Service Cloud Voice Implementation Guide

Version 66.0, Spring ’26

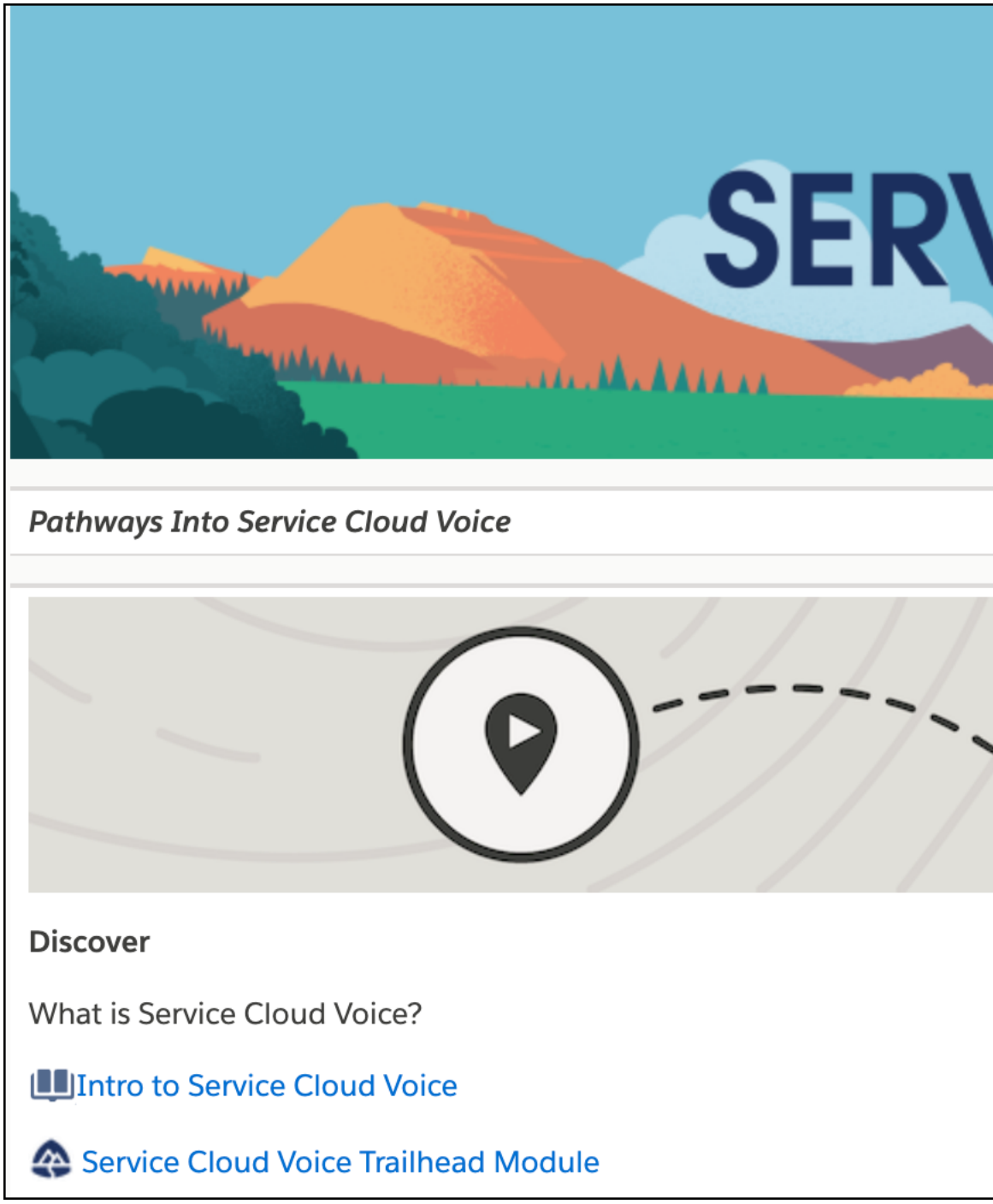
Last updated: March 27, 2026

CONTENTS

CHAPTER 1 Service Cloud Voice Implementation Guide

This guide describes how a technical admin or a developer can fine-tune their Service Cloud Voice implementation after it is set up. It includes examples for implementing common use cases. It explains how to use contact flows, Lambda functions, the Telephony Integration REST API, the Connect API, and the Voice Toolkit API to extend the functionality. It also explains how Voice scales during periods of high call volumes.

Before using this guide, ensure that you already have Service Cloud Voice set up. And be sure to bookmark the Service Cloud Voice Learning Map, a centralized collection of Voice resources that provides documentation, demos, and more for every step in your Voice implementation.



SEE ALSO:

- Service Cloud Voice Learning Map
- Salesforce Help: Service Cloud Voice
- *Trailhead:* Service Cloud Voice

CHAPTER 2 Service Cloud Voice Implementation Guide Release Notes

This section describes recent changes that have been made to this guide.

To see the latest feature updates for Service Cloud Voice, visit Service Cloud Voice release notes.

Updates for Spring ‘26

- Added this release notes page to serve as a changelog for this guide.
- Added the Route a Voice Call on page 93 API to the Telephony Integration API.
- Added the Request a Callback on page 95 API to the Telephony Integration API.
- Updated the InvokeTelephonyIntegrationApiFunction Lambda Function on page 145 for the new APIs to route a voice call and request a callback.

CHAPTER 3 Examples of Common Use Cases

In this chapter ...

Salesforce and Amazon Web Services provide examples to help you get started using Service Cloud Voice. These examples illustrate how to implement common use cases such as SMS transfers, linking calls to cases, and voicemail setup. The examples are foundational, so you can build upon them to support your company's business processes.

Your business processes might be a bit different than those reflected in the examples. In many cases, you can modify an example to achieve your desired result. For instance, an example that shows how to automatically link a call record to a case record can be modified to link to a lead record or other type of record.

Each example indicates the ease of implementation and estimated time to implement. They assume a familiarity with basic Salesforce admin skills.

Amazon Connect's solutions team also offers implementation examples for Service Cloud Voice. These examples demonstrate how to complete tasks like letting customers leave voicemails for reps. We've introduced a selection of them in Examples from Amazon Connect. For a full list, see the Amazon Connect Campground for Service Cloud Voice in GitHub. Salesforce does not maintain the examples provided by Amazon Connect.

Pass Data from the Interactive Voice Response (IVR) System

This example explains how to pass data from the Interactive Voice Response (IVR) system in Amazon Connect to your Salesforce org.

Ease of Implementation	Easy
Estimated Time to Implement	20 minutes

Prerequisites

If your contact center was created before the Spring '21 release, download the Sample SCV Inbound Flow contact flow from the Service Cloud Voice repo on Github. Import it into Amazon Connect if it's not already imported. Be sure to update the Invoke AWS Lambda function block so that it correctly points to the Lambda function in your Amazon Connect instance. If your contact center was created after the Spring '21 release, this contact flow is already installed into Amazon Connect.

To implement this use case, you need either the AWS root user or AWS administrator credentials.

Step 1: Create a Custom Field on the VoiceCall Object

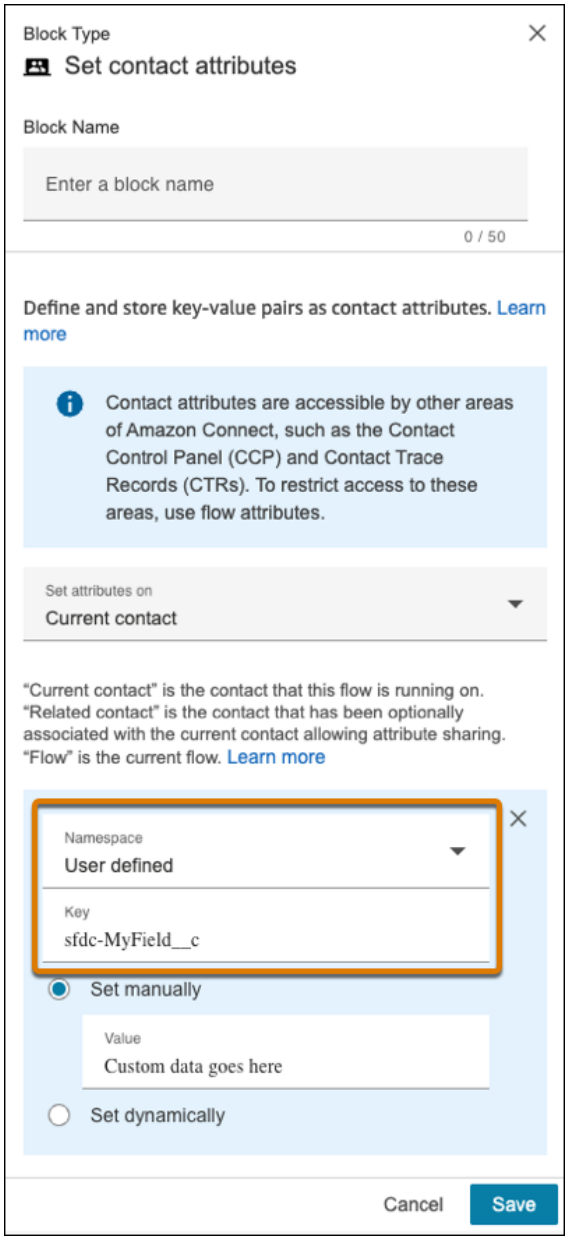
Create a custom field on the VoiceCall object in your Salesforce org that corresponds to the data that you want passed from the IVR.

1. From Setup, at the top of the page, select **Object Manager**.
2. In the Quick Find box, type *Voice Call* and select the **Voice Call** object.
3. Select **Fields & Relationships** from the left pane.
4. Select **New** to create a field.
5. Follow the steps to create a custom field to hold the data. A text field is the most flexible, but you can use any type of supported field. Geolocation and read-only fields aren't supported types for custom fields. To learn more about creating custom fields, see Create Custom Fields.

Step 2: Modify the Contact Flow

Add a "Set contact attributes" block in your contact flow to add custom field data.

1. Log in to your Amazon Connect instance.
2. Hover over **Routing**, then click **Flows**.
3. Select the **Sample SCV Inbound Flow** contact flow (or whichever inbound flow you're using).
4. Create a "Set contact attributes" block *before* the `InvokeTelephonyIntegrationApiFunction` on page 145 Lambda function call. For the destination attribute, specify the name of the custom field with the "sfdc-" prefix added. The value of this attribute is the data to pass into the custom field For example, `MyField__c` would become `sfdc-MyField__c`. For the value, specify whatever data you want stored in your VoiceCall record.



5. Publish the updated contact flow.

Test This Example

To test this example, perform an inbound call and review the VoiceCall record that is created for that call. The field is set with the value from the Amazon Connect contact flow.

Set Up Call Transcription

To convert call conversations to text, set up real-time call transcription. The transcripts appear in the voice call record in real time and after the call. Service Cloud Voice supports transcription from Amazon Transcribe and Contact Lens, and both services are available with Amazon Connect.

Note: When an additional rep joins a call between a rep and a customer, Amazon Connect stores the audio streams from each rep in a separate channel. As a result, you might see duplicate call transcripts, and the initial reps words might be attributed to the customer. This issue applies to transcription from both Amazon Transcribe and Contact Lens.

Transcribe Calls with Amazon Transcribe

Use Amazon Transcribe to transcribe calls in real time. With Amazon Transcribe, transcriptions are capped at 15 minutes because of a Lambda function limitation.

Transcribe Longer Calls with Contact Lens

Use Contact Lens to transcribe calls that are longer than 15 minutes. If you use Amazon Transcribe, only the first 15 minutes of a call are transcribed due to a Lambda function limitation. To trigger real-time transcription of a call with Contact Lens, enable Contact Lens and real-time transcription for Amazon Connect in your contact flow.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise** and **Unlimited** Editions

Available in: Sales Cloud, Service Cloud, and Government Cloud as an add-on license. Government Cloud is supported only on Service Cloud Voice with Amazon Connect and Service Cloud Voice with Partner Telephony from Amazon Connect.

Transcribe Calls with Amazon Transcribe

Use Amazon Transcribe to transcribe calls in real time. With Amazon Transcribe, transcriptions are capped at 15 minutes because of a Lambda function limitation.

Ease of Implementation	Medium
Estimated Time to Implement	45 minutes

For a list of supported transcription languages with Amazon Transcribe, see the Amazon documentation.

Set Up Call Transcription with Amazon Transcribe

To set up real-time call transcription with Amazon Transcribe, configure the Amazon Connect contact flow to invoke the kvsConsumerTrigger Lambda function, and verify the Amazon AWS region and service quota settings.

Improve Amazon Transcribe's Transcription Accuracy with Custom Vocabulary and Filters

Use a custom vocabulary to ensure that your transcript displays terms with the correct spelling and capitalization. Use vocabulary filters to filter or mask unwanted content, like profanity.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise** and **Unlimited** Editions

Available in: Sales Cloud, Service Cloud, and Government Cloud as an add-on license. Government Cloud is supported only on Service Cloud Voice with Amazon Connect and Service Cloud Voice with Partner Telephony from Amazon Connect.

Amazon Transcribe's Transcription Languages Supported by Service Cloud Voice
Service Cloud Voice supports multiple languages.

SEE ALSO:

kvsConsumerTrigger Lambda Function
Amazon Connect Flows
Salesforce Help: Update Your Contact Center

Set Up Call Transcription with Amazon Transcribe

To set up real-time call transcription with Amazon Transcribe, configure the Amazon Connect contact flow to invoke the kvsConsumerTrigger Lambda function, and verify the Amazon AWS region and service quota settings.

Ease of Implementation	Medium
Estimated Time to Implement	20 minutes

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise** and **Unlimited** Editions

Available in: Sales Cloud, Service Cloud, and Government Cloud as an add-on license. Government Cloud is supported only on Service Cloud Voice with Amazon Connect and Service Cloud Voice with Partner Telephony from Amazon Connect.

1. Configure the Amazon Connect contact flow to invoke the `kvsConsumerTrigger` Lambda function.

   If you need help with building a contact flow with this function, use one of these packaged contact flows that already have the `kvsConsumerTrigger` Lambda function set up.

   - Sample SCV Agent Whisper Flow for Amazon Transcribe
   - Sample SCV Outbound Flow With Transcription Using Amazon Transcribe

   To get the latest versions of these contact flows, see Update Your Contact Center.

2. Select an AWS region where Amazon Transcribe is available that is closest to where your contact center is located geographically. By default, the region for the transcription Lambda function is set to `us-west-2`, which is the Amazon Transcribe data center in Oregon. For best performance, select the AWS region that's closest to your contact center. For information about AWS regions that support Amazon Connect, see the AWS Regional Services topic in the AWS documentation.
3. If your organization streams many call transcriptions at the same time, ask AWS to increase the default service quota for concurrent transcription jobs for Amazon Transcribe. See Increase Amazon Service Quota.

Improve Amazon Transcribe's Transcription Accuracy with Custom Vocabulary and Filters

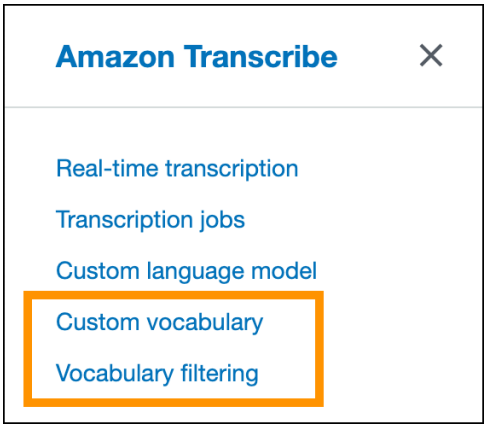
Use a custom vocabulary to ensure that your transcript displays terms with the correct spelling and capitalization. Use vocabulary filters to filter or mask unwanted content, like profanity.

Ease of Implementation	Easy
Estimated Time to Implement	30 minutes

Prerequisites

To implement this use case, you need either the AWS root user or AWS administrator credentials.

Step 1: Set Up Custom Vocabularies and/or Vocabulary Filtering in Amazon Transcribe



- To process speech more accurately, set up your custom vocabulary in Amazon Transcribe. This feature is useful for domain-specific terminology.
- To mask or remove words that you don't want to appear in the transcription, set up vocabulary filtering in Amazon Transcribe.

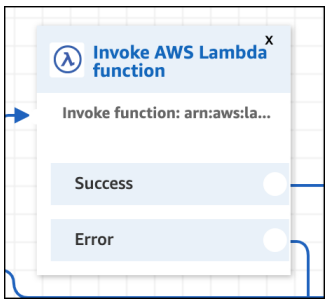
Be sure to record the name of the custom vocabulary and the name of the filter for the next step.

Step 2: Modify the Contact Flows That Perform Transcription

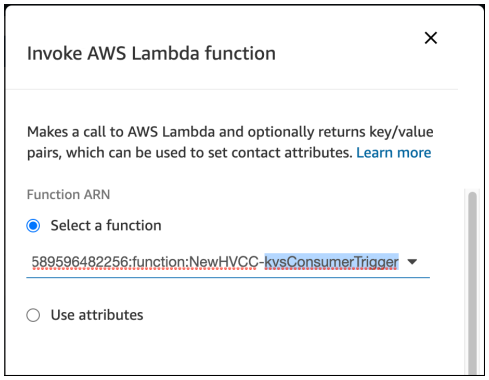
The kvsConsumerTrigger on page 163 Lambda function, which is installed with Service Cloud Voice, requires information about your vocabulary changes. This function starts the transcription process and is used by both inbound and outbound flows. Update these flows with additional parameters.

To update your flows, perform the following steps on all contact flows that use transcription:

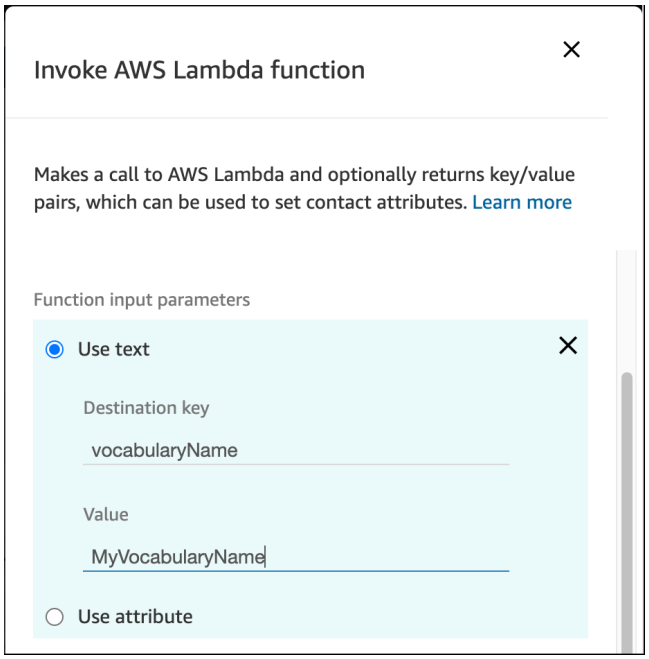
1. Find and edit any contact flow block that runs the kvsConsumerTrigger on page 163 Lambda function. This block should have the title **Invoke AWS Lambda function**.

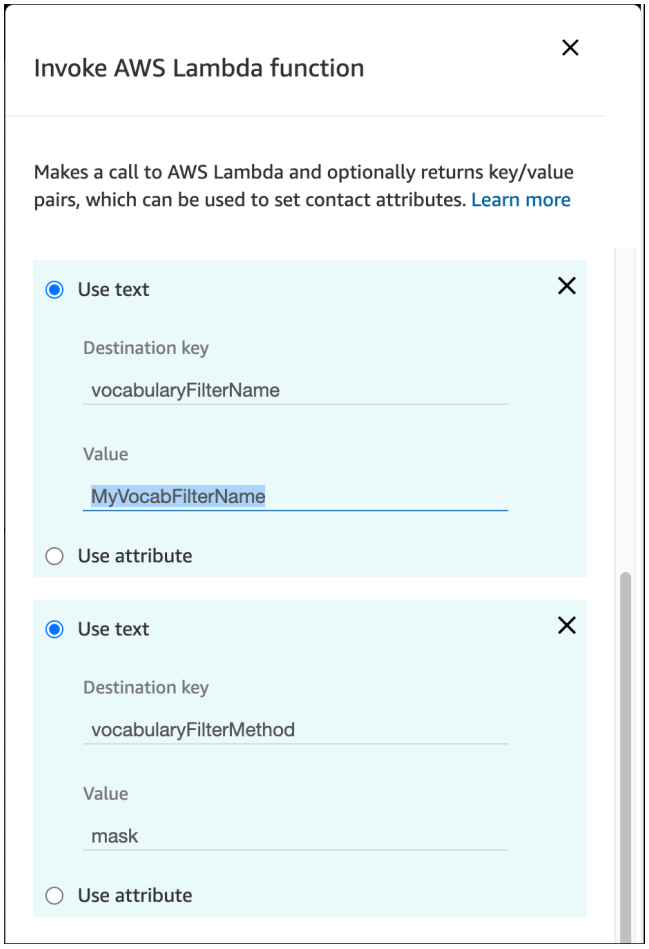
   To verify that you've selected the correct block, check that it ends with the string **kvsConsumerTrigger**.



2. If you have a custom vocabulary, add a `vocabularyName` parameter to this block with the name of your custom vocabulary that you created earlier.



3. If you have vocabulary filtering in place, add a `vocabularyFilterName` parameter to this block with the name of your vocabulary filter that you created earlier. Also, add a `vocabularyFilterMethod` parameter with one of the following values: `mask` or `remove`. `mask` replaces the term with asterisks. `remove` removes the term from the transcript. If no `vocabularyFilterMethod` parameter is provided, the behavior defaults to `mask`.



4. Save the block and contact flow, and then publish your contact flow.

Test This Example

To test this example, perform an inbound call and review the transcript that appears. The transcript should reflect your filters and custom vocabulary.

SEE ALSO:

Amazon Transcribe: Custom vocabularies

Amazon Transcribe: Vocabulary filtering

Amazon Transcribe's Transcription Languages Supported by Service Cloud Voice

Service Cloud Voice supports multiple languages.

Service Cloud Voice supports these languages for transcription.

- Chinese [Mainland China] (zh-CN)
- English [Australia] (en-AU)
- English [United Kingdom] (en-GB)
- English [United States] (en-US)

- French [France] (fr-FR)
- French [Canada] (fr-CA)
- German [Germany] (de-DE)
- Hindi [India] (hi-IN)
- Italian [Italy] (it-IT)
- Japanese [Japan] (ja-JP)
- Korean (ko-KR)
- Portuguese [Brazil] (pt-BR)
- Spanish [United States] (es-US)
- Thai [Thailand] (th-TH)

For information about Amazon Transcribe, see What is Amazon Transcribe?. For information about Amazon real-time transcription, see Streaming Transcription.

Transcribe Longer Calls with Contact Lens

Use Contact Lens to transcribe calls that are longer than 15 minutes. If you use Amazon Transcribe, only the first 15 minutes of a call are transcribed due to a Lambda function limitation. To trigger real-time transcription of a call with Contact Lens, enable Contact Lens and real-time transcription for Amazon Connect in your contact flow.

For a list of supported transcription languages with Contact Lens, see the Amazon documentation.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise** and **Unlimited** Editions

Available in: Sales Cloud, Service Cloud, and Government Cloud as an add-on license. Government Cloud is supported only on Service Cloud Voice with Amazon Connect and Service Cloud Voice with Partner Telephony from Amazon Connect.

Verify that Contact Lens in Amazon Connect is Enabled

Contact Lens must be enabled to transcribe calls. To verify that Contact Lens is enabled, you must have AWS IAM permissions with the ability to view Contact Lens details in Amazon Connect.

Create a Contact Flow with Contact Lens

To help you get started, use the packaged sample contact flows already set up with Contact Lens.

Enable Contact Lens and Transcription in an Existing Contact Flow

You can enable Contact Lens and real-time transcription in contact flows that you already created. To enable Contact Lens and transcription in an existing contact flow, add the recording block to the flow and enable real-time and post-call analytics.

Improve Contact Lens Transcription Accuracy with a Custom Vocabulary

Add one or more custom vocabularies to transcribe domain specific terminology correctly. Use it to fix the capitalization and spelling of domain-specific lingo and to mask text. To mask text, enter the term in the custom vocabulary file and display the term as ***.

Verify that Contact Lens in Amazon Connect is Enabled

Contact Lens must be enabled to transcribe calls. To verify that Contact Lens is enabled, you must have AWS IAM permissions with the ability to view Contact Lens details in Amazon Connect.

Ease of Implementation	Easy
Estimated Time to Implement	10 minutes

1. From the AWS Management Console, select **Amazon Connect** > **Analytics tools**.
2. Verify that Enable Contact Lens is enabled (selected).
3. Click **Save**.

Create a Contact Flow with Contact Lens

To help you get started, use the packaged sample contact flows already set up with Contact Lens.

Ease of Implementation	Medium
Estimated Time to Implement	45 minutes

Prerequisites

To run through this example, update your contact center to the latest version. To get the latest version of these contact flows, see Update Your Contact Center. If you don't update your contact center, you can have older versions of the contact flows with different names.

Create the Contact Flow

To create a contact flow with Contact Lens, start with these packaged contact flows that already has Contact Lens set up.

- Sample SCV Inbound Flow
- Sample SCV Outbound Flow With Transcription using Contact Lens
- Sample SCV Transfer Flow For Agent Transfers
- Sample SCV Transfer Flow For Queue Transfers

Enable Contact Lens and Transcription in an Existing Contact Flow

You can enable Contact Lens and real-time transcription in contact flows that you already created. To enable Contact Lens and transcription in an existing contact flow, add the recording block to the flow and enable real-time and post-call analytics.

Ease of Implementation	Medium
Estimated Time to Implement	45 minutes

Prerequisites

To run through this example, update your contact center to the latest version. To get the latest version of these contact flows, see Update Your Contact Center. If you don't update your contact center, you can have older versions of the contact flows with different names.

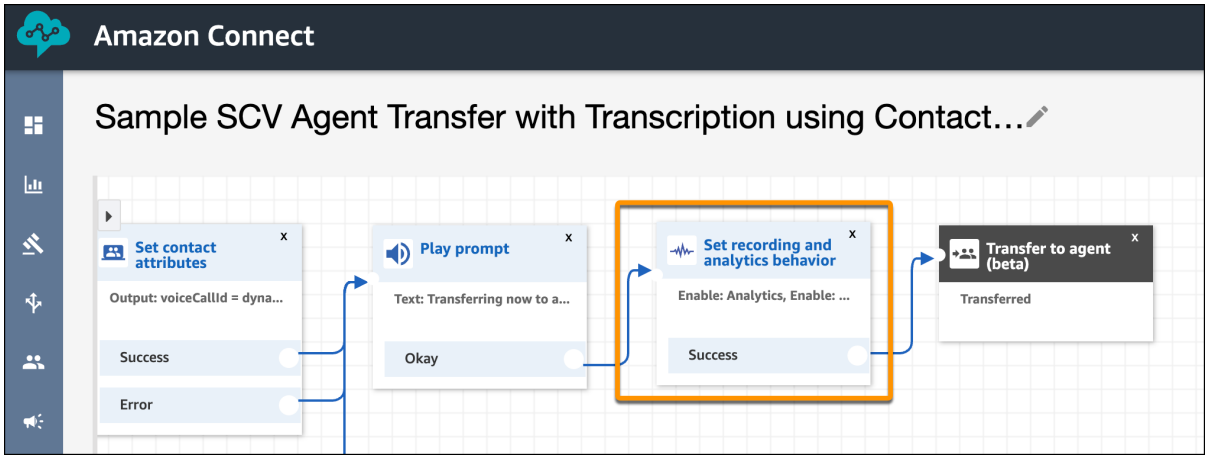
Add the Recording Block

Add the "Set recording and analytics behavior" recording block to the inbound, outbound, and transfer contact flows. In the block, select **Real-Time and post-call analytics in the block**.

Newer versions of the sample contact flows already have this recording block. If it's already added to your contact flows, you don't have to add it again.

Older versions of the sample contact flows use transcription with Amazon Transcribe. To use transcription with Contact Lens instead, remove the Invoke AWS Lambda function block that contains the kvsConsumerTrigger Lambda function and replace it with this recording block with the option enabled.

Tip: To learn how to set the Contact Lens options, view the settings for this block in the latest, packaged contact flows, which are available in GitHub.



SEE ALSO:

- kvsConsumerTrigger Lambda Function
- Amazon Connect Flows
- *Salesforce Help:* Update Your Contact Center
- *Amazon Documentation:* Contact Lens for Amazon Connect

Improve Contact Lens Transcription Accuracy with a Custom Vocabulary

Add one or more custom vocabularies to transcribe domain specific terminology correctly. Use it to fix the capitalization and spelling of domain-specific lingo and to mask text. To mask text, enter the term in the custom vocabulary file and display the term as ***.

Ease of Implementation	Medium
Estimated Time to Implement	30 minutes

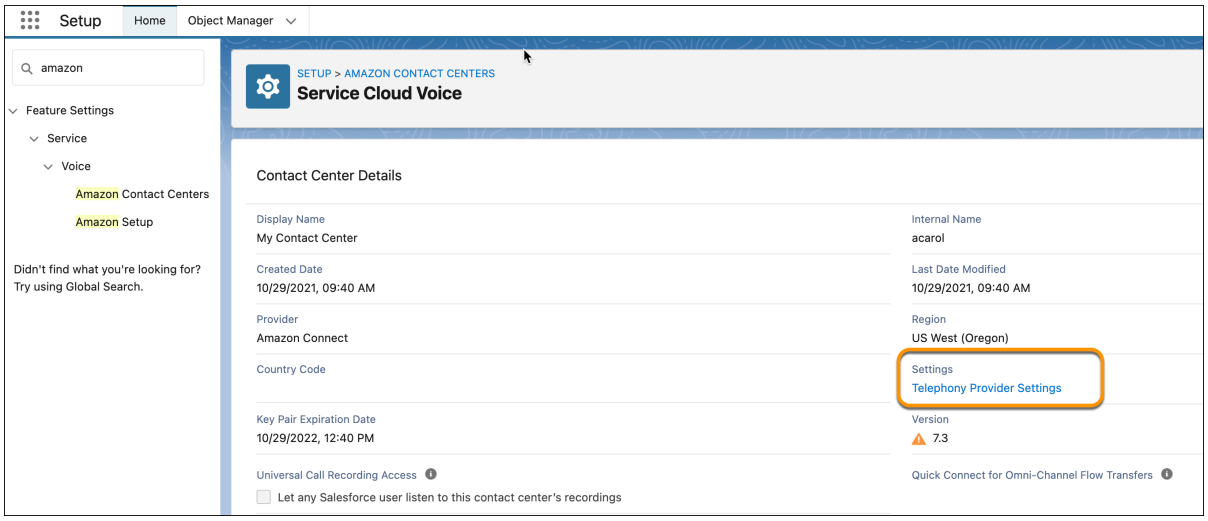
In this example, we show you how to add a custom vocabulary with Contact Lens.

Prerequisites

To run through this example, create your custom vocabulary file first. To learn how to create a file, review the text and links on the screen when asked to upload the file.

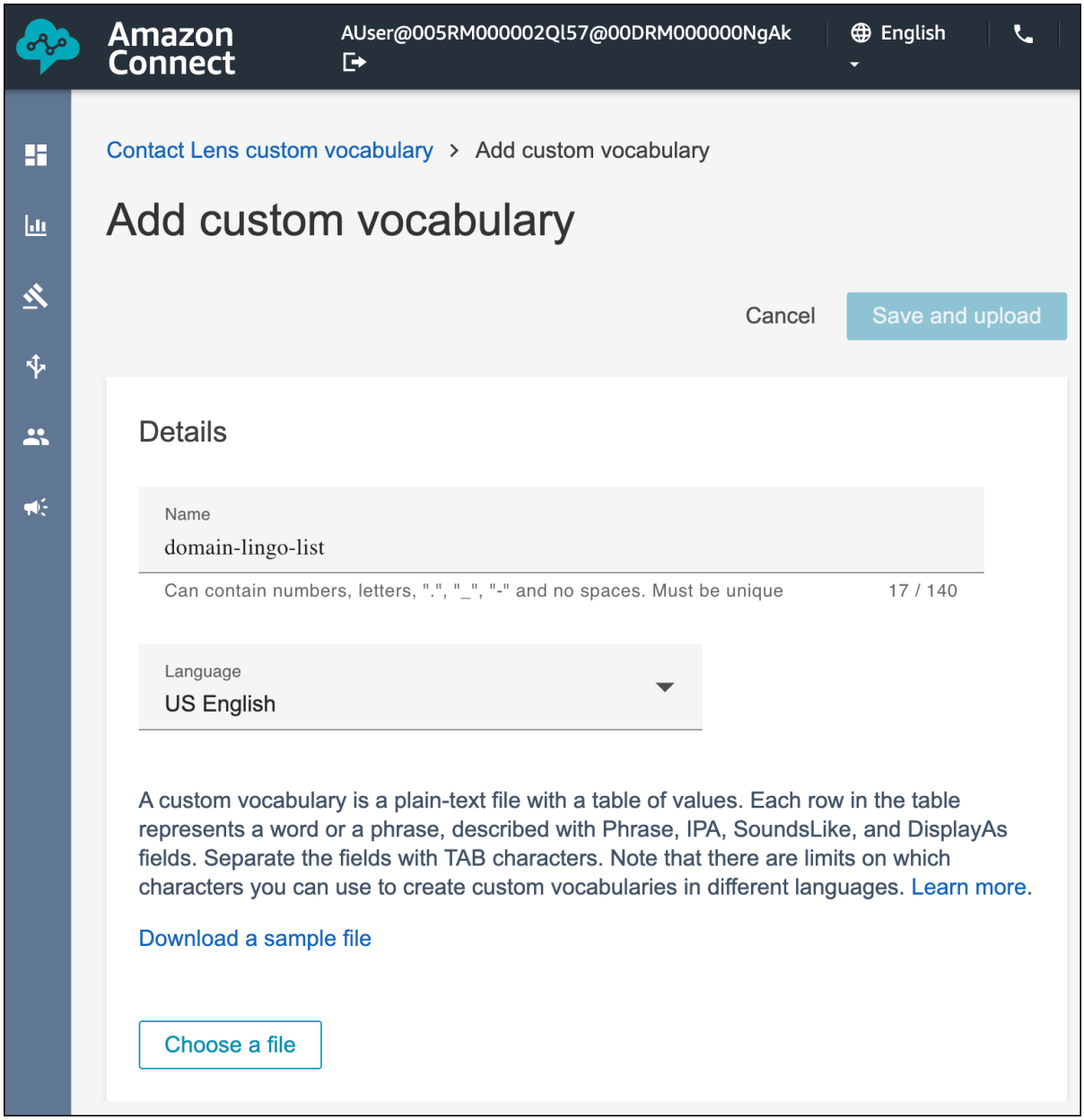
Step 1: Open Amazon Connect

To quickly access Amazon Connect, click **Telephony Provider Settings** in the Amazon contact center page.

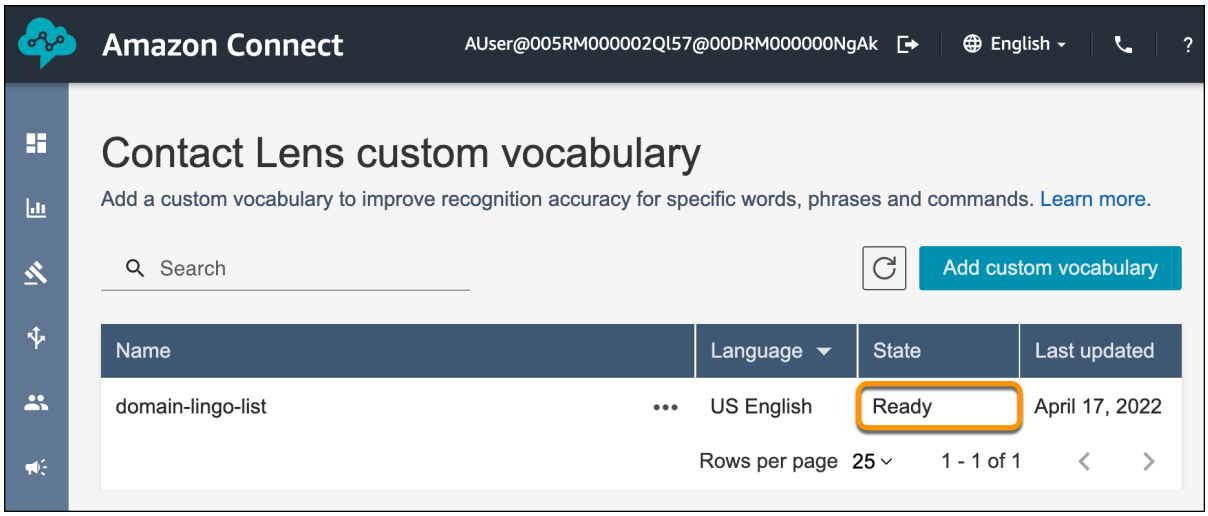


Step 2: Add a Custom Vocabulary

1. In Amazon Connect, click **Analytics** > **Custom vocabularies**.
2. In the Contact Lens Custom Vocabulary screen, click **Add custom vocabulary**.
3. Enter the name and language of the custom vocabulary.
4. Select **Choose a file**, and select the custom vocabulary file.



5. Click **Save and upload**.
6. Refresh the Amazon Connect page until the state shows Ready.
7. Click **...** to the right of the custom vocabulary, and select **Set as default**.



Test This Example

To test this example, perform an inbound call and review the transcript that appears. The transcript should reflect your custom vocabulary.

SEE ALSO:

Amazon Documentation: Create a Custom Vocabulary Using a Table

Link New or Existing Case to Voice Call Record

This example shows how to take a caller's inputs into the interactive voice response (IVR) system, pass them to Salesforce, and use them to create or link to Salesforce records. The example walks you through linking a case to a Voice Call record, but you can modify it to link to other Salesforce records, such as accounts and leads.

Ease of Implementation	Medium
Estimated Time to Implement	40 minutes

One of the advantages of Service Cloud Voice is that a Voice Call record is created to track every interaction. By capturing caller inputs from the IVR on the Voice Call record, you can use Salesforce automation tools, such as Process Builder, Flow, or APEX, to automatically create or link Salesforce records and display them to the rep for caller context. You can also report on caller selections.

In this example, we show you how to:

- Import a sample contact flow into Amazon Connect that updates two custom fields in Salesforce.
- Add a Salesforce flow that takes action in Salesforce based on contact flow values passed to the custom fields.

The first custom field is called `IVRType`. We use this field to store the calls IVR Type or selection (for example: Press 1 for a new case, Press 2 for an existing case). The second custom field, `IVRInput`, is used to store a customer input such as case subject or case number. The Salesforce flow creates a new case or finds an existing case and links it to the voice call record to add context for the rep.

You can modify this example to support your business processes. For example, create or find a lead instead of a case.

Prerequisites

To run through this example:

- You must be an AWS admin with the ability to import contact flows. To learn more, see Import/export contact flows in the Amazon Connect Administrator Guide.
- You must be a Salesforce admin with the ability to install packages. To learn more, see Install a package in Salesforce Help.

Step 1: Install the Example Package

Install the SCVCB - Voice Actions (New or Existing Case) package in Salesforce. This link opens a Salesforce login page if you're not already logged in. Log in to the org where you want to install the package. This package contains the following items.

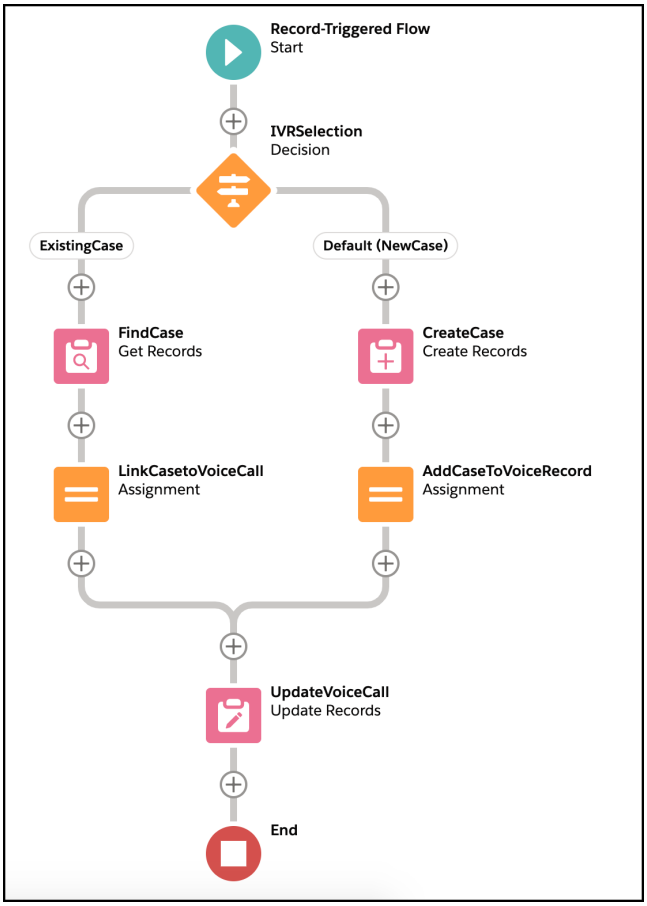
- Custom field: `IVRType__c`
- Custom field: `IVRInput__c`
- Record-triggered flow: "SCVCB - Voice Actions"

Step 2: Add Fields to the Voice Call Page

1. From Setup, open the **Object Manager**.
2. Search for and select the**Voice Call** object.
3. Click **Page Layouts**.
4. Select a page layout to update and add the two custom fields: `IVRType` and `IVRInput`.
5. Save your work.

Step 3: Activate the Flow

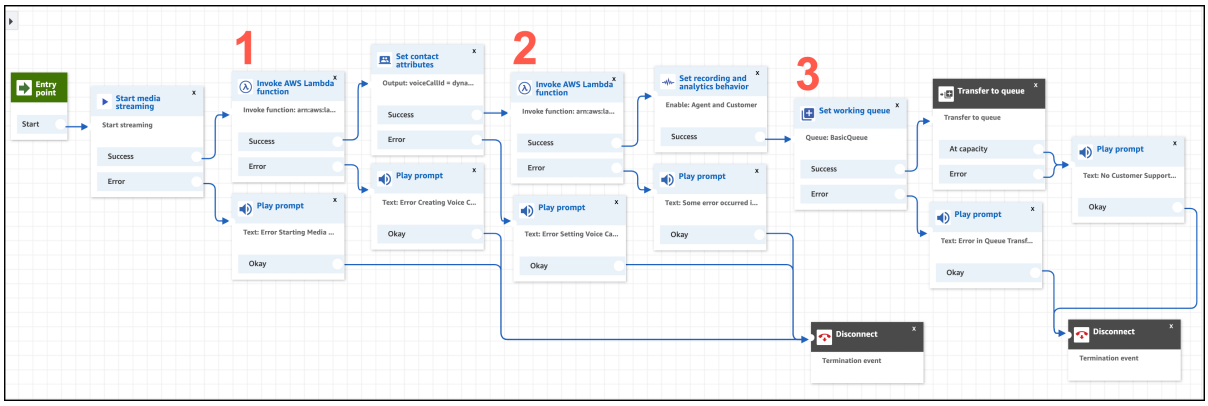
1. From Setup, enter *`Flows`* in the Quick Find box, then select **Flows** under Process Automation.
2. Open the "SCVB - Voice Actions" flow.



3. Click **Activate**. When active, the flow makes the following changes.
   - When `IVRType = ExistingCase`, the flow finds the case that matches the case number entered in the IVR and links it to the Voice Call record.
   - For any other `IVRType` value, a new case is created and linked to the Voice Call record.
4. (Optional) Modify the flow to add more paths by adding conditions and actions to the Decision Element.

Step 4: Import the "SCVCB - RouteToAgent" Contact Flow

The "SCVCB - RouteToAgent" contact flow is similar to the "Sample SCV Inbound Flow" contact flow that Salesforce already provides. However, we've removed several of the play prompts so it can be reused with future flows. This flow creates the VoiceCall record, enables recording, and enables transcription.



1. Download the "SCVCB - RouteToAgent" contact flow from our GitHub repo.

   Note: From GitHub, right-click the **Raw** button to download or save the linked file. After downloading, you must remove the `.txt` extension (so that there is no file extension) before attempting to upload it to Amazon Connect. Alternatively, clone our GitHub repository to download all examples at once. This contact flow resides in the `Recipes/LinkCasetoVoiceCall` folder.

2. Log into your Amazon Connect instance.
   a. From Setup in your Salesforce org, enter **Contact Centers** in the Quick Find box, then select **Contact Centers** under Voice.
   b. In the row containing your contact center, click **Telephony Provider Settings** to open Amazon Connect.
3. From Amazon Connect, select **Routing** > **Contact flows**.
4. Click **Create contact flow**.
5. On the far right of the contact flow page, click the disclosure triangle button and select **Import Flow (beta)**.
6. Click **Select** and browse to the "SCVCB - RouteToAgent" contact flow you downloaded.
7. In the first "Invoke AWS Lambda function" block (item 1 in screenshot), click the block to open the options and replace the function with your **xxxx-InvokeTelephonyIntegrationApiFunction** Lambda function. Click **Save**.

   Note: To learn what this function does, see InvokeTelephonyIntegrationApiFunction Lambda Function on page 145.

