False**

   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**.
4. Add a **Collect Digits** node:

   Rename node to **CollectPIN**

   Connect the **New Phone Contact** output node edge to this **Collect Digits** node

   Loop **No-Input Timeout** and **Unmatched Entry** to itself

   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: ***Please enter 4 digits pin code to activate emergency flow.***

   Set checkbox in **Make Prompt Interruptible**

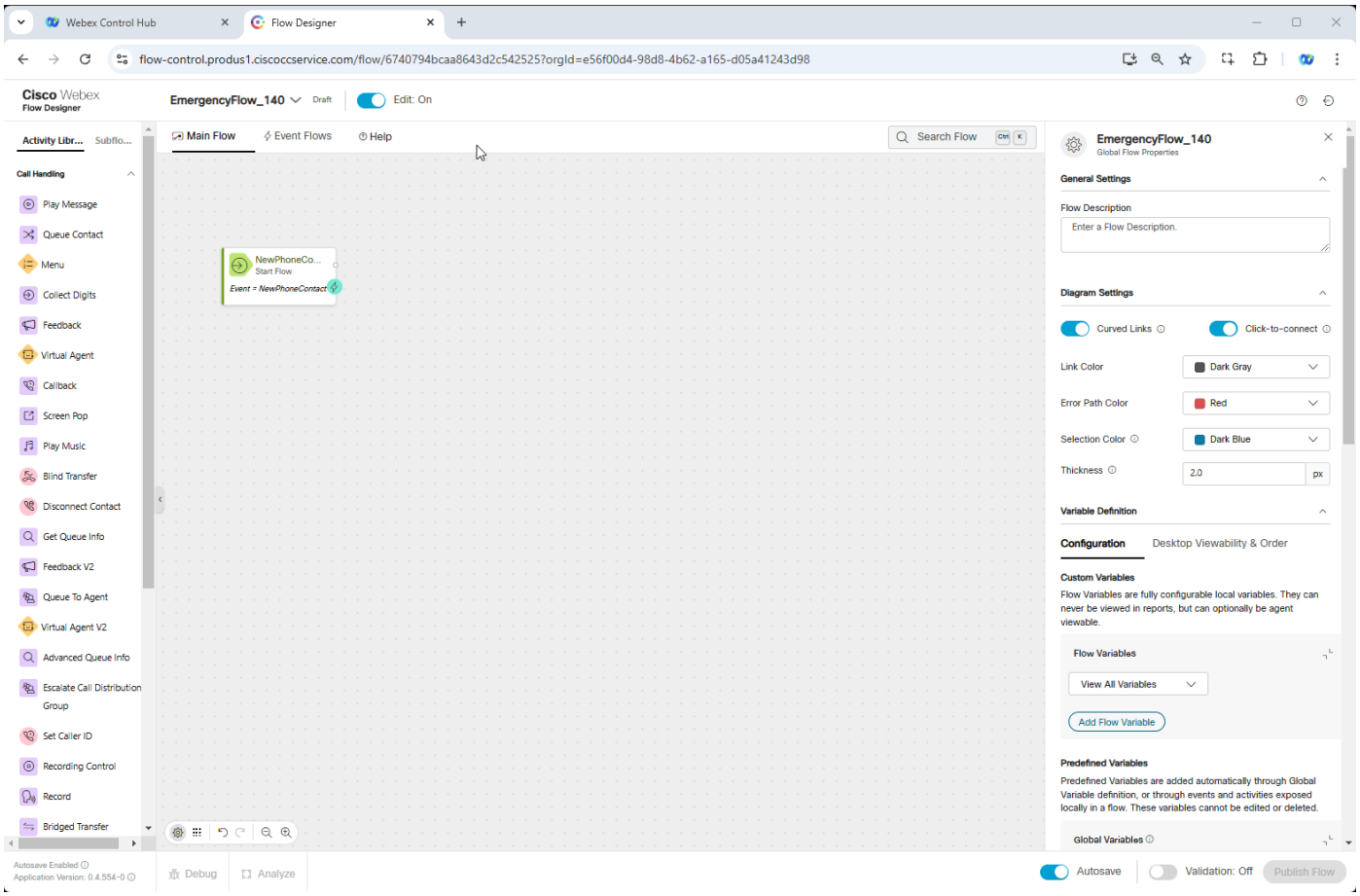
   Advanced Settings:

   No-Input Timeout: 3

   Inter-Digit Timeout: 3

   Minimum Digits: 1

   Maximum Digits: 10



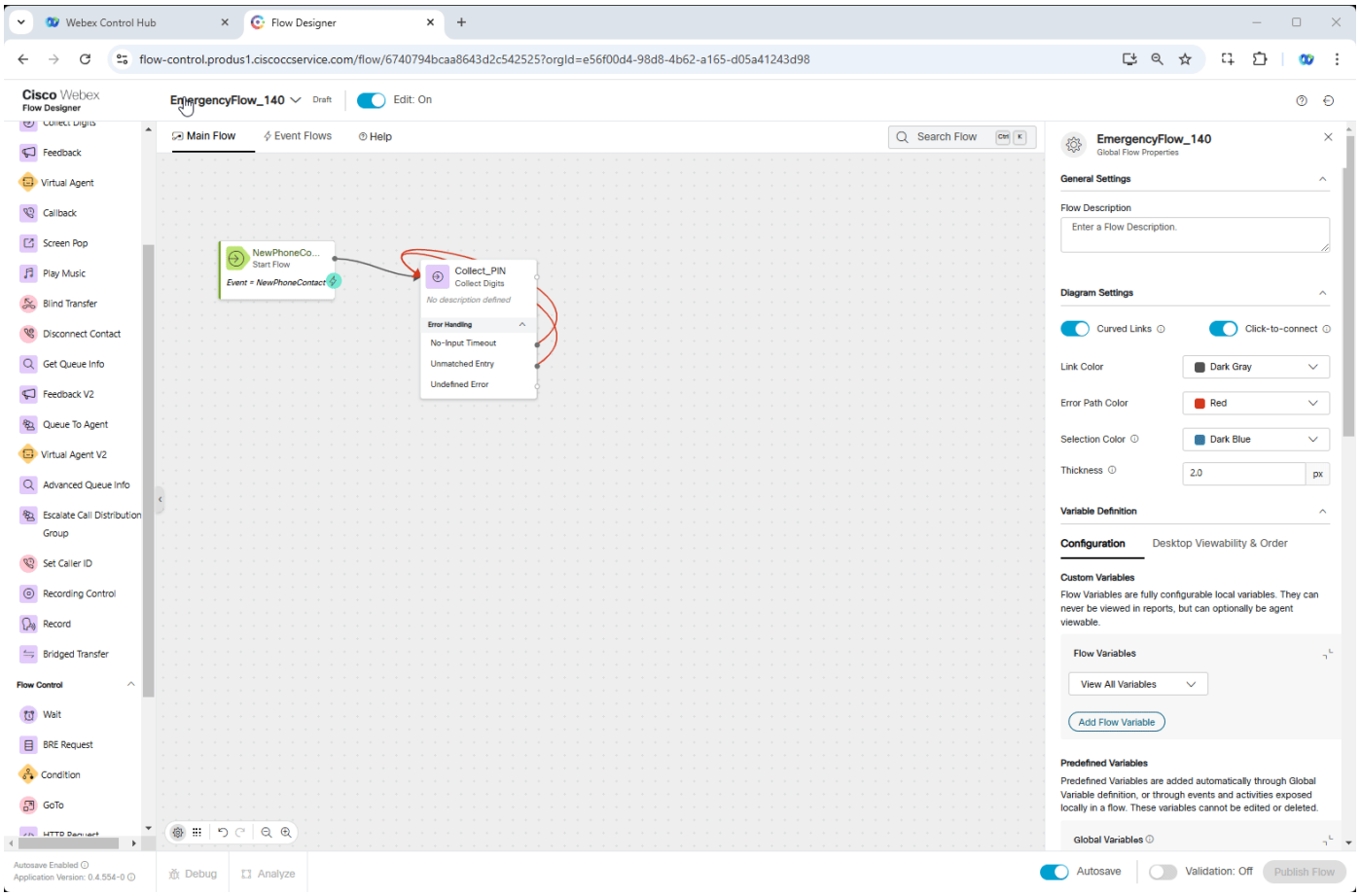

5. Add **Condition** node

   Activity Label: **PIN_Check**

   Connect the output node edge from the **Collect Digits** node to this node

   In the Expression section write an expresion ***{{CollectPIN.DigitsEntered == '1111'}}***

   [Optional] You can verify the expresion result by clicking on **Test Expression** icon in the Expresion 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.

