<VirtualAgent> with Google Dialogflow CCAI CX

Introduction

Twilio <VirtualAgent> with Dialogflow CX is a native telephony integration between Twilio and Google's DialogFlow CCAI CX. Through this integration, Twilio provides customers with a first class voice experience through our global, scalable telephony infrastructure, in addition to low-code, no-code APIs that encapsulate all the complexities of the underlying infrastructure so customers can focus on configuring the IVR and setting up the DialogFlow CX agent - while tapping into Google's Private Telephony Parter API's to provide advanced features for building conversational AI experiences.

The configuration of the DialogFlow CX agent(s) will be done on the DialogFlow CX console or by using the DialogFlow APIs. This document details how you can participate in the private beta for this integration.

Objective

The purpose of the Pilot is for select customers to build Proof of Concepts (POCs) to test out the new <VirtualAgent> functionality. Your feedback will inform the functionality before we launch to GA at which point customers can use the native integration in production.

Google DialogFlow CX Features

The <VirtualAgent> for DialogFlow CCAI CX supports advanced features such as:

- Native welcome intent support
  - Basic greeting to trigger when user calls in
- Passing custom parameters into the DialogFlow CX agent
  - Parameterize your agent's response or pass additional data about your user through custom parameters in the TwiML (e.g. First Name, Last Name, etc.)
- DTMF support
  - In addition to speech input, end user can provide a DTMF (Dual-tone multi-frequency signaling) input through their telephone keypad
- DialogFlow native barge-in
  - When enabled, an end-user can interrupt the virtual agent. When interrupted, DialogFlow will stop sending audio, and it will process the next end-user input
- Status callbacks

    - By adding a status callback URL in your TwiML, you can subscribe to a stream of events such as when agent processing starts/stops, intents are detected, transcripts from end users are generated, sentiment scores, etc.
- Live agent hand-off
    - Our support for status callbacks allows us to pass a stream of events to your application which would enable you to write custom code against it and build a live agent hand-off experience in your own contact center application.

Pre-requisites

1. Buy a Twilio number
2. Set up your DialogFlow CX agent within your Google Cloud Project
3. Configure the DialogFlow CX connector in your Twilio Console
4. Write a TwiML in your application to invoke the DialogFlow CX connector
5. Connect your Twilio number to the TwiML

Configuration

1. Set up your DialogFlow agent
2. Grant Twilio permissions to communicate with your Google DialogFlow project
3. Create your Conversation Profile ID (*Note*: you **MUST** complete steps #1 and #2 before this step)
4. Configure the DialogFlow CX Connector with your credentials

Setup your Google DialogFlow CX agent

See directions here on how to start using DialogFlow in Google Cloud.
Quickstart: https://cloud.google.com/dialogflow/cx/docs/quick
Prebuilt agents: https://cloud.google.com/dialogflow/cx/docs/concept/agents-prebuilt

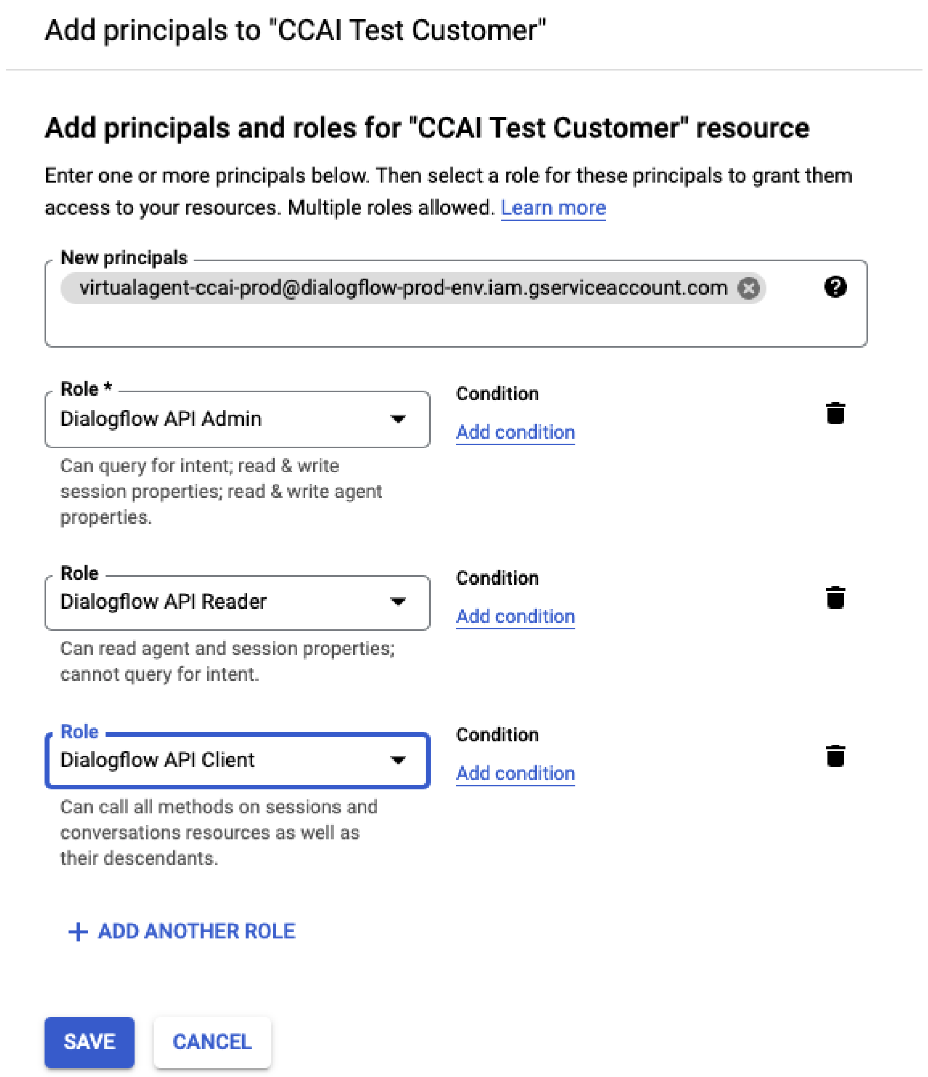
Grant Twilio Permissions

Note these one-time manual steps are required during private beta. We are working on a 1-click experience to further simplify this configuration. This is planned for release in public beta.

Provide Twilio's production service account access to communicate with your DialogFlow agent by following these steps (do not try to do this step elsewhere):

- Login in to your GCP console

- Select the Google Project in which you configured your DialogFlow Agent
- Navigate to IAM & Admin from the left navigation
- Click +Add
- Add virtualagent-ccai-prod@dialogflow-prod-env.iam.gserviceaccount.com as a member (ie. under "New principals") with the following roles:
    - DialogFlow API Admin
    - DialogFlow API Reader
    - DialogFlow API Client
- Click Save



Add principals to "CCAI Test Customer"

Add principals and roles for "CCAI Test Customer" resource

Enter one or more principals below. Then select a role for these principals to grant them access to your resources. Multiple roles allowed. Learn more

New principals: virtualagent-ccai-prod@dialogflow-prod-env.iam.gserviceaccount.com

