Integration Guide

AudioCodes Intuitive Human Communications for Chatbot Services

Voice.AI Gateway

Version 2.2

Notice

Information contained in this document is believed to be accurate and reliable at the time of printing. However, due to ongoing product improvements and revisions, AudioCodes cannot guarantee accuracy of printed material after the Date Published nor can it accept responsibility for errors or omissions. Updates to this document can be downloaded from https://www.audiocodes.com/library/technical-documents.

This document is subject to change without notice.

Date Published: November-16-2020

WEEE EU Directive

Pursuant to the WEEE EU Directive, electronic and electrical waste must not be disposed of with unsorted waste. Please contact your local recycling authority for disposal of this product.

Customer Support

Customer technical support and services are provided by AudioCodes or by an authorized AudioCodes Service Partner. For more information on how to buy technical support for AudioCodes products and for contact information, please visit our website at https://www.audiocodes.com/services-support/maintenance-and-support.

Documentation Feedback

AudioCodes continually strives to produce high quality documentation. If you have any comments (suggestions or errors) regarding this document, please fill out the Documentation Feedback form on our website at https://online.audiocodes.com/documentation-feedback.

Stay in the Loop with AudioCodes

Notes and Warnings

OPEN SOURCE SOFTWARE. Portions of the software may be open source software and may be governed by and distributed under open source licenses, such as the terms of the GNU General Public License (GPL), the terms of the Lesser General Public License (LGPL), BSD and LDAP, which terms are located at https://www.audiocodes.com/services-support/open-source/ and all are incorporated herein by reference. If any open source software is provided in object code, and its accompanying license requires that it be provided in source code as well, Buyer may receive such source code by contacting AudioCodes, by following the instructions available on AudioCodes website.

Related Documentation

Document Name
Voice.AI Gateway API Reference Guide
Voice.AI Gateway Product Description
Voice.AI Gateway Security Guidelines
Voice.AI Gateway with One-Click Dialogflow Integration Guide
AudioCodes Phone Number Connector

Document Revision Record

LTRT	Description
30920	Initial document release.
30921	Parameters added- sttContextId, , sttContextPhrases, sttContextBoost.
30922	"Amazon Lex" and "Google Dialogflow" terms added; AudioCodes API syntax example for initial sent message; hangupReason updated (CDR); handover changed to transfer; event type (e.g., string) added; sttEndpointID (description updated)' miscellaneous.
30923	Updated to Ver. 1.4. VOICE_AI_WELCOME event replaced by WELCOME.
30924	Dialogflow text length limitation; typo (Product Notice replaced by Product Description).
30926	Updated to Ver. 1.6; Nuance add for STT/TTS; transferSipHeaders (typo); transferReferredByURL (added); sttContextId (updated)

LTRT	Description
30927	Bot-controlled parameters added.
30928	Updated to Ver. 1.8. startRecognition and stopRecognition (added to activities); participant (added to initial message); participant, from, and participantUriUser (added to text message); sendMetaData (added to activities)
30929	No User Input Event section added; DTMF Event corrected re "value".
30930	Updated to Ver. 2.0. sending HTTP requests added to sendMetaData; text message for Dialogflow updated in Syntax of Text Message Sent by Voice.AI Gateway; Webhook examples added; DTMF handling parameters added (azureDTMFSequenceHandling, azureDTMFSequenceHandlingMS).
30931	azureDTMFSequenceHandling and azureDTMFSequenceHandlingMS removed.
30932	Updated to Ver. 2.2. New parameters (dtmfCollect, dtmfCollectInterDigitTimeoutMS, dtmfCollectMaxDigits, dtmfCollectSubmitDigit, sttSpeechContexts, callRecordingId, callRecordingDestUsername); new section 'startCallRecording and stopCallRecording'; Google Dialogflow CX added.

Table of Contents

1 Introduction

AudioCodes Voice.AI Gateway enhances chatbot functionality by allowing human communication with chatbots through voice (voicebot), offering an audio-centric user experience. Integrating the Voice.AI Gateway into your chatbot environment provides you with a single-vendor solution, assisting you in migrating your text-based chatbot experience into a voice-based chatbot.

- Prior to reading this document, it is recommended that you read the Voice.AI Gateway Product Description to familiarize yourself with AudioCodes Voice.AI Gateway architecture and solution.
- Most of the information provided in this document is relevant to all bot frameworks. Where a specific bot framework uses different syntax, a note will indicate this.

Purpose

This guide provides the following:

- Information that you need to supply AudioCodes for connecting the Voice.AI Gateway to the third-party cognitive services used in your chatbot environment - bot framework(s), speech-to-text (STT) engine(s), and text-to-speech (TTS) engine(s).
- Description of the messages sent by the Voice.AI Gateway to the bot, and messages sent by the bot to the Voice.AI Gateway to achieve the desired functionality. These descriptions allow the bot developer to adapt the bot's behavior to the voice and telephony engagement channels.

Targeted Audience

This guide is intended for IT Administrators and Bot Developers who want to integrate AudioCodes Voice.AI Gateway into their bot solution.

2 Required Information

This section lists the information that you need to supply AudioCodes for integrating and connecting the Voice.AI Gateway to the cognitive services of your chatbot environment. This includes information of the bot framework, Speech-to-Text (STT) provider, and Text-to-Speech (TTS) provider used in your environment.

Required Information of Bot Framework Provider

To connect the Voice.AI Gateway to bot frameworks, you need to provide AudioCodes with the bot framework provider's details, as listed in the following table.

Table 2-1: Required Information per Bot Framework

Bot Framework	Required Information
Microsoft Azure	To connect to Microsoft Azure Bot Framework, you need to provide AudioCodes with the bot's secret key. To obtain this key, refer to Azure's documentation at https://docs.microsoft.com/en-us/azure/bot-service/bot-service-channel-connect-directline. Note: Microsoft Azure Bot Framework Direct Line Version 3.0 must be used.
AWS	To connect to Amazon Lex, you need to provide AudioCodes with the following: ■ AWS account keys: ✔ Access key ✔ Secret access key To obtain these keys, refer to the AWS documentation at https://docs.aws.amazon.com/general/latest/gr/managing-aws-access-keys.html. **Note:** The same keys are used for all Amazon services (STT, TTS and bot framework). ■ Name of the specific bot ■ AWS Region (e.g., "us-west-2")
Google Dialogflow ES	To connect to Google Dialogflow, you need to provide AudioCodes with the following: ■ Private key of the Google service account. For information on how to create the account key, refer to Google's documentation at

Bot Framework	Required Information
	https://cloud.google.com/iam/docs/creating-managing-service-account-keys. From the JSON object representing the key, you need to extract the private key (including the "-----BEGIN PRIVATE KEY-----" prefix) and the service account email. ■ Client email ■ Project ID (of the bot)
Google Dialogflow CX	Dialogflow CX is currently only supported through AudioCodes Phone Number Connector (PNC). For more information, visit https://pnc.audiocodes.io/.
AudioCodes Bot API	To create the channel between the Voice.AI Gateway's Cognitive Service component and the bot provider, refer to the document Voice.AI Gateway API Reference Guide.

Required Information of STT Provider

To connect the Voice.AI Gateway to third-party, speech-to-text (STT) engines, you need to provide AudioCodes with the STT provider's details, as listed in the following table.

Table 2-2: Required Information per Supported STT Provider

STT Provider	Required Information from STT Provider	
	Connectivity	**Language Definition**
Microsoft Azure Speech Services	To connect to Azure's Speech Service, you need to provide AudioCodes with your subscription key for the service. To obtain the key, see Azure's documentation at https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/get-started. **Note:** The key is only valid for a specific region.	To connect to Azure Speech Services, you need to provide AudioCodes with the following: ■ Relevant value in the 'Locale' column in Azure's Text-to-Speech table (see below). For example, for Italian (Italy), the 'Locale' column value is "it-IT". For languages supported by Azure's Speech Services, see the Speech-to-text table in Azure's documentation at https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/language-support.

STT Provider	Required Information from STT Provider	
		The Voice.AI Gateway can also use Azure's Custom Speech service. For more information, see Azure's documentation at https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/how-to-custom-speech-deploy-model . If you do use this service, you need to provide AudioCodes with the custom endpoint details.
Google Cloud Speech-to-Text	To connect to Google Cloud Speech-to-Text service, see Required Information of Bot Framework Provider on page 2 for required information.	To connect to Google Cloud Speech-to-Text, you need to provide AudioCodes with the following: ■ Relevant value in the 'languageCode' column in Google's Cloud Speech-to-Text table (see below). For example, for English (South Africa), the 'Language code' column value is "en-ZA". For languages supported by Google Cloud Speech-to-Text, see Google's documentation at https://cloud.google.com/speech-to-text/docs/languages.
Yandex	Contact AudioCodes for more information.	Contact AudioCodes for more information.
Nuance	Contact AudioCodes for more information.	Contact AudioCodes for more information.