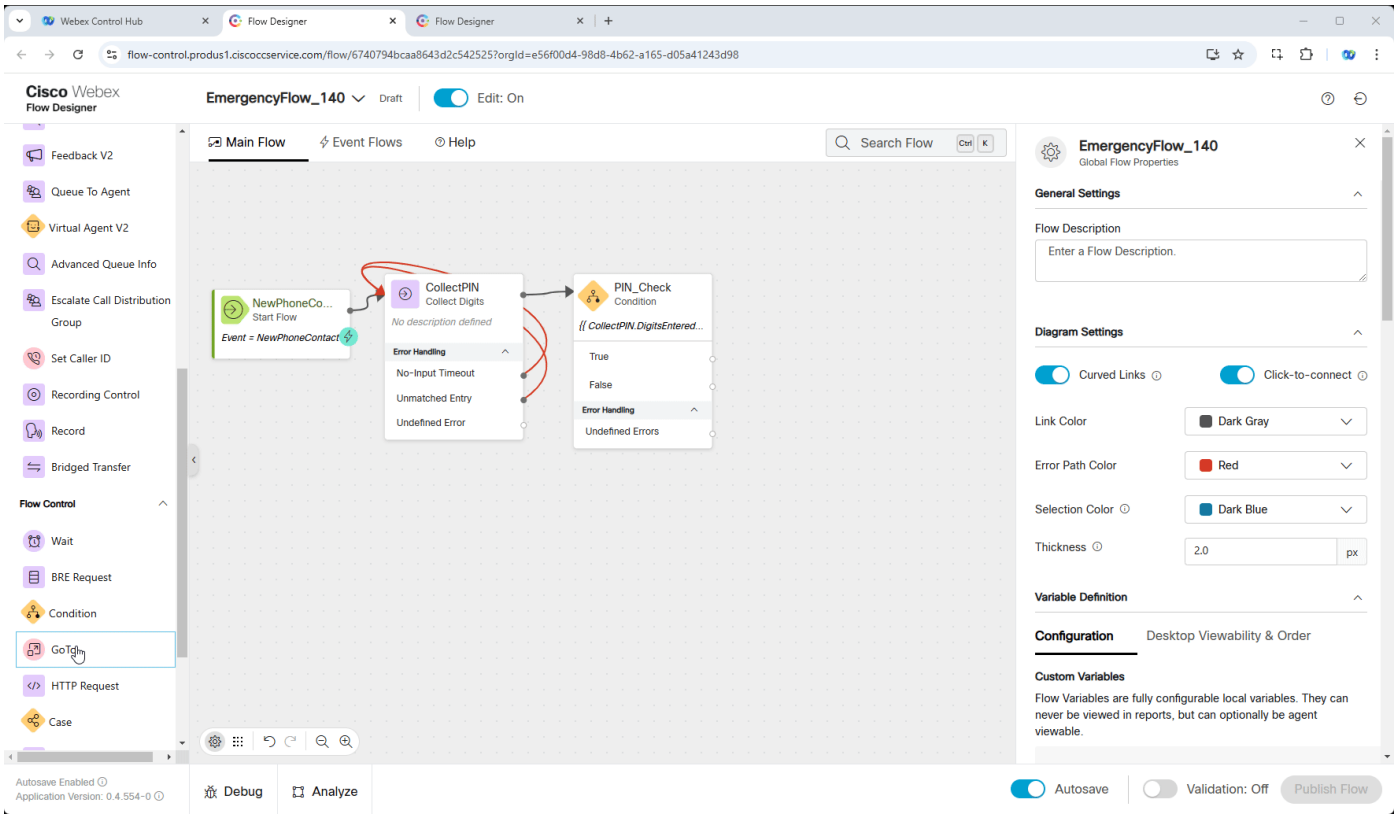
   Method: **PUT**

   Content Type: **Application/JSON**

   Request Body:

```
{
    "active": true,
    "agentEditable": false,
    "agentViewable": false,
    "variableType": "Boolean",
    "defaultValue": "true",
    "desktopLabel": "",
    "id": "yourGlobalVariableID created in step 1",
    "name": "yourGlobalVariable 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***

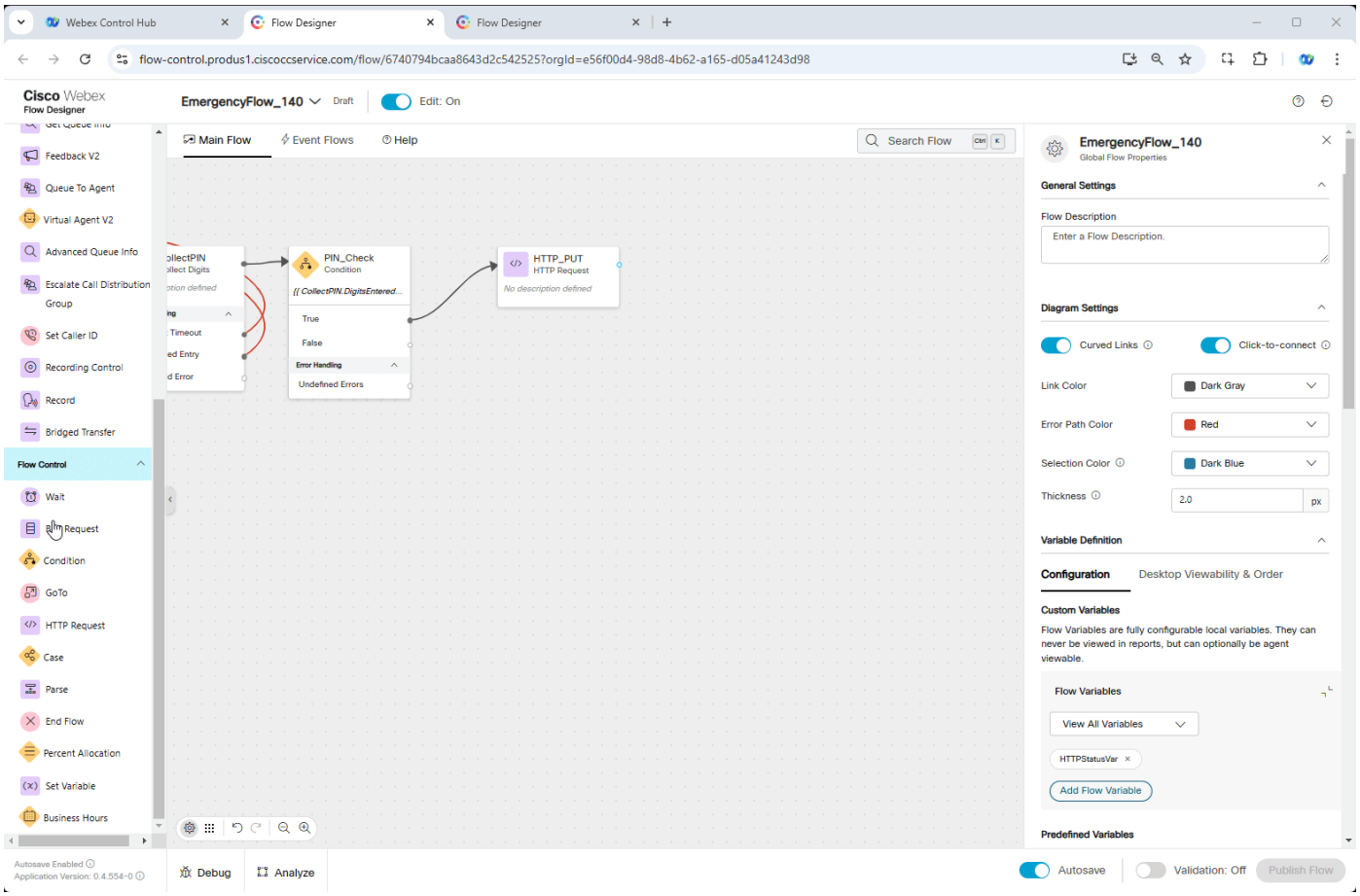



7. Add one more **Condition Node**. In this node we are going to check the status of our API PUT request. If it is **200 OK** the output will be **True** and if other than **200** then **False**.

   Activity Label: **HTTPStatusCode**

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

   In the Expression section write an expresion ***{{HTTP_PUT.httpStatusCode == 200}}***




8. Add a **Play Message** node

   Connect the **HTTPStatusCode** TRUE output node edge to this **Play Message** node

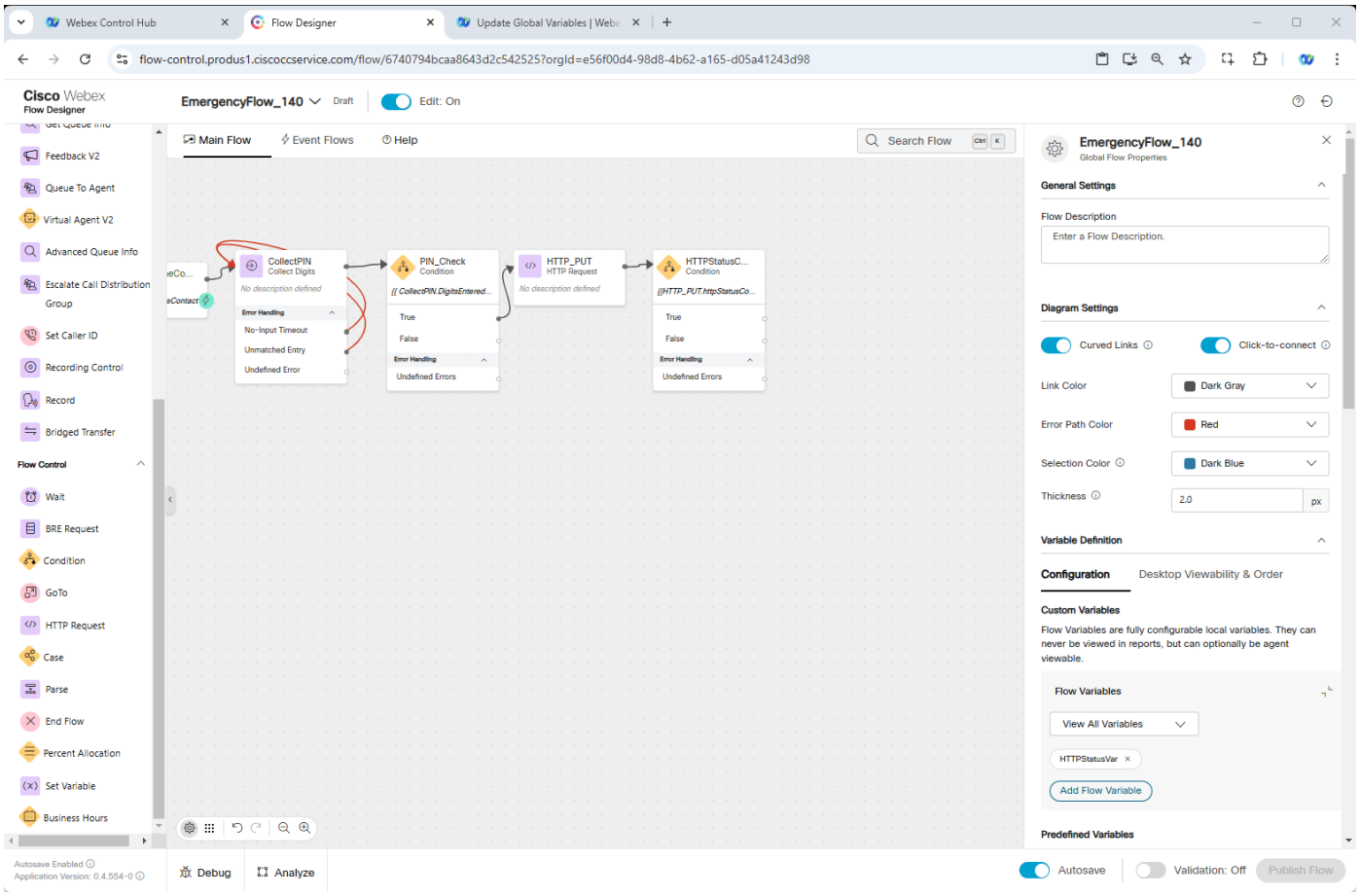
   Enable Text-To-Speech

   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: ***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

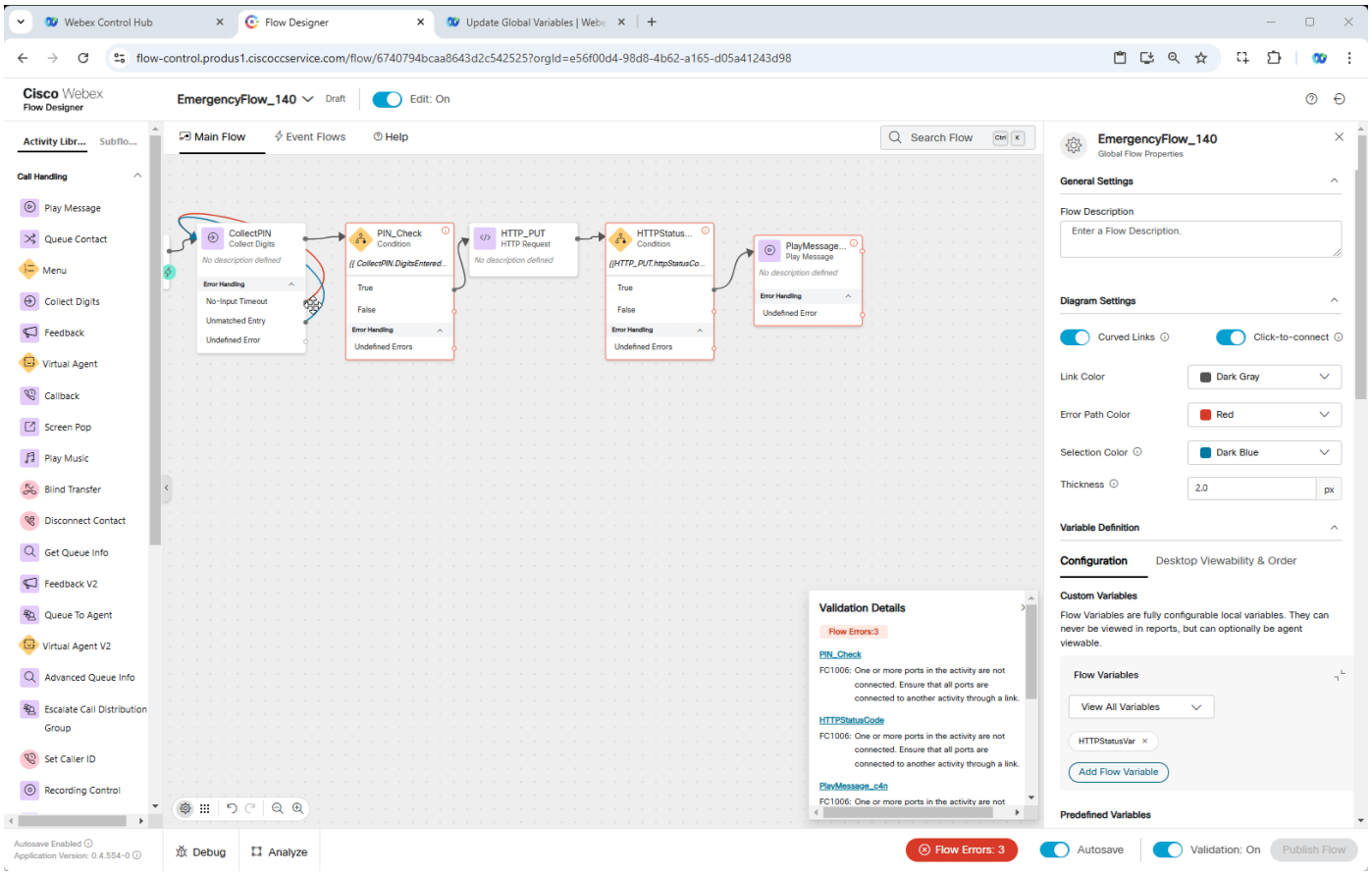
   Enable Text-To-Speech

   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**

    Connect both **Play Message** nodes created in **Steps 8** and **9** to this node

11. Publish your flow

    Turn on Validation at the bottom right corner of the flow builder

    If there are no Flow Errors, Click **Publish**

    Add a publish note

    Add Version Label(s): **Latest**

    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

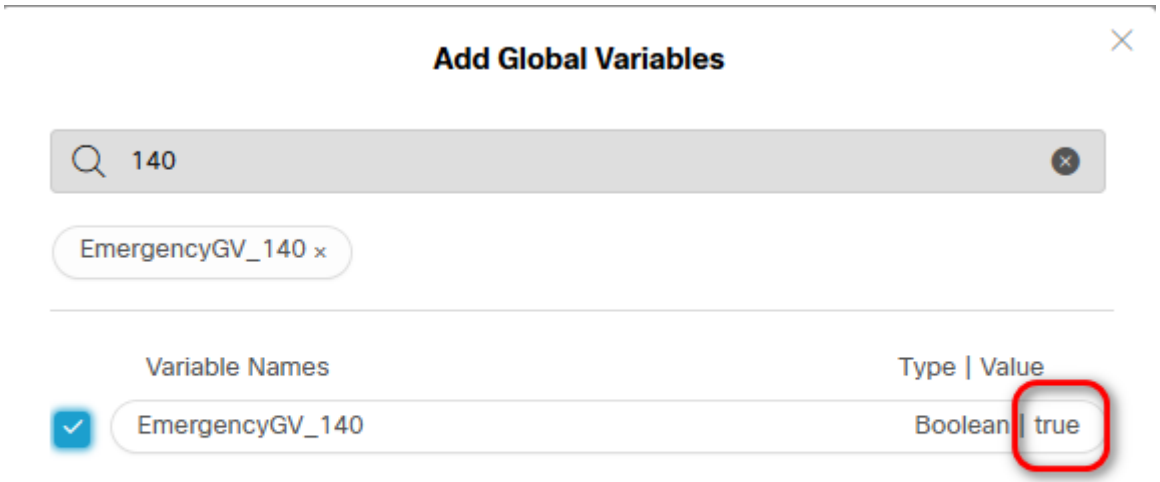
### Testing

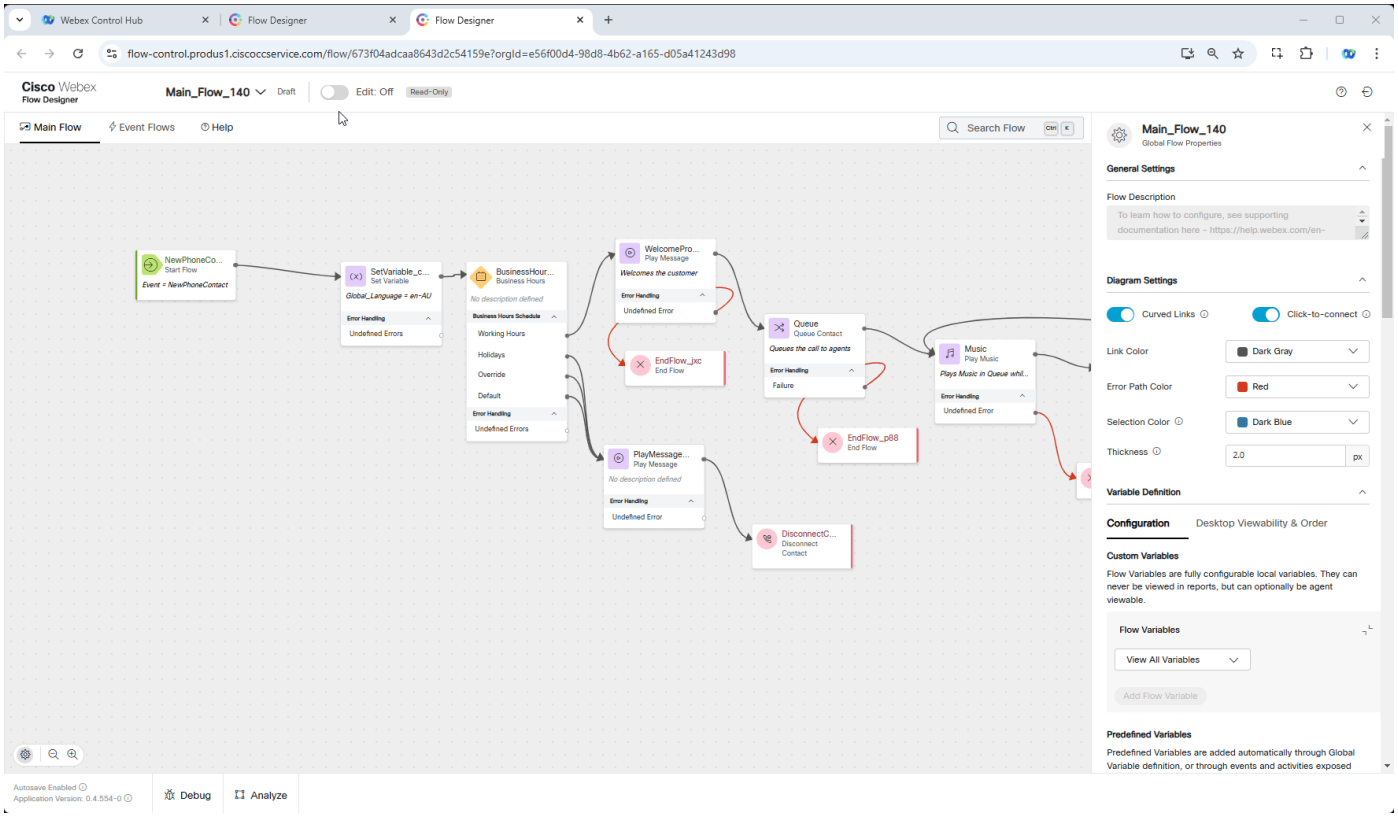
1. Open your Global Variable **EmergencyGV_Your_Attendee_ID** and make sure Default Value is set to **False**
2. Make a call to your Support Number, when asked provide a pin code 1111# and listen the next message:

   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 **False** in General Settings of the flow as shown on the following picture.




6. Add **Condition** node:

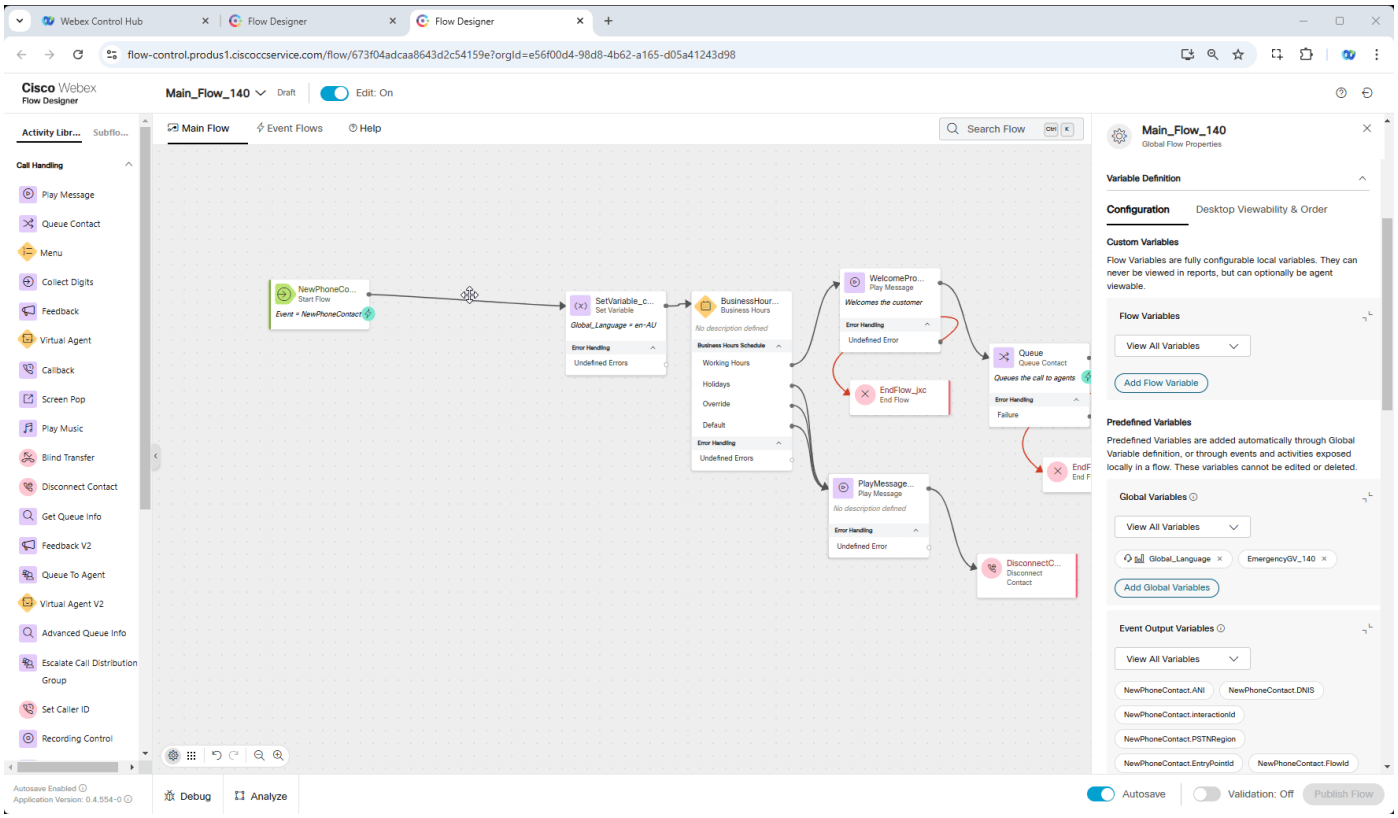
   Connect the output node edge of the **NewPhoneContact** node to this node

   Connect the output False node edge from the **Condition** Node to **Set Variable**

   In the Expression section write an expresion ***{{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 **DisconnectContact node**.

   Connect the **TRUE** output node edge of the **Condition Node** node to this node

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

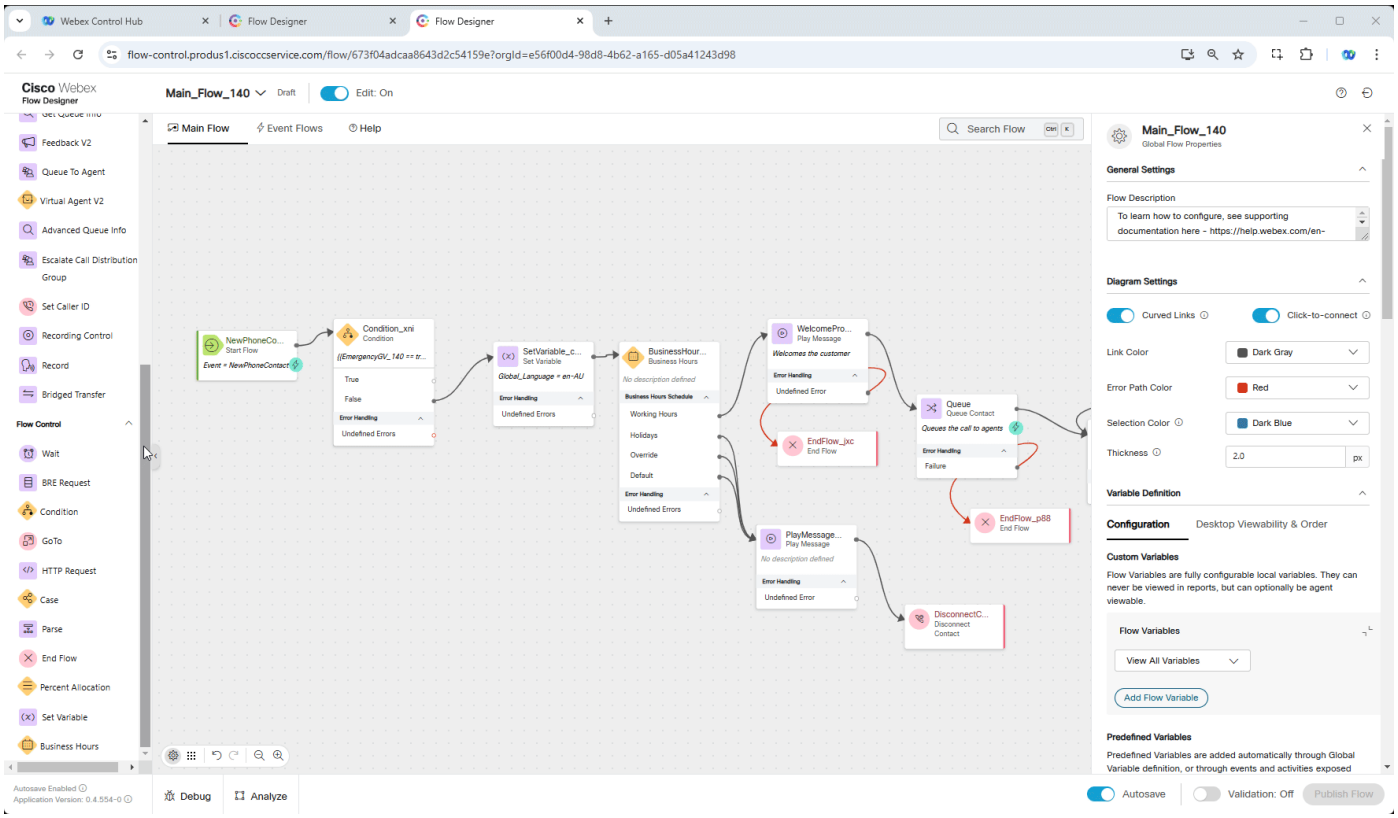
   Enable Text-To-Speech

   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: ***Sorry, Emergency flow has been enabled. All operators have been evacuated. Please call later.***




8. Publish your flow

   Turn on Validation at the bottom right corner of the flow builder

   If there are no Flow Errors, Click **Publish**

   Add a publish note

   Add Version Label(s): **Latest**

   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

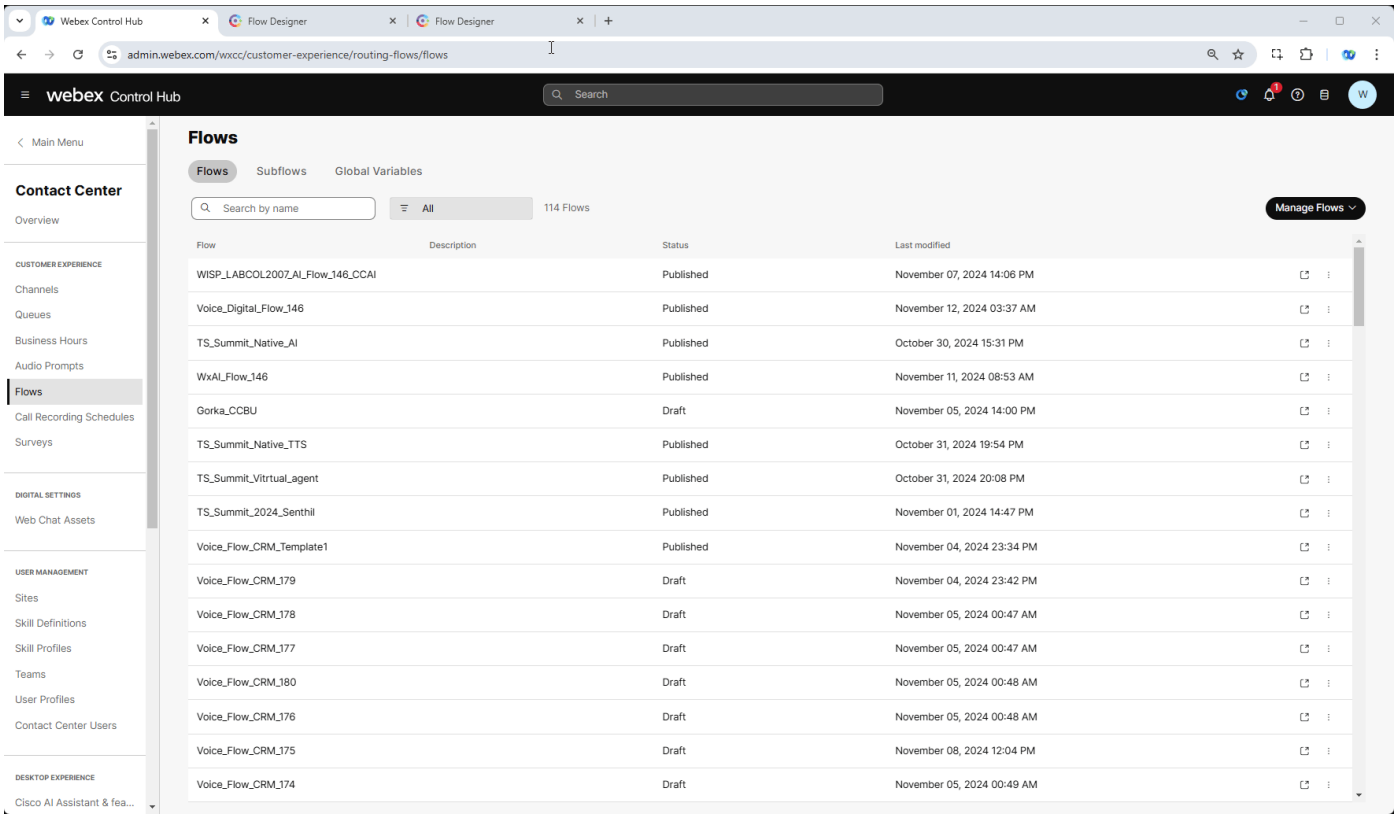
10. Make a call and you should hear the message we configured on **Step 7**.

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 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 WorkingHours 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 pre-configured for you.

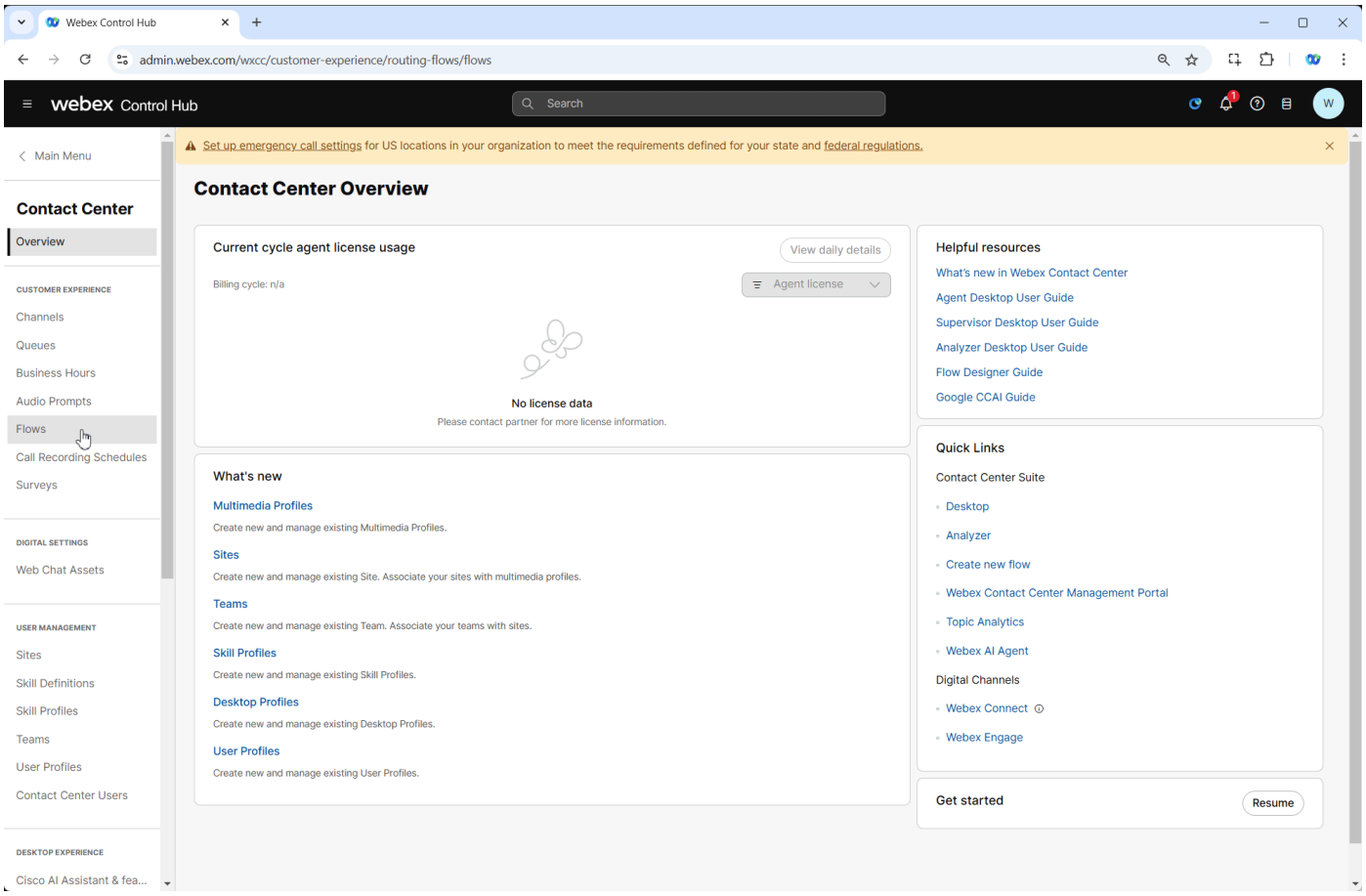
**od to Know [Optional]**

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

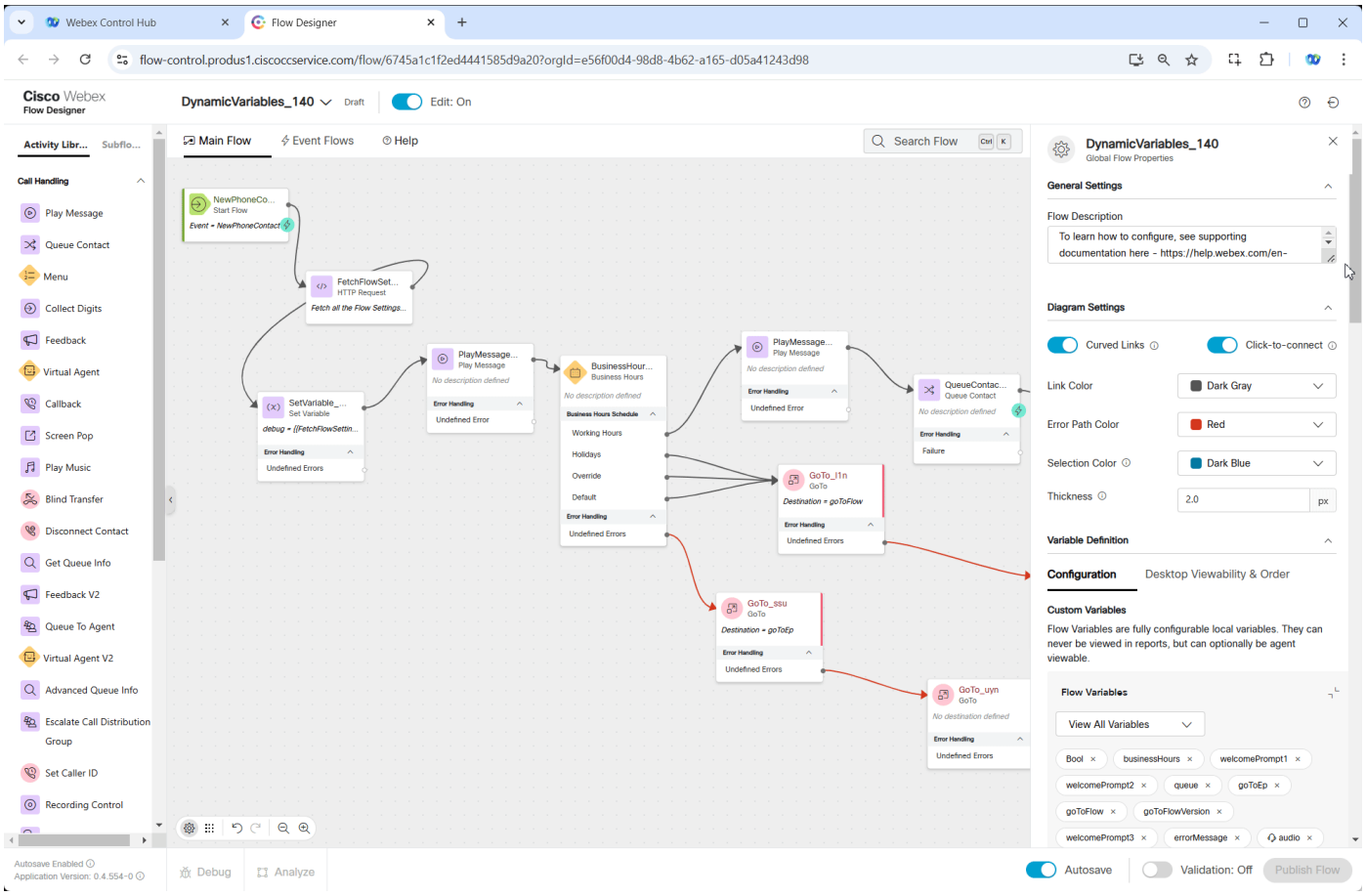
### Steps

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 statice values. We will adjust them while going through further steps.



6. Select **FetchFlowSettings** HTTP Node and paste your GET request in Request URL field by replacing a templated one. ***https://674481b1b4e2e04abea27c6e.mockapi.io/flowdesigner/Lab/DynVars?dn={{NewPhoneContact.DNIS | slice(2) }}***

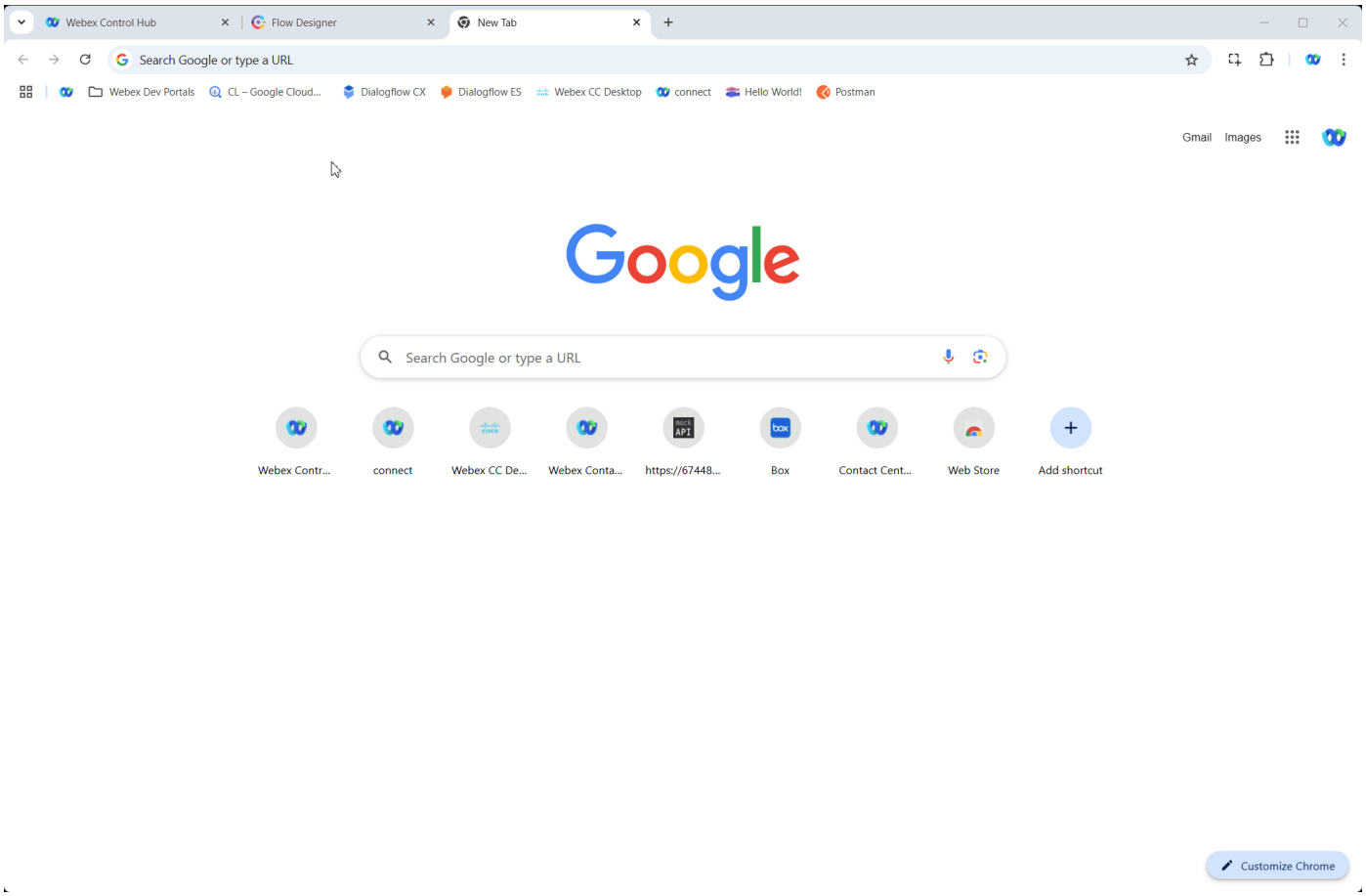
Test your API Source**[Optional]**

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

b. Replace DNIS with the provided DNIS number stripping +1

[Example:] If your number **+14694096861**, then your GET Query should be ***https://674481b1b4e2e04abea27c6e.mockapi.io/flowdesigner/Lab/DynVars?dn=4694096861***

a. Open Chrome browser and past your URL. You should get the follwoing result




a. Test JSON Path in the following tool https://jsonpath.com/

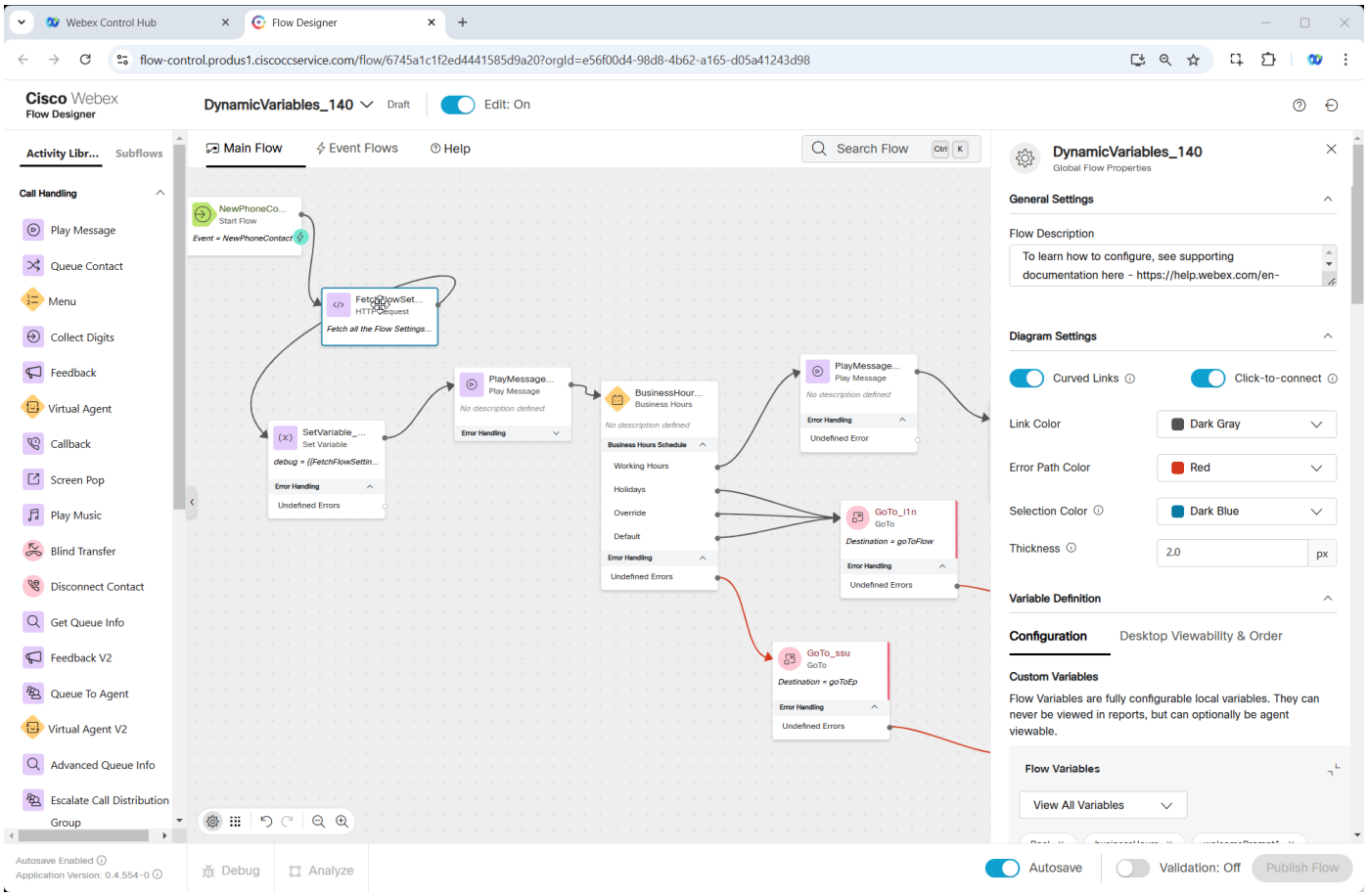
b. Paste your GET URL into the Browser address line and copy the output in square brackets (including brackets)

c. Open https://jsonpath.com/ and paste the copied response into **Inputs** window

d. In **JSONPath** box copy and paste one of the path expression from **FetchFlowSettings** to verify your results.

Profiles

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




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

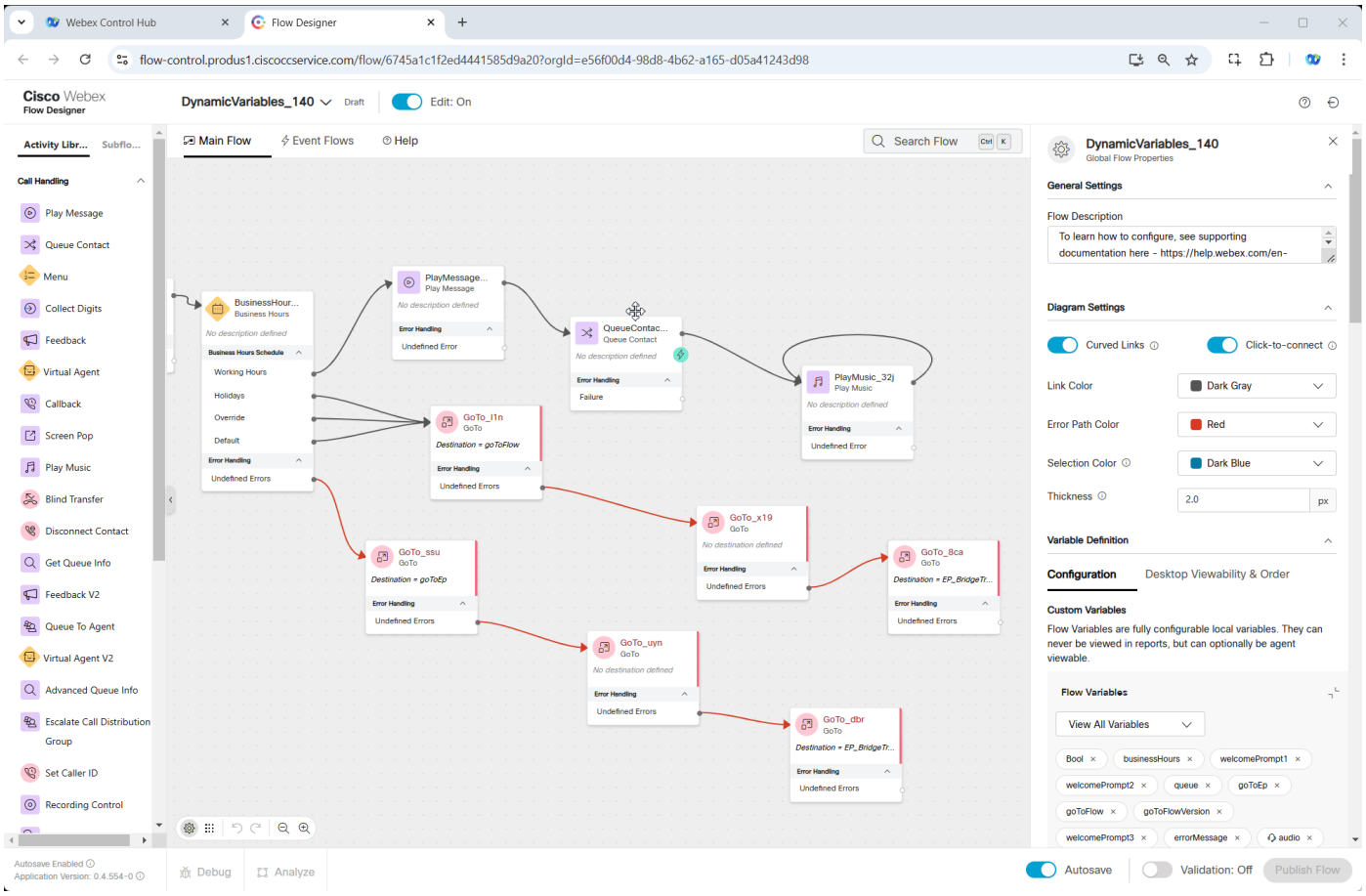
   Flow: **CLTS_ErrorHandling_Flow**

   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**



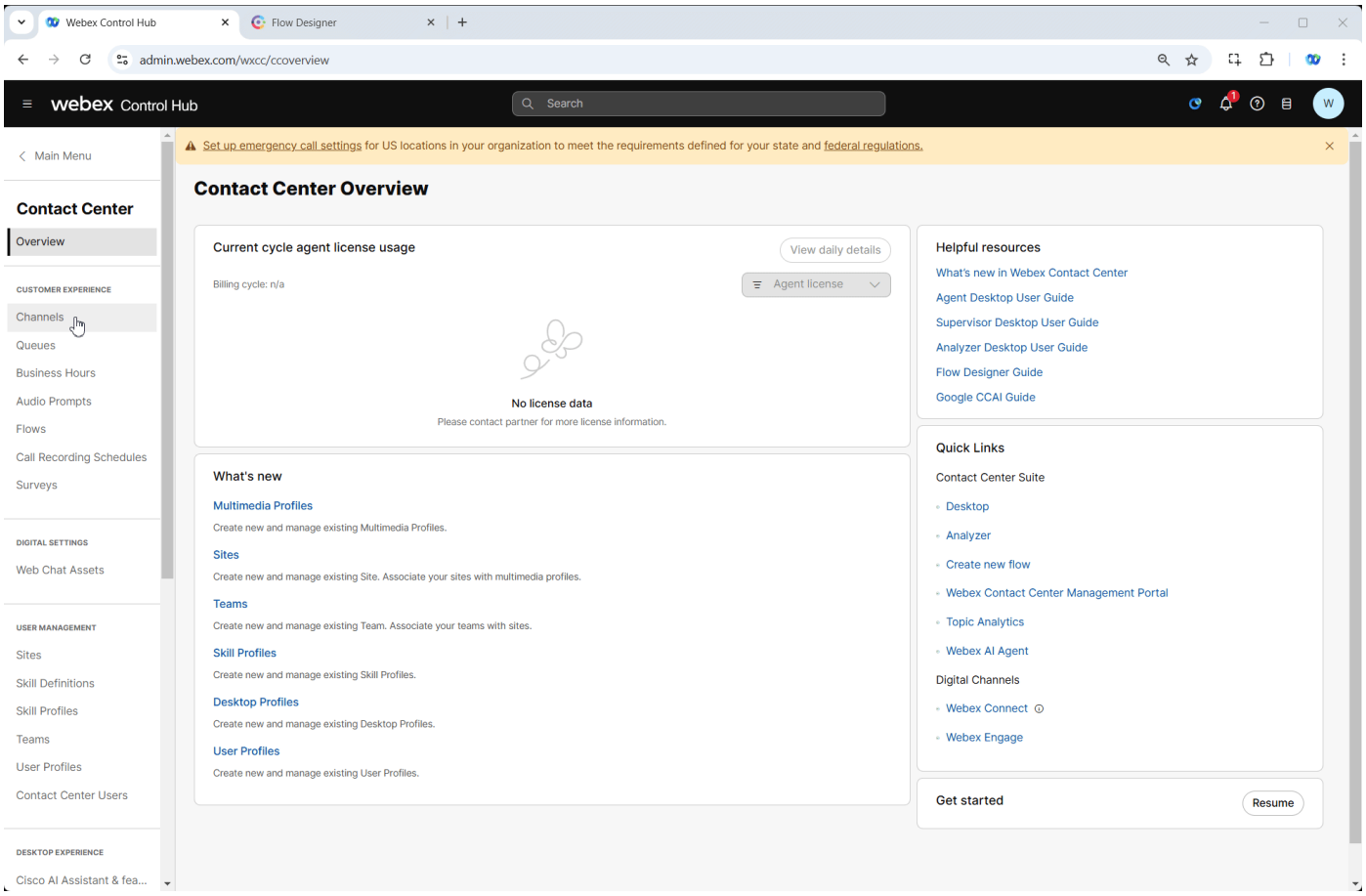

13. **Validate** and **Publish** flow
14. In Popped up window click on dropdown menu to select **Latest** label, then click **Publish**
15. Switch to Control Hub and navigate to **Channels** under Customer Experience Section

    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

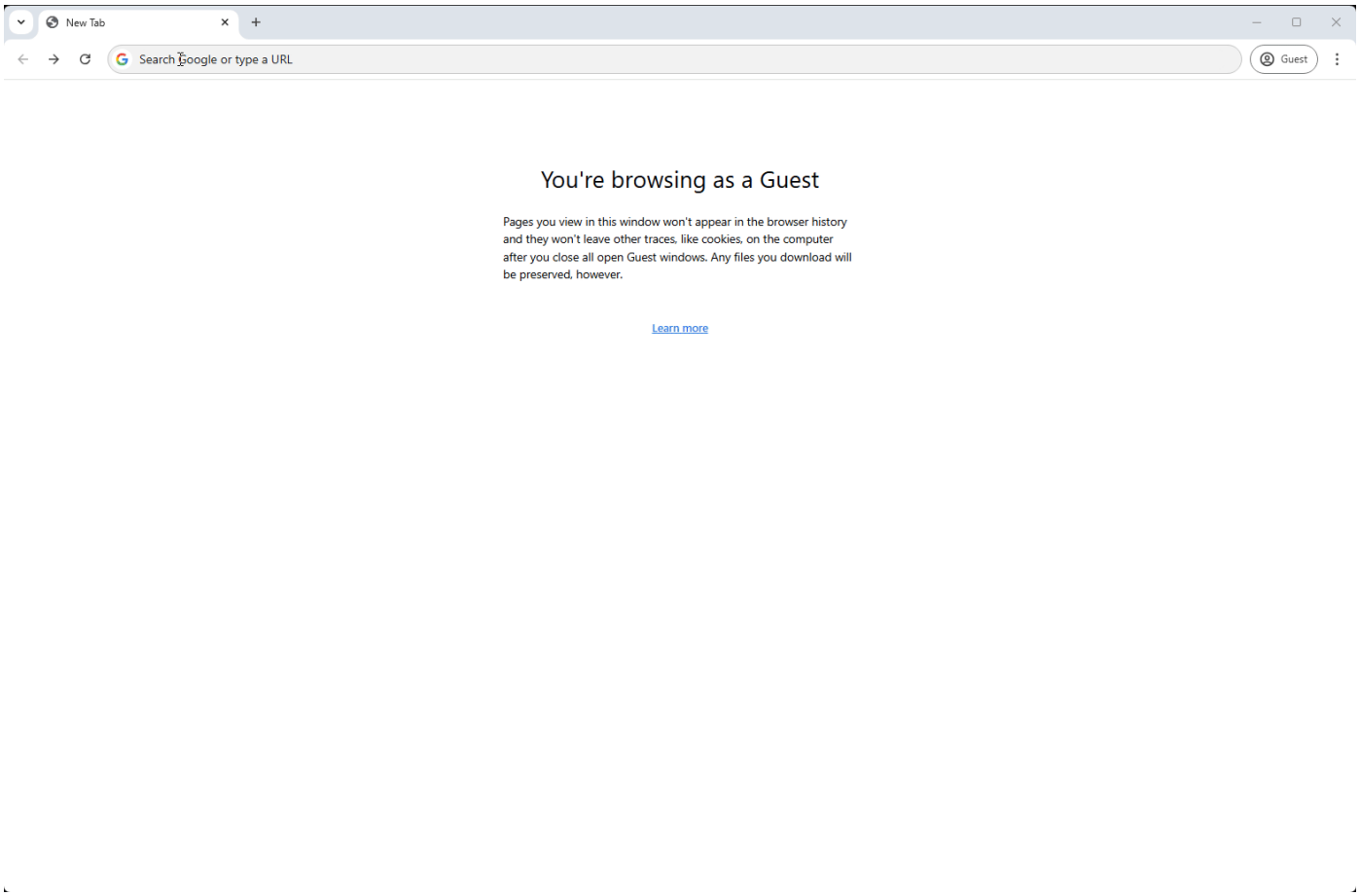



### Testing

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 OKTA 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.

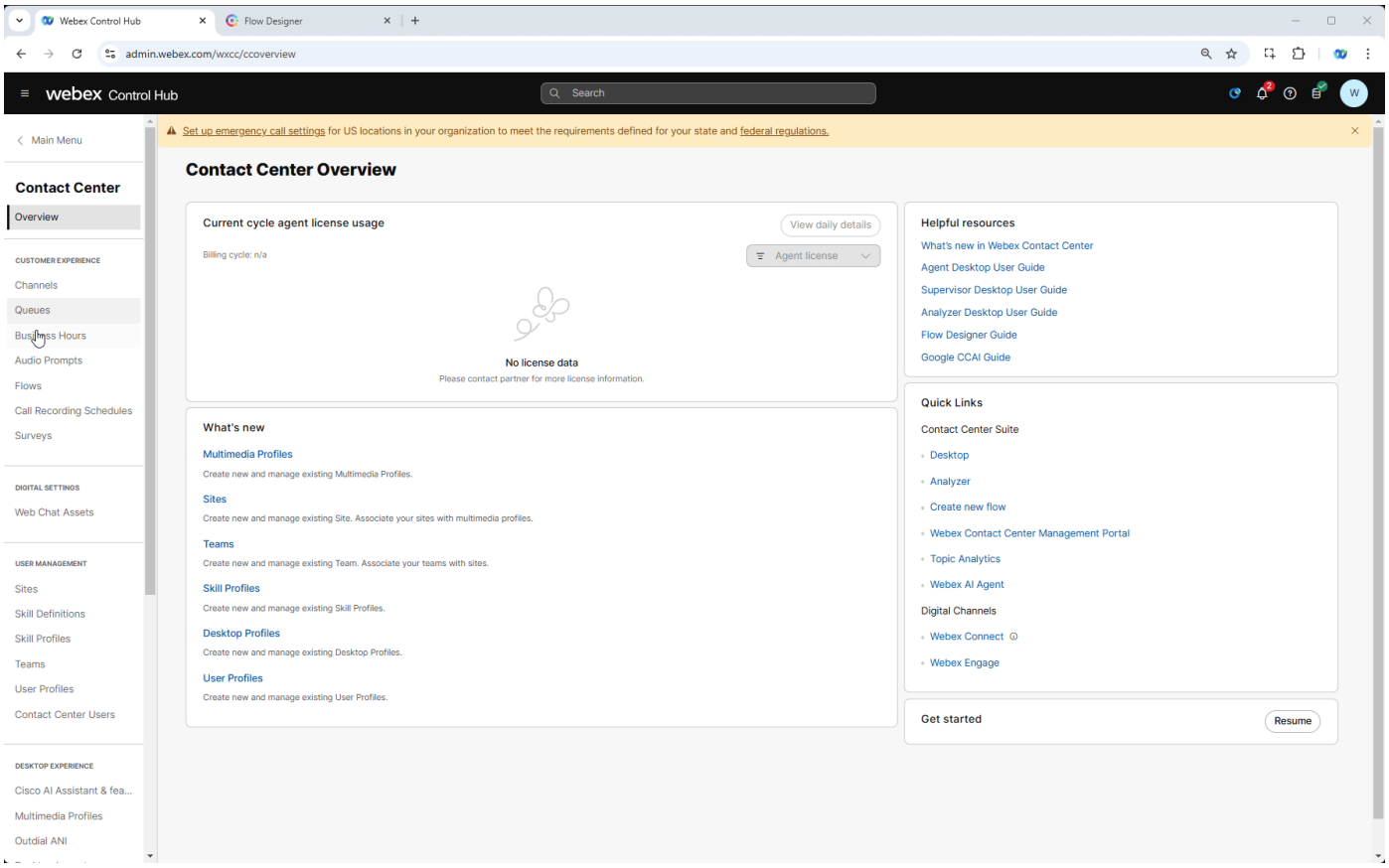



4. Change the status of your agent to **Available** and make a call to test you flow. If everyhing configured as per instructions you should hear a **welcome1** message that is a value of ***$[0].welcomePrompt1*** and then ***$[0].welcomePrompt2***. Finally the call should land on the ***$[0].queue***

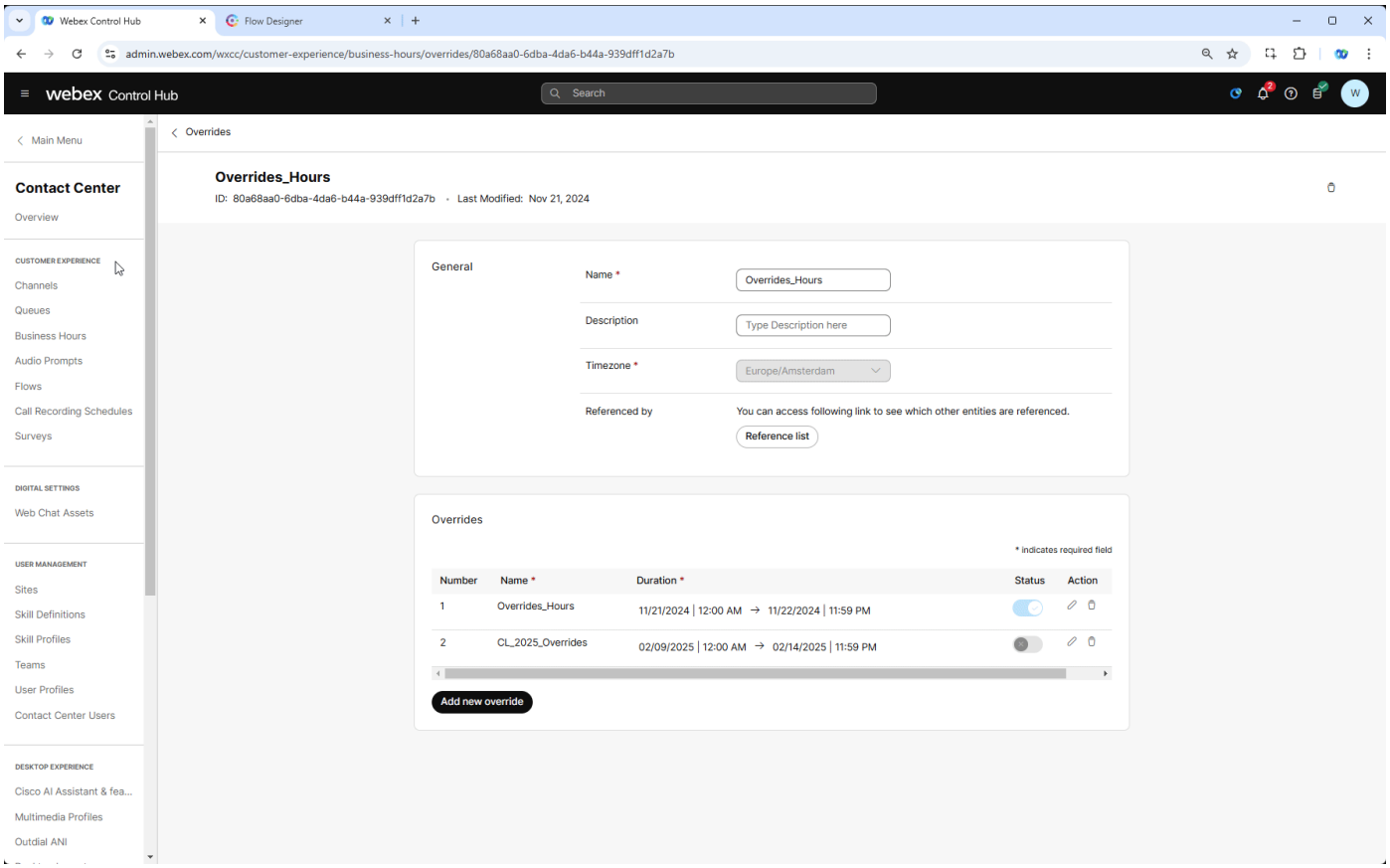
[OPTIONAL] TEST OTHER VARIABLES

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 call. You are now on Error Handling flow and will be redirected to Global Support line in a moment. Good bye."**
3. Now we need to revert the configuration we made in Step 1. Open **Your_Attendee_ID_Bussiness_Hours** in **Control Hub** 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

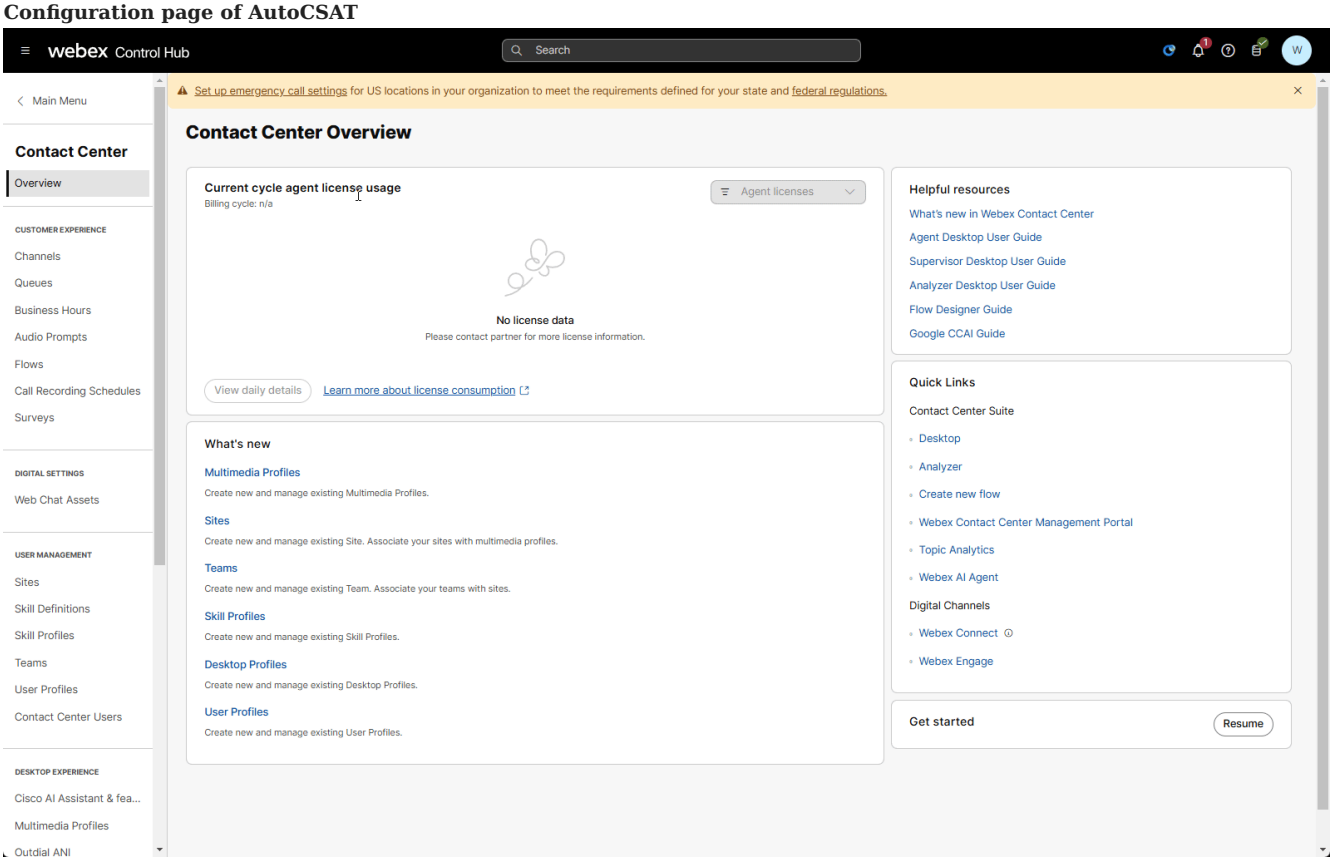
### Story

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, AutoCSAT 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 AutoCSAT teaching. With a sufficient number of provided scores, AutoCSAT will eventually be able to score calls automatically.

**Note**

**Configuration page of AutoCSAT**




### 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 AutoCSAT
3. If the AutoCSAT 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. AutoCSAT 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 flow named **LastAgentRouting_Your_Attendee_ID** and add these flow variables:

- Agent ID variable:

  Name: **agentID**

  Type: **String**

  Default Value: **empty**
- Variable to write HTTP Response into it:

  Name: **JSONResponse**

  Type: **String**

  Default Value: **empty**
- String type AutoCSAT variable:

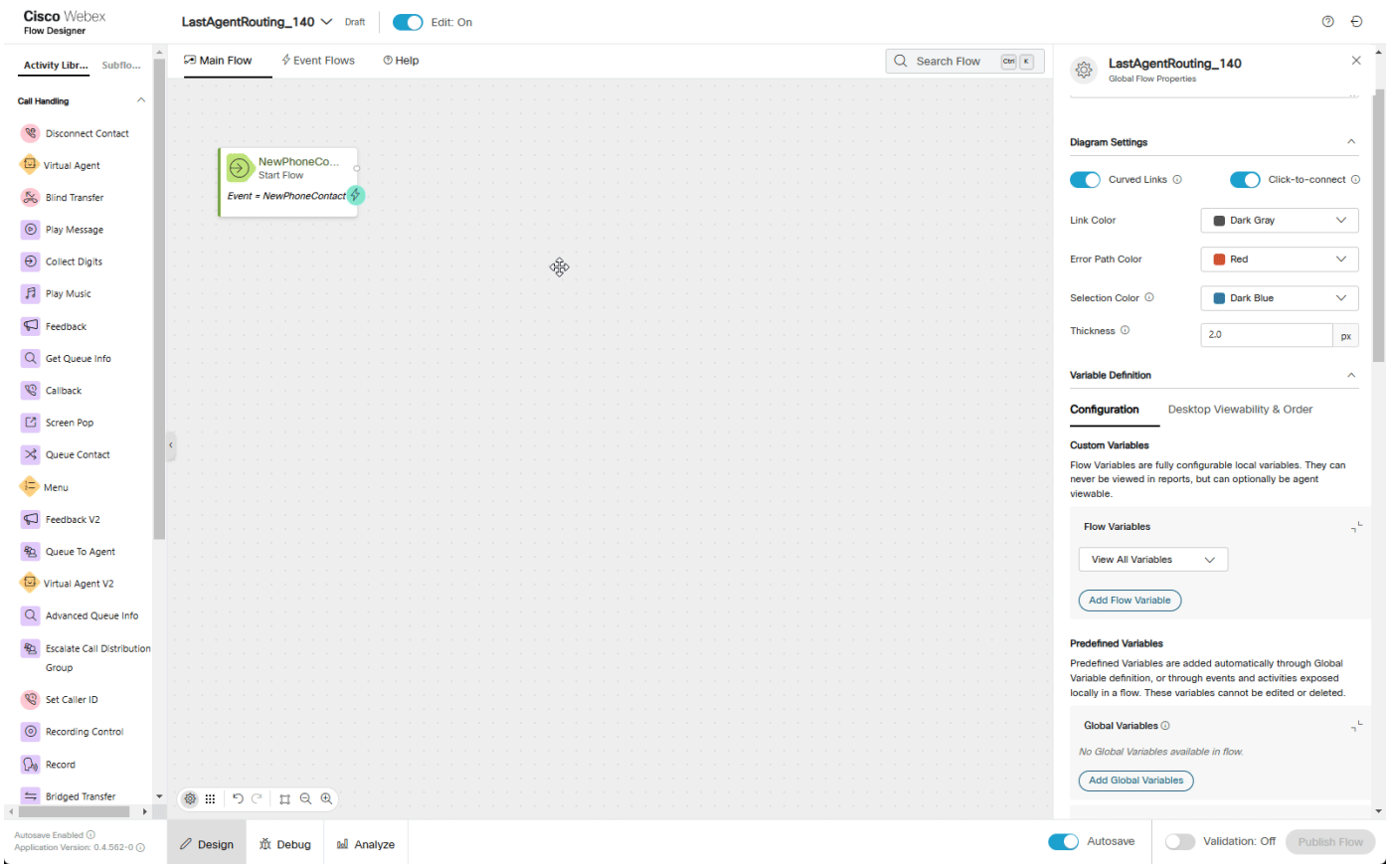
  Name: **AutoCSATVar**

  Type: **Decimal**

  Default Value: **0.0**

  Switch on **Make Agent Viewable**

  Desktop Label: **Auto CSAT**




2. Add a **Play Message** node

   Connect the **New Phone Contact** node edge to this **Play Message** node

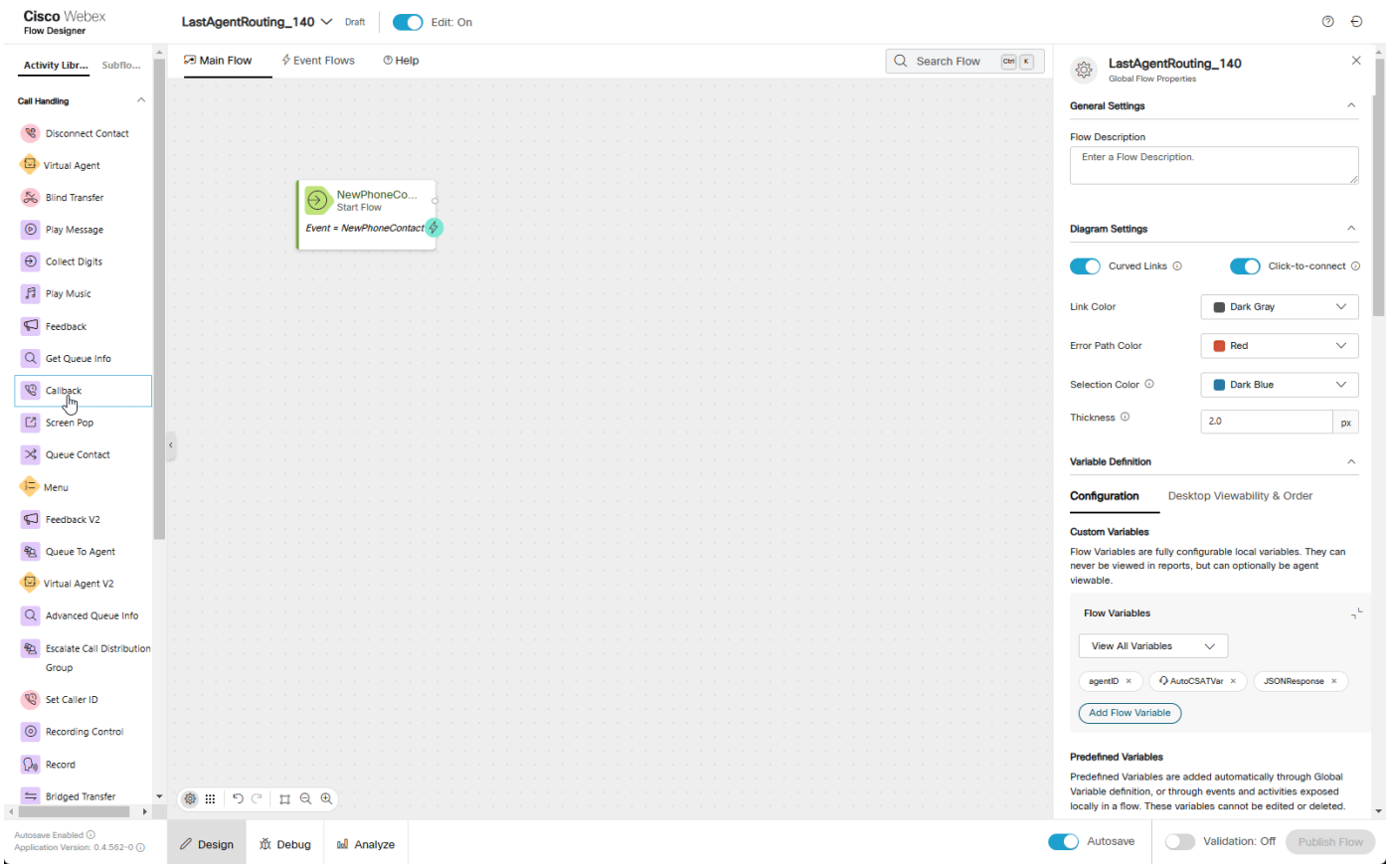
   Enable Text-To-Speech

   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: ***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**

Path: **/search**

Method: **POST**

Content Type: **Application/JSON**

Copy this GraphQL query into the request body:

```
{"query":"query($from: Long!, $to: Long!)\n{\n  taskDetails(\n      from: $from\n      to: $to\n    filter: {\n      and: [\n       { lastEntryPoint: { id: { equals: \"{{NewPhoneContact.EntryPointId}}\" } } }\n       { status: { equals: \"ended\" } }\n       { doubleGlobalVariables: {name:{equals:\"AutoCSAT_GV\"}, value: {gte:4} } }\n\n        ]\n    }\n  ) {\n    tasks {\n      csatScore  \n      autoCsat\n      owner {\n        id\n        name\n      }\n      doubleGlobalVariables(name: \"AutoCSAT_GV\"){\n        name\n        value\n      }\n\n    }\n  }\n}","variables":{"from":"{{now() | epoch(inMillis=true) - 1800000}}","to":"{{now() | epoch(inMillis=true)}}"}}
```

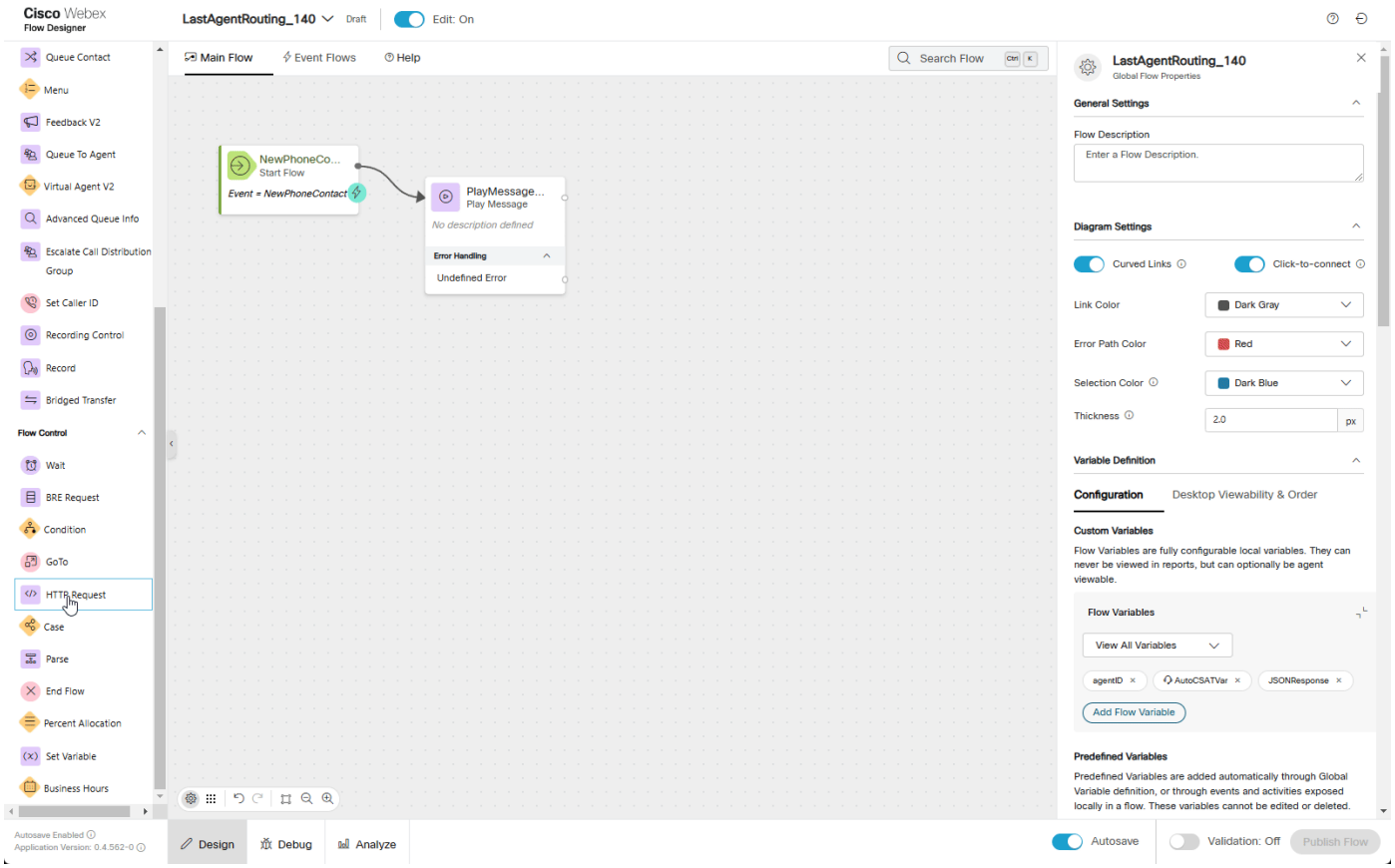
**Expanded Query For Understanding (optional)**

```
query($from: Long!, $to: Long!)
  {
    taskDetails(
        from: $from
        to: $to
      filter: {
        and: [
         { lastEntryPoint: { id: { equals: "{{NewPhoneContact.EntryPointId}}" } } }
         { status: { equals: "ended" } }
         { doubleGlobalVariables: {name:{equals:"AutoCSAT_GV"}, value: {gte:4} } }

          ]
      }
    ) {
      tasks {
        autoCsat
        owner {
          id
          name
        }
        doubleGlobalVariables(name: "AutoCSAT_GV"){
          name
          value
        }

      }
    }
  }
```

```
Expected Response:

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

Parse Settings:

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




4. Add **Set Variable** node

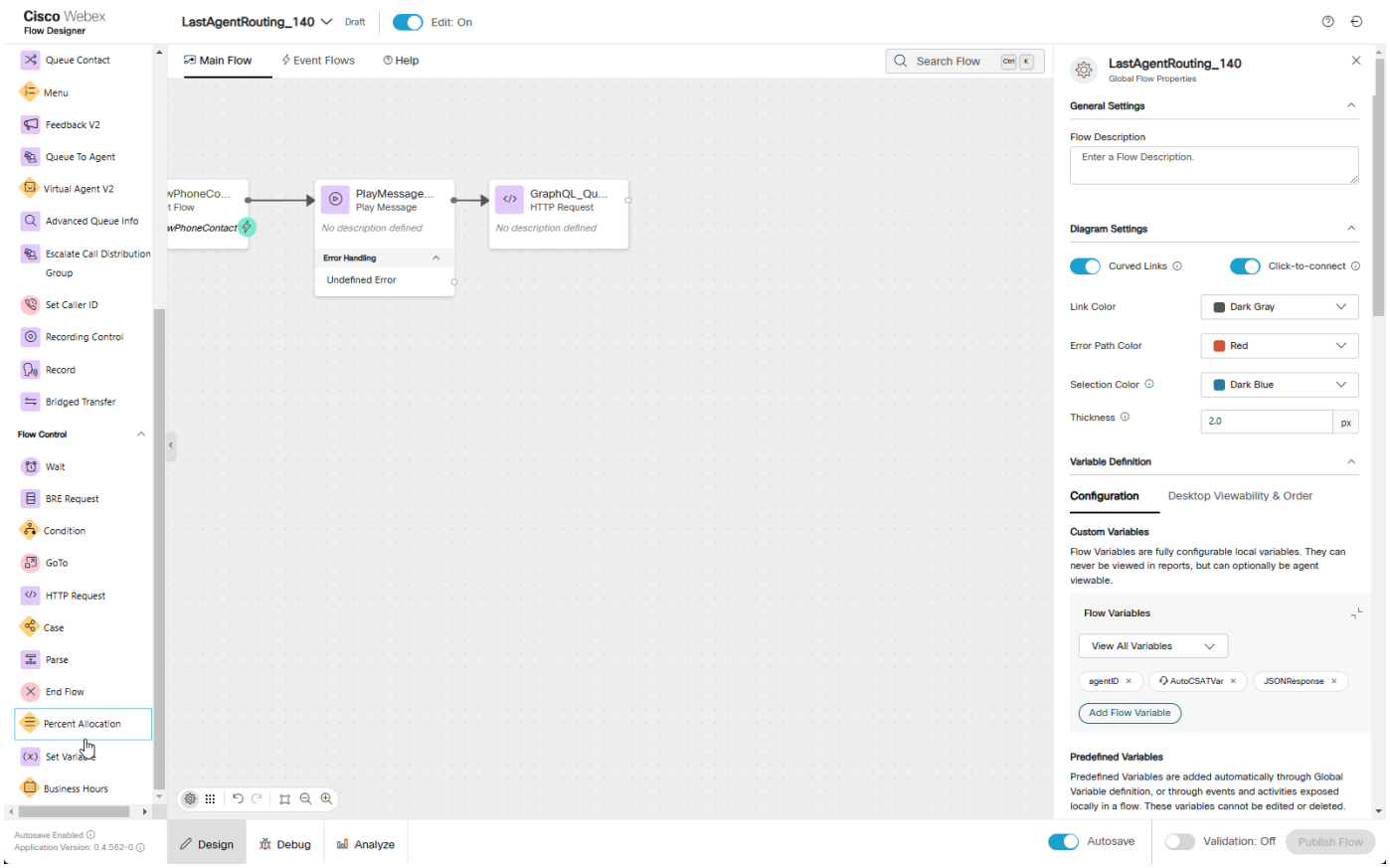
   Activity Label: **GraphQL_Response**

   Connect **GraphQL_Query** to this node

   We will connct **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

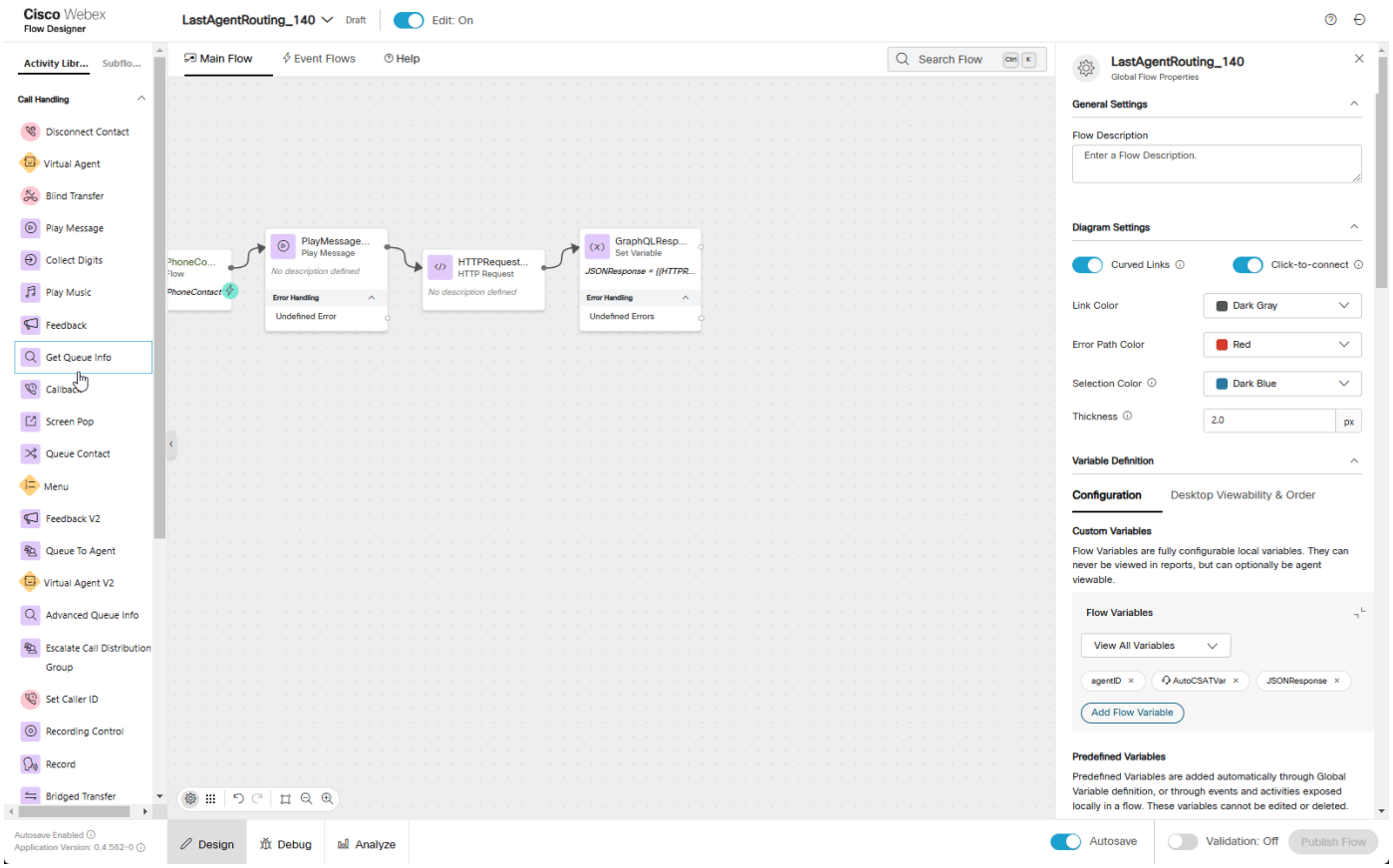
   Select **Build Expression**

   Expression: `{{agentID is empty}}`

   Change **Case 0** to **true**

   Change **Case 1** to **false**

   We will connect the **true** and **false** in future steps.



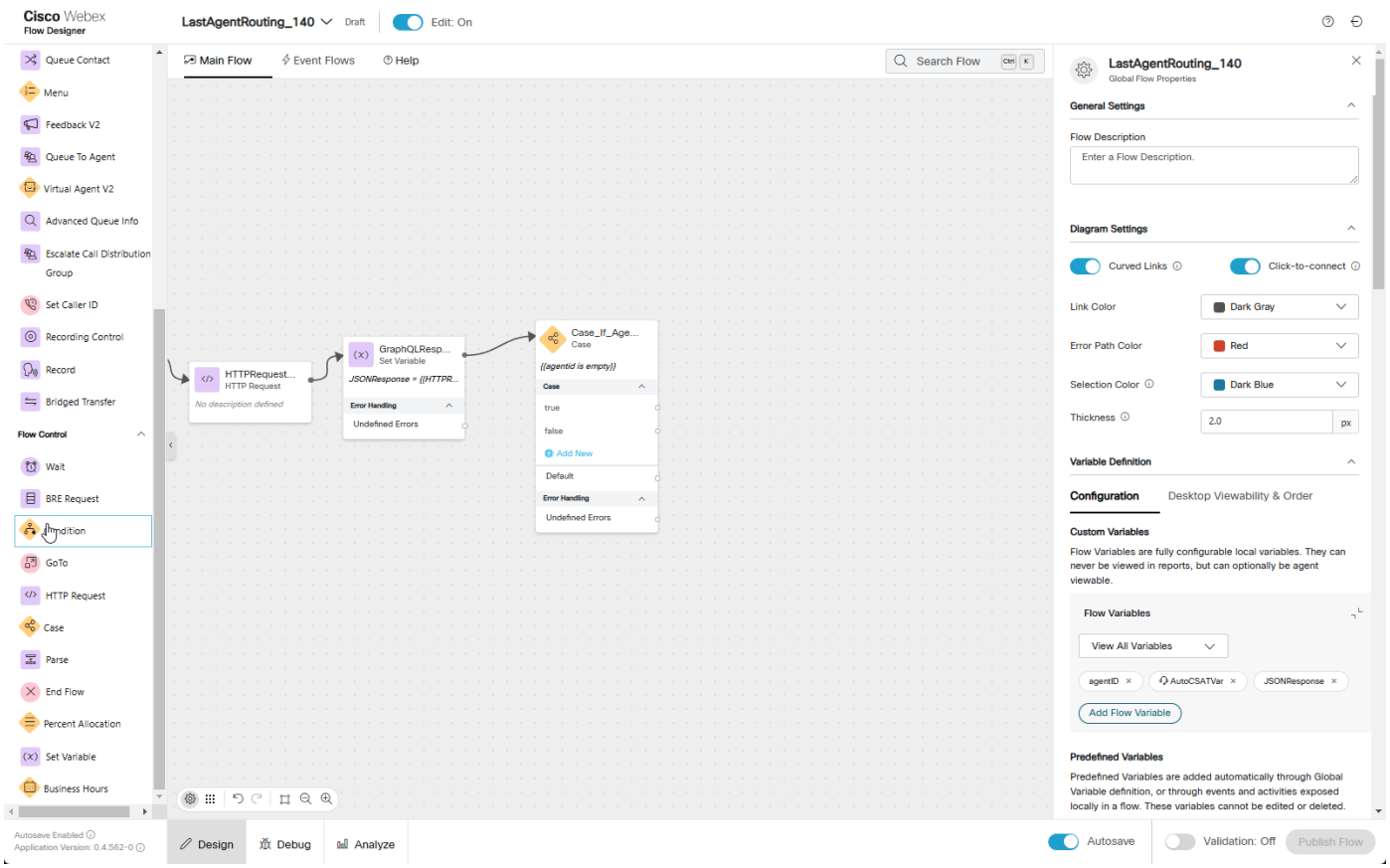

6. Add a **Condition** node

   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** node created in previous step to this **Queue To Agent**.

   Agent Variable: **agentID**

   Agent Lookup Type: **ID**

   Set Contact Priority: **True**

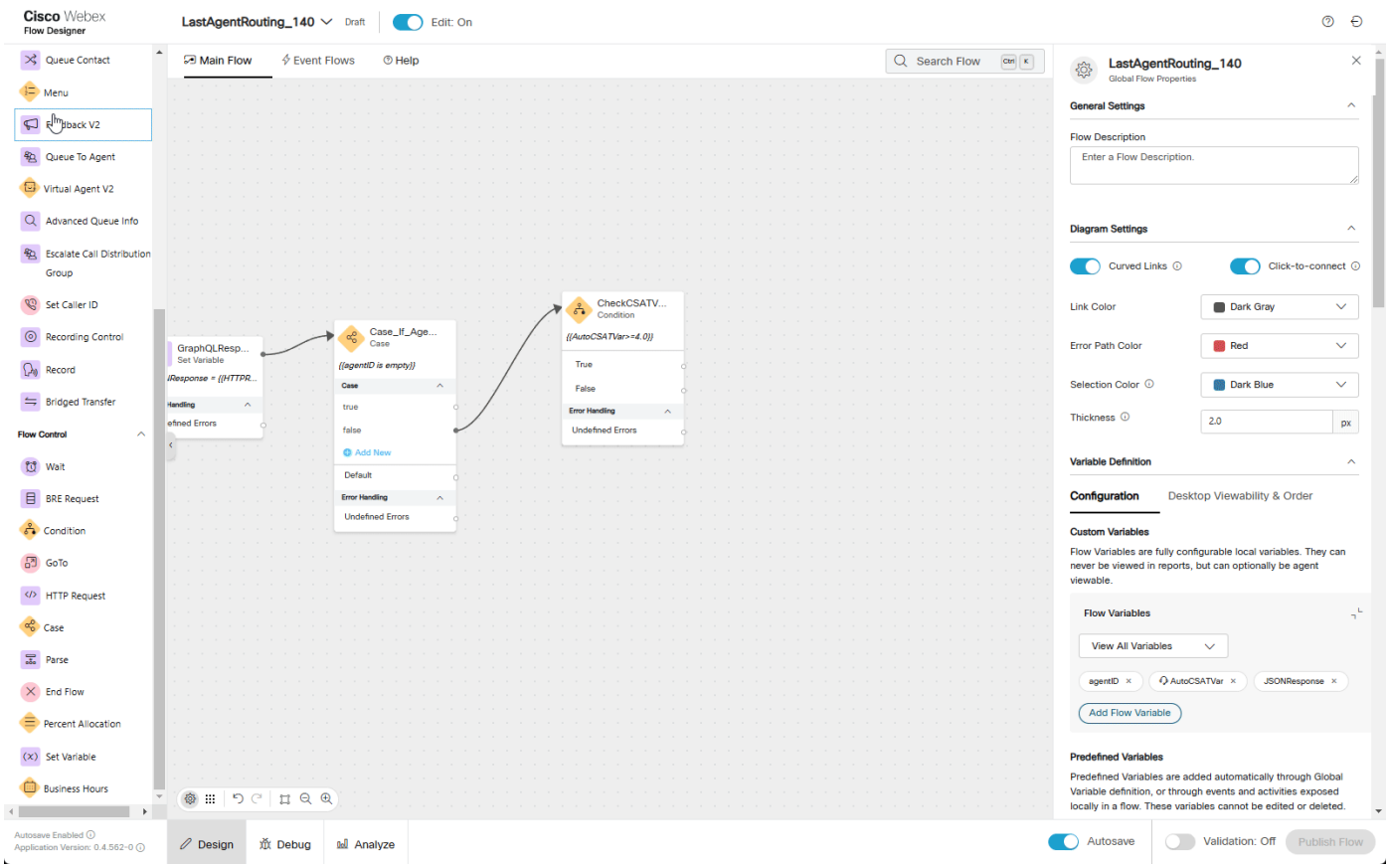
   Select Static Priority

   Static Priority Value: **P1**

   Reporting Queue: **Your_Attendee_ID_Queue**

   Park Contact if Agent Unavailable: **False**

   Recovery Queue: **Your_Attendee_ID_Queue**



8. Add a **Queue Contact** node

   Connect the **False** node edge from the **CheckCSATValue** node created in **Step 6** to this node

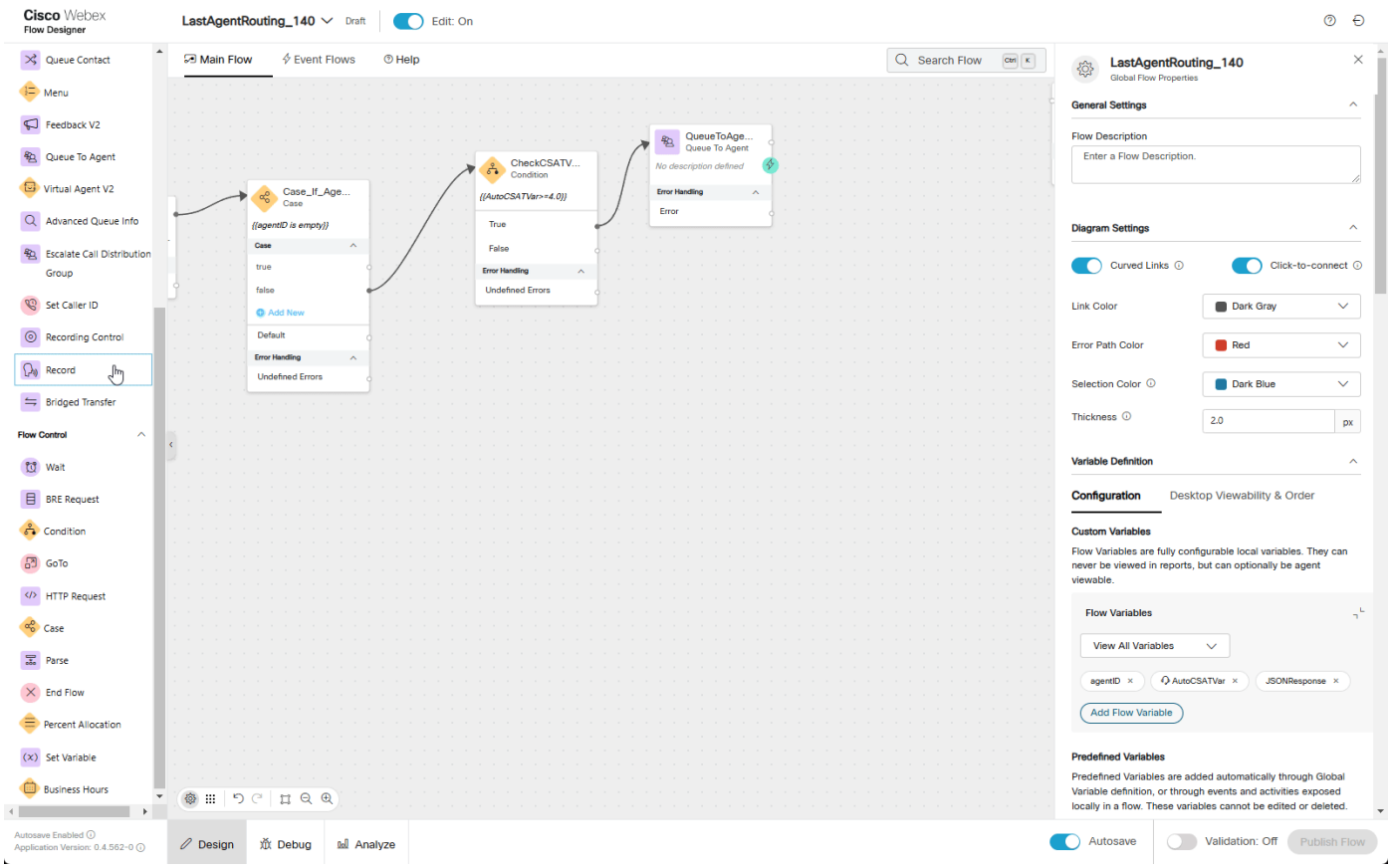
   Connect **true** node edge of **Case_If_AgentIDEmpty** node created in **Step 5** to this node

   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.