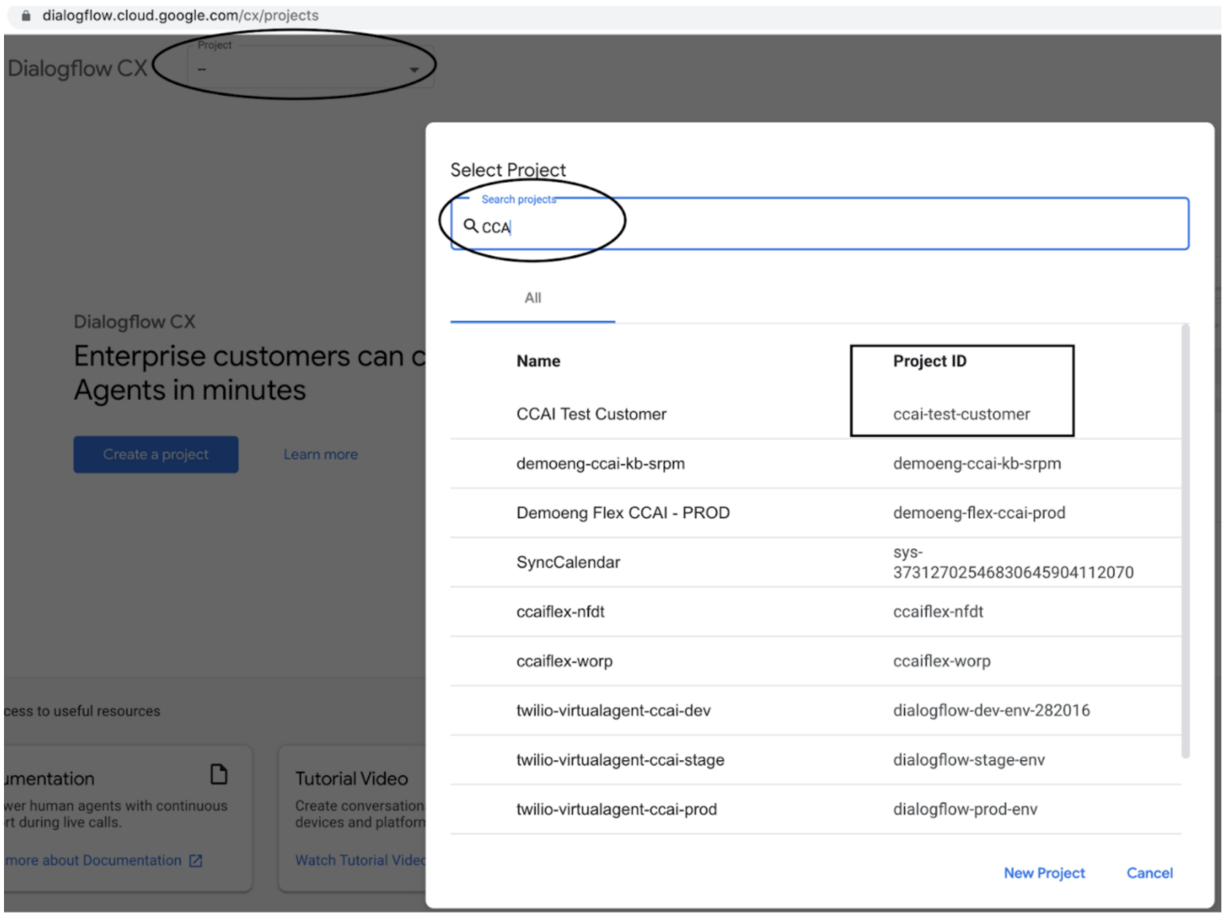
Role *: Dialogflow API Admin
Can query for intent; read & write session properties; read & write agent properties.
Condition: Add condition

Role: Dialogflow API Reader
Can read agent and session properties; cannot query for intent.
Condition: Add condition

Role: Dialogflow API Client
Can call all methods on sessions and conversations resources as well as their descendants.
Condition: Add condition

Get Google Cloud Project ID

Your Google Project ID is the **Project ID** field that can be found on the GCP console or on the settings page for your DialogFlow Agent (see screenshot below)
https://dialogflow.cloud.google.com/cx/projects


dialogflow.cloud.google.com/cx/projects

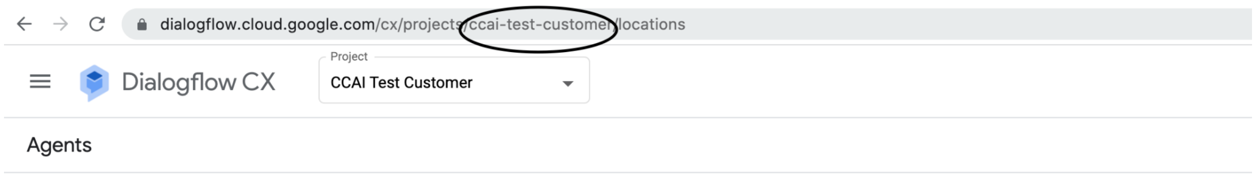
Select Project

Search projects: CCA

Name	Project ID
CCAI Test Customer	ccai-test-customer
demoeng-ccai-kb-srpm	demoeng-ccai-kb-srpm
Demoeng Flex CCAI - PROD	demoeng-flex-ccai-prod
SyncCalendar	sys-37312702546830645904112070
ccaiflex-nfdt	ccaiflex-nfdt
ccaiflex-worp	ccaiflex-worp
twilio-virtualagent-ccai-dev	dialogflow-dev-env-282016
twilio-virtualagent-ccai-stage	dialogflow-stage-env
twilio-virtualagent-ccai-prod	dialogflow-prod-env

Select a project by typing in a Display Name -> You will see Project ID here.

Alternatively, you can click on this project, and copy the Project ID from the URL as follows:


dialogflow.cloud.google.com/cx/projects/ccai-test-customer/locations

Get Conversation Profile ID

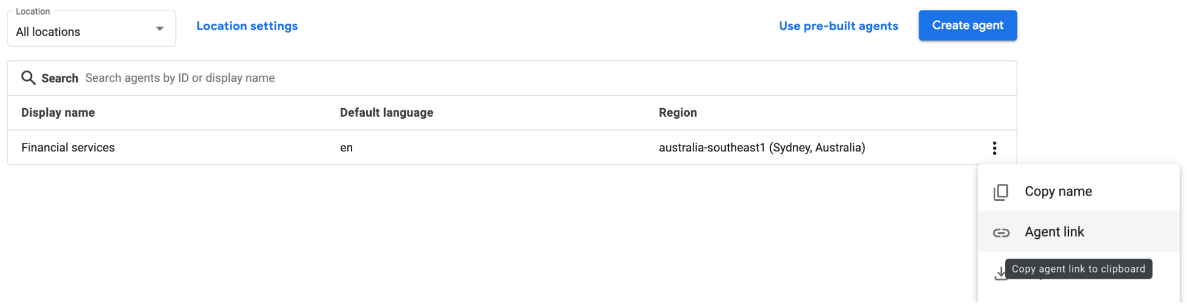
Conversation Profile ID is required to establish a connection between your Twilio and DialogFlow accounts. You can generate the Conversation Profile ID from your GCP console by going to this link:

https://agentassist.cloud.google.com/projects/[YOUR PROJECT ID]/locations/**[YOUR AGENT LOCATION]**/conversation-profiles

Note: The conversation profile should be in same edge/region as the agent ie. us-east1.

Step 1: Get URL data

In order to get the URL data: Google project ID and agent location, go to list of agents -> select the agent by clicking on 3 dots to the right -> click on **Agent link** (see below).



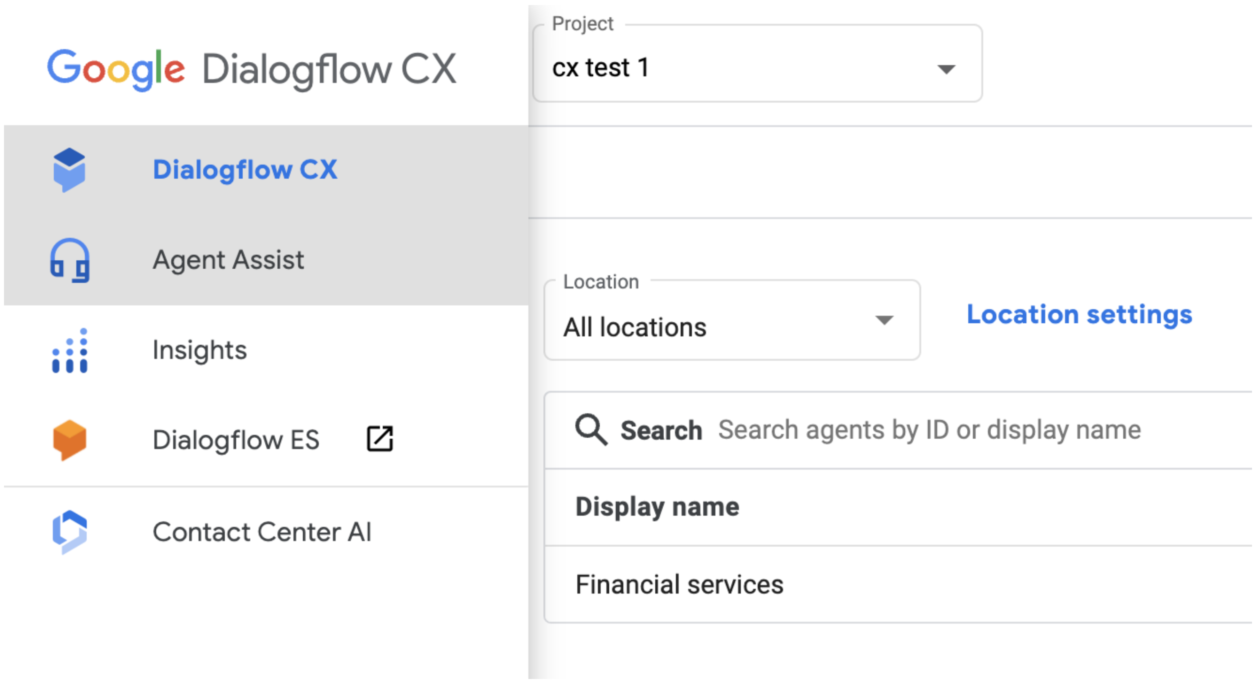
Display name	Default language	Region
Financial services	en	australia-southeast1 (Sydney, Australia)