Required Information of TTS Provider

To connect the Voice.AI Gateway to third-party, text-to-speech (TTS) engines, you need to provide AudioCodes with the TTS provider's details, as listed in the following table.

Table 2-3: Required Information per Supported TTS Provider

TTS Provider	Required Information from TTS Provider	
	Connectivity	**Language Definition**
Microsoft Azure Speech Services	To connect to Azure's Speech Service, you need to provide AudioCodes with your subscription key for the service. To obtain the key, see Azure's documentation at https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/get-started. **Note:** The key is valid only for a specific region.	To connect to Azure Speech Services, you need to provide AudioCodes with the following: ■ Relevant value in the 'Locale' column in Azure's Text-to-Speech table (see below link). ■ Relevant value in the 'Short voice name' column in Azure's Text-to-Speech table (see below link). For example, for Italian (Italy), the 'Locale' column value is "it-IT" and the 'Short voice name' column value is "it-IT-ElsaNeural". For languages supported by Azure's Speech Services, see the Text-to-Speech table in Azure's documentation at https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/language-support.
Google Cloud Text-to-Speech	To connect to Google Cloud Text-to- Speech service, see Required Information of Bot Framework Provider on page 2 for required information.	To connect to Google Cloud Text-to-Speech, you need to provide AudioCodes with the following: ■ Relevant value in the 'Language code' column in Google's table (see below link). ■ Relevant value in the 'Voice name' column in Google's table (see below link). For example, for English (US), the 'Language code' column value is "en-US" and the 'Voice name' column value is "en-US-Wavenet-A". For languages supported by Google Cloud Text-to-Speech, see Google's documentation at https://cloud.google.com/text-to-speech/docs/voices.
AWS Amazon Polly	To connect to Amazon Polly Text-to-Speech service, see Required Information of Bot	To connect to Amazon Polly TTS service, you need to provide AudioCodes with the following: ■ Relevant value in the 'Language' column in Amazon Polly TTS table (see below link).

TTS Provider	Required Information from TTS Provider	
	Framework Provider on page 2 for required information.	■ Relevant value in the 'Name/ID' column in Amazon Polly TTS table (see below link). For example, for English (US), the 'Language' column value is "English, US (en-US)" and the 'Name/ID' column is "Matthew". For languages supported by Amazon Polly TTS service, see the table in https://docs.aws.amazon.com/polly/latest/dg/voicelist.html.
Yandex	Contact AudioCodes for more information.	Contact AudioCodes for more information.
Almagu	Contact AudioCodes for more information.	Contact AudioCodes for more information.
Nuance	Contact AudioCodes for more information.	Contact AudioCodes for more information.

3 Messages Sent by Voice.AI Gateway

This section describes the messages that are sent by the Voice.AI Gateway.

Initial Message

When the conversation starts, a message is sent with the details of the call. These details include (when available) the following:

Table 3-1: Description of Initial Message Sent by Voice.AI Gateway

Property	Type	Description
`callee`	String	Dialed phone number. This is typically obtained from the SIP To header.
`calleeHost`	String	Host part of the destination of the call. This is typically obtained from the SIP To header.
`caller`	String	Caller's phone number. This is typically obtained from the SIP From header.
`callerHost`	String	Host part of the source of the call. This is typically obtained from the SIP From header.
`callerDisplayName`	String	Caller's display name. This is typically obtained from the SIP From header.
`<Additional attributes>`	-	Defines additional attributes such as values from various SIP headers. These can be added by customization. The Voice.AI Gateway can be configured to extract values from the SIP INVITE message and then send them as additional attributes in the initial message to the bot.
`participants`	Array of Objects	Participants of the conversation when the Voice.AI Gateway is used with the SBC's SIPRec feature (e.g., for the Agent Assist solution). This parameter includes the following sub-parameters: ■ `participant`: (String) Role of the participant, which can be one of the following values: ✔ `caller` ✔ `callee` ✔ user defined

Property	Type	Description

		The value is obtained from the 'ac:role' element in the SIPRec XML body. The values should be set in the SIPRec XML using the SBC's Message Manipulation functionality, under the <participant> element, as shown in the following example:
		`<participant id="+123456789" session="0000-0000-0000-0000-b44497aaf9597f7f">`
		`<nameID aor="+123456789@example.com"></nameID>`
		`<ac:role>caller</ac:role>`
		`</participant>`
		The values must be unique.
		For agent-assist calls, each `message` activity that is sent to the bot includes a `participant` parameter.
		■ `uriUser`: (String) User-part of the URI of the participant. The value is obtained from the user-part of the 'aor' property of the 'nameID' element in the SIPRec XML body.
		■ `uriHost`: (String) Host-part of the URI of the participant. The value is obtained from the host-part of the 'aor' property of the 'nameID' element in the SIPRec XML body.
		■ `displayName`: (String) Display name of the participant. The value is obtained from the 'name' sub-element of the 'nameID' element in the SIPRec XML body.

outboundTarget	String	ontains the "target" of the dial-out request. **Note:** This is applicable only to the Outbound Calling feature.
metadata	String	Metadata from the Dialer app that is sent through the Voice.AI Gateway to the bot. For example: `"metadata": {`

Property	Type	Description
		"participantName": "Alice" } Note: This is applicable only to the Outbound Calling feature.

The syntax of the initial message depends on the specific bot framework:

Table 3-2: Syntax of Initial Message Sent by Voice.AI Gateway

Bot Framework	Message Syntax

AudioCodes Bot API	The message is sent as a `start` event, with the details inside the `parameters` property.
	Example:
	`{`
	`  "type": "event",`
	`  "name": "start",`
	`  "parameters": {`
	`    "callee": "12345678",`
	`    "calleeHost": "10.20.30.40",`
	`    "caller": "12345678",`
	`    "callerHost": "10.20.30.40"`
	`  }`
	`}`

Microsoft Azure	The message is sent as a `channel` event, with the details inside the `channelData` property.
	Example:
	`{`
	`  "type": "event",`
	`  "name": "channel",`
	`  "value": "telephony",`
	`  "channelData": {`
	`    "callee": "12345678",`
	`    "calleeHost": "10.20.30.40",`
	`    "caller": "12345678",`
	`    "callerHost": "10.20.30.40"`
	`  },`

Bot Framework	Message Syntax

	`"from": {`
	`"id": "12345678"`
	`},`
	`"locale": "en-US"`
	`}`

Google Dialogflow	The message is sent as a `WELCOME` event, with the details as `event` parameters.
	Example:
	`{`
	`"queryInput": {`
	`"event": {`
	`"languageCode": "en-US",`
	`"name": "WELCOME",`
	`"parameters": {`
	`"callee": "12345678",`
	`"calleeHost": "10.20.30.40",`
	`"caller": "12345678",`
	`"callerHost": "10.20.30.40"`
	`}`
	`}`
	`}`
	`}`
	Note: These parameters can be used when generating the response text, by using a syntax such as this:
	`"#WELCOME.caller"`

End of Conversation Message

The syntax of the end-of-conversation message depends on the specific bot framework:

Table 3-3: Syntax of End-of-Conversation Message Sent by Voice.AI Gateway

Bot Framework	Message Syntax
AudioCodes Bot API	The conversation is terminated according to the AC Bot API documentation.
Microsoft	The conversation is terminated by sending an `endOfConversation`

Bot Framework	Message Syntax

Azure	activity, with an optional `text` property with a textual reason.
	Example:
	`{`
	`"type": "endOfConversation",`
	`"text": "Client Side"`
	`}`

Google Dialogflow	Currently, no indication is sent for the end of conversation.

Text Message

When the speech-to-text engine detects user utterance, it is sent as a message to the bot. The message may contain details gathered by the speech-to-text engine. These details include:

Table 3-4: Description of Text Message Sent by Voice.AI Gateway

Property	Type	Description
`confidence`	Number	Numeric value representing the confidence level of the recognition.
`recognitionOutput`	Object	Raw recognition output of the speech-to-text engine (vendor specific).
`recognitions`	Array of Objects	If Continuous ASR mode is enabled, this array contains the separate recognition outputs.
`participant`	String	Indicates the participant (“role”) on which the speech recognition occurred. **Note:** The parameter is applicable only to Agent Assist calls.
`participantUriUser`	String	URI of the participant. **Note:** The parameter is applicable only to Agent Assist calls.