   Connect the output node edge from this node to the **DisconnectContact** node.

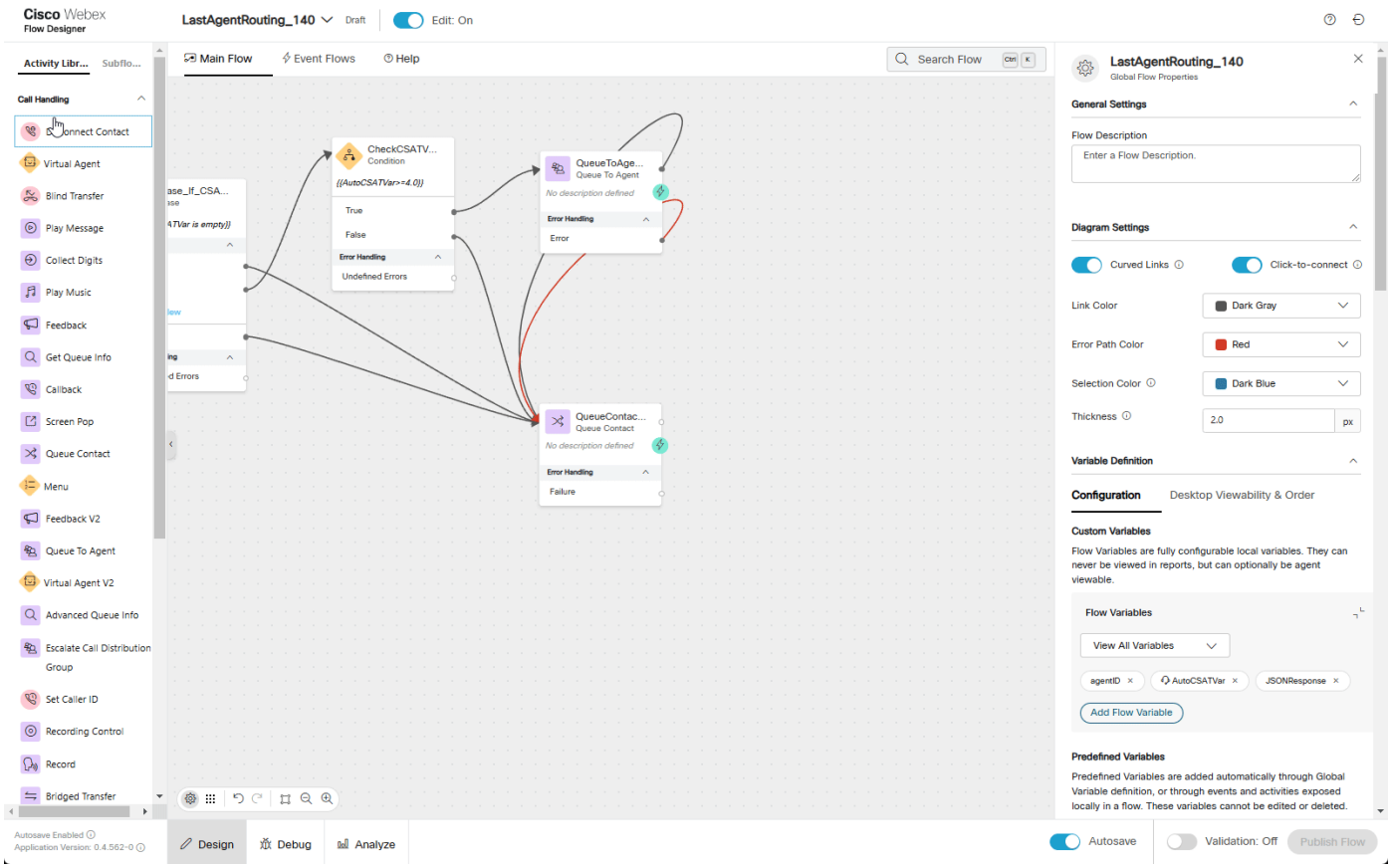
   Connect the **Queue Contact** node edge that we created in previous step to this **Subflow** node

   Subflow Label: **Latest**

   Enable automatic updates: **True**

   Subflow Input Variables: **None**

   Subflow Output Variables: **None**



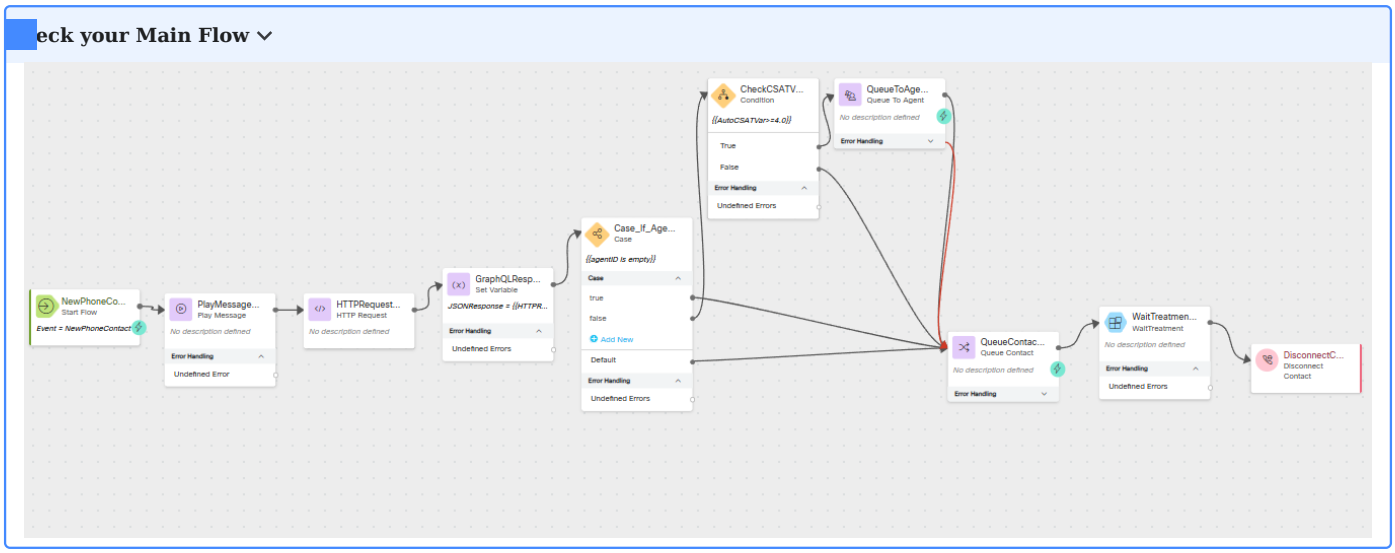

10. Navigate to **Event Flows** and add **GoTo** node to the canvas.

    Connect **AgentDisconnect** event node edge this node

    Destination Type: **Flow**

    Flow: **CCBU_PostCallSurvey_AutoCSAT**

    Choose Version Label: **Latest**




11. Publish your flow

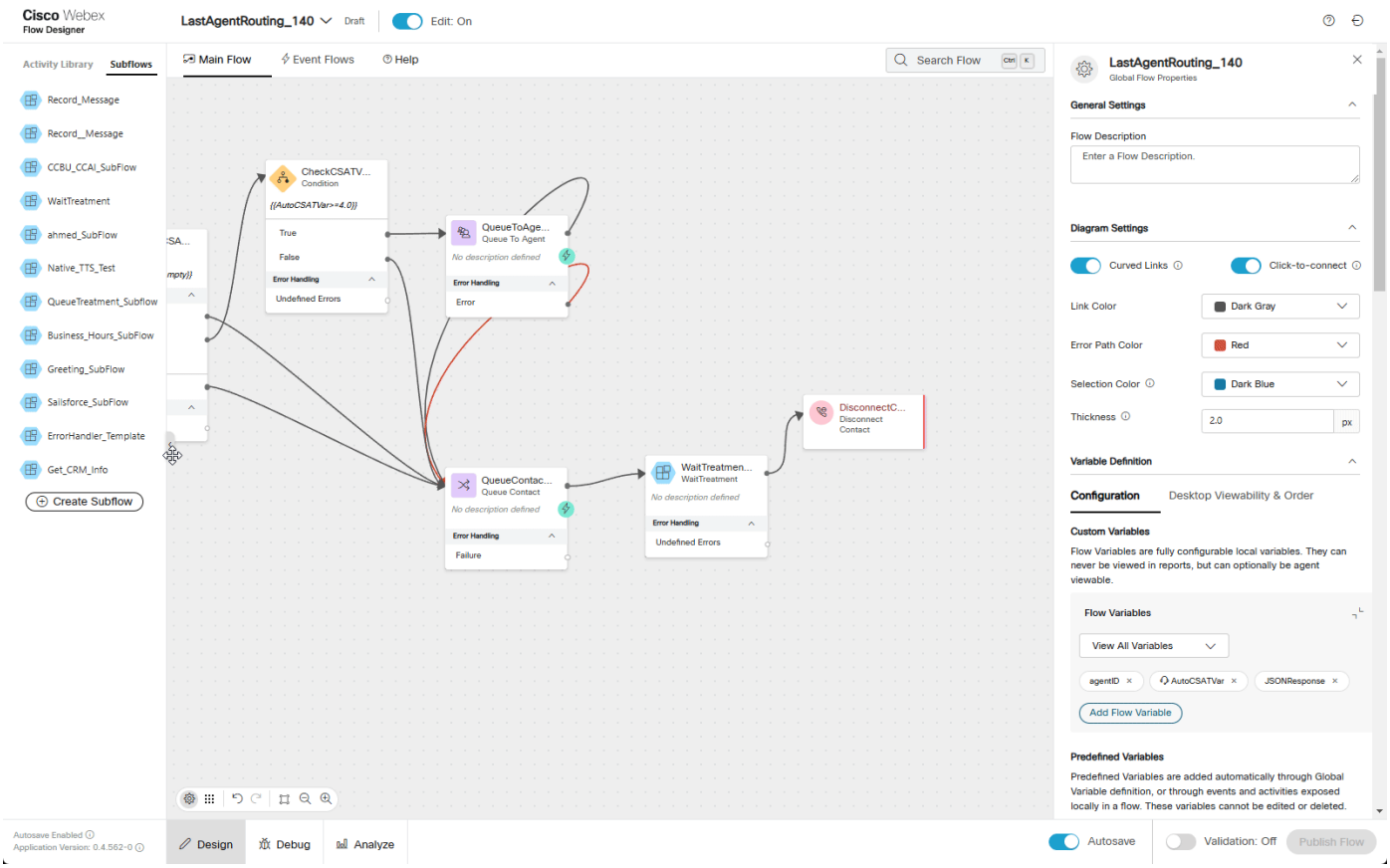
    Turn on Validation at the bottom right corner of the flow builder

    If there are no Flow Errors, Click **Publish**

    Add a publish note

    Add Version Label(s): **Latest**

    Click **Publish** Flow



12. Map your flow to your inbound channel

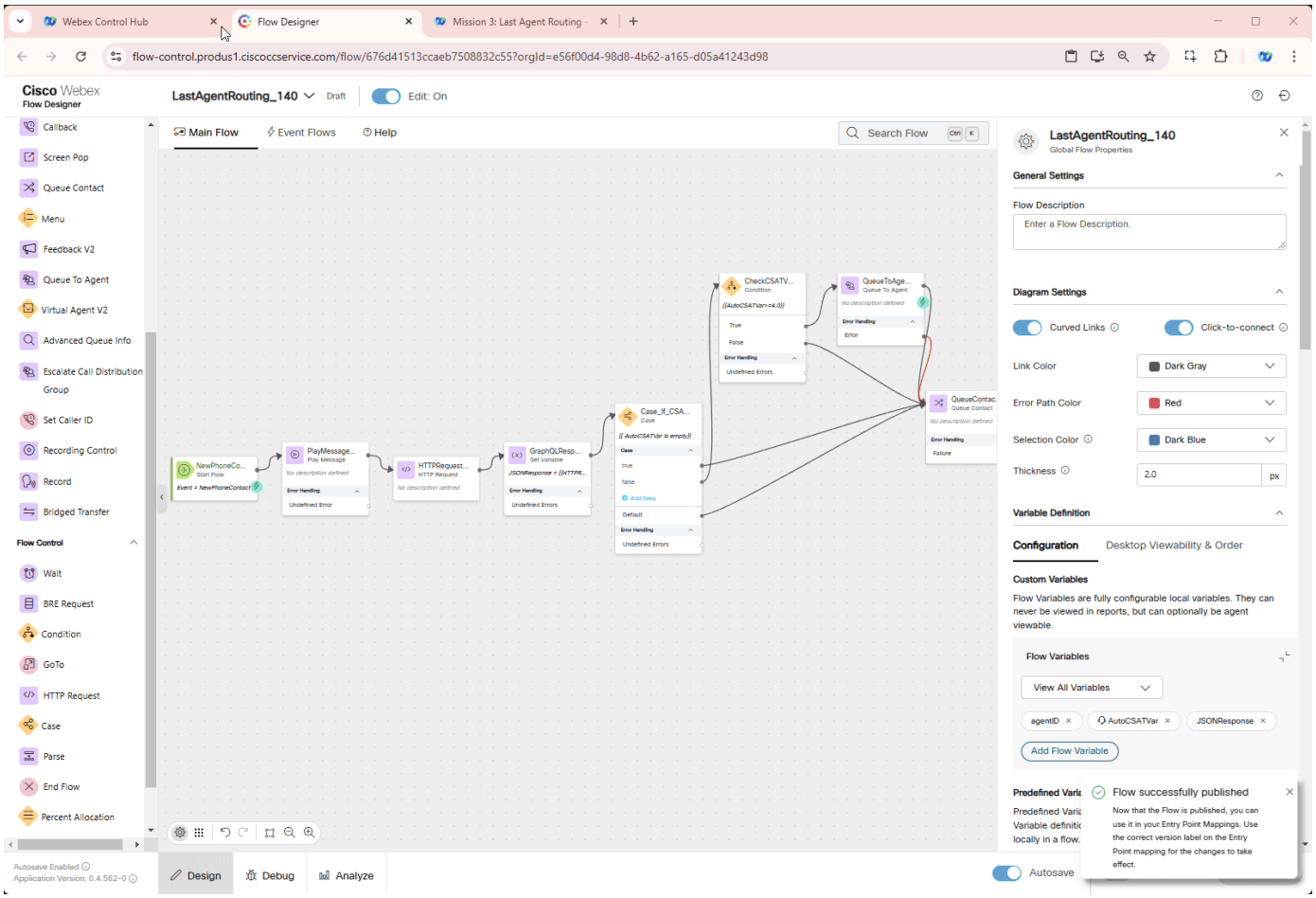
    Navigate to Control Hub > Contact Center > Channels

    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 with OKTA on it 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, place a call to your Inbound Channel number **Your_Attendee_ID_Channel**

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

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

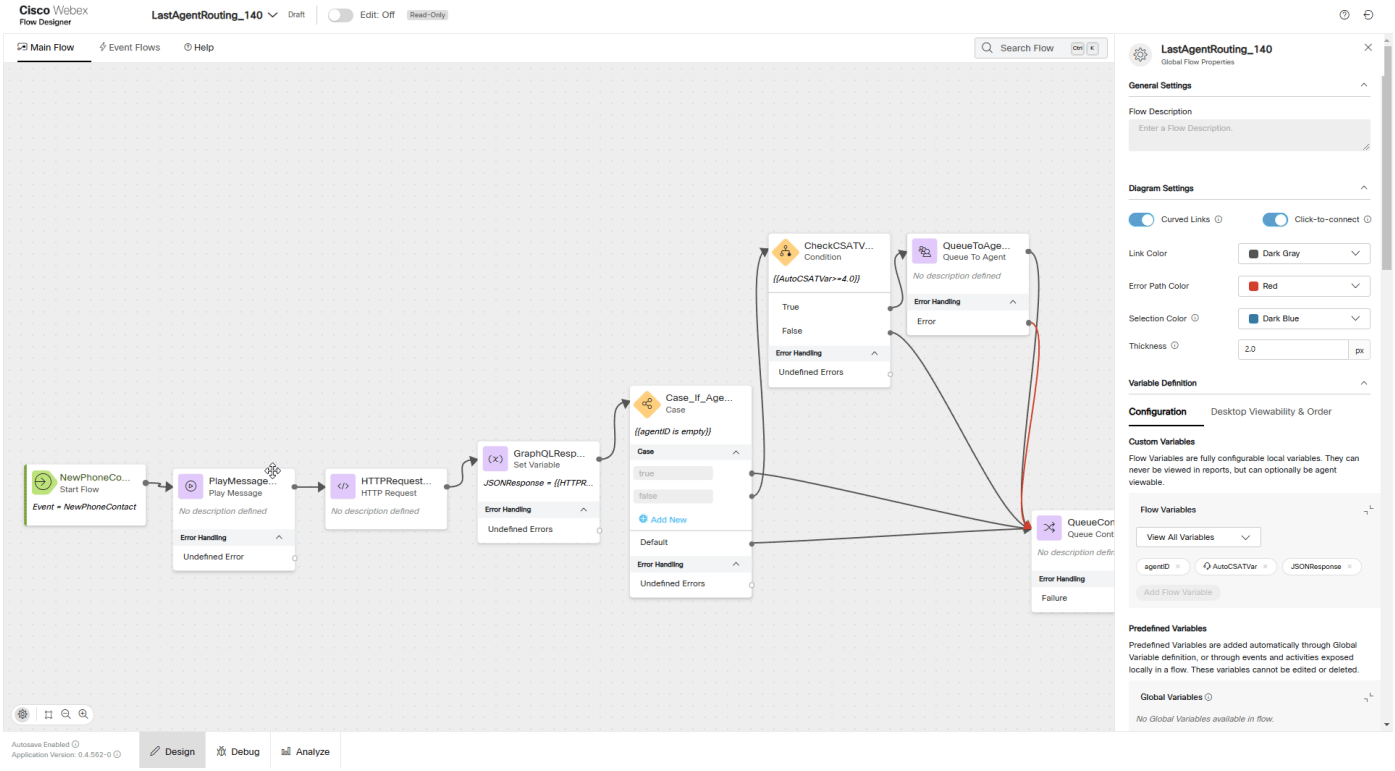
d. Select **4** or **5** on Webex App keypad.

3. In your flow, open the debuger 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.




4. Make sure your agent status is set to **Available**

5. 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 accept button.

b. If everything set correctly you should see Auto CSAT set to **4.0**

c. End the call and select a wrapup code if asked.

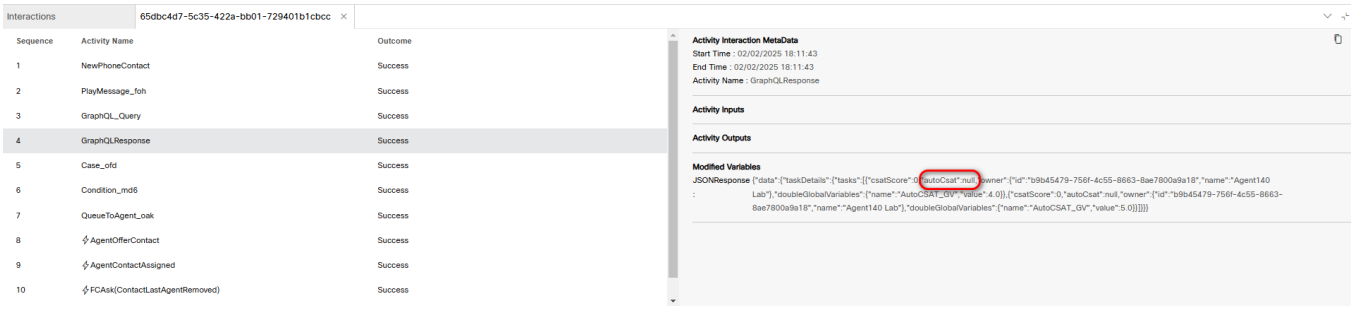
6. In your flow, open the debuger and select the latest call from the list (on top of the list).

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.

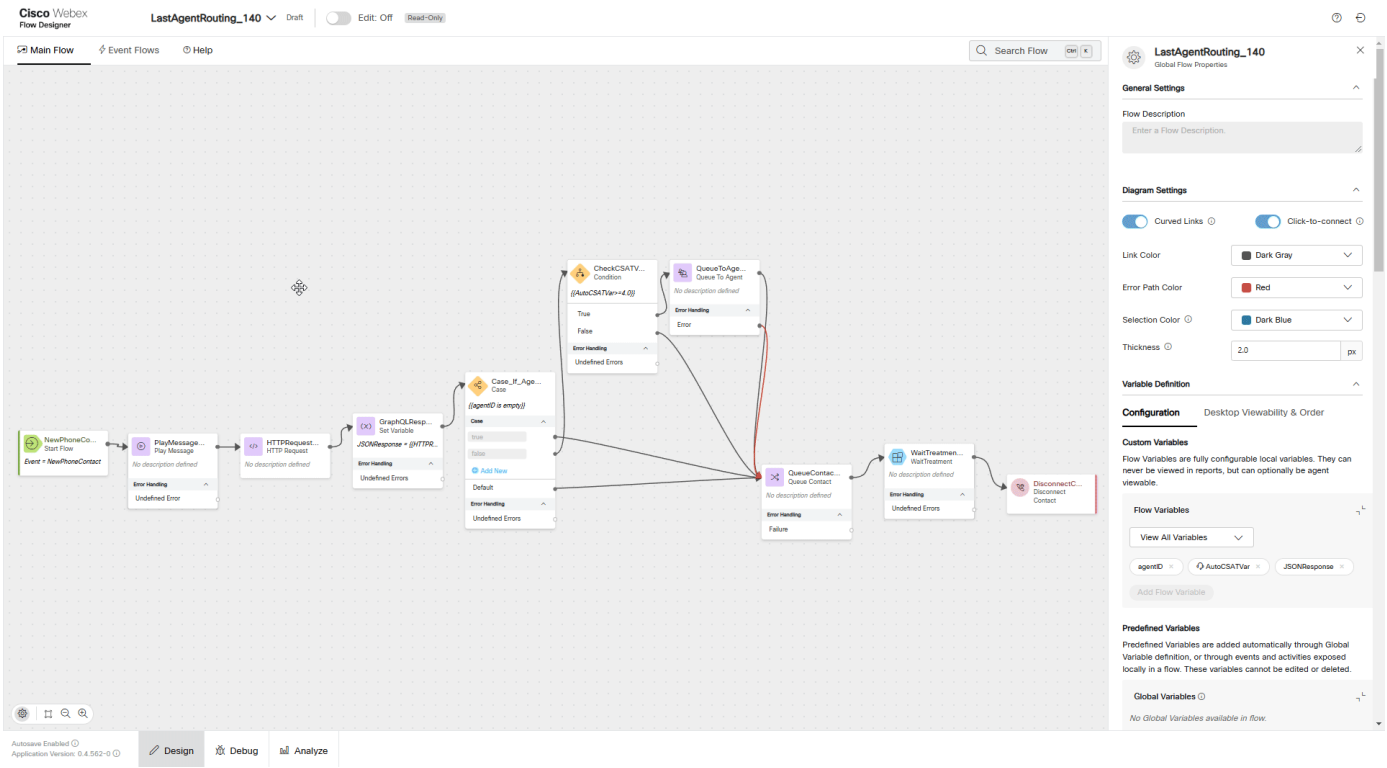
c. Select **GraphQLResponse**. In details panel on the right-hand side you should see **Modified Variables** has a JSON response.

> **Note**
>
> In JSON Response the **autoCsat** is set to **null**. This is expected because the lab environment lacks sufficient data to train the AI model for accurate scoring.




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

e. **CheckCSATValue** is now either equals **4** or **5** (depends on what you selected on previous call) which matches the condition hence the call arrived to your agent via **QueueToAgent** node.




**Congratulations, you have officially 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. Wait treatment Subflow which will provide Music in Queue and Queue Messages.
2. Connector for calling Webex Contact Center APIs

---

#### Build

1. Create a flow named **ReturningCaller_Your_Attendee_ID** then create a flow variable

   Name: **previousID**

   Type: **String**

   Default Value: Leave empty

2. Add a **Play Message** node for our welcome message

   Connect the **New Phone Contact** output node edge 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: **Welcome to the advanced routing and API integrations lab.**

3. 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: **Application/JSON**

   Copy this GraphQL query into the request body:

```
{"query":"query lastTen($from:Long! $to:Long! $timeComparator:QueryTimeType $filter:TaskFilters){task(from:$from,to:$to,timeComparator:$timeComparator,filter:
$filter){tasks{id status channelType createdTime endedTime origin destination direction terminationType isActive isCallback lastWrapupCodeName}}}","variables":
{"from":"{{now() | epoch(inMillis=true) - 600000}}","to":"{{now() | epoch(inMillis=true)}}","timeComparator":"endedTime","filter":{"and":[{"status":{"equals":"e
nded"}},{"origin":{"equals":"{{NewPhoneContact.ANI}}"}},{"connectedCount":{"gte":1}}]}}}
```

**panded Query For Understanding (optional)**

```
query lastTen(
  $from: Long!
  $to: Long!
  $timeComparator: QueryTimeType
  $filter: TaskFilters
) {
  task(from: $from, to: $to, timeComparator: $timeComparator, filter: $filter) {
    tasks {
      id
      status
      channelType
      createdTime
      endedTime
      origin
      destination
      direction
      terminationType
      isActive
      isCallback
      lastWrapupCodeName
    }
  }
}
```

```
Variables:
{
  "from": "{{now() | epoch(inMillis=true) - 600000}}", # time now - 10 minutes represented in EPOCH time(ms)
  "to": "{{now() | epoch(inMillis=true)}}", # time now represented in EPOCH time(ms)
  "timeComparator": "endedTime",
  "filter": {
    "and": [
      {
        "status": {
          "equals": "ended"
        }
      },
      {
        "origin": {
          "equals": "{{NewPhoneContact.ANI}}" # ANI or caller phone number
        }
      },
      {
        "connectedCount": {
          "gte": 1
        }
      }
    ]
  }
}
```

   Parse Settings:

   Content Type: `JSON`

   Output Variable: **previousID**

   Path Expression: `$.data.task.tasks[0].id`

4. 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.

5. Add a **Play Message** node

   Connect the **False** node edge from the previous step to this 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: **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.**

6. 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**

   Select Static Priority

   Static Priority Value: **P1**

7. Add a **Subflow** 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.

   Connect the output node edge from the **Queue Contact** node added in the previous step to this node.

   Subflow Label: **Latest**

   Enable automatic updates: **True**

   Subflow Input Variables: **None**

   Subflow Output Variables: **None**

   Connect the output node edge from this node to the **Disconnect Contact** node added in the next step.

8. Add a **Disconnect Contact** node

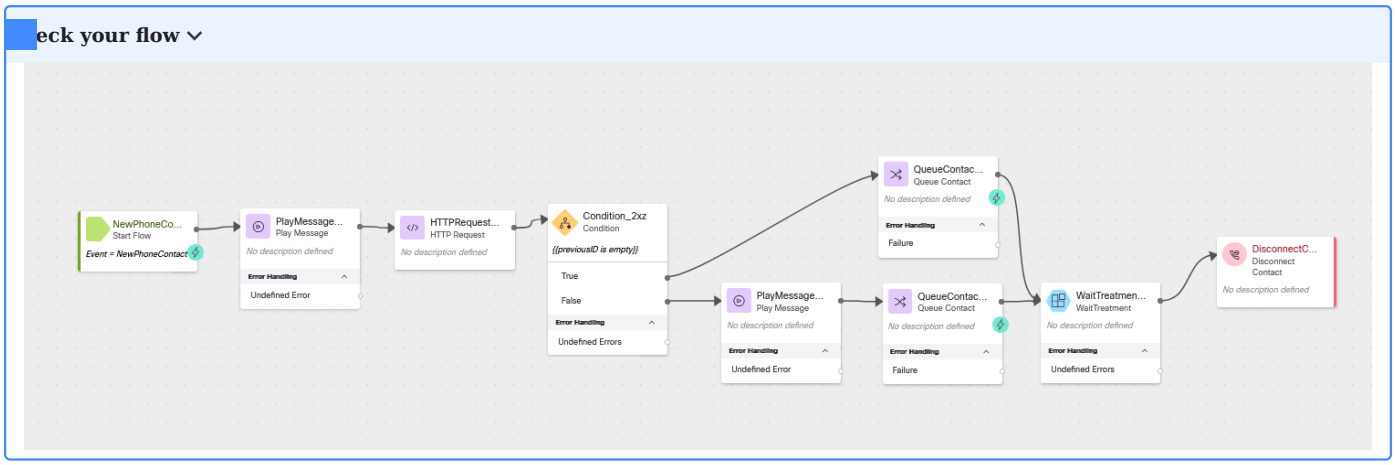
9. Add a **Queue Contact** node

   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 **Subflow** node




10. Publish your flow

    Turn on Validation at the bottom right corner of the flow builder

    If there are no Flow Errors, Click **Publish**

    Add a publish note

    Add Version Label(s): **Latest**

    Click **Publish** Flow

11. 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 with OKTA on it 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 not set to **Available**

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

b. After you hear the queue treatment start, you can abandon the call

3. Using Webex, place another call to your Inbound Channel number **Your_Attendee_ID_Channel**

4. On your Agent Desktop, set your status to available

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

b. After a few moments end the call and select a wrapup code.

5. In your Flow:

a. Open the Debugger

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

c. Trace the steps taken in the flow

6. Answer these questions:

a. Was the call queued with priority?

i. Why or why not?

7. Close the Debugger
8. Using Webex, place another call to your Inbound Channel number **Your_Attendee_ID_Channel**
9. On your Agent Desktop, set your status to available
   a. You should be offered a call, click on the accept button. (You may want to mute the mic on both Webex and the Agent Desktop)
   b. After a few moments end the call and select a wrapup code.
10. In your Flow:
    a. Open the debugger
    b. Select the last interaction (at the top of the list)
    c. Trace the steps taken in the flow
11. Answer these questions:
    a. Was the call queued with priority?
       i. Why or why not?
    b. If you called another Inbound Channel number with the same flow logic, would your call be prioritized?
       i. How could you change this behavior?

---

**Congratulations, you have officially completed Routing Returning Callers 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.

Key missions included:

- **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 a call ends and they call back within 10 minutes, maintaining conversation continuity.

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.

> **ıy Flow Templates? [Optional]**

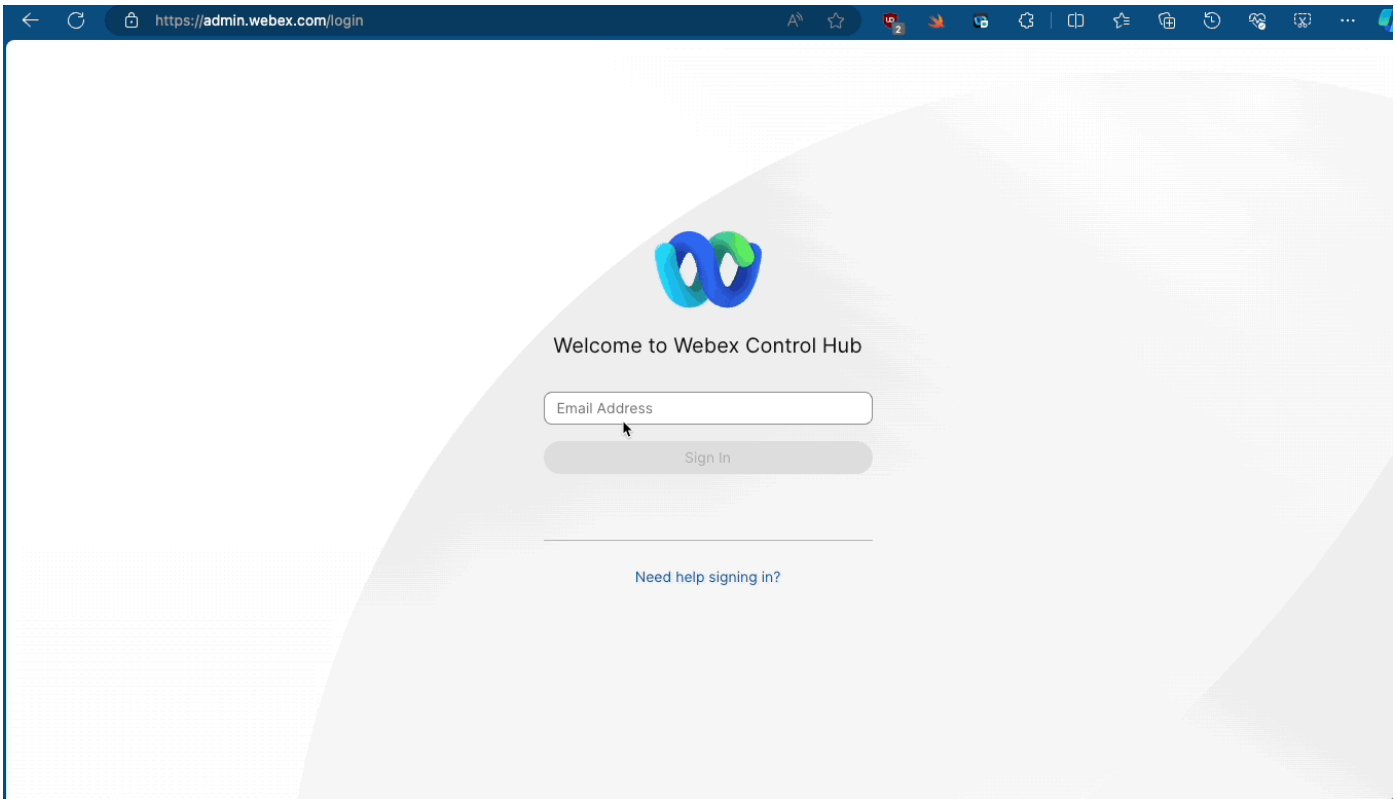
Flow Templates in Webex Contact Center are an essential feature for flow developers, offering a range of benefits that streamline the development process and enhance the efficiency and consistency of flow creation. Here’s what they bring to the table:

- **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

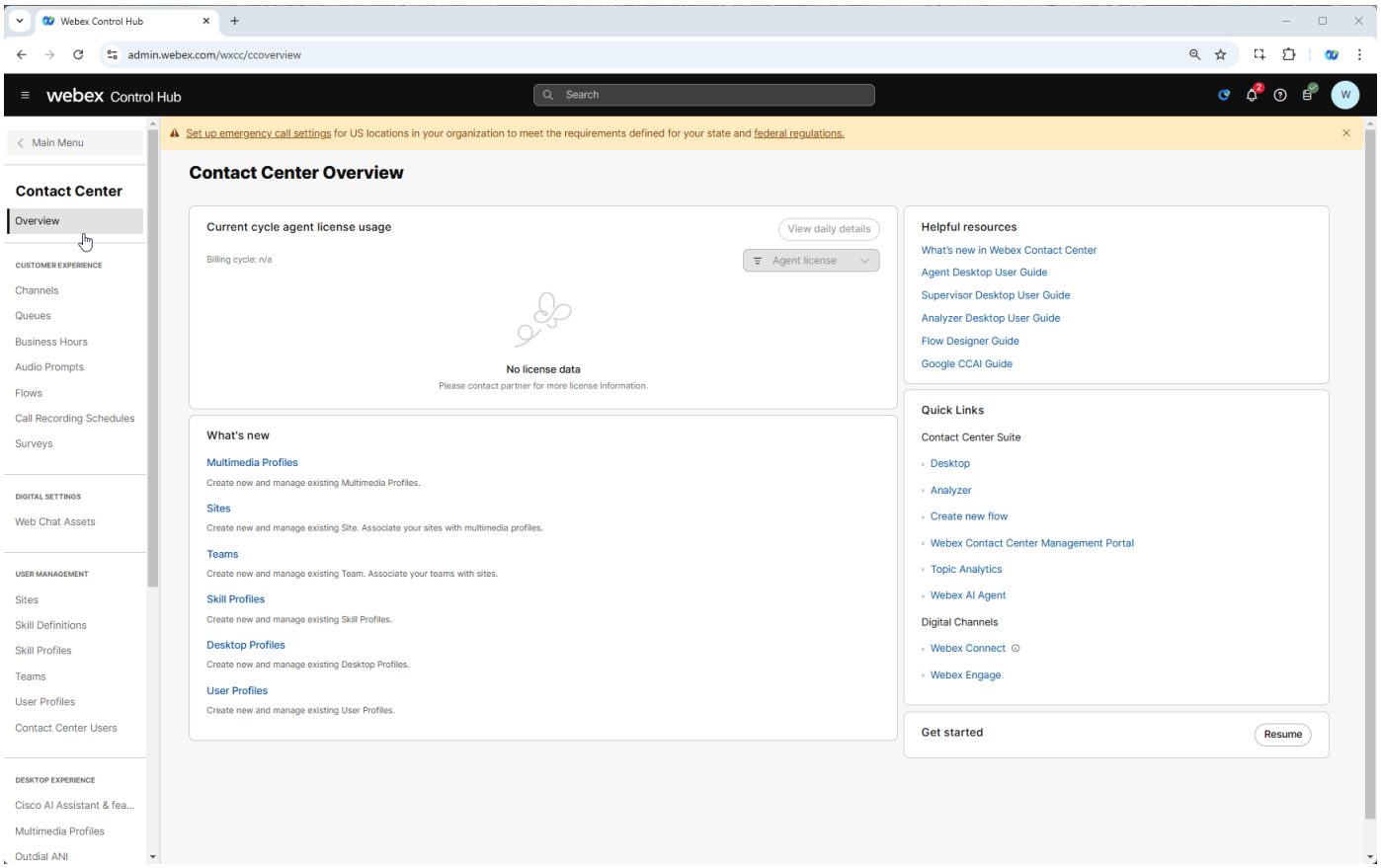
1. Login into Webex Control Hub by using your Admin profile. Your login will be of the format **wxcclabs+admin_IDYour_Attendee_ID@gmail.com**. You will see another login screen with OKTA on it where you may need to enter the email address again and the password provided to you.