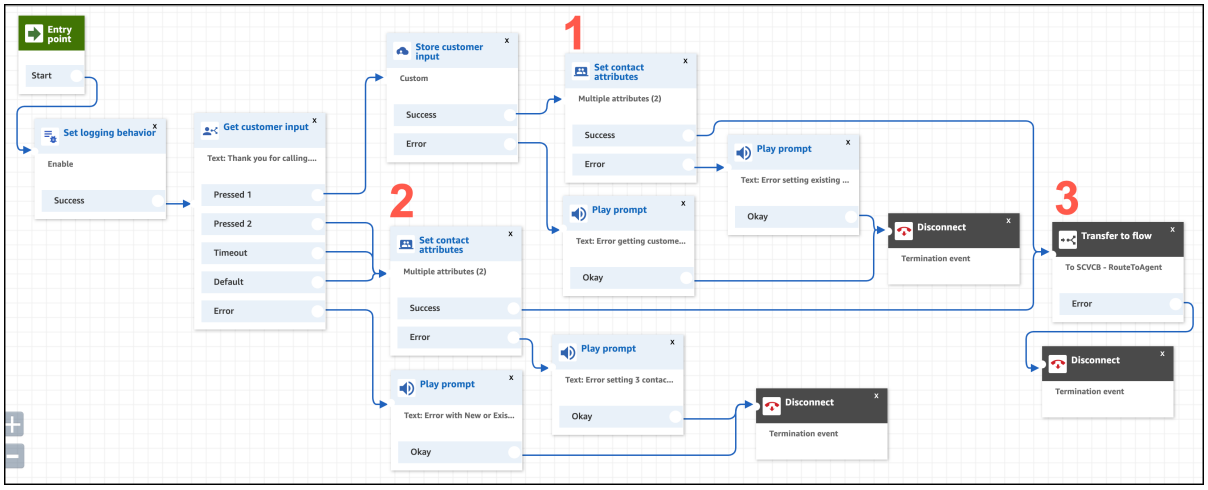
8. In the second "Invoke AWS Lambda function" block (item 2 in screenshot), click the block to open the options and replace the function with your **xxxx-kvsConsumerTrigger** Lambda function. Click **Save**.

   Note: To learn what this function does, see kvsConsumerTrigger Lambda Function on page 163.

9. In the "Set working queue" block (item 3 in screenshot), click the block to open the options and replace the queue with your queue (for example, "BasicQueue").
10. Click **Publish** to save and publish the flow.

Step 5: Import the "SCVCB - New or Existing Case" Contact Flow

The "SCVCB - New or Existing Case" contact flow asks the customer if they're calling about an existing case or if they want to create a new case.



In this contact flow, we map contact attributes to custom fields on the Voice Call record by using the naming convention "sfdc-{DeveloperName}". We use the developer name for the custom field in Salesforce, including the "__c" suffix used for all custom fields. When the InvokeTelephonyIntegrationApiFunction Lambda function creates a Voice Call record, it looks for all contact attributes with this format and maps them to the appropriate Salesforce field.

1. Download the "SCVCB - New or Existing Case" contact flow from our GitHub repo.

   Note: From GitHub, right-click the **Raw** button to download or save the linked file. After downloading, you must remove the `.txt` extension (so that there is no file extension) before attempting to upload it to Amazon Connect. Alternatively, clone our GitHub repository to download all examples at once. This contact flow resides in the `Recipes/LinkCasetoVoiceCall` folder.

2. From Amazon Connect, select **Routing** > **Contact flows**.
3. Click **Create contact flow**.
4. On the far right of the contact flow page, click the disclosure triangle button and select **Import Flow (beta)**.
5. Click **Select** and browse to the "SCVCB - New or Existing Case" contact flow you downloaded.
6. In the upper "Set contact attributes" block (item 1 in screenshot), click the block to view the options. Note that two contact attributes are set: sfdc-IVRType__c is set to *`ExistingCase`*, and sfdc-IVRInput__c is set to the four-digit input from the caller.
7. In the lower "Set contact attributes" block (item 2 in screenshot), click the block to view the options. Note that two contact attributes are set: sfdc-IVRType__c is set to *`NewCase`*, and sfdc-IVRInput__c is set to *`New Voice Case`*. You can change this value to customize the case subject based on your business process.
8. In the "Transfer to flow" block (item 3 in screenshot), click the block to view the options. Remove the existing flow and replace it with "SCVCB - RouteToAgent", which you published earlier.
9. Click **Publish**.

Test This Example

Here's how to test this example.

1. In Amazon Connect, from the Routing menu, select **Phone numbers**.
2. Click the phone number you use for testing.
3. Change the **Contact Flow / IVR** value to *`SCVCB - New Or Existing Case`*.

4. Click **Save**.
5. In Salesforce, log in to Omni-Channel so you are ready to accept calls.
6. Dial the number and test the new contact flow.
7. Accept the call in Omni-Channel:
   a. If you selected New Case, you see that a new case has been created and linked to the Voice record in the Related Record field.
   b. If you selected Existing Case, you see the case linked to the Voice record in the Related Record field.

Note: This example assumes a four-digit case number. If your case numbers use a different number of digits, modify the example.

1. In the "SCVCB - New Or Existing Case" contact flow, open the "Store customer contact" block and change the Custom Maximum Digits value.
2. In the "SCVCB - Voice Actions" flow, update the Get8DigitCaseNumber formula to append the necessary number of zeros to match the Salesforce case number.

Give Customers the Option to Transfer to SMS in the Interactive Voice Response (IVR)

This example builds on the methods created in the "Link New or Existing Case to Voice Call Record" recipe by adding another IVR option to switch to SMS. When a customer selects the SMS option, a Voice Call record is created and a flow sends an automated outbound SMS message.

Ease of Implementation	Medium
Estimated Time to Implement	1 hour

Prerequisites

To run through this example:

- You must have at least one Digital Engagement license to enable SMS capabilities.
- You must have a working SMS channel in Salesforce.
- You must complete the previous example: Link New or Existing Case to Voice Call Record on page 17.

To learn more about SMS Text Messaging with Salesforce, see Set Up SMS Text Messaging.

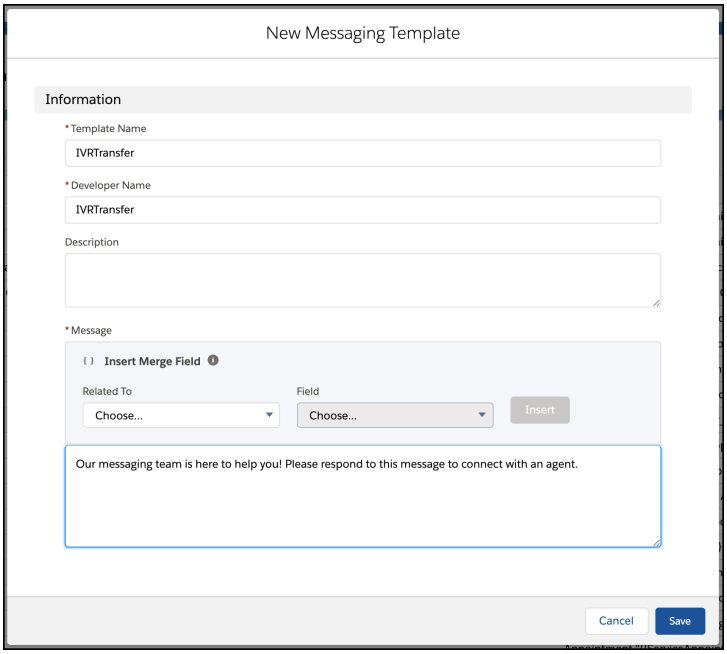
Step 1: Create an SMS Messaging Template

1. From Setup, enter *Messaging Templates* in the Quick Find box, then select **Messaging Templates**.
2. Click **New**.
3. Complete the fields as follows.

Field Name	Value
Template Name	*IVRTransfer*
Developer Name	*IVRTransfer*

Field Name	Value
Message	Enter your message. For example: `Our messaging team is here to help you! Please respond to this message to connect with an agent.`

4. Click **Save**.

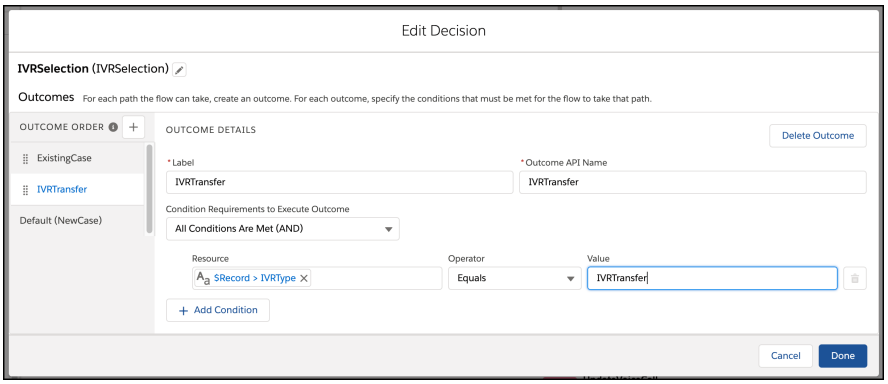


Step 2: Update Your "SCVB - Voice Actions" Flow

1. Copy the **Developer Name** for your Messaging channel. You'll use this exact value in the flow that you're going to update.
   a. From Setup, enter `Messaging Settings` in the Quick Find box, then select **Messaging Settings**.
   b. Click **Edit** for your desired SMS channel.
   c. Record the value in the **Developer Name** field. For example, `Text_US_15551234567`. This value is used later for the Messaging Channel Unique Name.
2. From Setup, enter `Flows` in the Quick Find box, then select **Flows** under **Process Automation**.
3. Open the "SCVB - Voice Actions" flow.
4. Add an outcome to the "IVRSelection" decision element.
   a. Click **Edit Element** on the decision element.
   b. Click the **+** next to **Outcome Order**.
   c. Complete the fields as follows.

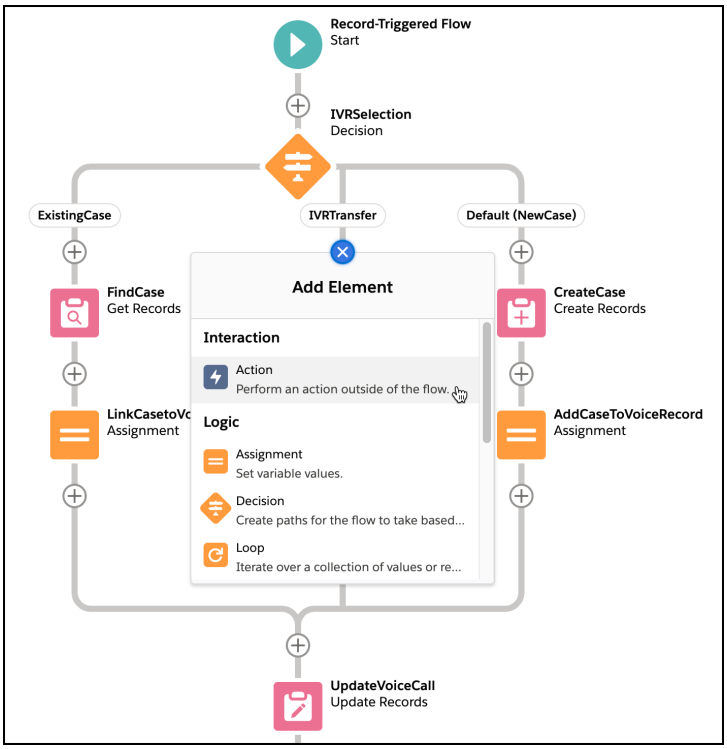
Field Name	Value
Label	`IVRTransfer`
Condition Requirements to Execute Outcome	**All Conditions are Are Met (AND)**
Resource	**$Record.IVRType__c** IVRType__c is a custom field for the VoiceCall Salesforce object.
Operator	**Equals**
Value	`IVRTransfer` This value matches the value set in the Contact Flow.

d. Click **Done** to save the new outcome.



5. On the "IVRTransfer" branch, click the **+** to add an element action.

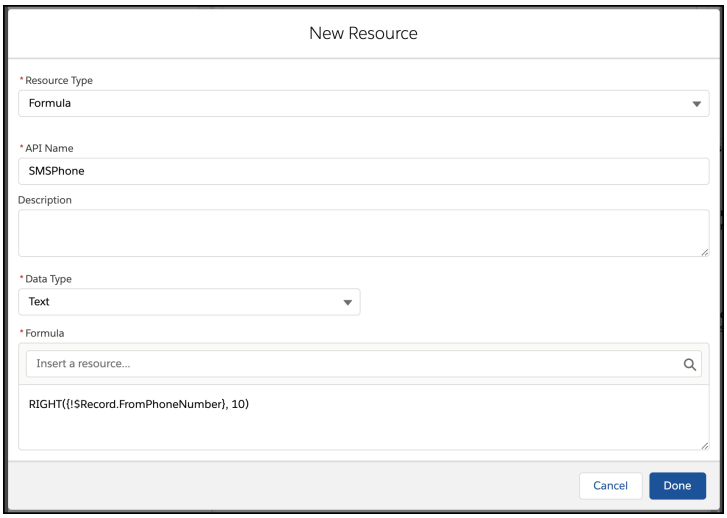
a. On the Add Element screen, select **Action**.



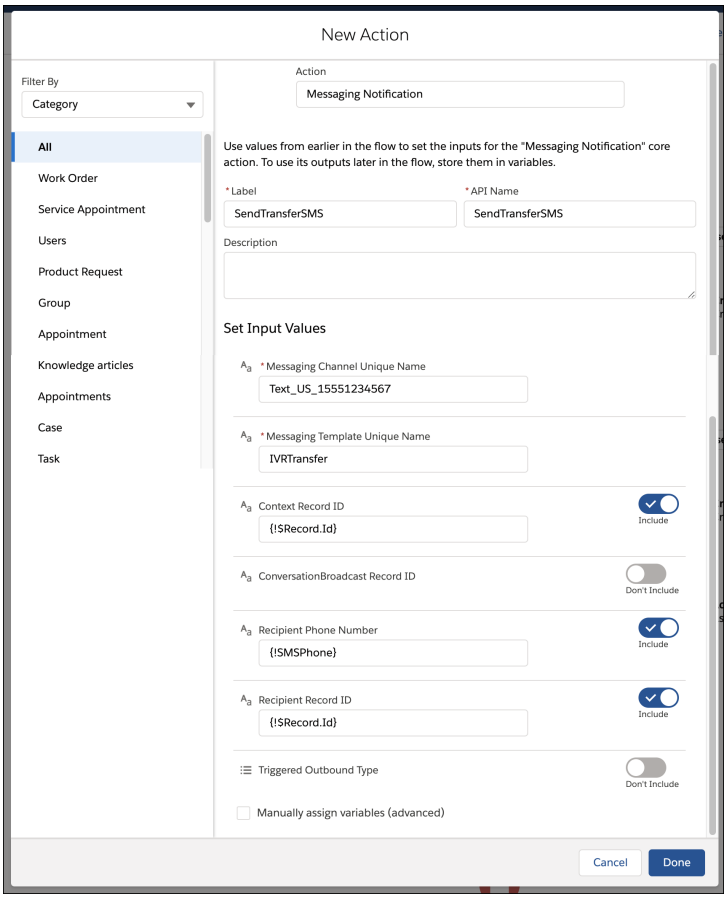
b. From the New Action page, search for and select **Messaging Notification**.

c. Complete the fields as follows.

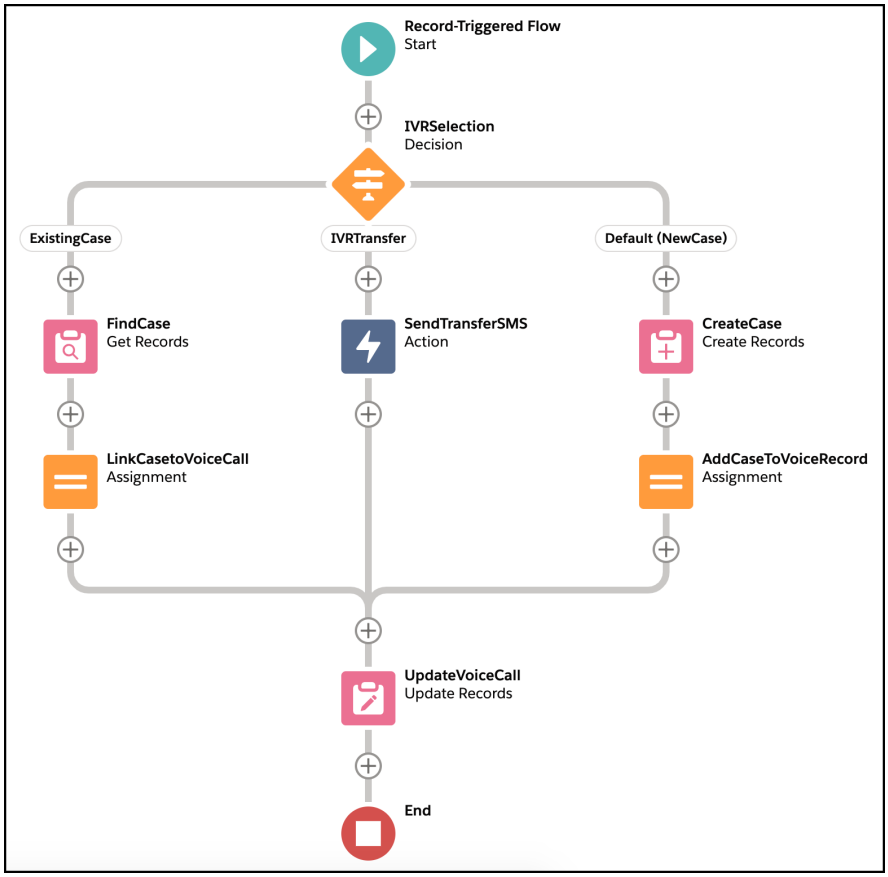
Field Name	Value
Label	`SendTransferSMS`
Messaging Channel Unique Name	Enter the Developer Name for your messaging channel that you recorded earlier. For example, `Text_US_15551234567`.
Messaging Template Unique Name	Select the template you previously created (`IVRTransfer`).
Context Record ID	Change the toggle to "Include" and enter `{!$Record.Id}`.
Recipient Phone Number	Change the toggle to "Include" and select **New Resource** for the field value. Complete the fields as follows, and then click **Done**. • Resource Type: **Formula** • API Name: `SMSPhone` • Data Type: **Text** • Formula: Enter `RIGHT({!$Record.FromPhoneNumber}, 10)`. In other markets, such as in the UK, if the caller number is already in E164 format, use `{!$Record.FromPhoneNumber}` to ensure the outbound SMS is sent. Note: Amazon Connect passes in the caller's phone number in E164 format. We use a formula to convert the number to the necessary format. (This example assumes US-based phone numbers. For example, +155512234567 is converted to 5551234567.)

Field Name	Value
	
Recipient Record ID	Change the toggle to "Include" and enter `{!$Record.Id}`.

d. Click **Done** to save the new action.



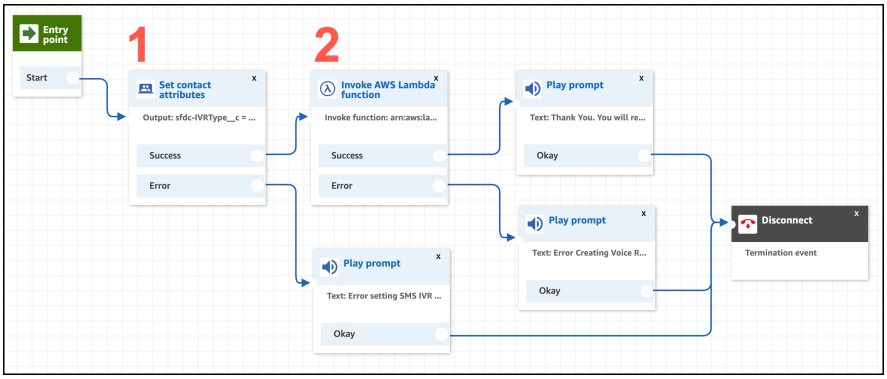
6. On your flow with the new outcome, click **Save As**, and then click **Save**.



7. Click **Activate** to activate your flow.

Step 3: Import the "SCVCB - SMS Transfer" Contact Flow

When the "SCVCB - SMS Transfer" contact flow is called, it creates a voice call record with the IVRSelection equal to SMSTransfer.



1. Download the "SCVCB - SMS Transfer" contact flow from our GitHub repo.

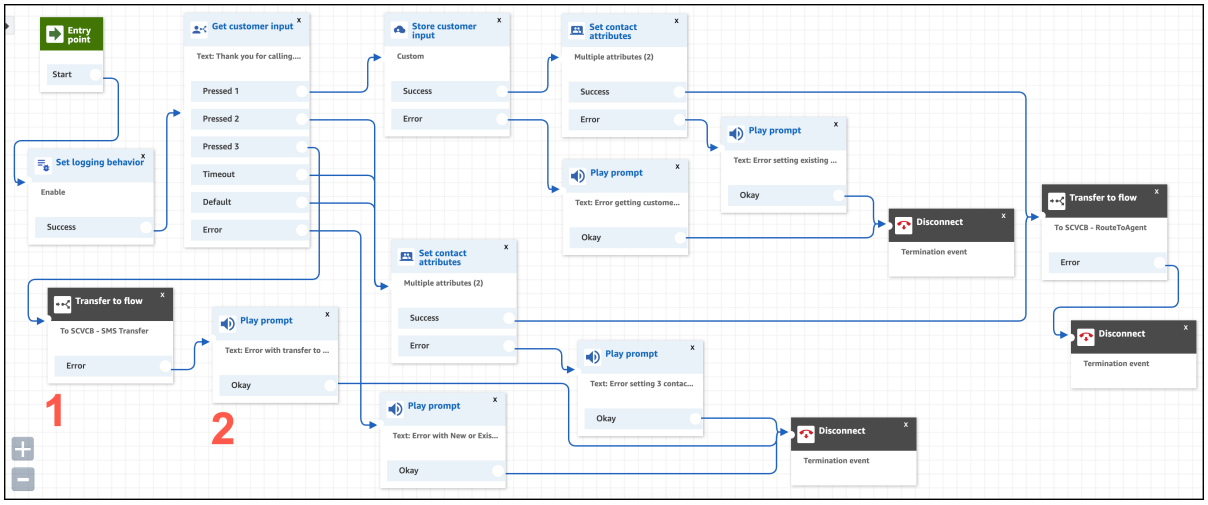
   Note: From GitHub, right-click the **Raw** button to download or save the linked file. After downloading, you must remove the `.txt` extension (so that there is no file extension) before attempting to upload it to Amazon Connect. Alternatively, clone

our GitHub repository to download all examples at once. This contact flow resides in the `Recipes/TransfertoSMSinIVR` folder.

2. From Amazon Connect, select **Routing** > **Contact flows**.
3. Click **Create contact flow**.
4. On the far right of the contact flow page, click the disclosure triangle button and select **Import Flow (beta)**.
5. Click **Select** and browse to the "SCVCB - SMS Transfer" contact flow you downloaded.
6. In the "Set contact attributes" block (item 1 in screenshot), ensure that the Destination attribute matches the API name of the IVRSelection field you have on the Voice Call record. For example, *`sfdc-IVRType__c`*.
7. Update the function in the "Invoke AWS Lambda function" block. Select **xxxx-InvokeTelephonyIntegrationApiFunction**.
8. In the "Invoke AWS Lambda function" block (item 2 in screenshot), click the block to open the options and replace the function with your **xxxx-InvokeTelephonyIntegrationApiFunction** Lambda function. Click **Save**.
9. Click **Publish**.

Step 4: Add SMS to the "SCVCB - New Or Existing Case" Contact Flow

Update the "SCVCB - New Or Existing Case" contact flow from the previous example (Link New or Existing Case to Voice Call Record on page 17) to include the SMS option.



1. Open the contact flow "SCVCB - New Or Existing Case".
2. Update the "Get customer input" block.
   a. Click the "Get customer input" block.
   b. To update the text-to-speech entry to include SMS as option 3, change the text to: *`Thank you for calling. Press 1 if you are calling about an existing case. Press 2 if you are calling about a new case. Or press 3 to Message with our team.`*
   c. Scroll to the bottom of the block and click **Add another condition**.
   d. For the new option, enter *3*.
   e. Click **Save**.

3. Add a new "Transfer to flow" block (item 1 in screenshot).
   a. From the left panel, in the "Terminate/Transfer" section, drag the "Transfer to flow" block onto the contact flow.
   b. Open the block and select **SCVB - SMS Transfer**, which is the contact flow you previously created.

      Tip: If you don't see this flow, go back to that contact flow and click **Publish**. Only published flows appear in the list.

   c. Click **Save**.
4. Connect Option 3 of the "Get customer input" block to the left side of this "Transfer to flow" block.
5. Add an error message to the "Transfer to flow" block (item 2 in screenshot).
   a. From the left panel, in the "Interact" section, drag the "Play prompt" block to the right of the "Transfer to flow" block.
   b. Open the "Play prompt" block and select **Text-to-speech or chat text**.
   c. Enter this text: `Error with transfer to SMS flow.`.
   d. Click **Save**.
6. Connect the output of the "Play prompt" block to one of the closest "Disconnect" blocks.
7. Click **Publish**.

Test This Example

To test this example, call your number. You should now hear an option 3 for SMS transfer. Press 3 on your phone. The following events should occur:

1. A Voice Call record is created with IVRSelection equal to SMSTransfer.
2. You receive an SMS on your phone with the message that you configured in your template. Reply to the message to initiate a messaging session. Make sure your rep is logged in with the appropriate Omni-Channel presence status for Messaging.

If you don't received an SMS, check the Messaging logs.

Enable the Option to Request a Callback

This example shows how to set up a rep-first callback in your inbound contact flow. The contact flow defines the interactive voice response (IVR) and prompts the customer to leave a phone number for a callback.

Ease of Implementation	Medium
Estimated Time to Implement	30 minutes

The IVR looks at how long the last person in the queue has been waiting. The wait time is five minutes by default. If the wait time is greater than the specified time, the IVR asks if the caller wants to continue waiting or to request a callback.

If the caller opts for the callback, the IVR asks the caller to enter a phone number.

When a rep becomes available, the callback request is pushed to the rep in the Omni-Channel utility. If the rep accepts the callback request, the phone number is automatically dialed.

Prerequisites

If your contact center was created before the Spring '21 release, download the updated Sample SCV Inbound Flow contact flow from the Service Cloud Voice repo on Github. Import it into Amazon Connect.

If your contact center was created after the Spring '21 release, then it already contains the Sample SCV Inbound Flow, so you can skip the first step.

To implement this use case, you need either the AWS root user or AWS administrator credentials.

Step 1: Import the Contact Flow into Amazon Connect

Note: This step is only necessary if your contact center was created before the Spring '21 release. In contact centers created in Spring '21 and later, this contact flow is automatically installed, so you can skip this step.

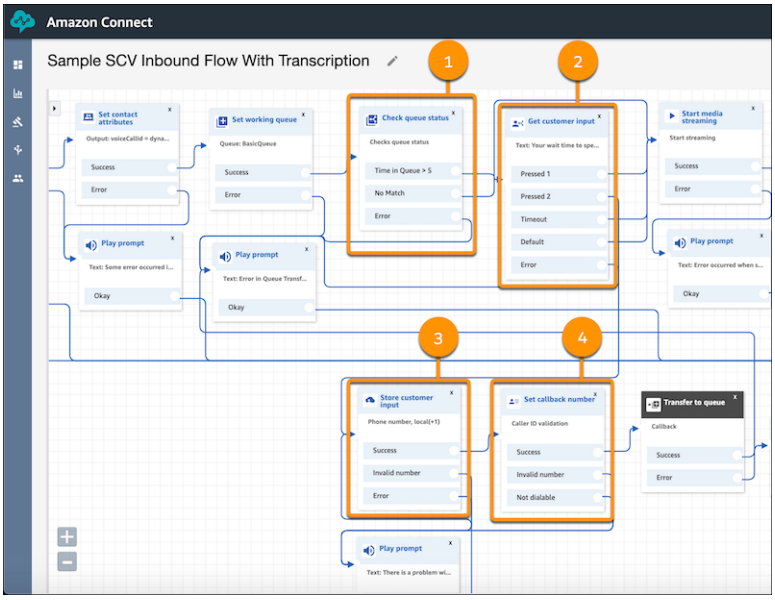
1. Log in to your Amazon Connect instance.
2. Hover over **Routing**, then click **Contact flows**.
3. Click **Create contact flow**.
4. Click the down arrow next to **Save** in the top-right corner.
5. Click **Import flow (beta)**.
6. Click **Select**. In the file browser, select the `Sample SCV Inbound Flow` file from your downloads folder.
7. Click **Import**. The contact flow is imported and opens in a window where you can edit it. Next, modify the contact attribute blocks.

Step 2: Modify the Contact Flow Blocks

1. Log in to your Amazon Connect instance.
2. Hover over **Routing**, then click **Contact flows**.
3. Select the **Sample SCV Inbound Flow** contact flow.
4. Optionally modify the following blocks. To modify a block, select the block, edit the settings, and save your work.
   a. The `Check queue status` block tells the system to check how long the last person in the queue has been waiting (item 1 in screenshot). You can modify the time; for example, you can make it shorter or longer.
   b. The `Get customer input` block tells the system to play a prompt. The prompt asks the customer to press 1 to remain in the queue, or to press 2 to request a callback (item 2 in screenshot). You can modify the prompt's wording.
   c. The `Store customer input` block asks the customer to enter a phone number for the callback (item 3 in screenshot).
   d. The `Transfer to queue` block adds the callback request to the callback queue so callbacks are pushed to available reps (item 4 in screenshot). There are three settings that you can customize:
      - *Initial delay:* The initial delay is the minimum amount of time after the first call ends before the system attempts the callback. By default, the delay is 60 seconds.
      - *Max number of retries:* If the customer doesn't answer when the rep calls back, the callback request goes back in the queue and the system attempts to make another callback.

        Note: If a callback gets voicemail, Amazon Connect treats it as a completed callback and doesn't retry the call.

      - *Minimum time between attempts:* If the customer doesn't answer the phone, this setting is how long the system waits before trying again.



Step 3: Publish the Contact Flow

1. To make the contact flow active, click **Publish**.

Test This Example

To test this example, you need two phones.

1. Log in to Salesforce, open the Omni-Channel utility, and log in to Voice.
2. Call your contact center with one phone. The call appears in the Call Controls tab in the Omni-Channel utility.
3. Don't accept the call, and make sure that another rep doesn't accept it. Stay on the line and wait in the queue for at least five minutes, or for however long you specified in the contact attribute block.
4. Use the second phone to call your contact center. If the first caller is waiting for at least five minutes, then callback prompt is played for the second caller.
5. To select the callback option, press 2 and enter your phone number, and then hang up.

The callback request is pushed to the next available rep. When the rep accepts the call, the outbound call is dialed and your phone rings.

SEE ALSO:

Amazon Connect Administrator Guide: Set up queued callback

Amazon Connect Administrator Guide: About queued callbacks in metrics

Contact Record Sync with Amazon Connect

By default, contact record (previously called contact trace record or CTR) data is automatically stored in Voice Call records, allowing your Amazon Connect instance to stay in sync with Salesforce.

Retry Contact Record Sync for Voice Call Records

Amazon Connect contact record (previously called contact trace record or CTR) data is stored automatically in Voice Call records through data synchronization. Sometimes this sync doesn't occur, and it isn't always possible to access contact record data in Amazon Connect after the call. Back up contact record data to a separate Amazon S3 bucket, then check for Voice Call records that don't have contact record data, and then resync the contact record data to your org using the backup data.

Ease of Implementation	Advanced
Estimated Time to Implement	2–3 hours

Contact Trace Records are generated on Amazon for each call. This data contains details about the call, including caller and rep information, call statistics, and queue information. The CTRDataSyncFunction on page 156 Lambda function automatically runs for each call to store this data in a Voice Call record. CTRs aren't retained by Amazon, and if the CTR sync process fails for any reason, you can lose important information about the call.

This solution provides a way to reattempt a sync of the CTR data by first backing up CTR data in a separate Amazon S3 bucket, then checking for any sync failures, and then re-uploading CTR data whenever necessary.

Prerequisites

To run this example:

1. Have your AWS root user or AWS administrator credentials ready.
2. Be familiar with Amazon S3 buckets. See Amazon's documentation: Creating and configuring an S3 bucket.
3. Be familiar with modifying Amazon Connect Lambda functions. See Amazon's documentation: AWS Lambda Developer Guide.
4. Configure the OAuth support for the InvokeSalesforceRestApiFunction Lambda on page 136 function. This step is required to support the `getAccessToken` function.

Step 1: Create an Amazon S3 Bucket

Create an Amazon S3 bucket on the same instance as your contact center. This bucket must be accessible within the region where your Amazon instance is configured. When creating your S3 bucket, make sure that you set the right access control policies for the bucket so it's not accessible outside of your instance.

1. Under **General Configuration > Bucket Type**, select **General Purpose**.
2. Under **Object Ownership**, select **ACLs disabled (recommended)**.

3. Under **Block Public Access settings for this bucket**, select **Block all public access**.
4. Under **Bucket Versioning**, select **Enable**.
5. Under **Default Encryption > Encryption Type**, select **Server-side encryption with Amazon S3 managed keys (SSE-S3)**.
6. Under **Default Encryption > Bucket Key**, select **Enable**.

See Amazon's documentation: Creating and configuring an S3 bucket.

When using the code provided by your S3 bucket service in AWS, replace the resource account ID and ARN with your account ID and S3 bucket ARN, respectively. Delete all code comments before you save your changes.

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "PutObjectAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<your account id>:root"    // REPLACE THIS WITH YOUR
 ACCOUNT ID
            },
            "Action": "s3:*",
            "Resource": "<your S3 Bucket ARN>"    // REPLACE THIS WITH YOUR S3 BUCKET ARN

        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "transcribe.amazonaws.com"
            },
            "Action": "s3:*",
           "Resource": "<your S3 Bucket ARN>"     // REPLACE THIS WITH YOUR S3 BUCKET ARN

        }
    ]
}
```

Step 2: Set Data Expiration

After you store the CTR data in the S3 bucket, it's a good idea to purge them periodically. To remove old data, specify the object expiration for the S3 bucket and define how long to wait before the files are deleted. See Amazon's documentation: Expiring Objects and Managing your storage lifecycle.

Step 3: Save Contract Trace Records in S3

The CTRDataSyncFunction on page 156 is configured to receive events from the Kinesis stream. These events are generated whenever a CTR is generated. Modify this Lambda function to read those events and write them to the S3 bucket as they're generated. Update the `handler.js` file so that it contains a new function, `writeCTRToS3`, that writes data to the new bucket. Then, update the existing code to call this new function.

When modifying existing code, it's crucial to add code in the correct context. Inserting code snippets out of context or in the wrong order breaks the code.

In this example, the text in bold is where the new `writeCTRToS3` function must be added to the `handler.js` file. Delete all code comments before you save your changes.

```
const aws = require("aws-sdk");
const SCVLoggingUtil = require("./SCVLoggingUtil");
const s3 = new aws.S3();

const lambda = new aws.Lambda();
const utils = require("./utils");

function updateVoiceCallRecord(voiceCall) {
  SCVLoggingUtil.info({
    message: "CTR/updateVoiceCallRecord Request created",
    context: { contactId: voiceCall.contactId },
  });
  const payload = {
    Details: {
      Parameters: {
        methodName: "updateVoiceCall",
        fieldValues: voiceCall.fields,
        contactId: voiceCall.contactId,
      },
    },
  };

  const params = {
    FunctionName: process.env.INVOKE_TELEPHONY_INTEGRATION_API_ARN,
    Payload: JSON.stringify(payload),
  };

  return lambda.invoke(params).promise();
}

function shouldProcessCtr(ctrRecord) {
  return (
    ["INBOUND", "OUTBOUND", "TRANSFER", "CALLBACK", "API"].includes(
      ctrRecord.InitiationMethod
    ) &&
    ctrRecord.ContactId &&
   // if the call is a voicemail, no need to process it because the packager lambda updates
 the voicecall record
    !(
      ctrRecord.Attributes &&
      ctrRecord.Attributes.vm_flag &&
      ctrRecord.Recordings
    )
  );
}

exports.handler = async (event) => {
  const promises = [];
  SCVLoggingUtil.debug({
    message: "CTRDataSync event received",
    context: event,
  });
```

```
  event.Records.forEach((record) => {
    const ctr = utils.parseData(record.kinesis.data);
    if (shouldProcessCtr(ctr)) {
      const voiceCall = utils.transformCTR(ctr);
      SCVLoggingUtil.debug({
        category: "ctrDataSync.handler",
        message: "Transformed CTR to voice call",
        context: voiceCall,
      });
      const updatePromise = updateVoiceCallRecord(voiceCall);

      promises.push(updatePromise);

      updatePromise.then((response) => {
        SCVLoggingUtil.info({
          message: "updateVoiceCallRecord response",
          context: response,
        });
      });

      writeCTRToS3(ctr);

    } else {
      SCVLoggingUtil.error({
        message: "Encountered Non supported CTR Events: failing fast",
        context: {},
      });
    }
  });

  return Promise.all(promises);
};


function writeCTRToS3(ctr) {
    var params = {
        Bucket : "<your S3 bucket Name>",   // REPLACE THIS WITH YOUR S3 BUCKET NAME
        Key: ctr.ContactId,
        Body: Buffer.from(JSON.stringify(ctr)),
    }

    s3.putObject(params, function(err, data) {
        if (err) {
            console.log("Error uploading CTR to S3: " + JSON.stringify(err));
        } else {
            console.log("Successfully uploaded CTR to S3: " + JSON.stringify(data));
        }
    });
}


exports.shouldProcessCtr = shouldProcessCtr;
```

There's one object in S3 for every CTR generated. The object is named with the `ContactId` of the CTR, so it's easy to look up.

In the CTRDataSyncFunction associated with your contact center, create a new inline policy for the IAM role and delete any comments in the code before saving:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:ListBucket",
                "s3:PutObjectAcl"
            ],
            "Resource": "<your S3 Bucket ARN>/*" // REPLACE THIS WITH YOUR S3 BUCKET ARN
        }
    ]
}
```

Important: Creating a new inline policy for the IAM role is required for the CTRDataSyncFunction to run, but it is not the only required policy. The other required policies come incorporated out of the box, but it's a good idea to confirm that they're present. These policies are:

- AWSLambdaBasicExecutionRole
- AWSLambdaKinesisExecutionRole
- InvokeSalesforceRestApiFunctionRolePolicy

Depending on how your org is configured, out of the box policies can come with additional attributes.

Step 4: Re-Sync Voice Calls That Aren't Synced

Modify the InvokeSalesforceRestApiFunction on page 142 Lambda to perform three key actions:

1. Query for Voice Calls that aren't synced. Query for all Voice Call records (over a given time period) that weren't updated by the CTRDataSyncFunction on page 156. To identify Voice Calls that aren't updated, you can filter by fields that are only updated during a CTR sync. In this example, we specify a condition based on `CustomerHoldDuration`.

   ```
   SELECT Id, VendorCallKey FROM VoiceCall WHERE CustomerHoldDuration = null
   ```

2. Fetch CTRs from S3. If there are results returned from the previous step, then read from the S3 bucket to fetch CTRs for the given Voice Call records. Look up this data with the VendorCallKey.
3. Update Voice Call records in the org. Update Voice Call records based on the information returned from the previous step.

Important: This code calls the InvokeTelephonyIntegrationApiFunction on page 145 Lambda function. You can call it in one of two ways:

- Declare an environment variable for that Lambda and reference it from your code.
- Hard-code the ARN of the InvokeTelephonyIntegrationApiFunction on page 145 Lambda into your code.

This sample code assumes that there's an environment variable named INVOKE_TELEPHONY_INTEGRATION_API_ARN (option a), but you can hard-code the ARN if you prefer.

This code updates to InvokeSalesforceRestApiFunction on page 142 perform all these actions.

Updates to `sfRestApi.js`. In-context changes are in bold. Remove any comments in the code before saving.

```
const utils = require("./utils");
const axiosWrapper = require("./axiosWrapper");
const SCVLoggingUtil = require("./SCVLoggingUtil");

const aws = require('aws-sdk');
const lambda = new aws.Lambda();
const s3 = new aws.S3();

….
async function searchRecord(sosl) {
  try {
    const response = await sendRequest(
      "get",
      `/search/?q=${encodeURIComponent(sosl)}`
    );
    SCVLoggingUtil.debug({
      category: "sfRestApi.searchRecord",
      message: "search Record response",
      context: response,
    });
    if (response.data.searchRecords.length === 0) {
      return {};
    }
    return response.data.searchRecords[0];
  } catch (e) {
    return buildError(e);
  }
}

// NEW function that performs CTR sync retry
async function retryCtrSync(soql) {
    // Query For VoiceCalls that are missing CTR Sync
    const response = await sendRequest(
      "get",
      `/query/?q=${encodeURIComponent(soql)}`
    );
    console.log("Querying for VoiceCalls that were not updated");

    if (response.data.errorType) {
        console.log("Error querying VoiceCalls: " + response.data.errorMessage);

        return {
            success: false,
            errorMessage: response.data.errorMessage
        };
    } else if (response.data.totalSize === 0) {
        console.log("No VoiceCalls were found that were missing CTR update");
        return {};
    } else {
        console.log("Successfully queried for VoiceCalls");
        const result = response.data.records;
```

```
        // Fetch CTRs from S3
        var ctrRecords = await getCtrsFromS3(result);

        // Update VoiceCalls with the CTRs fetched from S3
        var results = await updateVoiceCallRecords(ctrRecords);
        return results;
    }
}


// NEW Function to fetch CTRs from S3
async function getCtrsFromS3(result) {
    var ctrData = [];
    for (var i=0; i<result.length; i++) {
        console.log("Vendor Call Key " + result[i].VendorCallKey);
        var getParams = {
            Bucket: '<your S3 bucket Name>',      // your bucket name,
            Key:  result[i].VendorCallKey // path to the object you're looking for
        }

        await s3.getObject(getParams, function(err, data1) {
            if (err) {
                console.log("ERROR in querying S3 " + err);
                return err;
            } else {
              var objectData = data1.Body.toString('utf-8'); // Use the encoding necessary

                var ctrObj = JSON.parse(objectData);
                console.log("CTR found for ContactId: " + ctrObj.ContactId);
                ctrData.push(ctrObj);
            }
        }).promise();
    }
    return ctrData;
}

// NEW function to update VoiceCall Records
async function updateVoiceCallRecords(ctrRecords) {
    let promises = [];
    ctrRecords.forEach((record) => {
        const ctr = record;

        if (ctr.ContactId) {
            const voiceCall = utils.transformCTR(ctr);
            promises.push(updateVoiceCallRecord(voiceCall));
        } else {
            console.log("No ContactId found in CTR");
        }
    });

    return await Promise.all(promises);
}
```

```
// UPDATED: Add environment variable
function updateVoiceCallRecord(voiceCall) {
    console.log("Updating VoiceCall");
    const payload = {
        Details: {
            Parameters: {
                methodName: 'updateVoiceCall',
                fieldValues: voiceCall.fields,
                contactId: voiceCall.contactId
            }
        }
    };

    const params = {
        // NEW: Create this as an environment variable
        //      or hard-code the ARN of the Lambda
        FunctionName: process.env.INVOKE_TELEPHONY_INTEGRATION_API_ARN,

        Payload: JSON.stringify(payload)
    };
    return lambda.invoke(params).promise();
}


// UPDATED: Include the new function (retryCtrSync)
module.exports = {
    createRecord,
    updateRecord,
    queryRecord,
    searchRecord,
    sendRealtimeAlertEvent,
    retryCtrSync,
};
```

Updates to `QueryEngine.js`. In-context changes are in bold. Remove any comments in the code before saving.

```
const api = require("./sfRestApi");
const SCVLoggingUtil = require("./SCVLoggingUtil");

function formatQuery(args, queryStr) {
  let query;
  Object.keys(args).forEach((key) => {
    const replacement = `{${key}}`;
    query = queryStr.replace(replacement, args[key]);
  });
  return query;
}

// invokes the query from sf rest api
// can take the query as a formatted string of sorts,
// replacing {key} with its value in the js object
async function invokeQuery(query, args) {
  const formattedQuery = formatQuery(args, query);
  SCVLoggingUtil.debug({
```

```
    message: "invoke query from SfRestApi",
    context: { payload: formattedQuery },
  });
  return api.queryRecord(formattedQuery);
}


// NEW function to that exposes the Retry CTR
async function retryCTRSync(query, args) {
    var formattedQuery = formatQuery(args, query);
    return await api.retryCtrSync(formattedQuery);
}


// UPDATED: Export the new function (retryCtrSync)
module.exports={
    invokeQuery,
    formatQuery,
    retryCTRSync
};
```

Updates to `Handler.js`. In-context changes are in bold:

```
const flatten = require("flat");
const SCVLoggingUtil = require("./SCVLoggingUtil");
const api = require("./sfRestApi");
const queryEngine = require("./queryEngine");
const utils = require("./utils");
const SFSPhoneCallFlow = require("./SFSPhoneCallFlow");

// --------------- Events -----------------------
async function dispatchSearch(sosl) {
  const searchResult = await api.searchRecord(sosl);
  return flatten(searchResult);
}


async function dispatch_ctrsync(soql, event){
    const parameters = event.Details.Parameters;
    console.log(event)
    let response;
    try {
        const queryResult = await queryEngine.retryCTRSync(soql, parameters);
        response = {
            statusCode: 200,
            result: queryResult
        }
    }
    catch (e) {
        response = {
            statusCode: e.response && e.response.status ? e.response.status : 500,
            result: e
        }
    }
```

```
    return flatten(response);
}

// --------------- Main handler ----------------------;
    case "SFSPhoneCallFlowQuery": {
      const res = await SFSPhoneCallFlow.entryPoint(event);
      result = flatten(res);
      break;
    }

    case 'retryCTRSync': {
            result = dispatch_ctrsync(soql, event);
            break;
    }


    default: {
      SCVLoggingUtil.warn({
        message: "Unsupported method",
        context: { payload: event },
      });
      throw new Error(`Unsupported method: ${methodName}`);
    }
  }

  if (result.success === false) {
    throw new Error(result.errorMessage);
  } else {
    return result;
  }
};

    }
```

Updates to `utils.js`. In-context changes are in bold. Remove any comments in the code before saving.