The syntax of the text message depends on the specific bot framework:

Table 3-5: Syntax of Text Message Sent by Voice.AI Gateway

<table>
<tr><th>Bot Framework</th><th>Message Syntax</th></tr>
<tr><td>AudioCodes Bot API</td><td>The message is sent as a <code>message</code> activity. Additional details are sent in the <code>parameters</code> property.
Example:
<pre>{
    "type": "message",
    "text": "Hi.",
    "parameters": {
       "confidence":0.6599681,
    }
}</pre></td></tr>
<tr><td>Microsoft Azure</td><td>The message is sent as a <code>message</code> activity. Additional details are sent in the <code>channelData</code> property.
Example:
<pre>{
    "type": "message",
    "text": "Hi.",
    "channelData": {
       "confidence":0.6599681,
  }
}</pre></td></tr>
<tr><td>Google Dialogflow</td><td>The message is sent as text input. Additional details are sent as the request payload, which can be accessed from a webhook, using the <code>originalDetectIntentRequest.payload</code> field of the request.
Example payload:
<pre>{
   "parameters": {
      "confidence": 0.6599681
  }
}</pre>In addition, for agent-assist calls, a context with the name "vaig-participant-<participant>" (e.g., "vaig-participant-caller") is set for each text input message. These contexts can be used as input contexts for filtering intents of specific participants.
For examples of using webhooks, see Dialogflow Webhook Examples on page 19.
Note:</td></tr>
</table>

Bot Framework	Message Syntax
	■ Dialogflow supports a maximum text input length of 256 characters. Therefore, if the input received from the speech-to-text engine is longer than 256 characters, the Voice.AI Gateway truncates the message before sending it to Dialogflow. ■ When activated through AudioCodes Phone Number Connector (PNC), the additional details are currently not sent to the bot.

DTMF Event

The syntax for DTMF tone signals (i.e., keys pressed on phone keypad by user) depends on the specific bot framework.

Table 3-6: Syntax of DTMF Sent by Voice.AI Gateway

Bot Framework	Message Syntax

AudioCodes Bot API / Microsoft Azure	This message is sent as a `DTMF` event with the digits as the value of the event.
	Example:
	`{`
	`  "type": "event",`
	`  "name": "DTMF",`
	`  "value": "3"`
	`}`

Google Dialogflow ES	This message is sent as a `DTMF` event with the digits as the event parameters.
	Example:
	`{`
	`  "queryInput": {`
	`    "event": {`
	`      "languageCode": "en-US",`
	`      "name": "DTMF",`
	`      "parameters": {`
	`        "value": "3"`
	`      }`
	`    }`
	`  }`
	`}`
	Note: The digits can be used when generating the response

Bot Framework	Message Syntax
	text, by using a syntax such as this: `"#DTMF.value"`
Google Dialogflow CX	DTMF events are sent according to the Dialogflow CX specification.

No User Input Event

The Voice.AI Connector can send an event message to the bot if there is no user input (for the duration configured by the `userNoInputTimeoutMS` parameter), indicating how many times the timeout expired ('value' field). The message is sent only if the `userNoInputSendEvent` is configured to `true`.

Table 3-7: Syntax of No User Input Event Sent by Voice.AI Gateway

Bot Framework	Message Syntax

AudioCodes Bot API / Microsoft Azure	This message is sent as a `noUserInput` event with the number of times that the timeout expired as the value of the event.
	Example:
	`{`
	`  "type": "event",`
	`  "name": "noUserInput",`
	`  "value": "1`
	`}`

Google Dialogflow	This message is sent as a `noUserInput` event with the number of times that the timeout expired as the value of the event.
	Example:
	`{`
	`  "queryInput": {`
	`    "event": {`
	`      "languageCode": "en-US",`
	`      "name": "noUserInput",`
	`      "parameters": {`
	`        "value": "1"`
	`      }`
	`    }`
	`  }`
	`}`

4 Messages Sent by Bot

When the Voice.AI Gateway handles messages from the bot, it treats them as activities.

The syntax for sending the activities in the different bot frameworks is described in Section Bot Framework Specific Details below.

Activities sent by the bot contain actions to be performed and parameters. The parameters can affect the current action or change the behavior of the whole conversation. A list of the configurable parameters are described in Section Parameters Controlled Also by Bot on page 30.

The Voice.AI Gateway handles activities synchronously and therefore, an activity is not executed before the previous one has finished. For example, when the Voice.AI Gateway receives two activities—to play text to the user and to hang up the call—the `hangup` activity is only executed after it has finished playing the text.

Bot Framework Specific Details

This section provides details specific to bot frameworks.

AudioCodes Bot API

For AudioCodes Bot API, the activities can be sent as is, with the addition of the attributes `id` and `timestamp`, as defined in the AudioCodes API Reference Guide.

Microsoft Azure

For Azure bots, the `sessionParams` and `activityParams` properties should be placed inside the `channelData` property.

Example:

```
{
  "type": "event",
  "name": "transfer",
  "channelData": {
    "activityParams": {
      "handoverReason": "userRequest",
      "transferTarget": "tel:123456789"
    }
  }
}
```

Google Dialogflow

The messages sent by a Dialogflow agent are based on intent's responses (on Dialogflow CX, responses are called *fulfillment*), which have a different syntax than that of the activities described below. To overcome this, the Voice.AI Gateway uses two approaches, both of which can be used by the bot.

> ⚠ For Dialogflow ES, responses can be attached to platforms (e.g., Google Assistant). The Voice.AI Gateway only uses the responses of the DEFAULT platform.

The first approach is using native Dialogflow responses. The Voice.AI Gateway automatically translates supported responses into activities. For example, a Text Response is translated into a `message` activity (for playing the text to the user). As another example, an intent that is marked to end the conversation is translated into a `hangup` activity.

The second approach is for the bot to send activities according to Voice.AI Gateway syntax. For doing this, Custom Payload response with an `activities` property should be added (see below). The value of the `activities` property should contain an array of activities to be executed by the Voice.AI Gateway.

The following example shows a Custom Payload for performing the `transfer` activity:

```
{
  "activities": [
    {
      "type": "event",
      "name": "transfer",
      "activityParams": {
        "transferTarget": "tel:+123456789"
      }
    }
  ]
}
```

Setting Activity Parameters for Native Dialogflow Responses

If you wish to set parameters for a native Dialogflow response, you can do so by adding an `activityParams` property to a Custom Payload response.

> ⚠ The activities must reside within the `activities` array.

For example, if you have a Text Response and you want to disable TTS caching of that response, you can add the following Custom Payload:

```
{
 "activityParams": {
  "disableTtsCache": true
 }
}
```

Note that the position of the `activityParams` property determines its scope. If placed in the root of the Custom Payload, it affects the native Dialogflow response. If placed inside an activity (within the `activities` array), it affects the specific activity.

Custom Payload Responses

As described previously, Custom Payload responses are used for both sending activities by the bot and for setting parameters for native Dialogflow responses.

Documentation of Custom Payload can be found in the following links:

- For Dialogflow ES, go to https://cloud.google.com/dialogflow/es/docs/intents-rich-messages#custom
- For Dialogflow CX, go to https://cloud.google.com/dialogflow/cx/docs/concept/fulfillment

The Custom Payload should contain a JSON object with one or more of the following properties:

Table 4-1: Google Dialogflow Custom Payload Properties

Property	Description
`activityParams`	Parameters applied to the native Dialogflow response.
`sessionParams`	Parameters applied to the whole session.
`activities`	Array of activities to be executed after the native Dialogflow responses.

Custom Payload configuration (example) through Dialogflow user interface:

Custom payload

```
{
  "activities": [
    {
      "type": "event",
      "name": "transfer",
      "activityParams": {
        "transferTarget": "tel:+123456789"
      }
    }
  ]
}
```

},

Dialogflow Webhook Examples

This section provides examples of using Dialogflow webhooks for bots.