**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 it
3. Navigate to **Flows**, click on **Manage Flows** dropdown list and select **Create Flows**
4. New Tab will be opened. Navigate to **Flow Templates**
5. Choose **Simple Inbound Call to Queue** template and click **Next**. You can open View Details and to see observe flow structure and read flow description
6. Name your flow as **Main_Flow_Your_Attendee_ID**. Then click on Create Flow



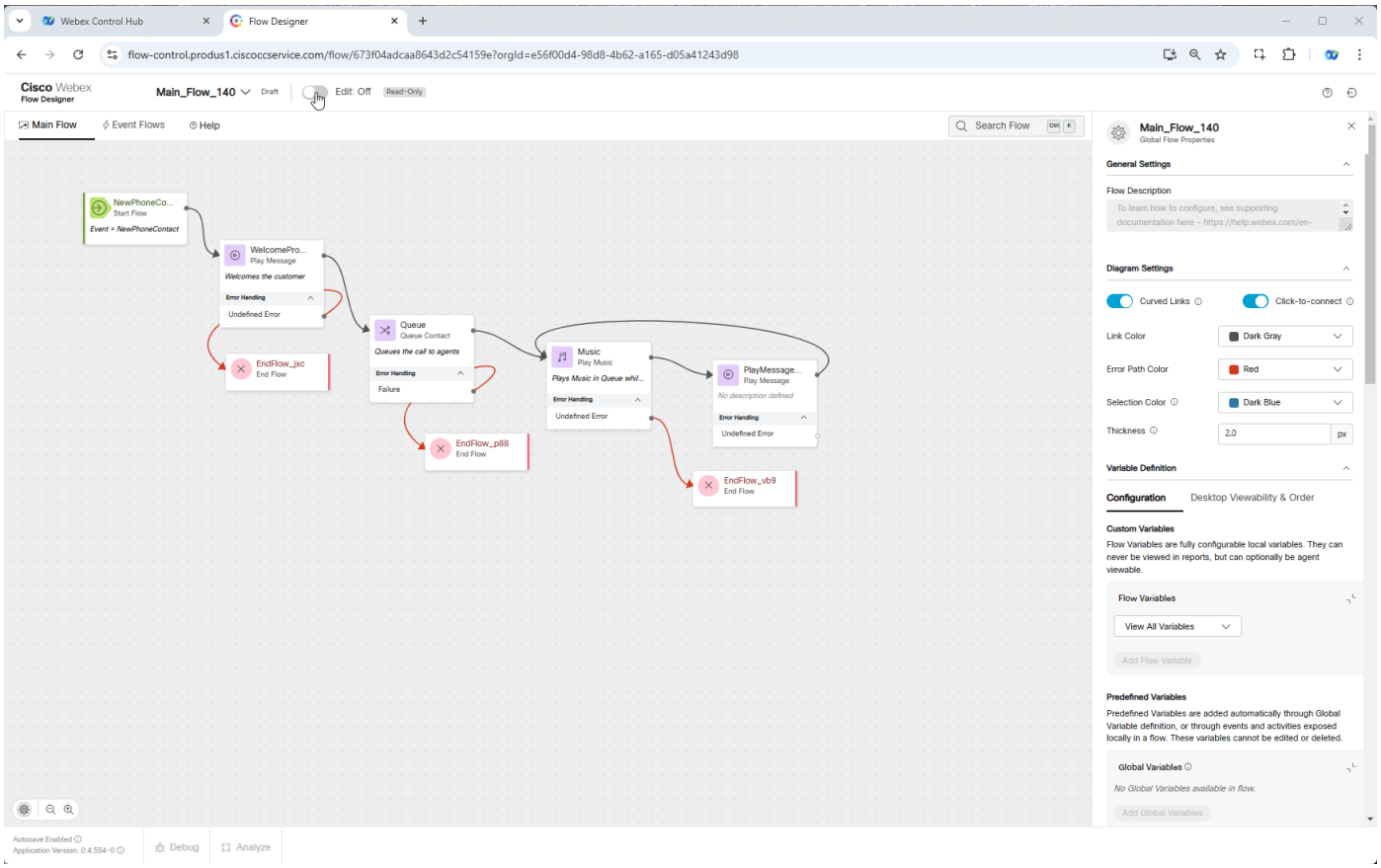

7. **Edit** should be set to **On** when you create new flow, but if not switch it from **Edit: Off** mode to **Edit: On**. Select **Play Message** node with label **WelcomePrompt** and on the node settings modify **Text-to-Speech Message** to any greetings you like. This message will be the first message you hear while calling to your script.
8. Select **Queue** node. On the **General settings** keep Static Queue checked and select queue **Your_Attendee_ID_Queue** from the drop down list

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

9. [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.
10. On 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.

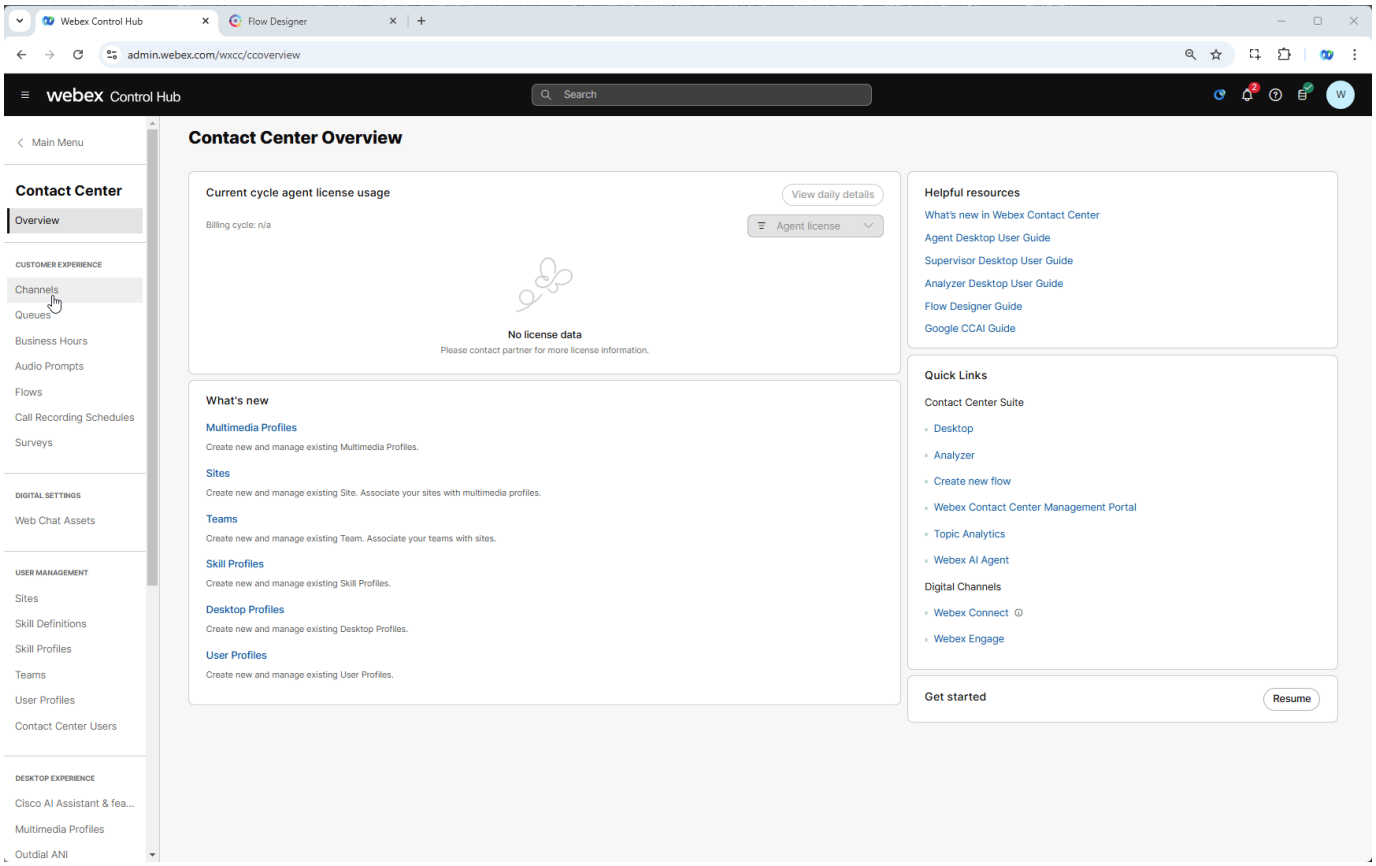
11. Click **Publish** Flow




12. In popped-up window, click on dropdown menu to select **Latest** label, then click **Publish**.
13. 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** .
14. Click on **Your_Attendee_ID_Channel**
15. In **Entry Point** settings section change the following, then click **Save** button:

    Routing Flow: **Main_Flow_Your_Attendee_ID**

    Version Label: **Latest**

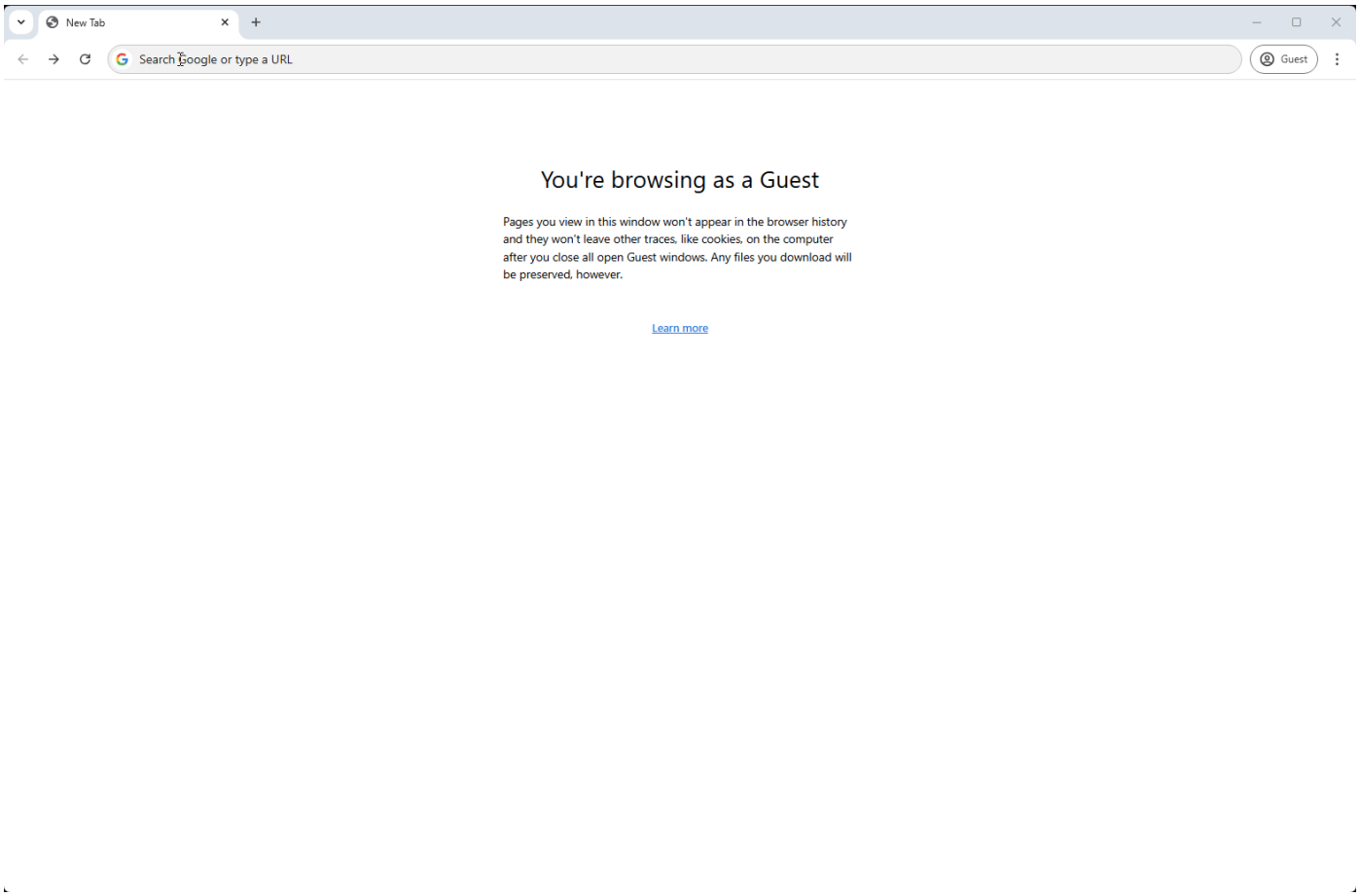



---

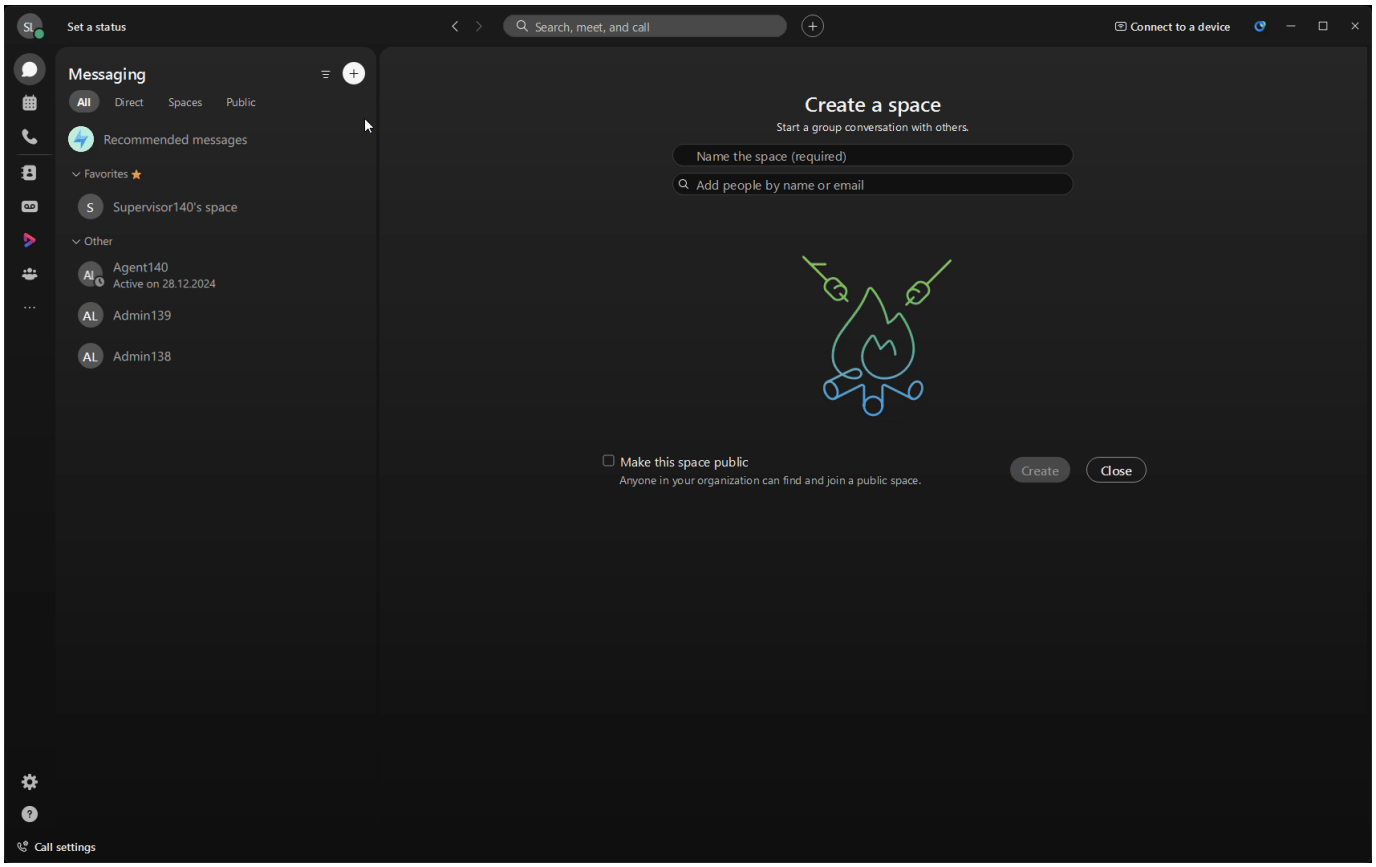
CHECKPOINT TEST

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 OKTA on it where you may need to enter the email address again and the password provided to you.
2. Select your Team **Your_Attendee_ID_Team**. Click **Submit**. Allow browser to access Microphone by clicking **Allow** on every visit.
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.




---

### Enhance Your Flow by adding Language

MISSION DETAILS

Your mission is to:

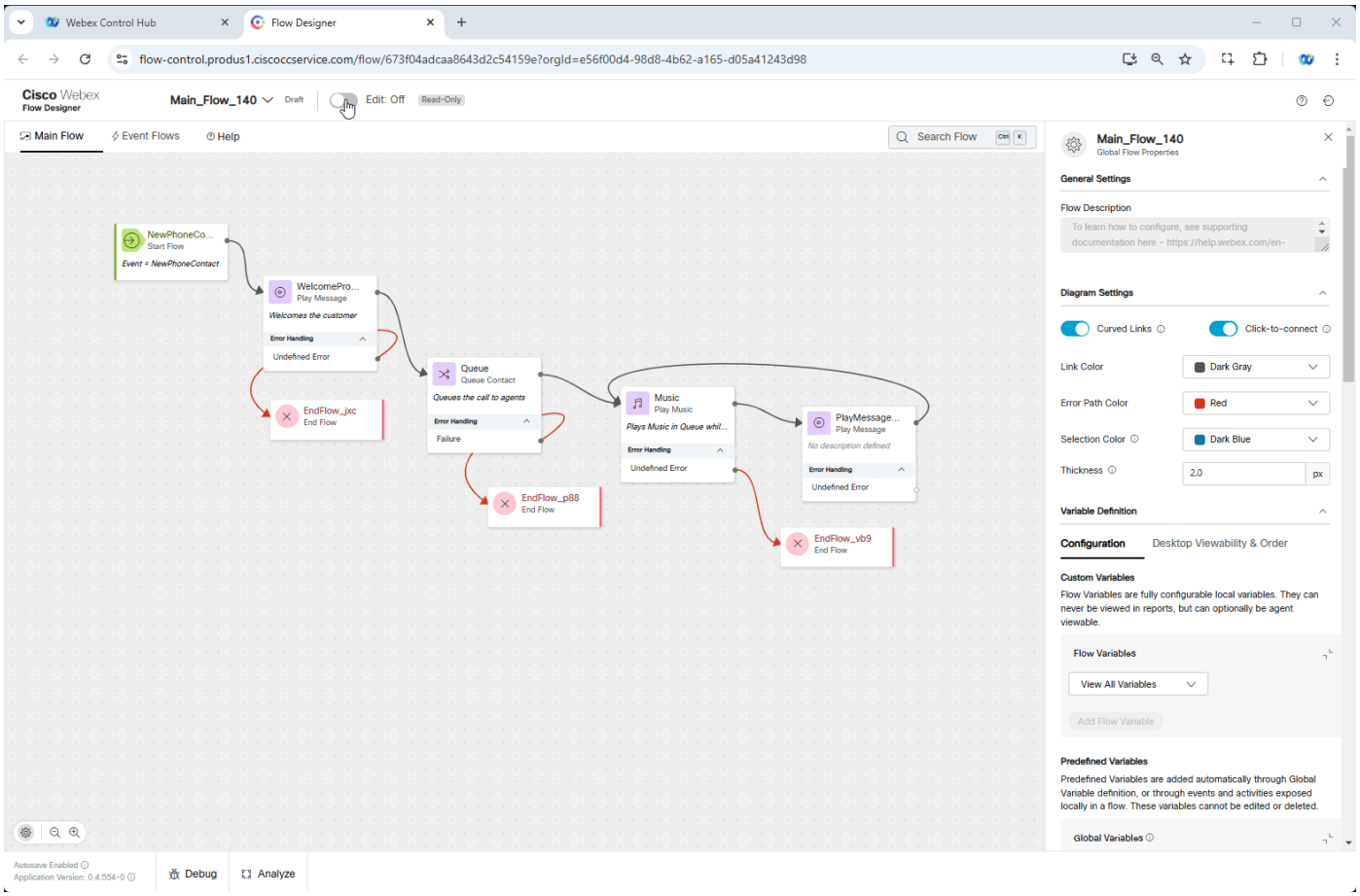
- Use the same flow created in the previous section.
- Modify the TTS section to use **en-AU** (English - Australia) and connect the Set Variable node as illustrated below.
- Place a call to verify and validate the speech functionality.

**xt-to-Speech (TTS) in Webex Contact Center[Optional]**

All supported languages can be found 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. On the right hand side you will see the **Global Flow Properties** Panel. 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.



3. Add a **Set Variable** with following configuration.

   Delete connection between **NewPhoneContact** and **WelcomePrompt**

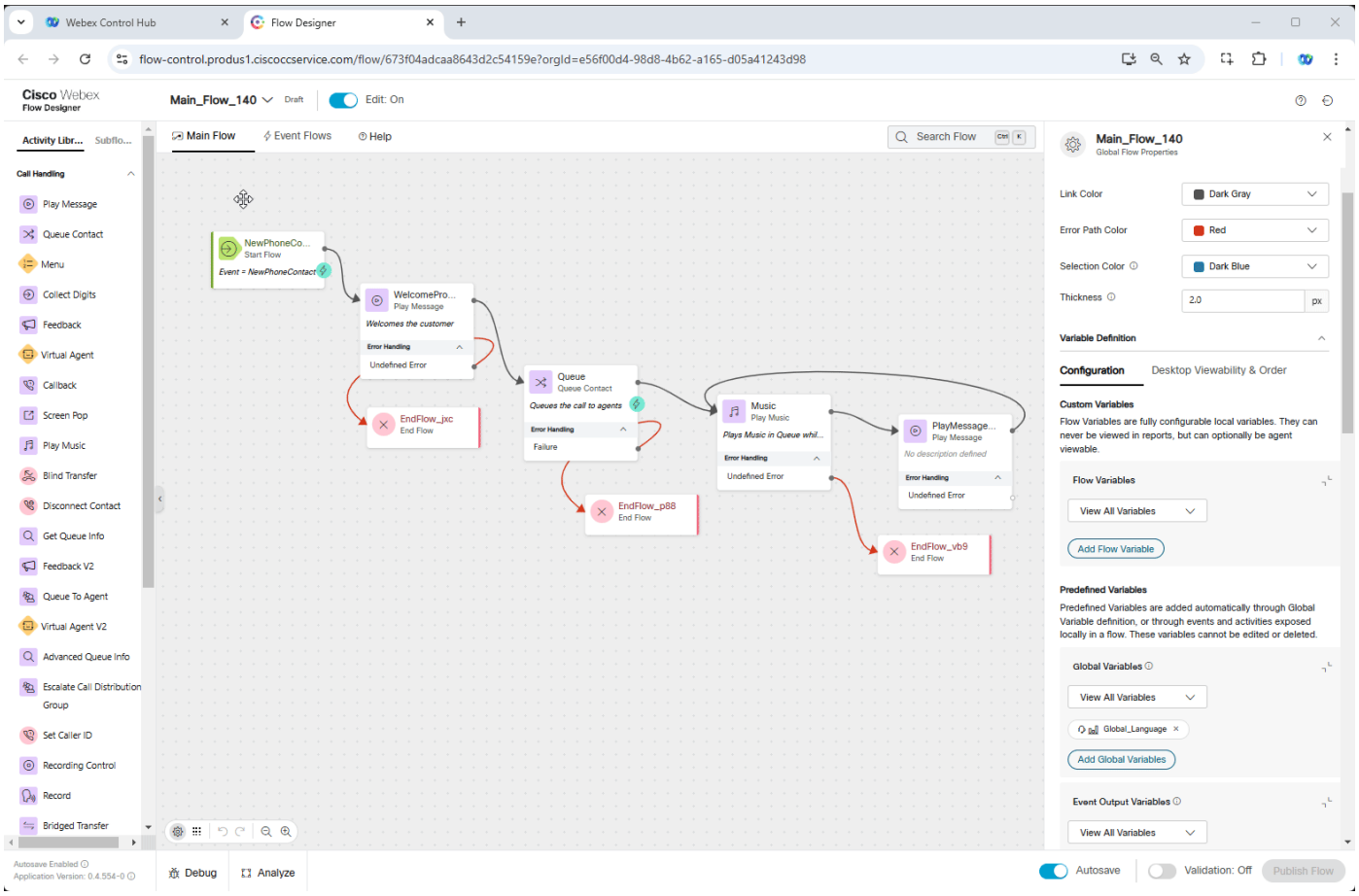
   Connect **NewPhoneContact** to **Set Variable**

   Connect **Set Variable** to **WelcomePrompt**

   Variable: **Global_Language**

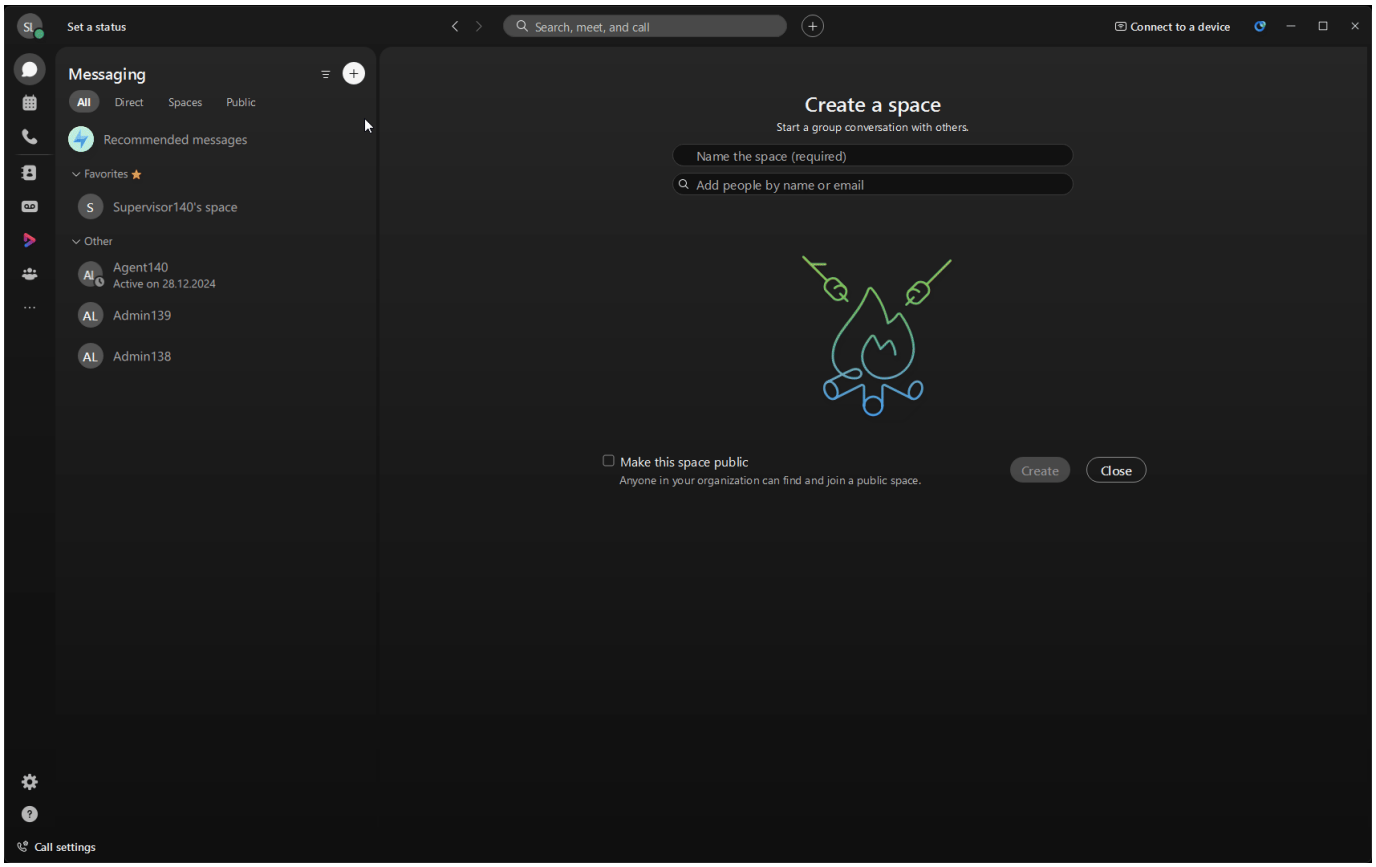
   Set Value: **en-AU**

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




TESTING

1. Open your Webex Desktop and make your agent **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.




1. Verify if the TTS language changed

**Congratulations on completing another mission.**

## 4.1.2 Mission 2: Adding Call-back functionality to your flow

#### 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. Drag **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

   Under Custom Menu Links:

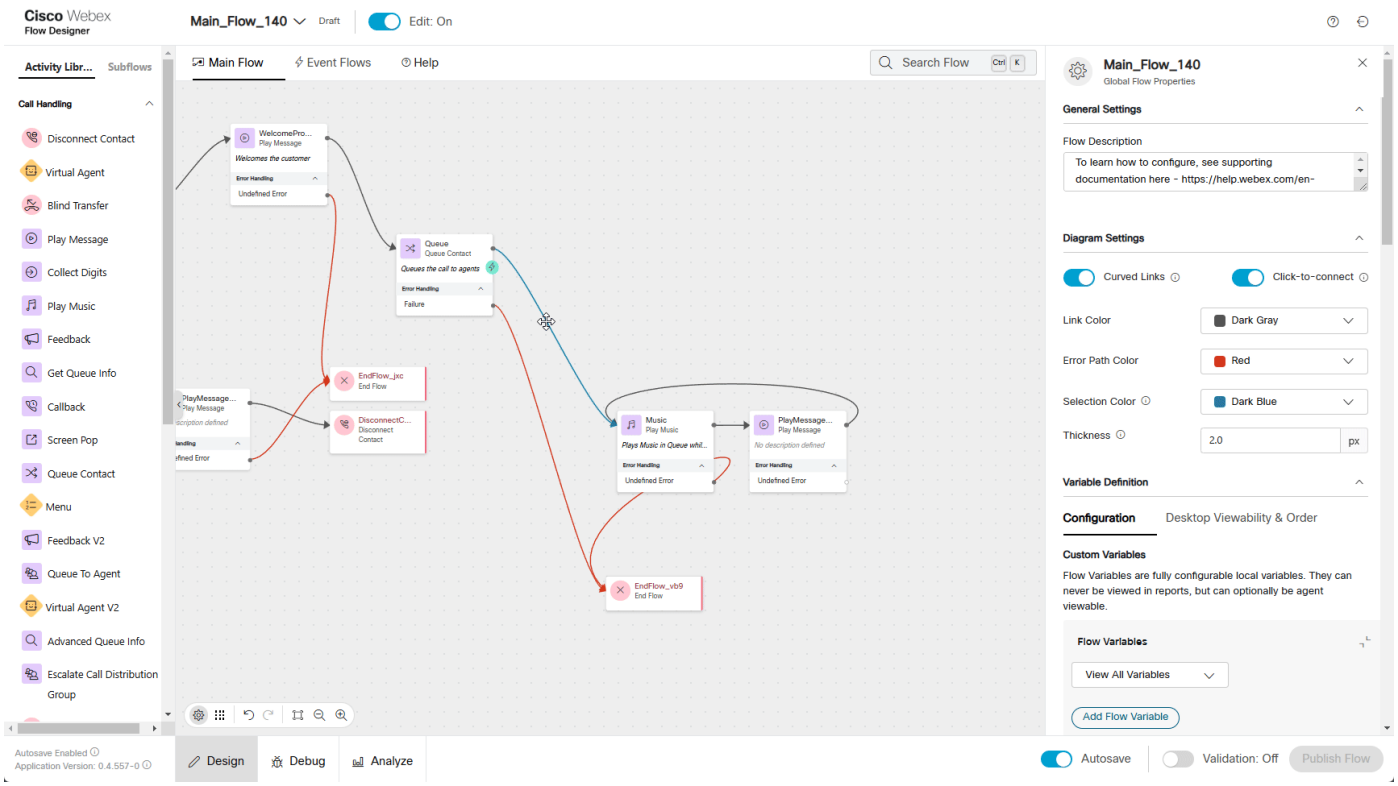
   Change first Digit Number **0** to **1**, add Link Description as **Callback**

   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. Drag **Collect Digits** nodes

   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

   Advanced Settings:

   No-Input Timeout **5**

   Make Prompt Interruptible: **True**

   Minimum Digits: **11**

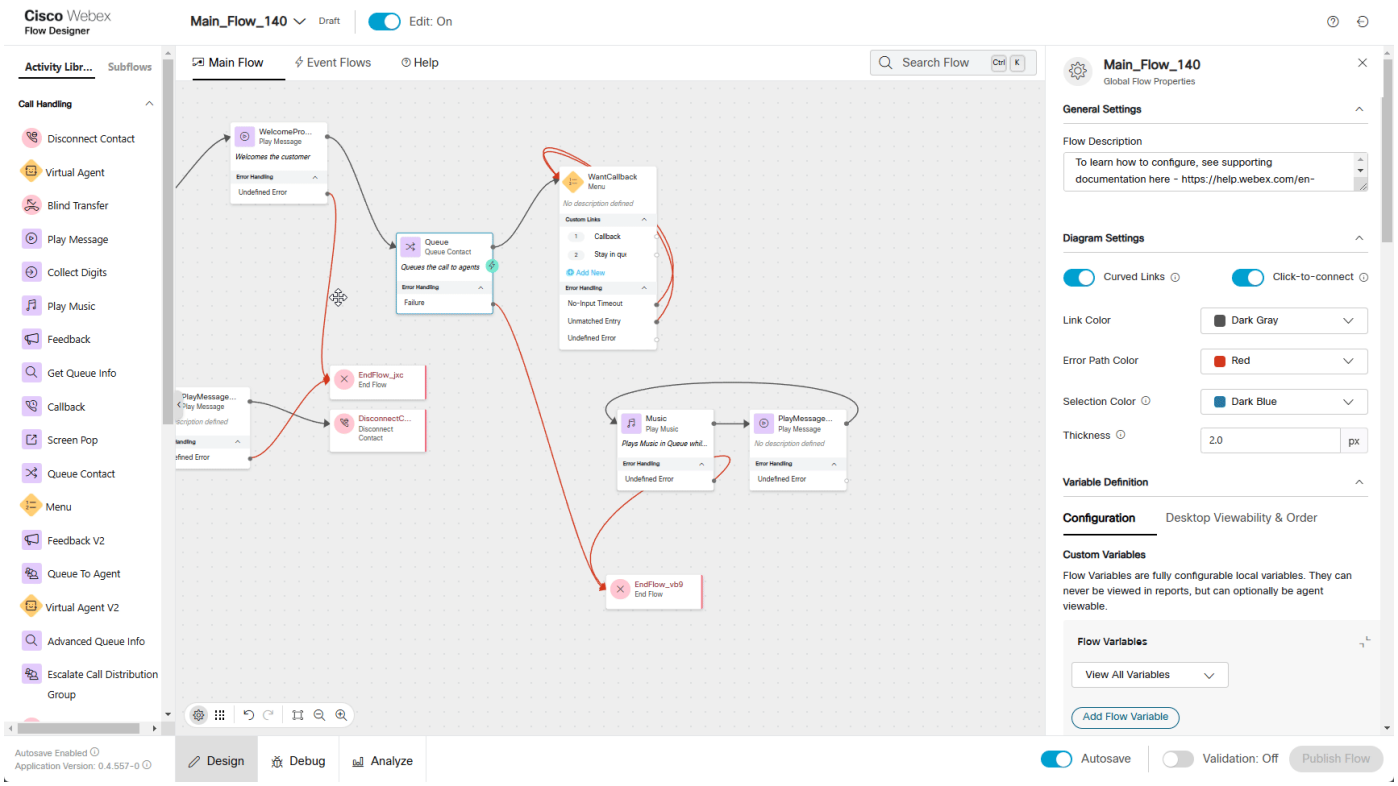
   Maximum Digits: **11**

   Connect **No-Input Timeout** to the front of the **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. Drag 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 text: ***You entered*** *{{NewNumber.DigitsEntered}}*. ***Press 1 if the number is correct. Press 2 if you want to re-enter the number.***