Your URL would be something like:
https://dialogflow.cloud.google.com/cx/projects/**ccai-test-customer**/locations/**us-east1**/agents/7d3793a9-64af-4e49-a23c-a69ae90bc6cf

Your Google Project ID or Project ID is **ccai-test-customer**
Your agent location is **us-east1**

Step 2: Go to Agent Assist page

Now, go to the Agent Assist page for the project by clicking on the hamburger icon at top left corner of your page and selecting Agent Assist

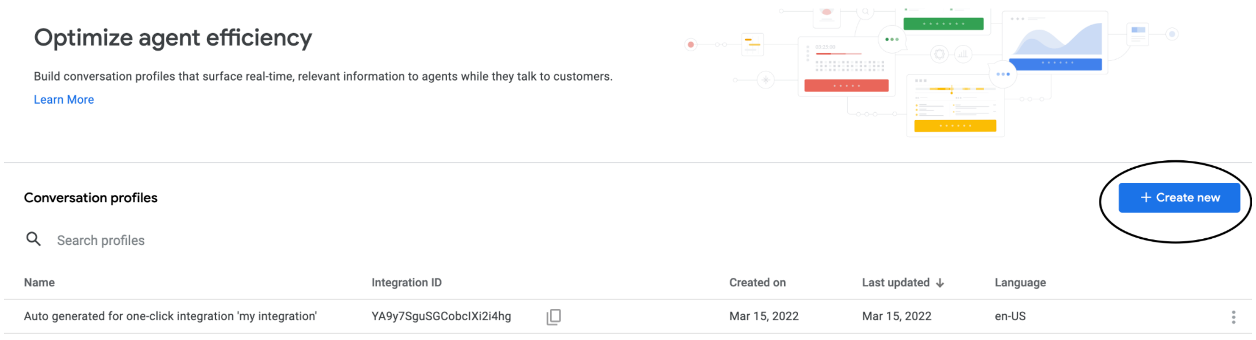


Display name
Financial services

Next, populate your Google Project ID and the agent's location in this link. This is the link to get a conversation profile ID: [https://agentassist.cloud.google.com/projects/**ccai-test-custome**r/locations/**us-east1**/conversation-profiles](https://agentassist.cloud.google.com/projects/ccai-test-customer/locations/us-east1/conversation-profiles)

Step 3: Create Conversation Profile ID

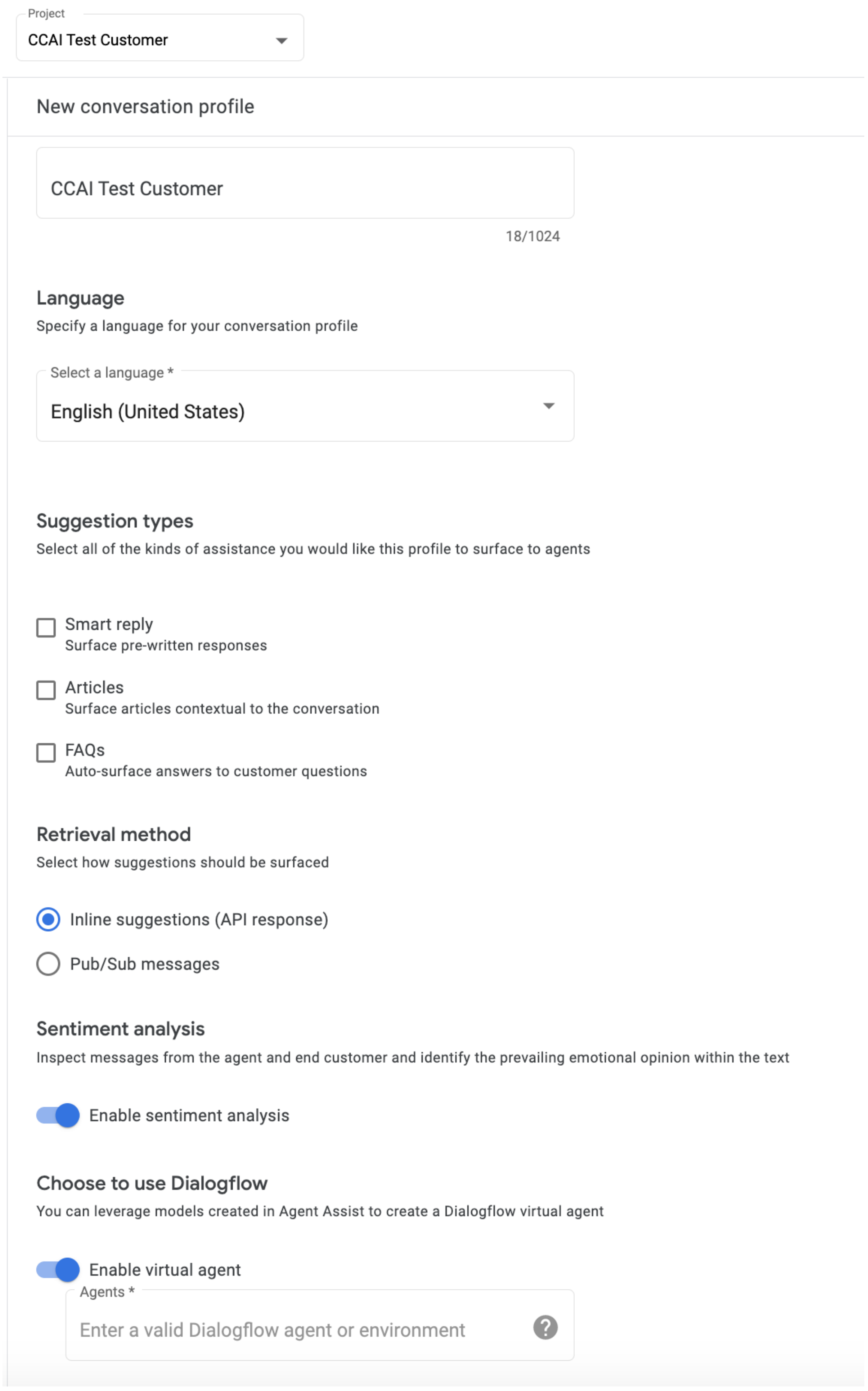
Now you should see the screenshot below where you can create a conversation profile ID. Click on **Create new.**


Optimize agent efficiency

Build conversation profiles that surface real-time, relevant information to agents while they talk to customers.

Name	Integration ID	Created on	Last updated	Language
Auto generated for one-click integration 'my integration'	YA9y7SguSGCobclXi2i4hg	Mar 15, 2022	Mar 15, 2022	en-US

You will see the below page that asks you to configure the details to generate a Conversation Profile ID


Project
CCAI Test Customer

New conversation profile

CCAI Test Customer

18/1024

Language

Specify a language for your conversation profile

Select a language *
English (United States)

Suggestion types

Select all of the kinds of assistance you would like this profile to surface to agents

- [ ] Smart reply
  Surface pre-written responses
- [ ] Articles
  Surface articles contextual to the conversation
- [ ] FAQs
  Auto-surface answers to customer questions

Retrieval method

Select how suggestions should be surfaced

● Inline suggestions (API response)
○ Pub/Sub messages

Sentiment analysis

Inspect messages from the agent and end customer and identify the prevailing emotional opinion within the text

[✓] Enable sentiment analysis

Choose to use Dialogflow

You can leverage models created in Agent Assist to create a Dialogflow virtual agent

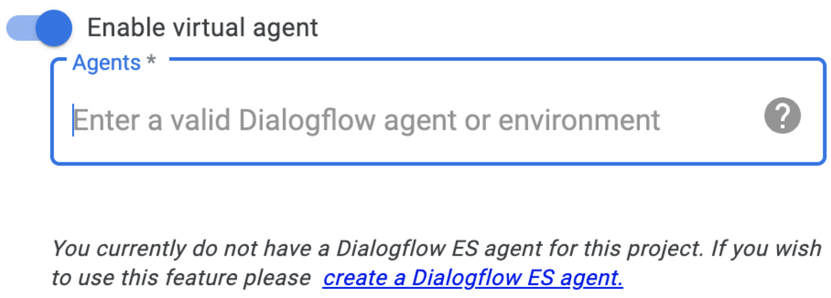
[✓] Enable virtual agent

Agents *
Enter a valid Dialogflow agent or environment

For instructions on filling out the form, see the Create a Conversation Profile documentation. At a minimum, you need to toggle **Enable virtual agent** on and select the Agent you want this Conversation Profile to have access to.