The `transformCTR` and `getCallAttributes` functions are copied over from the CTRDataSyncFunction on page 156 `utils.js` file. (Alternatively, directly modify CTRDataSyncFunction on page 156 to support taking a single CTR and updating the Voice Call record from there.) These functions must be inserted under the declaration of the `getRealtimeAlertEventFieldValuesFromConnectLambdaParams` function.

```
function getRealtimeAlertEventFieldValuesFromConnectLambdaParams(params) {
  const fieldValues = {};
  Object.entries(params).forEach((entry) => {
    const key = entry[0];
    if (key !== "methodName") {
      fieldValues[key] = entry[1];
    }
  });
  return fieldValues;
}

// COPIED function to transform CTR
function transformCTR(ctr) {
    const voiceCall = {};
```

```
voiceCall.startTime = ctr.InitiationTimestamp;
voiceCall.endTime = ctr.DisconnectTimestamp;
voiceCall.parentCallIdentifier = ctr.PreviousContactId;

if (ctr.Agent) {
    voiceCall.acceptTime = ctr.Agent.ConnectedToAgentTimestamp;
    voiceCall.totalHoldDuration = ctr.Agent.CustomerHoldDuration;
    voiceCall.longestHoldDuration = ctr.Agent.LongestHoldDuration;
    voiceCall.agentInteractionDuration = ctr.Agent.AgentInteractionDuration;
    voiceCall.numberOfHolds = ctr.Agent.NumberOfHolds;
}

if (ctr.Queue) {
    voiceCall.enqueueTime = ctr.Queue.EnqueueTimestamp;
    voiceCall.queue = ctr.Queue.Name;
}

if (ctr.InitiationMethod) {
    voiceCall.initiationMethod = ctr.InitiationMethod;
    if (ctr.InitiationMethod === 'OUTBOUND') {
        if (ctr.SystemEndpoint) {
            voiceCall.fromNumber = ctr.SystemEndpoint.Address;
        }
        if ( ctr.CustomerEndpoint) {
            voiceCall.toNumber = ctr.CustomerEndpoint.Address;
        }
    } else {
        if (ctr.SystemEndpoint) {
            voiceCall.toNumber = ctr.SystemEndpoint.Address;
        }
        if ( ctr.CustomerEndpoint) {
            voiceCall.fromNumber = ctr.CustomerEndpoint.Address;
        }
    }
}

if (ctr.Recording) {
    voiceCall.recordingLocation = ctr.Recording.Location;
}

// Check if there are custom contact attributes
if (ctr.Attributes) {
    var callAttributes = {};

    // Get contact attributes data into call attributes
    callAttributes =  getCallAttributes(ctr.Attributes);

    voiceCall.callAttributes = callAttributes;
}

Object.keys(voiceCall).forEach(function (key) {
    if (voiceCall[key] === null || voiceCall[key] === undefined) {
```

```
            delete voiceCall[key];
        }
    });

    return {contactId: ctr.ContactId, fields: voiceCall};
}

// COPIED function to filter call attributes
/**
 * Filter call attributes to be included in API payload based on prefix and strip prefix
 *
 * @param {object} rawCallAttributes - Contact flow attributes
 *
 * @return {string} - Stringified contact flow attributes with prefix removed
 */
function getCallAttributes(rawCallAttributes) {
    const prefix = 'sfdc-';
    const prefixLen = prefix.length;
    let callAttributes = {};

    for (const key in rawCallAttributes) {
        if (rawCallAttributes.hasOwnProperty(key) && key.startsWith(prefix)) {
            callAttributes[key.substring(prefixLen)] = rawCallAttributes[key];
        }
        // Set SCV Limits Error if the specific contact attribute is set
        if (rawCallAttributes.sf_realtime_transcription_status) {
            callAttributes.sf_realtime_transcription_status =
rawCallAttributes.sf_realtime_transcription_status;
        }
    }

    return JSON.stringify(callAttributes);
}


// UPDATED: Export the two new functions (transformCTR, getCallAttributes)
module.exports = {
    generateJWT,
    getAccessToken,
    formatObjectApiName,
    getSObjectFieldValuesFromConnectLambdaParams,

    transformCTR,
    getCallAttributes

};
```

In InvokeSalesforceRestApiFunction, add the AmazonS3FullAccess policy to the role:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "s3:*",
                "s3-object-lambda:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Important: Adding the AmazonS3FullAccess policy to the role is required for the CTRDataSyncFunction to run, but it's not the only required policy. The other required policies come incorporated out of the box, but it's a good idea to confirm that they're present. These policies are:

- AWSLambdaBasicExecutionRole
- InvokeSalesforceRestApiFunctionRolePolicy

Depending on how your org is configured, out of the box policies can come with additional attributes.

Next, publish a new version of the Lambda functions. Make sure that you update this code in your Lambda functions to ensure compatibility after the contact center update.

Step 6: Call the New Method on the InvokeSalesforceRestApiFunction Lambda

Call this new method in InvokeSalesforceRestApiFunction on page 145 by specifying a `methodName` of `retryCTRSync` and a `soql` attribute of `SELECT Id, VendorCallKey FROM VoiceCall WHERE CustomerHoldDuration = null`. You can schedule this call to run periodically. See Schedule AWS Lambda Functions Using CloudWatch Events in the Amazon documentation.

Test This Example

To test this example:

1. Dial your phone number in order to create a Voice Call record. After the call is answered, hang up.
2. In your org, find the newly created Voice Call record and remove some CTR fields, including `CustomerHoldDuration`.
3. Call InvokeSalesforceRestApiFunction on page 145 with the information specified in the final step (step 6) of this example. However, instead of scheduling the function to run, you can programmatically call the Lambda from the console as shown in Amazon's documentation (Create a Lambda function with the console). Use a payload similar to this:

```
{
  "Details": {
    "Parameters": {
      "methodName": "retryCTRSync",
      "soql": "SELECT Id, VendorCallKey FROM VoiceCall WHERE CustomerHoldDuration =
null"
    }
  }
}
```

4. Check that the Voice Call record now has the CTR fields filled in.

Disable Contact Record Sync for Voice Call Records

When a contact record (previously called contact trace record or CTR) is created in Amazon, the CTRDataSyncFunctionLambda Lambda function invokes the UpdateVoiceCallRecords function, synchronizing contact record data to a VoiceCall object. However, there are times when you don't want to sync the contact record data between your Amazon Connect instance and Salesforce. For example, during a phased rollout, automatic synchronization can cause redundancies. In these cases, disable contact record sync.

Ease of Implementation	Medium
Estimated Time to Implement	15 minutes

In this example, we show you how to disable contact record sync so that changes to the contact record data in Amazon Connect are not pushed automatically to Salesforce.

This configuration applies to the following telephony models.

- Service Cloud Voice with Amazon Connect
- Service Cloud Voice with Partner Telephony from Amazon Connect

Prerequisites

Before you begin:

- Verify that you have AWS root user or AWS administrator credentials.
- Verify the AWS Account you are working with is already connected to a Salesforce org.

Disable CTR Sync

1. From the Amazon Connect console, select **Routing > Flows**.
2. Open the inbound flow where you want to disable contact record sync.
3. In the **Set contact attributes** flow block, add the following attribute:
   - Namespace: User defined
   - Key: NoSync
   - Select **Set manually** and set the value to `true`.
4. Click

   ```
   Save
   ```

Test This Example

To test this example:

1. Dial your phone number in order to create a record in Amazon Connect. After the call is answered, hang up.
2. Confirm that the contact record data resulting from your phone call is saved in Amazon Connect and not Salesforce.

Enable Voice Call Transfers Using Omni-Channel Flows and Amazon Connect

Configure this feature to enable voice call transfers via Salesforce Omni-Channel flows.

This configuration applies to the following telephony models:

- Service Cloud Voice with Amazon Connect
- Service Cloud Voice with Partner Telephony from Amazon Connect

To enable this feature for Service Cloud Voice with Partner Telephony, go to Enable Voice Call Transfers Using Omni-Channel Flows.

Omni-Channel flows can be used to transfer voice calls through External Routing. Configure this feature to enable voice call transfers using Omni-Channel flows. When this feature is enabled, all active flows of process type "Omni-Channel Flow" that are assigned to the phone channel appear in the Omni-Channel widget for reps to select as transfer destinations.

Configuring the feature includes: 1) creating an Omni-Channel flow in Salesforce to route calls to a rep or queue, 2) creating a Service Cloud Voice contact flow in Amazon Connect to transfer and route voice calls, 3) creating an Amazon Connect quick connect of type “queue” to transfer voice calls using the newly created contact flow, and editing the Salesforce contact center details by selecting the quick connect to be used for the Omni-Channel flow transfer.

To enable voice call transfers using Omni-Channel flow and Amazon Connect:

1. Log in to Salesforce and create an Omni-Channel flow, using the “Voice Calls Routed to Agents and Queues” template as a guide.
2. Log in to Amazon AWS and create a contact flow in Amazon Connect, using the Sample SCV Transfer Flow for Omni-Channel Transfers contact flow in GitHub as a template. The contact flow generates a transfer voice call and routes the call using the Execute an Omni-Channel Flow REST API.
3. Log in to Amazon AWS and create an Amazon Connect quick connect with the following settings:
   - **Name**: Enter a unique name for the quick connect. This is the name that you’ll select when you connect the quick connect to your contact center in Salesforce.
   - **Type**: Select **Queue**.
   - **Destination**: Select the destination queue. This can be any queue, such as Amazon Connect’s default BasicQueue.
   - **Contact flow**: Select the name of the contact flow you created.
   - **Description**: Optionally enter a brief description of the quick connect.
4. Log in to Salesforce and edit the Contact Center Details page in Salesforce by setting the Quick Connect for Omni-Channel Flow Transfers field to the name of the Amazon Connect quick connect you created.

When a rep transfers a voice call to the Omni-Channel flow, Salesforce uses the Amazon Connect `UpdateContactAttributes` action to set a contact attribute as the fully qualified name of the Omni-Channel flow of the parent contact. The call is transferred to the quick connect that’s selected in the “Quick Connect for Omni-Channel Flow Transfers” field of the Contact Center Details page in Salesforce. The contact flow defined in the quick connect creates a voice call like it would with other voice call transfers, and then executes the Omni-Channel flow using the Execute an Omni-Channel Flow API passing the flow name from contact attribute.

Note:

- Make sure the reps and queues are mapped to Amazon Connect ARN. If you don't, the Execute an Omni-Channel Flow request returns an error. Rep and queue mappings are configured in the Contact Center Details page in Salesforce.
- Don't add the newly created quick connect directly to the list of transfer destinations in the Omni-Channel softphone. If you do, transfers to the quick connect will fail.

Refresh a Sandbox

Refreshing a sandbox updates its metadata from the source org.

Ease of Implementation	Medium
Estimated Time to Implement	30 minutes

In this example, we show you how to reuse an AWS Account to refresh your sandbox and relink the refreshed sandbox to the Amazon Connect instance.

This configuration applies to users of Service Cloud Voice with telephony provided by Amazon Connect.

- If your telephony model is Service Cloud Voice with Amazon Connect, complete all the refresh steps in this page.
- If your telephony model is Service Cloud Voice with Partner Telephony from Amazon Connect, complete the steps in this page except for the ones you are asked to skip.

In this example, you will perform the following steps to refresh the sandbox:

1. Retrieve (back up) the metadata for the following packages:
   - *ConversationVendorInfo
   - CallCenter
   - Single sign-on connected app
2. Refresh the sandbox.
3. *Deploy the ConversationVendorInfo metadata to the refreshed sandbox.
4. Deploy the single sign-on connected app metadata to the refreshed sandbox and update the CallCenter metadata with the new connected app values.
5. Create the REST API OAuth connected app and update the CallCenter metadata with the new connected app values.
6. Deploy the CallCenter metadata to the refreshed sandbox.
7. Map the identity provider (IdP) certificate to the connected apps.
8. Update the AWS IAM SalesforceServiceVoiceIdp identity provider with the latest IdP certificate.
9. Update the Service Cloud Voice AWS Lambda functions with the latest environment variable settings.

*Skip this step if your telephony model is Service Cloud Voice with Partner Telephony from Amazon Connect.

Prerequisites

Before you begin:

- Verify that you have AWS root user or AWS administrator credentials.
- Verify the AWS Account you want to use isn't being used by another Salesforce org.
- Review the Sandbox Org Guidelines guidance around sandbox orgs and sandbox refreshes.

Refresh the Sandbox

1. Skip this step if your telephony model is Service Cloud Voice with Partner Telephony from Amazon Connect.

Retrieve the **ConversationVendorInfo** metadata from the sandbox org to create a backup copy named package.xml. The ConversationVendorInfo package contains information about the AWS Account.

Here's a sample version of the `ConversationVendorInfo` package:

```
<!— ConversationVendorInfo Package package.xml-->
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
<types>
<members>SERVICE_CLOUD_VOICE</members>
<name>ConversationVendorInfo</name>
</types>
<version>55.0</version>
</Package>
```

2. Retrieve the **CallCenter** metadata from the sandbox org to create a backup copy named package.xml. The CallCenter package contains the call center definition used to integrate the Salesforce contact center with the Amazon Connect instance.

   Here's a sample version of the CallCenter package, where `ContactCenterWest` is the internal name of the contact center:

   ```
   <!— CallCenter Package package.xml-->
   <?xml version="1.0" encoding="UTF-8"?>
   <Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
   <members>ContactCenterWest</members>
   <name>CallCenter</name>
   </types>
   <version>55.0</version>
   </Package>
   ```

3. Retrieve the **Single Sign-On Connected App** metadata from the sandbox org to create a backup copy named package.xml. The single sign-on connected app package contains the app configuration used to integrate Salesforce with the Amazon Connect instance.

   Here's a sample version of the single sign-on connected app package, where `ContactCenterWest` is the internal name of the contact center:

   ```
   <!— ConnectedApp Package package.xml-->
   <?xml version="1.0" encoding="UTF-8"?>
   <Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
   <members>ContactCenterWest_Connected_App</members>
   <name>ConnectedApp</name>
   </types>
   <version>55.0</version>
   </Package>
   ```

4. Refresh your sandbox.
5. Skip this step if your telephony model is Service Cloud Voice with Partner Telephony from Amazon Connect.

   Deploy the **ConversationVendorInfo** metadata to the refreshed sandbox org and verify that you receive an email from Salesforce confirming Service Cloud Voice has successfully connected to the AWS Account.

6. Skip this step if your telephony model is Service Cloud Voice with Partner Telephony from Amazon Connect.

   Perform the following steps to finish turning on Service Cloud Voice in the refreshed sandbox org:

- Go to **Setup > Voice > Amazon Setup** and verify that **Turn on Voice with Amazon Connect is ON** and grayed out.
- In the Register Tax Number section, click **Confirm Settings**.
- Verify that you receive an email from Salesforce confirming Service Cloud Voice is turned on.

7. Deploy the **single sign-on connected app** metadata to the refreshed sandbox org.
8. Copy the following values from the newly deployed single sign-on connected app to the CallCenter metadata XML file:
   - **reqConnectedAppId**. From the Developer Console, open the ConnectedApplication.obj file and copy the 15- or 18-digit org ID of the newly deployed connected app. Replace the value of `reqConnectedAppId` in the XML file with the value you just copied.
   - **reqIdentityUrl**. Go to **Setup > App Manager**. Click **Manage** next to the newly deployed connected app and copy the IdP-Initiated Login URL. Replace the value of `reqIdentityUrl` in the XML file with the value you just copied.
9. Give users access to the single sign-on connected app using permission sets (preferred) or profiles. Go to **Setup > App Manager**. Click **Manage** next to the newly deployed connected app and assign the profiles to the app or enable the Service Cloud Voice Permission Set in the app.
10. Create the **REST API OAuth connected app** in the refreshed sandbox org by following the steps in the Set Up OAuth in Your Service Cloud Voice Connected App document.
11. Copy the following value from the newly created REST API OAuth connected app to the CallCenter metadata XML file:
    - **reqRestApiConnectedAppId**. From the Developer Console, open the ConnectedApplication.obj file of the newly deployed connected app and copy the `reqRestApiConnectedAppId` value. Replace the value of `reqRestApiConnectedAppId` in the XML file with the value you just copied.
12. Delete the `contactCenterChannels` element from the CallCenter metadata XML file.
13. Deploy the CallCenter metadata containing the values you just updated to the refreshed sandbox org.
14. Map the IdP certificate to the single sign-on connected app.
    a. To find your IdP certificate for single sign-on, select **Setup > Apps > Connected Apps > Manage Connected Apps**, and open the connected app with the label `{salesforce_contact_center_internal_name} Connected App`. Write down the label name of the currently chosen certificate.
    b. Go to **Setup > Identity > Identity Provider** and make sure that the current Idp certificate label is the same as that of the connected app certificate label noted down in step 14.a. If the labels do not match, update the single sign-on connected app's certificate to match with the Salesforce Identity Provider certificate.
    c. Click **Download Metadata** on the Salesforce Identity Provider page to download the SAML metadata in XML format.
15. Update the SalesforceServiceVoiceIdp provider in AWS IAM with the latest IdP metedata from Salesforce.
    a. Go to **IAM > Identity Providers**, and click **SalesforceServiceVoiceIdp** to drill down to the details.
    b. Click **Replace Metadata** and replace the metadata with the XML file you downloaded in step 14.c.
16. Set the `SALESFORCE_ORG_ID` value depending on your contact center version.
    - For contact center versions below 19.0, set the `SALESFORCE_ORG_ID` environment variable in the AWS Lambda functions to the org ID of the refreshed sandbox.
      a. Find and copy the 15- or 18-digit org ID.
      b. Go to AWS Lambda.
      c. Click the name of the Lambda function that has the `SALESFORCE_ORG_ID` environment variable. For example, click **InvokeTelephonyIntegrationApiFunction**.

d. Change the value of `SALESFORCE_ORG_ID` to the org ID you just copied.

e. Repeat these steps for all Lambda functions - including custom ones - that have the `SALESFORCE_ORG_ID` environment variable.

After you change the `SALESFORCE_ORG_ID` environment variable for each of the impacted Lambda functions. make sure you publish the updated version of the Lambda function and set the Alias to the right version.

- For contact center versions 19.0 or later, set the value of the `SALESFORCE_ORG_ID` key in the Secret of the contact center.
  a. In Secrets Manager in the AWS console, select the secret for the contact center. The secret is in the format <Contact Center Internal Name>-salesforce-secret.
  b. In the Secret Value section, select Retrieve Secret and click Edit.
  c. Select the Plaintext view, and paste the org ID of the refreshed sandbox as the value for the SALESFORCE_ORG_ID key.
  d. Save your changes.

SEE ALSO:

Salesforce Help: Refresh Your Sandbox
Salesforce Help: Set Up Service Cloud Voice with Amazon
Salesforce Help: Set Up Service Cloud Voice with Partner Telephony from Amazon Connect
Salesforce Help: Developer Console
Amazon Documentation: AWS Lambda Developer Guide
Metadata API Developer Guide
Salesforce Help: Manage Contact Center Certificates
Salesforce Help: Configure Single Sign-on (SSO) to Your Telephony Provider

Enable Voicemail Support

Configure voicemail support to route voicemail recordings and transcriptions to reps. Reps can play back the recordings and transcriptions routed to them at any time.

Ease of Implementation	Medium
Estimated Time to Implement	30 minutes

This configuration applies to the Service Cloud Voice with Partner Telephony from Amazon Connect telephony setup.

This feature is available in the Spring '23 and later releases.

In this example, we show you how to set up your system to support voicemail. There are two ways to enable Voicemail support, depending on how you provisioned the Amazon Connect instance when you created the contact center:

- If the contact center was created using a new Amazon Connect instance, you must upgrade to contact center version 11.0 or later to get the latest Lambda functions.
- If the contact center was created using an existing Amazon Connect instance, you must configure Kinesis as the event source to work with the Lambda functions in the ServiceCloudVoiceLambdas serverless app.

In this example, you'll perform the following steps to enable voicemail support for Service Cloud Voice with Partner Telephony from Amazon Connect.

1. Add and configure the VoiceMailAudioProcessingFunction Lambda function. This step only applies if the contact center was created using an existing Amazon Connect instance.
2. Enable Amazon EventBridge.
3. Configure an EventBridge Rule for the `VoiceMailTranscribeFunction` Lambda function.
4. Configure an EventBridge Rule for the `VoiceMailPackagingFunction` Lambda function.
5. Bind the voicemail subflow to the inbound flow in Amazon Connect.

This example applies to both contact centers created using a new or existing Amazon Connect instance.

See Lambda Functions for more information about the VoiceMailAudioProcessingFunction, VoiceMailTranscriberFunction, and VoiceMailPackagingFunction Lambda functions.

To access Amazon Web Services documentation, go to https://docs.aws.amazon.com.

Prerequisites

Complete the prerequisites before you perform the steps:

- Verify that you have AWS root user or AWS administrator credentials.
- Deploy the `ServiceCloudVoiceLambdas`.
- If the contact center was originally created using a new Amazon Connect instance, upgrade the contact center to version 11.0 or later.
- Note the name of the S3 bucket you'll use to store the voicemail recordings and transcription files. The S3 object name is in the format <contactcentername-salesforceuniquerandomID>.

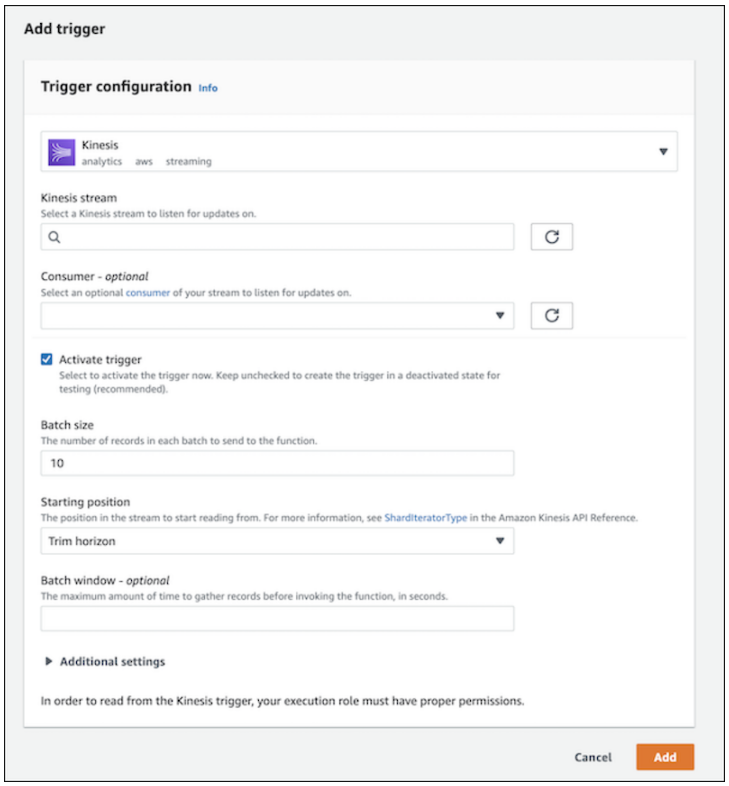
To implement this use case:

Step 1: Add and Configure the VoiceMailAudioProcessingFunction Lambda Function.

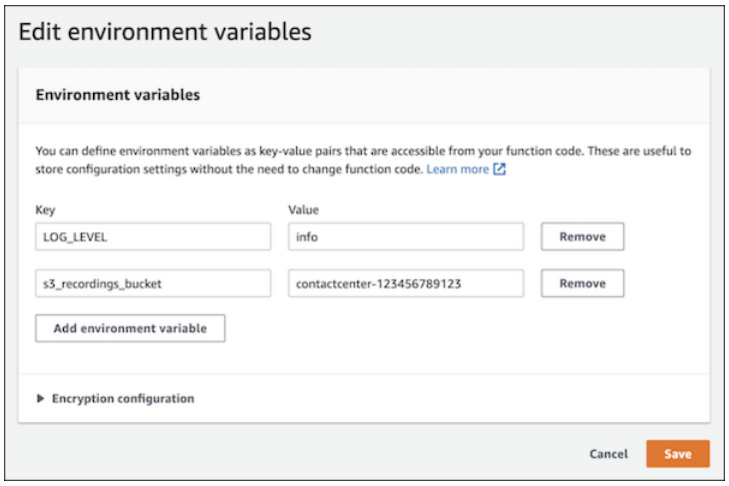
Perform this step if the contact center was created using an existing Amazon Connect instance. Skip this step if the contact center was created using a new Amazon Connect instance.

Add and configure an Amazon Kinesis trigger to notify the VoiceMailAudioProcessingFunction Lambda function when an Amazon Contact Record (previously called contact trace record or CTR) event is generated after a caller leaves a voicemail message and hangs up.

1. From the AWS Management Console, go to the AWS Lambda service and select **Functions**.
2. In the Function name list, click **VoiceMailAudioProcessingFunction**.
3. Click **+Add trigger**.
4. In the Trigger configuration section, select **Kinesis** from the dropdown menu.
5. Set the Batch size to **10**.
6. Set the Starting position to **Trim Horizon**.



7. Click **Add**.
8. Click the **Configuration** tab, select **Environment variables**, and then click **Edit**.
9. Find the s3_recordings_bucket key and replace the `testVoiceMailBucket` value with the name of the contact center S3 bucket that will store the voicemail recordings and transcriptions. For example, contactcenter-123456789123. You can select an S3 bucket that you already use for Service Cloud Voice.

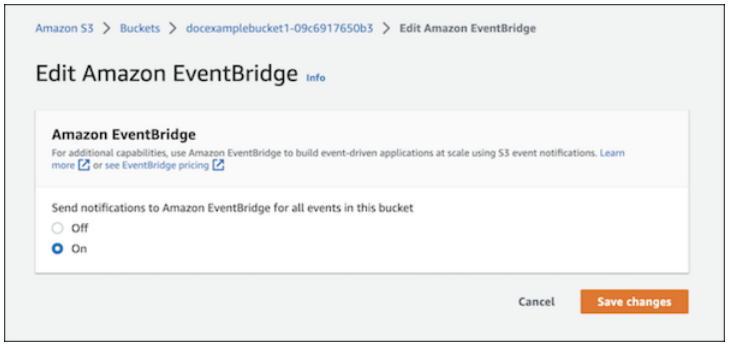


10. Click **Save**.

Step 2: Enable Amazon EventBridge

Enable Amazon EventBridge so it can trigger actions.

1. From the AWS Management Console, open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the Buckets list, click the name of the S3 bucket that will store the voicemail recordings and transcriptions. You can select an S3 bucket that's already used for Service Cloud Voice purposes.
3. Click the **Properties** tab.
4. In the Amazon EventBridge section, click **Edit**.
5. In the Edit Amazon EventBridge screen, under **Send notifications to Amazon EventBridge for all events in this bucket**, select the **On** option.

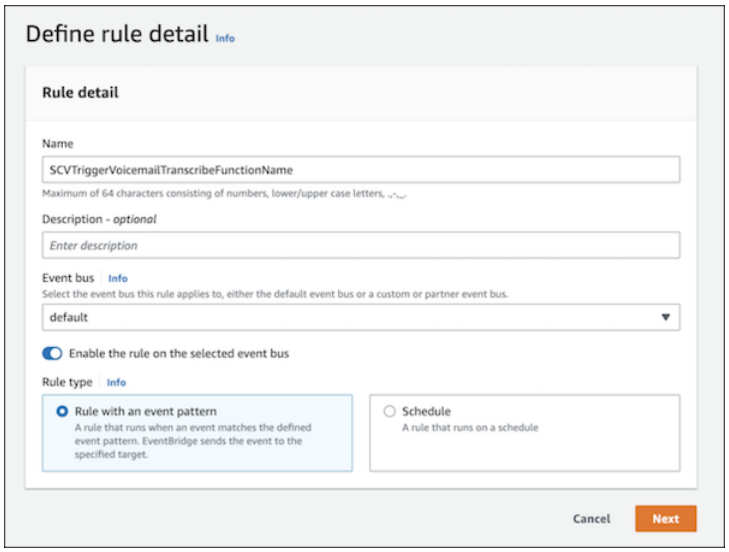


6. Click **Save changes**.

Step 3: Configure an Amazon EventBridge Rule for VoiceMailTranscribeFunction

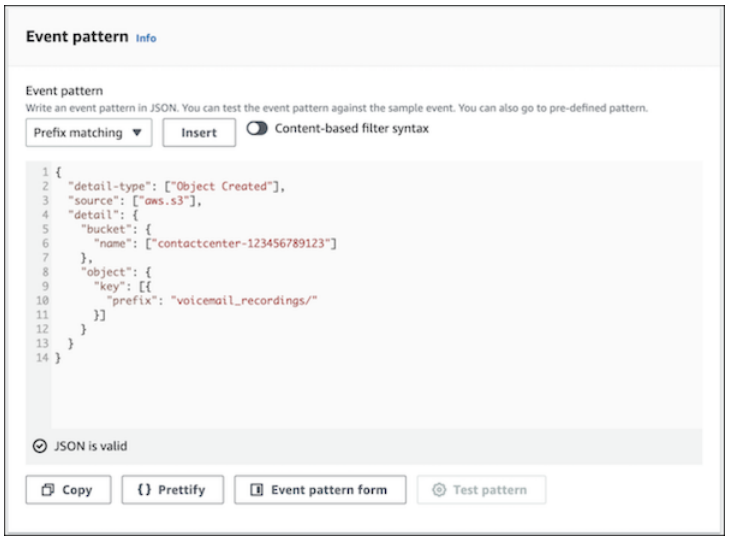
Configure an EventBridge rule to notify the VoiceMailTranscribeFunction when voicemail recordings are added to the voicemail_recordings folder in the contact center's Amazon S3 bucket.

1. From the AWS Management Console, go to **Amazon EventBridge > Rules**.
2. Click **Create rule**.
3. In the Define rule detail screen, give the rule a name and optional description. Leave the other default settings.

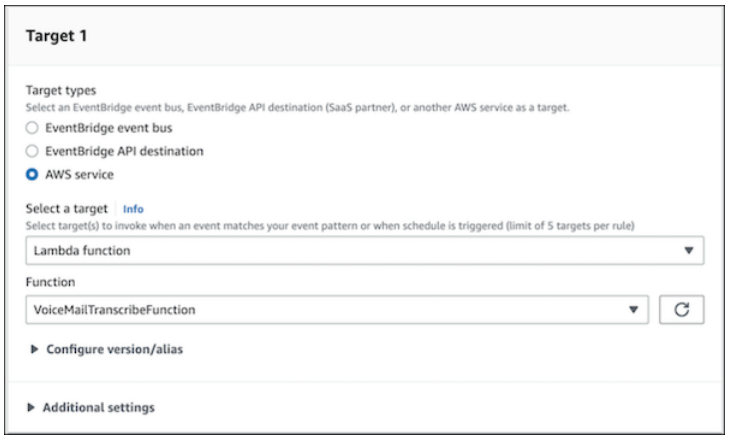


4. Click **Next**.
5. In the Build event pattern screen, scroll down to the Event pattern section and click **Custom patterns (JSON editor)**.
6. Copy the following JSON sample event and paste it into the Event pattern text box, replacing the name of the bucket with your contact center's S3 object name:

```
{
  "detail-type": ["Object Created"],
  "source": ["aws.s3"],
  "detail": {
    "bucket": {
      "name": ["contactcenter-123456789123"] // replace with contact center S3 bucket
name
    },
    "object": {
      "key": [{
        "prefix": "voicemail_recordings/"
      }]
    }
  }
}
```



7. Click **Next**.
8. In the Select target(s) screen, set the Select a target dropdown to Lambda function, and set the Function to the `VoiceMailTranscribeFunction` function.



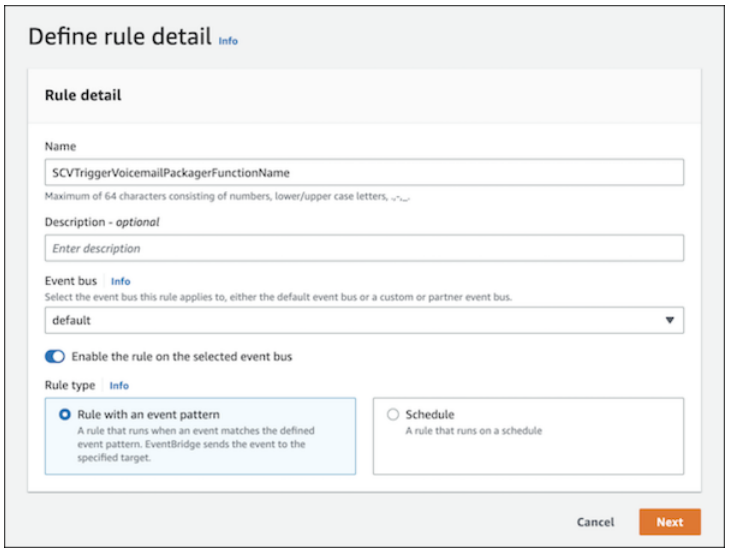
9. Click **Next** twice.
10. In the Review and create screen, review the settings for your new EventBridge rule and click **Create rule**.

Step 4: Configure an Amazon EventBridge Rule for VoiceMailPackagingFunction

Configure an EventBridge rule to notify the VoiceMailPackagingFunction when voicemail transcriptions are added to the voicemail_transcriptions folder in the contact center's Amazon S3 bucket.

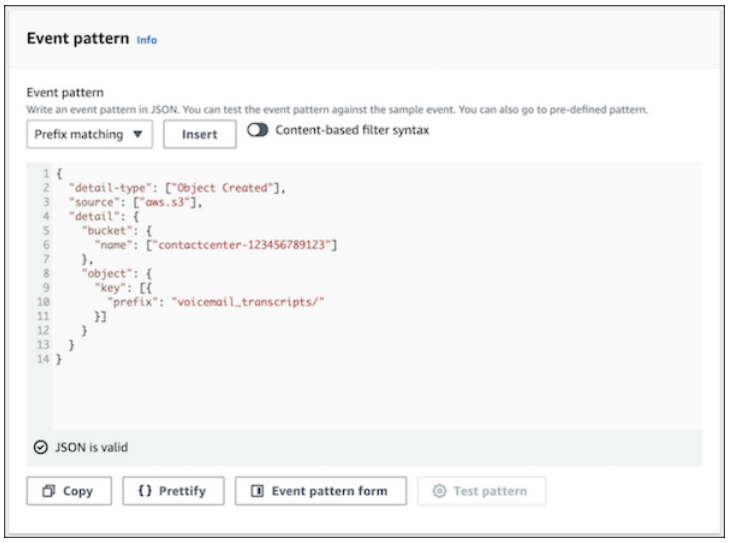
1. From the AWS Management Console, go to Amazon EventBridge > Rules.
2. Click Create rule.

3. In the Define rule detail screen, give the rule a name and optional description. Leave the other default settings.



4. Click **Next**.
5. In the Build event pattern screen, scroll down to the Event pattern section and click Custom patterns (JSON editor).
6. Copy the following JSON sample event and paste it into the Event pattern text box, replacing the name of the bucket with your contact center's S3 object name:

```
{
  "detail-type": ["Object Created"],
  "source": ["aws.s3"],
  "detail": {
    "bucket": {
      "name": ["contactcenter-123456789123"] // replace with contact center S3 bucket
name
    },
    "object": {
      "key": [{
        "prefix": "voicemail_transcripts/"
      }]
    }
  }
}
```



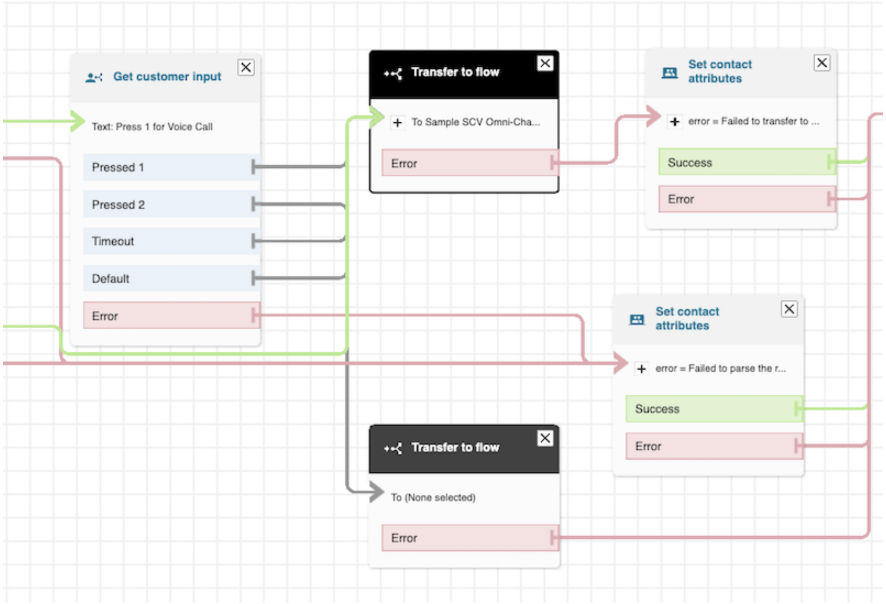
7. Click **Next**.
8. In the Select target(s) screen, click the Select a target dropdown and select **Lambda function**. Click the **Function** dropdown and select **VoiceMailPackagingFunction**.
9. Click **Next** twice.
10. In the Review and create screen, review the settings for your new EventBridge rule and click **Create rule**.

Step 5: Add the Voicemail Subflow in Amazon Connect

Bind the voicemail subflow to the inbound flow. You must have permissions to create flows in Amazon Connect.

1. From the Amazon Connect console, select **Routing > Contact Flow**.
2. Make a copy of the Sample SCV Voicemail Subflow. Click the **Sample SCV Voicemail Subflow**, click the dropdown arrow next to the Save button, select **Save As**, enter a name for the voicemail subflow, and click **Save**.
3. Click the inbound flow where you'll add the voicemail subflow.
4. Add a **Transfer to flow** flow block and configure it to **Set manually: <Name of the voicemail subflow>**.

5. Add a **Get customer input** flow block and configure prompts to give the caller the option to either proceed with the voice call or leave a voicemail message.
6. Insert the **Get customer input** flow block into the inbound flow so the voice call option transfers to the **Transfer to flow: Omni-Channel** flow block and the voicemail option transfers to the **Transfer to flow: Voicemail** flow block.
7. Click **Save**.



Voicemail Process

The diagram illustrates the voicemail process that happens when a caller initiates a voice call with the contact center.

(1) A caller initiates a voice call with the contact center, which triggers an Amazon Connect inbound flow to begin. When the flow reaches the Get customer input prompt flow block, the caller chooses to leave a voicemail message.

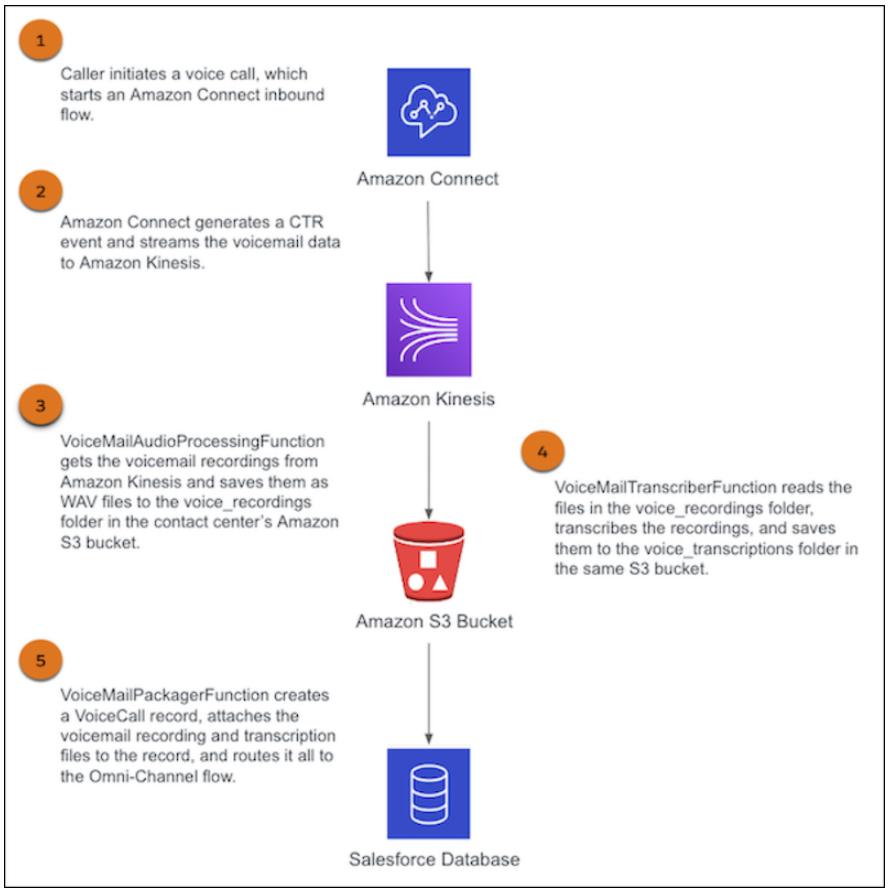
(2) After the caller leaves the voicemail message and hangs up, Amazon Connect generates a Contact Record (previously called contact trace record or CTR) event and streams the voicemail data to Amazon Kinesis.

(3) The contact record event triggers the VoiceMailAudioProcessingFunction Lambda function to get the voicemail recordings from Amazon Kinesis and saves them as WAV files to the voicemail_recordings folder in the contact center's Amazon S3 bucket.

(4) Every time a voicemail recording file is added to the voicemail_recordings folder, an EventBridge rule triggers the VoiceMailTranscriberFunction Lambda function to read the files in the S3 bucket, transcribe the recordings, and save the files into the voicemail_transcriptions folder in the contact center's Amazon S3 bucket.

(5) Every time a voicemail recording file is added to the voicemail_transcriptions folder, an EventBridge rule triggers the VoiceMailPackagingFunction Lambda function to create a VoiceCall record, attach the voicemail recording and transcription files to the record, and route it all to the Omni-Channel flow that was configured in the contact center.

The rep who receives the voice call can play back the voicemail recording and read the transcription at any time.



Ensure Correct Delivery of Voicemails

If you have voicemail enabled but voicemails aren't being delivered to the reps, update your Voicemail subflow to ensure the callOrigin field in the VoiceCall record is updated in time. For voicemails, the value of the callOrigin field must be Voicemail. You can update the Voicemail subflow to add a block to invoke the Lambda to call the SF REST API to make sure that the callOrigin field is updated in time, and voicemails are correctly delivered to the reps.

SEE ALSO:

VoiceMailAudioProcessingFunction Lambda Function
VoiceMailTranscribeFunction Lambda Function
VoiceMailPackagingFunction Lambda Function

Ensure Correct Delivery of Voicemails

If you have voicemail enabled but voicemails aren't being delivered to the reps, update your Voicemail subflow to ensure the callOrigin field in the VoiceCall record is updated in time. For voicemails, the value of the callOrigin field must be Voicemail. You can update the Voicemail subflow to add a block to invoke the Lambda to call the SF REST API to make sure that the callOrigin field is updated in time, and voicemails are correctly delivered to the reps.

applies to these telephony models
Service Cloud Voice with Amazon Connect
Service Cloud Voice with Partner Telephony from Amazon Connect

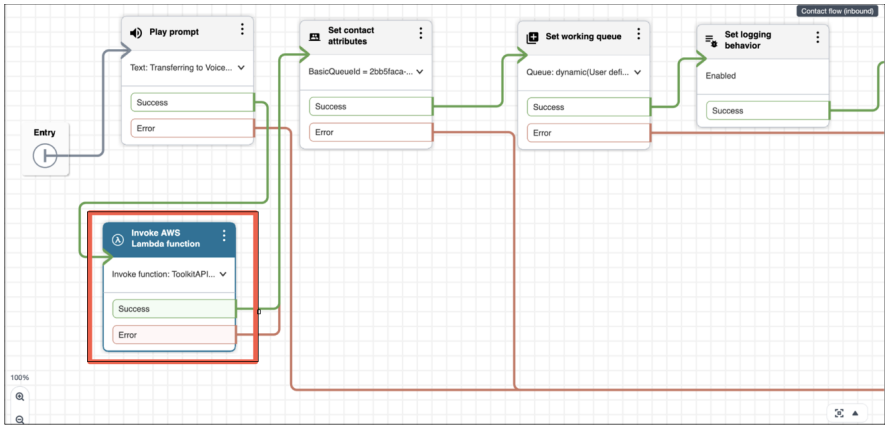
Before using the `InvokeSalesforceRestApiFunction` Lambda function, you must have OAuth authentication set up. To set up or modify authentication, see Service Cloud Voice Authentication When Using InvokeSalesforceRestApiFunction.

To ensure voicemails are delivered to reps, add the `InvokeSalesforceRestApiFunction` Lambda function to the Voicemail subflow, and configure the Lambda function to include the `callOrigin` destination key. Once configured, every time the Voicemail subflow is triggered, it invokes the Lambda function, which updates the `callOrigin` field to `Voicemail` in the `VoiceCall` record.

Important: Perform these steps only if the voicemails aren't correctly delivered to the reps.