   Delete the selection for Audio File

   Custom Menu Links:

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

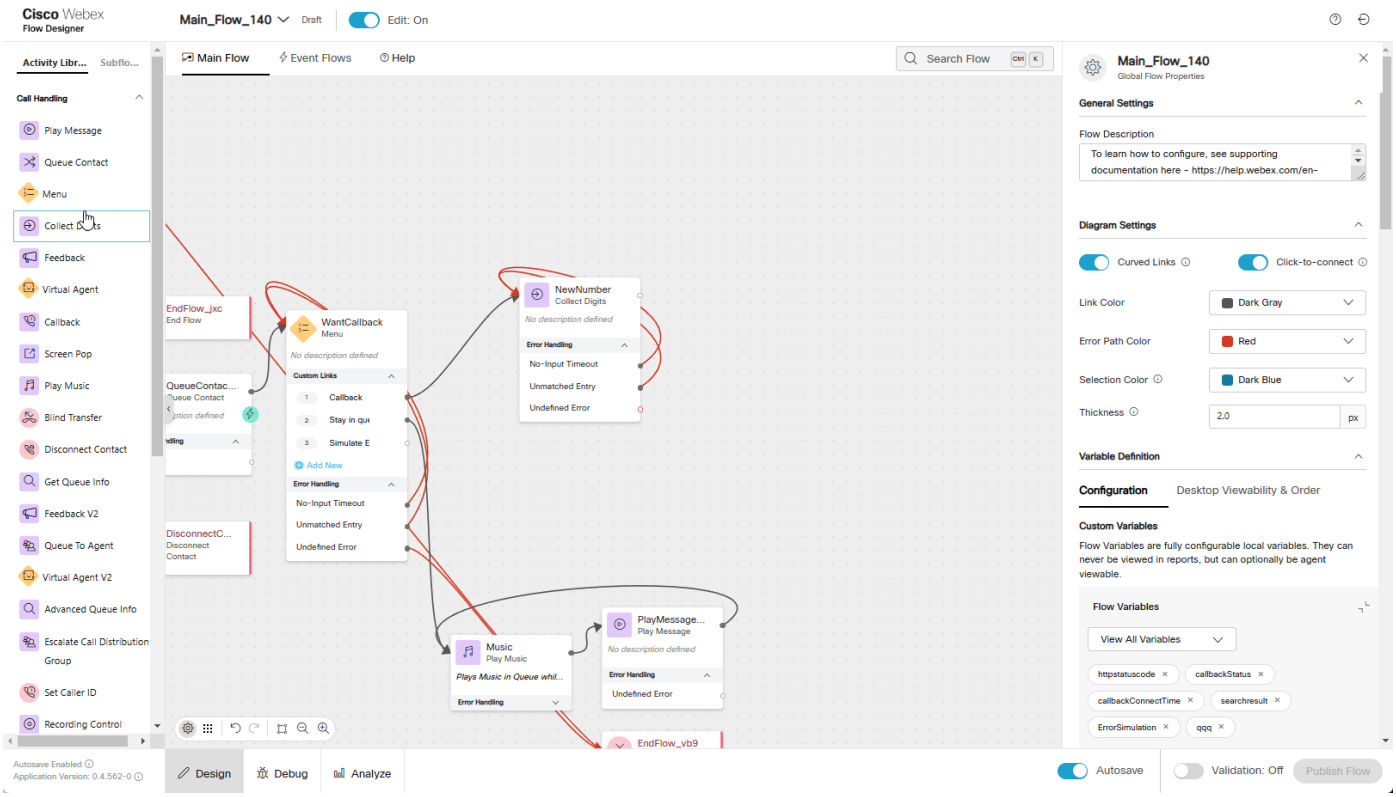
   Add New Digit Number as **2** with Link Description **Number Not OK**

   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 **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 **CallBack** node

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

   Enable Text-To-Speech

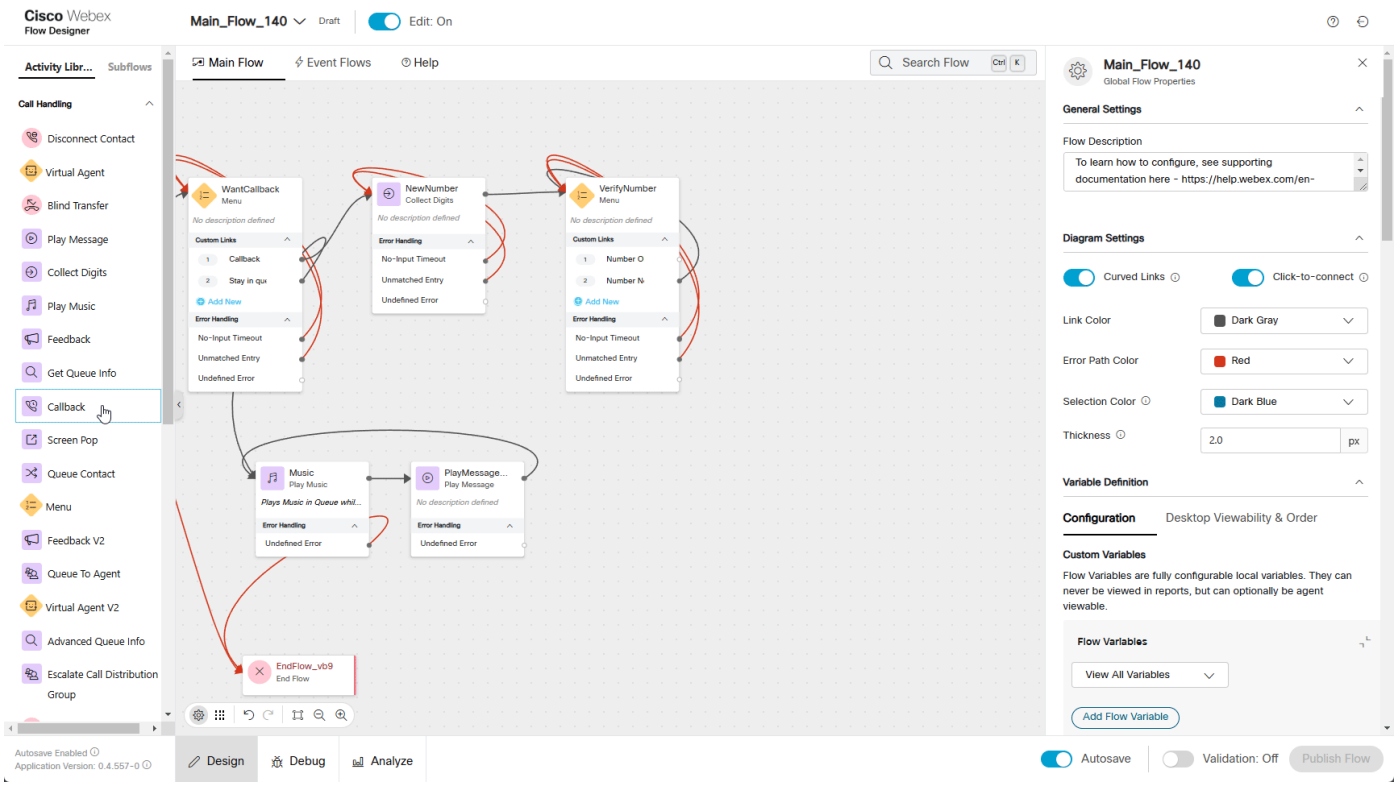
   Select the Connector: **Cisco Cloud Text-to-Speech**

   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 **Play Message** node

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




8. Validate the flow by clicking **Validate**, **Publish** and select the Latest version of the 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 and if success you should hear configured messages.
3. When callback is proposed, press 1 on Webex App DialPad 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 Support contact number **1 408 526 7209** as a callback number. Use the DialPad to provide the Cisco TAC number, then confirm when asked.
5. Once done, another message about successful scheduling should play.
6. Make your agent **Available**. Contact Center will reserve you right away and propose to answer a callback call.

**Congratulations on completing another 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.

For more information of how you can use MockAPI please watch these Vidcasts: **[ADVANCED] Use MockAPI to enhance your Demos - PART 1** and **[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: **empty**

- Custom ANI variable:

  Name: **customani**

  Type: **String**

  Default Value: **empty**

- HTTP GET Result variable:

  Name: **getresult**

  Type: **String**

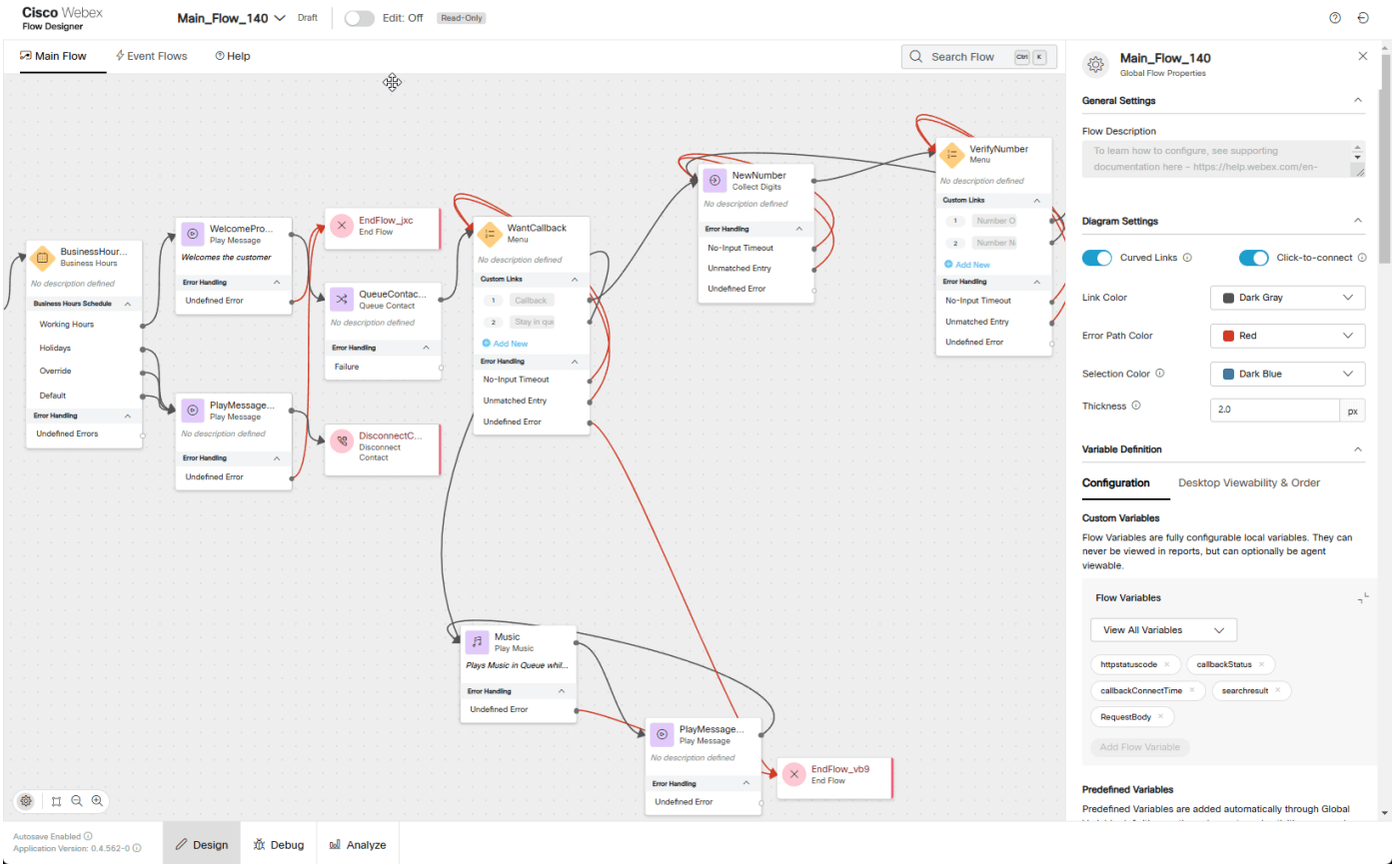
  Default Value: **empty**

- Simulated Error variable:

  Name: **simulatederror**

  Type: **String**

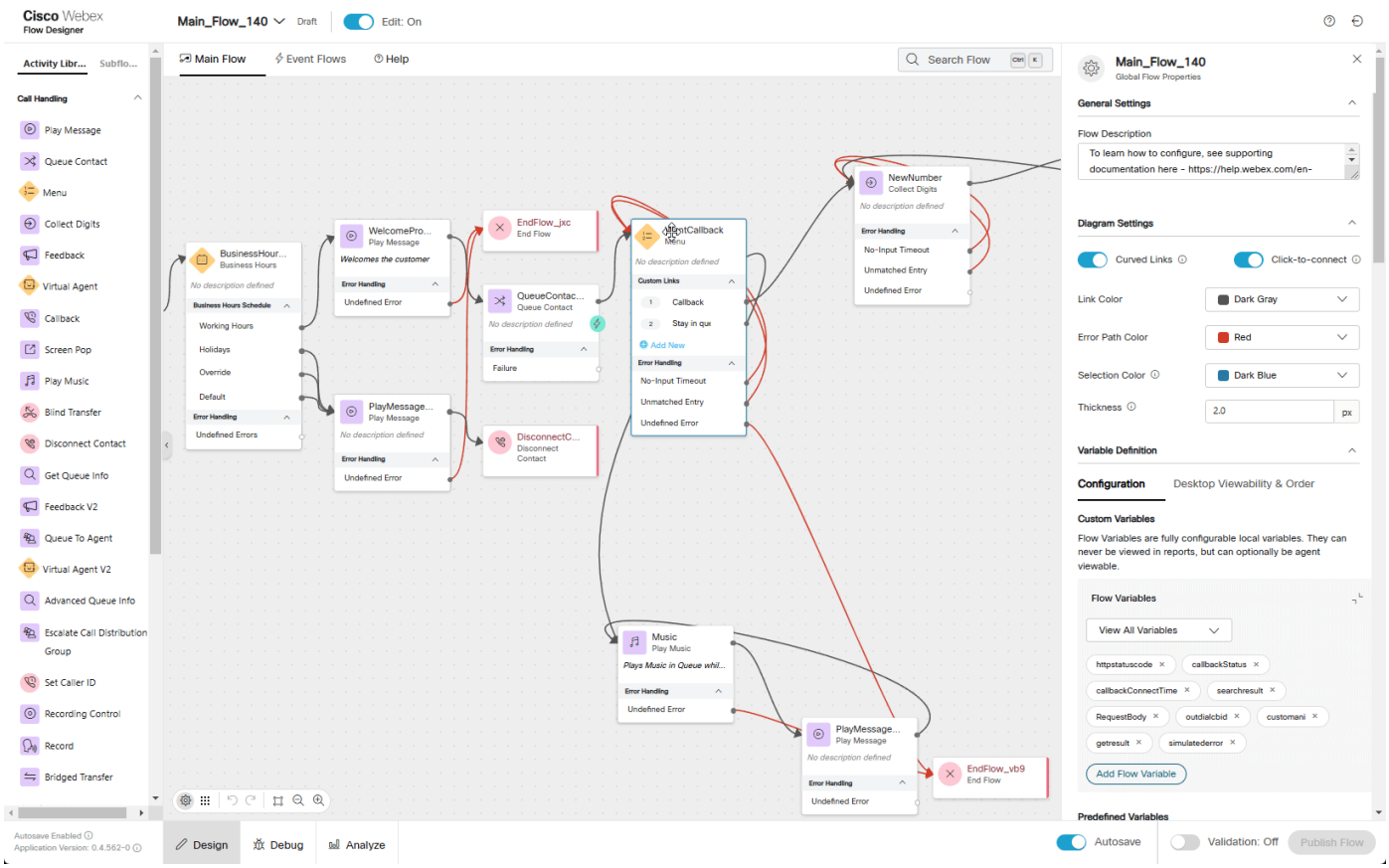
  Default Value: **empty**




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 Support contact number in Mission 3 of Fundamental labs.

   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**

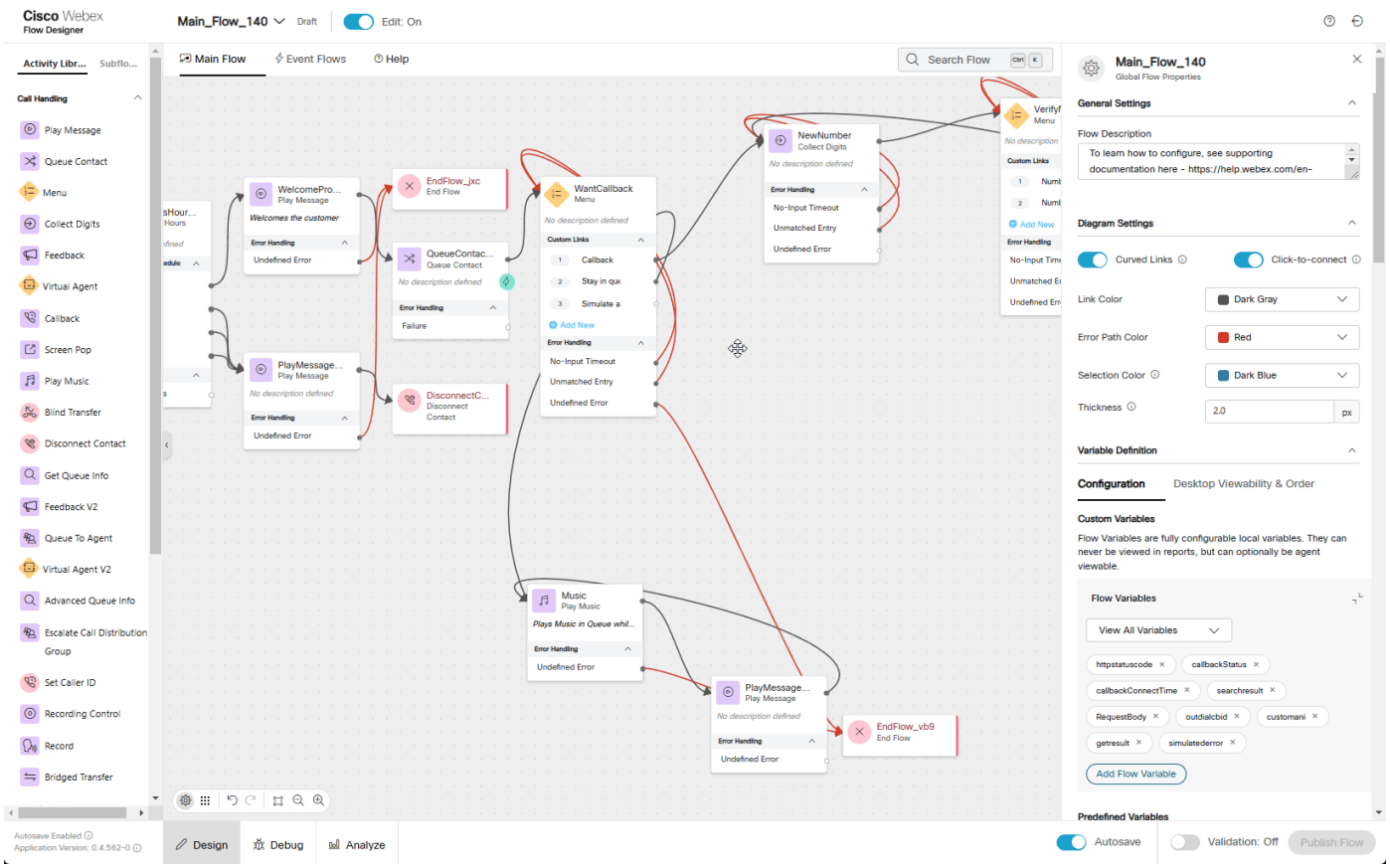
   Output Variable: **outdialcbid**

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

   Click **Add New**

   Output Variable: **customani**

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



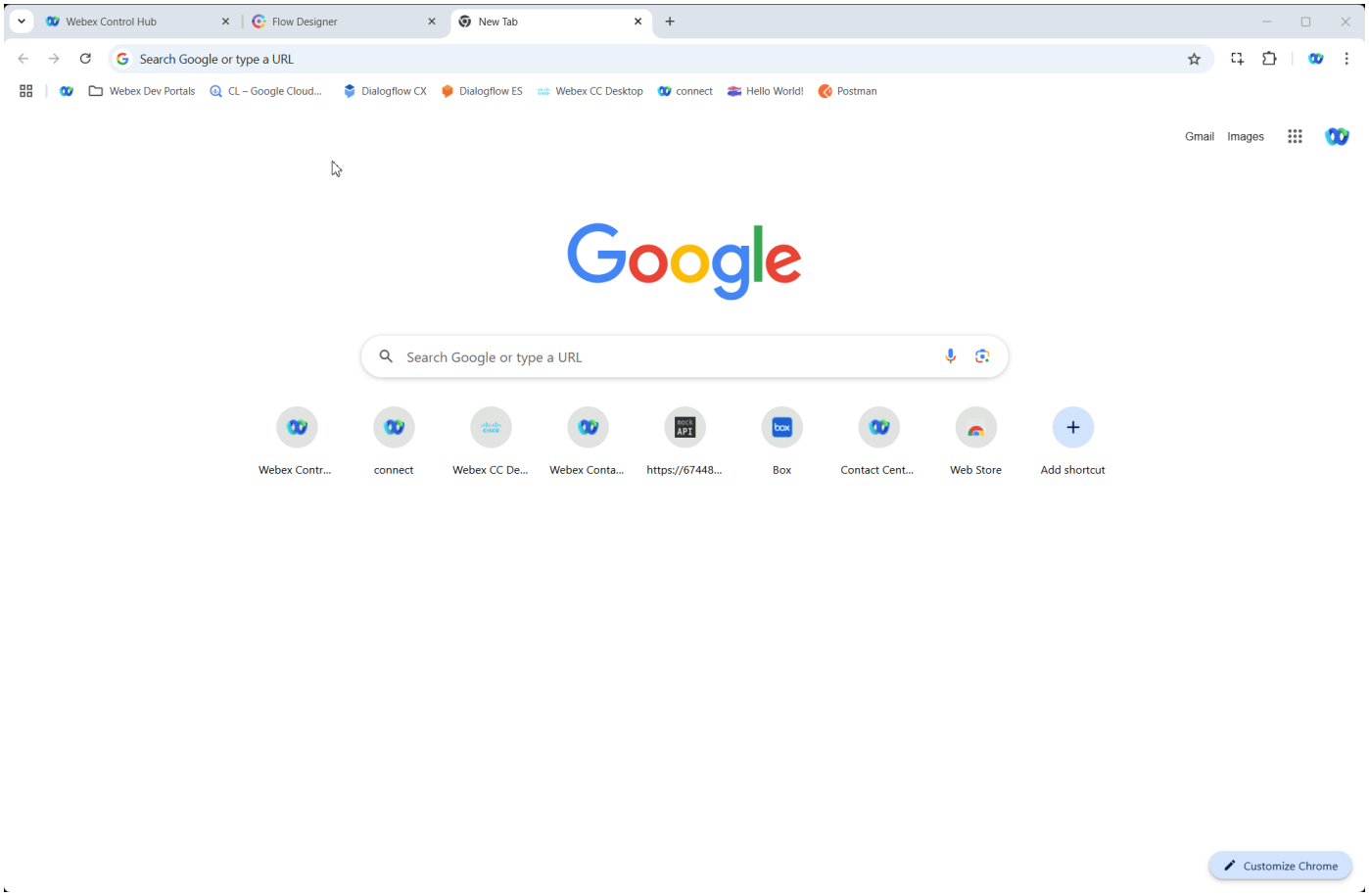
Test your API Source**[Optional]**

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

b. Replace {DNIS} with the provided DNIS number stripping +1

[Example:] If your number **+14694096861**, then your GET Query should be ***https://674481b1b4e2e04abea27c6e.mockapi.io/flowdesigner/Lab/DynVars?dn=4694096861***

a. Open Chrome browser and past your URL. You should get the follwoing result

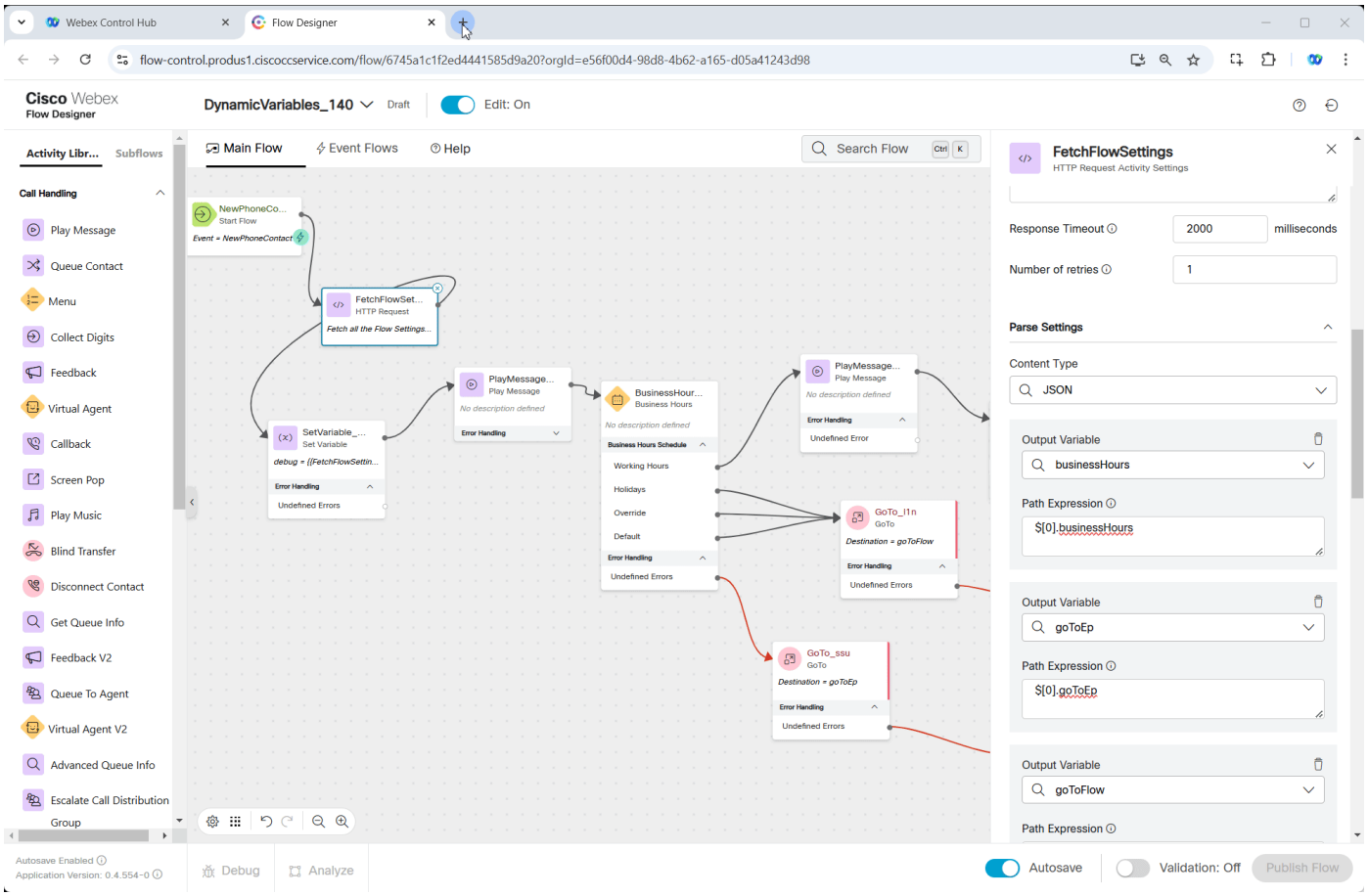



a. Test JSON Path in the following tool https://jsonpath.com/

b. Paste your GET URL into the Browser address line and copy the output in square brackets (including brackets)

c. Open https://jsonpath.com/ and paste the copied response into **Inputs** window

d. In **JSONPath** box copy and paste one of the path expression from **FetchFlowSettings** to verify your results.




5. Add **Set Variable** node

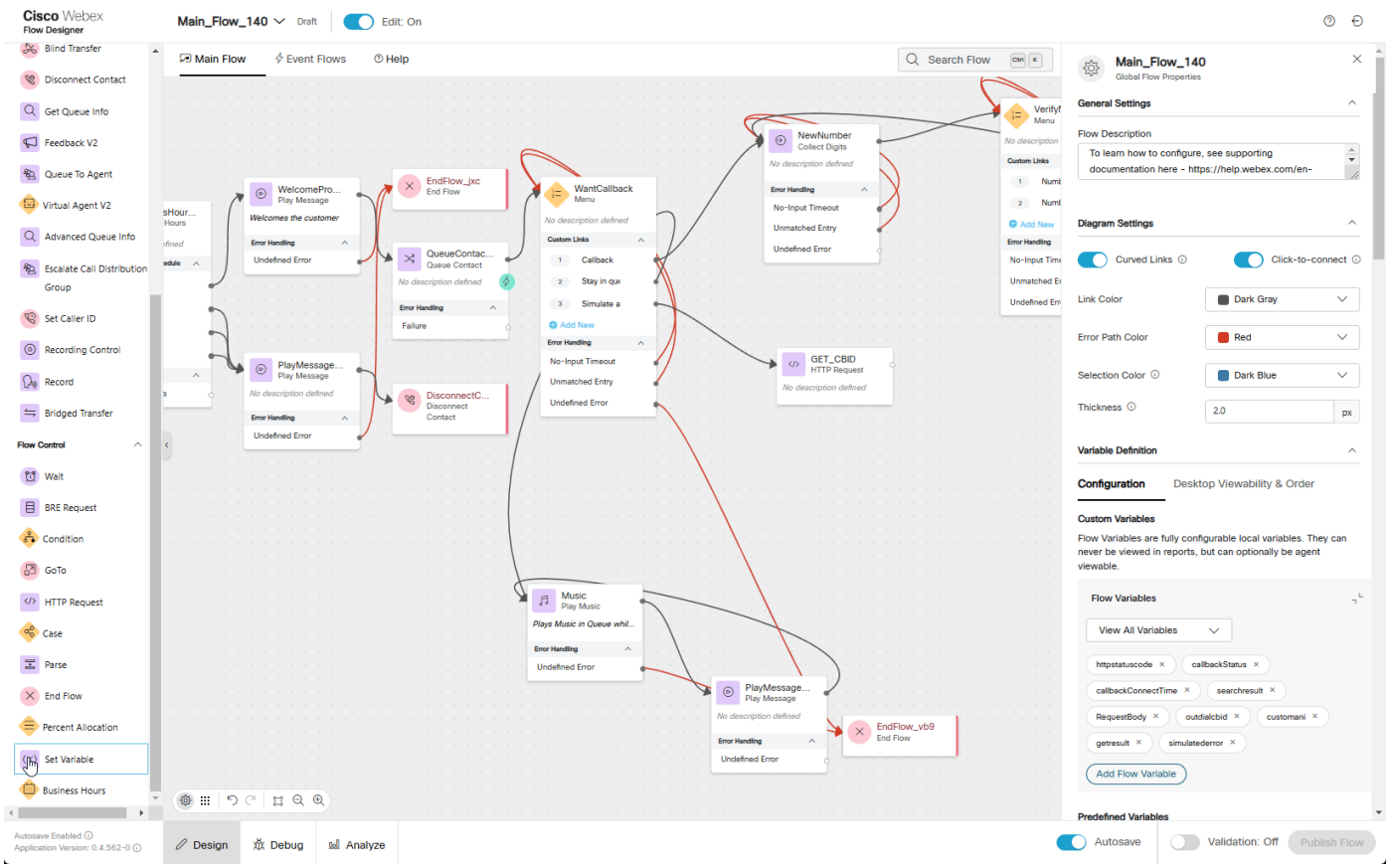
   Activity Label: **SetGetResult**

   Connect **GET_CBID** to this node

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

   Variable: **getresult**

   Set To 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.

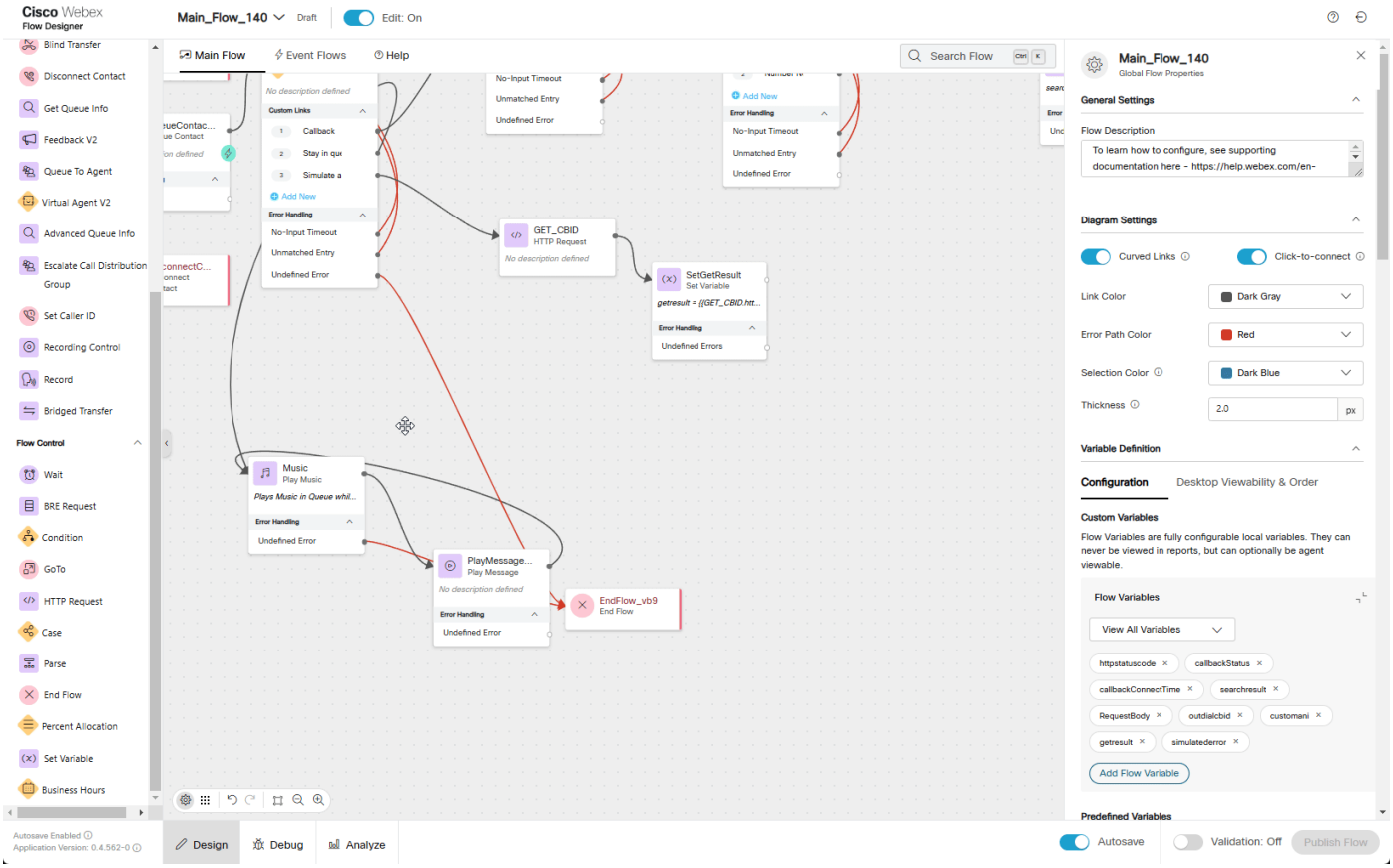
   Activity Label: **SimulateGlobalError**

   Connect **SetGetResult** to this node

   Connect this node to **Disconnect Contact**

   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**

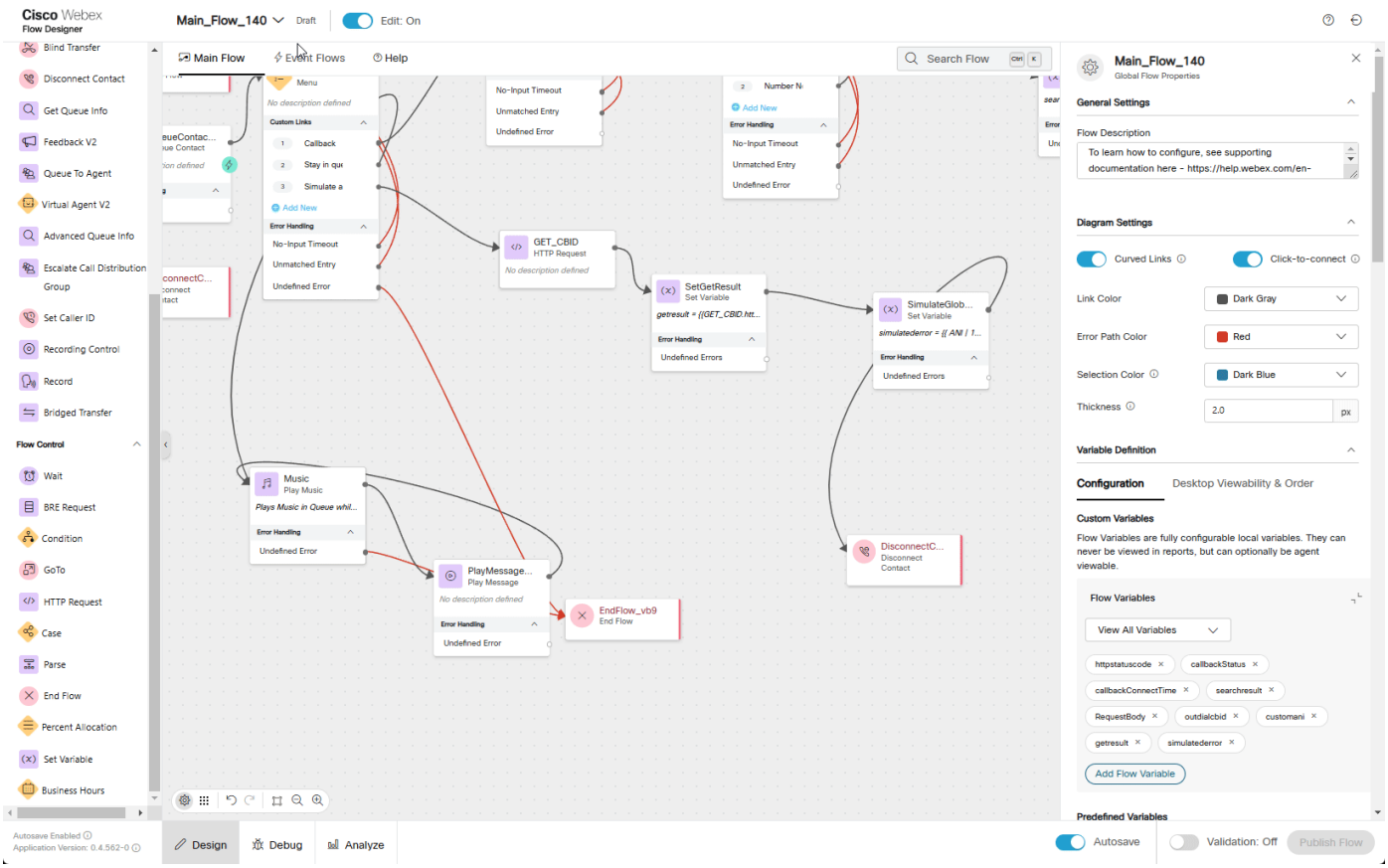
   Request Path: **/v1/tasks**

   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**.