*Note: currently Twilio has only tested support for the "**Enable virtual agent**" turned on. Any features you choose to enable beyond the "Enable virtual agent" will need to be tested and vetted independently.*

Note: Google requires that you've first created a CX agent before it will be available to select from the "Enable virtual Agent" field. If you try to proceed without an existing agent, the below prompt will popup, directing you to create an ES agent. Please ignore the "ES" part of this, and instead double check the earlier steps that direct you to create a CX Agent first. Once you've created a CX agent, you'll be able to come back to this form and select it.


Enable virtual agent

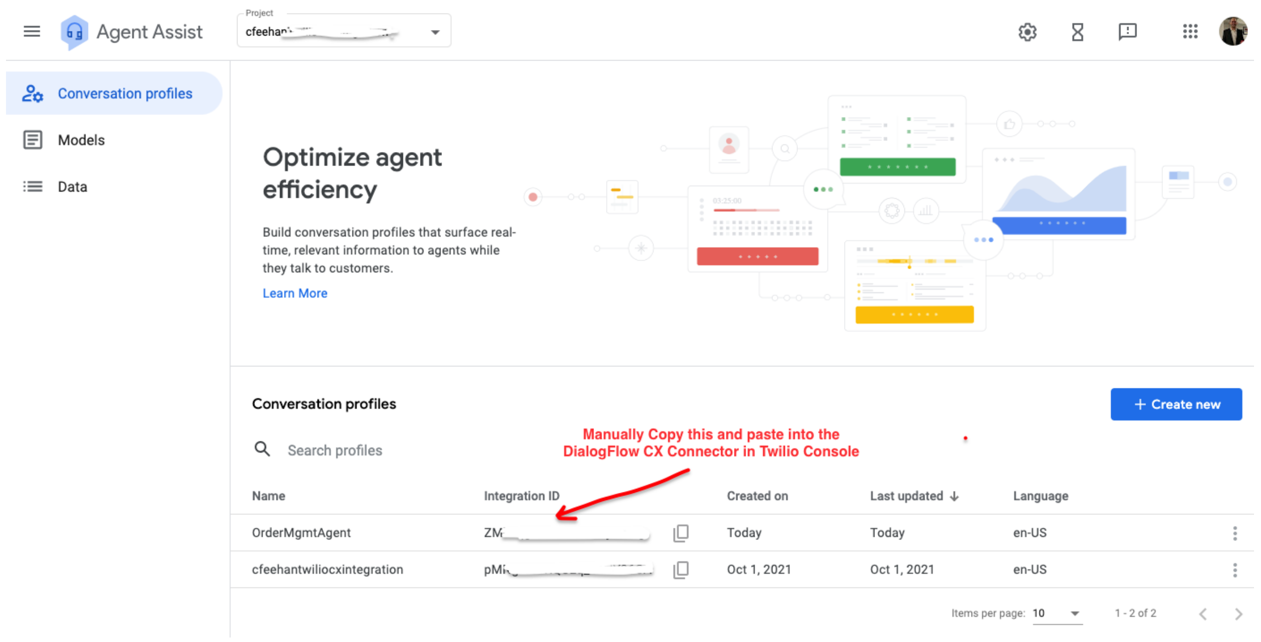
Agents *

Enter a valid Dialogflow agent or environment

You currently do not have a Dialogflow ES agent for this project. If you wish to use this feature please create a Dialogflow ES agent.

Once the conversation profile ID is created, you will see the Conversation Profile ID listed as the "**Integration ID**". You can now manually copy this ID as you will use it in the next step when you configure the Twilio DialogFlow CX Connector.

Note: the "copy" link on this page copies the full URL. You only want the "Integration ID" which is why we suggest a manual copy


Agent Assist

Project
cfeehan[redacted]

Conversation profiles

Models

Data

Optimize agent efficiency

Build conversation profiles that surface real-time, relevant information to agents while they talk to customers.

Learn More

Conversation profiles

+ Create new

Search profiles

Manually Copy this and paste into the DialogFlow CX Connector in Twilio Console

Name	Integration ID	Created on	Last updated ↓	Language
OrderMgmtAgent	ZM[redacted]	Today	Today	en-US
cfeehantwiliocxintegration	pMi[redacted]	Oct 1, 2021	Oct 1, 2021	en-US

Items per page: 10 1 - 2 of 2

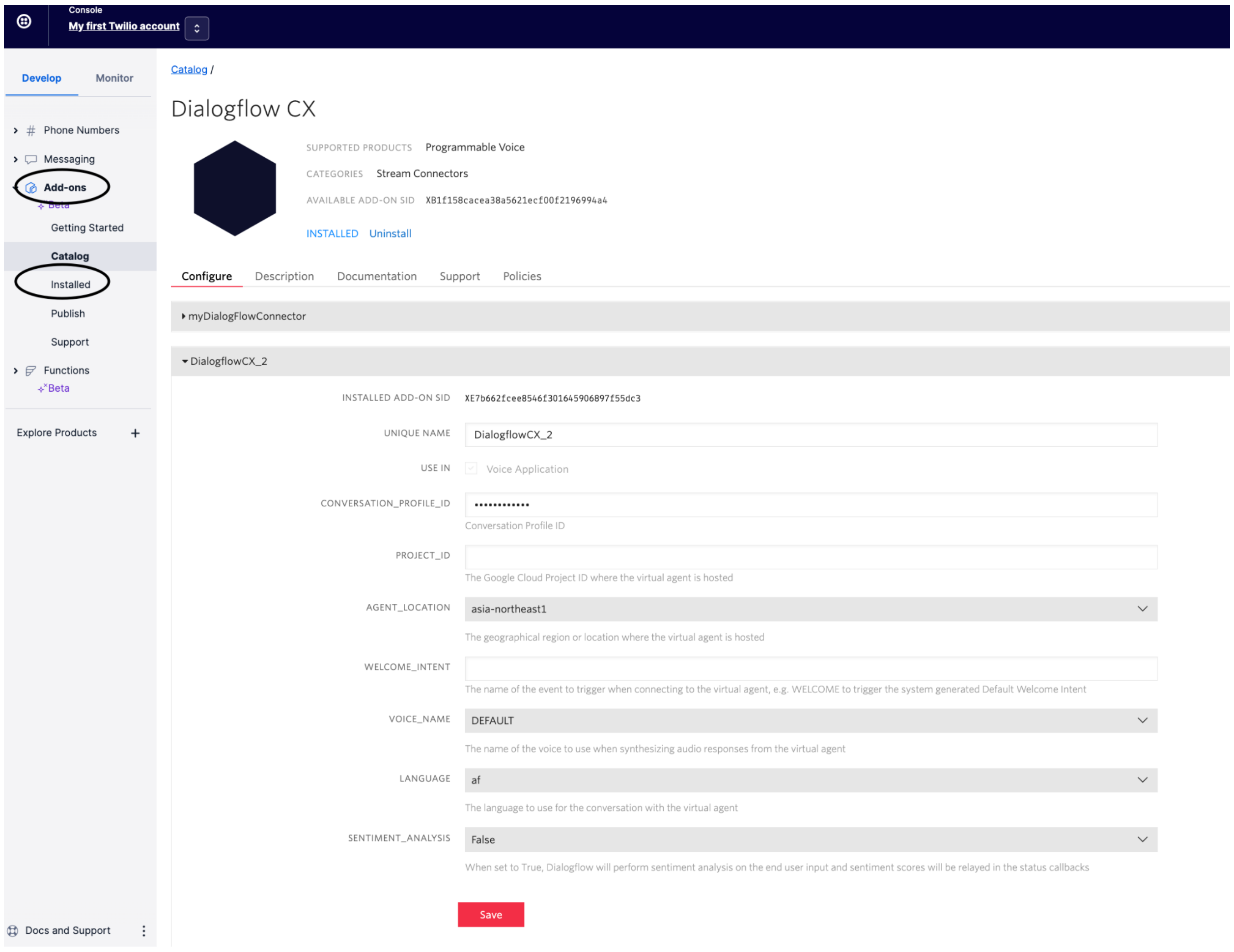
Now, it's time to configure your Twilio's Dialogflow CX connector.