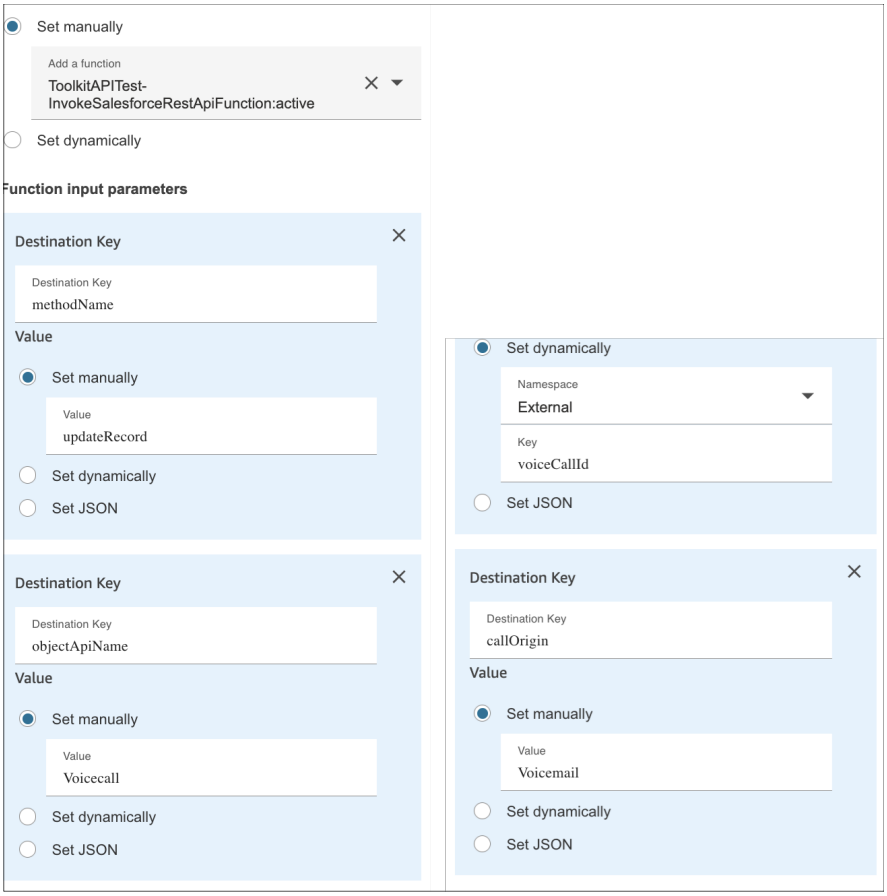
To configure the Voicemail subflow to ensure voicemails are delivered to reps:

1. Set up or modify OAuth.
2. In the sample SCV Voicemail Subflow, add the Invoke AWS Lambda function block to invoke the SF REST API. Add the Lambda block between the Play prompt and Set contact attributes blocks in your contact flow.



3. Configure the `InvokeSalesforceRestApiFunction` Lambda Function to update the value of the `callOrigin` destination key as `Voicemail`. Add a Set contact attributes flow block to the flow, making sure it appears before the Invoke AWS Lambda function block.
4. Add the following attributes to the Set contact attributes flow block:
   - Namespace: `External`

- Key: `voiceCallId`



5. Click **Save**.

This makes sure that every time the voicemail subflow is triggered, it invokes the Lambda function to update the `callOrigin` field to `Voicemail` in the `VoiceCall` record.

An example flow is available in the Github.

Set the Voice Call Record Type

Configure an Amazon Connect flow to automatically set the record type for a voice call record.

The record type determines the page layout of a voice call record. Configure an Amazon Connect flow to automatically set the record type for a voice call record. For example, configure an inbound flow in Amazon Connect to automatically set inbound voice calls to a "Platinum" record type. Whenever a call comes in through the inbound flow, the voice call record displays the page layout that's associated with the "Platinum" record type. After a voice call record is set, reps can manually change its record type through the Salesforce Lightning Experience UI.

This configuration applies to the following telephony models:

- Service Cloud Voice with Amazon Connect
- Service Cloud Voice with Partner Telephony from Amazon Connect

This feature is available in the Winter '23 and later releases.

This example explains how to configure the Sample SCV Inbound Flow to automatically set the record type whenever a voice call goes through the flow.

To run through this example:

- You must be an AWS admin with the ability to create flows in Amazon Connect.
- You must be a Salesforce admin.

To configure a flow so that voice calls are automatically set to a record type:

1. Find the 18-digit alphanumeric ID of the record type you want the flow to use. For example, `012S30000004B5VIAU`.
2. From the Amazon Connect console, select **Routing > Flows**.
3. Click the name of the flow.
4. Add a **Set contact attributes** flow block to the flow, making sure it appears before the Invoke AWS Lambda function block.
5. Add the following attribute to the Set contact attributes flow block:
   - Namespace: User defined
   - Key: sfdc-RecordTypeId
   - Select **Set manually** and set the value to the 18-digit ID of the record type you want to display.
6. Click **Save**.

Whenever a call comes in through this flow, the voice call record displays the page layout for the record type you specified.

Disable Call Recording

Configure an Amazon Connect flow to disable recording for a voice call.

There are times when you don't want to record a voice call. Follow these steps to disable call recording and to disable the Recording control for reps in the Omni-Channel widget. With the Recording control disabled, reps understand that the call isn't recorded and aren't able to toggle recording on or off.

This configuration applies to these telephony models:

- Service Cloud Voice with Amazon Connect
- Service Cloud Voice with Partner Telephony from Amazon Connect

This feature is available in Spring '24 and later releases.

This example explains how to configure a sample flow to disable call recording for a voice call that goes through the flow.

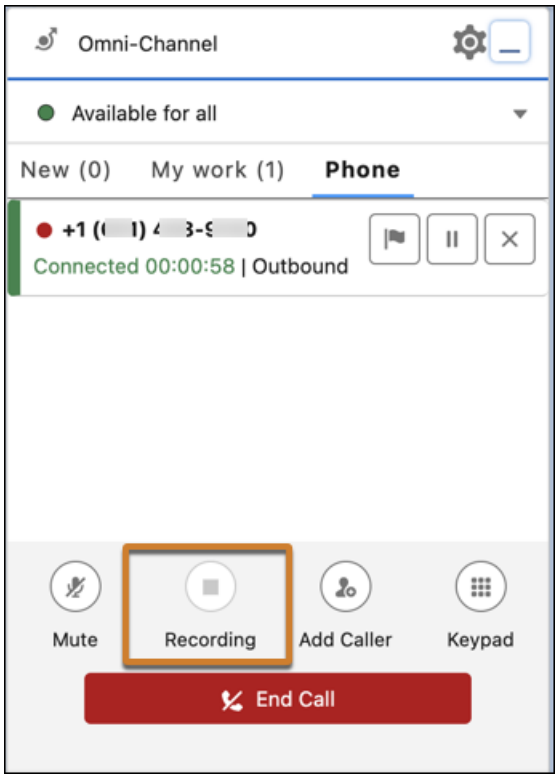
To run through this example:

- You must be an AWS admin with the ability to create flows in Amazon Connect.
- You must be a Salesforce admin.

To configure a flow so that voice calls aren't recorded:

1. From the Amazon Connect console, select **Routing > Flows**.
2. Select the name of the flow.
3. If the flow has a Set recording and analytics behavior block, confirm that **Call recording** is set to **Off**.
4. Add a Set contact attributes flow block to the flow, or edit an existing Set contact attributes block.
5. Add these attributes to the Set contact attributes flow block:
   - Namespace: User defined
   - Key: callRecordingDisabled
   - Select **Set manually** and set the value to *`true`*.

6. Save your changes.
   Whenever a call comes in through this flow, the voice call isn't recorded, and the Omni-Channel widget accurately reflects the recording state.



Enable the Voice Extension Page in Lightning App Builder

Configure this feature to let administrators add custom voice controls to their Omni-Channel softphones using the Voice Extension FlexiPage in the Lightning App Builder.

Ease of Implementation	Medium
Estimated Time to Implement	30 minutes

To use this feature, you must have Service Cloud Voice installed. This configuration applies to the following telephony models:

- Service Cloud Voice with Amazon Connect
- Service Cloud Voice with Partner Telephony
- Service Cloud Voice with Partner Telephony from Amazon Connect

Administrators can create and customize Voice Extension pages, then enable them in a contact center so that any rep in the org can use the feature. For more information about customizing call controls and voice extensions through the Voice Extension page in the Lightning App Builder, see Customize Call Controls and Voice Extensions.

Before an administrator can create Voice Extension pages, you must perform several configuration steps to expose the page in the Lightning App Builder and enable the component types you want the page to support.

The Voice Extension page supports the following Lightning App Builder components: Lightning Web Components, Aura, and Visualforce. You can further extend the Voice Extension feature by linking Aura components to the Service Cloud Voice Toolkit API, allowing reps to perform call actions, such as muting a call, placing a call on hold, and starting a preview call.

You can either perform the configuration directly in the org through the Developer Console, or build and bundle the custom component configuration files and deploy the managed package to AppExchange for administrators to download.

Configuring this feature includes:

1. Creating the Voice Extension FlexiPage metadata file
2. Configuring Lightning Web Components for Voice Extensions
3. Configuring Aura components for Voice Extensions
4. Linking components to the Service Cloud Voice Toolkit API
5. Linking components to the telephony system using Lightning Message Service
6. Deploying and distributing the managed package.

If you're creating custom configuration files for package distribution, use the sample quickstart templates in the /salesforce/scv-partner-telephony-quickstart/tree/main/force-app/main/default/flexipages folder in GitHub as a starting point.

Steps

To enable the Voice Extension page in the Lightning App Builder:

Step 1: Create the Voice Extension FlexiPage Metadata File

To create the Voice Extension FlexiPage Metadata file, add the VoiceExtension metadata type to the XML FlexiPage component definition file. See the FlexiPage page of the Metadata API Developer Guide for more information about the VoiceExtension FlexiPage metadata type. See the XML Configuration File Elements page of the Lightning Web Components Dev Guide for more information about the lightning__VoiceExtension target.

For example,

```
<?xml version="1.0" encoding="UTF-8"?>
<FlexiPage xmlns="http://soap.sforce.com/2006/04/metadata">
  …
  <type>VoiceExtension</type>
</FlexiPage>
```

See the Demo_Voice_Extension.flexipage-meta.xml file in GitHub for a sample implementation.

To refresh the sandbox, back up (`retrieve()`) the metadata before you perform the refresh and then import (`deploy()`) the metadata into the refreshed sandbox. See the Metadata API Developer Guide.

Step 2: Configure Lightning Web Components for Voice Extensions

To use a Lightning Web Component in the Voice Extension page, add the `lightning__VoiceExtension` target to the component's Lightning Web Component bundle. See the LightningComponentBundle page of the Metadata API Developer Guide and the Lightning Web Components Dev Guide for more information about the VoiceExtension metadata type.

For example,

```
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
  <apiVersion>57.0</apiVersion>
```

```
  ...
  <targets>
    <target>lightning__VoiceExtension</target>
  </targets>
  <capabilities>
    <capability>lightning__ServiceCloudVoiceToolkitApi</capability>
  </capabilities>
</LightningComponentBundle>
```

See the lwcVoiceExtension.js-meta.xml file in GitHub for a sample implementation.

Step 3: Configure Aura Components for Voice Extensions

To use an Aura component in the Voice Extension page, for each Aura component in the Voice Extension page, implement the `opencti:availableForVoiceExtension` marker interface. Since this is a marker interface, you don't need to implement any actions for the component. See the AuraVoiceExtension.cmp file in GitHub for a sample implementation.

For example,

```
<aura:component
implements="opencti:availableForVoiceExtension,flexipage:availableForAllPageTypes,flexipage:availableForRecordHome"
 access="global">
  ...
</aura:component>
```

See the Demo_Voice_Extension.flexipage-meta.xml file in GitHub for a sample implementation.

Step 4: Link the Aura Components to the Service Cloud Voice Toolkit API

You can further extend the Voice Extension feature by linking Aura components to the Service Cloud Voice Toolkit API, enabling reps to perform call operations, such as muting a call, placing a call on hold, and starting a preview call.

To link an Aura component to Service Cloud Voice, see the Service Cloud Voice Aura Toolkit API on page 117 page. For example,

For example,

```
<aura:component implements="flexipage:availableForRecordHome,force:hasRecordId"
access="global">
  <!-- subscribe to service cloud voice toolkit api -->
  <force:serviceCloudVoiceToolkitApi aura:id="voiceToolkitApi"/>
  <aura:handler name="init" value="{!this}" action="{!c.onInit}"/>
  <aura:handler name="destroy" value="{!this}" action="{!c.onDestroy}"/>
</aura:component>
```

See the SampleComponent.cmp file in GitHub for a sample implementation.

Step 5: Link the Components to the Telephony Connector Using Lightning Message Bridge

If you set up Service Cloud Voice with Partner Telephony, to enable communication between the components and the telephony connector, create a Lightning Message Bridge (LMB) component. See the Use the Lightning Message Bridge page for information.

If you set up Service Cloud Voice with Amazon Connect or Service Cloud Voice with Partner Telephony from Amazon Connect, skip this step.

Step 6: Deploy and Distribute the Managed Package

To distribute the configured files, create an SFDC project with the customized connectors, components, and metadata. Deploy the updated package so administrators can install it and start creating Voice Extension pages. See the Create a Service Cloud Voice Package page for information.

SEE ALSO:

Salesforce Help: Customize Call Controls and Voice Extensions

Examples from Amazon Connect

Review Service Cloud Voice implementation examples created by Amazon Connect's solutions team. The examples cover scenarios such as voicemail setup and external line routing for reps.

For more examples, see the Amazon Connect Campground for Service Cloud Voice in GitHub.

Note: Salesforce does not maintain the examples provided by Amazon Connect.

- **Voicemail Express V2**: Provides a basic voicemail capability for Service Cloud Voice. With Voicemail Express, customers can leave a voicemail for a rep or queue. The voicemail is stored in an S3 bucket as a .wav file, a transcription of the voicemail is created, and a new case is created in Salesforce that contains the transcription, a link to the recording, and a clickable callback number.
- **Follow Me Routing**: Provides an option that allows reps to flag themselves as available using an external phone number. When customers call specifically for that rep, the call is redirected to their external line.

CHAPTER 4 Using the Telephony Integration API

In this chapter ...

The Telephony Integration API allows you to programmatically manage a voice call. The provided Amazon Connect Lambda functions automatically use these endpoints to perform telephony functions, but you can also use these APIs separate from the provided Lambda functions.

Telephony Integration REST API Authorization

The Telephony Integration REST API requires JWT authorization. Usually, the provisioning process already sets up this authorization for you. This content simply provides instructions in case you must manually set up this authorization.

While creating a contact center instance with the Service Cloud Voice setup flow, a private/public key pair is auto-generated. The private key is stored as a secure string in AWS Systems Manager Parameter Store. The private key can be a 1024-, 2048, or 4096-bit RSA key length. We recommend an RSA key length of 2048. For contact centers using Amazon Connect as their telephony partner, the private key is stored as a secure string in AWS Systems Manager Parameter Store. Beginning in the Winter '26 release, private keys for contact centers using Amazon Connect are automatically migrated and stored in AWS Secrets Manager.

The public key is stored in the corresponding Salesforce CallCenter record. The private key is used to sign the JWT claim which must be included in the Bearer header for any HTTP request targeting the Service Cloud Voice REST API.

Before you begin, generate an RSA private key and a signed X509 certificate using OpenSSL.

Once you've satisfied this prerequisite, set up authorization for this API.

1. Construct a JWT header with this format: `{"alg":"RS256"}`.

   Example header:

   ```
   {
     "alg": "RS256",
     "typ": "JWT"
   }
   ```

2. Base64url encode the JWT header as defined in Base 64 Encoding with URL and Filename Safe Alphabet. For example: `eyJhbGciOiJSUzI1NiJ9`.
3. Construct a JSON claims set for the JWT with the following parameters:
   - **iat**—The issued at time is the time since the Unix epoch when the JWT token was issued, expressed as the number of seconds since `1970-01-01T0:0:0Z`.
   - **iss**—The issuer is the Salesforce org ID where you set up your contact center and registered the certificate.
   - **sub**—The subject must contain the Salesforce Call Center API Name.
   - **exp**—The expiration time of the assertion within 24 hours, expressed as the number of seconds from `1970-01-01T0:0:0Z`, measured in UTC.
   - **jti**—The unique ID for the JWT token. Recommended but not required.

     Note: Salesforce doesn't require JWT ID (JTI) claims in your JWT bearer tokens. However, if you pass a JTI claim in your JWT bearer token, Salesforce validates that the JTI claim hasn't been sent before. This validation prevents JWT replay attacks.

   Sample JSON claim set for the JWT:

   ```
   {
     "iat": 1756767613,
     "exp": 1756854000,
     "iss": "00DHs00000Kmpyr",
     "sub": "scv4cc1",
     "jti": "6af41080-8787-11f0-a7ad-736f74082154"
   }
   ```

4. Base64url encode the JWT claims set without any line breaks.
5. Sign the JWT using RSA SHA256 with the encoded header, encoded payload, and private key to generate a JWT token.

6. Use the generated JWT token as part of the Authorization header with the value as `Bearer {signedJwtToken}` when making the API request.

Create a Voice Call Record

Creates a voice call (`VoiceCall`) record containing the participants (that is, the caller and recipient) in the call.

When you create a voice call record, it creates a conversation in Salesforce. Only use this API in real time, which means that you should invoke this API only when a call is initiated.

This API can also be used to create voice call records for transfer by including the `parentVoiceCallId` parameter in the request payload.

URI

`/telephony/v1/voiceCalls`

HTTP Method

POST

Headers

Authorization: Bearer <token>

String. Standard header. The authorization token, where <token> is the JSON Web Token (JWT). Required.

Content-Type: <format>

String. Standard header. The format of the request body. Valid formats include JSON and XML. For example, `application/json` or `application/xml`. Required.

Telephony-Provider-Name: <telephony provider name>

String. Custom header. The name of the telephony provider that calls this API. For example, `Amazon Connect`.

Parameters

Property Name	Type	Description	Required
`callCenterApiName`	string	The API name of the contact center created in Salesforce associated with the voice call record.	Yes
`parentVoiceCallId`	string	For transfers, set this value to a Salesforce Voice Call record ID or the voice call's vendor call key (`vendorCallKey`) representing the parent record to this newly created voice call record. This parameter is required if initiationMethod=transfer Only use this property to create voice call records for transfer. Don't include this property when creating the initial voice call record.	No

Property Name	Type	Description	Required
`vendorCallKey`	string	A unique key that identifies the voice call record within the telephony system. For example, if the telephony system is Amazon Connect, this value is the contact ID in Amazon Connect. The vendorCallKey record can be associated with multiple VoiceCall records in Salesforce. For example, during a call transfer, if Rep 1 transfers a call to Rep 2, but Rep 2 isn't available, a second VoiceCall record is created with `VoiceCall.previousCallId` set to the unique ID of the previous call (`voiceCallId`). If Rep 1 transfers the call to yet another rep, a third VoiceCall record is created in the same way.	Yes
`to`	string	The phone number or email address to where the voice call is made. For inbound calls and transfers, set this value to the contact center's phone number or email address. If callSubtype is set to `PSTN`, set this value to the default phone number. If callSubtype is set to `webRTC`, set this value to the default email address. Alternatively, you can pass in the contact information from the third party telephony provider. • If the contact information comes from Amazon Connect, set this value to `WEBRTC_DEFAULT` and configure an Attributes block in the Amazon Connect inbound or transfer flow to pass in the email address. • If the contact information comes from another third party telephony provider, configure the system to pass in an identifying attribute that's relevant to the call, such as the email address or name. If not set, this value defaults to `WEBRTC_DEFAULT`.	Yes
`from`	string	The phone number or email address from where the voice call is made or transferred. • For inbound calls, set this value to the end user's phone number or email address. • For transfers, set this value to the contact center's phone number or email address. If callSubtype is set to `PSTN`, set this value to the default phone number. If callSubtype is set to `webRTC`, set this value to the default email address. Alternatively, you can pass in the contact information from the third party telephony provider.	Yes

Property Name	Type	Description	Required
		• If the contact information comes from Amazon Connect, set this value to `WEBRTC_DEFAULT` and configure an Attributes block in the Amazon Connect inbound or transfer flow to pass in the email address. • If the contact information comes from another third party telephony provider, configure the system to pass in an identifying attribute that's relevant to the call, such as the email address or name. If not set, this value defaults to `WEBRTC_DEFAULT`.	
`callOrigin`	string	Information about how this call originated. Possible values are: • Preview. Represents a preview dialer. • Progressive. Represents a progressive dialer. • Voicemail. Represents a voicemail call.	No
`initiationMethod`	string	The initiation method of the voice call. Possible values are: • Inbound. Represents a voice call that originated from an inbound call. • Transfer. Represents a voice call that was transferred from another rep. If `initiationMethod` is set to `Transfer`, `parentVoiceCallId` must also be set. Outbound calls aren't supported.	Yes
`queue`	string	Optional queue ID to associate with this voice call. This parameter is only used when `initiationMethod` is set to `Transfer`. It's ignored for other initiation methods. You can pass in either the Salesforce queue object ID or the vendor's external queue ID. Salesforce resolves this value using the entries in the CallCenterRoutingMap record. If the queue is resolved to a valid Salesforce queue, the VoiceCall owner is changed to the queue and the supervisor is able to see the transferred call waiting in the queue. See Map Your Salesforce Queues to Telephony Provider Queues.	No
`startTime`	string	The date and time (in UTC) when the voice call started.	Yes
`callSubtype`	string	The network or protocol over which the phone or Voice over Internet Protocol (VoIP) calls are made. Possible values are: • PSTN • WebRTC	No

Property Name	Type	Description	Required
		Set the value to `PSTN` (public switched telephone network) for phone calls. Set the value to `webRTC` (web real-time communication) for VoIP calls. The default value is PSTN, meaning voice calls are made from landlines and cell phones over the PSTN. Available in API version 62.0 and later.	

`participants`	array	The end-user participant on the voice call. Only one end user can be specified. If there are multiple people in the array, the first person is assigned as the end user.
		This property uses the following key-value pair, where `participantKey` is any unique identifier for the end user, and `type` is always "END_USER":
		`{ "participantKey": "UNIQUE ID FOR THE END USER", "type": "END_USER" }`
		For example,
		`{ "participantKey":"5324881f-1e84-4367-8930-f69a74b30ca6", "type": "END_USER" }`
		While `participantKey` takes the end user's unique identifier, Salesforce internally sets it to "END_USER" for privacy reasons.
			Yes

`callAttributes`	string	Represents additional standard and custom fields in the voice call record, where each key-value pair value corresponds to a standard or custom field and its values.
		Possible standard fields are:
		• SourceType - Represents the general purpose of the call. Valid values are: `Sales` and `Service`.
		• CallerContactInfo - Represents the recipient of the call.
		Here's an example using the standard field SourceType:
		`"callAttributes": "{\"SourceType\":\"Sales\"}"`
		Here's an example using a custom field named "Department__c":
		`"callAttributes": "{\"Department__c\":\"Support\", \"Sales\"}"`
			No

Example

Request:

```
POST /telephony/v1/voiceCalls

{
  "callCenterApiName": "MyContactCenter",
  "vendorCallKey": "5324881f-1e84-4367-8930-f69a74b30ca6",
  "to": "+14152988103",
  "from": "+18669483147",
  "initiationMethod": "Inbound",
  "startTime": "2019-07-02T17:32:28Z",
  "callSubtype": "PSTN",
  "participants": [
    {
      "participantKey": "4081456688",
      "type" : "END_USER"
    }
  ],
  "callAttributes": "{\"MyField__c\": \"abc\"}"
}
```

Response:

```
{
  "voiceCallId": "0LQRM0000006CSz",
  "errors": []
}
```

Note: You may encounter errors when creating a voice call record. These errors can be caused by a number of factors due to triggers, flows, required fields, and field validations. Salesforce bypasses the errors, creates the voice call record, and returns an `errors` response like the one in the following example.

```
{
  "voiceCallId": "0LQS70000004YDG",
  "errors": "We created your voice call but there were some errors : Required fields
are missing:
             [Picklist], Text__c: data value too large: sample123456890"
}
```

If you encounter an `errors` response when creating a voice call record, notify your Salesforce administrator.

SEE ALSO:

Object Reference for Salesforce and Lightning Platform: VoiceCall

Update a Voice Call Record

Updates a voice call (`VoiceCall`) record.

Use this API to update the parameters of a voice call record. The endpoint is an asynchronous operation. You can't query for the status of the API call. This endpoint can also be used to create a voice call even after the call has ended. This behavior is useful in scenarios

where you want to log a record in Salesforce for abandoned or missed calls, or for any other scenario where a voice call wasn't already created.

Note: Don't run this API call more than one time. If you run this API call more than one time, each subsequent API call overwrites the following properties with the latest values: totalHoldDuration, numberOfHolds, and agentInteractionDuration.

This API call functions by making two separate database updates. The first database update passes the parameters `recordingLocation` and `totalRecordingDuration`, which are used to create a corresponding `VoiceCallRecording` record. The second database update passes all the other parameters. If you're using the Update a Voice Call Record API call as an Apex trigger, the trigger is invoked only by the second database update.

URI

`/telephony/v1/voiceCalls/{CALL ID}`

Where `{CALL ID}` is the Salesforce voice call ID (`voiceCallId`) or the telephony vendor's contact ID.

HTTP Method

PATCH

Headers

Authorization: Bearer <token>

String. Standard header. The authorization token, where <token> is the JSON Web Token (JWT). Required.

Content-Type: <format>

String. Standard header. The format of the request body. Valid formats include JSON and XML. For example, `application/json` or `application/xml`. Required.

Telephony-Provider-Name: <telephony provider name>

String. Custom header. The name of the telephony provider that calls this API. For example, `Amazon Connect`.

Parameters

Property Name	Type	Description	Required
`startTime`	string	The date and time (in UTC) when the voice call started. The VoiceCall.CallDurationInSeconds value is only updated if both `startTime` and `endTime` are passed.	No
`endTime`	string	The date and time (in UTC) when the voice call ended. The VoiceCall.CallDurationInSeconds value is only updated if both	No

Property Name	Type	Description	Required
		`startTime` and `endTime` are passed.	
`enqueueTime`	string	The date and time (in UTC) when the voice call was placed in the queue.	No
`acceptTime`	string	The date and time (in UTC) when the rep accepted the voice call.	No
`fromNumber`	string	The number from which the voice call was made. For example, for an inbound call, this value is the end user's phone number.	No
`totalHoldDuration`	int32	The total length of time (in seconds) the rep put the voice call on hold. Salesforce sets this value to zero (0) if no value is passed. If you set this value, you must also set the recording location (`recordingLocation`). If you run this API call more than one time after the voice call, each subsequent API call overwrites the `totatlHoldDuration` property with the latest value.	No
`longestHoldDuration`	int32	Within the entire call, the longest length of time (in seconds) the rep put the voice call on hold.	No
`numberOfHolds`	int32	The total number of holds within the entire call. Salesforce sets this value to zero (0) if no value is passed. If you run this API call more than one time after the voice call, each subsequent API call overwrites the `numberOfHolds` property with the latest value.	No

Property Name	Type	Description	Required
`queue`	string	The name of the queue that the rep who serviced the voice call belonged to.	No
`agent`	string	The rep's username.	No
`agentInteractionDuration`	int32	The total length of time (in seconds) of the rep interaction. Salesforce sets this value to zero (0) if no value is passed. If you set this value, you must also set the recording location (recordingLocation). If you run this API call more than one time after the voice call, each subsequent API call overwrites the `agentInteractionDuration` property with the latest value.	No
`callOrigin`	string	Information about how this call originated. Possible values are: • Preview. Represents a preview dialer. • Progressive. Represents a progressive dialer. • Voicemail. Represents a voicemail call.	No
`recordingLocation`	string	The location of the voice call recording if call recording was enabled. **NOTE:** This field is required when updating any fields related to a recording.	No
`totalRecordingDuration`	int32	The total length of time (in seconds) of the voice call recording. If a value isn't passed, this value defaults to the sum of `agentInteractionDuration + totalHoldDuration`.	
`callAttributes`	string	Use this property to pass fields for the voice call record that must be populated as part of the	No

Property Name	Type	Description	Required
		creation process. The value is a string representation using key-value pairs in JSON, where each key-value pair corresponds to a custom field and its value.	
`isActiveCall`	boolean	Indicates whether the conversation is still open and active (`true`). The default value is `false`. If set to `true`, the voice call conversation is still open, and any real-time transcription messages sent through the `POST` /telephony/v1/voiceCalls/{voiceCallId}/messages API will continue to be accepted. Any pending service routing records associated with the voice call remain pending. VoiceCall.CallDisposition remains "in-progress." If set to `false`, the voice call conversation is closed if VoiceCall.NextCallId is NULL. Any pending service routing records associated with the voice call are cleared. VoiceCall.CallDisposition is set to "completed," and the VoiceCall.CustomerHoldDuration and VoiceCall.NumberOfHolds values are reset to zero (0).	No
`disconnectReason`	string	The reason why the voice call was disconnected. If the parameter is passed in with the payload, the `value` and `isError` attributes are required, where `value` is the message stating how the call was disconnected, and `isError` is set to `true`. For example, `disconnectReason: { "value": "TELECOM_PROBLEM",`	No

Property Name	Type	Description	Required
		<pre> "isError": true }</pre> Set the `disconnectReason` field to `CALLBACK` when a customer drops a call after accepting the callback offer. Without this setting the initial call can be incorrectly classified as abandoned. For Amazon Connect instances, see DisconnectReason in the Amazon Connect contact records data model page for a list of possible reasons why a voice call may be disconnected.	

Example

Request:

```
PATCH /telephony/v1/voiceCalls/0LQRM0000006CSz

{
  "startTime": "2020-08-26T21:21:14Z",
  "endTime": "2019-08-26T21:21:34Z",
  "enqueueTime": "2019-08-26T21:21:34Z",
  "acceptTime": "2019-08-26T21:21:24Z",
  "numberOfHolds": 20,
  "longestHoldDuration": 10,
  "callAttributes": "{\"dateCustomField__c\": \"2019-08-28T21:21:34Z\", \"checkbox__c\":
false}",
  "disconnectReason": {
    "value": "TELECOM_PROBLEM",
    "isError": true
  }
}
```

Response:

```
{
  "status": "pending"
}
```

SEE ALSO:

Object Reference for Salesforce and Lightning Platform: VoiceCall

Create a Transcript

Creates a transcribed version of a voice call in real time.

Use this API to create transcripts for one voice call in real time. Voice calls can include inbound calls, transfer calls, outbound calls, callback calls, and consult calls. The transcript data is sent to the rep console synchronously in real time and will be persisted in Salesforce asynchronously.

This API and the Create Transcripts in Bulk API (`/telephony/v1/voiceCalls/messages`) share several characteristics.

- Both APIs create voice call transcripts if the timestamp of the transcript is between the start and end times of the conversation.
- Both APIs pause call transcription whenever the rep pauses the recording or places the call participant on hold. Any messages being sent in won't be created or stored by the API during this time.

Where the two APIs differ is by the number of transcripts they can process per request. This API only creates transcripts for one voice call at a time, while the Create Transcripts in Bulk API creates transcripts in bulk.

To create or update transcripts in bulk *after* the call has ended, use the Salesforce Connect API.

URI

`/telephony/v1/voiceCalls/{vendorCallKey}/messages`

Where `vendorCallKey` is the unique ID of the voice call (VoiceCall) record within the telephony system. For example, if the telephony system is Amazon Connect, this value is the contact ID in Amazon Connect.

In transfer use cases where there are multiple VoiceCall objects associated to a single call, use the vendorCallKey of the first voice call record created for the call. This value can only be the unique call identifier of the first contact for the conversation. This value can't be a Salesforce `voiceCallId`.

In consult call use cases where a consult call hasn't merged with a multiparty call, use the vendorCallKey of the voice call record created for the consult call. To transcribe a consult call after it has merged with a multiparty call, use the vendorCallKey of the first voice call record created for the multiparty call.

HTTP Method

POST

Headers

Authorization: Bearer <token>

String. Standard header. The authorization token, where <token> is the JSON Web Token (JWT). Required.

Content-Type: <format>

String. Standard header. The format of the request body. Valid formats include JSON and XML. For example, `application/json` or `application/xml`. Required.

Telephony-Provider-Name: <telephony provider name>

String. Custom header. The name of the telephony provider that calls this API. For example, `Amazon Connect`.

Parameters

Property Name	Type	Description	Required
`participantId`	string	The call participant that uttered this text. • If senderType is END_USER, set this value to `{vendorCallKey}END_USER`, where {vendorCallKey} is the vendor call key (vendorCallKey) of the first voice call (VoiceCall) record created for the conversation. For consult calls that haven't been merged with a multiparty call, {vendorCallKey} is the vendorCallKey of the VoiceCall created for the consult call. If you set this property to any other value, Salesforce will set it to `{vendorCallKey}END_USER` internally for privacy reasons. • If senderType is EXTERNAL_USER, set this value to the unique ID for the external call participant generated by the telephony partner. • If senderType is HUMAN_AGENT, enter the rep's 15-digit or 18-digit Salesforce user ID (starts with 005). • If senderType is SUPERVISOR, set this value to `{vendorCallKey}SUPERVISOR`, where {vendorCallKey} is the vendor call key (vendorCallKey) of the first voice call (VoiceCall) record created for the conversation. If you set this property to any other value, Salesforce will set it to	Yes

Property Name	Type	Description	Required
		{vendorCallKey}SUPERVISOR internally for privacy reasons. • If senderType is VIRTUAL_AGENT, set this value to {vendorCallKey}VIRTUAL_AGENT, where {vendorCallKey} is the vendor call key (vendorCallKey) of the first voice call (VoiceCall) record created for the conversation. If you set this property to any other value, Salesforce will set it to {vendorCallKey}VIRTUAL_AGENT internally for privacy reasons.	
`messageId`	string	The unique ID of the message. This value has to be unique for the given conversation.	Yes
`startTime`	int64	The date and time (in UTC) when this utterance started. Measured in milliseconds since the Unix epoch.	Yes
`endTime`	int64	The date and time (in UTC) when this utterance ended. Measured in milliseconds since the Unix epoch.	Yes
`content`	string	The actual text of the utterance.	Yes
`senderType`	string	The voice call participant role that generated the utterance. Possible values are: • END_USER - Represents a customer. • EXTERNAL_USER - Represents a contact external to your Salesforce organization. • HUMAN_AGENT - Represents a rep. • SUPERVISOR - Represents a supervisor. For example, a	Yes

Property Name	Type	Description	Required
		supervisor generates an utterance when they barge into a call between a customer and rep and start talking. • VIRTUAL_AGENT - Set this value to VIRTUAL_AGENT when you have a conference between multiple reps and you want to create all rep-side utterances as one single rep since you have no way to differentiate between the different reps in the audio stream. The sender in this case must be the vendor call key (`vendorCallKey`) of the first voice call record created for the conversation.	

Example

Request:

```
POST telephony/v1/voiceCalls/5324881f-1e84-4367-8930-f69a74b30ca6/messages

{
  "participantId": "5324881f-1e84-4367-8930-f69a74b30ca6VIRTUAL_AGENT",
  "messageId": "57904eb6-5352-4c5e-adf6-5f100572cf5d116",
  "startTime": 1573503300000,
  "endTime": 1573503300000,
  "content": "Hello All",
  "senderType": "VIRTUAL_AGENT"
}
```

Request:

```
POST telephony/v1/voiceCalls/5324881f-1e84-4367-8930-f69a74b30ca6/messages

{
  "participantId": "005123456789abc",
  "messageId": "57904eb6-5352-4c5e-adf6-5f100572cf5d116",
  "startTime": 1573503300000,
  "endTime": 1573503300000,
  "content": "Hello All",
  "senderType": "HUMAN_AGENT"
}
```

Request:

```
POST telephony/v1/voiceCalls/5324881f-1e84-4367-8930-f69a74b30ca6/messages

{
  "participantId": "a12355ce-03be-4335-a28891293sz",
  "messageId": "57904eb6-5352-4c5e-adf6-5f100572cf5d116",
  "startTime": 1573503300000,
  "endTime": 1573503300000,
  "content": "Hello All",
  "senderType": "EXTERNAL_USER"
}
```

Response:

```
{
  "result": "Accepted"
}
```

SEE ALSO:

Create Transcripts in Bulk

Upload or Update Transcripts with Connect REST API

Create Transcripts in Bulk

Creates transcripts in bulk across multiple voice calls in real time. These voice calls can include inbound calls, transfer calls, outbound calls, callback calls, and consult calls.

This API allows you to process transcripts quickly by creating multiple transcripts across different voice calls in one request. The transcript data is sent to the rep console synchronously in real time and persists in Salesforce asynchronously.

If your telephony provider can aggregate multiple transcripts across voice calls, use this API for faster processing, resulting in faster delivery to reps in real time. If your telephony provider is Amazon Connect and you use Contact Lens for Amazon Connect, this API is by default used to process your transcripts.

This API and the Create a Transcript API (`/telephony/v1/voiceCalls/{vendorCallKey}/messages`) share several characteristics.

- Both APIs create voice call transcripts if the timestamp of the transcript is between the start and end times of the conversation.
- Both APIs pause call transcription whenever the rep pauses the recording or places the call participant on hold. Any messages being sent in aren't created or stored by the API during this time.

The APIs also differ in terms of their capacity.

- The Create a Transcript API creates a transcript for one voice call only.
- This Create Transcripts in Bulk API lets you create transcripts for up to 5 different voice calls at a time. Each voice call included in the payload can have up to 5 transcripts (also called conversation entries).

To make sure conversation entries are stored in chronological order for this API, insert transcript data one participant at a time. For example, include a one-sided conversation with entries from the end user in the messages object. For a sender type of VIRTUAL_AGENT, include conversation entries from multiple reps in the messages object, in order of utterance.

Since this API can contain transcripts for multiple voice calls, if a rep from one voice call pauses a recording or places a call participant on hold, transcription stops for just that voice call; the hold doesn't interrupt transcription processing for the other voice calls.

To create or update transcripts in bulk *after* the call has ended or to upload a binary file of transcripts up to 512 MB, use the Salesforce Connect API.

URI

`/telephony/v1/voiceCalls/messages`

HTTP Method

POST

Headers

Authorization: Bearer <token>

String. Standard header. The authorization token, where <token> is the JSON Web Token (JWT). Required.

Content-Type: <format>

String. Standard header. The format of the request body. Valid formats include JSON and XML. For example, `application/json` or `application/xml`. Required.

Telephony-Provider-Name: <telephony provider name>

String. Custom header. The name of the telephony provider that calls this API. For example, `Amazon Connect`.

Parameters

Property Name	Type	Description	Required
`entries`	array	An array of create transcript objects for one or more voice call (`VoiceCall`) records.	Yes
`vendorCallKey`	string	The unique ID of the voice call (`VoiceCall`) record within the telephony system for which messages need to be created.	Yes
`messages`	array	An array of individual transcript objects for the voice call (`VoiceCall`) record.	Yes
`participantId`	string	The call participant that uttered this text. • If senderType is END_USER, set this value to `{vendorCallKey}END_USER`, where {vendorCallKey} is the vendor call key (vendorCallKey) of the first voice call (VoiceCall) record	Yes

Property Name	Type	Description	Required
		created for the conversation. For consult calls that haven't been merged with a multiparty call, {vendorCallKey} is the vendorCallKey of the VoiceCall created for the consult call. If you set this property to any other value, Salesforce sets it to `{vendorCallKey}END_USER` internally for privacy reasons. • If senderType is EXTERNAL_USER, set this value to the unique ID for the external call participant generated by the telephony partner. • If senderType is HUMAN_AGENT, enter the rep's 15-digit or 18-digit Salesforce user ID (starts with 005). • If senderType is SUPERVISOR, set this value to `{vendorCallKey}SUPERVISOR`, where {vendorCallKey} is the vendor call key (vendorCallKey) of the first voice call (VoiceCall) record created for the conversation. If you set this property to any other value, Salesforce will set it to `{vendorCallKey}SUPERVISOR` internally for privacy reasons. • If senderType is VIRTUAL_AGENT, set this value to `{vendorCallKey}VIRTUAL_AGENT`, where {vendorCallKey} is the vendor call key (vendorCallKey) of the first voice call (VoiceCall) record	

Property Name	Type	Description	Required
		created for the conversation. If you set this property to any other value, Salesforce sets it to `{vendorCallKey}VIRTUAL_AGENT` internally for privacy reasons.	
`messageId`	string	The unique ID of the message. This value has to be unique for the given conversation.	Yes
`startTime`	int64	The date and time (in UTC) when this utterance started. Measured in milliseconds since the Unix epoch.	Yes
`endTime`	int64	The date and time (in UTC) when this utterance ended. Measured in milliseconds since the Unix epoch.	Yes
`content`	string	The actual text of the utterance.	Yes
`senderType`	string	The voice call participant role that generated the utterance. Possible values are: • END_USER - Represents a customer. • EXTERNAL_USER - Represents a contact external to your Salesforce organization. • HUMAN_AGENT - Represents a rep. • SUPERVISOR - Represents a supervisor. For example, a supervisor generates an utterance when they barge into a call between a customer and rep and start talking. • VIRTUAL_AGENT - Set this value to VIRTUAL_AGENT when you have a conference between multiple reps and you want to create all rep-side utterances as one	Yes

Property Name	Type	Description	Required
		single rep since you have no way to differentiate between the different reps in the audio stream. The sender in this case must be the vendor call key (`vendorCallKey`) of the first voice call (`VoiceCall`) record created for the conversation.	

Example

Request:

```
POST /telephony/v1/voiceCalls/messages

{
  "entries":  [
    {
      "vendorCallKey": "abc955ce-03be-4335-a28891293zs-12267",
      "messages":  [
        {
          "startTime": 1770307118000,
          "endTime": 1770307120500,
          "participantId": "abc955ce-03be-4335-a28891293zs-12267END_USER",
          "messageId": "a09955ce-03be4335xz-a28891293zs",
          "senderType": "END_USER",
          "content": "Hi, I need help with my plane reservation."
        },
        {
          "startTime": 1770307125000,
          "endTime": 1770307128200,
          "participantId": "abc955ce-03be-4335-a28891293zs-12267END_USER",
          "messageId": "c09955ce-03be4335xz-a28891293zu",
          "senderType": "END_USER",
          "content": "Yes, it was canceled."
        },
        {
          "startTime": 1770307132000,
          "endTime": 1770307134100,
          "participantId": "abc955ce-03be-4335-a28891293zs-12267END_USER",
          "messageId": "d09955ce-03be4335xz-a28891293zw",
          "senderType": "END_USER",
          "content": "Do you have anything direct?"
        },
        {
          "startTime": 1770307139000,
          "endTime": 1770307142500,
          "participantId": "abc955ce-03be-4335-a28891293zs-12267END_USER",
```

```
        "messageId": "f09955ce-03be4335xz-a28891293zz",
        "senderType": "END_USER",
        "content": "Yes, that should work."
      },
      {
        "startTime": 1770307147000,
        "endTime": 1770307149800,
        "participantId": "abc955ce-03be-4335-a28891293zs-12267END_USER",
        "messageId": "g09955ce-03be4335xz-a28891293zx",
        "senderType": "END_USER",
        "content": "Thanks for your help."
      }
    ]
  },
  {
    "vendorCallKey": "b09955ce-03be-4335-a28891293zs-18895",
    "messages":  [
      {
        "startTime": 1770307800000,
        "endTime": 1770307802800,
        "participantId": "b09955ce-03be-4335-a28891293zs-188955END_USER",
        "messageId": "1b9955ce-03be4335xz-a28891293zu",
        "senderType": "END_USER",
        "content": "Hi, I'd like to ask about an upgrade."
      },
      {
        "startTime": 1770307808200,
        "endTime": 1770307811900,
        "participantId": "b09955ce-03be-4335-a28891293zs-18895END_USER",
        "messageId": "b23455ce-03be4335xz-a28891293zu",
        "senderType": "END_USER",
        "content": "Yes, for my trip next month."
      },
      {
        "startTime": 1770307818000,
        "endTime": 1770307820500,
        "participantId": "b09955ce-03be-4335-a28891293zs-18895END_USER",
        "messageId": "b09955ce-03be4335xz-a28891293zu",
        "senderType": "END_USER",
        "content": "Perfect, thanks."
      }
    ]
  },
  {
    "vendorCallKey":  "m07855ce-03be-4335-a28891293zs-78543",
    "messages":  [
      {
        "startTime": 1770308400000,
        "endTime": 1770308403200,
        "participantId": "m07855ce-03be-4335-a28891293zs-78543END_USER",
        "messageId": "m86655ce-03be4335xz-a28891293zw",
        "senderType": "END_USER",
        "content": "I have a complaint about my meal."
      },
```

```
        {
          "startTime": 1770308409500,
          "endTime": 1770308412100,
          "participantId": "m07855ce-03be-4335-a28891293zs-78543END_USER",
          "messageId": "n5998ce-03be4335xz-a28891293zw",
          "senderType": "END_USER",
          "content": "It was so delicious I wish it was bigger."
        }
      ]
    },
    {
      "vendorCallKey": "k18968ce-03be-4335-a28891293zs-98736",
      "messages":  [
        ...
      ]
    },
    {
      "vendorCallKey": "s79758ce-03be-4335-a28891293zs-48346",
      "messages":  [
        ...
      ]
    }
  ]
}
```

Request:

```
POST /telephony/v1/voiceCalls/messages

{
  "entries": [
    {
      "vendorCallKey": "5324881f-1e84-4367-8930-f69a74b30ca6",
      "messages": [
        {
          "startTime": 1770309300000,
          "endTime": 1770309302500,
          "participantId": "5324881f-1e84-4367-8930-f69a74b30ca6VIRTUAL_AGENT",
          "messageId": "8hjcd32b-bb06-477b-8947-74660b4db74g",
          "senderType": "VIRTUAL_AGENT",
          "content": "My colleague in the service department has joined our call."
        },
        {
          "startTime": 1770309309000,
          "endTime": 1770309312200,
          "participantId": "5324881f-1e84-4367-8930-f69a74b30ca6VIRTUAL_AGENT",
          "messageId": "6gjcd32b-bb06-477b-8947-74660b4db74g",
          "senderType": "VIRTUAL_AGENT",
         "content": "Hi, I'm going to help you with your service appointment rescheduling."

        },
        {
          "startTime": 1770309319000,
          "endTime": 1770309321800,
          "participantId": "5324881f-1e84-4367-8930-f69a74b30ca6VIRTUAL_AGENT",
```

```
          "messageId": "4jfcd32b-bb06-477b-8947-74660b4db74g",
          "senderType": "VIRTUAL_AGENT",
          "content": "I'll let our service department take it from here."
        }
      ]
    },
    {
      "vendorCallKey": "5324881f-1e84-4367-8930-f69a74b30ca6",
      "messages": [
        {
          "startTime": 1770309480000,
          "endTime": 1770309483000,
          "participantId": "5324881f-1e84-4367-8930-f69a74b30ca6VIRTUAL_AGENT",
          "messageId": "1efcd32b-bb06-477b-8947-74660b4db74m",
          "senderType": "VIRTUAL_AGENT",
          "content": "Thank you. Just one minute while we look up your account number."
        },
        {
          "startTime": 1770309489500,
          "endTime": 1770309492100,
          "participantId": "5324881f-1e84-4367-8930-f69a74b30ca6VIRTUAL_AGENT",
          "messageId": "9sdcd32b-bb06-477b-8947-74660b4db74m",
          "senderType": "VIRTUAL_AGENT",
          "content": "We're both here to help your with your request. Can you provide more
 information?"
        }
      ]
    },
    {
      "vendorCallKey": "9273881f-1e84-4367-8930-f69a74b30ab8",
      "messages": [
        {
          "startTime": 1770309660000,
          "endTime": 1770309664500,
          "participantId": "9273881f-1e84-4367-8930-f69a74b30ab8VIRTUAL_AGENT",
          "messageId": "1efcd32b-bb06-477b-8947-74660b4db74z",
          "senderType": "VIRTUAL_AGENT",
          "content": "Thank you for your patience. Now that my teammate has joined, let's
 address your issue."
        }
      ]
    }
  ]
}
```

Response:

```
// All transcripts were successfully processed.
{
```

```
  "result": "Accepted"
}
```

SEE ALSO:

- ContactLensConsumerFunction Lambda Function
- ContactLensProcessorFunction Lambda Function
- Create a Transcript
- Upload or Update Transcripts with Connect REST API

Execute an Omni-Channel Flow

Executes the Omni-Channel flow to route voice calls. It passes the call ID (Salesforce VoiceCallId or telephony vendor ContactId) as parameters to the flow and returns the rep or queue routing instructions to the contact flow. By default, Service Cloud Voice uses the Omni-Channel flow (or fallback queue) specified for the phone channel that matches the dialed number. If the dialed number doesn't match an existing phone channel, you can optionally set a new dialed number, Omni-Channel flow, and fallback queue as input parameters to this API call.