- This is an example of a webhook that handles the `WELCOME` event and performs `startRecognition` to all participants:

```
function welcome(agent) {
  const activities = request.body.queryResult.outputContexts.find(
    (c) => c.name.endsWith('welcome')).parameters.participants.map(
      (p) => ({
        "activityParams": {
          "targetParticipant": p.participant
        },
        "name": "startRecognition",
        "type": "event"
      }));

  const payload = new Payload(
    'PLATFORM_UNSPECIFIED',
    { activities },
    { rawPayload: true, sendAsMessage: true }
  );

  agent.add(payload);
}
```

- This is an example of handling text messages and performing `sendMetadata`:

```
function fallback(agent) {
  const participant =
    request.body.originalDetectIntentRequest.payload.parameters.participant;
  agent.add(new Payload('PLATFORM_UNSPECIFIED',
    {
      "activities": [
        {
          "name": "sendMetaData",
          "type": "event",
          "value": {
          "participant": participant,
          "text": request.body.queryResult.queryText
            }
        }
      ]
    },
```

```
  { rawPayload: true, sendAsMessage: true }));
}
```

Basic Activity Syntax

Each activity is a JSON object that has the following properties:

Table 5-1: Properties of JSON Object Activities

Property	Type	Description
`type`	String	Either `message` or `event`.
`name`	String	Name of event for the `event` activity. For supported events, see event Activities on the next page.
`text`	String	Text to be played for the `message` activity.
`activityParams`	Params object	Set of parameters that affect the current activity.
`sessionParams`	Params object	Set of parameters that affect the remaining duration of the conversation.

The Params object is comprised of key-value pairs, were the key is the parameter name and the value is the desired value for the parameter. For a list of the supported parameters, see Parameters Controlled Also by Bot on page 30.

message Activity

The most common activity is the `message` activity, which indicates to the Voice.AI Gateway to play the given text to the user.

Example:

```
{
  "type": "message",
  "text": "Hi, how may I assist you?"
}
```

A `message` activity can also contain parameters that affect its handling. For example, to disable caching of the text-to-speech generated voice for the current activity, the following activity can be sent:

```
{
  "type": "message",
```

```
  "text": "I have something sensitive to tell you.",
  "activityParams": {
   "disableTtsCache": true
  }
 }
```

The `text` field can contain Speech Synthesis Markup Language (SSML). The SSML can be one of the following:

- A full SSML document, for example:

```
<speak>
 This is <say-as interpret-as="characters">SSML</say-as>.
</speak>
```

- Text with SSML tags, for example:

```
This is <say-as interpret-as="characters">SSML</say-as>.
```

- The SSML is parsed by the text-to-speech engine. Refer to their documentation for a list of supported features.
- When using SSML, all invalid XML characters, for example, the ampersand (&), must be properly escaped.
- The supported SSML elements depend on the text-to-speech provider:
  - ✔ **Google:** https://cloud.google.com/text-to-speech/docs/ssml
  - ✔ **Azure:** https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/speech-synthesis-markup#supported-ssml-elements
  - ✔ **AWS:** https://docs.aws.amazon.com/polly/latest/dg/supportedtags.html

event Activities

This section lists the supported events. Each event is shown with a list of associated parameters. These parameters can be set either in the configuration of the bot or by sending them as part of the `activityParams` (to be used once) or as part of the `sessionParams` (to be used for the remaining duration of the conversation).

The list only includes parameters that are specific to that event, but other parameters can also be updated by the event. For example, the `language` parameter can be updated by `playUrl`, by adding it to the `activityParams` or `sessionParams` properties.

hangup

The `hangup` event disconnects the conversation.

> ⚠ For Dialogflow CX, the conversation can also be disconnected by adding a transition to "End Session" or "End Flow". For more information, visit https://cloud.google.com/dialogflow/cx/docs/concept/flow.

The following table lists the parameters associated with this event.

Table 5-2: Parameters for hangup Event

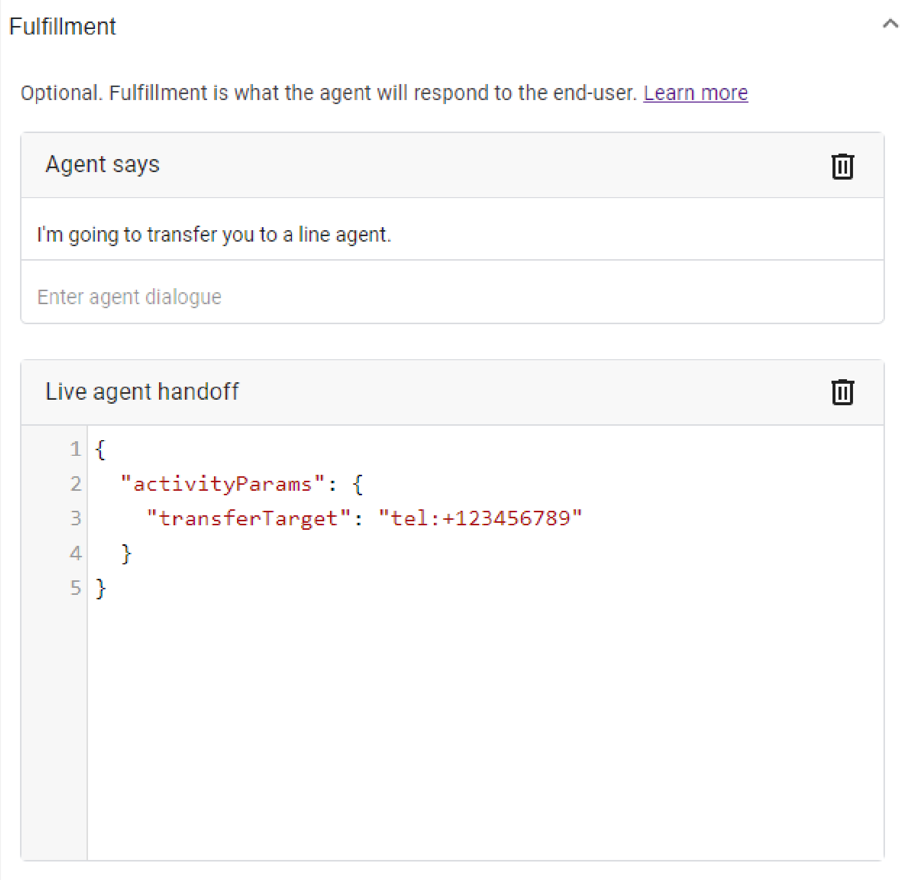
Parameter	Type	Description
`hangupReason`	String	Conveys a textual reason for hanging up. This reason appears in the CDR of the call.

Example:

```
{
 "type": "event",
 "name": "hangup",
 "activityParams": {
  "hangupReason": "conversationCompleted"
 }
}
```

transfer

The `transfer` event transfers the call to a human agent or to another bot. The `handover` event is a synonym for the `transfer` event.

> For Dialogflow CX, call transfer can also be done by adding the "Live agent handoff" fulfillment, as shown in the following example of configuration through the Dialogflow CX user interface:
>
> 
>

The following table lists the parameters associated with this event.

Table 5-3: Parameters for transfer Event

Parameter	Type	Description
`transferTarget`	String	URI to where the call must be transferred call to. Typically, the URI is a "tel" or "sip" URI.
`handoverReason`	String	Conveys a textual reason for the transfer.
`transferSipHeaders`	Array of Objects	Array of objects listing SIP headers that should be sent to the transferee. Each object comprises a `name` and a `value` attribute. For more information, see Adding SIP Headers on Call Transfer on the next page.
`transferReferredByURL`	String	Defines the party (URL) who initiated the call referral. If this parameter exists, the SBC

Parameter	Type	Description
		adds a SIP Referred-By header to the outgoing INVITE or REFER message (according to the 'Remote REFER Mode' parameter). If the SBC handles locally (termination), the SBC adds it to a new outgoing INVITE. If not handled locally (regular), the SBC adds it to the forwarded REFER message.

Example:

```
{
 "type": "event",
 "name": "transfer",
 "activityParams": {
  "handoverReason": "userRequest",
  "transferTarget": "tel:123456789"
 "transferReferredByURL": "sip:456@ac.com",
 }
}
```

The above example can be configured through the Dialogflow user interface, as described in Google Dialogflow on page 16.

Adding SIP Headers on Call Transfer

When the bot performs a call transfer using the `transfer` event, it can add data to be sent as SIP headers in the generated SIP message (REFER or INVITE). This is done by the `transferSipHeaders` parameter. This parameter contains an array of JSON objects with the following attributes:

Table 5-4: Attributes of transferSipHeaders Parameter

Attribute	Type	Description
`name`	String	Name of the SIP header.
`value`	String	Value of the SIP header.