Configure your DialogFlow CX connector on Twilio console

We will provision an empty DialogFlow CX connector and you will need to configure all the details as follows:

- Navigate to the Twilio console -> Add Ons on the left navigation -> Installed -> DialogFlow CX
- Click on DialogFlow CX - you will see an empty connector below

Note: If you have already configured the connector, you may not see the voice name field. You may add another instance by clicking on "Add another instance" to get voice name and CX supported language fields in the connector.



Dialogflow CX

SUPPORTED PRODUCTS Programmable Voice

CATEGORIES Stream Connectors

AVAILABLE ADD-ON SID XB1f158cacea38a5621ecf00f2196994a4

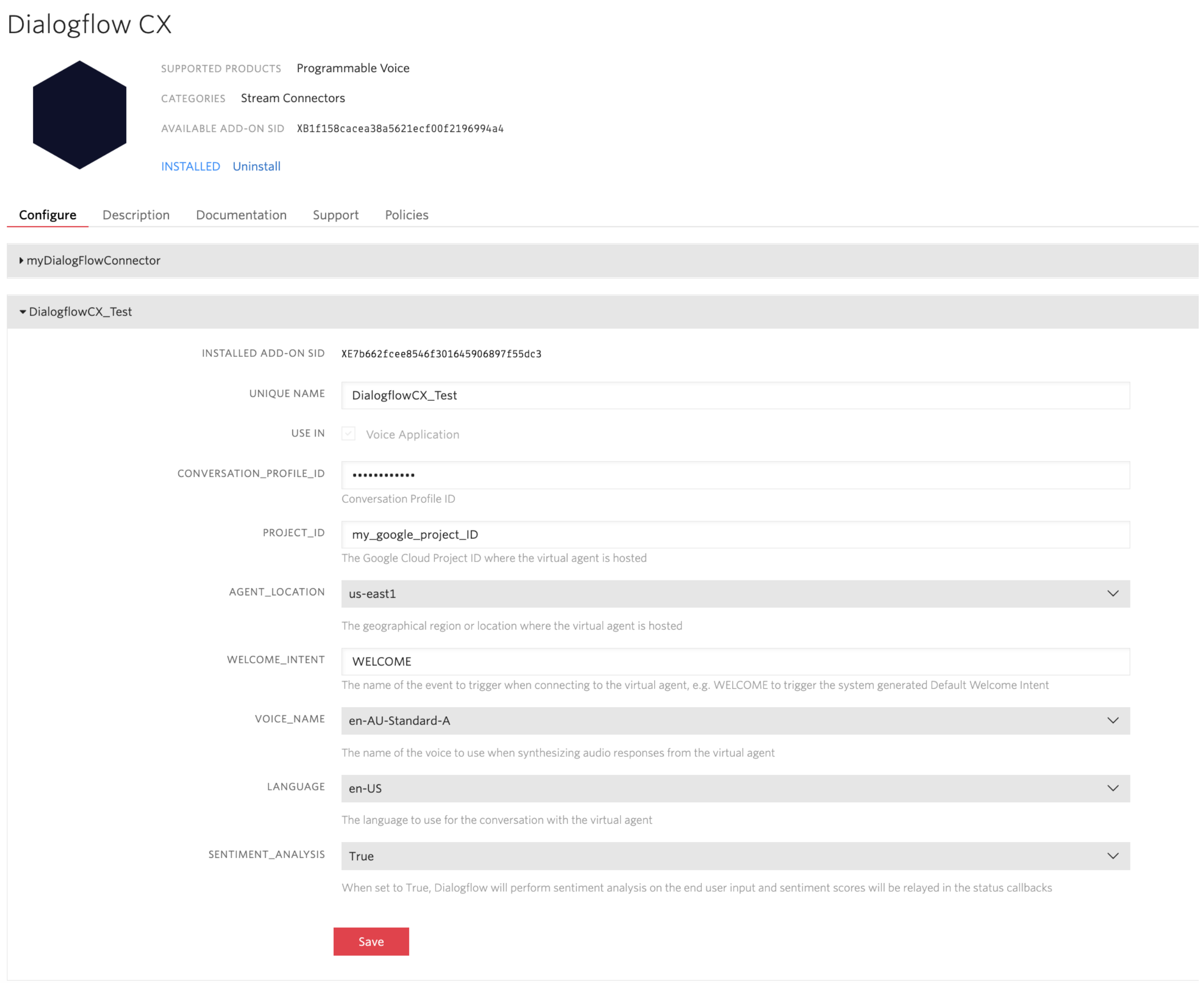
INSTALLED Uninstall

INSTALLED ADD-ON SID	XE7b662fcee8546f301645906897f55dc3
UNIQUE NAME	DialogflowCX_2
USE IN	Voice Application
CONVERSATION_PROFILE_ID	••••••••••••• Conversation Profile ID
PROJECT_ID	The Google Cloud Project ID where the virtual agent is hosted
AGENT_LOCATION	asia-northeast1 The geographical region or location where the virtual agent is hosted
WELCOME_INTENT	The name of the event to trigger when connecting to the virtual agent, e.g. WELCOME to trigger the system generated Default Welcome Intent
VOICE_NAME	DEFAULT The name of the voice to use when synthesizing audio responses from the virtual agent
LANGUAGE	af The language to use for the conversation with the virtual agent
SENTIMENT_ANALYSIS	False When set to True, Dialogflow will perform sentiment analysis on the end user input and sentiment scores will be relayed in the status callbacks

- Configure the add-on with following information
    - Unique name
        - This is the unique name of the connector which will also be referenced in your <VirtualAgent> TwiML as the "connectorName" attribute
    - Conversation Profile ID
        - Your conversation profile ID is the **Integration ID** field associated with your agent and can be found here. Example: XMk2cgINW5aTHO8pxm-llg
        - Note, each instance of a DialogFlow CX connector you configure on Twilio will be tied to one Conversation Profile therefore one agent.
    - Project ID
        - this is the **Project ID** field of your DialogFlow CX project, which can be found here
    - Agent_Location
        - Your agent location is the **Region** field associated to your agent and can be found here
    - Welcome_Intent

        - You can use **WELCOME** as a default value provided by Dialogflow CX bot
    - Voice_Name
        - The field indicates the TTS voice to use when synthesizing audio responses from the bot
        - Here is the link to voices currently supported by Google
    - Language
        - This field indicates the language you want the bot to talk in
        - Here is the link to languages currently supported by Google
    - Sentiment Analysis - TRUE/FALSE
        - When set to TRUE, Dialogflow will perform sentiment analysis on the end user input and sentiment scores will be relayed in the status callbacks

Once you have configured the connector with all the details, click on Save.


Dialogflow CX

SUPPORTED PRODUCTS Programmable Voice

CATEGORIES Stream Connectors

AVAILABLE ADD-ON SID XB1f158cacea38a5621ecf00f2196994a4

INSTALLED ADD-ON SID	XE7b662fcee8546f301645906897f55dc3
UNIQUE NAME	DialogflowCX_Test
USE IN	✓ Voice Application
CONVERSATION_PROFILE_ID	•••••••••••• Conversation Profile ID
PROJECT_ID	my_google_project_ID The Google Cloud Project ID where the virtual agent is hosted
AGENT_LOCATION	us-east1 The geographical region or location where the virtual agent is hosted
WELCOME_INTENT	WELCOME The name of the event to trigger when connecting to the virtual agent, e.g. WELCOME to trigger the system generated Default Welcome Intent
VOICE_NAME	en-AU-Standard-A The name of the voice to use when synthesizing audio responses from the virtual agent
LANGUAGE	en-US The language to use for the conversation with the virtual agent
SENTIMENT_ANALYSIS	True When set to True, Dialogflow will perform sentiment analysis on the end user input and sentiment scores will be relayed in the status callbacks

You are now ready to begin using <VirtualAgent> API and reference this connector name.

Development

The <VirtualAgent> TwiML allows customers to connect their Twilio voice call to their DialogFlow agent.