URI

`/telephony/v1/voiceCalls/{CALL ID}/omniFlow`

Where `{CALL ID}` is the Salesforce `voiceCallId` or the telephony vendor's contact ID.

HTTP Method

PATCH

Headers

Authorization: Bearer <token>

String. Standard header. The authorization token, where <token> is the JSON Web Token (JWT). Required.

Content-Type: <format>

String. Standard header. The format of the request body. Valid formats include JSON and XML. For example, `application/json` or `application/xml`. Required.

Telephony-Provider-Name: <telephony provider name>

String. Custom header. The name of the telephony provider that calls this API. For example, `Amazon Connect`.

Parameters

Property Name	Type	Description	Required
`dialedNumber`	string	Set a new dialed number to account for a dialed number that doesn't have a phone channel set up. Dialed number is the phone number called.	Yes

Property Name	Type	Description	Required
`flowDevName`	string	Set the Omni-Channel flow using its developer name.	No
`fallbackQueue`	string	Set the fallback queue using its Salesforce queue developer name or ID.	No
`flowInputParameters`	string	Set additional inputs to the Omni-Channel flow.	No

Important: Service Cloud Voice uses this order of precedence to route voice calls.

1. Uses the Omni-Channel Flow and Fallback Queue settings for the phone channel that matches the dialed number. The flow takes precedence. If the flow fails, the fallback queue is used.
2. Uses the `flowDevName` and `fallbackQueue` parameters specified in the `Execute Omniflow` API call.

If none of this criteria is satisfied or the admin doesn't map the Salesforce and telephony queues, the call fails.

Example

Request:

```
PATCH /telephony/v1/voiceCalls/0LQRM0000006CSz/omniFlow
{
    "dialedNumber": "+18445791189"
    "flowDevName": "Route_VoiceCall",
    "fallbackQueue": "00G111222333444",
    "flowInputParameters": {
        "Input1": "one",
        "Input2": "two"
    }
}
```

Response without errors:

```
{
  "agent": "[AGENT_INFO]",
  "queue": "[QUEUE_INFO]"}
```

Note: AGENT_INFO and QUEUE_INFO correspond to the ExternalId field in the CallCenterRoutingMap

Response with errors:

```
{
  "errors": ["error1", "error2"]
}
```

Route a Voice Call

Routes a voice call to a rep, agent, queue, or flow if Omni-Channel Unified Routing is enabled.

Use this API to route a voice call or transfer a voice call with Omni-Channel to a rep or agent, queue, or Omni-Channel flow. When a rep selects a transfer target in Omni-Channel, this API passes the call ID (Salesforce VoiceCallId or telephony vendor ContactId) as a parameter.

Unlike the Execute an Omni-Channel Flow API on page 92, this API provides a single integration point for routing and transferring voice calls with Unified Routing.

The Route a Voice Call API returns only the routing request status of success or failure. Use this API to instruct Omni Channel to route the voice call to a specific target, and don't depend on the response.

This API is processed if the call belongs to a contact center that has Unified Routing with Omni-Channel enabled.

URI

`/telephony/v1/voiceCalls/{CALL ID}/routeVoiceCall`

Where `{CALL ID}` is the Salesforce `voiceCallId` or the telephony vendor's contact ID.

HTTP Method

POST

Headers

Authorization: Bearer <token>

String. Standard header. The authorization token, where <token> is the JSON Web Token (JWT). Required.

Content-Type: <format>

String. Standard header. The format of the request body. Valid formats include JSON and XML. For example, `application/json` or `application/xml`. Required.

Telephony-Provider-Name: <telephony provider name>

String. Custom header. The name of the telephony provider that calls this API. For example, `Amazon Connect`.

Parameters

Property Name	Type	Description	Required
`routingTarget`	string	Specifies the Omni-Channel routing target as a Salesforce agent ID, queue ID, or Omni-Channel flow ID.	Yes
`fallbackQueue`	string	Specifies the fallback queue by its Salesforce queue ID. Used if routing to a flow fails.	No
`flowInputParameters`	string	Specifies more inputs to the Omni-Channel flow, if `routingTarget` is a flow.	No

Example

Request:

```
POST /telephony/v1/voiceCalls/89328b83-ff42-4c85-a2af-e948124365de/routeVoiceCall

{
  "routingTarget": "300SG000015iJ38YAE",
  "fallbackQueue": "00G111222333444",
  "flowInputParameters": {
      "Input1": "one",
      "Input2": "two"
  }
}
```

Response:

The API returns the routing status only. The Omni Channel routing engine routes the call to a specific user, queue, or flow.

```
{
  "status": "Success"
}
```

Request a Callback

Creates a callback request for a voice call if Omni-Channel Unified Routing is enabled.

Invoke this API to request a callback from Salesforce by creating a ContactRequest record and returning its recordId. The ContactRequest can then be routed through Omni-Channel Unified Routing in Salesforce.

The Request a Callback API will only be processed if the call belongs to a contact center which has Unified Routing with Omni-Channel enabled.

URI

`/telephony/v1/voiceCalls/{CALL ID}/requestCallback`

Where `{CALL ID}` is the Salesforce `voiceCallId` or the telephony vendor's contact ID.

HTTP Method

POST

Headers

Authorization: Bearer <token>
String. Standard header. The authorization token, where <token> is the JSON Web Token (JWT). Required.

Content-Type: <format>
String. Standard header. The format of the request body. Valid format is JSON. For example, `application/json`. Required.

Telephony-Provider-Name: <telephony provider name>
String. Custom header. The name of the telephony provider that calls this API. For example, `Amazon Connect`.

Parameters

Property Name	Type	Description	Required
`callbackNumber`	string	Set the number that the customer requested to be called back at.	Yes

Example

Request:

```
POST /telephony/v1/voiceCalls/89328b83-ff42-4c85-a2af-e948124365de/requestCallback

{
  "callbackNumber": "+18445791189"
}
```

Response:

```
{
  "recordId": "0LQRM0000006CSz"
}
```

Send a Real-Time Conversation Event

Send real-time conversation events generated from intelligence sources to the rep console.

Send real-time conversation events generated from intelligence sources to the rep console so administrators can create rules based on the signals to trigger Einsten Next Best Action or auto-launched flows. The signals are also sent to the Service Cloud Voice Toolkit API.

URI

`/telephony/v1/voiceCalls/{vendorCallKey}/realtimeConversationEvents`

Where `vendorCallKey` is the unique ID of the voice call (VoiceCall) record within the telephony system. For example, if the telephony system is Amazon Connect, this value is the contact ID in Amazon Connect. In transfer use cases where there are multiple VoiceCall objects associated with a single call, use the vendorCallKey of the first voice call record created for the call. This value can only be the unique call identifier of the first contact for the conversation. This value can't be a Salesforce `voiceCallId`.

HTTP Method

POST

Headers

Authorization: Bearer <token>

String. Standard header. The authorization token, where <token> is the JSON Web Token (JWT). Required.

Content-Type: <format>
String. Standard header. The format of the request body. Valid formats include JSON and XML. For example, `application/json` or `application/xml`. Required.

Telephony-Provider-Name: <telephony provider name>
String. Custom header. The name of the telephony provider that calls this API. For example, `Amazon Connect`.

Parameters

Property Name	Type	Description	Required
`service`	string	The intelligence source for the events. This is a partner-provided value. For example, set this value to *`AmazonConnectContactLens`* for Amazon Connect, *`CXoneAgentAssistService`* for NICE, or *`VonageConversationalInsights`* for Vonage. Contact your Salesforce representative to verify that your partner contact center supports this feature.	Yes
`events`	array	A piece of signal data generated in real time from a voice call between a rep and end user. Events must be sorted in chronological ascending order by startTime. For example, this snippet displays a piece of signal data. `{ "type": "IntelligenceSignal__Category", "value": "CustomerAngry", "startTime": 1573503450 }`	Yes
`events.type`	string	The signal type of the event. Possible values are: • IntelligenceSignal__Category • IntelligenceSignal__Sentiment	Yes
`events.value`	string	If `events.type` is *`IntelligenceSignal__Category`*, this is the value of the signal generated in real time by the telephony vendor. For example, it could be a Contact Lens rule defined in Amazon Connect. If `events.type` is *`IntelligenceSignal__Sentiment`*, this represents the sentiment score for the participant (event.participant) in the call. The score is used to evaluate rep performance over time and identify coaching opportunities. The score must be between -1000 and 1000, with 1000 being the best, most positive score possible. The value is rounded up to two (2) decimal points.	Yes

Property Name	Type	Description	Required
`events.startTime`	int64	The date and time (in UTC) when this event started. Measured in milliseconds since the Unix epoch.	Yes
events.score	string	The confidence score, which evaluates how confident the vendor is about the accuracy of the value (events.value). The score must be between 0 and 1000, where 1000 means the vendor is very confident that the signal is accurate. This value comes from the partner telephony service provider.	No
event.participant	string	The voice call participant type that generated the signal. Possible values are: • AGENT - Represents a rep. • CUSTOMER - Represents a customer. This field is only required if `events.type` is *`IntelligenceSignal__Sentiment`*.	Depends. See description.

Example

Request:

```
POST /telephony/v1/voiceCalls/5324881f-1e84-4367-8930-f69a74b30ca6/realtimeConversationEvents

{
  "service": "AmazonConnectContactLens",
  "events": [
    {
      "type": "IntelligenceSignal__Category",
      "value": "OrderStatusCheckingHelp",
      "startTime": 1573503300
    },
    {
      "type": "IntelligenceSignal__Category",
      "value": "CustomerAngry",
      "startTime": 1573503450
    }
  ]
}
```

Response:

```
{
  status: “202 Accepted”
}
```

Example

Request:

```
POST /telephony/v1/voiceCalls/5324881f-1e84-4367-8930-f69a74b30ca6/realtimeConversationEvents

{
 "service":"CxoneAgentAssistService",
  "persist":false,
  "events":   [
    {
      "type": "IntelligenceSignal__Sentiment",
      "value": "400.13",
      "startTime": 16898867880000,
      "score": 10,
      "participant": "CUSTOMER"
    }
  ]
}
```

Response:

```
{
  status: “202 Accepted”
}
```

Store a Post-Call Conversation Event

Store post-call conversation events generated from intelligence sources, such as Amazon Connect Contact Lens, into the conversation data store. Users can also specify the target objects (for example, VoiceCall, ConversationEntry) related to the specific event.

Salesforce uses these stored events and relationships to display call insights after a call ends. The data can also be linked to other Salesforce standard objects, such as VoiceCall and ConversationEntry.

This API should only be invoked after a conversation closes.

URI

`/telephony/v1/voiceCalls/{vendorCallKey}/postConversationEvents`

Where `vendorCallKey` is the unique ID of the voice call (VoiceCall) record within the telephony system. For example, if the telephony system is Amazon Connect, this value is the contact ID in Amazon Connect. In transfer use cases where there are multiple VoiceCall objects associated to a single call, use the vendorCallKey of the first voice call record created for the call. This value can only be the unique call identifier of the first contact for the conversation. This value can't be a Salesforce `voiceCallId`.

HTTP Method

POST

Headers

Authorization: Bearer <token>

String. Standard header. The authorization token, where <token> is the JSON Web Token (JWT). Required.

Content-Type: <format>

String. Standard header. The format of the request body. Valid formats include JSON and XML. For example, `application/json` or `application/xml`. Required.

Telephony-Provider-Name: <telephony provider name>

String. Custom header. The name of the telephony provider that calls this API. For example, `Amazon Connect`.

Parameters

Property Name	Type	Description	Required
`service`	string	The intelligence source of the events. Possible values are: • AmazonConnectContactLens	Yes
`events`	array	A piece of signal data generated post-call from a voice call between a rep and end user. This data is used for post-call event analysis. Events must be sorted in chronological ascending order by startTime. For example, the following snippet displays event data used for post-call analysis: `{ "type": "IntelligenceSignal__Sentiment", "value": "-1000", "startTime": 1573503301, "endTime": 1573503320, "participant": "CUSTOMER" }`	Yes
`events.type`	string	The signal type of the event. Possible values are: • IntelligenceSignal__Sentiment	Yes
`events.value`	string	The value of the signal detected. For type IntelligenceSignal__Sentiment, the value represents the sentiment of the words spoken by the participants in the call, ranging between -1000 (most negative) and 1000 (most positive). A value of zero (0) represents a neutral sentiment.	Yes
`events.startTime`	int64	The date and time (in UTC) when this event started. Measured in milliseconds since the Unix epoch.	Yes
`events.endTime`	int64	The date and time (in UTC) when this event ended. Measured in milliseconds since the Unix epoch.	Yes

Property Name	Type	Description	Required
`events.participant`	string	The voice call participant type that generated the signal. Possible values are: • AGENT - Represents a rep. • CUSTOMER - Represents a customer.	Yes

Example

Request:

```
POST /telephony/v1/voiceCalls/5324881f-1e84-4367-8930-f69a74b30ca6/postConversationEvents

{
  "service": "AmazonConnectContactLens",
  "events": [ // sort by startTime
    {
      "type": "IntelligenceSignal__Sentiment",
      "value": "1000",
      "startTime": 1573503300,
      "endTime": 1573532140,
      "participant": "AGENT"
     },
     {
      "type": "IntelligenceSignal__Sentiment",
      "value": "-1000",
      "startTime": 1573503301,
      "endTime": 1573503320,
      "participant": "CUSTOMER"
    }
  ]
}
```

Response:

```
{
  status: “202 Accepted”
}
```

Reroute a Voice Call

Reroutes a voice call through Omni-Channel if Unified Routing with Omni-Channel is enabled.

Use this API to reroute a voice call with Omni-Channel if routing has failed initially. For example, routing may fail if an assigned rep isn't part of the Amazon Connect routing queue or Amazon Connect isn't able to start a WebRTC connection with a rep's browser. This API closes any agentWork that is already accepted and reroutes the call to a different rep. The Reroute a Voice Call API will only be processed if the call belongs to a contact center which has Unified Routing with Omni-Channel enabled.

URI

`/telephony/v1/voiceCalls/{CALL ID}/reroute`

Where `{CALL ID}` is the Salesforce `voiceCallId` or the telephony vendor's contact ID.

HTTP Method

POST

Headers

Authorization: Bearer <token>

String. Standard header. The authorization token, where <token> is the JSON Web Token (JWT). Required.

Content-Type: <format>

String. Standard header. The format of the request body. Valid formats include JSON and XML. For example, `application/json` or `application/xml`. Required.

Telephony-Provider-Name: <telephony provider name>

String. Custom header. The name of the telephony provider that calls this API. For example, `Amazon Connect`.

Parameters

None

Example

Request:

```
POST /telephony/v1/voiceCalls/89328b83-ff42-4c85-a2af-e948124365de/reroute
```

Response:

```
{"status": "Success"}
```

Clear Routing

Deletes the PendingServiceRouting (PSR) record for a voice call. This API doesn't need to be called for most scenarios; the PSR record is automatically deleted when the call is no longer being routed. However, there are some scenarios, like for missed or abandoned calls when using partner telephony systems (excluding Amazon Connect), where you must explicitly call this API to clear the PSR record.

URI

`/telephony/v1/voiceCalls/{CALL ID}/clearRouting`

Where `{CALL ID}` is the Salesforce `voiceCallId` or the telephony vendor's contact ID.

HTTP Method

PATCH

Headers

Authorization: Bearer <token>

String. Standard header. The authorization token, where <token> is the JSON Web Token (JWT). Required.

Content-Type: <format>

String. Standard header. The format of the request body. Valid formats include JSON and XML. For example, `application/json` or `application/xml`. Required.

Telephony-Provider-Name: <telephony provider name>

String. Custom header. The name of the telephony provider that calls this API. For example, `Amazon Connect`.

Parameters

None.

SEE ALSO:

Salesforce Help: The Routing Lifecycle

Telephony Integration REST API Sample Code

The following JavaScript code sample performs the authorization process and then invokes the Voice API.

Note: If enhanced domains aren't enabled in your org, your URL format is different. For details, see My Domain URL Formats in Salesforce Help.

```
const jwt = require('jsonwebtoken');
const SSM = require('aws-sdk/clients/ssm');
const uuid = require('uuid/v1');

// SCRT EndPoint Base example: https://MyDomainName.my.salesforce-scrt.com/telephony/v1
// Contact your admin to get the exact URL. This should have been configured as part
// of the Service Cloud Voice Setup.

const axios = require('axios').create({
    baseURL: "ScrtEndpointBase"
});

// Example function to retrieve privateKey parameter using Amazon Systems Manager
async function getSSMParameterValue(paramName, withDecryption) {
    return await new Promise(resolve => {
        const ssm = new SSM();
        const query = {
            Names:  [paramName],
            WithDecryption: withDecryption
        };
```

```
        ssm.getParameters(query, (err, data) => {
            let paramValue = null;

            if (!err && data && data.Parameters && data.Parameters.length) {
                paramValue = data.Parameters[0].Value;
            }

            resolve(paramValue);
        });
    });
}

// Generate a JWT based on the specified parameters.
async function generateJWT(params) {
    const { privateKeyParamName, orgId, callCenterApiName, expiresIn } = params;
    // Retrieve privateKey from the param store
    const privateKey = await getSSMParameterValue(privateKeyParamName, true);
    const signOptions = {
        issuer: orgId,
        subject:  callCenterApiName,
        expiresIn:  expiresIn,
        algorithm:  'RS256',
        jwtid: uuid()
    };

    return jwt.sign({}, privateKey, signOptions);
}

// Function to Create VoiceCall
async function createVoiceCall(fieldValues) {
    const generateJWTParams = {
        privateKeyParamName   : "YourPrivateKeyParamName",
        orgId                 : "YourSalesforceOrganizationId",
        callCenterApiName     : "YourSalesforceCallCenterAPIName",
        expiresIn             : "TokenExpiryTime"
    };

    const jwt = await generateJWT(generateJWTParams);

    var fieldValues = {
        "vendorCallKey":"5324881f-1e84-4367-8930-f69a74b30ca6",
        "callCenterApiName":"HVCC",
        "to":"8002345678",
        "from":"4081456688",
        "initiationMethod":"Inbound",
        "startTime":"2020-07-13T11:43:01Z",
        "participants":
            [{"participantKey":"+18033568299",
            "type": "END_USER"
            }]
    };

    const response = await axios.post('/voiceCalls', fieldValues, {
        headers: {
```

```
            'Authorization': 'Bearer ${jwt}',
            'Content-Type': 'application/json',
            'Telephony-Provider-Name': 'Amazon Connect'
        }
    })
    .then(response => {
        return response;
    })
    .catch(error => {
        let context = {};
        if (error.response) {
            // The request was made and the server responded with
            // a status code that falls out of the range of 2xx
            context = error.response.data;
        } else if (error.request) {
            // The request was made but no response was received
            context = error.request;
        } else {
            // Something happened in setting up the request that triggered an error
            context = error.message;
        }
         throw new Error('Error creating VoiceCall record');
    });

    return response.data;
}
```

CHAPTER 5 Service Cloud Voice Toolkit API

In this chapter ...

Service Cloud Voice Toolkit API is a collection of front-end event listeners and API methods for interacting with the Service Cloud Voice product. The API is available for Lightning components built using both the Lightning Web Component framework and the Aura framework.

This toolkit is available to all customers. All customers may use this toolkit to interact with Einstein Next Best Action. Some portions of this toolkit rely on features available only through Service Cloud Voice.

Note: Components (such as the Lightning Message Bridge) that implement the `lightning:backgroundUtilityItem` interface aren't supported locations for the Service Cloud Voice Toolkit API. Use this API in a component that resides on a page, or resides in a visible utility bar component.

Service Cloud Voice Lightning Web Component (LWC) Toolkit API

The Service Cloud Voice Toolkit API allows you to build Lightning web components that have access to Service Cloud Voice features.

This component is supported in Lightning Experience. It requires API version 52.0 or later.

To use the Service Cloud Voice Toolkit API in your component:

1. Add the `lightning__ServiceCloudVoiceToolkitApi` component to your component's HTML template file.

```
// yourComponent.html

<template>
  <lightning-service-cloud-voice-toolkit-api>
  </lightning-service-cloud-voice-toolkit-api>
  ...
</template>
```

2. Include the `lightning__ServiceCloudVoiceToolkitApi` target in your component's configuration file.

```
// yourComponent.js-meta.xml

<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
  <apiVersion>52.0</apiVersion>
  <isExposed>true</isExposed>
  <capabilities>
    <capability>lightning__ServiceCloudVoiceToolkitApi</capability>
  </capabilities>
</LightningComponentBundle>
```

For more information about configuration files, see XML Configuration File Elements.

Service Cloud Voice LWC Toolkit API Telephony Events

Make your LWC components context-aware by listening to events that take place during phone calls.

The following events are available.

Event Name	Description
audiostats	Sent during a call, publishing real-time WebRTC audio statistics every 30 seconds.
callstarted	Sent when the call starts.

Event Name	Description
callconnected	Sent when the call connects.
callended	Sent when the call ends.
hold	Sent when the call is put on hold.
resume	Sent when the call resumes after hold.
mute	Sent when the call is muted.
unmute	Sent when the call is unmuted.
participantadded	Sent when a participant is added to the call.
participantremoved	Sent when a participant is removed from the call.
conference	Sent when participants on a three-way call are all taken off hold.
swap	Sent when participants on a three-way call have their hold status swapped.
pauserecording	Sent when the rep pauses call recording.
resumerecording	Sent when the rep resumes call recording.
transcript	Sent when a new utterance is received by the transcription system.
wrapupended	Sent when the rep exits after call work status.
flagraise	Sent when a rep, while communicating with a customer, quietly raises a flag to get help from a supervisor.
flaglower	Sent when a rep or supervisor lowers a raised flag to cancel the request for supervisor help.
note	Sent when a rep or supervisor sends a message without raising a flag. For example, an event is sent when a user jots down notes but doesn't raise a flag.

To subscribe to these events, add an event listener for each event you want to listen to.

```
// Subscribe
const toolkitApi = this.template.querySelector('lightning-service-cloud-voice-toolkit-api');
toolkitApi.addEventListener('callstarted', <listener>);

// Unsubscribe
const toolkitApi = this.template.querySelector('lightning-service-cloud-voice-toolkit-api');
toolkitApi.removeEventListener('callstarted', <listener>);
```

When an event occurs, you receive a JSON payload that contains the event type, along with any relevant data. For instance, the `callstarted` event contains the following payload.

```
{
  "type": "callstarted",
  "detail": {
```

```
    "callId": "d7a9f1b7-fc17-43fe-8ca7-e584a2b34792",
    "callType": "outbound",
    "participant": "+1 (415) 999-0000"
  }
}
```

All payloads contain some basic information, such as the event type (type) and a unique call identifier (callId). Events can contain additional values within the detail object. Refer to the following table for all the possible properties that can appear within the detail object of the payload.

Event Name	Payload "detail" Properties
audiostats	• `callId` (string): Unique ID of the voice call within the telephony system. For example, if the telephony system is Amazon Connect, this value is the contact ID. • `stats` (string): Media type. Possible values are: "inputChannelStats" and "outputChannelStats". • `packetsLost` (int): Number of packets lost during transmission. • `packetsCount` (int): Number of packets transmitted. • `jitterBufferMillis` (int): Time it took to buffer incoming packets, measured in milliseconds. • `roundTripTimeMillis` (int): Time it took for the packet to travel to and from the receiver, measured in milliseconds. • `statsCount` (int): Number of times the statistics have been collected or reported. For internal use only.
callstarted	• `callId` (string): Unique call identifier value. • `callType` (string): "inbound" or "outbound". • `participant` (string): Phone number.
callconnected	• `callId` (string): Unique call identifier value. • `callType` (string): "inbound" or "outbound". • `participant` (string): Phone number.
callended	• `callId` (string): Unique call identifier value.
hold	• `callId` (string): Unique call identifier value. • `participant` (string): Phone number or Amazon Quick Connect name.
resume	• `callId` (string): Unique call identifier value. • `participant` (string): Phone number or Amazon Quick Connect name.

Event Name	Payload "detail" Properties
mute	• `callId` (string): Unique call identifier value.
unmute	• `callId` (string): Unique call identifier value.
participantadded	• `callId` (string): Unique call identifier value. • `participant` (string): Phone number or Amazon Quick Connect name.
participantremoved	• `callId` (string): Unique call identifier value. • `participant` (string): Phone number or Amazon Quick Connect name.
conference	• `callId` (string): Unique call identifier value.
swap	• `callId` (string): Unique call identifier value.
pauserecording	• `callId` (string): Unique call identifier value.
resumerecording	• `callId` (string): Unique call identifier value.
wrapupended	• `callId` (string): Unique call identifier value.
transcript	• `id` (string): Unique identifier for the message. • `clientSentTimestamp` (number): Date and time (in UTC) when the client sent the content. Measured in milliseconds since the Unix epoch. • `serverReceivedTimestamp` (number): Date and time (in UTC) when the server received the content. Measured in milliseconds since the Unix epoch. • `content` (object): Information about the content. • `content.formatType` (string): Content format type of the transcript. Possible value is "Text". • `content.text` (string) The body text of the transcript. • `callId` (string): Unique ID of the voice call within the telephony system. For example, if the telephony system is Amazon Connect, this value is the contact ID. • `sender` (object): Information about the user whose action initiated the event. • `sender.role` (string): Role of the user whose action initiated the event. Possible values are "Agent" and "Supervisor".

Event Name	Payload "detail" Properties
	For example: `{ "type": "transcript", "detail": { "id": "3115b389-ab50-400e-b8ba-978b7ec51d7a" "clientSentTimestamp": 1594944652299, "serverReceivedTimestamp": 1594944652328, "content":{ "formatType": "Text", "text": "Hello" }, "callId": "c5d93c19-e03b-44f8-85e6-a11f02e70c45", "sender": { "role": "Agent" } } }`
flagraise	• `id` (string): Unique call identifier value. • `clientSentTimestamp` (number): Date and time (in UTC) when the client sent the content. Measured in milliseconds since the Unix epoch. • `serverReceivedTimestamp` (number): Date and time (in UTC) when the server received the content. Measured in milliseconds since the Unix epoch. • `content` (object): Information about the content. • `content.formatType` (string): Content format type of the raised flag. Possible value is "Text". • `content.text` (string) The body text of the raised flag. • `callId` (string): Unique conversation ID of the voice call within the telephony system. For example, if the telephony system is Amazon Connect, this value is the contact ID. For transferred calls, the conversation ID is the primary call ID. • `sender` (object): Information about the user whose action initiated the event. • `sender.role` (string): Role of the user whose action initiated the event. Possible values are "Agent" and "Supervisor". • `sender.displayName` (string): Name of the user whose action initiated the event. • `voiceCallId` (string): Unique identifier of the Salesforce voice call in which a rep raised the flag. For transferred calls,

Event Name	Payload "detail" Properties

	the new call segment has a different voiceCallId from the original call.
	For example:
	<pre>{
	"type": "flagraise",
	"detail": {
	"id": "3115b389-ab50-400e-b8ba-978b7ec51d7a"
	"clientSentTimestamp": 1594944652299,
	"serverReceivedTimestamp": 1594944652328,
	"content":{
	"formatType": "Text",
	"text": "The customer wants a credit note."
	},
	"callId": "c5d93c19-e03b-44f8-85e6-a11f02e70c45",
	"sender": {
	"role": "Agent",
	"displayName": "Angela Agent"
	},
	"voiceCallId": "0LQRM0000006CSz"
	}</pre>

flaglower	• `id` (string): Unique call identifier value. • `clientSentTimestamp` (number): Date and time (in UTC) when the client sent the content. Measured in milliseconds since the Unix epoch. • `serverReceivedTimestamp` (number): Date and time (in UTC) when the server received the content. Measured in milliseconds since the Unix epoch. • `content` (object): Information about the content. • `content.formatType` (string): This value is `NULL` because it doesn't apply to the flaglower event. • `content.text` (string) This value is `NULL` because it doesn't apply to the flaglower event. • `callId` (string): Unique conversation ID of the voice call within the telephony system. For example, if the telephony system is Amazon Connect, this value is the contact ID. For transferred calls, the conversation ID is the primary call ID. • `sender` (object): Information about the user whose action initiated the event. • `sender.role` (string): Role of the user whose action initiated the event. Possible values are "Agent" and "Supervisor".

Event Name	Payload "detail" Properties
	• `sender.displayName` (string): Name of the user whose action initiated the event. • `voiceCallId` (string): Unique identifier of the Salesforce voice call in which a rep lowered the flag. For transferred calls, the new call segment has a different voiceCallId from the original call. For example: `{ "type": "flaglower", "detail": { "id": "3115b389-ab50-400e-b8ba-978b7ec51d7a" "clientSentTimestamp": 1594944652299, "serverReceivedTimestamp": 1594944652328, "content":{ "formatType": NULL, "text": NULL }, "callId": "c5d93c19-e03b-44f8-85e6-a11f02e70c45", "sender": { "role": "Supervisor", "displayName": "Samir Supervisor" }, "voiceCallId": "0LQRM0000006CSz" }`
note	• `id` (string): Unique call identifier value. • `clientSentTimestamp` (number): Date and time (in UTC) when the client sent the content. Measured in milliseconds since the Unix epoch. • `serverReceivedTimestamp` (number): Date and time (in UTC) when the server received the content. Measured in milliseconds since the Unix epoch. • `content` (object): Information about the content. • `content.formatType` (string): Content format type of the note. Possible value is "Text". • `content.text` (string) The body text of the note. • `callId` (string): Unique ID of the voice call within the telephony system. For example, if the telephony system is Amazon Connect, this value is the contact ID. • `sender` (object): Information about the user whose action initiated the event.

Event Name	Payload "detail" Properties
	• `sender.role` (string): Role of the user whose action initiated the event. Possible values are "Agent" and "Supervisor". • `sender.displayName` (string): Name of the user whose action initiated the event. For example: `{ "type": "note", "detail": { "id": "3115b389-ab50-400e-b8ba-978b7ec51d7a" "clientSentTimestamp": 1594944652299, "serverReceivedTimestamp": 1594944652328, "content":{ "formatType": "Text", "text": "The customer wants a refund or store credit." }, "callId": "c5d93c19-e03b-44f8-85e6-a11f02e70c45", "sender": { "role": "Agent", "displayName": "Angela Agent" } }`

Payloads may contain other system-generated information.

Service Cloud Voice LWC Toolkit API Telephony Actions

Make your components context-aware by listening to and raising events during phone calls.

Programmatically raise softphone call control events or even build your own softphone UI. For example, build integrations with peripherals like Bluetooth headsets, or programmatically perform actions on the rep's behalf, such as adding a participant to a conference call.

The following methods are available.

Method Name	Description	Arguments
mute	Mute the call.	None.
unmute	Unmute the call.	None.
hold	Put the call on hold.	**Name: participantType** **Type**: string

Method Name	Description	Arguments
		Description: The participant type whose call was put on hold. Possible values are "Initial_Caller" and "Third_Party".
resume	Resume the call if it's on hold.	**Name: participantType** **Type**: string **Description**: The participant type whose call was resumed. Possible values are "Initial_Caller" and "Third_Party".
endCall	End the call.	None.
acceptCall	Accept the call.	None.
declineCall	Decline the call.	None.
pauseRecording	Pause the call recording.	None.
resumeRecording	Resume the call recording.	None.
startPreviewCall	Start the preview call.	**Name: phoneNumber** **Type**: string **Description**: The phone number for the preview call.
merge	Resume the call for all participants on a three-way call.	None.
swap	Swap two of the participants on a three-way call, where one participant is on hold. The call for the participant on hold resumes while the other participant is put on hold.	None.
addParticipant	Add a third participant to an existing call between a rep and a caller.	**Name: contactType** **Type**: string **Description**: The transfer destination type for addParticipant provided by the customer. Possible values are "PhoneNumber" and "AgentOrQueueId". **Name: contactValue** **Type**: string **Description**: The transfer destination value for addParticipant provided by the customer. Possible values are the phone number if `contactType` is "PhoneNumber", or the agent ID or queue ID if `contactType` is "AgentOrQueueId".

Method Name	Description	Arguments
		Name: isBlindTransfer **Type**: boolean **Description**: Indicates whether the participant was added through a blind transfer. If `true`, the participant was added through a blind transfer. For Service Cloud Voice with Amazon Connect and Service Cloud Voice with Partner Telephony from Amazon Connect, the only accepted value is `false`.
sendDigits	Send digits to the active call.	**Name: digit** **Type**: string **Description**: The digits of the phone number sent by the rep. Possible values are `0–9`, `*`, `#`, `A`, `B`, `C`, `D`, and comma (`,`).
endCall	End the call.	**Name: participantType** **Type**: string **Description**: The participant who ended the call. Possible values are "Initial_Caller", "Third_Party", and "Agent".

To perform telephony actions, load the Service Cloud Voice Lightning Web Components `ToolkitApi`, and then add a telephony method for each call control you want to allow administrators to perform.

For example, add the following components to accept calls, decline calls, end calls, pause call recordings, and resume paused recordings.

```
({
   onAcceptCall() {
     this.getToolkitApi().acceptCall();
   }
   onDeclineCall() {
     this.getToolkitApi().declineCall();
   }
   onEndCall() {
     this.getToolkitApi().endCall(this.comboBoxRemoveParticipantValue);
   }
   onPauseRecording() {
     this.getToolkitApi().pauseRecording();
   }
   onResumeRecording() {
     this.getToolkitApi().resumeRecording();
   }
})
```

See the Toolkit API sampleLWCComponent.js file in GitHub for more examples.

Service Cloud Voice LWC Toolkit Next Best Action

Give your reps recommendations in real time with Next Best Action.

To incorporate Next Best Action into your Service Cloud Voice solution, follow these instructions.

1. Build a recommendation strategy that uses the filter element and $Request variable. The $Request variable is a map containing key value pairs. For example, if the recommendation strategy receives the data {intent: "refund"} then $Request.intent evaluates to "refund". To learn more, see Write an Expression.
2. Enable real time recommendations in your Actions & Recommendations deployment and select the strategy you just built.
3. Invoke Einstein Next Best Action with a record ID and key:value pairs in the updateNextBestActions() function parameters. Recommendations are displayed for record pages matching the provided ID. The value given to a particular key can be retrieved in the Recommendation Strategy using $Request.key in the filter element.

The following filter element expression shows recommendations that contain a particular keyword in the recommendation name. To learn more, see Write an Expression.

```
const toolkitApi = this.template.querySelector('lightning-service-cloud-voice-toolkit-api');
toolkitApi.updateNextBestActions(<recordId>, <payload>);
```

Note: Next Best Action has usage-based entitlements. Invoking recommendations in Service Cloud Voice counts towards that usage.

Service Cloud Voice Aura Toolkit API

The Service Cloud Voice Toolkit API allows you to build Aura components that have access to Service Cloud Voice features.

In order to protect our customers' data, Service Cloud Voice Toolkit API methods and event listeners are available only to Aura components built or installed within the org's namespace. Custom components built within the org and components installed using unmanaged packages will function properly. Custom components installed through managed packages, such as on the AppExchange, are not able to use the Service Cloud Voice Aura Toolkit API. In order to use the Toolkit API in a managed package, refer to the Service Cloud Voice Lightning Web Component (LWC) Toolkit API on page 107.

Service Cloud Voice Aura Toolkit API Telephony Events

Make your components context-aware by listening to events that take place during phone calls.

The following telephony events are available.

Event Name	Description
AUDIO_STATS	Sent during a call, publishing real-time WebRTC audio statistics every 30 seconds.
CALL_STARTED	Sent when the call starts.
CALL_CONNECTED	Sent when the call connects.
CALL_ENDED	Sent when the call ends.
HOLD	Sent when the call is put on hold.
RESUME	Sent when the call resumes after hold.
MUTE	Sent when the call is muted.
UNMUTE	Sent when the call is unmuted.
PARTICIPANT_ADDED	Sent when a participant is added to the call.
PARTICIPANT_REMOVED	Sent when a participant is removed from the call.
CONFERENCE	Sent when participants on a three-way call are all taken off hold.
SWAP	Sent when participants on a three-way call have their hold status swapped.
PAUSE_RECORDING	Sent when the rep pauses call recording.
RESUME_RECORDING	Sent when the rep resumes call recording.
WRAP_UP_ENDED	Sent when the rep exits after call work status.

To subscribe to these events, add a telephony event listener for each event you want to listen to.

```
// Subscribe
cmp.find('voiceToolkitApi')
   .addTelephonyEventListener('CALL_STARTED', telephonyEventListenerFunc);

// Unsubscribe
cmp.find('voiceToolkitApi')
   .removeTelephonyEventListener('CALL_STARTED', telephonyEventListenerFunc);
```

When an event occurs, you receive a JSON payload that contains the event type, along with any relevant data. For instance, the CALL_STARTED event contains this payload.

```
{
  "type": "CALL_STARTED",
  "detail": {
    "callId": "d7a9f1b7-fc17-43fe-8ca7-e584a2b34792",
    "callType": "outbound",
    "participant": "+1 (415) 999-0000"
  }
}
```

All payloads contain some basic information, such as the event type (type) and a unique call identifier (callId). Events can contain additional values within the detail object. Refer to the table for all the possible properties that can appear within the detail object of the payload.

Event Name	Payload "detail" Properties
AUDIO_STATS	• `callId` (string): Unique ID of the voice call within the telephony system. For example, if the telephony system is Amazon Connect, this value is the contact ID. • `stats` (string): Media type. Possible values are: "inputChannelStats" and "outputChannelStats". • `packetsLost` (int): Number of packets lost during transmission. • `packetsCount` (int): Number of packets transmitted. • `jitterBufferMillis` (int): Time it took to buffer incoming packets, measured in milliseconds. • `roundTripTimeMillis` (int): Time it took for the packet to travel to and from the receiver, measured in milliseconds. • `statsCount` (int): Number of times the statistics have been collected or reported. For internal use only.
CALL_STARTED	• `callId` (string): Unique call identifier value. • `callType` (string): "inbound" or "outbound". • `participant` (string): Phone number.
CALL_CONNECTED	• `callId` (string): Unique call identifier value. • `callType` (string): "inbound" or "outbound". • `participant` (string): Phone number. • `callAttributes` (object): additional call attributes provided by the telephony provider For example: `{ "type": "CALL_CONNECTED", "detail": { "callId": "4ba06a63-f1d3-44b6-a9ff-f57143a37c1", "callType": "outbound", "participant": "+1 (90 274-0615", "callAttributes":{ "isTranscribed": "true", "languageCode": "en-US" } } }` The structure of the `callAttributes` varies depending on the telephony provider. Not all values of

Event Name	Payload "detail" Properties
	`callAttributes` have the properties shown in the example.
CALL_ENDED	• `callId` (string): Unique call identifier value.
HOLD	• `callId` (string): Unique call identifier value. • `participant` (string): Phone number or Amazon Quick Connect name.
RESUME	• `callId` (string): Unique call identifier value. • `participant` (string): Phone number or Amazon Quick Connect name.
MUTE	• `callId` (string): Unique call identifier value.
UNMUTE	• `callId` (string): Unique call identifier value.
PARTICIPANT_ADDED	• `callId` (string): Unique call identifier value. • `participant` (string): Phone number or Amazon Quick Connect name.
PARTICIPANT_REMOVED	• `callId` (string): Unique call identifier value. • `participant` (string): Phone number or Amazon Quick Connect name
CONFERENCE	• `callId` (string): Unique call identifier value .
SWAP	• `callId` (string): Unique call identifier value.
PAUSE_RECORDING	• `callId` (string): Unique call identifier value.
RESUME_RECORDING	• `callId` (string): Unique call identifier value.
WRAP_UP_ENDED	• `callId` (string): Unique call identifier value .

Payloads may contain other system-generated information.

Service Cloud Voice Aura Toolkit API Telephony Actions

Make your components context-aware by listening to and raising events during phone calls.

Programmatically raise softphone call control events or even build your own softphone UI. For example, build integrations with peripherals like Bluetooth headsets, or programmatically perform actions on the rep's behalf, such as adding a participant to a conference call.

The following methods are available.

Method Name	Description	Arguments
mute	Mute the call.	None.
unmute	Unmute the call.	None.
hold	Put the call on hold.	**Name: participantType** **Type**: string **Description**: The participant type whose call was put on hold. Possible values are "Initial_Caller" and "Third_Party".
resume	Resume the call if it's on hold.	**Name: participantType** **Type**: string **Description**: The participant type whose call was resumed. Possible values are "Initial_Caller" and "Third_Party".
endCall	End the call.	None.
acceptCall	Accept the call.	None.
declineCall	Decline the call.	None.
pauseRecording	Pause the call recording.	None.
resumeRecording	Resume the call recording.	None.
startPreviewCall	Start the preview call.	**Name: phoneNumber** **Type**: string **Description**: The phone number for the preview call.
merge	Resume the call for all participants on a three-way call.	None.
swap	Swap two of the participants on a three-way call, where one participant is on hold. The call for the participant on hold resumes while the other participant is put on hold.	None.
addParticipant	Add a third participant to an existing call between a rep and a caller.	**Name: contactType** **Type**: string **Description**: The transfer destination type for addParticipant provided by the

Method Name	Description	Arguments
		customer. Possible values are "PhoneNumber" and "AgentOrQueueId". **Name: contactValue** **Type**: string **Description**: The transfer destination value for addParticipant provided by the customer. Possible values are the phone number if `contactType` is "PhoneNumber", or the agent ID or queue ID if `contactType` is "AgentOrQueueId". **Name: isBlindTransfer** **Type**: boolean **Description**: Indicates whether the participant was added through a blind transfer. If `true`, the participant was added through a blind transfer.
sendDigits	Send digits to the active call.	**Name: digit** **Type**: string **Description**: The digits of the phone number sent by the rep. Possible values are `0–9`, `*`, `#`, `A`, `B`, `C`, `D`, and comma (`,`).
endCall	End the call.	**Name: participantType** **Type**: string **Description**: The participant who ended the call. Possible values are "Initial_Caller", "Third_Party", and "Agent".

To perform telephony actions, load the Service Cloud Voice Aura Toolkit `voiceToolkitApi`, and then add a telephony method for each call control you want to allow admins to perform.

For example, add the following components to let admins accept calls, end calls, decline calls, pause a call recording, and resume a paused recording.

```
({
    acceptCall: function(cmp) {
        cmp.find('voiceToolkitApi').acceptCall()
    },

    endCall: function(cmp) {
        cmp.find('voiceToolkitApi').endCall()
    },
```

```
    declineCall: function(cmp) {
        cmp.find('voiceToolkitApi').declineCall()
    },

    pauseRecording: function(cmp) {
        cmp.find('voiceToolkitApi').pauseRecording()
    },

    resumeRecording: function(cmp) {
        cmp.find('voiceToolkitApi').resumeRecording()
    }
    startPreviewCall: function(cmp) {
        var params = cmp.get('v.phoneNumber');
        cmp.find('voiceToolkitApi').startPreviewCall(params);
    }
})
```

See the Toolkit API SampleComponentHelper.js file in GitHub for more examples.