For example, the following `transfer` event can be used to add the header "X-My-Header" with the value "my_value":

```
{
 "type": "event",
```

```
  "name": "transfer",
  "activityParams": {
   "transferTarget": "sip:john@host.com",
   "transferSipHeaders": [
    {
     "name": "X-My-Header",
     "value": "my_value"
    }
   ]
  }
}
```

If the Voice.AI Gateway is configured to handle transfer by sending a SIP INVITE message, it will contain the header, for example:

```
X-My-Header: my_value
```

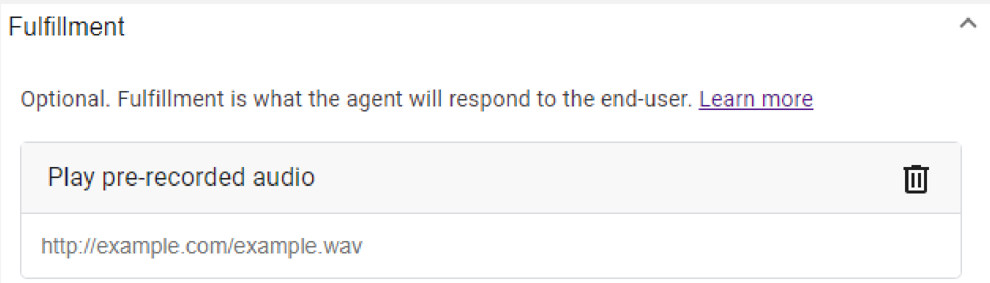
If the Voice.AI Gateway is configured to handle transfer by sending a SIP REFER message, it will contain the value in the URI of the Refer-To header, for example:

```
Refer-To: <sip:john@host.com?X-My-Header=my_value>
```

playUrl

The `playURL` event plays audio to the user from a given URL.

- The format of the file must match the format specified by the `playUrlMediaFormat` parameter; otherwise, the audio will be played corruptly.
- For Dialogflow CX, playing an audio from a URL can also be done by adding the "Play pre-recorded audio" fulfillment, as shown in the following example of configuration through the Dialogflow CX user interface:



The following table lists the parameters associated with this event.

Table 5-5: Parameters for playURL Event

Parameter	Type	Description
`playUrlUrl`	String	URL of where the audio file is located.
`playUrlCaching`	Boolean	Enables caching of the audio: ▪ `true`: Enables caching ▪ `false`: (Default) Disables caching
`playUrlMediaFormat`	String	Defines the format of the audio: ▪ `wav/lpcm16` (default) ▪ `raw/lpcm16`
`playUrlAltText`	String	Defines the text to display in the transcript page of the user interface while the audio is played.

Example:

```
{
 "type": "event",
 "name": "playUrl",
 "activityParams": {
  "playUrlUrl": "https://example.com/my-file.wav",
  "playUrlMediaFormat": "wav/lpcm16"
 }
}
```

config

The `config` event updates the session parameters, regardless of specific activity.

There are no parameters that are associated with this event.

The following is an example of the `config` event, enabling the Barge-In feature:

```
{
 "type": "event",
 "name": "config",
 "sessionParams": {
  "bargeIn": true
 }
}
```

startRecognition and stopRecognition

The `startRecognition` and `stopRecognition` activities are used for Agent Assist calls. The STT engine only starts when a `startRecognition` activity is received from the bot and stops when a `stopRecognition` activity is received from the bot.

The following table lists the parameter associated with this event.

Table 5-6: Parameter for startRecognition and stopRecognition Events

Parameter	Type	Description
`targetParticipant`	String	Defines the participant for which to start or stop speech recognition.

Example:

```
{
 "type": "event",
 "name": "startRecognition",
 "activityParams": {
  "targetParticipant": "caller"
 }
}
```

startCallRecording and stopCallRecording

The `startCallRecording` and `stopCallRecording` activities are used by the bot to start and stop voice recording of the conversation (at any stage of the call). Recording is done by the SBC (SRC) using the SIPRec protocol and functioning as the Session Recording Client (SRC). The session recording server (SRS) can be AudioCodes SmartTAP recording solution or any third-party SRS. The bot sends the message to the Voice.AI Gateway over REST, and the Voice.AI Gateway in turn sends the message to the SBC over a WebSocket tunnel.

The following table lists the parameters associated with this event.

Table 5-7: Parameters for startCallRecording and stopCallRecording Events

Parameter	Type	Description
`callRecordingServer`	String	Defines the SRS as an IP Group name (as configured on the SBC) to record the call. **Note:** Due to security considerations, the parameter is not sent by the bot, but is supported as a mandatory activity parameter in the Voice.AI Gateway configuration.

Parameter	Type	Description
`callRecordingId`	String	Defines the recording ID session, which is forwarded by the SBC to the SRS in the XML body (attribute "ac:call-recording-id") of the SIP INVITE request for SIPRec. The Administrator can later retrieve the bot's recordings from the SRS, by using this ID. Below is an example of an XML body with the ID shown in the XML attribute "ac:call-recording-id": <?xml version="1.0" encoding="UTF-8"?> <recording xmlns="urn:ietf:params:xml:ns:recording" xmlns:ac="http://AudioCodes"> <datamode>complete</datamode> <group id="00000000-0000-0072-57a7-310000000072"> <associate-time>2018-01-01T02:22:29</associate-time> </group> <session id="0000-0000-0000-0000-b44497aaf9597f7f"> <group-ref>00000000-0000-0072-57a7-310000000072</group-ref> <associate-time>2018-01-01T02:22:29</associate-time> <**ac:call-recording-id**>KUYdgtofldi76YpkP09J4J-a:1</ac:call-recording-id> </session> <participant id="+123456789" session="0000-0000-0000-0000-b44497aaf9597f7f"> .... **Note:** - The parameter is applicable only to the `startCallRecording` event.

Parameter	Type	Description
		■ The XML attribute name "ac:call-recording-id" can be changed using the SBC's Message Manipulation feature.
`callRecordingDestUsername`	String	Defines the username that is used in the SIP Request-URI and To headers of the INVITE request for SIPRec. **Note:** The parameter is applicable only to the `startCallRecording` event.

Examples:

- Start recording:

```
{
  "type": "event",
  "name": "startCallRecording",
  "activityParams": {
     "callRecordingId": "KUYdgtofIdi76YpkP09J4J-a:1",
     "callRecordingDestUsername: "AgentAssist7"
  }
}
```

- Stop recording:

```
{
  "type": "event",
  "name": "stopCallRecording",
}
```

- Message sent by Voice.AI Gateway to SBC when `startCallRecording` event is received from bot:

```
{
  "message": "StartCallRecording",
  "callRecordingServer": "SmartTAP5",
  "callRecordingId": "KUYdgtofIdi76YpkP09J4J-a:1",
  "callRecordingDestUsername": "AgentAssist7",
  "sessionID": "08b34396-992a-492e-818e-d4b0df51cb82"
}
```

sendMetaData

The `sendMetaData` event can be used for sending data (using SIP INFO messages) to the peer of the conversation. For example, for Agent Assist calls, the bot can send suggestions to the human agent. The bot passes the data in the "value" parameter, which can contain any valid JSON object. When handling the activity, the Voice.AI Gateway sends a SIP INFO request with a body containing the data as JSON.

Instead of sending SIP INFO messages, the `sendMetaData` event can be used for sending (POST) HTTP requests to an HTTP server (defined as a URL).

Example:

```
{
 "type": "event",
 "name": "sendMetaData",
 "value": {
  "myParamName": "myParamValue"
 }
}
```

Parameters Controlled Also by Bot

These parameters can be configured on the Voice.AI Connector, but they can also be determined and updated by the bot dynamically. The bot takes precedence (i.e., overrides Voice.AI Connector configuration). Parameters that are specific to a single event type are documented in Section event Activities on page 21. As explained in Section Basic Activity Syntax on page 20, these parameters can be included in the `activityParams` or the `sessionParams` of any activity sent by the bot.

Table 5-8: Bots Section Parameter Descriptions (Also Controlled by Bot)

Parameter	Type	Description
`azureSpeechRecognitionMode`	String	Defines the Azure STT recognition mode. ▪ `conversation` (default) ▪ `dictation` ▪ `interactive` **Note:** The parameter is applicable only to the Microsoft Azure STT service.
`bargeIn`	Boolean	Enables the Barge-In feature. ▪ `true`: Enabled, When the bot is playing a response to the user (playback of bot

Parameter	Type	Description
		message), the user can "barge-in" (interrupt) and start speaking. This terminates the bot response, allowing the bot to listen to the new speech input from the user (i.e., Voice.AI Gateway sends detected utterance to the bot). ▪ `false`: (Default) Disabled. The Voice.AI Gateway doesn't expect speech input from the user until the bot has finished playing its response to the user. In other words, the user can't "barge-in" until the bot message response has finished playing.
`bargeInOnDTMF`	Boolean	Enables the Barge-In on DTMF feature. ▪ `true`: (Default) Enabled. When the bot is playing a response to the user (playback of bot message), the user can "barge-in" (interrupt) with a DTMF digit. This terminates the bot response, allowing the bot to listen to and process the digits sent from the user. ▪ `false`: Disabled. The Voice.AI Connector doesn't expect DTMF input from the user until the bot has finished playing its response to the user. In other words, the user can't "barge-in" until the bot message response has finished playing. **Note:** ▪ If you enable this feature (i.e., `bargeInOnDTMF` configured to `true`), you also need to enable the sending of DTMF digits (see the `sendDTMF` parameter). ▪ Currently, this parameter is not supported when speech-to-text is performed by the bot framework.
`bargeInMinWordCount`	Number	Defines the minimum number of words that the user must say for the Voice.AI Gateway to consider it a barge-in. For example, if configured to 4 and the user only says 3 words during the bot's playback response, no barge-in occurs.