<VirtualAgent/> TwiML

Attribute	**Description**	**value**
connectorName (required)	Configured in DialogFlow CX connector instance in the Unique Name Field *Appears as "Unique Name" in the connector.*	string
statusCallback	URL to post status callbacks from Twilio	string
statusCallbackMethod	The HTTP method to use when requesting the statusCallback URL. Accepted values are GET or POST. Default is POST	string

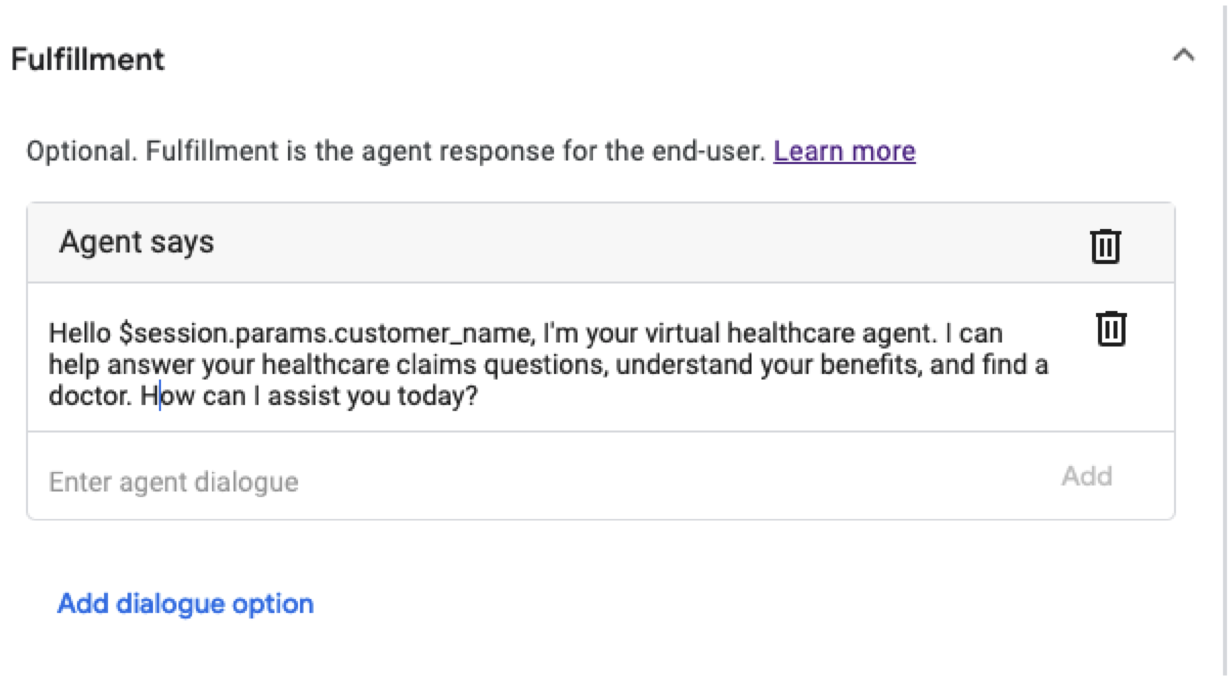
Custom Parameters

It is possible to pass custom key-value pairs to your DialogFlow CX agent by using nested <Parameter/> TwiML nouns. These will be made available as session parameters which can be referenced in your agent as $session.params.parameter-name.

Attribute	**Description**	**value**
name (required)	The name of the custom parameter	string
value (required)	The value of the custom parameter	string

For example, to provide a personalized agent greeting, you can supply a nested <Parameter/> element with the customer name and reference it in their agent fulfillment, e.g.

<Parameter name=”customer_name” value=”John”/>
This key-value pair will be passed in as a session parameter which can be referenced under $session.params.cutomer_name at runtime in your Dialogflow CX agent.


Fulfillment

Optional. Fulfillment is the agent response for the end-user. Learn more

Agent says

Hello $session.params.customer_name, I'm your virtual healthcare agent. I can help answer your healthcare claims questions, understand your benefits, and find a doctor. How can I assist you today?

Overriding Connector Properties

It is possible to override some of the properties of your DialogFlow CX connector by using nested <Config/> TwiML nouns. These will take precedence over the values defined in the corresponding DialogFlow CX connector. The following properties are supported as overrides:

- language: The language e.g. “es-ES”
- sentimentAnalysis: Either “true” or “false”
- voiceName: The TTS voice to use e.g. “en-US-Standard-C”
- welcomeIntent: The name of the event to trigger when connecting to the virtual agent, e.g. WELCOME to trigger the system generated Default Welcome Intent
- voiceModel: The TTS custom voice model to use e.g. “projects/662817328969/locations/us-central1/models/TTS5037753371260354560”.
    - If specified in addition to voiceName, this attribute will take precedence.
    - Requires that the additional IAM permission “automl.models.predict” is granted on the resource model to Twilio’s service account (virtualagent-ccai-prod@dialogflow-prod-env.iam.gserviceaccount.com).

Attribute	Description	value
name (required)	The name of the configuration to override	string
value (required)	The value of the configuration property	string

<Connect> TwiML

Attribute	Description	value
action (required)	Url at which your TwiML app is hosted i.e. after DialogFlow execution is done a customer may want to execute additional TwiML E.g., https://yourappurl.com/twiml *Default: none*	string

Status Callback

When a status change happens in DialogFlow, Twilio will make an HTTP request to the URL you specified in the statusCallback parameter in your TwiML with the following information:

Intent StatusCallbackEvent

Parameter	Description	Type	Optional?
AccountSid	Twilio Account SID	string	no
CallSid	Twilio Call SID	string	no
Timestamp	Time of the event, conformant to UTC ISO 8601 Timestamp.	string	no
StatusCallbackEvent	The event type (e.g. intent)	string	no
VirtualAgentProvider	The VirtualAgent provider (e.g. DialogFlowCX)	string	no
VirtualAgentProvi	JSON string containing the provider specific	JSON object	no

derData	intent event data (see the parameters below)		
	ConversationId: Unique identifier for this conversation provided by Google	string	no
	EndUserId: Unique identifier for the end user participant provided by Google	string	no
	ReplyText: Response text of the bot	string	no
	LanguageCode: The language that was triggered during intent detection.	string	yes
	Parameters: JSON object with key-value pairs containing the collected session parameters	object	yes
	Confidence: The intent detection confidence. Values range from 0.0 (completely uncertain) to 1.0 (completely certain)	number	yes
	ResolvedInput: Final text input matched against. This value may be different from the original input because of spelling correction or other processing.	string	yes
	IntentId: Intent ID provided by Google	string	yes
	IntentDisplayName: The human readable name of this intent	string	yes
	SentimentAnalysisScore: Sentiment score between -1.0 (negative sentiment) and 1.0 (positive sentiment).	number	yes
	SentimentAnalysisMagnitude: A non-negative number in the [0, +inf) range, which represents the absolute magnitude of sentiment, regardless of score (positive or negative).	number	yes

You can navigate to Monitor -> Logs - Calls in your Twilio console and see the call logs for all the status callbacks (events, transcripts, matched intents, etc.) fired over the lifecycle of the call.

Action Callback

In addition to the standard <Connect> action callback parameters (CallSid, AccountSid, From, To, etc), they will also be enriched with the following <VirtualAgent> context parameters:

Parameter	**Description**	**Type**	**Optional?**
VirtualAgentProvider	The VirtualAgent provider e.g. DialogFlowCX	string	no
VirtualAgentStatus	The status of the VirtualAgent processing. One of: ● Completed: VirtualAgent session was terminated by one of the following: ○ VirtualAgent indicated end of conversation ○ Caller hangup ● Live-agent-handoff: VirtualAgent returned a live agent handoff response ● Failed: An error occurred during VirtualAgent processing	string	no
VirtualAgentProviderData	JSON string containing the provider specific action callback event data (see below)	JSON object	no
	ConversationId: Unique identifier for this conversation provided by Google	string	yes
	EndUserId: Unique identifier for the end user participant provided by Google	string	yes
	AgentHandoffParameters	object	yes

Examples

Sample TwiML you write in your application:

```
<Response>
 <Connect action=”https://myactionurl.com”>
 <VirtualAgent connectorName=”myDialogFlowConnector”
statusCallback=”https://mycallbackurl.com”>
      <Config name=”sentimentAnalysis” value=”true”/>
      <Parameter name=”FirstName” value =”John”/>
      <Parameter name=”LastName” value =”Doe” />
      </VirtualAgent>
  </Connect>
</Response>
```

Status Callback Example:

CallSid	"CA1e0a9f65062923fc8b82f9dfa8cf3506"
StatusCallbackEvent	"intent"
Timestamp	"2022-03-10T17:25:05.409Z"
VirtualAgentProvider	"DialogFlowCX"
VirtualAgentProviderData	"{\"ConversationId\":\"projects/ccai-test-customer/locations/us-east1/conversations/103yVjufttzTIqgHcN0WqFOsg\",\"EndUserId\":\"projects/ccai-test-customer/locations/us-east1/conversations/103yVjufttzTIqgHcN0WqFOsg/participants/Nf39pAdZTzGIEuTMXWt3oA\",\"ReplyText\":\"What type of doctor would you like to see?\",\"LanguageCode\":\"en\",\"Parameters\":{\"customer_name\":\"Burton Guster\",\"date_of_birth\":{\"day\":5.0,\"month\":10.0,\"year\":1984.0},\"location\":\"California\",\"member_id\":\"Esther\",\"primary_subscriber\":\"self\"},\"Confidence\":0.3,\"ResolvedInput\":\"I am the primary subscriber\",\"SentimentAnalysisScore\":0.2,\"SentimentAnalysisMagnitude\":0.2}"
AccountSid	"ACbefe636847d9926046da228e9084ee59"