Service Cloud Voice Aura Toolkit API Conversation Events

Listen to events related to a conversation.

The following conversation events are available.

Event Name	Description
FLAG_RAISE	Sent when a rep, while communicating with a customer, quietly raises a flag to get help from a supervisor.
FLAG_LOWER	Sent when a rep or supervisor lowers a raised flag to cancel the request for supervisor help.
NOTE	Sent when a rep or supervisor sends a message without raising a flag. For example, an event is sent when a user jots down notes but doesn't raise a flag.
TRANSCRIPT	Sent when a new utterance is received by the transcription system.
INTELLIGENCE_SIGNAL	Sent when a new intelligence signal is received.

To subscribe to an event, add a conversation event listener. For example, add the following event listener to subscribe to the TRANSCRIPT event:

```
// Subscribe
cmp.find('voiceToolkitApi')
   .addConversationEventListener('TRANSCRIPT', conversationEventListenerFunc);

// Unsubscribe
cmp.find('voiceToolkitApi')
   .removeConversationEventListener('TRANSCRIPT', conversationEventListenerFunc);
```

You can add multiple conversation event listeners. For example, you can add the following event listeners to subscribe to all the events related to raising and lowering flags:

```
// Subscribe
cmp.find('voiceToolkitApi')
     .addConversationEventListener('FLAG_RAISE', conversationEventListenerFunc);
     .addConversationEventListener('FLAG_LOWER', conversationEventListenerFunc);

// Unsubscribe cmp.find('voiceToolkitApi')
     .removeConversationEventListener('FLAG_RAISE', conversationEventListenerFunc);
     .removeConversationEventListener('FLAG_LOWER', conversationEventListenerFunc);
```

When an event occurs, you receive a JSON payload that contains the event type, along with any relevant data. For instance:

```
{
    "type": "TRANSCRIPT",
    "detail": {
        "id": "3115b389-ab50-400e-b8ba-978b7ec51d7a"
        "clientSentTimestamp": 1594944652299,
        "serverReceivedTimestamp": 1594944652328,
        "content":{
            "formatType": "Text",
            "text": "Hello"
        },
        "callId": "c5d93c19-e03b-44f8-85e6-a11f02e70c45",
        "sender": {
            "role": "Agent"
        }
    }
}
```

The following table describes the payload properties for the FLAG_RAISE, FLAG_LOWER, NOTE, and TRANSCRIPT events:

Property Name	Type	Description
`type`	string	The type of conversation event.
`detail`	object	Identifying information associated with this voice call event.
`detail.id`	string	Unique identifier for the message.
`detail.clientSentTimestamp`	number	The date and time (in UTC) when the client sent the content. Measured in milliseconds since the Unix epoch.
`detail.serverReceivedTimestamp`	number	The date and time (in UTC) when the server received the content. Measured in milliseconds since the Unix epoch.
`content`	object	Information about the content.
`content.formatType`	string	The content format type of the raised flag, note, or transcript. Possible value is "Text."
`content.text`	string	The body text of the raised flag, note, or transcript.
`callId`	string	The unique ID of the voice call within the telephony system. For example, if the telephony system is Amazon Connect, this value is the contact ID.

Property Name	Type	Description
`sender`	object	Information about the user whose action initiated the event.
`sender.role`	string	The role of the user whose action initiated the event. Possible values are "Agent" and "Supervisor".
`sender.displayName`	string	The name of the user whose action initiated the event.
`voiceCallId`	string	Unique identifier of the Salesforce voice call in which a rep raised or lowered the flag. For transferred calls, the new call segment has a different voiceCallId from the original call.

The following table describes the payload properties for the INTELLIGENCE_SIGNAL event:

Property Name	Type	Description
`type`	string	The type of conversation event.
`detail`	object	Identifying information associated with this voice call event.
`detail.events`	string	A piece of signal data generated in real time from a voice call between a rep and end user. Events must be sorted in chronological ascending order by startTime. For example, the following snippet displays a piece of signal data. `{ "type": "IntelligenceSignal__Category", "value": "CustomerAngry", "startTime": 1573503450 }`
`event.service`	number	The intelligence source of the events. Possible values are: • AmazonConnectContactLens
`event.type`	number	The signal type of the event. Possible values are: • IntelligenceSignal__Category
`event.value`	object	The value of the signal generated in real time by the telephony vendor. For example, this could be a Contact Lens rule defined in Amazon Connect.
`event.startTime`	string	The date and time (in UTC) when this event started. Measured in milliseconds since the Unix epoch.
`event.endTime`	string	The date and time (in UTC) when this event ended. Measured in milliseconds since the Unix epoch.
`event.score`	string	The confidence score for the intelligence signal within the range of 1 and 1000, with 1000 being the highest level of confidence.

Service Cloud Voice Aura Toolkit Next Best Action

Give your reps recommendations in real time with Next Best Action.

To incorporate Next Best Action into your Service Cloud Voice solution, follow these instructions.

Note: Next Best Action has usage-based entitlements. Invoking recommendations in Service Cloud Voice counts towards that usage.

1. Build a recommendation strategy that uses the filter element and $Request variable. The $Request variable is a map containing key value pairs. For example, if the recommendation strategy receives the data {intent: "refund"} then $Request.intent evaluates to "refund". To learn more, see Write an Expression.
2. Enable real time recommendations in your Actions & Recommendations deployment and select the strategy you just built.
3. Invoke Einstein Next Best Action with a record ID and key:value pairs in the updateNextBestActions() function parameters. Recommendations are displayed for record pages matching the provided ID. The value given to a particular key can be retrieved in the Recommendation Strategy using $Request.key in the filter element.

The following filter element expression shows recommendations that contain a particular keyword in the recommendation name. To learn more, see Write an Expression.

```
$Request.key != '' && CONTAINS(Name, $Request.key)
```

The following code invokes the next best action function.

```
cmp.find('voiceToolkitApi').updateNextBestActions(recordId, params);
```

Sample function call:

```
cmp.find('voiceToolkitApi')
   .updateNextBestActions('0LQxx0000004GcSGAU', { intent: 'refund' });
```

SEE ALSO:

Salesforce Help: Einstein Next Best Action

Salesforce Help: Einstein Next Best Action Entitlements

Service Cloud Voice Lightning Data Service

Harness the power of Lightning Data Service to interact with VoiceCall records without writing a single line of back-end code.

READ and UPDATE operations are supported for VoiceCall records. UPDATE is restricted to custom fields and the following standard fields: Call Resolution, RelatedRecord, Description.

SEE ALSO:

Salesforce Help: Lightning Data Service

Service Cloud Voice Toolkit API Sample Components

These Lightning component examples illustrate how to use the Service Cloud Voice Toolkit API.

See the examples in our GitHub repository.

SEE ALSO:

Toolkit API Sample Component in GitHub

CHAPTER 6 Upload or Update Transcripts with Connect REST API

Use Connect REST API to upload and update transcripts for voice calls. This functionality can also be used for redacting content in transcripts.

These Connect API resources provide programmatic access to voice transcripts (also called "conversation entries") so that customers can build custom solutions with voice transcripts.

/connect/conversations/upload (GET)
Get the status of conversation uploads.

/connect/conversations/upload (POST)
Upload bulk conversations.

/connect/conversation/*conversationIdentifier*/entries (GET)
Get conversation entries.

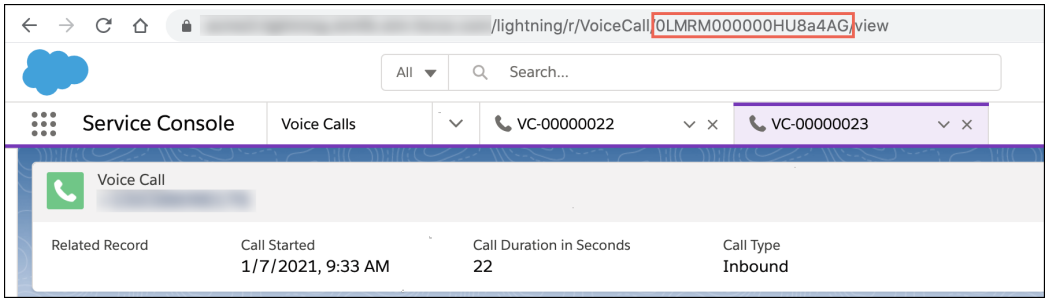
/connect/conversation/*conversationIdentifier*/entries (PATCH)
Update conversation entries.

Usage Example

This example describes how to perform bulk upload of transcripts to a voice call after the conversation. The bulk upload API call preserves the order of transcripts (conversation entries) as they appear in the uploaded payload file, but it doesn't sort or reorder entries based on timestamps.

1. Retrieve the VoiceCall ID for a conversation.

   In order to perform the subsequent steps, you must get the VoiceCall record ID.



2. Using the VoiceCall ID, retrieve the Conversation record and the ConversationParticipant records.

   Using SOQL, you can query for the Conversation ID. For example:

```
SELECT ConversationId FROM VoiceCall WHERE Id = '0LMRM000000HU8a4AG'
```

With the Conversation ID, you can query for additional fields on the Conversation record. For example:

```
SELECT
ConversationChannelId,ConversationIdentifier,CreatedById,CreatedDate,EndTime,Id,IsDeleted,LastModifiedById,LastModifiedDate,Name,StartTime

        FROM Conversation WHERE Id = '0dwRMR000003M5pYAE'
```

You can also get information about the ConversationParticipant records. For example:

```
SELECT
AppType,ConversationId,CreatedById,CreatedDate,Id,IsDeleted,LastModifiedById,LastModifiedDate,ParticipantChannelId,ParticipantDisplayName,ParticipantEmail,ParticipantKey,ParticipantRole

        FROM ConversationParticipant WHERE ConversationId =
'0dwRMR000003M5pYAE'
```

Values returned from this SOQL call are used in the subsequent upload API POST payload. Specifically:

- `AppType` in the SOQL call is used for the “appType” value in the payload. For API version 52.0 and later, refer to the **AppType to appType Mapping** table.
- `ParticipantRole` is used for the “role” value. For API version 52.0 and later, refer to the **ParticipantRole to role Mapping** table.
- `ParticipantKey` is used for the “subject” value.

Table 1: AppType to appType Mapping

AppType Value in SOQL	appType Value in Payload
bmapi	BMAPI
live_message	LIVE_MESSAGE
chatbot	CHATBOT
messaging	MESSAGING
perftool	PERFTOOL
agent	AGENT
telephony_integration	TELEPHONY_INTEGRATION
translation	TRANSLATION
iamessage	IAMESSAGE
conversation	CONVERSATION

Table 2: ParticipantRole to role Mapping

ParticipantRole Value in SOQL	role Value in Payload
System	SYSTEM
Agent	AGENT
Chatbot	CHATBOT
EndUser	END_USER

ParticipantRole Value in SOQL	role Value in Payload
Supervisor	SUPERVISOR
Router	ROUTER

3. Call the bulk upload API using the desired payload.

   Make a multi-part POST call to the Connect REST API for uploading conversation data. For example:

```
POST
https://MY_ORG_DOMAIN/services/data/v52.0/connect/conversations/upload
```

   To prepare the request, save the payload to a binary file. Use a payload that contains the desired conversation participant info, along with the conversation entries. For example, the following payload sample applies to API version 52.0 and later.

```
{"type": "conversation", "payload": {"conversationId":
"ab123456-76d6-4d60-a53d-fb69d800b10c"}}
{"type": "conversationEntry", "payload": {"conversationId":
"ab123456-76d6-4d60-a53d-fb69d800b10c", "id": "1",
"clientSentTimestamp": 1610580214000, "clientDuration": 123432,
"messageText":"Hello, how can I help you?","sender": {"appType":
"AGENT", "subject": "005RM00000225FB","role":
"AGENT"},"relatedRecords": ["0LQXXXXXXXXXXX"] }}
{"type": "conversationEntry", "payload": {"conversationId":
"ab123456-76d6-4d60-a53d-fb69d800b10c", "id": "2",
"clientSentTimestamp": 1610580214001, "clientDuration": 123432,
"messageText":"I want to return this merchandise. Can you please
help?","sender": {"appType": "TELEPHONY_INTEGRATION", "subject":
"END_USER","role": "END_USER"},"relatedRecords":
["0LQXXXXXXXXXXX"]}}
```

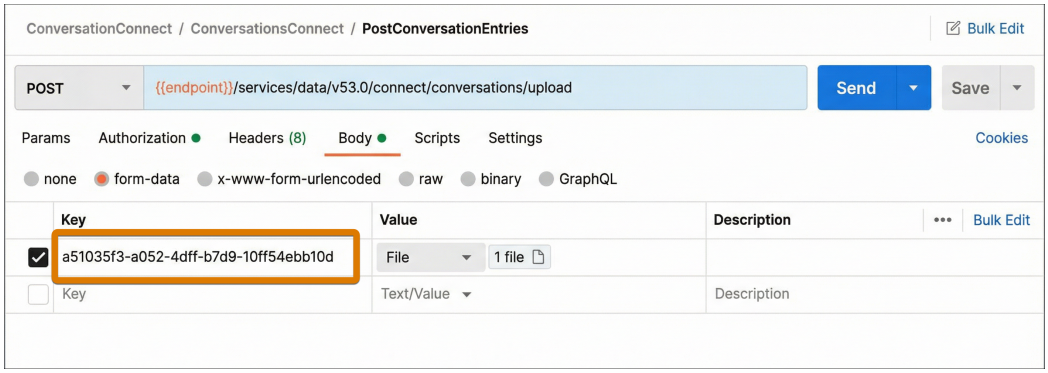
   Use the following guidance when filling out this payload:

   - Within each part of the payload, the first object contains the conversation details (1). The latter set of objects contains the conversation entry details (2).

   - The `clientSentTimestamp` must be between `startTimestamp` and `endTimestamp`. Timestamps are specified using the UNIX Epoch time.
   - `contextParamMap` is used to add contextual information. This parameter is optional.
   - `appType`, `role`, and `subject` come from the `ConversationParticipant` SOQL call described in the previous step.
   - `relatedRecords` is the list of object records that are associated with the conversation entry. Valid values for voiceCallIds in the relatedRecords list are 15 characters long.
   - In the multi-part POST request, the key name (that is, the binary file) must be the `conversationId`. For details, see Uploading Binary Files in the Connect REST API Developer Guide.

To test this operation, save the payload (request body) in a file named `transcript.json`. Use a tool like Postman to execute the request. In the multi-part POST request, make sure the name of the part containing the binary file is the `conversationId`. For Value, select **File** and upload the JSON file with your sample payload.



Note: Currently, this API only supports existing closed conversations. A previous version of this API required a conversation channel, the start timestamp, and the end timestamp in the conversation object, and it required conversation participant objects. These values are now optional.

You get the following response when the call is successful.

```
{
    "conversationBulkUploadsResults": [
        {
            "conversationIdentifier":
"fc803778-76d6-4d60-a53d-fb69d800b10c",
            "errorDetail": null,
            "status": "SUCCESS",
            "uploadId": "333e7382-ef11-3396-b7c2-156fb6b5f7ad"
        }
    ]
}
```

4. Check the status of the upload.

   To check the status, perform a GET request to the same resource using the upload IDs.

```
GET
https://MY_ORG_DOMAIN/services/data/v51.0/connect/conversations/upload?uploadIds=uploadId1,uploadId2,uploadId3
```

   If the upload was successful, you see this type of response.

```
{
    "conversationBulkUploadsResults": [
        {
            "conversationIdentifier":
"fc803778-76d6-4d60-a53d-fb69d800b10c",
            "errorDetail": null,
            "status": "SUCCESS",
            "uploadId": "333e7382-ef11-3396-b7c2-156fb6b5f7ad"
        }
```

```
    ]
}
```

If the upload failed, you see this type of response.

```
{
    "conversationBulkUploadStatuses": [
        {
            "conversationId":
"fc803778-76d6-4d60-a53d-fb69d800b10c",
            "errorDetail": "Failed to process any conversation
entries.",
            "failedEntryCount": 2,
            "failedEntryIds": [
                "1",
                "2"
            ],
            "lastUpdatedTimestamp": 1611621077219,
            "status": "FAILED",
            "successEntryCount": 0,
            "uploadId": "333e7382-ef11-3396-b7c2-156fb6b5f7ad"
        }
    ]
}
```

You can get the new conversation entries using an HTTP GET to /connect/conversation/*conversationIdentifier*/entries.

CHAPTER 7 Using the Service Cloud Voice Lambda Functions for Amazon Connect

In this chapter ...

Salesforce provides a set of Lambda functions, which are available within your Amazon Connect instance after provisioning the instance with Service Cloud Voice. You can use these Lambdas in Amazon Connect contact flows.

These Lambdas interact with Salesforce to perform the following actions:

- Create a voice call (InvokeTelephonyIntegrationApiFunction on page 145).
- Start real-time transcription and save transcripts to Salesforce (kvsConsumerTrigger on page 163, kvsTranscriber on page 163).
- Sync data between a contact record (previously called contact trace record or CTR) and a VoiceCall object (CTRDataSyncFunction on page 156).
- Invoke the Salesforce REST API to perform CRUD operations on Salesforce objects, such as cases, accounts, or custom objects (InvokeSalesforceRestApiFunction on page 142).

For more detail, refer to the reference pages for the Lambda functions.

SEE ALSO:

Sample Amazon Connect Flows for Service Cloud Voice

How to Use Salesforce Lambda Functions

- VoiceMailPackagingFunction Lambda Function
- VoiceMailTranscribeFunction Lambda Function
- kvsConsumerTrigger Lambda Function
- kvsTranscriber Lambda Function
- Query AWS Lambda Functions to Analyze a Voice Call

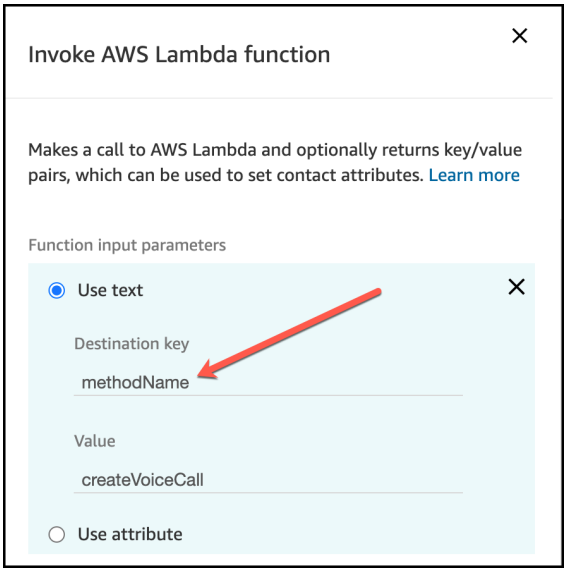
How to Use Salesforce Lambda Functions

Salesforce provides several Lambda functions for use in your Service Cloud Voice contact flows. These functions are used by the sample contact flows, and you can also use them in your own contact flows.

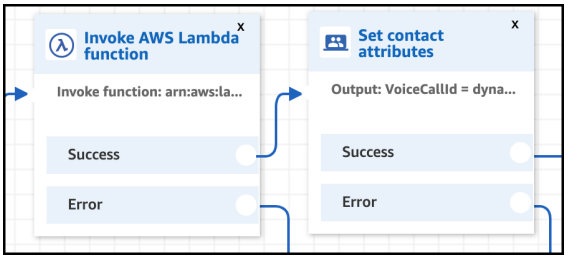
To learn more about Lambda functions, see Amazon's AWS Lambda Developer Guide. When fine-tuning your contact flows, you often access contact attributes using Amazon's JSONPath syntax. Be sure to review Amazon documentation on how to reference these attributes.

Salesforce occasionally updates the Lambda functions. To learn how to update these functions, see Update Your Contact Center in Salesforce Help.

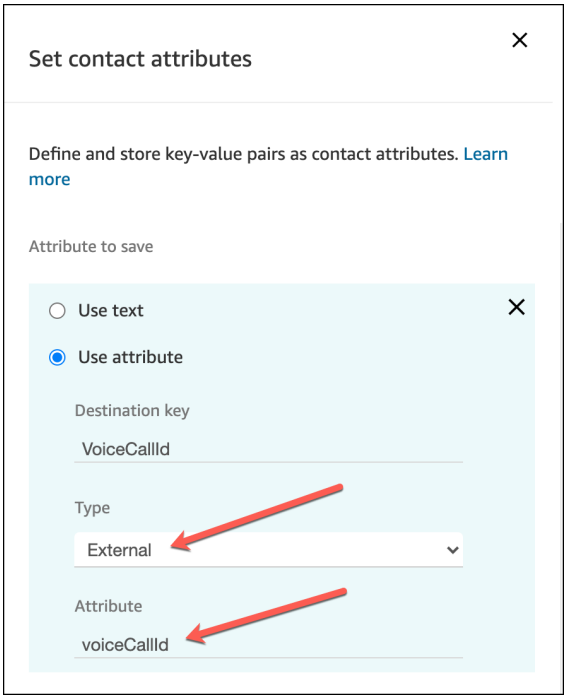
For functions with multiple actions (`InvokeTelephonyIntegrationAPI` and `InvokeSalesforceRestApiFunction`), you must specify the action name in an input parameter named **methodName**. Create an attribute for the **methodName** key, as documented with each function. Additional parameters may be necessary for each method.



After calling a Lambda function, be sure to store the output value using a **Set contact attributes** block. If you don't store the output, subsequent calls can overwrite this value.



In the **Set contact attributes** block, the **Attribute** must match the output variable and the type must be **External**.



The **Destination key** can be whatever value you want to use as a variable later in your contact flow.

SEE ALSO:

- AWS Lambda Developer Guide
- *Salesforce Help:* Update Your Contact Center

Service Cloud Voice Authentication When Using InvokeSalesforceRestApiFunction

OAuth is required if you want to use the `InvokeSalesforceRestApiFunction` Lambda function. If you need to set up or modify the OAuth, refer to these instructions.

OAuth is required if you want to perform Salesforce REST API calls to create, update, or query records in any of your contact flows.

Set Up OAuth in Your Service Cloud Voice Connected App
: These instructions describe how to set up OAuth in a Salesforce connected app.

Set Up OAuth in the AWS Lambda Function
: These instructions describe how to set up OAuth on your Amazon Connect instance.

Test OAuth with Service Cloud Voice Lambda Function
: To test that you have OAuth set up for the `InvokeSalesforceRestApiFunction` Lambda function, create a test event.

SEE ALSO:

- *Salesforce Help:* Create a Private Key and Self-Signed Digital Certificate
- InvokeSalesforceRestApiFunction Lambda Function

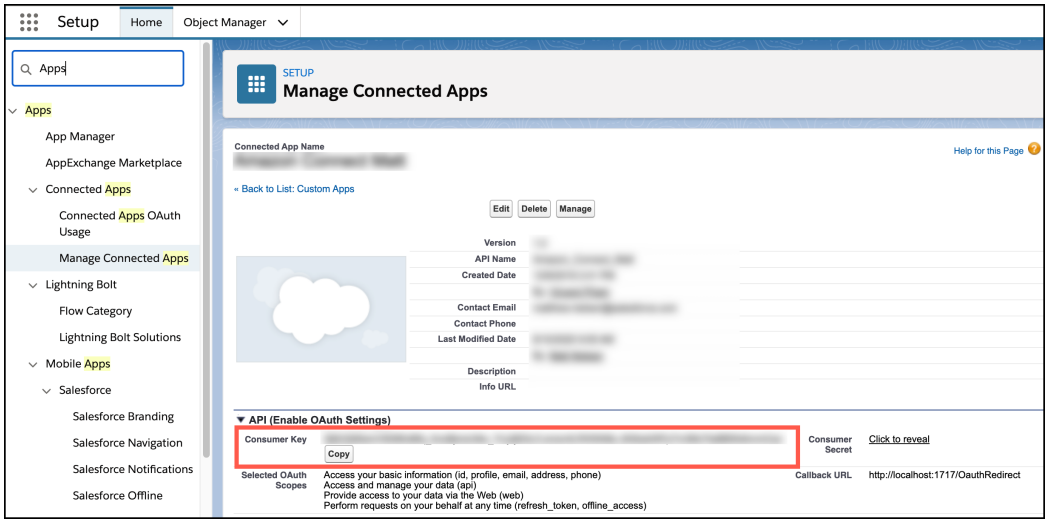
Set Up OAuth in Your Service Cloud Voice Connected App

These instructions describe how to set up OAuth in a Salesforce connected app.

Note: Starting with the Spring '26 release, the creation of connected apps is disabled to promote the adoption of External Client Apps (ECA). To create a connected app, you must contact Customer Support. For more information, see this knowledge Article.

Before you start, create a private key and a self-signed certificate.

1. From **Setup**, enter `Apps` in the Quick Find box, and select **App Manager**. If you don't already have a connected app for Amazon Connect, create a connected app.
2. Click **Edit** in the dropdown menu.
3. Click Enable OAuth Settings.
4. For the **Callback URL**, if it's not already specified, enter: `http://localhost:1717/OauthRedirect`
5. Select **Use digital signatures** and upload your self-signed certificate file.
6. For the Selected OAuth Scopes, ensure that these scopes are selected.
   - Manage user data via APIs (api)
   - Perform requests on your behalf at any time (refresh_token, offline_acess)
7. Save your settings.
8. In the action menu for the connected app, click **Manage**, and then click **Manage Profiles**. To choose which users have access to this connected app, select the appropriate profiles. We recommend including a System Administrator profile. Verify that the Permitted Users value is set to **Admin approved users are pre-authorized** for the selected profiles.
9. Save your work.
10. From the saved connected app with OAuth, copy the **Consumer Key**, which is needed on the Amazon Connect instance.



SEE ALSO:

Salesforce Help: Create a Private Key and Self-Signed Digital Certificate

Salesforce Help: Create a Connected App

Salesforce Help: Enable OAuth Settings for API Integration

Salesforce Help: Manage OAuth Access Policies for a Connected App

Set Up OAuth in the AWS Lambda Function

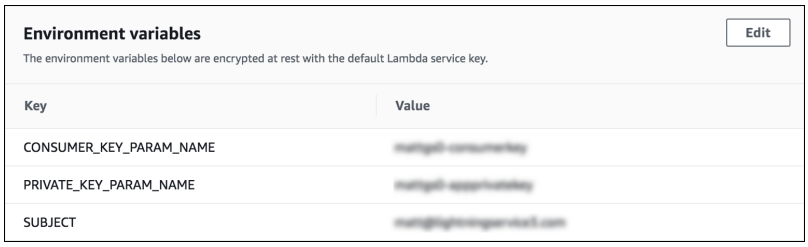
These instructions describe how to set up OAuth on your Amazon Connect instance.

Before you start, open a text file so you can copy some parameter values or keys.

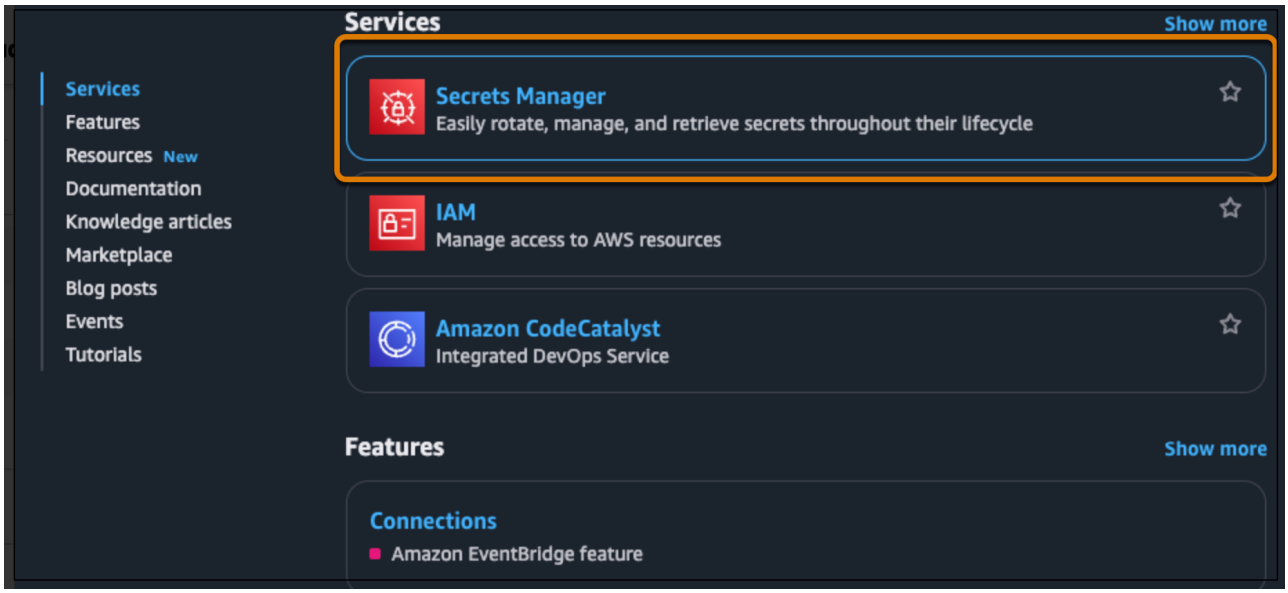
Starting with contact center version 19.0, configuration parameters are stored in the AWS Secrets Manager instead of the SSM Parameter Store. The contact center agnostic configuration values are stored in environment variables. The contact center specific configuration values are stored in the respective Secrets Manager for each contact center.

All the Lambda functions read the configuration parameters from the Secrets Manager. For the contact center versions before 19.0, the Lambda functions continue to read the configuration parameters from the SSM Parameter Store

1. Log in to the AWS Console > Lambda > Functions. Log in with your root user email which you used to create a contact center during Service Cloud Voice setup. If this is your first login, reset your password.
2. Select the `{Your Contact Center Name}-InvokeSalesforceRestApiFunction` Lambda function.
3. For contact center versions below 19.0, click the Configuration tab and scroll down to **Environment variables**. Some of the variables are already configured for your instance, but you have to set the values for `CONSUMER_KEY_PARAM_NAME`, `PRIVATE_KEY_PARAM_NAME`, and `SUBJECT` parameters.



For contact center versions 19.0 or later, go to Secrets Manager from Services. In the Secrets page, locate your contact center's secret. The secret name is prefixed with `callCenterApiName` for partner Amazon contact centers or LambdaPrefix for multi-org contact centers. For provisioned contact centers, the secret name is also available in the Lambda functions environment variable. Then, set the `CONSUMER_KEY_PARAM_NAME`, `PRIVATE_KEY_PARAM_NAME`, and `SUBJECT` keys within that secret.



Note: If you're testing this functionality in a sandbox org, make sure that the secrets for contact center versions 19.0 or later, and environment variables for contact center versions below 19.0, point to the sandbox domain (or to `test.salesforce.com`) instead of `login.salesforce.com`: `AUDIENCE`, `SALESFORCE_AUTH_ENDPOINT`, `SALESFORCE_REST_API_ENDPOINT_BASE`.

For contact center versions below 19.0, the name of the secret is specified in the Lambda environment variables. Use the secret name to locate the corresponding AWS secret and update the values within the secret to use your sandbox domain.

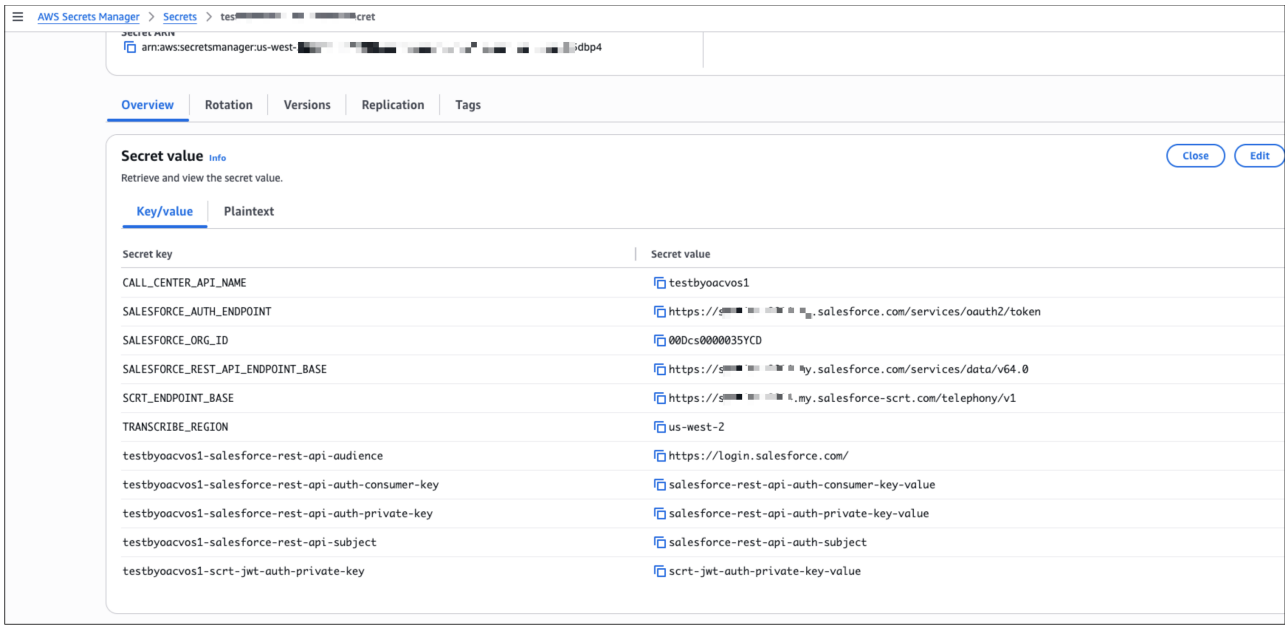
a. Copy the **values** for the `CONSUMER_KEY_PARAM_NAME`, `PRIVATE_KEY_PARAM_NAME` keys or parameters and paste them into a text file. For contact center versions 19.0 or later, these values are the names of the keys within a single secret that hold the actual credentials. For contact center versions below 19.0, these values are the names of the SSM parameters that hold the actual credentials.

b. For the `SUBJECT` variable, enter the Salesforce username of the user you want to log in to Salesforce through the Lambda function. The user must have access to the objects that the Lambda function will be executed on.

c. In the connected app in Salesforce, check that the user is added to the connected app profile. From **Setup** in your Salesforce org, go to **App Manager**. Open the action menu for the connected app, click **Manage**, and scroll to the bottom of the page to see the profiles. Verify that the **Permitted Users** value is set to "Admin approved users are pre-authorized" for the desired profile. To learn more about OAuth access policies, see Manage OAuth Access Policies for a Connected App.

4. For contact center versions below 19.0, in the parameter store, search for the `CONSUMER_KEY_PARAM_NAME` value and the `PRIVATE_KEY_PARAM_NAME` value that you copied to the text file.

For contact center versions 19.0 or later, there are no `CONSUMER_KEY_PARAM_NAME` and `PRIVATE_KEY_PARAM_VALUE` secrets, instead these are keys. For example `testbyoacvos1-salesforce-rest-api-auth-consumer-key`.



5. For the Consumer Key, update the value by clicking **Edit** and pasting the value of the consumer key from the connected app into the **Value** field.

Note: When editing key and value of secrets, use Plaintext mode so that it does not reformat the certificate.

To update keys in a secret, format the key and then update the key in Secrets Manager.

a. To format the multi-line key into a single line, copy the private key in a notepad and append line breaks `\r\n` or run the command in terminal.

```
cat <<EOF | awk '{printf "%s\\r\\n", $0}' PASTE YOUR Certificate > EOF
```

If you use Windows, you can also run this command to format the key: `(Get-Content privateKey.pem | Where-Object {$_ -ne ""}) -join "\n" | Set-Content private_five.key`

Copy the formatted single-line key.

Multi-line key before formatting:

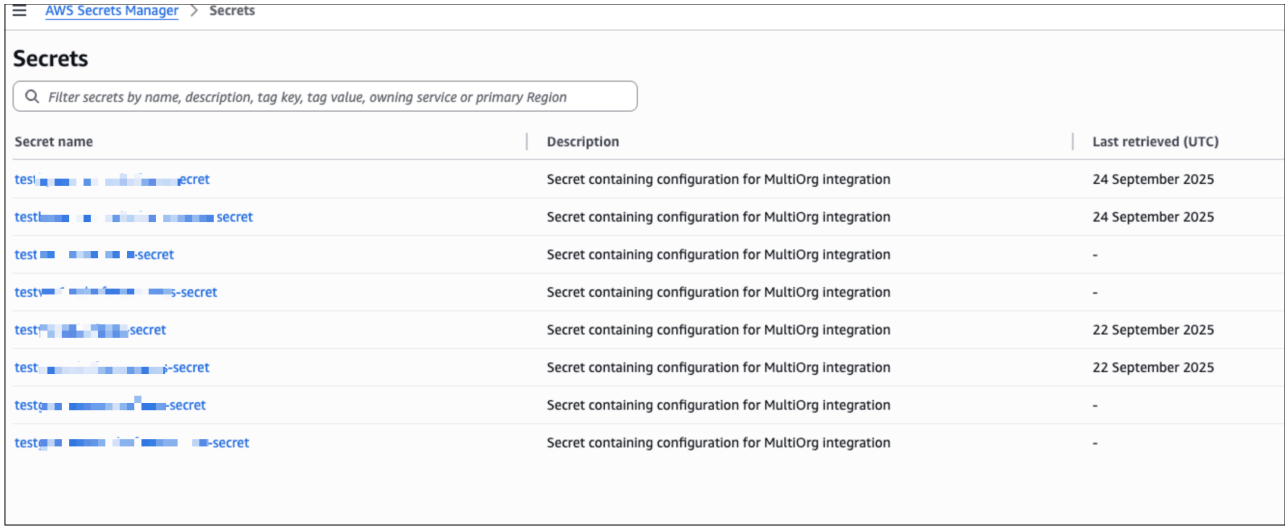
Formatted single-line key:

b. In Secrets Manager, select the secret for the contact center.

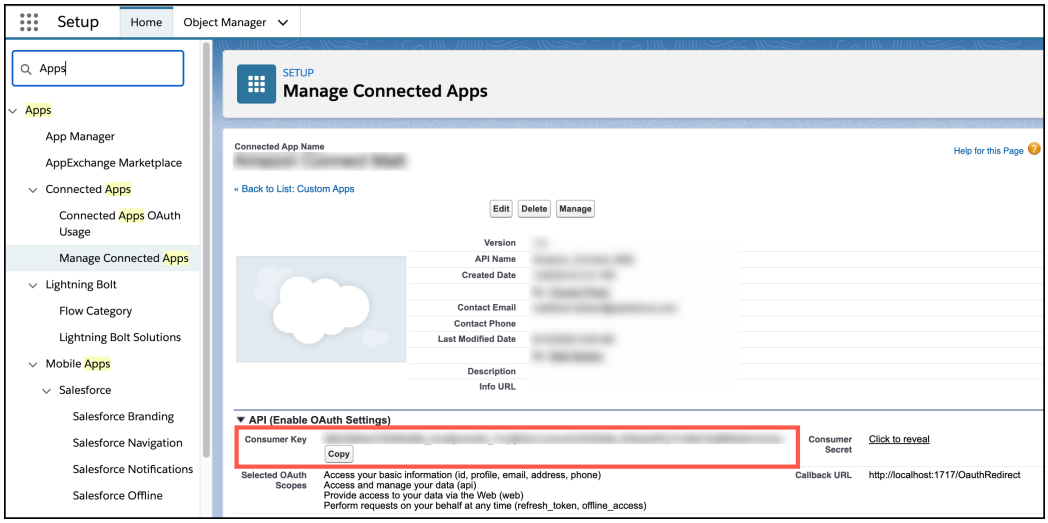
c. Select **Retrieve secret value** and click **Edit**, select the Plaintext view, and paste the formatted single-line key for the appropriate key.

d. Save your changes.

You can also use AWS sdk to update the key in the secret.



Note: If you didn't already copy the Consumer Key from your app, you can get that value from your Salesforce org. From **Setup**, enter `Apps` in the Quick Find box, and select **App Manager**. Click **View** in the dropdown menu for your connected app. **Copy** the value for the Consumer Key from the connected app page.



6. Save your changes.
7. Update the value of the private key by clicking Edit and pasting the private key into the **Value** field. For the private key, you need the key that you previously used to sign the self-signed certificate. By default, this key is stored locally to a file named `server.key`.

Note: The private key isn't the same as the consumer secret, although both are generated by the same key. The private key begins with "`-----BEGIN RSA PRIVATE KEY-----`" and ends with "`-----END RSA PRIVATE KEY-----`". Be sure to include that BEGIN and END text as part of the key value.

8. Save your changes.

Note: It can take a few minutes before the changes are applied to the connected app.

SEE ALSO:

Salesforce Help: Create a Private Key and Self-Signed Digital Certificate

Test OAuth with Service Cloud Voice Lambda Function

To test that you have OAuth set up for the `InvokeSalesforceRestApiFunction` Lambda function, create a test event.

1. Log in to AWS Console > Lambda > Functions.
2. Select the `{Your Contact Center Name}-InvokeSalesforceRestApiFunction` Lambda function.
3. Click the **Test** dropdown menu and select **Configure test event**.
4. Copy the JSON information to use as your test event. Put the entire query on one line. Contact Salesforce Support if you don't already have some test data to use.

   For example:

```
{
  "Details": {
    "Parameters": {
      "methodName": "queryRecord",
      "soql": "SELECT Id FROM Case WHERE ContactPhone = '+14155551234' AND IsClosed =
false"
    }
  }
}
```

5. Select the test event and click **Test**.
6. Verify that the test succeeded.

InvokeSalesforceRestApiFunction Lambda Function

This Lambda function calls the Salesforce REST API. You can place this Lambda function within a contact flow to create, update, or query Salesforce records.

Configuration: This function is available to Service Cloud Voice customers who are using an Amazon Connect instance that's provided by Salesforce. This function uses OAuth, which requires an associated connected app in your org. This app is automatically set up for you when you enable Voice.

Important: Before using this function, you must have OAuth authentication set up. To set up or modify authentication, see Service Cloud Voice Authentication When Using InvokeSalesforceRestApiFunction on page 136.

This Lambda function uses the environment variable `SALESFORCE_REST_API_ENDPOINT_BASE` to point to the Salesforce org API endpoint. (For example, `https://MY_DOMAIN.my.salesforce.com/services/data/v50.0`.) This variable is

automatically created when you set up Service Cloud Voice. However, if you get a response of `{ success: false }`, verify that this Lambda environment variable is correct.

Sample Contact Flows That Use This Function: Sample_SCV_REST_Check_For_Open_Cases, Sample_SCV_REST_Link_Call_To_Case. To download these flows, visit our Sample Contact Flows folder in GitHub.

Limitations: This Lambda function is subject to Salesforce API request limits and allocations.

Usage: Use this function to create a record, update an existing record, or query for a record. To learn more about programmatically managing Salesforce records, refer to the Object Reference for Salesforce and Lightning Platform and the Salesforce REST API Developer Guide.

This Lambda takes a "methodName" attribute as an argument. This attribute indicates the specific action that the Lambda must perform.

Table 3: Input Attributes

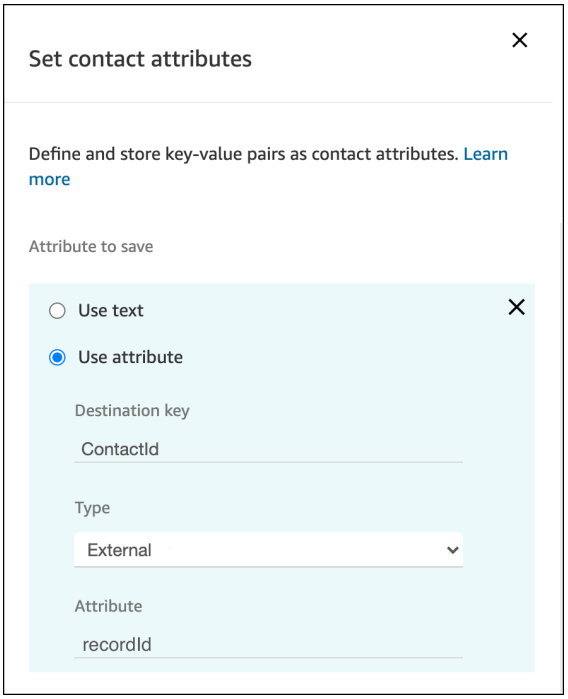
Action	methodName Value	Other Required Attributes
Create a Salesforce record	`createRecord`	• `objectApiName` containing the name of the Salesforce object. • A list of fields as required to create the record. Each key name is the field name.
Update an existing Salesforce record	`updateRecord`	• `objectApiName` containing the name of the Salesforce object. • `recordId` containing the ID of the record. • The list of fields that you want to update. Each key name is the field name.
Query for a Salesforce record	`queryRecord`	`soql` containing the desired SOQL query. For example: `SELECT Id FROM Contact WHERE Phone = '$.Attributes.MY_CONTACT_ATTRIBUTE'` You must store the correct information into a contact attribute first. For details about the $ variable, see the Amazon documentation on contact attributes.