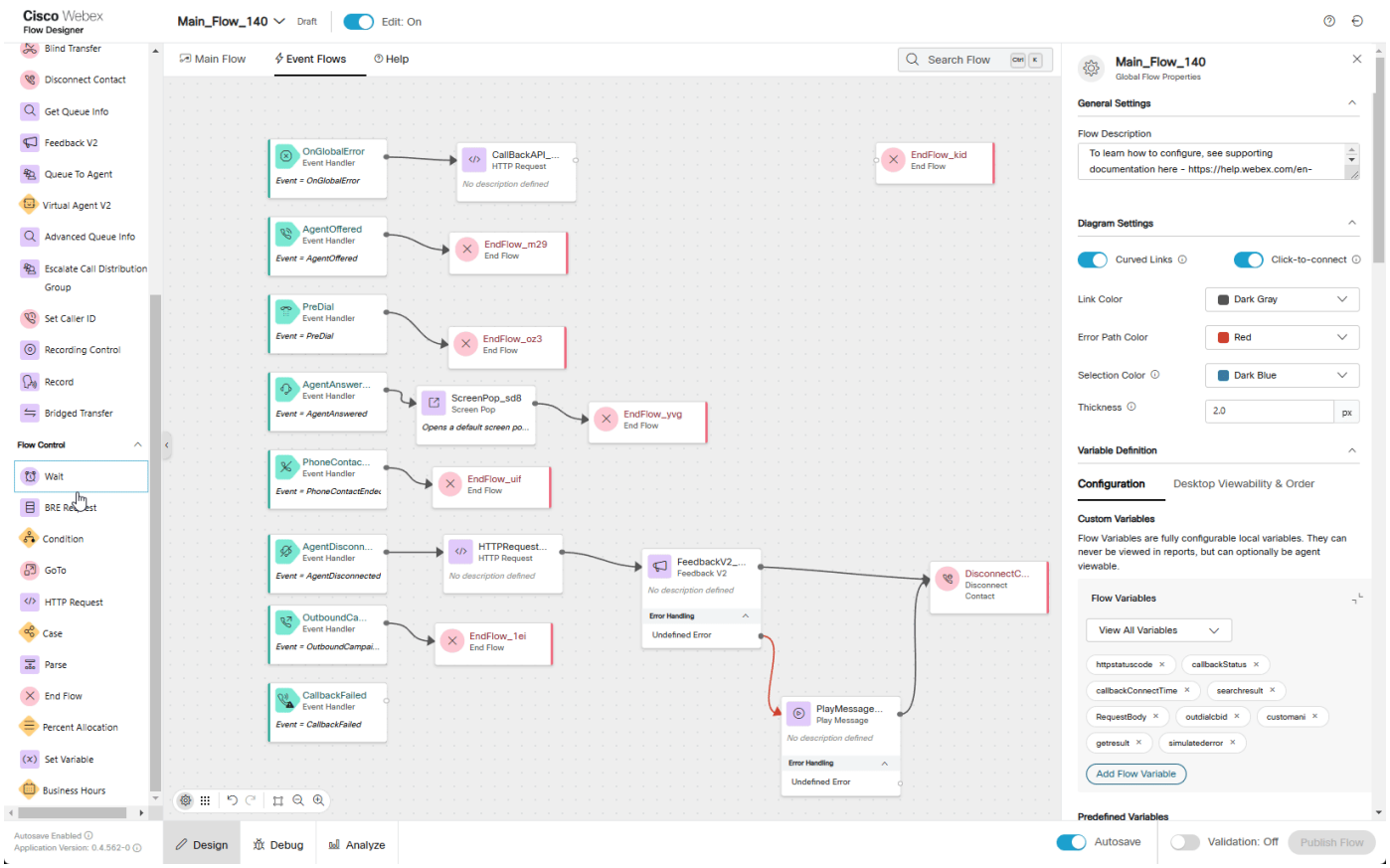
   Activity Label: **HTTPStatusCode**

   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 the flow by clicking **Validate**, **Publish** and select the Latest version of the 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**. 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 DialPad 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** n 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 succesfully simulated **Global Error** event.
9. Click on next activity name **GlobalErrorHandling** which goes after **SimulateGlobalError** activity name. Flow Designer automaticaly 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 succesfully.
11. On Webex Desktop, make your agent **Available**. Contact Center will reserve your agent right away and propose to answer a callback call.

**Congratulations on completing another mission.**

## 4.1.4 Mission 4: Preventing Callback duplication

> **Note**
>
> This task relies on completing Mission 3 of Fundamental Labs. 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 calback 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: **empty**

- Callback Connect Time variable:

  Name: **callbackConnectTime**

  Type: **String**

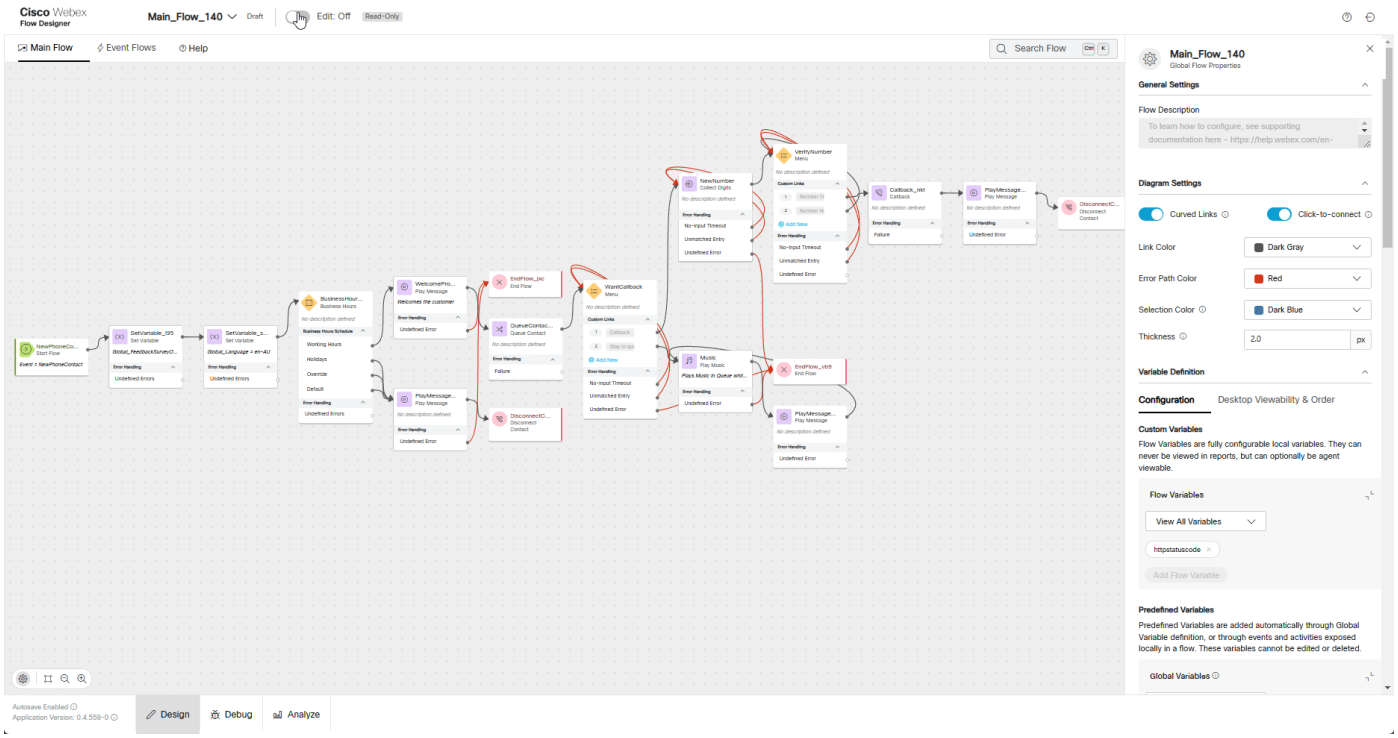
  Default Value: **empty**

- Search Result variable:

  Name: **searchresult**

  Type: **String**

  Default Value: **empty**



3. Add an **HTTP Request** node for our query as shown in the following video.

   Remove the existing connection between **VerifyNumber** Option 1 and **Callback** node

   Connect **VerifyNumber** Option 1 to this HTTP 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: **Application/JSON**

   Copy this GraphQL query into the request body:

```
{"query":"query($from: Long!, $to: Long!)\n{\n  taskDetails(\n      from: $from\n      to: $to\n    filter: {\n      and: [\n       { callbackData: { equals:
{ callbackNumber: \"{{NewNumber.DigitsEntered}}\" } } }\n       { lastEntryPoint: { id: { equals: \"{{NewPhoneContact.EntryPointId}}\" } } }\n      ]\n    }
\n  ) {\n    tasks {\n      callbackData {\n        callbackRequestTime\n        callbackConnectTime\n        callbackNumber\n        callbackStatus\n
callbackOrigin\n        callbackType\n      }\n       lastEntryPoint {\n        id\n        name\n      }\n    }\n  }\n}","variables":{"from":"{{now() |
epoch(inMillis=true) - 15000000}}","to":"{{now() | epoch(inMillis=true)}}"}}
```

**Expanded Query For Understanding (optional)**

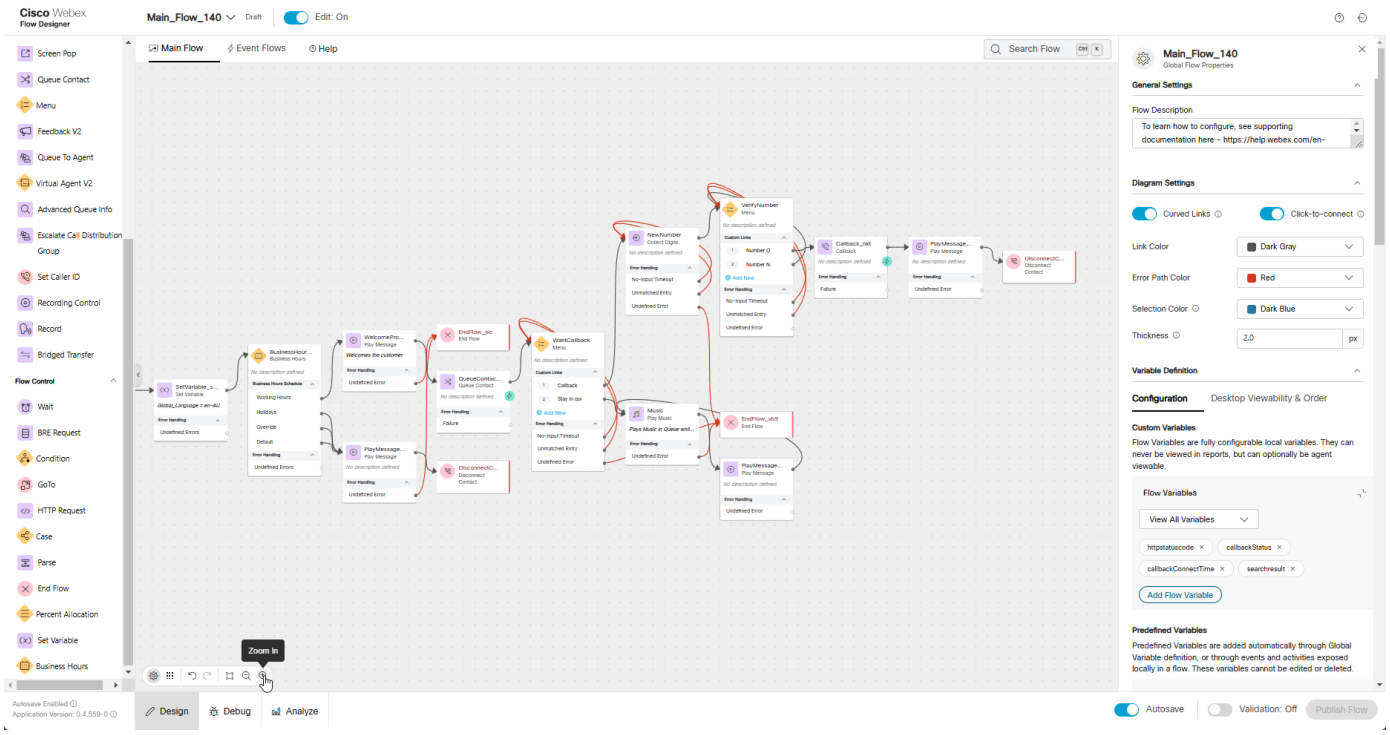
```
query($from: Long!, $to: Long!)
{
  taskDetails(
      from: $from
      to: $to
    filter: {
      and: [
       { callbackData: { equals: { callbackNumber: "{{NewNumber.DigitsEntered}}" } } }
       { lastEntryPoint: { id: { equals: "{{NewPhoneContact.EntryPointId}}" } } }
      ]
    }
  ) {
    tasks {
      callbackData {
        callbackRequestTime
        callbackConnectTime
        callbackNumber
        callbackStatus
        callbackOrigin
       callbackType
      }
       lastEntryPoint {
        id
        name
      }
    }
  }
}
```

   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`



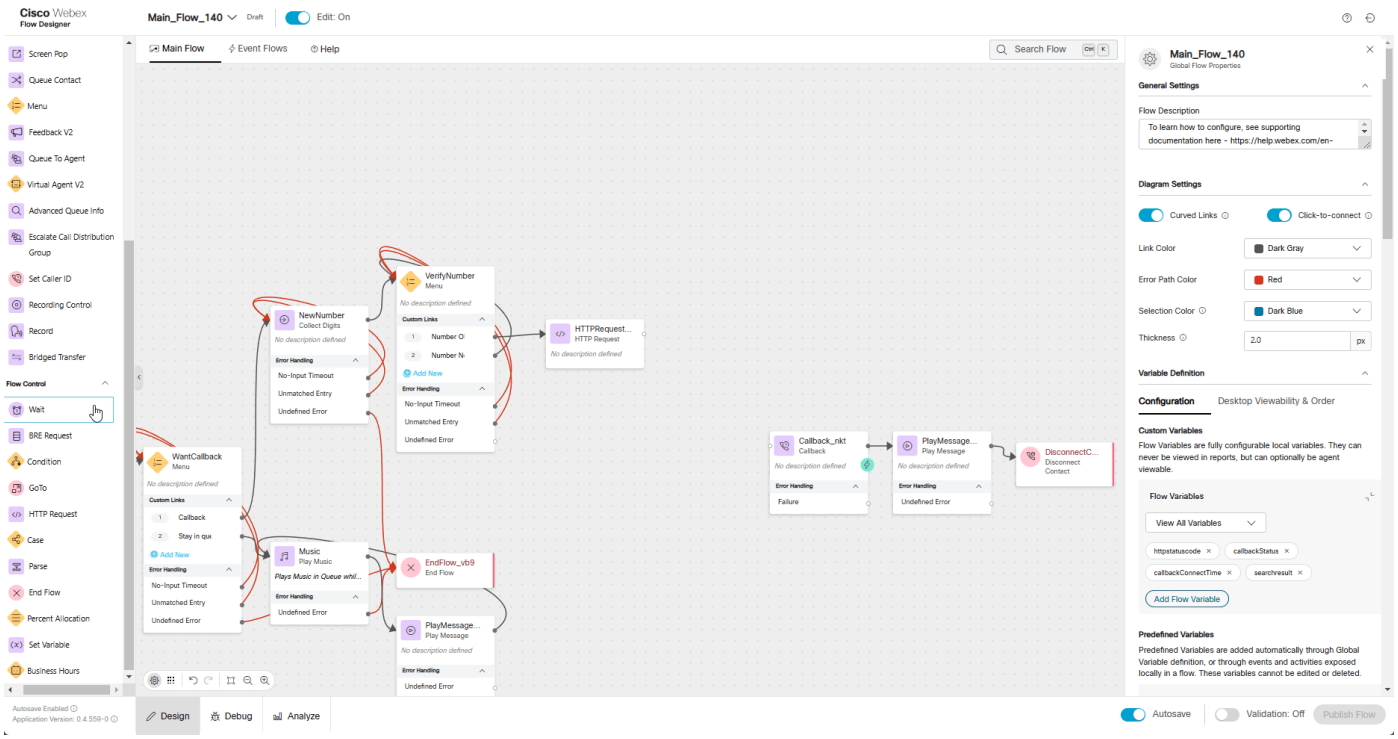

4. Add **Set Veriable** node

   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

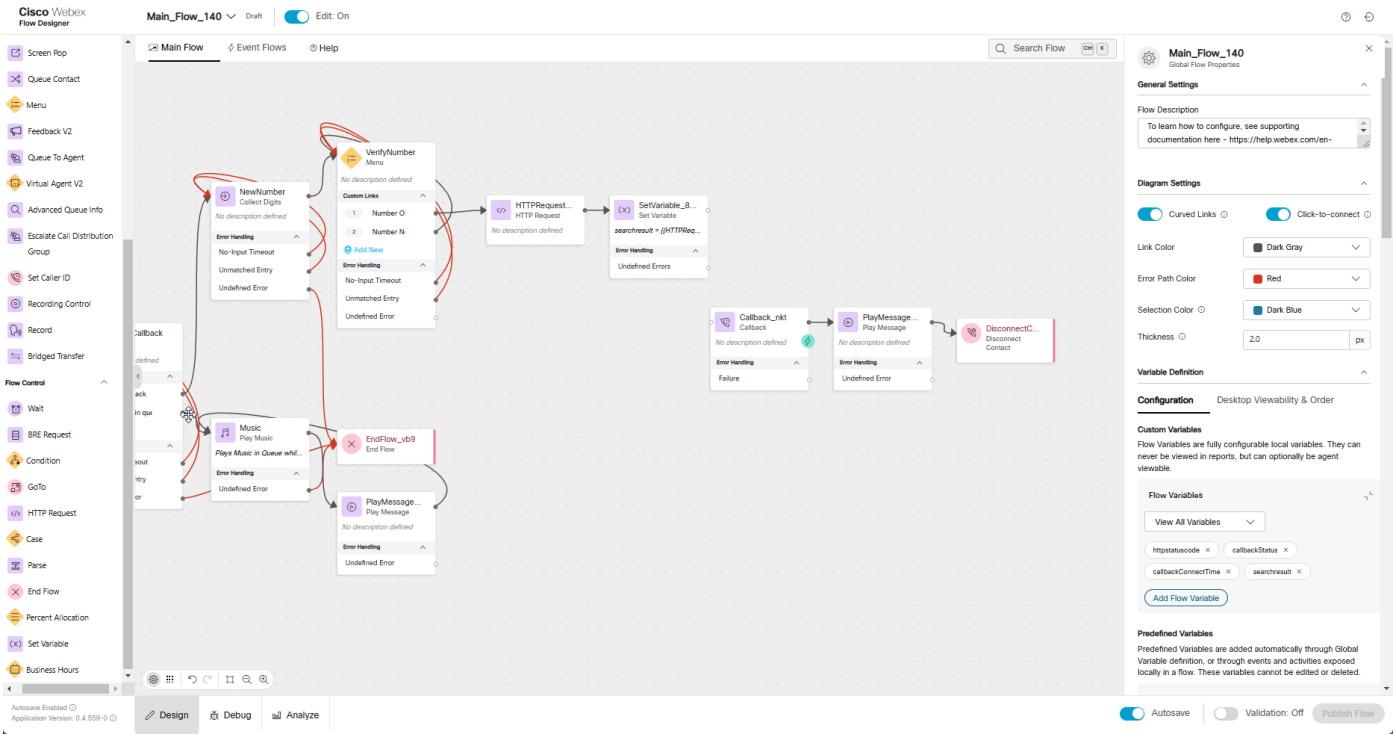
   Connect **False** exit path to existing CallBack node

   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** and **Disconnect Contact** nodes:

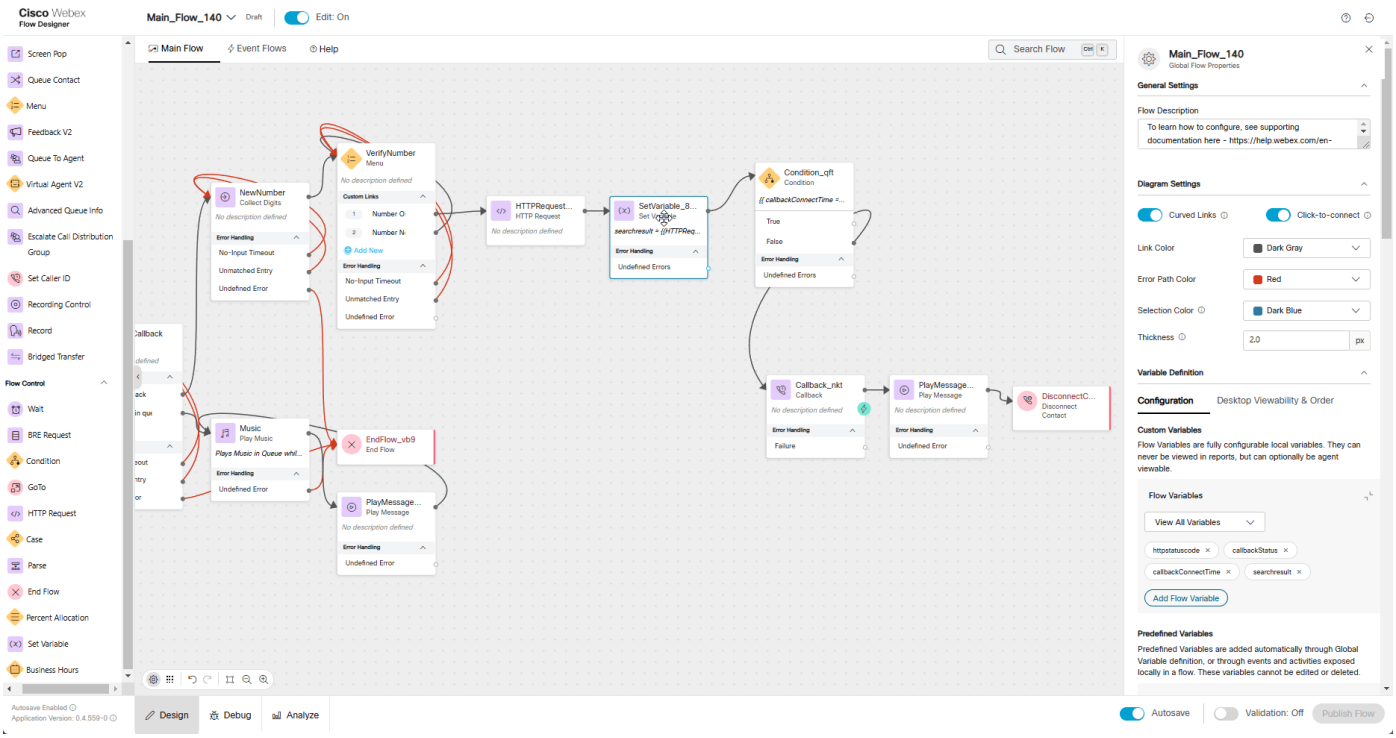
   Enable Text-To-Speech

   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

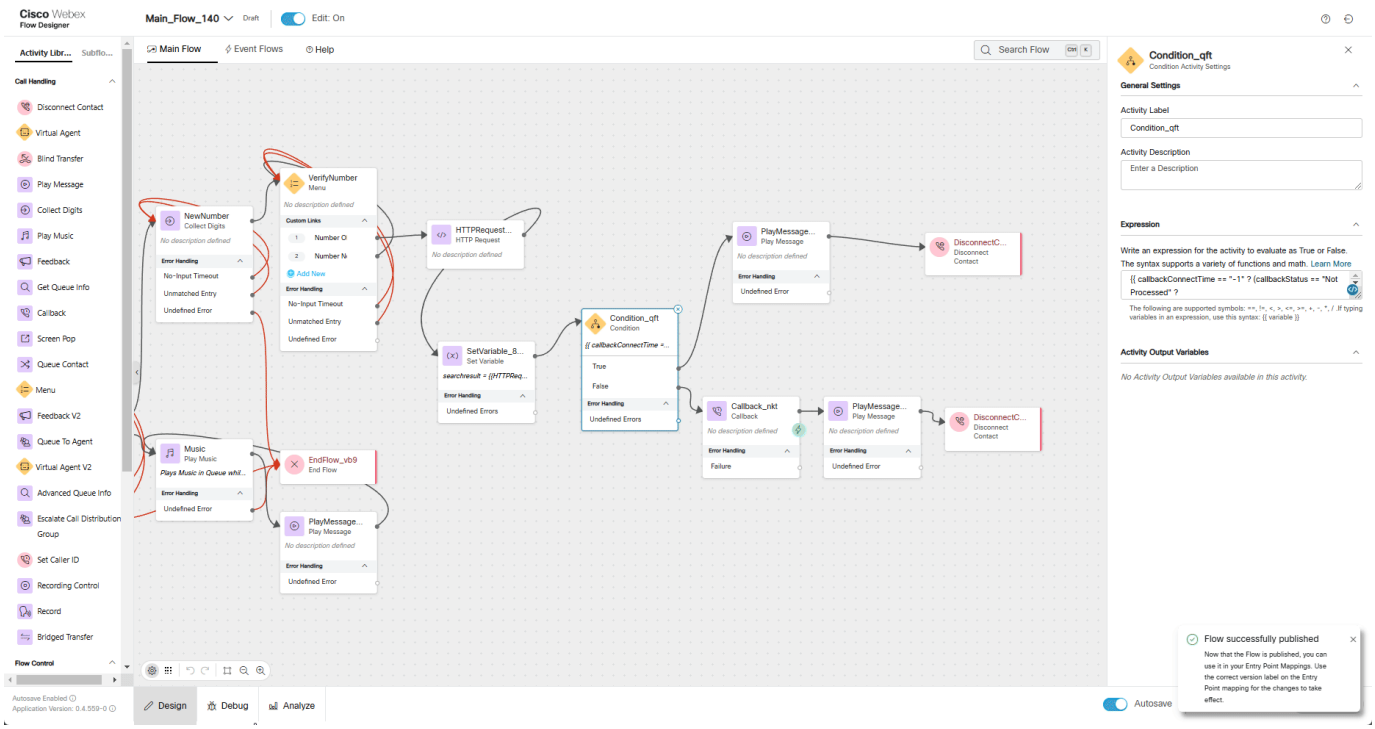
   Connect **True** exit path of **Condition** node created in previous step to **Play Message** node Connect this **Play Message** to **Disconnect Contact** node




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

### Testing

1. Make sure your Agent either **Logged Out** or in **Not Available** state. In this case call will not be assigned to an agent and callback will be proposed to a caller.
2. Make sure your **Main_Flow_Your_Attendee_ID** is assigned to **Your_Attendee_ID_Channel**. If not, do that (refer to previous very first Mission where this step was explained in details).
3. Make a call to your Support Number and if success you should hear configured messages and ask to provide a new number for a callback. Because in current lab we are having number limitations we are going to provide a wellknown Cisco Worldwide Support contact number **1 408 526 7209**
4. While keeping your agent **Not Available**, make another test call to your flow and request for a callback to the same number **1 408 526 7209**.
5. You should hear a message configured in **Step 6** of the current mission.
6. Click on **Analyze** to visualy observe the call flow. Make sure you're viewing latest Published Version.
7. Review the flow and click on **HTTPRequest_CallBackSearch** where you can cross-launch debuger to that particalar call.
8. Navigate to **HTTPRequest_CallBackSearch** to see **Modified Variables** at the bottom of right hand side of the debuger.
9. 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 cusrrent mission configuration.




**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. AI AGENT TRACK

## 5.1 Lab Guide

### 5.1.1 Mission 7: AI Autonomous Agent in Action (BONUS)

**Disclaimer**

Please note that the AI Agent is scheduled for General Availability (GA) in Q1 of the 2025 calendar year (CY25). This exercise is designed to provide you with an early glimpse and understanding of the feature's capabilities. We appreciate your participation and value your feedback as we refine this offering ahead of its official release.

**od to Know [Optional]**

AI AUTONOMOUS AGENT OVERVIEW

The Autonomous AI Agent for performing actions can handle various tasks, including:

- Natural Language Processing (NLP)—Understand and respond to human language in a natural and conversational manner.
- Decision making—Make informed choices based on available information and predefined rules.
- Automation—Automate repetitive or time-consuming tasks.

**Story**

As a visitor to Amsterdam, you want to quickly find restaurants offering various international cuisines in the city and easily figure out how to get to them from your current location at the RAI.

CALL FLOW OVERVIEW

1. A new call enters the flow.
2. The caller asks about restaurants in Amsterdam.
3. The AI agent responds with information generated from the knowledge base configuration.

**Mission overview**

Your mission is to:

1. Create a knowledge base (KB) and AI agent to provide answers about Amsterdam, including places to visit, restaurants, nightclubs, and directions from the current RAI Amsterdam Convention Center.
2. Configure the AI agent with handoff functionality to transfer the conversation to a live agent when necessary.

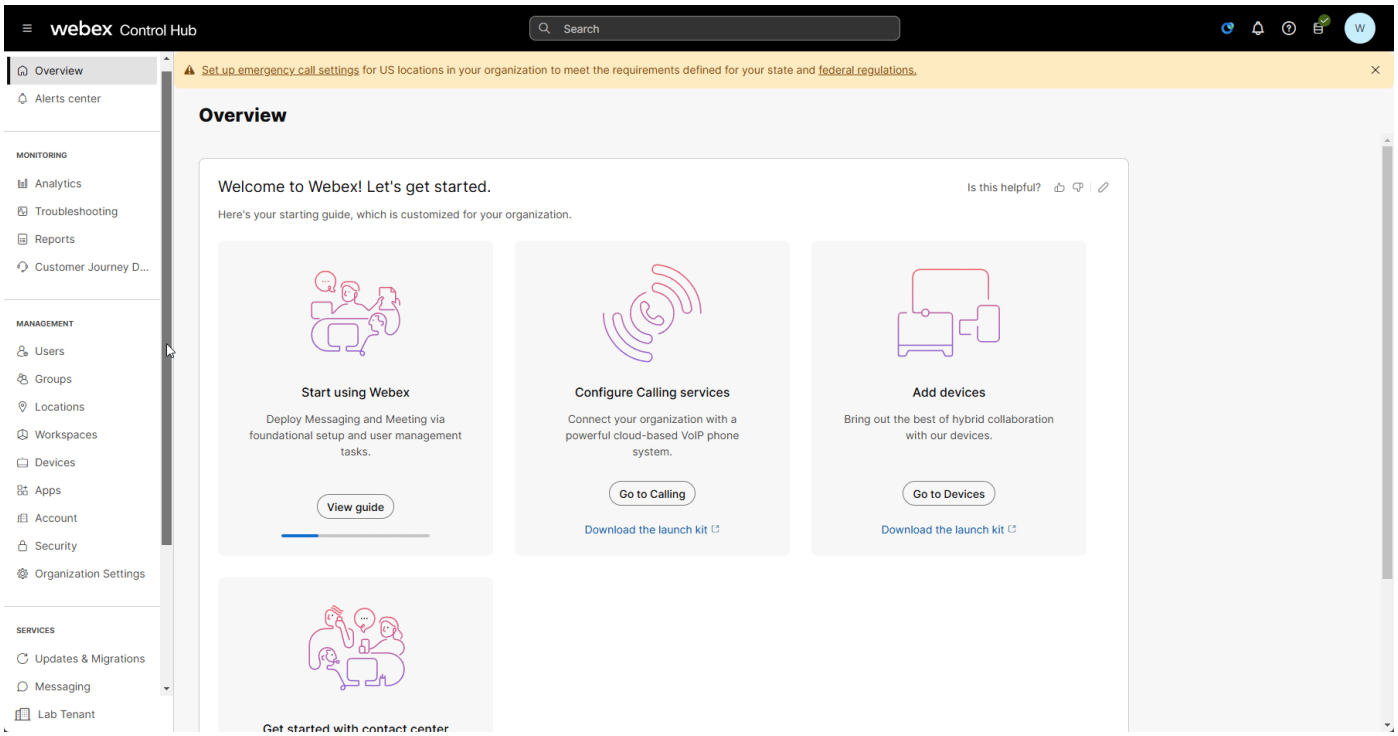
---

**Build**

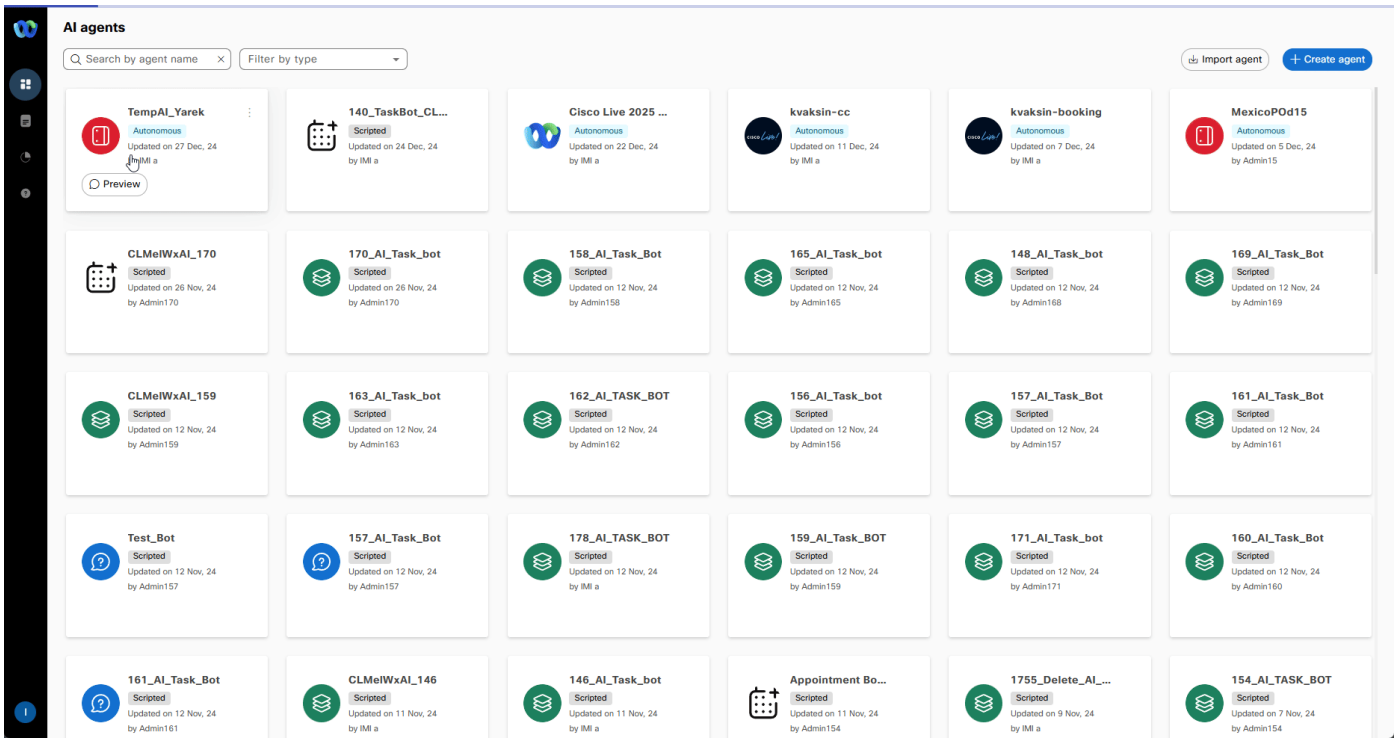
CREATING A KNOWLDGE BASE

1. [IMPORTANT] Download source file from shared folder.