Parameter	Type	Description
		The valid range is 1 to 5. The default is 1.
`botFailOnErrors`	Boolean	Defines what happens when the Azure bot error "retry" occurs. ■ `true:` The error is printed to the log and the call is disconnected. ■ `false:` (Default) The error is printed to the log, but the call is not disconnected.
`botNoInputGiveUpTimeoutMS`	Number	Defines the maximum time that the Voice.AI Connector waits for a response from the bot. If no response is received when the timeout expires, the Voice.AI Connector disconnects the call with the SBC. The default is 0 (i.e., feature disabled). If the call is disconnected, the SIP BYE message sent by the SBC to the user indicates this failure, by prefixing the value in the Reason header with "Bot Err:". **Note:** In this scenario (disconnects), you can also configure the Voice.AI Connector to perform specific activities, for example, playing a prompt to the user or transferring the call (see the `generalFailoverActivities` parameter).
`botNoInputTimeoutMS`	Number	Defines the maximum time (in milliseconds) that the Voice.AI Connector waits for input from the bot framework. If no input is received from the bot when this timeout expires, you can configure the Voice.AI Connector to play a textual (see the `botNoInputSpeech` parameter) or an audio (see the botNoInputUrl parameter) prompt to the user. The default is 0 (i.e., feature disabled).
`botNoInputRetries`	Number	Defines the maximum number of allowed timeouts (configured by the `botNoInputTimeoutMS` parameter) for no bot input. If you have configured a prompt to

Parameter	Type	Description
		play (see the `botNoInputSpeech` or `botNoInputUrl` parameter), the prompt is played to the user each time the timeout expires. The default is 0 (i.e., only one timeout – no retries). For more information on the no bot input feature, see the `botNoInputTimeoutMS` parameter. **Note:** If you have configured a prompt to play upon timeout expiry, the timer is triggered only after playing the prompt to the user.

`botNoInputSpeech`	String	Defines the textual prompt to play to the user when no input has been received from the bot framework when the timeout expires (configured by `botNoInputTimeoutMS`).
		The prompt can be configured in plain text or in Speech Synthesis Markup Language (SSML) format:
		■ **Plain-text example:**
		{
		"name": "LondonTube",
		"provider": "my_azure",
		"displayName": "London Tube",
		"botNoInputTimeoutMS": 5000,
		"botNoInputSpeech": "Please wait for bot input"
		}
		■ **SSML example:**
		{
		"name": "LondonTube",
		"provider": "my_azure",
		"displayName": "London Tube",
		"botNoInputTimeoutMS": 5000,
		"botNoInputSpeech": <speak>"This is <say-as interpret-as="characters">SSML</say-

Parameter	Type	Description

		`as>"</speak>`
		`}`
		By default, the parameter is not configured.
		Note:
		■ For more information on the no bot input feature, see the `botNoInputTimeoutMS` parameter.
		■ If you have also configured to play an audio prompt (see the `botNoInputUrl` parameter), the `botNoInputSpeech` takes precedence.
		■ This feature requires a text-to-speech provider. It will not work when the speech is synthesized by the bot framework.
		■ The supported SSML elements depend on the text-to-speech provider:
		✔ **Google:** https://cloud.google.com/text-to-speech/docs/ssml
		✔ **Azure:** https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/speech-synthesis-markup#supported-ssml-elements
		✔ **AWS:** https://docs.aws.amazon.com/polly/latest/dg/supportedtags.html
		■ For more information on using SSML for text in the `message` activity, see message Activity on page 20.

`botNoInputUrl`	String	Defines the URL from where the audio prompt is played to the user when no input has been received from the bot when the timeout expires (configured by `botNoInputTimeoutMS`). By default, the parameter is not configured. For more information on the no bot input feature, see the `botNoInputTimeoutMS`.

Parameter	Type	Description
		Note: If you have also configured to play a textual prompt (see the `botNoInputSpeech` parameter), the `botNoInputSpeech` takes precedence.
`userNoInputTimeoutMS`	Number	Defines the maximum time (in milliseconds) that the Voice.AI Connector waits for input from the user. If no input is received when this timeout expires, you can configure the Voice.AI Connector to play a textual (see the `userNoInputSpeech` parameter) or an audio (see the userNoInputUrl parameter) prompt to ask the user to say something. If there is still no input from the user, you can configure the Voice.AI Connector to prompt the user again. The number of times to prompt is configured by the `userNoInputRetries` parameter. If the `userNoInputSendEvent` parameter is configured to `true` and the timeout expires, the Voice.AI Connector sends an event to the bot, indicating how many times the timer has expired. The default is 0 (i.e., feature disabled). **Note:** - DTMF (any input) is considered as user input (in addition to user speech) if the `sendDTMF` parameter is configured to `true`. - If you have configured a prompt to play when the timeout expires, the timer is triggered only after playing the prompt to the user.
`userNoInputRetries`	Number	Defines the maximum number of allowed timeouts (configured by the `userNoInputTimeoutMS` parameter) for no user input. If you have configured a prompt to play (see the `userNoInputSpeech` or userNoInputUrl parameter), the prompt is payed each time the timeout expires. The default is 0 (i.e., only one timeout). For more information on the no user input

Parameter	Type	Description
		feature, see the `userNoInputTimeoutMS` parameter. **Note:** If you have configured a prompt to play upon timeout expiry, the timer is triggered only after playing the prompt to the user.
`userNoInputSendEvent`	Boolean	Enables the Voice.AI Connector to send an event message to the bot if there is no user input for the duration configured by the `userNoInputTimeoutMS` parameter, indicating how many times the timer has expired ('value' field): `{ "type": "event", "name": "noUserInput", "value": 1 }` ■ `true`: Enabled. ■ `false`: (Default) Disabled. **Note:** The feature is applicable only to Azure, Google, and AudioCodes API (`ac-api`).
`userNoInputSpeech`	String	Defines the textual prompt to play to the user when no input has been received from the user when the timeout expires (configured by `userNoInputTimeoutMS`). The prompt can be configured in plain text or in Speech Synthesis Markup Language (SSML) format: By default, the parameter is not configured. ■ **Plain-text example:** `{ "name": "LondonTube", "provider": "my_azure", "displayName": "London Tube", "userNoInputTimeoutMS": 5000,`

Parameter	Type	Description

		"userNoInputSpeech": "Hi there. Please say something"
		}
		
		■ **SSML example:**
		
		{
		"name": "LondonTube",
		"provider": "my_azure",
		"displayName": "London Tube",
		"userNoInputTimeoutMS": 5000,
		"userNoInputSpeech": <speak>"This is <say-as interpret-as="characters">SSML</say-as>"</speak>
		}
		
		For more information on the no user input feature, see the `userNoInputTimeoutMS`.
		Note:
		■ If you have also configured to play an audio prompt (see the `userNoInputUrl` parameter), the `userNoInputSpeech` takes precedence.
		■ This feature requires a text-to-speech provider. It will not work when the speech is synthesized by the bot framework.
		■ The supported SSML elements depend on the text-to-speech provider:
		✔ **Google:** https://cloud.google.com/text-to-speech/docs/ssml
		✔ **Azure:** https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/speech-synthesis-markup#supported-ssml-elements
		✔ **AWS:** https://docs.aws.amazon.com/polly/latest/dg/supportedtags.html