Table 4: Output Attributes

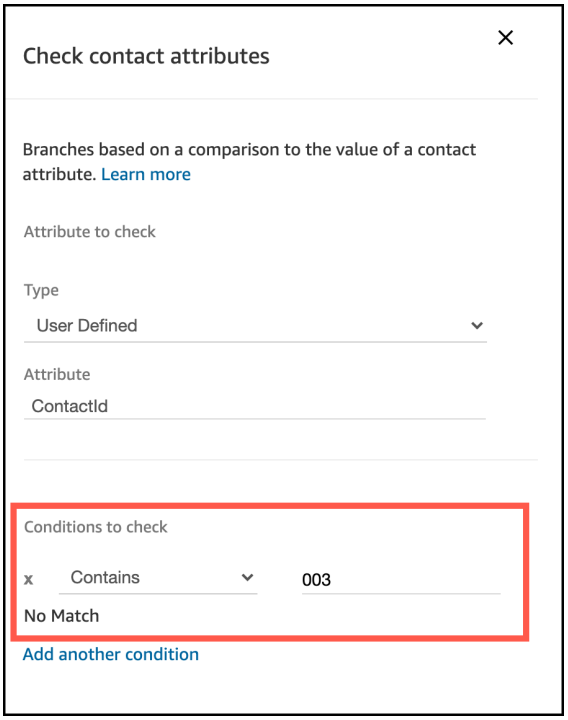
methodName	Output Attribute Name	Description
`createRecord`	`recordId`	ID of the newly created record.
`updateRecord`	None	N/A
`queryRecord`	Names match the names of the fields you queried.	A list of the selected fields. Each attribute contains a different field. Regardless of your

methodName	Output Attribute Name	Description
		query, only the first resulting record is returned.

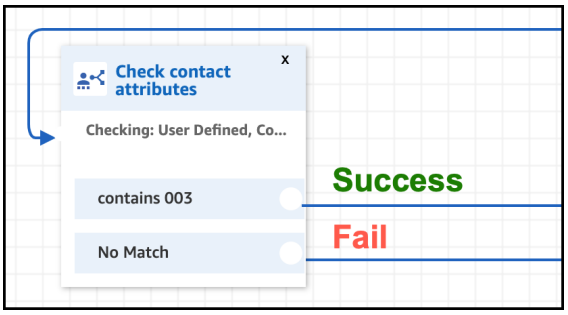
TIP: To check that the function returned information successfully, add a condition to a **Check contact attributes** block to check an output attribute based on some detail you expect to see in the result. For instance, let's say you're creating a contact record. A contact ID always contains "003" in the ID value, so you can use that information to check the success or failure of the call. The first step is to store the return attribute in a user-defined attribute by using a **Set contact attributes** block:



And then you can use a **Check contact attributes** block to access this user-defined attribute:



Build your contact flow logic based on this result.



SEE ALSO:

Service Cloud Voice Authentication When Using InvokeSalesforceRestApiFunction

InvokeTelephonyIntegrationApiFunction Lambda Function

The invokeTelephonyIntegrationApi function contains actions which orchestrate voice call handling and routing. The function integrates telephony events with Salesforce, ensuring calls are properly processed and directed through the appropriate Omni-Channel logic.

Configuration: This function is available to Service Cloud Voice customers who are using an Amazon Connect instance that's provided by Salesforce. No additional configuration is necessary. See a sample version of the invokeTelephonyIntegrationApi Lambda function in GitHub.

Description: Use this Lambda function to create a VoiceCall record, execute an Omni-Channel flow, create a ContactRequest record for a callback, or route a voice call to a rep, queue, or flow. Creating a VoiceCall is essential to get the call working, which is why this function is typically one of the first blocks in your contact flow for an inbound call. This Lambda is invoked to perform the following actions:

- Create a VoiceCall for an inbound call. This action occurs when a customer calls into the call center.

- Create a VoiceCall for a call transfer. This action occurs when a rep transfers to another rep or queue.
- Create a ContactRequest for a callback. This action occurs when a customer requests a callback and provides their phone number for a rep to return their call.
- Execute an Omni-Channel flow. This action occurs when work is ready to be routed.
- Cancel an Omni-Channel flow. This action occurs when an existing flow needs to be canceled.
- Route a voice call. This action occurs when a voice call or transferred voice call is ready to be routed through Omni-Channel Unified Routing with Salesforce as the routing engine.

Important: When a Lambda function hasn't been invoked after a period of time, the AWS framework unloads the function from memory. The next time the function is invoked, it is "cold" and takes longer to load into memory. This delay can cause a timeout in the contact flow. To avoid this behavior, learn how to keep this function warm on page 148.

Usage: Place this Lambda within one of the relevant contact flows (either for an inbound call or a transfer). This Lambda takes a `methodName` attribute as an argument. This attribute indicates the specific action that the Lambda must perform.

Table 5: Input Attributes

Action	Attribute Key	Attribute Value
Create a voice call for an inbound call	`methodName`	`createVoiceCall`
Create a voice call for a transfer	`methodName`	`createTransferVC`
Request routing instructions from an Omni-Channel flow and create a PendingServiceRequest. Note: Calls routed with this method appear in the Queue Backlog tab of Command Center for Service. If a PendingServiceRequest isn't created and the voice call isn't routed by Omni-Channel, then the Queue Backlog tab doesn't show the call. For example, calls that are manually assigned to a rep don't appear in that tab. Optionally, provide these inputs: • `flowDevName`. Set the Omni-Channel flow using its developer name. • `fallbackQueue`. Set the fallback queue using its Salesforce queue developer name or ID. • Custom parameters with prefix *`flowInput-`*. Set additional inputs to the Omni-Channel flow. By default, Service Cloud Voice uses the Omni-Channel flow (or fallback queue) specified for the phone channel that matches the dialed number. If the dialed number doesn't match an existing phone channel, you can optionally set the Omni-Channel flow or fallback queue in these input parameters. The fallback queue is used only if the specified flow doesn't exist or one isn't specified.	`methodName`	`executeOmniFlow`
Cancels an Omni-Channel flow that was previously executed. This method also calls the Clear Routing on page 102 REST API to clear the PendingServiceRouting (PSR) record.	`methodName`	`cancelOmniFlowExecution`
Creates a callback request for a voice call by creating a ContactRequest record. Omni-Channel routes the contact request based on the contact center channel's Callback Routing selection. For Omni-Channel Unified Routing only.	`methodName`	`callbackExecution`

Action	Attribute Key	Attribute Value
Routes or transfers a voice call to a rep, agent, queue, or flow. Creates a PendingServiceRequest. For Omni-Channel Unified Routing only. Provide these inputs: • `routingTarget`. Specify the Omni-Channel routing target using a Salesforce agent ID, queue ID, or Omni-Channel flow ID. Required. • `fallbackQueue`. Set the fallback queue using its ID, if a flow is the `routingTarget`. Optional. • Custom parameters with prefix *`flowInput-`*. Set additional inputs to the Omni-Channel flow, if a flow is the `routingTarget`. Optional.	`methodName`	`routeVoiceCall`

When creating a voice call record, you can also specify custom data to be added to fields on the VoiceCall record. To add custom data, create a **Set contact attributes** block before the Lambda call. For the destination attribute, specify the name of the custom field with the "sfdc-" prefix added. The value of this attribute is the data to pass into the custom field For an example, see Pass Data from the Interactive Voice Response (IVR) System on page 5.

Table 6: Output Attributes for `createVoiceCall` and `createTransferVC`

Output Attribute Name	Description
`voiceCallId`	ID of the voice call.

Table 7: Output Attributes for `executeOmniFlow`

Output Attribute Name	Description
`agent`	The rep to route the call to
`queue`	The queue to route the call to
`errors`	List of errors why the call couldn't be routed

Table 8: Output Attributes for `requestCallback`

Output Attribute Name	Description
`recordId`	ID of the contact request for a callback.

Keep the InvokeTelephonyIntegrationApiFunction Lambda Function Warm

When a Lambda function hasn't been invoked after a period of time, the AWS framework unloads the function from memory. The next time the function is invoked, it is "cold" and takes longer to load into memory. This delay can cause a timeout in the contact flow. These instructions describe how to keep your Lambda function "warm" using a Lambda function trigger.

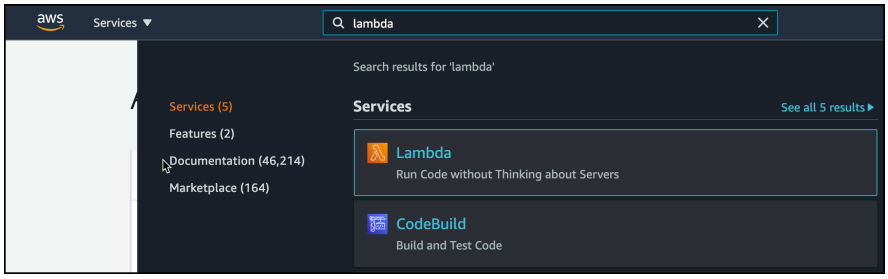
SEE ALSO:

Keep the InvokeTelephonyIntegrationApiFunction Lambda Function Warm

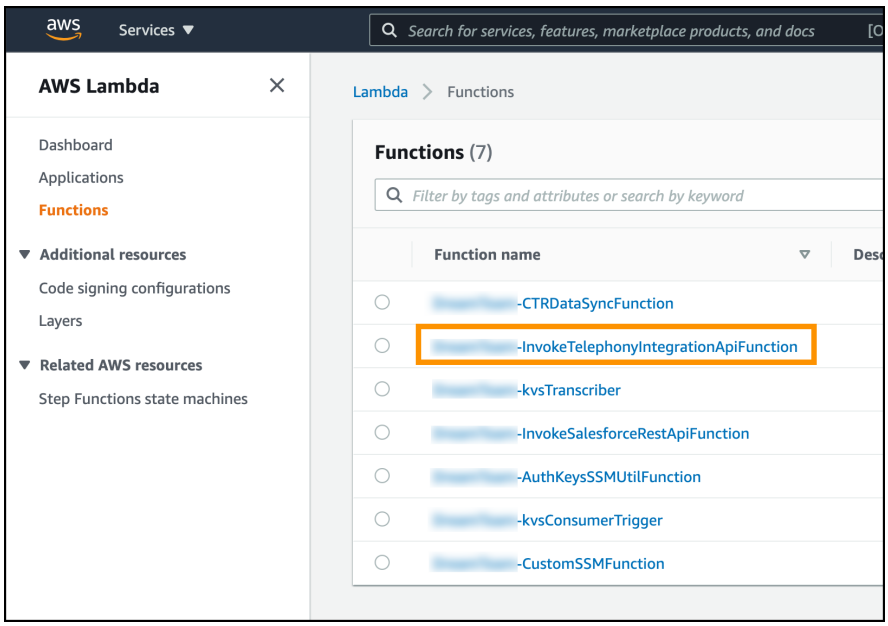
Keep the InvokeTelephonyIntegrationApiFunction Lambda Function Warm

When a Lambda function hasn't been invoked after a period of time, the AWS framework unloads the function from memory. The next time the function is invoked, it is "cold" and takes longer to load into memory. This delay can cause a timeout in the contact flow. These instructions describe how to keep your Lambda function "warm" using a Lambda function trigger.

1. In Amazon Connect, search for the Lambda function page.

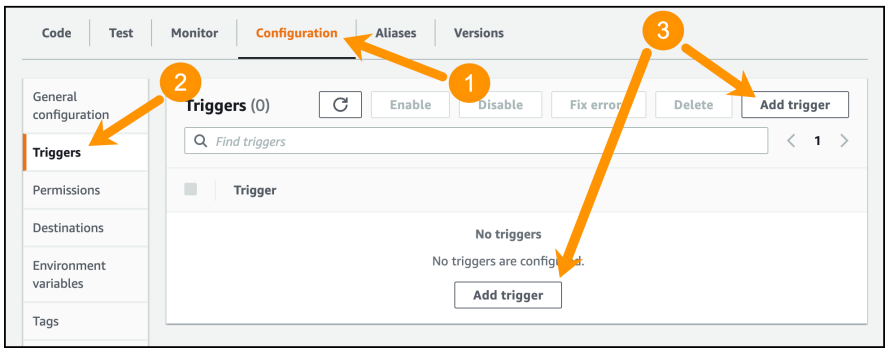


2. From the AWS Lambda function page, select the **InvokeTelephonyIntegrationApiFunction** Lambda function.

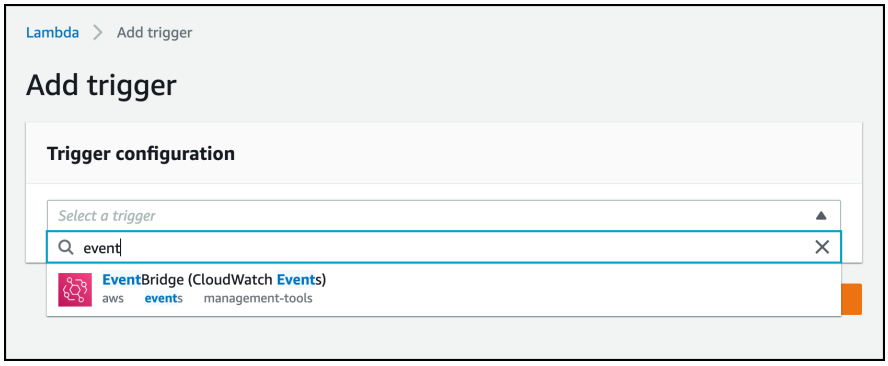


Note: The Lambda function has your contact center name as the prefix.

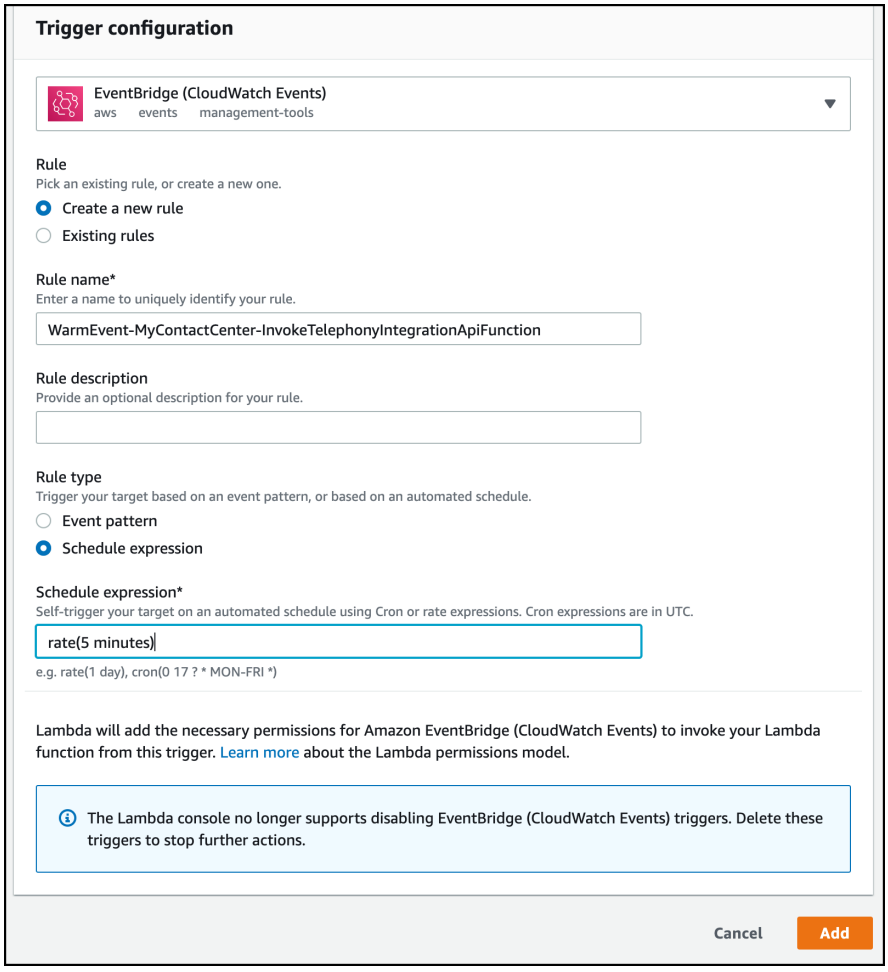
3. Copy the name of the function to your clipboard so that you can create a unique name for your rule later.
4. From the function overview page, select the **Configuration** tab (1). Click the **Triggers** subtab in the sidebar (2). To create a trigger, click the **Add trigger** button (3).



5. From the Trigger configuration page, search for the EventBridge trigger and select it.



6. Set up your trigger so that it runs every five minutes using the following values.



a. For the **Rule**, select **Create a new rule**.

b. For the **Rule name**, specify something globally unique. We suggest the prefix "WarmEvent-" plus the unique name of your InvokeTelephonyIntegrationApiFunction Lambda function that you previously copied to the clipboard.

c. For the **Rule type**, specify **Schedule expression**.

d. For the schedule expression, specify `rate(5 minutes)`. This expression ensures that the Lambda is called every five minutes.

e. To add this new trigger, click the **Add** button.

The **InvokeTelephonyIntegrationApiFunction** Lambda function automatically detects when it's being executed by a scheduled event trigger and only performs enough processing to stay warm.

SEE ALSO:

Amazon EventBridge

ContactLensConsumerFunction Lambda Function

This Lambda function gets real-time transcripts and intelligence signals generated from Contact Lens for Amazon Connect and sends them to the ContactLensProcessorFunction Lambda function.

Configuration: This function is available to Service Cloud Voice customers who are using an Amazon Connect instance that's provided by Salesforce. This function is available in Service Cloud Voice contact center 9.0 and later versions. Starting in contact center version 11.1, we split this Lambda function into two by introducing the ContactLensProcessorFunction Lambda function to improve performance.

Usage: This function performs these actions:

1. Reads the transcripts generated from Contact Lens in real time and sends them to the ContactLensProcessorFunction on page 151 Lambda function.
2. Reads the conversation intelligence signals generated from Contact Lens in real time and sends them to the ContactLensProcessorFunction Lambda function.

This function supports inbound calls, transfer calls, outbound calls, and callback calls.

SEE ALSO:

Create Transcripts in Bulk

ContactLensProcessorFunction Lambda Function

ContactLensProcessorFunction Lambda Function

This Lambda function processes real-time transcripts and intelligence signals generated from Contact Lens for Amazon Connect and sends them to Salesforce. This Lambda function is invoked by the ContactLensConsumerFunction Lambda function with the generated real-time transcripts and events.

Configuration: This function is available to Service Cloud Voice customers who are using an Amazon Connect instance that's provided by Salesforce. This function is available in Service Cloud Voice contact center 11.1 and later versions.

Usage: This function performs these actions:

1. Reads the transcripts sent from the ContactLensConsumerFunction on page 150 Lambda function in real time and sends the transcripts to Salesforce.
2. Processes the transcripts using the Create Transcripts in Bulk on page 84 (`/telephony/v1/voiceCalls/messages`) REST API.
3. Processes the conversation intelligence signals sent from the ContactLensConsumerFunction Lambda function in real time and sends the signals to Salesforce.

This function supports inbound calls, transfer calls, outbound calls, and callback calls.

Increase Performance of the ContactLensProcessor Lambda Function

For contact center 11.1 and above, you can enable Contact Lens to use the AWS SSM Lambda extension to fetch keys from the parameter store. This extension increases the Lambda function's performance. To use this enhancement, configure the ContactLensProcessor Lambda function by adding the AWS-Parameters-and-Secrets-Lambda-Extension layer and the USE_SSM_LAMBDA_EXTENSION environment variable for the Lambda function.

SEE ALSO:

Create Transcripts in Bulk

ContactLensConsumerFunction Lambda Function

Increase Performance of the ContactLensProcessor Lambda Function

Increase Performance of the ContactLensProcessor Lambda Function

For contact center 11.1 and above, you can enable Contact Lens to use the AWS SSM Lambda extension to fetch keys from the parameter store. This extension increases the Lambda function's performance. To use this enhancement, configure the ContactLensProcessor Lambda function by adding the AWS-Parameters-and-Secrets-Lambda-Extension layer and the USE_SSM_LAMBDA_EXTENSION environment variable for the Lambda function.

1. In AWS, log in to the Identity and Access Management console.
2. Select the region where the Lambda functions are deployed.
3. Go to Lambda, and then select **<contact center name>-ContactLensProcessorFunction**.
4. In the Layers section, click **Add a layer**.
5. Select **AWS layers**.
6. From the AWS Layers dropdown, select **AWS-Parameters-and-Secrets-Lambda-Extension**.
7. In the Version field, select the **latest provided version**.
8. Click **Add**.
9. Open the Configuration tab for this Lambda function.
10. Select **Environment variables**.
11. Click **Edit**.
12. Select **Add environment variable**.
13. Enter `USE_SSM_LAMBDA_EXTENSION` as the environment variable key.
14. Enter `true` as the value environment variable.
15. Save your work.
16. Open the Versions tab for this Lambda function.
17. Click **Publish new version**.
18. Enter a description of the change for this new version, and then click **Publish**.
19. To view the Lambda function details and exit the Versions page, click **<contact center name>-ContactLensProcessorFunction**.
20. Open the Aliases tab.
21. Click the alias.
22. Click **Edit**.
23. Select the latest version.
24. Save your work.

SEE ALSO:

ContactLensProcessorFunction Lambda Function

ContactDataSync Lambda Function

This Lambda function fetches real-time transcripts generated by Contact Lens for Amazon Connect and uploads them to Salesforce using the Connect REST API. You can manually invoke this function to backfill missing transcripts in bulk. For multi-org use cases, the function reads the secretName and accessTokenSecretName from its payload and fetches the correct configuration values from the fetched secretName.

Applies to this telephony model.

- Service Cloud Voice with Partner Telephony from Amazon Connect (Voice manually integrated with your Amazon Connect through XML import)

Configuration: This function is available if you have a Partner Amazon Contact Center with a multi-org set up created by running the multi-org cloudformation stack. This Lambda is installed with the Service Cloud Voice contact center version 19.0 and later versions.

Usage: Call this function manually from the AWS Console or by using the AWS CLI. From the AWS Console, select the contactDataSync Lambda function. From the Test tab, you can run the Lambda with a custom payload.

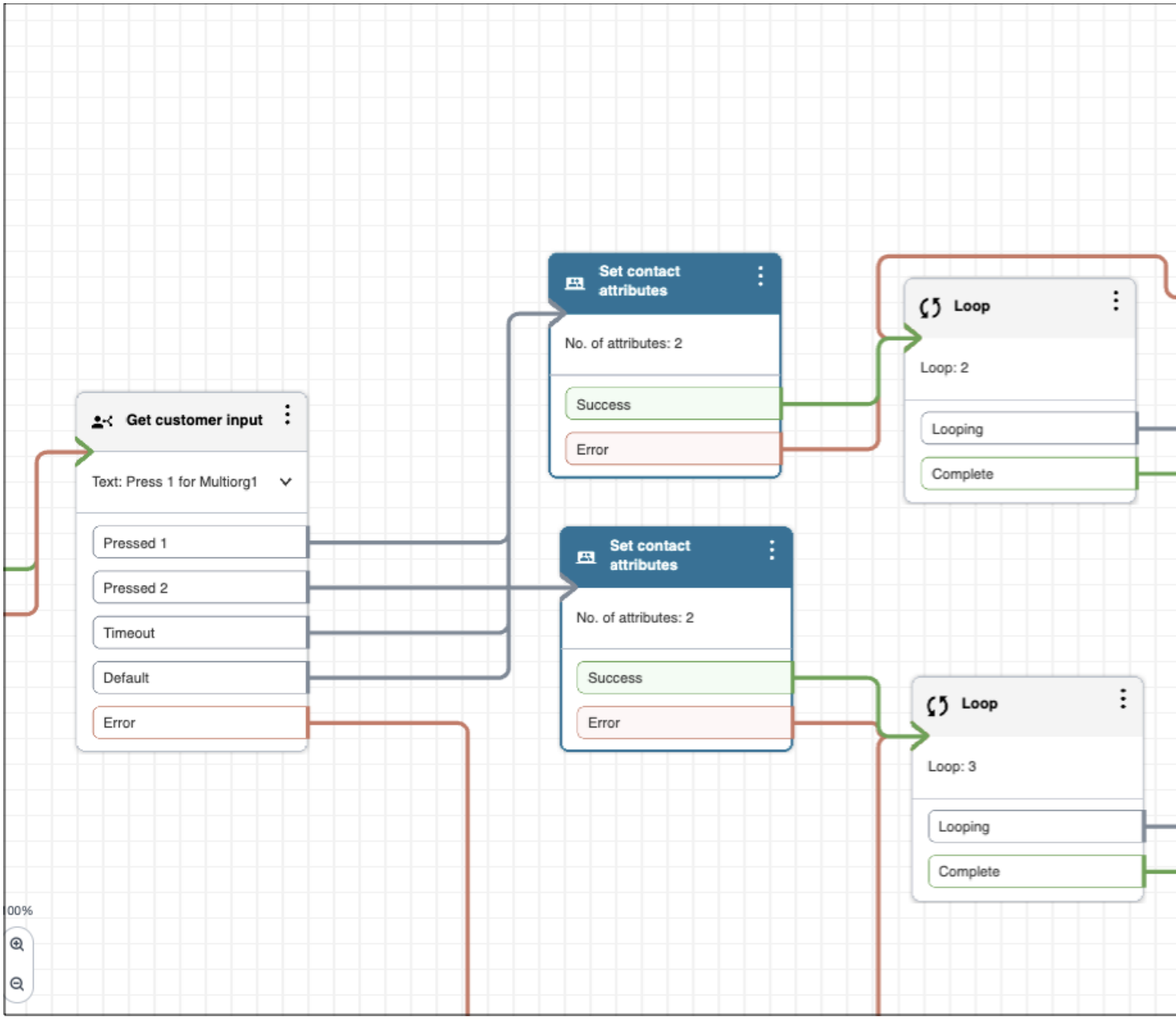
For the uploadTranscript operation, provide the secretName and accessTokenSecretName for the target Salesforce org. The contactId parameter in Amazon Connect is the unique ID of the voice call record, equivalent to vendorCallKey in Salesforce. The relatedRecords array lists the 15-digit VoiceCallId.

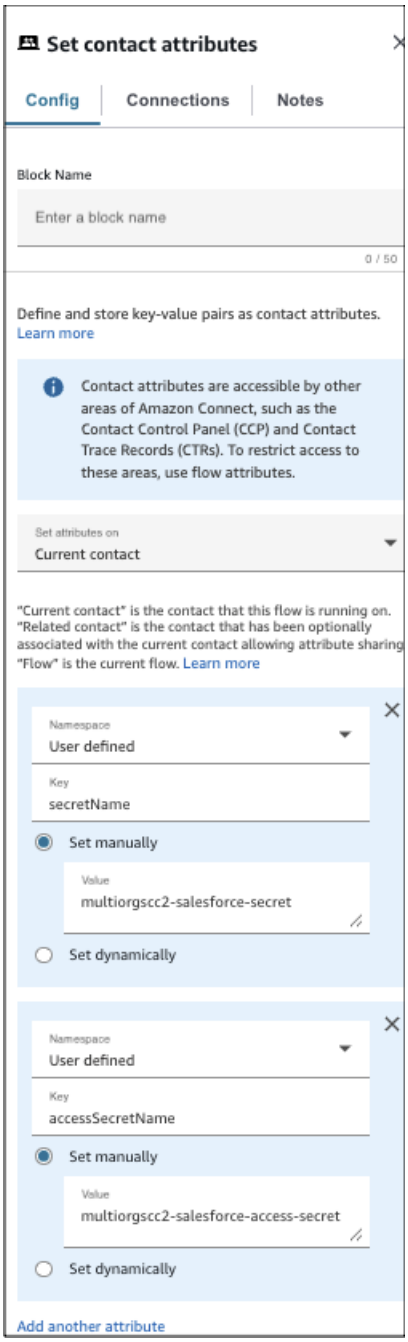
Here's an example request payload for a multiorg setup:

```
{ "operation": "uploadTranscript", "secretName": "callCenterApiName-salesforce-secret",
// org. secret name is at the top payload level, and customer needs to make sure all the
contactIds in this payload are for one org. for this corresponding org.'s secret
"accessTokenSecretName": "callCenterApiName-salesforce-access-token-secret", "payload": [
 { "contactId": "8c6258f0-66fa-4137-a61f-68311bb6d300", "relatedRecords": [ "0LQSB000001m5RR"
 ] }, { "contactId": "0e900fcd-6b3b-4445-8769-b32429eb3537", "relatedRecords": [
"0LQSB000001m1RQ" ] } ] } { "operation": "fetchUploadIdsStatus", "secretName":
"callCenterApiName-salesforce-secret", "accessTokenSecretName":
"callCenterApiName-salesforce-access-token-secret", "uploadIds": [
"f230e3d9-6c0f-3909-a0a8-1b78c634bbde", "68279b80-a394-3228-b99b-c60f02ff8227" ] }
```

Contact Flow Configuration

In the contact flow, use a Set contact attributes block to define the secretName and accessSecretName attributes for each org. Because each Salesforce org has a different secret, these attribute values must be set accordingly whenever a call is redirected to an IVR for a specific org. Subsequent Lambda functions use these contact attributes to get the correct org-specific configuration.


Get customer input
Text: Press 1 for Multiorg1
Pressed 1
Pressed 2
Timeout
Default
Error
Set contact attributes
No. of attributes: 2
Success
Error
Set contact attributes
No. of attributes: 2
Success
Error
Loop
Loop: 2
Looping
Complete
Loop
Loop: 3
Looping
Complete
00%



Within the Set contact attributes block, create two attributes with a User defined namespace.

- Key: secretName

  Value: Set this to the name of the org's secret in AWS Secret Manager, for example, multiorgscc2-salesforce-secret.

- Key: accessSecretName

  Value: Set this to the name of the org's access token secret, for example, multiorgscc2-salesforce-access-secret.

A sample contact flow is available in the GitHub.

Limitations of Multi-Org Setup

- Doesn't support call transfers across orgs in a multi-org set up because of Salesforce multi-tenancy principles that prevent record transfers from one org to another without explicit consent from the user.
- Amazon S3 bucket TTL deletion delay might be more than one day, even if it is configured as one day.
- The real-time alerts from the RealtimeAlert stack for shared AWS resources such as Lambda functions and Amazon Connect aren't specific to any org. In a multi-org setup, the real-time alert metrics for shared AWS resources are available in the dashboard in the AWS, but no notifications are available for the individual orgs.

SEE ALSO:

Salesforce Help: Update Your Contact Center

Upload or Update Transcripts with Connect REST API

CTRDataSyncFunction Lambda Function

After the contact record (previously called contact trace record or CTR) is created in Amazon, this Lambda function automatically syncs contact record data to the VoiceCall object. It finalizes the values of multiple VoiceCall object fields. This function is available to Service Cloud Voice customers who are using an Amazon Connect instance and connecting to a Kinesis stream.

Configuration: This function is available to Service Cloud Voice customers who are using an Amazon Connect instance and connecting to a Kinesis stream. The configuration is automatically performed when you create a contact center in Voice.

Description: This function performs these actions:

1. Synchronizes data between a contact record from Amazon and a VoiceCall record by taking the data from the Amazoncontact record and updating the VoiceCallrecord.
2. As part of the update, if the function finds that a VoiceCall record doesn't exist, it creates one. This behavior is useful for missed or abandoned calls.
3. If the CallDisposition field in the VoiceCall record isn't already set, sets it to Completed.

The following data is synced between the Data model for Amazon Connect contact records and a VoiceCall record:

Amazon Connect Contact Record Attribute	VoiceCall Field	Notes
InitiationTimestamp	CallStartDateTime	
DisconnectTimestamp	CallEndDateTime	
InitiationTimestamp and DisconnectTimestamp	CallDurationInSeconds	CallDurationInSeconds is inferred from InitiationTimestamp and DisconnectTimestamp.
Agent.ConnectedToAgentTimestamp	CallAcceptDateTime	The CallAcceptDateTime value matches the Agent.ConnectedToAgentTimestamp value, even if it's null.
Agent.CustomerHoldDuration	CustomerHoldDuration	The DurationInSeconds field on the VoiceCallRecording record gets its value by adding Agent.CustomerHoldDuration and

Amazon Connect Contact Record Attribute	VoiceCall Field	Notes
		Agent.AgentInteractionDuration from the Amazon contact record.
Agent.LongestHoldDuration	LongestHoldDuration	
Agent.NumberOfHolds	NumberOfHolds	
Agent.AgentInteractionDuration		This value isn't mapped to a VoiceCall field. However, the DurationInSeconds field on the VoiceCallRecording record gets its value by adding Agent.CustomerHoldDuration and Agent.AgentInteractionDuration from the Amazon contact record.
Queue.EnqueueTimestamp	CallQueuedDateTime	
Queue.Name	QueueName	
InitiationMethod	CallType (only when creating a new record)	This value is used to infer values for ToPhoneNumber and FromPhoneNumber. Also used for the CallType field if creating a new VoiceCall record. Supported InitiationMethod values include: • API • INBOUND • OUTBOUND • CALLBACK • TRANSFER
SystemEndpoint.Address	ToPhoneNumber or FromPhoneNumber	If InitiationMethod is Inbound, the value is mapped to ToPhoneNumber (when creating a new record). If InitiationMethod is Outbound, this value is mapped to FromPhoneNumber.
CustomerEndpoint.Address	ToPhoneNumber or FromPhoneNumber	If InitiationMethod is Inbound, the value is mapped to FromPhoneNumber. If initiationMethod is Outbound, this value is mapped to ToPhoneNumber (when creating a new record).
Recording.Location	CallRecordingId	The VoiceCall record creates a reference to the VoiceCallRecording record using the CallRecordingId reference ID field.

Usage: The data synchronization automatically starts when a contact record is generated. You don't need to call this function manually.

Tip: The CustomerHoldDuration and NumberOfHolds fields don't have values until after the CTRDataSyncFunction Lambda function runs. As such, to determine if the Lambda function is done, search for these values in these fields.

SEE ALSO:

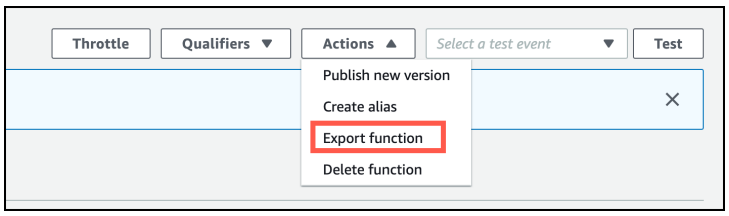
Salesforce Help: Configure Your Amazon Connect Instance

Customize Synced Contact Record Fields Using CTRDataSyncFunction

Although the provided CTRDataSyncFunction Lambda automatically syncs many of the fields between Amazon's contact record (previously called contact trace record or CTR) and a VoiceCall record, you can customize this function to sync additional fields.

The CTRDataSyncFunction Lambda documentation on page 156 lists all the fields that are automatically synced with a VoiceCall record. Use these instructions to sync more fields.

1. In your Salesforce org, create a custom field on the VoiceCall object for the desired attribute in the CTR data model.
2. In Amazon Connect, select the CTRDataSyncFunction Lambda function. From the menu select **Actions** > **Export function** so that you can get access to the function source code. If you already have the source code for this function, you don't need to perform this step.



3. In the `utils.js` file for the CTRDataSyncFunction Lambda, update the end of the **transformCTR(ctr)** function (where there's a comment to `Add custom fields here`).

```
// Check if there are custom contact attributes
if (ctr.Attributes) {
    let callAttributes = {};

    // Get contact attributes data into call attributes
    callAttributes = getCallAttributes(ctr.Attributes);

    // Add custom fields here ← ADD YOUR CODE HERE!!!

    voiceCall.callAttributes = JSON.stringify(callAttributes);
}
...
```

In that location, you can add any custom attributes to the **callAttributes** object. Add this code before the **JSON.stringify(callAttributes)** line. Your custom attributes are synced along with the automatically-synced attributes. The following example adds two fields to the list of synced fields:

```
// Check if there are custom contact attributes
if (ctr.Attributes) {
```

```
    let callAttributes = {};

    // Get contact attributes data into call attributes
    callAttributes = getCallAttributes(ctr.Attributes);

    // Add custom fields here

    // Custom field from ctr.Attributes
    if (ctr.Attributes.ConnectionAttempts) {
        callAttributes.ConnectionAttempts__c = ctr.Attributes.ConnectionAttempts;
    }
    // Custom field in ctr (but not in ctr.Attributes)
    if (ctr.DisconnectReason) {
        callAttributes.DisconnectReason__c = ctr.DisconnectReason;
    }

    voiceCall.callAttributes = JSON.stringify(callAttributes);
}
...
```

4. Deploy the new version of the Lambda function in your Amazon Connect instance.

   To learn more about managing Lambda function versions, see Lambda function versions in the AWS Lambda Developer Guide.

In subsequent calls, the new data is synced.

SEE ALSO:

AWS Lambda Developer Guide

HandleContactEventsFunction Lambda Function

This Lambda function handles the disconnect event for a missed or abandoned call and ensures that the PendingServiceRouting (PSR) record is cleaned up in Salesforce. This function is automatically invoked by an Amazon EventBridge rule after a disconnect. You don't need to call this function manually.

Configuration: This function is available to Service Cloud Voice customers who are using an Amazon Connect instance that's provided by Salesforce. This Lambda is installed with Service Cloud Voice contact center version 8.0 or later. See Update Your Contact Center for more details. No additional configuration is necessary. See a sample version of the invokeTelephonyIntegrationApi Lambda function in GitHub.

Usage: You don't need to call this function manually.

This function calls the InvokeTelephonyIntegrationApiFunction on page 145 Lambda function using the `cancelOmniFlowExecution` method. This method cancels the the Omni-Channel flow and calls the Clear Routing on page 102 REST API to clear the PendingServiceRouting (PSR) record.

MultiorgContactLensConsumerFunction Lambda Function

This Lambda function gets real-time transcripts and intelligence signals generated from Contact Lens for Amazon Connect and sends them to the ContactLensProcessorFunction Lambda function.

Applies to this telephony model.

- Service Cloud Voice with Partner Telephony from Amazon Connect (Voice manually integrated with your Amazon Connect through XML import)

Configuration: This Lambda is installed with the Service Cloud Voice contact center version 19.0 and later versions. This Lambda function is created in your AWS account, when you execute the CloudFormation template to set up multi-org.

Usage: This function reads the SecretName from a cache using the contactID.

MultiorgHandleContactEventsFunction Lambda Function

This Lambda function handles the disconnect event for a missed or abandoned call, and ensures that the Pending Service Routing (PSR) record is cleared in Salesforce.

Applies to this telephony model.

- Service Cloud Voice with Partner Telephony from Amazon Connect (Voice manually integrated with your Amazon Connect through XML import)

Configuration: This function is available if you have a Partner Amazon Contact Center with a multi-org set up created by running the multi-org cloudformation stack. This Lambda is installed with the Service Cloud Voice contact center version 19.0 and later versions. It is installed in your AWS account when you run the multi-org cloudformation stack.

Usage: This Lambda function is required for the multi-org set up to read the SecretName from a cache using the contactID. You don't have to invoke this Lambda function manually. After a call disconnects, the Amazon EventBridge automatically invokes this Lambda function.

MultiorgPostCallAnalysisTriggerFunction Lambda Function

This Lambda function automatically gets sentiment data from a contact center's Amazon S3 bucket for post call analysis.

Applies to this telephony model.

- Service Cloud Voice with Partner Telephony from Amazon Connect (Voice manually integrated with your Amazon Connect through XML import)

Configuration: This function is available if you have a Partner Amazon Contact Center with a multi-org set up created by running the multi-org cloudformation stack. This Lambda is installed with the Service Cloud Voice contact center version 19.0 and later versions.

Usage: This Lambda function is designed for use in a multi-org set up for post call analysis.

MultiorgVoiceMailPackagingFunction Lambda Function

This Lambda function creates a VoiceCall record, attaches the voicemail recording and transcription files to the record, and routes it to the Omni-Channel flow.

Applies to this telephony model.

- Service Cloud Voice with Partner Telephony from Amazon Connect (Voice manually integrated with your Amazon Connect through XML import)

Configuration: This function is available if you have a Partner Amazon Contact Center with a multi-org set up created by running the multi-org cloudformation stack. This Lambda is installed with the Service Cloud Voice contact center version 19.0 and later versions.. It is installed in your AWS account when you run the multi-org cloudformation stack.

Usage: This Lambda function is designed for use in a multi-org set up.

PostCallAnalysisTriggerFunction Lambda Function

This Lambda function automatically gets sentiment data from a contact center's Amazon S3 bucket for post-call analysis.

Configuration: This function is available to Service Cloud Voice customers who are using an Amazon Connect instance that's provided by Salesforce. This Lambda is installed with Service Cloud Voice contact center version 11.0 or later. See Update Your Contact Center for more details. No additional configuration is necessary. See a sample version of the invokeTelephonyIntegrationApi Lambda function in GitHub.

Usage: You don't need to call this function manually.

When set up, an Amazon EventBridge rule automatically triggers these actions.

1. The PostCallAnalysis Lambda function reads the post-call intelligence files from the contact center Amazon S3 bucket.
2. The PostCallAnalysis Lambda function parses the signal data in the files and reshapes it for the Salesforce Telephony Integration API to consume.
3. The Telephony Integration API saves the signals in the Salesforce database, ready to use for post-call sentiment analysis.

SEE ALSO:

Store a Post-Call Conversation Event

RealtimeAlertLambda Lambda Function

This Lambda publishes the RealtimeAlertEvent platform event using the Salesforce REST API. This function is called by CloudWatch alarms in your Amazon Connect instance. You don't need to call this function manually.

Configuration: This function is available to Service Cloud Voice customers who are using an Amazon Connect instance that's provided by Salesforce. This Lambda is installed with Service Cloud Voice contact center version 8.0 or later. See Update Your Contact Center for more details. No additional configuration is necessary. See a sample version of the invokeTelephonyIntegrationApi Lambda function in GitHub.

Usage: You don't need to call this function manually.

This Lambda function is called by the provided Cloudwatch alarms on page 174 through a Simple Notification Service (SNS) topic when an event occurs. This function performs the following steps:

1. Checks if the event is triggered by an SNS event.
2. Populates the RealtimeAlertEvent platform event fields using the SNS event data.
3. Publishes the event to Salesforce using the InvokeSalesforceRestApiFunction Lambda Function on page 142.

SEE ALSO:

Customer Alerts Alarms

Customer Alerts Platform Event

VoiceMailAudioProcessingFunction Lambda Function

This Lambda function processes voicemail recordings.

Configuration: Configure this function if you want to set up voicemail.

This function applies to the Service Cloud Voice with Partner Telephony from Amazon Connect telephony setup.

Description: Use this function to process voicemail recordings. This function isn't directly invoked from a contact flow. It's triggered when Amazon Connect generates a Contact Record (previously called contact trace record or CTR) event.

This Lambda function performs the following steps:

1. Gets the voicemail recordings from Amazon Kinesis.
2. Processes the data. Converts the chunks of recordings into a WAV file, calculates the duration of the file, and tags the file with metadata.
3. Puts the files into the voicemail_recordings folder in the contact center's Amazon S3 bucket.

For an example, see Set Up Voicemail.

SEE ALSO:

Enable Voicemail Support
VoiceMailTranscribeFunction Lambda Function
VoiceMailPackagingFunction Lambda Function

VoiceMailPackagingFunction Lambda Function

This Lambda function creates a VoiceCall record, attaches the voicemail recording and transcription files to the record, and routes it all to the Omni-Channel flow.

Configuration: Configure this function if you want to set up voicemail.

This function applies to the Service Cloud Voice with Partner Telephoney from Amazon Connect telephony setup.

Description: This function creates a VoiceCall record, attaches the voicemail recording and transcription files to the record, and routes it all to the Omni-Channel flow. This function isn't directly invoked from a contact flow. It's triggered when a voicemail transcription file is added to the voicemail_transcriptions folder in the contact center's Amazon S3 bucket.

This Lambda function performs the following steps:

1. Gets the voicemail recording and transcription files from the contact center's Amazon S3 bucket.
2. Creates a VoiceCall record. Sets the `callOrigin` parameter to Voicemail, and sets other parameters, including `from`, `to`, and `duration`.
3. Attaches the voicemail recording and the transcription files to the VoiceCall record.
4. Routes the VoiceCall record to the Omni-Channel flow that was configured in the contact center.

For an example, see Set Up Voicemail.

SEE ALSO:

Enable Voicemail Support
VoiceMailAudioProcessingFunction Lambda Function
VoiceMailTranscribeFunction Lambda Function

VoiceMailTranscribeFunction Lambda Function

This Lambda function transcribes voicemail recordings.

Configuration: Configure this function if you want to set up voicemail.

This function applies to the Service Cloud Voice with Partner Telephoney from Amazon Connect telephony setup.

Description: This function transcribes voicemail recordings. This function isn't directly invoked from a contact flow. It's triggered when a voicemail recording file is added to the voicemail_recordings folder in the contact center's Amazon S3 bucket.

This Lambda function performs the following steps:

1. Gets the voicemail recording files from the contact center's Amazon S3 bucket.
2. Transcribes the recordings.
3. Puts the files into the voicemail_transcriptions folder in the contact center's Amazon S3 bucket.

For an example, see Set Up Voicemail.

SEE ALSO:

- Enable Voicemail Support
- VoiceMailAudioProcessingFunction Lambda Function
- VoiceMailPackagingFunction Lambda Function

kvsConsumerTrigger Lambda Function

This Lambda function is the initiation point for starting real-time transcription.

Configuration: This function is available to Service Cloud Voice customers who are using an Amazon Connect instance that's provided by Salesforce. No additional configuration is necessary. See a sample version of the invokeTelephonyIntegrationApi Lambda function in GitHub.

Sample Contact Flows That Use This Function: Sample_SCV_Inbound_Flow_With_Transcription, Sample SCV Outbound Flow with Transcription Using Contact Lens, Sample SCV Outbound Flow with Transcription Using Amazon Transcribe. To download these flows, visit our Sample Contact Flows folder in GitHub.