Status Callback Twilio Functions:

```
exports.handler = function(context, event, callback) {
  console.log('status callback', event);
  return callback(null, null);
};
```

Action Callback Example:

Called	"+14155270824"
ToState	"CA"
VirtualAgentStatus	"completed"
CallerCountry	"US"

Direction	"inbound"
CallerState	"CA"
ToZip	"94114"
VirtualAgentProviderData	"{\"ConversationId\":\"projects/ccai-test-customer/locations/us-east1/conversations/103yVjufttzTIqgHcN0WqFOsg\",\"EndUserId\":\"projects/ccai-test-customer/locations/us-east1/conversations/103yVjufttzTIqgHcN0WqFOsg/participants/Nf39pAdZTzGIEuTMXWt3oA\"}"
CallSid	"CA1e0a9f65062923fc8b82f9dfa8cf3506"
To	"+14155270824"
CallerZip	"95050"
ToCountry	"US"
CalledZip	"94114"
ApiVersion	"2010-04-01"
CalledCity	"SAN FRANCISCO"
CallStatus	"in-progress"
AddOns	"{\"status\":\"successful\",\"message\":null,\"code\":null,\"results\":{}}"
From	"+14084708848"
AccountSid	"ACbefe636847d9926046da228e9084ee59"
CalledCountry	"US"
CallerCity	"SAN JOSE"
ToCity	"SAN FRANCISCO"
FromCountry	"US"
Caller	"+14084708848"
FromCity	"SAN JOSE"
CalledState	"CA"
FromZip	"95050"
VirtualAgentProvider	"DialogFlowCX"
FromState	"CA"

Actions Callback Twilio Functions:

```
exports.handler = function(context, event, callback) {
  console.log('action callback', event);

  let callsid = event.CallSid;
  let status = event.VirtualAgentStatus;
  let provider = event.VirtualAgentProvider;
  let data = JSON.parse(event.VirtualAgentProviderData);
  console.log('context', callsid, status, provider, data);

  let twiml = new Twilio.twiml.VoiceResponse();
  if (status == 'completed') {
    twiml.hangup();
  } else if (status == 'live-agent-handoff') {
    twiml.enqueue({}, data.AgentHandoffParameters.routing_attributes.department);
  }
  return callback(null, twiml);
};
```

Native welcome intent support

You can use the system generated default welcome intent called “WELCOME” and configure it in your DialogFlow CX connector within your Twilio console. The intent can be edited as desired.

Custom Parameters Support

You can parameterize the agent’s response by passing custom parameters in the TwiML and also configuring the parameters in your agent’s setting in the GCP console.

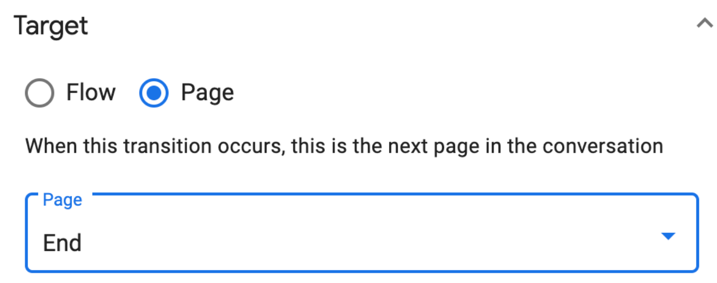
Here is an example of sending your customer’s First name and Last name to the agent so that agent can use it in the greetings.

```
<VirtualAgent connectorName=”myDialogFlowConnector” statusCallback=”https://mycallbackurl.com”>
    <Config name=”sentimentAnalysis” value=”true”/>
    <Parameter name=”FirstName” value =”John”/>
    <Parameter name=”LastName” value =”Doe” />
</VirtualAgent>
```

End the conversation with an agent

Mark an intent as end of interaction if you would like to end the DialogFlow session and end the phone call or execute the next TwiML in your application, either by using the action parameter, or TwiML following the <Connect> verb.

On the UI, you can typically set the transition target to ’End’ page to indicate that the conversation with the agent should end after this DialogFlow response.


Target

Flow [✓ Page]

When this transition occurs, this is the next page in the conversation

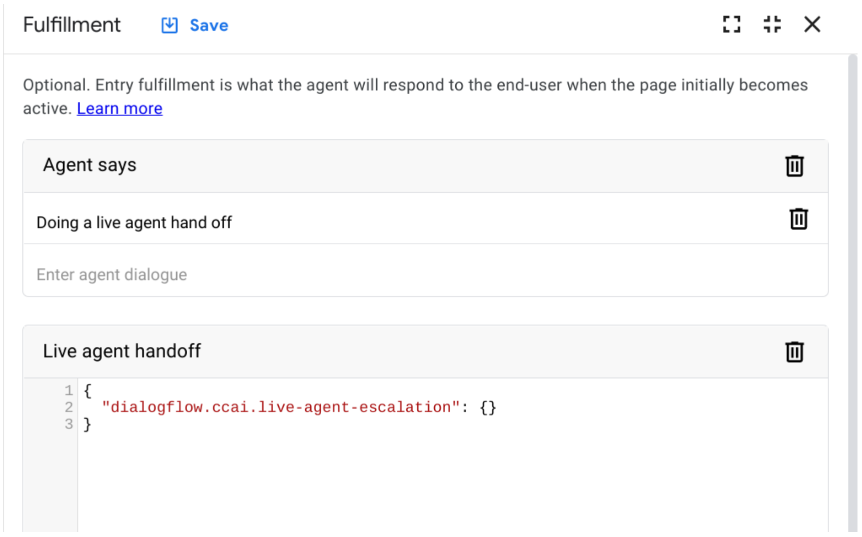
Page: End

Live Agent Hand-off

When the user escalates the call to a human agent, VirtualAgentStatus is set to 'live-agent-handoff'. This indicates that you should end the conversation between a user and the DialogFlow agent and hand over the call to a live human agent. This field value has a sub field `metadata`, which contains a JSON-like value for specifying additional information on how to do the handoff.

```
{
  "dialogflow.ccai.live-agent-escalation": {}
}
```

The DialogFlow Console directly exposes this message type as one of the options for agent prompt:



Optional. Entry fulfillment is what the agent will respond to the end-user when the page initially becomes active. Learn more