Parameter	Type	Description
		■ For more information on using SSML for text in the `message` activity, see message Activity on page 20.
`userNoInputUrl`	String	Defines the URL from where the audio prompt is played to the user when no input has been received from the user when the timeout expires (configured by `userNoInputTimeoutMS`). By default, the parameter is not configured. For more information on the no user input feature, see the `userNoInputTimeoutMS.` **Note:** If you have also configured to play a textual prompt (see the `userNoInputSpeech` parameter), the `userNoInputSpeech` takes precedence.
`continuousASR`	Boolean	Enables the Continuous ASR feature. Continuous ASR enables the Voice.AI Gateway to concatenate multiple STT recognitions of the user and then send them as a single textual message to the bot. ■ `true`: Enabled ■ `false`: (Default) Disabled For an overview of the Continuous ASR feature, refer to the Voice.AI Gateway Product Description.
`continuousASRDigits`	String	This parameter is applicable when the Continuous ASR feature is enabled. Defines a special DTMF key, which if pressed, causes the Voice.AI Gateway to immediately send the accumulated recognitions of the user to the bot. For example, if configured to "#" and the user presses the pound key (#) on the phone's keypad, the device concatenates the accumulated recognitions and then sends them as one single textual message to the bot. The default is "#". **Note:** Using this feature incurs an additional delay from the user’s perspective because the speech is not sent immediately to the bot after it has been

Parameter	Type	Description
		recognized. To overcome this delay, configure the parameter to a value that is appropriate to your environment.
`continuousASRTimeoutInMS`	Number	This parameter is applicable when the Continuous ASR feature is enabled. Defines the automatic speech recognition (ASR) timeout (in milliseconds). When the device detects silence from the user for a duration configured by this parameter, it concatenates all the accumulated STT recognitions and sends them as one single textual message to the bot. The valid value is 2,500 (i.e., 2.5 seconds) to 60,000 (i.e., 1 minute). The default is 3,000.
`disableTtsCache`	Boolean	Defines caching of TTS (audio) results from the bot. Therefore, if the Voice.AI Connector needs to send a request for TTS to a TTS provider and this text has been requested before, it retrieves the result from its cache instead of requesting it again from the TTS provider. ■ `true`: TTS caching is disabled. ■ `false`: (Default) TTS caching is enabled. **Note:** This parameter is not applicable when speech-to-text is performed by the bot framework.
`googleInteractionType`	String	Defines the Google STT interaction type. For more information, see https://cloud.google.com/speech-to-text/docs/reference/rest/v1p1beta1/RecognitionConfig#InteractionType.
`handoverReason`	String	Defines the textual reason when the call is transferred to another party (e.g., another bot or a human agent). By default, the parameter is not defined.
`hangupReason`	String	Conveys a textual reason for hanging up (disconnecting call). This reason appears in the CDR of the call.

Parameter	Type	Description

		Example message:
		{
		"type": "event",
		"name": "hangup",
		"activityParams": {
		"hangupReason": "conversationCompleted"
		}
		}

`language`	String	Defines the language (e.g., "en-ZA" for South African English) of the bot conversation and is used for TTS and STT functionality. The value is obtained from the service provider. ■ **STT:** ✔ Azure: The parameter is configured with the value from the 'Locale' column in Azure's Speech-Text table (e.g., "en-GB"). ✔ Google: The parameter is configured with the value from the 'languageCode' (BCP-47) column in Google's Cloud Speech-to-Text table (e.g., "nl-NL"). For more information, refer to section Required Information of STT Provider on page 3. ■ **TTS:** ✔ Azure: The parameter is configured with the value from the 'Locale' column in Azure's Text-to-Speech table (e.g., "it-IT"). ✔ Google: The parameter is configured with the value from the 'Language code' column in Google's Cloud Text-to-Speech table (e.g., "en-US"). ✔ AWS: The parameter is configured with the value from the 'Language' column in Amazon's Polly TTS table (e.g., "de-DE").

Parameter	Type	Description
		For more information, refer to section Required Information of TTS Provider on page 4. **Note:** This string is obtained from the TTS or STT service provider by the Customer and must be provided to AudioCodes. For more information, see the Voice.AI Gateway Integration Guide.
`playUrlAltText`	String	Defines the text to display in the transcript page of the user interface while the audio is played.
`playUrlCaching`	Boolean	Enables caching of the audio in the TTS cache: ■ `true`: Enables caching ■ `false`: (Default) Disables caching
`playUrlMediaFormat`	String	Defines the format of the audio: ■ `wav/lpcm16` (default) ■ `raw/lpcm16`
`playUrlUrl`	String	Defines the HTTP-based server by URL where the audio file to be played is located. This allows the play of pre-recorded prompts (audio file) to the user from a remote third-party server.
`resumeRecognitionTimeoutMS`	Number	When Barge-In is disabled, speech input is not expected before the bot's response has finished playback. If no reply from the bot arrives within this configured timeout (in milliseconds), the Voice.AI Gateway expects speech input from the user and STT recognition is re-activated. The valid value is 0 (i.e., no automatic resumption of recognition) to 600,000 (i.e., 10 minutes). The default is 10,000.
`sendDTMF`	Boolean	Enables the sending of DTMF events to the bot. ■ `true`: Enabled ■ `false`: (Default) Disabled **Note:** For configuring the DTMF collection and sending method, see the `dtmfCollect` parameter.

Parameter	Type	Description
`dtmfCollect`	Boolean	Defines the DTMF digit collection and sending method. - `true`: Enabled. The Voice.AI Gateway first collects all the DTMF digits entered by the user, and only then sends them all together to the bot. - `false`: (Default) Disabled. As the Voice.AI Gateway receives a DTMF digit entered by the user, it sends that single digit to the bot. In other words, it sends each DTMF digit one at a time to the bot. **Note:** - When enabled, you can configure additional settings using the following parameters: `dtmfCollectInterDigitTimeoutMS`, `dtmfCollectMaxDigits`, and `dtmfCollectSubmitDigit`. - If the `sendDTMF` parameter is configured to `false` (default), incoming DTMF digits are ignored by the Voice.AI Gateway even if the `dtmfCollect` parameter is configured to `true`. - DTMF collection is not applicable to Dialogflow CX bots, as it is performed by the bot framework.
`dtmfCollectInterDigitTimeoutMS`	Number	Defines the timeout (in milliseconds) that the Voice.AI Gateway waits for the user to press another digit before it sends all the digits to the bot. If the timeout expires since the last digit entered by the user, the Voice.AI Gateway sends all the collected digits to the bot (as a DTMF message), without waiting for the maximum number of expected digits or for the "submit" digit. The timeout is triggered after the user enters the first DTMF digit and is reset after each digit. The valid value range is 0 to unlimited. The default is 2000.

Parameter	Type	Description
		Note: ▪ The parameter is applicable only when the `dtmfCollect` parameter is configured to `true`. ▪ Once the Voice.AI Gateway sends all the DTMF digits to the bot, any additional DTMF digits entered by the user is ignored by the Voice.AI Gateway until the bot responds to the DTMF event message.
`dtmfCollectMaxDigits`	Number	Defines the maximum number of DTMF digits that the Voice.AI Gateway expects to receive from the user. Once the Voice.AI Gateway receives and collects this number of digits entered by the user, it immediately sends all the digits to the bot (as a DTMF message), without waiting for the timeout to expire or for the "submit" digit. The valid value range is 0 (disabled) to unlimited. The default is 5. If configured to 0, the DTMF collection and sending method is according to `dtmfCollectInterDigitTimeoutMS` or `dtmfCollectSubmitDigit`. **Note:** ▪ The parameter is applicable only when the `dtmfCollect` parameter is configured to `true`. ▪ Once the Voice.AI Gateway sends all the DTMF digits to the bot, any additional DTMF digits entered by the user is ignored by the Voice.AI Gateway until the bot responds to the DTMF event message.
`dtmfCollectSubmitDigit`	String	Defines a special DTMF "submit" digit that when received from the user, the Voice.AI Gateway immediately sends all the collected digits to the bot (as a DTMF message), without waiting for the timeout to expire or for the maximum number of expected digits. The valid value is any symbol on a phone keypad. The default is # (pound key). If you want to

Parameter	Type	Description
		disable this parameter, configure it to "" (empty string). **Note:** - The parameter is applicable only when the `dtmfCollect` parameter is configured to `true`. - The Voice.AI Gateway doesn't include this "submit" digit in the DTMF event message sent to the bot. - Once the Voice.AI Gateway sends all the DTMF digits to the bot, any additional DTMF digits entered by the user is ignored by the Voice.AI Gateway until the bot responds to the DTMF event message.
`sttContextId`	String	- **Azure speech-to-text engine:** This parameter controls Azure's Custom Speech model. The parameter can be set to the endpoint ID that is used when accessing the STT engine. For more information on how to obtain the endpoint ID, go to https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/how-to-custom-speech-deploy-model. - **AudioCodes DNN speech-to-text engine:** This parameter controls the context. **Note:** - The parameter can be used by all bot providers, as long as the STT engine is Azure or AudioCodes DNN. - For Azure STT, the Custom Speech model must be deployed on the same subscription used for the Azure STT engine. - When using other STT engines, the parameter has no affect.
`sttContextPhrases`	Array of	When using Google's Cloud STT engine, this parameter controls Speech Context phrases.