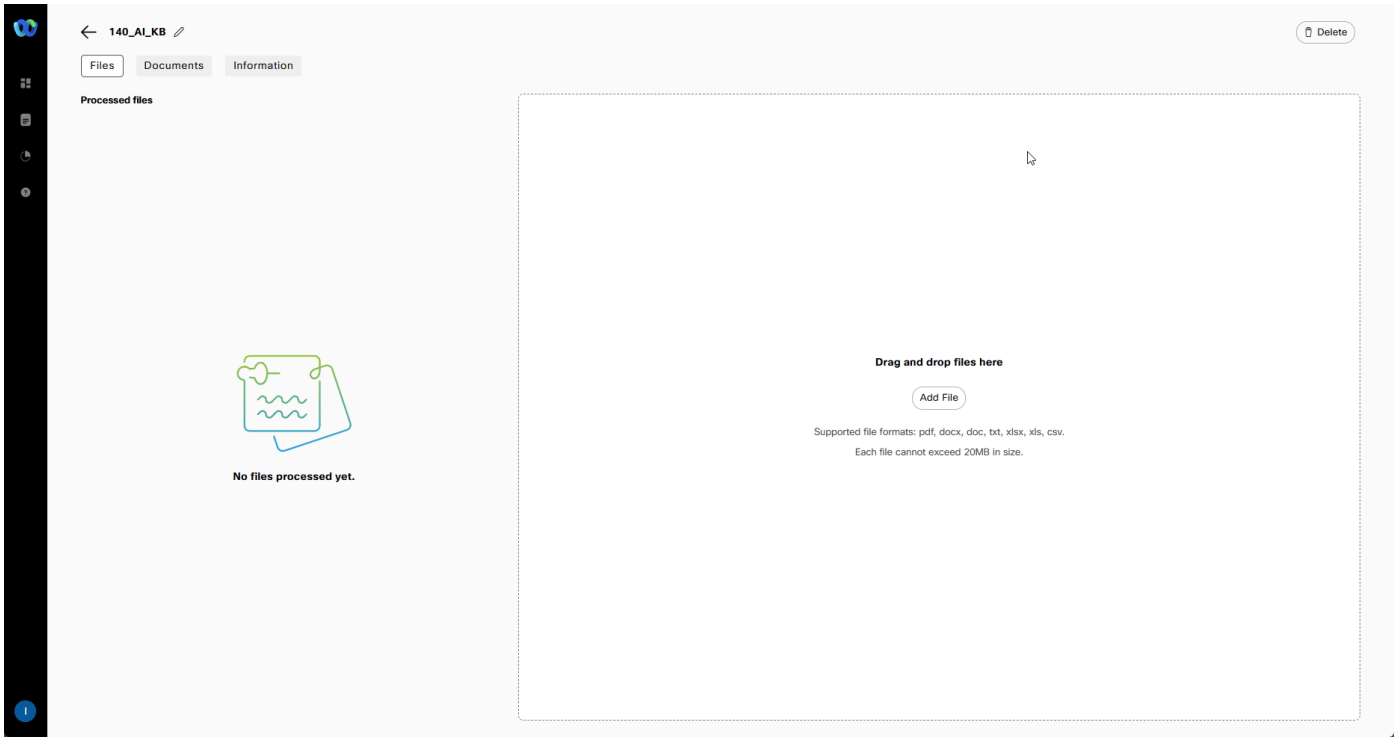
   **Amsterdam_Tourist_Guide.txt** - file contains information for tourists like places to visit, restaurants, pubs etc. and how to reach those places from RAI Amsterdam Convention Center
2. Login into Webex Control Hub by using your Admin profile **wxcclabs+admin_IDYour_Attendee_ID@gmail.com**. You will see another login screen with OKTA on it where you may need to enter the email address again and the password provided to you.
3. Go to Contact Center from the left side navigation panel, and under Quick Links, click on **Webex AI Agent**




4. In AI Agent Builder navigate to **Knowledge** from left hand side menu panel.
5. Click **Create Knowledge Base**, provide Knowledge base name as **Your_Attendee_ID_AI_KB** , then click **Create**.




6. Click **Add File** or drag and drop file **Amsterdam_Tourist_Guide.txt** you downloaded from external drive on **Step 1**. Then click **Process Files**.




7. Navigate to **Dashboard** from the right-hand side menu panel and click **Create Agent**
8. Select **Start from Scratch** and click **Next**
9. On **Create an AI agent** page select the following select the type of agent: **Autonomous**
10. New section **Add the essential details** will appear. Provide the following information, then click **Create**:

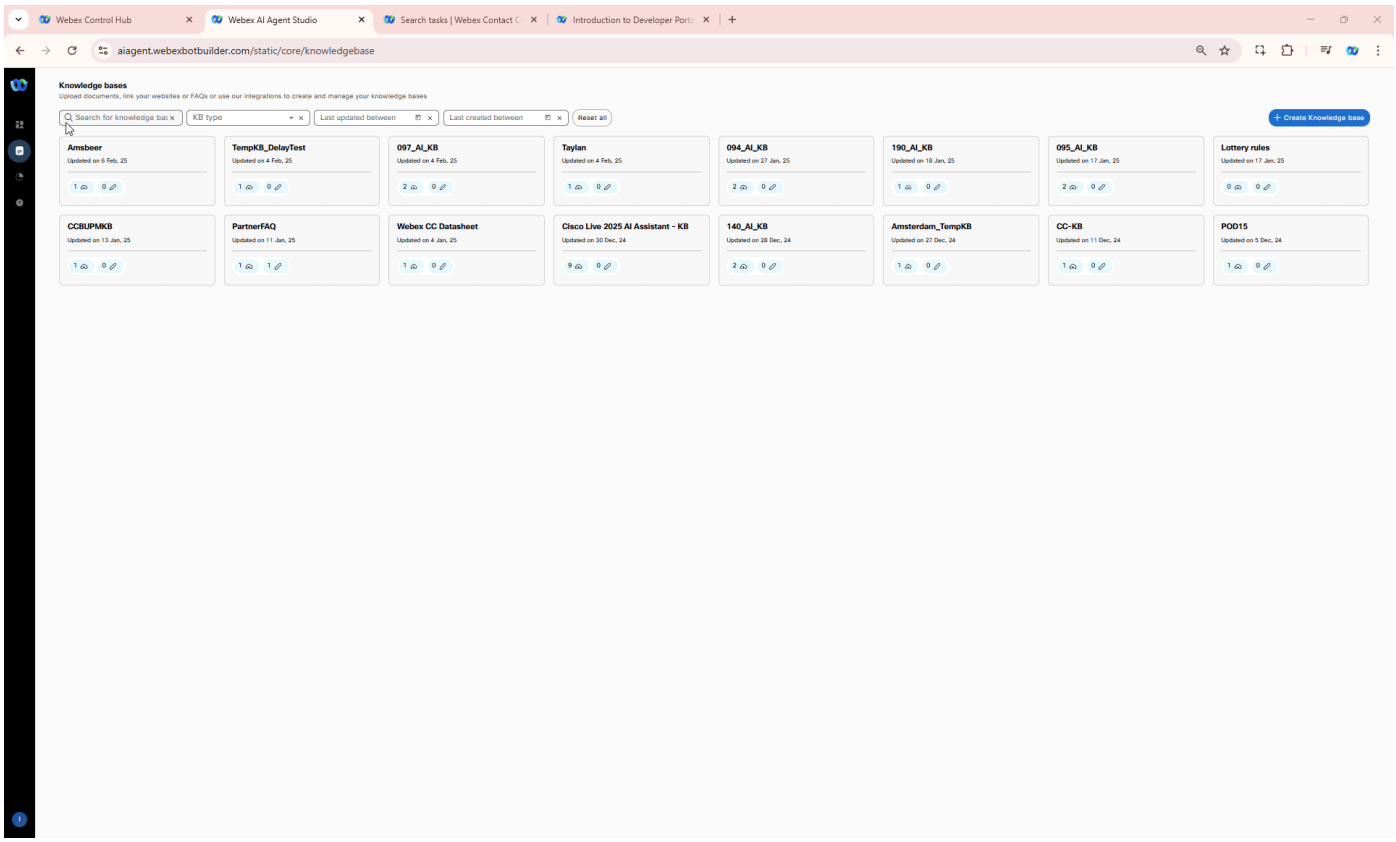
    Agent Name: **Your_Attendee_ID_AutoAI_Lab**

    System ID is created automatically

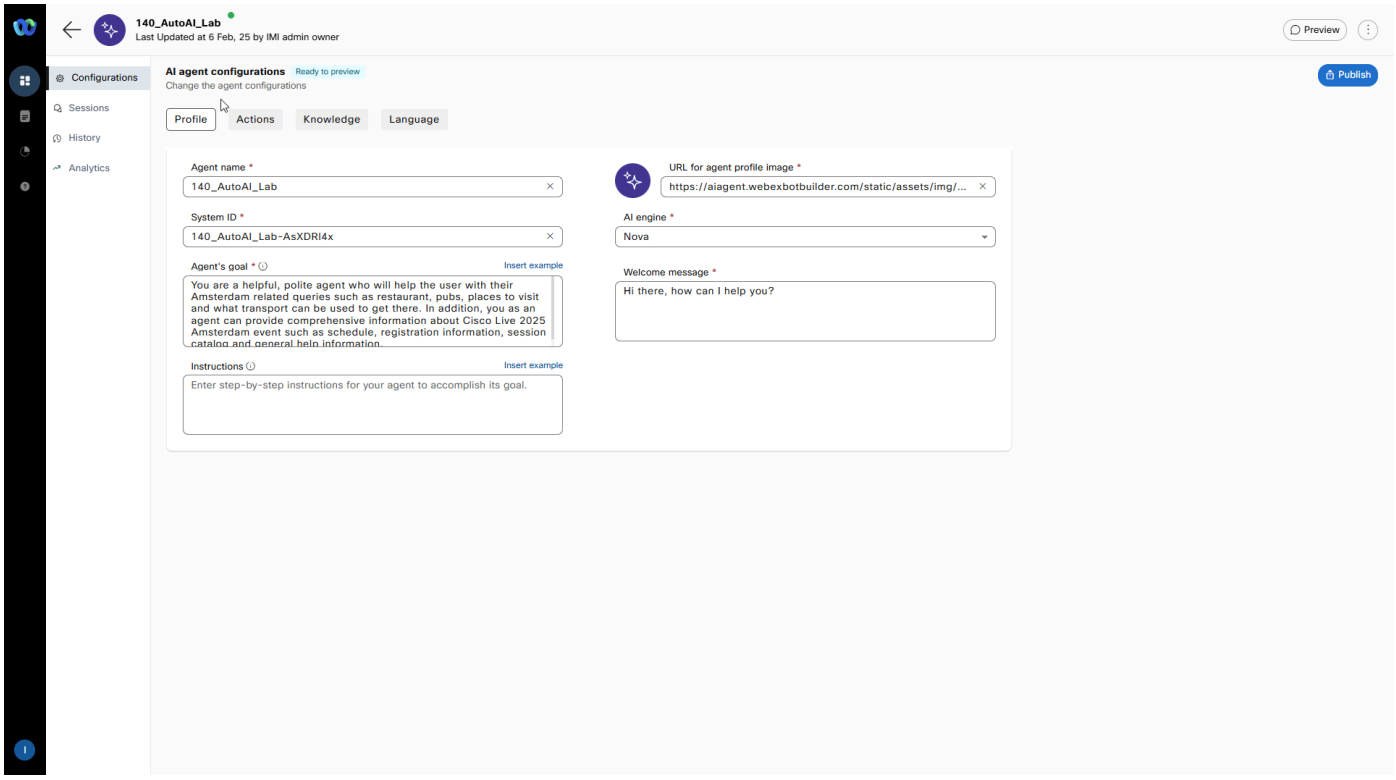
    AI engine: **Nova**

    Knowledge base: **Your_Attendee_ID_AI_KB**

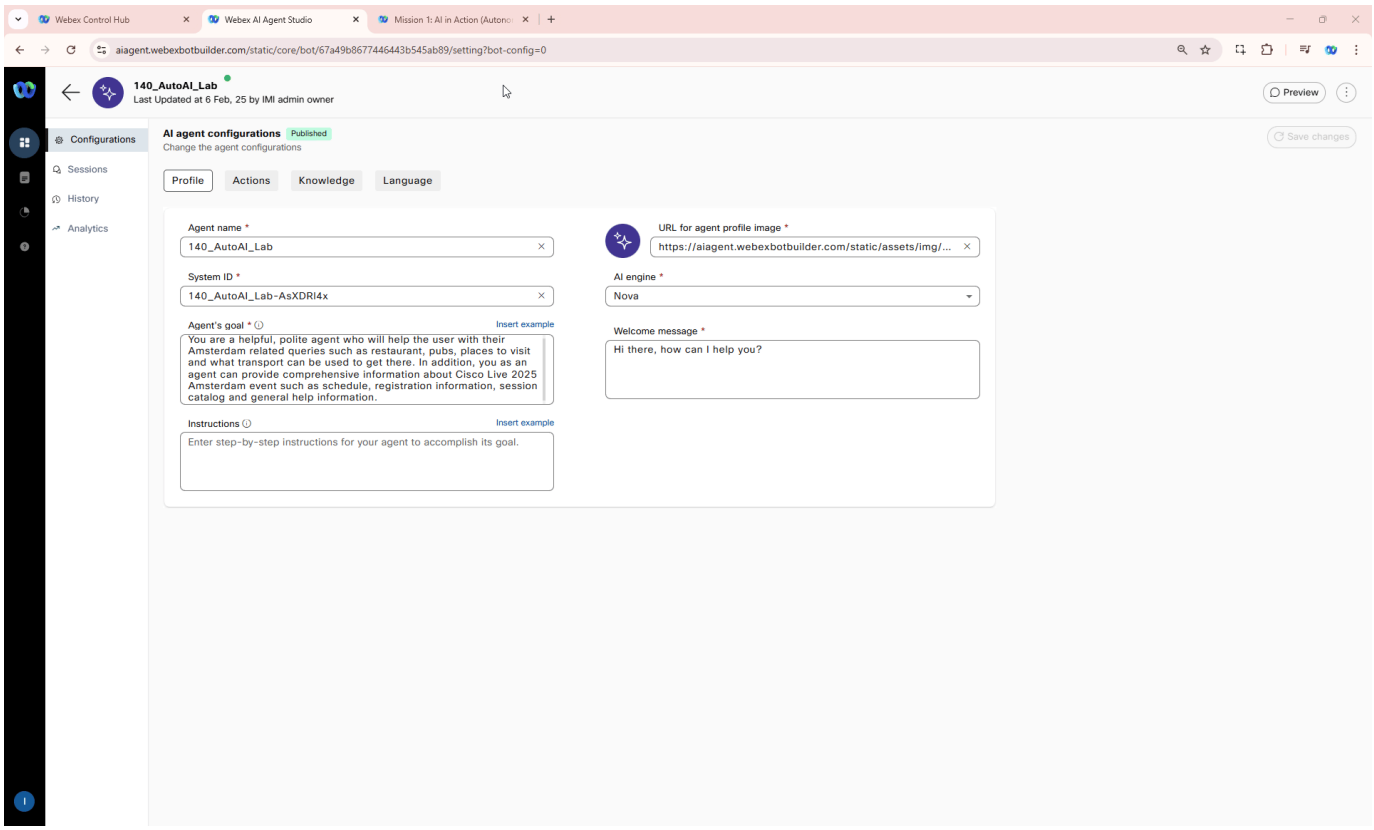
    Agent's goal: ***You are a helpful, polite agent who will help the user with their Amsterdam related queries such as restaurant, pubs, places to visit and what transport can be used to get there.***




11. Switch to **Actions** tab ans make sure **Agent handover** togle is turned on. This will allow you to handoff calls to human agent on request while talking to your Virtual Agent.
12. Switch to **Knowledge** tab and from **Knowledge base** drop-down list select **Your_Attendee_ID_AI_KB**
13. Click **Save Changes**, then click **Publish**. Provide any version name in popped up window (ex. "1.0").



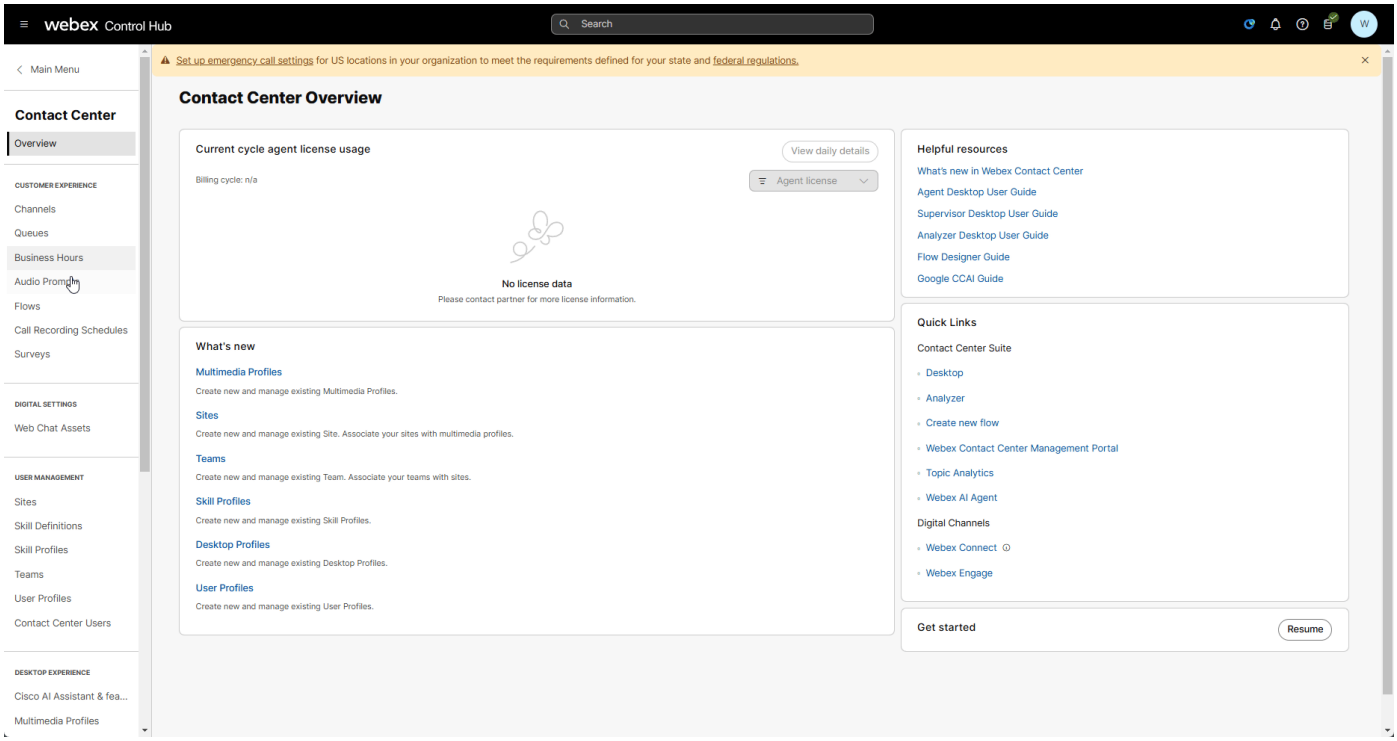

14. Click on **Preview** to test your AI Agent and ask the following: **"I'm looking for an Italian restaurant close to RAI."**




---

### Integrating the Bot with Flow for Voice Calls

1. In Control Hub navigate to **Flows**, click on **Manage Flows** dropdown list and select **Create Flows**
2. Select Start Fresh and name the new flow **AutonomousAI_Flow_Your_Attendee_ID**



3. Make sure the **Edit** mode at the top is set to **ON**. Then, drag and drop the **Virtual Agent V2** and **DisconnectContact** activities from the left panel onto the canvas.

> **Note**
>
> Please make sure to use **VirtualAgentV2** activity and **NOT VirtualAgent** also present on the Activity Library for Backward Compatability.

Connect the **New Phone Contact** output node edge to this **VirtualAgentV2** node

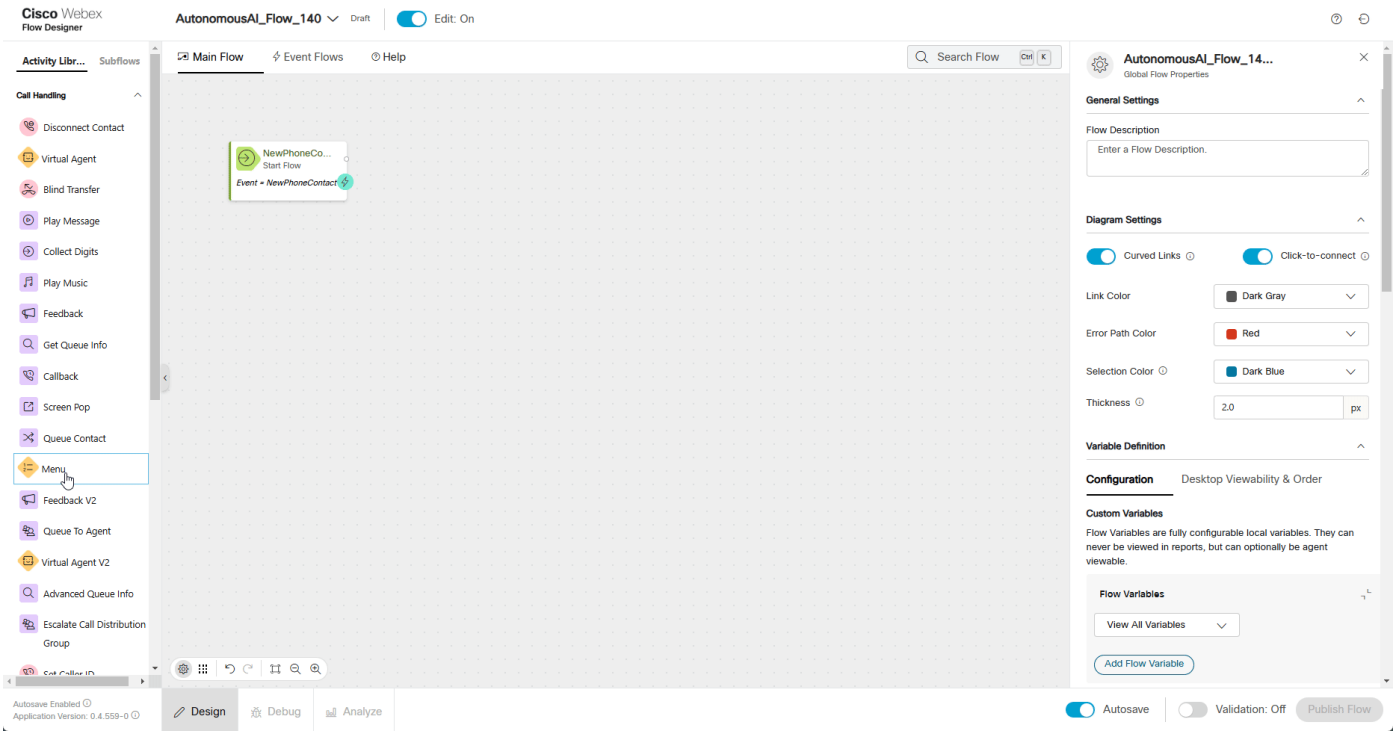
Connect the Handled outputs to **DisconnectContact**

Connect the Errored outputs to **DisconnectContact**

Set **Static Contact Center AI Config**

Contact Center AI Config: **Webex AI Agent (Autonomous)**

Virtual Agent: **Your_Attendee_ID_AutoAI_Lab**




4. Drag and drop following nodes:

- **Queue Contact** activity onto the Flow from the left side panel

  Connect the **Escalated** path from the **Virtual Agent V2** activity to the **Queue Contact** activity.

  Connect the **Queue Contact** activity to the **Play Music** activity

  Connect the **Failure** path from the **Queue Contact** activity to the **Disconnect Contact** activity.

  Select **Static Queue**

  Queue name: **Your_Attendee_ID_Queue**

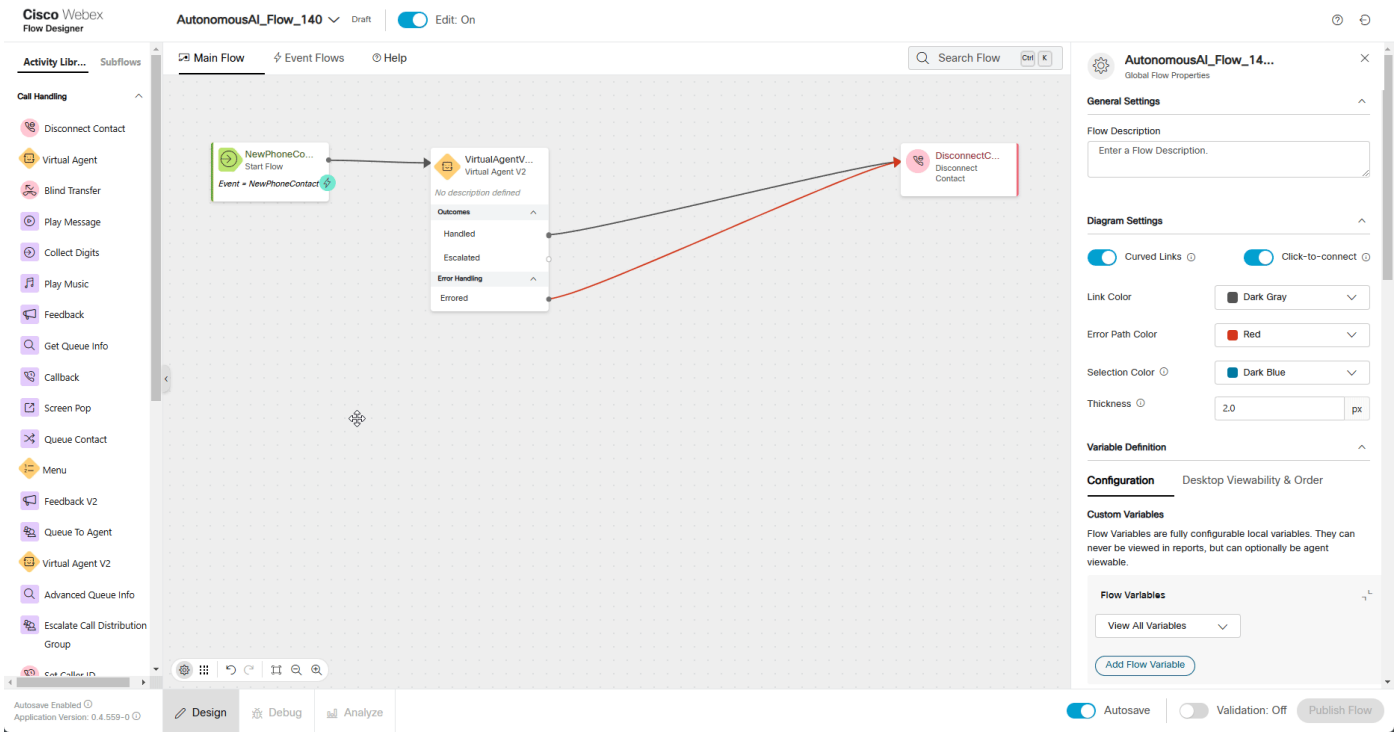
- **Play Music**

  Create a loop by connecting the Play Music activity back to itself - to create a music loop, following the diagram provided.

  Connect the **Failure** path from the **Play Music** activity to the **Disconnect Contact** activity.

  Music File: **defaultmusic_on_hold.wav**

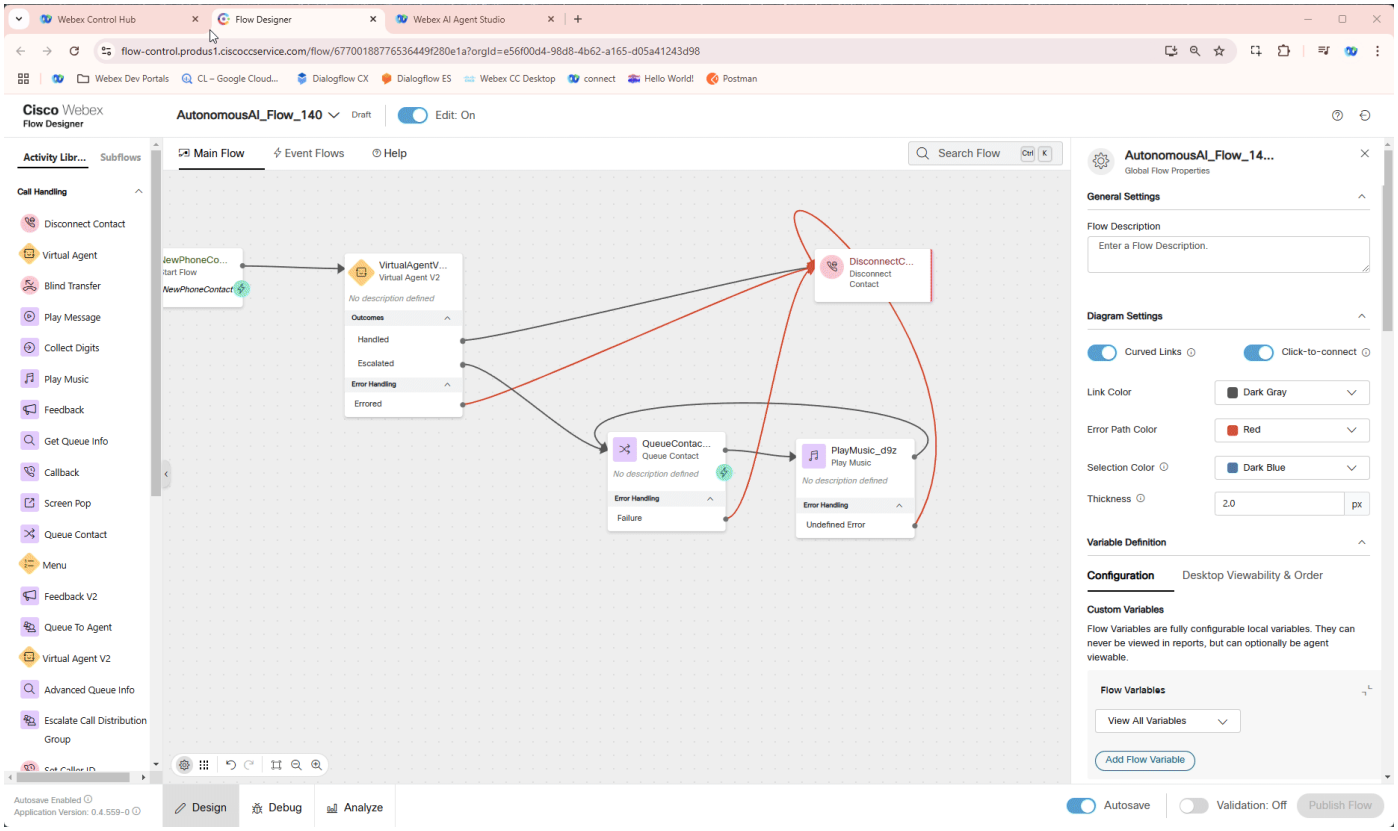
5. **Validate** and **Publish** Flow. In popped up window click on dropdown menu to select **Latest** label, then click **Publish**




6. Assign the Flow to your **Channel (Entry Point)** - Do this by first going to **Channel** > Search for your channel **Your_Attendee_ID_Channel**.
7. Click on **Your_Attendee_ID_Channel**
8. In **Entry Point** Settings section change the following:

   Routing Flow: **AutonomousAI_Flow_Your_Attendee_ID**

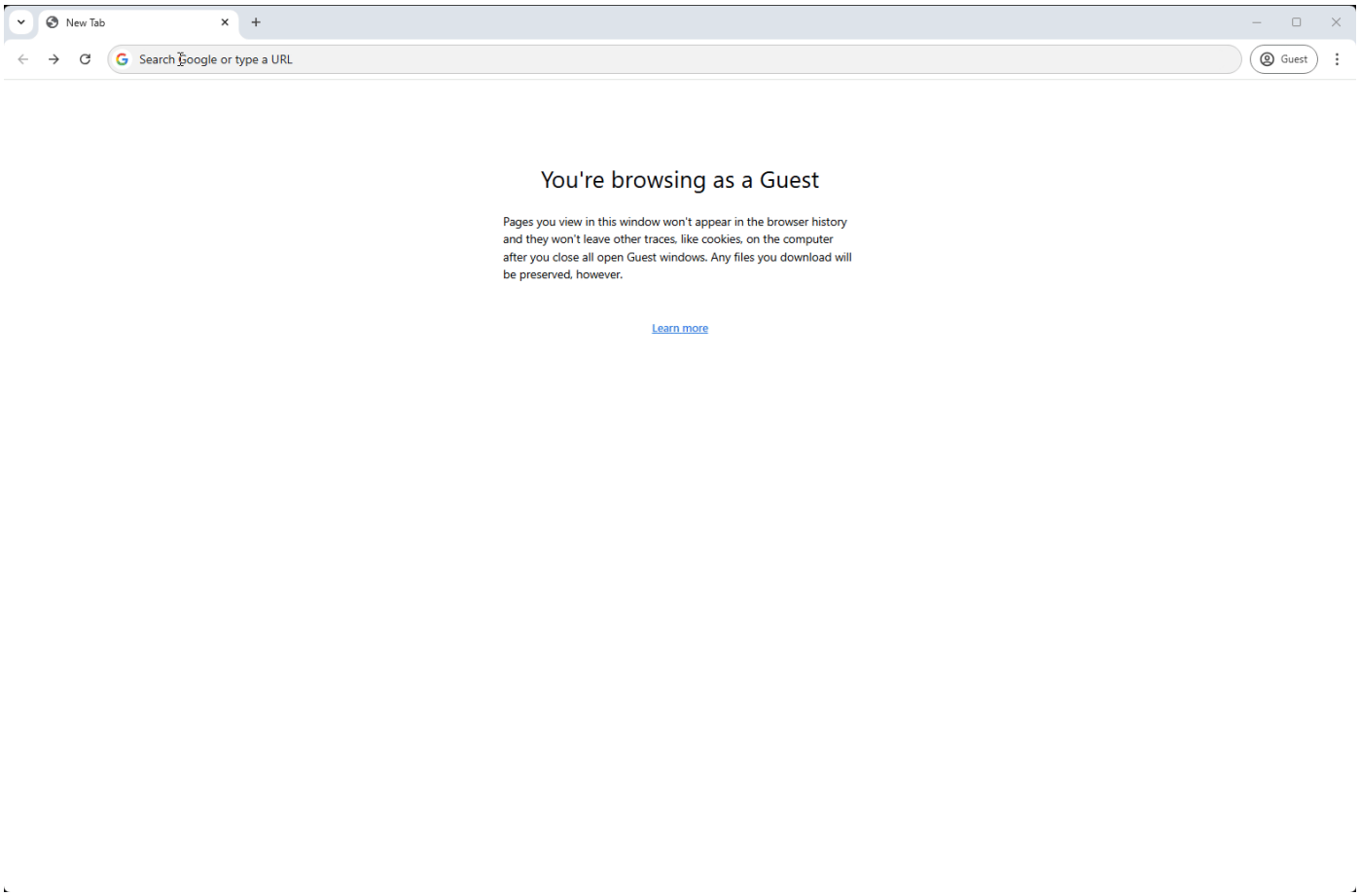
   Version Label: **Latest**



9. Dial Support Number assigned to your **Your_Attendee_ID_Channel** to test the Autonomous Virtual Agent over a voice call.

### 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 OKTA 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 every visit.
3. Make your agent ***Available*** and you're ready to make a call.




4. Dial the support number assigned to your **Your_Attendee_ID_Channel** channel, and during the conversation with the virtual agent, ask about restaurants in Amsterdam or places to visit to explore historical sites.
5. Any time during conversation request to connect you with a live agent. The call will be transferred to your agent.

**Congratulations, you have officially completed the Autonomous Virtual Agent mission!** 🎉🎉

# 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 |
| **Queue** | Your_Attendee_ID_Queue |
| **Bussiness Hours** | Your_Attendee_ID_Bussiness_Hours |
| **Control Hub** | https://admin.webex.com/ |
| **Agent Desktop** | https://desktop.wxcc-us1.cisco.com/ |

# 7.

**Question Best Practices**

Please provide either your Name or Pod Number

This helps the proctors determine if they should engage you directly based on your question

slido