Doing a live agent hand off

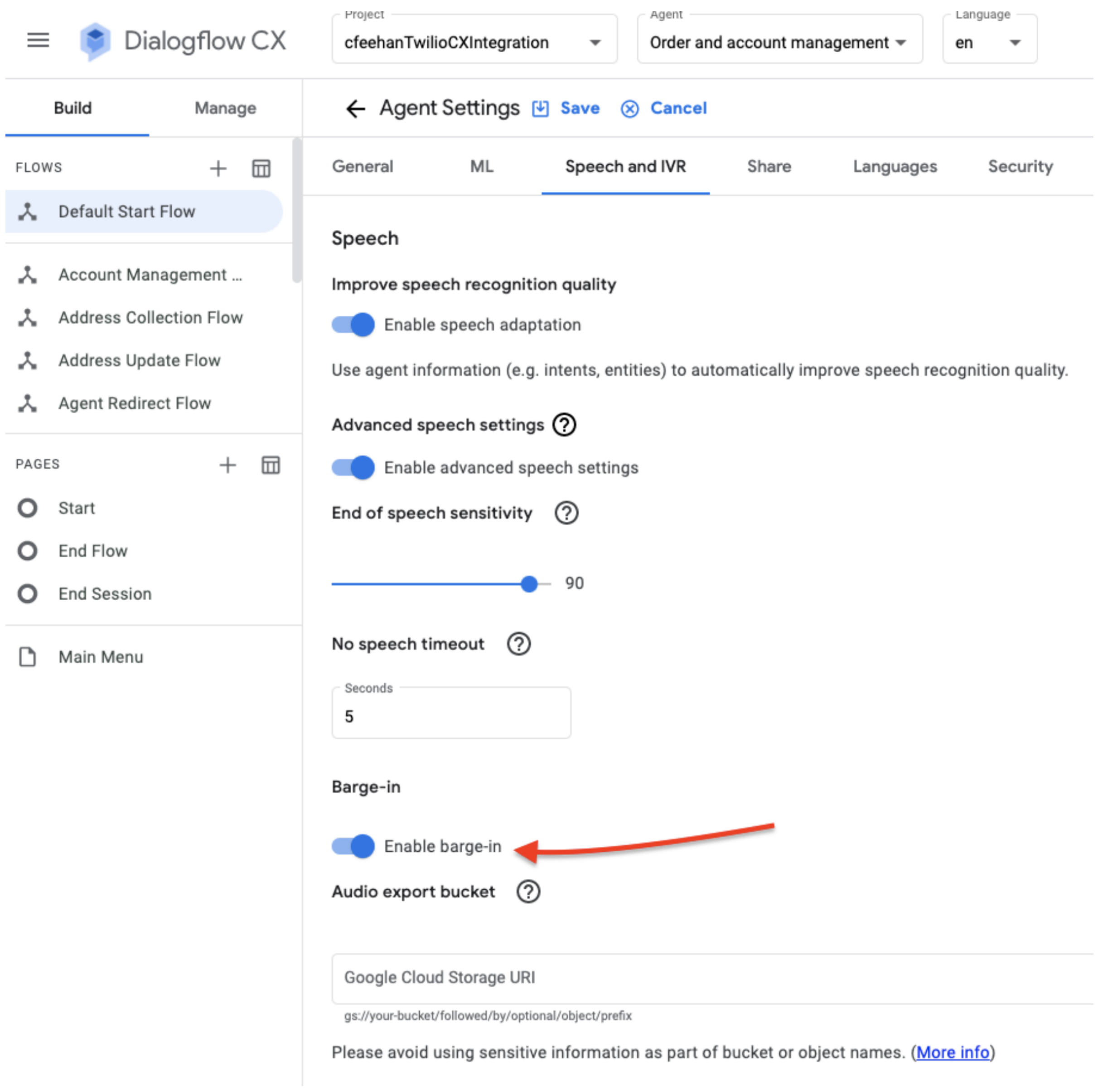
Enter agent dialogue

```
{
  "dialogflow.ccai.live-agent-escalation": {}
}
```

Note: We will provide all the data through status callbacks that would help you build a live agent experience within Twilio Flex or your own contact center application.

Barge-In

In order to turn on the barge-in feature, go to your Agent Settings in your GCP console -> Speech and IVR section -> Enable barge-in. Turn on this toggle button. Google recommends setting the End of speech sensitivity to 90.



Speech

Improve speech recognition quality

Enable speech adaptation

Use agent information (e.g. intents, entities) to automatically improve speech recognition quality.

Advanced speech settings

Enable advanced speech settings

End of speech sensitivity

90

No speech timeout

Seconds
5

Barge-in

Enable barge-in

Audio export bucket

Google Cloud Storage URI

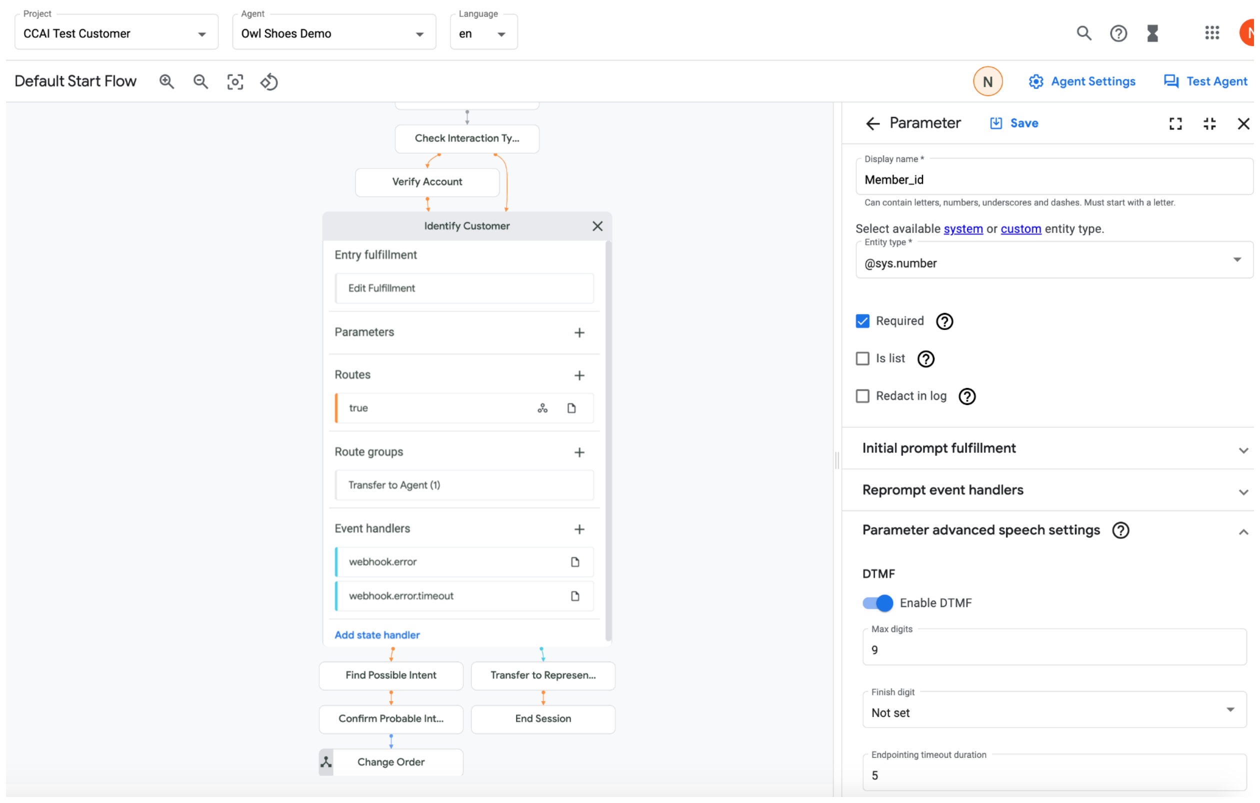
gs://your-bucket/followed/by/optional/object/prefix

Please avoid using sensitive information as part of bucket or object names. (More info)

DTMF Support

You can enable and configure DTMF (Dual-tone multi-frequency signaling) for a parameter. Once enabled, your customer can use their telephone keypad to provide parameter values for an agent using a telephony integration.

To enable DTMF, go to your agent in the GCP console. Note, each step that requires collecting DTMF input from the user to be used by your bot, you will need to enable DTMF by clicking on **Enable DTMF** toggle. (See below).



Project: CCAI Test Customer
Agent: Owl Shoes Demo
Language: en

Display name *: Member_id
Can contain letters, numbers, underscores and dashes. Must start with a letter.
Select available system or custom entity type.
Entity type *: @sys.number
[✓ Required]
Is list
Redact in log

DTMF
[✓ Enable DTMF]
Max digits: 9
Finish digit: Not set
Endpointing timeout duration: 5

Gaps addressed from ES

In our DialogFlow CX integration, we have addressed following gaps from DialogFlow ES:

- Native welcome intent support,
- DTMF support,
- DialogFlow native barge-in,
- Passing custom parameters into the DialogFlow CX agent,
- Passing information back to application through status callbacks such as transcripts containing caller and bot conversation, sentiment scores, etc.

Current Limitations in CX

- TwiML bins in the Twilio console may erroneously claim your “TwiML syntax is invalid” for <VirtualAgent/> TwiML. This will be addressed before public beta
- Role that does integration with DialogFlow is not regionalised. If customers have a DialogFlow agent set outside of the US, then there may be some latency issues, data residency, etc.

- In the creation of a conversation profile ID within a GCP account, Twilio has currently only tested support for the "**Enable virtual agent**" turned on. Any features you choose to enable beyond the "Enable virtual agent" while configuring your conversation profile ID - will need to be tested and vetted independently.

Public Beta

When Twilio's native integration with DialogFlow officially launches you'll be able to leverage the one-click telephony experience from DialogFlow's console to seamlessly associate a Twilio phone number with a DialogFlow agent.

We will also add enhanced support for our Twilio Studio Flow <VirtualAgent> widget supporting DialogFlow CX and update the helper libraries.

Feedback

We are looking forward to hearing from you! Please reach out to your Twilio Account Manager and nchheda@twilio.com. We're looking for feedback on, but not limited to:

- Developer experience
- Level of configurability
- Status callbacks
- Ease of transferring control of the call back to Twilio (to execute the next TwiML)
- Regional implications
- Applicable use cases