Parameter	Type	Description
	Strings	The parameter can list phrases or words that is passed to the STT engine as "hints" for improving the accuracy of speech recognitions. For more information on speech context (speech adaptation) as well details regarding tokens (class tokens) that can be used in phrases, go to https://cloud.google.com/speech-to-text/docs/speech-adaptation. For example, whenever a speaker says "weather" frequently, you want the STT engine to transcribe it as "weather" and not "whether". To do this, the parameter can be used to create a context for this word (and other similar phrases associated with weather): `"sttContextPhrases": ["weather"]` **Note:** ■ The parameter is only for backward compatibility and will be deprecated in the next applicable release. Please use the `sttSpeechContexts` parameter instead. ■ The parameter can be used by all bot providers when the STT engine is Google. ■ When using other STT engines, the parameter has no affect.
`sttContextBoost`	Number	Defines the boost number for context recognition of the speech context phrase configured by `sttContextPhrases`. Speech-adaptation boost allows you to increase the recognition model bias by assigning more weight to some phrases than others. For example, when users say "weather" or "whether", you may want the STT to recognize the word as weather. For more information, see https://cloud.google.com/speech-to-text/docs/context-strength. **Note:** ■ The parameter is only for backward

Parameter	Type	Description
		compatibility and will be deprecated in the next applicable release. Please use the `sttSpeechContexts` parameter instead. ■ The parameter can be used by all bot providers when the STT engine is Google. ■ When using other STT engines, the parameter has no affect.
`sttSpeechContexts`	Array of Strings	When using Google's Cloud STT engine, this parameter controls Speech Context phrases. The parameter can list phrases or words that is passed to the STT engine as "hints" for improving the accuracy of speech recognitions. For example, whenever a speaker says "weather" frequently, you want the STT engine to transcribe it as "weather" and not "whether". To do this, the parameter can be used to create a context for this word (and other similar phrases associated with weather). You can also use the parameter to define the boost number (0 to 20, where 20 is the highest) for context recognition of the specified speech context phrase. Speech-adaptation boost allows you to increase the recognition model bias by assigning more weight to some phrases than others. For example, when users say "weather" or "whether", you may want the STT to recognize the word as "weather". For more information, see https://cloud.google.com/speech-to-text/docs/context-strength. You can also use Google's class tokens to represent common concepts that occur in natural language, such as monetary units and calendar dates. A class allows you to improve transcription accuracy for large groups of words that map to a common concept, but that don't always include identical words or phrases. For example, the audio data may include recordings of people saying their street address. One may say "my house is 123 Main Street, the fourth house on the

Parameter	Type	Description

		left." In this case, you want Speech-to-Text to recognize the first sequence of numerals ("123") as an address rather than as an ordinal ("one-hundred twenty-third"). However, not all people live at "123 Main Street" and it's impractical to list every possible street address in a SpeechContext object. Instead, you can use a class token in the phrases field of the SpeechContext object to indicate that a street number should be recognized no matter what the number actually is.
		For example:
		`"sttSpeechContexts": [{`
		`"phrases”: ["weather"],`
		`"boost": 18`
		`},`
		`{`
		`"phrases”: ["whether"],`
		`"boost": 2`
		`},`
		`{`
		`"phrases”: ["fair"]`
		`},`
		`{`
		`“phrases”: [“$ADDRESSNUM”]`
		`}`
		`]`
		Note:
		■ The parameter can be used by all bot providers when the STT engine is Google.
		■ When using other STT engines, the parameter has no affect.
		■ For more information on speech context (speech adaptation) as well details regarding tokens (class tokens) that can be used in phrases, go to https://cloud.google.com/speech-to-text/docs/speech-adaptation.

Parameter	Type	Description
`sttDisablePunctuation`	Boolean	Prevents the STT response from the bot to include punctuation marks. ■ `true`: Enabled. Punctuation is excluded. ■ `false`: (Default) Disabled. Punctuation is included. **Note:** This requires support from the STT engine.
`sttEndpointID`	String	This parameter has been deprecated in Version 2.2 and replaced by the `sttContextId` parameter.
`azureEnableAudioLogging`	Boolean	Enables recording and logging of audio from the user (endpoint) that the Voice.AI Gateway sends to the STT engine. The recording is done by the STT engine and stored on the STT engine. ■ `true`: Instructs the STT engine to enable audio logging. ■ `false`: Instructs the STT engine to disable audio logging. When the parameter is not defined (default), audio logging is according to the STT engine. **Note:** The parameter and audio logging is applicable only when using the Azure STT.
`targetParticipant`	String	Defines the participant on which to apply the events `startRecognition` and `stopRecognition` for starting and stopping (respectively) speech recognition by the STT engine. **Note:** The parameter is applicable only to Agent Assist calls.
`transferReferredByURL`	String	Defines the party (URL) who initiated the referral. If this parameter exists, the SBC adds a SIP Referred-By header to the outgoing INVITE/REFER message (according to the 'Remote REFER Mode' parameter). If the SBC handles locally (termination), the SBC adds it to a new outgoing INVITE. If not handled locally (regular), the SBC adds it to the forwarded REFER message.

Parameter	Type	Description
`transferSipHeaders`	Array of Objects	Array of objects listing SIP headers that should be sent to the transferee. Each object comprises a `name` and a `value` attribute.
`transferTarget`	String	Defines the URI to where the call must be transferred. Typically, the URI is a "tel" or "sip" URI.
`voiceName`	String	Defines the voice name for the TTS service. ■ Azure: The parameter is configured with the value from the 'Short voice name' column in Azure's Text-to-Speech table (e.g., "it-IT-ElsaNeural"). ■ Google: The parameter is configured with the value from the 'Voice name' column in Google's Cloud Text-to-Speech table (e.g., "en-US-Wavenet-A"). ■ AWS: The parameter is configured with the value from the 'Name/ID' column in Amazon's Polly TTS table (e.g., "Hans"). ■ Almagu: The parameter is configured with the value from the 'Voice' column in Almagu's TTS table (e.g., "Osnat"). **Note:** This string is obtained from the TTS service provider by the Customer and must be provided to AudioCodes. For more information, refer to Section Required Information of TTS Provider on page 4.
`callRecordingId`	String	Defines the recording ID session, which is forwarded by the SBC to the SRS in the XML body (attribute "ac:call-recording-id") of the SIP INVITE request for SIPRec. The Administrator can later retrieve the bot's recordings from the SRS, by using this ID. The parameter is used by the feature where the bot starts and stops voice recording of the conversation. For more information, refer to the *Voice.AI Gateway Integration Guide*.

Parameter	Type	Description
		Below is an example of an XML body with the ID shown in the XML attribute "ac:call-recording-id": <?xml version="1.0" encoding="UTF-8"?> <recording xmlns="urn:ietf:params:xml:ns:recording" xmlns:ac="http://AudioCodes"> <datamode>complete</datamode> <group id="00000000-0000-0072-57a7-310000000072"> <associate-time>2018-01-01T02:22:29</associate-time> </group> <session id="0000-0000-0000-0000-b44497aaf9597f7f"> <group-ref>00000000-0000-0072-57a7-310000000072</group-ref> <associate-time>2018-01-01T02:22:29</associate-time> <**ac:call-recording-id**>KUYdgtofldi76YpkP09J4J-a:1</ac:call-recording-id> </session> <participant id="+123456789" session="0000-0000-0000-0000-b44497aaf9597f7f"> .... **Note:** - The parameter is applicable only to the `startCallRecording` event. - The XML attribute name "ac:call-recording-id" can be changed using the SBC's Message Manipulation feature.
`callRecordingDestUsername`	String	Defines the username that is used in the SIP Request-URI and To headers of the INVITE request for SIPRec. The parameter is used by the feature where the bot starts and stops voice recording of the

Parameter	Type	Description
		conversation. For more information, refer to the *Voice.AI Gateway Integration Guide*. **Note:** The parameter is applicable only to the `startCallRecording` event.

This page is intentionally left blank.

International Headquarters

1 Hayarden Street,

Airport City

Lod 7019900, Israel

Tel: +972-3-976-4000

Fax: +972-3-976-4040

AudioCodes Inc.

200 Cottontail Lane

Suite A101E

Somerset NJ 08873

Tel: +1-732-469-0880

Fax: +1-732-469-2298

Contact us: https://www.audiocodes.com/corporate/offices-worldwide

Website: https://www.audiocodes.com/

Documentation Feedback: https://online.audiocodes.com/documentation-feedback

Document #: LTRT-30932