Usage: The kvsConsumerTrigger Lambda is invoked from a contact flow to begin transcription. The function triggers the kvsTranscriber on page 163 Lambda function with the necessary information to start real-time transcription. To learn how to use Amazon Transcribe's custom vocabulary and vocabulary filtering, see Improve Amazon Transcribe's Transcription Accuracy with Custom Vocabulary and Filters on page 8.

kvsTranscriber Lambda Function

This Lambda function does the actual real-time transcription work. This function is not directly invoked from a contact flow. It is triggered by the kvsConsumerTrigger Lambda function. You do not need to call this function manually.

Configuration: This function is available to Service Cloud Voice customers who are using an Amazon Connect instance that's provided by Salesforce. No additional configuration is necessary. See a sample version of the invokeTelephonyIntegrationApi Lambda function in GitHub.

Usage: The kvsConsumerTrigger on page 163 Lambda function automatically calls this function. You do not need to call this function manually.

This Lambda function performs the following steps:

1. Generates the JWT token that is required to authenticate into the Salesforce Telephony Integration API.
2. Starts Amazon real-time transcription and hooks the audio stream (between rep and end user) to the transcription service.

3. Parses the generated transcripts and invokes the CreateMessage API on the telephony Integration Service to stream the transcripts into Salesforce.
4. Stops the transcription when the call ends.

To learn how to use Amazon Transcribe's custom vocabulary and vocabulary filtering, see Improve Amazon Transcribe's Transcription Accuracy with Custom Vocabulary and Filters on page 8.

Query AWS Lambda Functions to Analyze a Voice Call

Query AWS Lambda function logs and Amazon Connect flows, and use the query results to analyze and troubleshoot the events that happen throughout the stages of a single voice call.

A single voice call journey invokes several different AWS Lambda functions and Amazon Connect flows. Perform a query through Amazon CloudWatch Log Insights to find all voice call events across all Lambda functions for a specific voice call.

This feature is installed with Service Cloud Voice contact center version 13.0 or later. See the Update Your Contact Center Knowledge Article for more details. No additional configuration is necessary.

To analyze a voice call through an AWS Lambda function query:

1. Log in to Salesforce and find the name of the vendor call key (for example, `12faf34a-f1d5-6b07-89fc-6e3bce61d65f`) and the date range of the voice call record you want to analyze.
2. Log in to Amazon AWS.
3. Open the CloudWatch console.
4. In the navigation pane, select **Logs > Logs Insights**.
5. In the **Select log group(s)** picklist, select the log groups that include the name of the contact center you want to query. The following example is a list of log groups for a contact center named ContactCenter1. The list includes AWS Lambda and Amazon Connect Flow log groups.

```
/aws/connect/contactcenter100db0000006.nep
/aws/lambda/ContactCenter1-AuthKeysSSMultiFunction
/aws/lambda/ContactCenter1-CTRDataSyncFunction
/aws/lambda/ContactCenter1-HandleContactEventsFunction
/aws/lambda/ContactCenter1-InvokeSalesforceRestApiFunction
/aws/lambda/ContactCenter1-InvokeTelephonyIntegrationApiFunction
/aws/lambda/ContactCenter1-kvsConsumerTrigger
/aws/lambda/ContactCenter1-kvsTransriber
/aws/lambda/ContactCenter1-RealtimeAlert
/aws/lambda/ContactCenter1-RententionPeriodFunction
/aws/lambda/ContactCenter1-VoiceMailAudioProcessingFunction
```

6. In the query editor, type the following query, where `VENDOR_CALL_KEY` is the unique ID of the voice call (`VoiceCall`) record within the telephony system. The query is case sensitive.

```
fields @timestamp, @message, @logStream, @log
| filter ContactId = 'VENDOR_CALL_KEY' OR context.contactId = 'VENDOR_CALL_KEY' | sort
 @timestamp asc
```

For example,

```
fields @timestamp, @message, @logStream, @log
| filter ContactId = '12faf34a-f1d5-6b07-89fc-6e3bce61d65f' OR context.contactId =
```

```
'12faf34a-f1d5-6b07-89fc-6e3bce61d65f'
| sort @timestamp asc
```

To learn more about the query syntax, see the Amazon CloudWatch Logs Insights Query Syntax documentation.

7. Specify the date range of the voice call record you want to query, click **Custom**, select the **Absolute** tab, and then select and apply the date range of the voice call.
8. Click **Run query**. Amazon CloudWatch finds all logs across all Amazon Lambda functions and Amazon Connect flows that are related to the specified voice call record and displays the results in the Logs tab.

   The first event is created when the voice call record begins. Expand each event to view its details, including the name of the Amazon Connect flow or AWS Lambda output, and any error messages.

SEE ALSO:

Knowledge Article: Update Your Contact Center

Object Reference for the Salesforce Platform: VoiceCall

Amazon CloudWatch: CloudWatch Logs Insights Query Syntax

CHAPTER 8 Amazon Connect Flows

In this chapter ...

An Amazon Connect flow depicts a customer's voice call journey from beginning to end. Configure Service Cloud Voice flows for your contact center to determine how Salesforce handles inbound calls, outbound calls, and call transfers.

Each Service Cloud Voice flow is configured in Amazon Connect using a flow map made up of blocks, beginning with an Entry block and ending with a Transfer to flow or Disconnect block. To learn more about Amazon flows, see the Amazon Connect Administrator Guide.

Salesforce provides several out-of-the box Sample Service Cloud Voice flows in GitHub you can customize for your Service Cloud Voice contact center environment. For example, the Sample SCV Inbound Flow is used to route incoming voice calls. When a customer calls the contact center, the call triggers the inbound flow starting at the Entry block. The call then traverses the flow, block by block, stopping at each block to determine what happens next. The flow eventually arrives at the Transfer to flow block, which ends this flow and transfers the call to the Omni-Channel Subflow. If errors are encountered at any point in the flow, the call gets routed to the Disconnect block.

A subflow is a flow within a flow. Salesforce offers several out-of-the-box subflows that let you add features to a flow, such as Omni-Channel routing, callbacks, and voicemails. A subflow can't be used on its own; instead, it must be added to a flow using the **Transfer to flow** block. You can add multiple subflows to a flow by daisy-chaining them together. In fact, some of the sample flows include several subflows daisy-chained together. For example, the Sample SCV Inbound Flow includes the Sample SCV Omni-Channel Subflow, which then transfers out to a series of other subflows ending with the Sample SCV Transcription Subflow with Contact Lens.

Here's how an inbound call flows through the daisy chain of subflows using the Salesforce-provided "Sample SCV" flows:

1. A call comes in, which triggers the Sample SCV Inbound Flow.
2. The call moves through the Sample SCV Inbound Flow, creating a voice call (`VoiceCall`) record and setting the contact attribute for `voiceCallId` before it's transferred to the Sample SCV Omni-Channel Subflow.
3. The call moves through the Sample SCV Omni-Channel Subflow, routing and prioritizing the call based on the routing logic of the specified Omni-Channel flow. After this, the call is transferred to the Sample SCV Callback Subflow.
4. The call moves through the Sample SCV Callback Subflow, giving the caller the option to schedule a callback if the queue is busy. If the caller chooses to schedule a callback, the call is transferred to the Callback queue. If the caller chooses to remain on the line, the call is transferred to the Sample SCV Transcription Subflow with Contact Lens.
5. The call moves through the Sample SCV Transcription Subflow with Contact Lens, enabling real-time transcription before finally transferring the call to the queue. This ends the entire flow.

The order of the subflow chain matters. For example, for inbound calls, if you place the Omni-Channel Subflow before the Callback Subflow, callback calls will be routed correctly. If you flip the subflows by

placing the Callback Subflow before the Omni-Channel Subflow, callback calls will be routed to the basic queue.

For inbound calls, you can't point a phone number directly to a subflow; instead, point your phone number to the inbound flow. Subflows don't create a voice call record in Salesforce; they rely on the inbound flow to do that.

Note: Service Cloud Voice doesn't support queue transfers for voice calls already in one queue and transferred to another queue using the Transfer to queue block in the Amazon Connect flow. In this scenario, the call recording and transcription are available in Amazon Connect but not Salesforce. Alternatively, you can use routing profiles to transfer voice calls to groups of reps to take calls for queues they don't typically handle when certain criteria are met. See Managing Increases in Call Volumes with Routing Profiles.

SEE ALSO:

GitHub: Sample Service Cloud Voice Flows

Amazon Connect Flow Documentation

Amazon Connect Flows Best Practices

Follow these best practices to ensure your Amazon Connect flows are configured optimally and correctly. If your environment isn't set up correctly for voice resiliency, you may experience missing calls.

Voice Resiliency

Voice resiliency ensures that the telephony system stays operational when the number of conversations is over limit or when certain background services are affected. See the Voice Resiliency for Service Cloud Voice knowledge article for more information about voice resiliency.

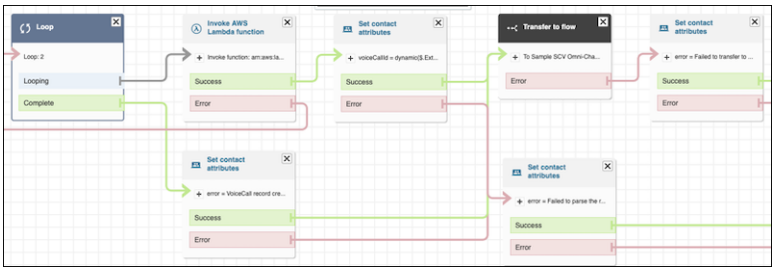
If you customized any of the following Sample Service Cloud Voice flows in GitHub, verify they are configured to support voice resiliency:

- Sample SCV Inbound Flow
- Sample SCV Transfer Flow For Agent Transfers
- Sample SCV Transfer Flow For Omni-Channel Transfers
- Sample SCV Transfer Flow For Queue Transfers
- Sample SCV Callback Subflow

To support voice resiliency, make sure the the inbound, transfer, and callback flows do the following if the Invoke AWS Lambda function block fails:

- Routes the voice call to a rep or queue.
- Doesn't hang up/disconnect the call, even if a contact flow block errors out.

The Sample SCV flows in GitHub for inbound, transfer, and callback flows are configured to support voice resiliency. Here's an example from the Sample SCV Inbound Flow contact flow:



Sample Flows for Service Cloud Voice

Salesforce provides several out-of-the box sample flows for Amazon Connect you can use to create inbound, outbound, and transfer flows for your Service Cloud Voice contact center environment.

The sample flows are automatically installed when you create a contact center. All sample flows begin with the name "Sample SCV." There are two ways to access the sample flows:

- Access the sample flows directly in Amazon Connect. Log in to your Amazon Connect instance and select Routing > Contact flows.
- Get the JSON sample flow files from GitHub. Download the JSON files from the Sample Service Cloud Voice flows page in GitHub and import them into your Amazon Connect instance. For instructions on how to import flows, see the Import/export flows page in the Amazon Connect Administrator Guide.

The sample flows are occasionally updated and released with Salesforce contact center updates. To ensure you see the latest versions of the sample flows in Amazon Connect, first verify that your contact center is updated to the latest version. The JSON files in GitHub are always updated to the latest version.

Salesforce offers the following sample flows, which you can customize for your Service Cloud Voice contact center environment:

Inbound Flows

- **Sample SCV Inbound Flow**: Creates a voice call record in Salesforce for the inbound call and transfers the call to the SCV Omni-Channel subflow. Associate your phone number with this flow so customers can reach your contact center. JSON file

  If you customize the Sample SCV Inbound Flow, make sure it's configured to support voice resiliency. See the Amazon Connect Flows Best Practices on page 168 page for more information.

- **Sample SCV Inbound Flow for Direct to Rep:** Creates a voice call record in Salesforce for the inbound call and transfers the call to the SCV Direct to Rep subflow. This flow is for use cases where each rep has a unique assigned phone number and receives calls at that number. JSON file

Outbound Flows

- **Sample SCV Outbound Flow With Transcription Using Amazon Transcribe**: Starts media streaming. Also starts transcription using Amazon Transcribe, capturing and transcribing audio for outbound voice calls. If you use this preferred flow, you can't use the Sample SCV Outbound Flow With Transcription Using Amazon Transcribe flow. Since the Sample SCV Outbound Flow With Transcription Using Contact Lens on page 12 offers more robust transcription features, consider using that flow instead of this one. JSON file
- **Sample SCV Outbound Flow With Transcription Using Contact Lens**: Starts transcription using Contact Lens for Amazon Connect, capturing and transcribing audio for outbound voice calls. If you use this preferred flow, you can't use the Sample SCV Outbound Flow With Transcription Using Amazon Transcribe flow. JSON file
- **Sample SCV Outbound Flow From Rep With CallerId Override**: Makes an outbound call with the caller ID displayed as the number assigned to a specific rep. JSON file
- **Sample SCV Outbound Flow From Rep With Voicemail Drop**: Starts transcription and media streaming for an outbound call, allowing the rep to drop a pre-recorded voicemail if an answering machine is detected. Reps can disconnect and move to the next work item while the system plays the recording. JSON file

Transfer Flows

- **Sample SCV Transfer Flow For Agent Transfers**: Creates a voice call record and transfers the call from one rep to another specified rep. JSON file
- **Sample SCV Transfer Flow For Omni-Channel Transfers**: Creates a voice call record and transfers the call from a rep to a specified Omni-Channel flow. JSON file
- **Sample SCV Transfer Flow For Omni Routing Transfers**: Creates a voice call record and transfers the call from a rep to a specified rep or queue using Omni-Channel Unified Routing. JSON file
- **Sample SCV Transfer Flow For Queue Transfers**: Creates a voice call record and transfers the call from a rep to a specified queue. JSON file

If you customize the Sample SCV Transfer flows, make sure it's configured to support voice resiliency. See the Amazon Connect Flows Best Practices on page 168 page for more information.

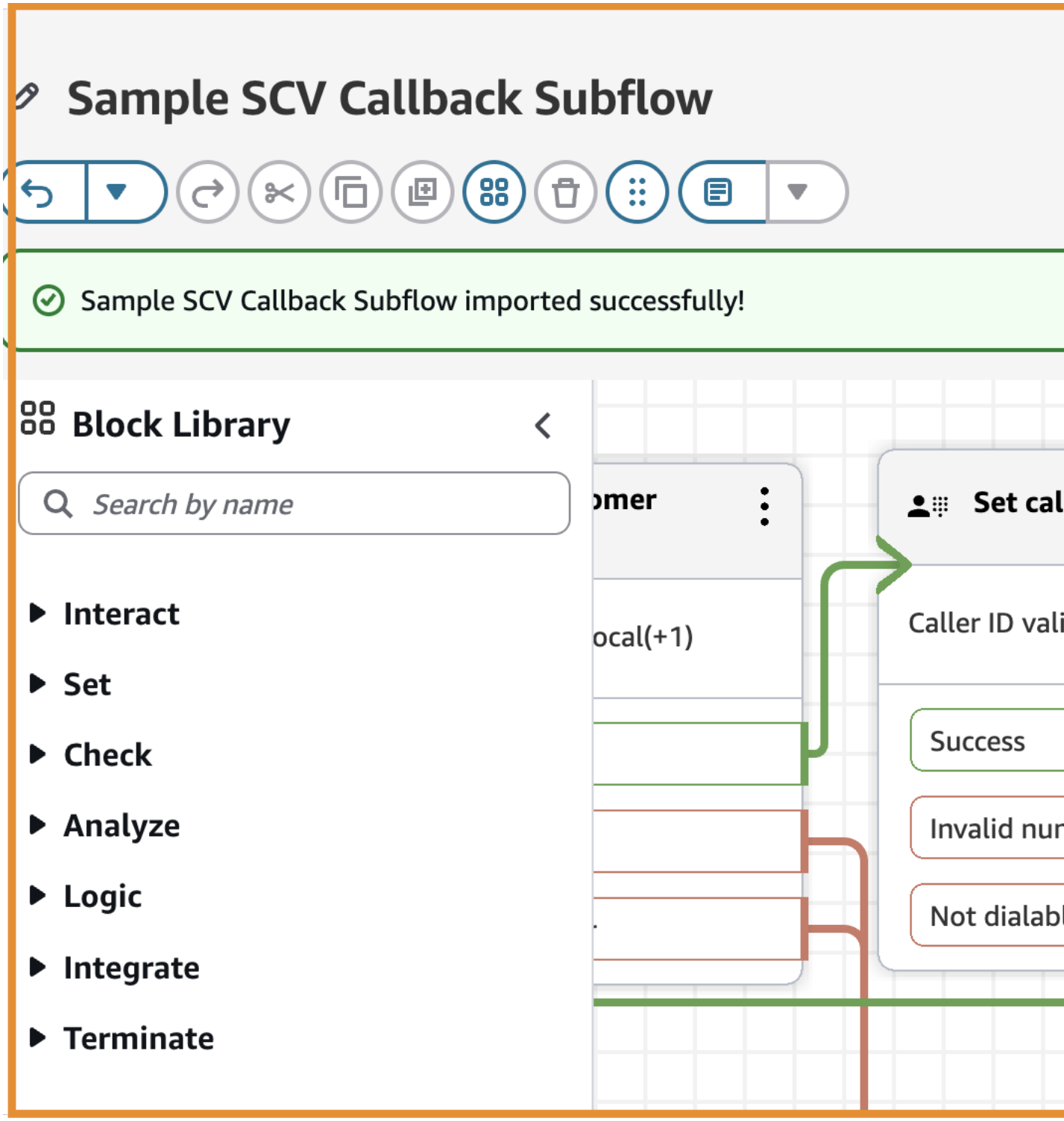
Rep Whisper Flows

- **Sample SCV Agent Whisper Flow For Amazon Transcribe**: Starts media streaming. Also starts transcription using Amazon Transcribe to prepare for inbound voice calls and voice call transfers. JSON file

Subflows

- **Sample SCV Callback Subflow**: Gives the caller the option to schedule a callback if the queue is busy. If the caller chooses to schedule a callback, transfers the voice call to the Callback queue. If the caller chooses to remain on the line, transfers the call to one of the SCV Transcription Subflows. JSON file

  If the caller schedules a callback and then disconnects, the call can be incorrectly classified as abandoned. To correctly classify callback request drops, use a Callback Contact Attribute block in the flow.



If you customize the Sample SCV Callback Subflow, make sure it's configured to support voice resiliency. See the Amazon Connect Flows Best Practices on page 168 page for more information.

- **Sample SCV Direct to Rep Subflow**: Routes the call to a specific rep associated with the dialed number and transfers the call to the Sample SCV Transcription Subflow With Contact Lens. JSON file
- **Sample SCV Field Service Phone Call Subflow**: Connects field reps to contact center reps. JSON file

- **Sample SCV Omni-Channel Subflow - Basic Routing With Case Creation**: Creates a case for each inbound voice call, routes the call to a queue, and opens a screen-pop for the new case record when a rep accepts the call. JSON file
- **Sample SCV Omni-Channel Subflow - Omni Routing With Case Creation**: Creates a case for each inbound voice call, routes the call using Omni-Channel Unified Routing, and opens a screen-pop for the new case record when a rep accepts the call. JSON file
- **Sample SCV Omni-Channel Subflow - Omni Routing for Skills**: Creates a case for each inbound voice call, routes the call to a rep with a skill using Omni-Channel Unified Routing, and opens a screen-pop for the new case record when a rep accepts the call. JSON file
- **Sample SCV Transcription Subflow With Amazon Transcribe**: Enables call recording and real-time transcription using Amazon Transcribe, and then transfers the voice call to the queue. JSON file
- **Sample SCV Transcription Subflow With Contact Lens**: Enables call recording and real-time transcription using Contact Lens, and then transfers the voice call to the queue. JSON file
- **Sample SCV Voicemail Subflow**: Gives the caller the option to leave a voicemail. Reps can play the recordings and read the transcriptions of all voicemails routed to them. JSON file
- **Sample SCV Voicemail Subflow for Omni Routing**: Gives the caller the option to leave a voicemail when using Omni-Channel Unified Routing. Reps can play the recordings and read the transcriptions of all voicemails routed to them. JSON file

Other Flows

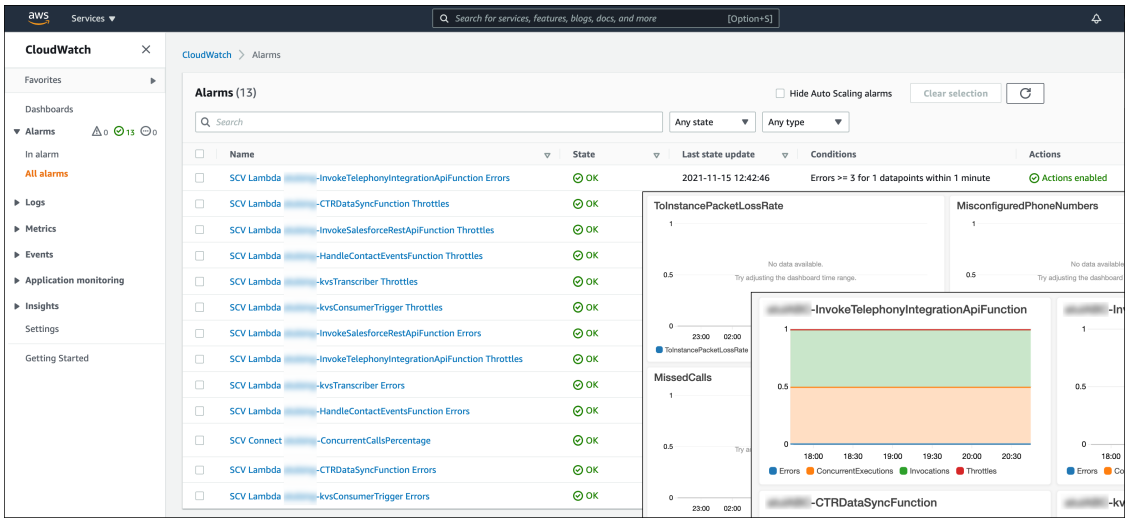
These flows demonstrate how you can use the `InvokeSalesforceRestApiFunction` Lambda function on page 142 to perform some common tasks with the Salesforce REST API.

- **Sample_SCV_REST_Check_For_Open_Cases:** Checks for open cases using the `InvokeSalesforceRestApiFunction` Lambda function. JSON file
- **Sample_SCV_REST_Link_Call_To_Case**: Links a voice call to an open case using the `InvokeSalesforceRestApiFunction` Lambda function. This flow adds on to the Sample SCV Invoke REST API Check For Open Cases flow. JSON file

CHAPTER 9 Service Cloud Voice Customer Alerts with CloudWatch

In this chapter ...

Ensure that your Service Cloud Voice implementation is running properly with the Amazon CloudWatch dashboards and alarms that we provide. Monitor the health of your contact center with two dashboards. Fine-tune the alarms so they trigger when something needs attention in your environment. Notify the right people about these events with the new RealtimeAlertEvent platform event.



These customer alert features are available to all customers running version 8.0 or later of the Service Cloud Voice contact center in Amazon. To learn how to view or update your contact center, see Update Your Contact Center in Salesforce Help.

SEE ALSO:

Salesforce Help: Update Your Contact Center

Customer Alerts Alarms

Salesforce provides a set of alarms that you should fine-tune so that you can monitor the status of your contact center and get notified when important events are triggered. Some of these alarms are enabled by default, others must be enabled if you want them to operate in your environment.

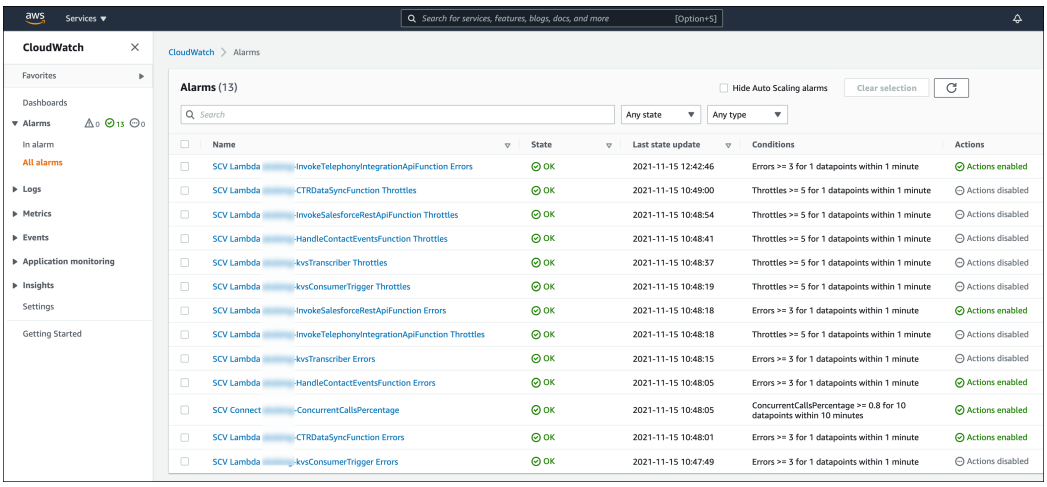
These alarms fall into several different categories:

Concurrent Calls
One alarm triggers when there are a high number of concurrent calls. This alert can indicate if you're reaching Amazon's concurrent call limit. Upon reaching the limit, you can either file an AWS support ticket to increase your quota, or you can adjust some other aspect of your implementation.

Errors
Several alarms trigger when there are errors with the Lambda functions that are used by Service Cloud Voice.

Throttle
Several alarms trigger when Lambda functions get throttled because of the large number of concurrent invocations.

You can fine-tune any of these alarms to suit your use case.



You can monitor these alarms directly from within CloudWatch, using the provided dashboards on page 178, or by listening for the RealtimeAlertEvent platform event on page 179 in your Salesforce org. The alarms automatically trigger this platform event using a Lambda function (RealtimeAlertLambda Lambda Function on page 161).

The following alarms are automatically installed with the Customer Alerts template. Review the **Initial State** column to see which alarms are automatically turned on by default. To learn how to enable alarms, or how to fine-tune alarms, see Using Amazon CloudWatch alarms in the Amazon CloudWatch User Guide.

Alarm Type	Alarm Name	Description	Initial State	Suggested Actions
General Alarms				
Concurrent Calls	SCV Connect Concurrent Calls Percentage	Triggers when the number of concurrent active voice calls in the connect instance during the evaluation period is greater than the threshold.	Enabled	Contact Amazon to increase number of concurrent voice calls allowable for your implementation. Contact your Salesforce admin for help.

Alarm Type	Alarm Name	Description	Initial State	Suggested Actions
		Criteria: ConcurrentCallsPercentage >= 0.8 for 10 data points within 10 minutes.		
InvokeTelephonyIntegrationApiFunction on page 145 Lambda Alarms				
Errors	SCV Lambda InvokeTelephonyIntegrationApiFunction Errors	The number of invocations of this Lambda function that resulted in a function error. Function errors include exceptions thrown by your code and exceptions thrown by the Lambda runtime. Criteria: Errors >= 3 for 1 data point within 1 minute.	Enabled	Check Salesforce Trust for maintenance or Service Cloud Voice service disruption. Contact the Salesforce support team.
Throttle	SCV Lambda InvokeTelephonyIntegrationApiFunction Throttles	The number of invocation requests of this Lambda function that are throttled. Review concurrent executions quota for the region, or the reserved concurrency limit that you configured on the function. Criteria: Errors >= 5 for 1 data point within 1 minute.	Disabled	Increase the concurrency limit for the Lambda function. Check for guidance with Amazon customer support.
InvokeSalesforceRestApiFunction on page 142 Lambda Alarms				
Errors	SCV Lambda InvokeSalesforceRestApiFunction Errors	The number of invocations of this Lambda function that resulted in a function error. Function errors include exceptions thrown by your code and exceptions thrown by the Lambda runtime. Criteria: Errors >= 3 for 1 data point within 1 minute.	Disabled	Check Salesforce Trust for maintenance or Service Cloud Voice service disruption. Contact the Salesforce support team.
Throttle	SCV Lambda InvokeSalesforceRestApiFunction Throttles	The number of invocation requests of this Lambda function that are throttled. Review concurrent executions quota for the region, or the reserved	Disabled	Increase the concurrency limit for the Lambda function. Check for guidance with Amazon customer support.

Alarm Type	Alarm Name	Description	Initial State	Suggested Actions
		concurrency limit that you configured on the function. Criteria: Errors >= 5 for 1 data point within 1 minute.		
HandleContactEvents on page 159 Lambda Alarms				
Errors	SCV Lambda HandleContactEvents Errors	The number of invocations of this Lambda function that resulted in a function error. Function errors include exceptions thrown by your code and exceptions thrown by the Lambda runtime. Criteria: Errors >= 3 for 1 data point within 1 minute.	Enabled	Check Salesforce Trust for maintenance or Service Cloud Voice service disruption. Contact the Salesforce support team.
Throttle	SCV Lambda HandleContactEvents Throttles	The number of invocation requests of this Lambda function that are throttled. Review concurrent executions quota for the region, or the reserved concurrency limit that you configured on the function. Criteria: Errors >= 5 for 1 data point within 1 minute.	Disabled	Increase the concurrency limit for the Lambda function. Check for guidance with Amazon customer support.
CTRDataSyncFunction on page 156 Lambda Alarms				
Errors	SCV Lambda CTRDataSyncFunction Errors	The number of invocations of this Lambda function that resulted in a function error. Function errors include exceptions thrown by your code and exceptions thrown by the Lambda runtime. Criteria: Errors >= 3 for 1 data point within 1 minute.	Enabled	Check Salesforce Trust for maintenance or Service Cloud Voice service disruption. Contact the Salesforce support team.
Throttle	SCV Lambda CTRDataSyncFunction Throttles	The number of invocation requests of this Lambda function that are throttled. Review concurrent executions quota for the region, or the reserved	Disabled	Customer should increase the concurrency limit for the Lambda function. Check for guidance with Amazon customer support.

Alarm Type	Alarm Name	Description	Initial State	Suggested Actions
		concurrency limit that you configured on the function. Criteria: Errors >= 5 for 1 data point within 1 minute.		
kvsConsumerTrigger on page 163 Lambda Alarms				
Errors	SCV Lambda kvsConsumerTrigger Errors	The number of invocations of this Lambda function that resulted in a function error. Function errors include exceptions thrown by your code and exceptions thrown by the Lambda runtime. Criteria: Errors >= 3 for 1 data point within 1 minute.	Disabled	Check Salesforce Trust for maintenance or Service Cloud Voice service disruption. Contact the Salesforce support team.
Throttle	SCV Lambda kvsConsumerTrigger Throttles	The number of invocation requests of this Lambda function that are throttled. Review concurrent executions quota for the region, or the reserved concurrency limit that you configured on the function. Criteria: Errors >= 5 for 1 data point within 1 minute.	Disabled	Increase the concurrency limit for the Lambda function. Check for guidance with Amazon customer support.
kvsTranscriber on page 163 Lambda Alarms				
Errors	SCV Lambda kvsTranscriber Errors	The number of invocations of this Lambda function that resulted in a function error. Function errors include exceptions thrown by your code and exceptions thrown by the Lambda runtime. Criteria: Errors >= 3 for 1 data point within 1 minute.	Disabled	Check Salesforce Trust for maintenance or Service Cloud Voice service disruption. Contact the Salesforce support team.

Alarm Type	Alarm Name	Description	Initial State	Suggested Actions
Throttle	SCV Lambda kvsTranscriber Throttles	The number of invocation requests of this Lambda function that are throttled. Review concurrent executions quota for the region, or the reserved concurrency limit that you configured on the function. Criteria: Errors >= 5 for 1 data point within 1 minute.	Disabled	Increase the concurrency limit for the Lambda function. Check for guidance with Amazon customer support.

SEE ALSO:

Amazon CloudWatch User Guide: Using Amazon CloudWatch alarms

Customer Alerts Dashboards

Customer Alerts Platform Event

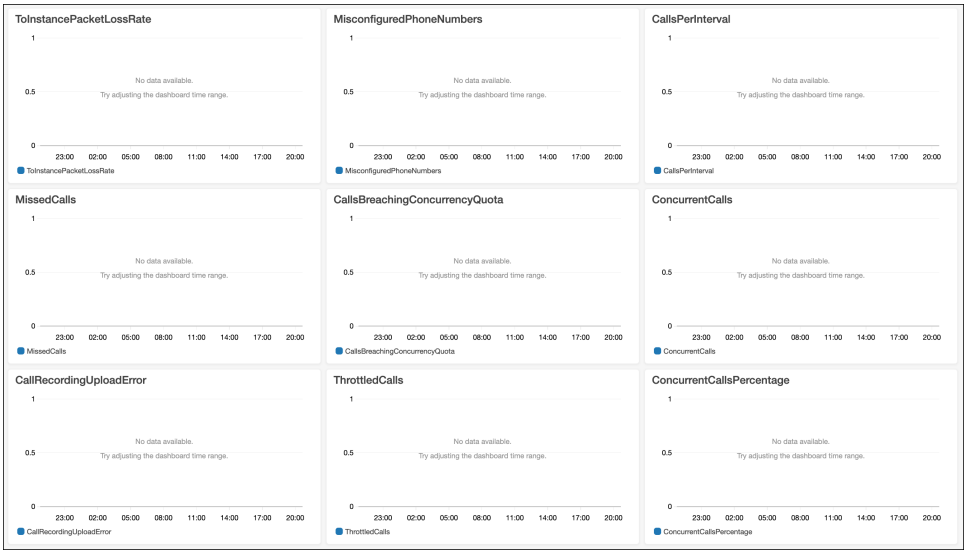
Customer Alerts Dashboards

Salesforce provides two CloudWatch custom dashboards to monitor potential issues with your contact center.

You can fine-tune these dashboards as necessary for your Voice implementation. To learn how to customize dashboards, see Using Amazon CloudWatch dashboards in the Amazon CloudWatch User Guide.

Service Cloud Voice Connect Dashboard

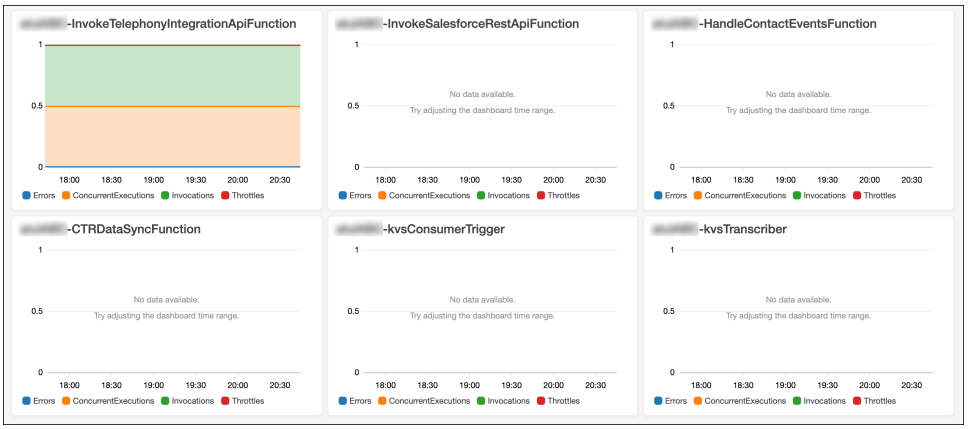
The Service Cloud Voice Connect Dashboard shows general activity associated with your Amazon Connect implementation.



To learn what each of the metrics means, see Monitoring your instance using CloudWatch in the Amazon Connection Administrative Guide.

Service Cloud Voice Lambda Dashboard

The Service Cloud Voice Lambda Dashboard shows issues associated with the Lambda functions in your Amazon Connect instance.



This dashboard tracks invocations, concurrent executions, and errors associated with each Lambda function.

SEE ALSO:

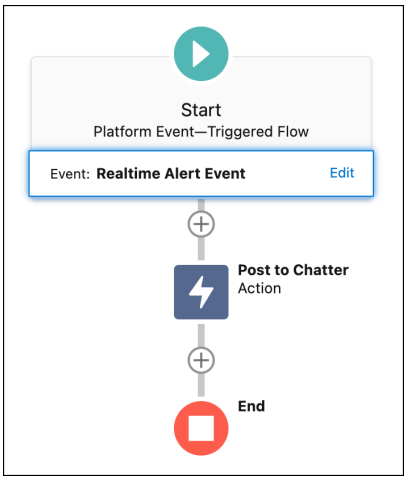
Amazon CloudWatch User Guide: Using Amazon CloudWatch dashboards

Amazon Connect Administrator's Guide: Monitoring your instance using CloudWatch

Customer Alerts Platform Event

The RealtimeAlertEvent platform event notifies subscribers about triggered alarms in your Amazon Connect instance.

This event is a standard Salesforce platform event and can be accessed from the following endpoint: `/event/RealtimeAlertEvent`. You can handle this event from within Apex code, or by using a Salesforce flow. For example, you can create a flow that is triggered by the platform event and then posts a message to Chatter.



To learn more about this platform event and to view some sample code, see RealtimeAlertEvent in the Platform Events Developer Guide.

SEE ALSO:

Platform Events Developer Guide: RealtimeAlertEvent

Customer Alerts Alarms

CHAPTER 10 Service Cloud Voice Limits and Scaling

In this chapter ...

Service Cloud Voice runs in a multitenant environment and uses both Salesforce resources and Amazon Connect resources to provide a native Salesforce telephony solution.

To ensure quality for all Voice users, Voice enforces the following limits on calls and transcription.

SEE ALSO:

Amazon Connect

Service Cloud Voice Limits per Salesforce Org

Service Cloud Voice observes limits for each Salesforce org.

Refer to *Salesforce Help*: Service Cloud Voice Limits and Limitations to confirm the limits for your Salesforce org.

Service Cloud Voice Best Practices for Error Handling

To ensure service availability and an optimal customer experience, follow these best practices for managing throughput limits and errors.

Monitor Transcription-Related Limit Errors
When your org exceeds the transcription limit, utterances are no longer transcribed. Track when your org hits the transcription limit by using a custom field on the VoiceCall object.

Monitor VoiceCall Limit Errors
When your org exceeds the limit for the maximum number of concurrent calls, the system can't create VoiceCall records. Track when your org hits the concurrent call limit or the phone calls initiated per second limit by checking output of the InvokeTelephonyIntegrationApi Lambda function.

Monitor Transcription-Related Limit Errors

When your org exceeds the transcription limit, utterances are no longer transcribed. Track when your org hits the transcription limit by using a custom field on the VoiceCall object.

When your org hits the transcription limit during a call, the kvsTranscriber Lambda function creates a contact attribute named sf_realtime_transcription_status__c and sets its value to the limit error message. When the call ends, the CTRDataSync Lambda function passes this value to Salesforce, updating the custom field on the affected VoiceCall record for the call where the transcription exceeded the limit. If the custom field on the VoiceCall record is populated, it means that the org exceeded the limit during that call. You can look at VoiceCall records or create a custom report to aggregate the data.

1. In Salesforce, create a custom field called `sf_realtime_transcription_status__c` on the VoiceCall object.
2. Add the custom field to the VoiceCall record page layout.
3. Create a custom report that pulls records where the `sf_realtime_transcription_status__c` field value is populated to see which calls were impacted by the transcription limit.

Important: If you change the name of the custom field to a different value than `sf_realtime_transcription_status__c`, you must manually modify both kvsTranscriber and CTRDataSync Lambda functions to accept the new name.

Monitor VoiceCall Limit Errors

When your org exceeds the limit for the maximum number of concurrent calls, the system can't create VoiceCall records. Track when your org hits the concurrent call limit or the phone calls initiated per second limit by checking output of the InvokeTelephonyIntegrationApi Lambda function.

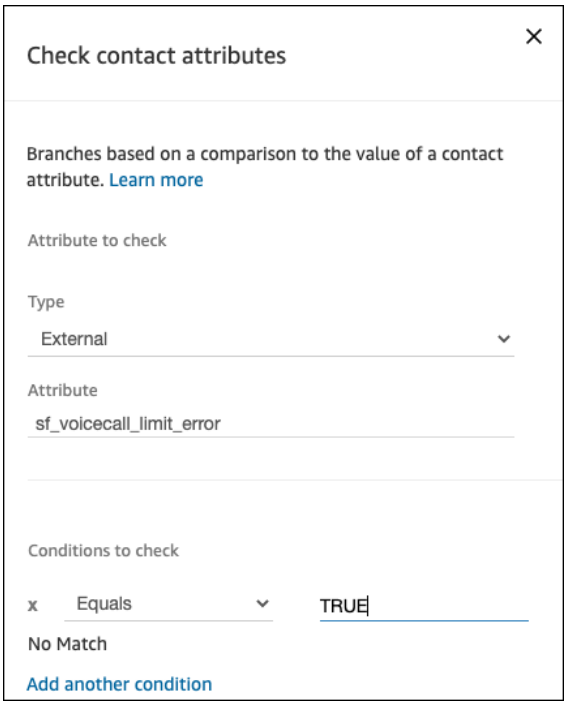
The output of the InvokeTelephonyIntegrationApi Lambda function is stored in a contact attribute. If the contact attribute's value is set to 429, it means that an error occurred. Go to AWS Cloudwatch to view the logs and search for error messages that contain the prefix SCV_LIMITS_ERROR. This prefix indicates a VoiceCall limit error.

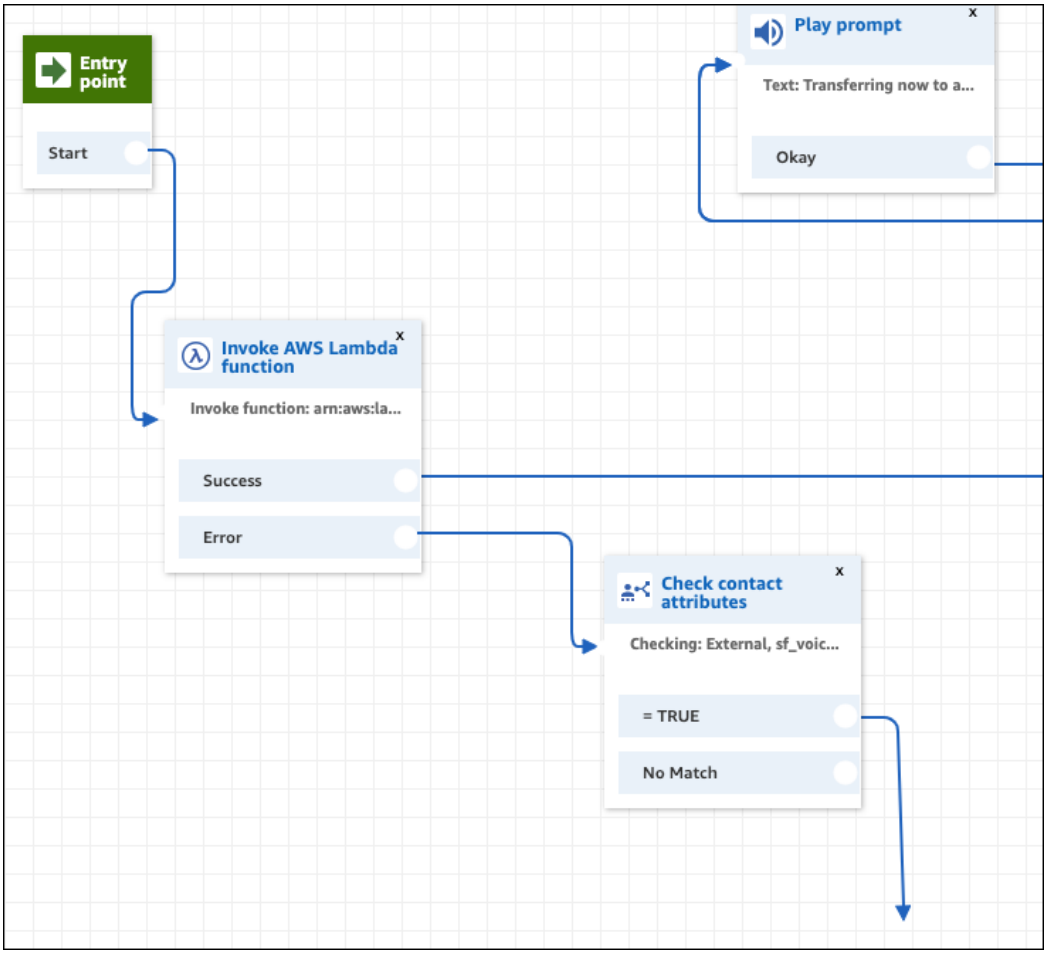
You can address these errors in two ways. One way creates a fallback mechanism that times out after a certain number of attempts. The other way writes a value to a contact attribute so you can see when errors occur.

OPTION 1: Use a fallback mechanism that times out

This approach creates a fallback mechanism that you can add to the contact flow. In this approach, the system attempts to create the VoiceCall record for a certain number of times and then times out.

1. From the inbound contact flow, set the contact attribute, name it, and set its value to `FALSE`. For this use case, we name it "sf_voicecall_limit_error". To learn more about contact attributes, see the Amazon Connect Administrator Guide on setting contact attributes.
2. Modify the InvokeTelephonyIntegrationApi Lambda function to catch any 429 API response codes returned from the CreateVoiceCall API. This 429 error code is returned by the API when it hits any limit errors. Once you catch the error, update the contact attribute you previously created by setting the value to `TRUE`.
3. From within the contact flow, the Lambda errors out, so as part of the error branch of the Lambda block, place a check contact attribute block and verify if the attribute is `TRUE`. If so, you can put a loop block in the contact flow and play a prompt before trying to create a VoiceCall record. Track of the number of retries using another contact attribute. You can put a limit on the number of times the system tries to create the VoiceCall record. To learn more about contact attributes, see the Amazon Connect Administrator Guide on checking contact attributes.





This approach uses a fallback mechanism when the per second limit is hit.

OPTION 2: See the number of failed calls

This approach shows you how many calls failed because of the limit error.

1. Write the error to a contact attribute such as Voice_Call_Creation_Status.
2. To identify how many calls failed as a result of the limit error, do a contact search using the contact attribute. Use a filter to search based on contract attributes. Update the Lambda function to check a condition on the contact attribute.