Integration Guide

AudioCodes Intuitive Human Communications for Chatbot Services

Voice.AI Gateway

audiocodes

Table of Contents

List of Tables

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: March-23-2020

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.

Stay in the Loop with AudioCodes

Abbreviations and Terminology

Each abbreviation, unless widely used, is spelled out in full when first used.

Related Documentation

Document Name
Voice.AI Gateway Product Description
Voice.AI Gateway Bot API Reference Guide

General Notes

Note: 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.

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).

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.

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.

Note:

- 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.

1.1 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.

1.2 Targeted Audience

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

This page is intentionally left blank.

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.

2.1 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	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 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)
AudioCodes Bot API	To create the channel between the Voice.AI Gateway's Cognitive Service component and the bot provider, refer to the Voice.AI Gateway API Reference Guide.

2.2 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. 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 Section Required Information of Bot Framework Provider 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.

2.3 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 Section Required Information of Bot Framework Provider 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.

TTS Provider	Required Information from TTS Provider	
	Connectivity	**Language Definition**
AWS Amazon Polly	To connect to Amazon Polly Text-to-Speech service, see Section Required Information of Bot Framework Provider for required information.	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). ▪ 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.

3 Messages Sent by Voice.AI Gateway

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

3.1 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.

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

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

<table>
<tr><th>Bot Framework</th><th>Message Syntax</th></tr>
<tr><td><b>AudioCodes Bot API</b></td><td>The message is sent as a <code>start</code> event, with the details inside the <code>parameters</code> property.
Example:
<pre>
{
  "type": "event",
  "name": "start",
  "parameters": {
    "callee": "12345678",
    "calleeHost": "10.20.30.40",
    "caller": "12345678",
    "callerHost": "10.20.30.40"
  }
}
</pre></td></tr>
</table>

<table>
<thead>
<tr><th>Bot Framework</th><th>Message Syntax</th></tr>
</thead>
<tbody>
<tr><td><b>Microsoft Azure</b></td><td>The message is sent as a <code>channel</code> event, with the details inside the <code>channelData</code> property.
Example:
<pre><code>{
  "type": "event",
  "name": "channel",
  "value": "telephony",
  "channelData": {
    "callee": "12345678",
    "calleeHost": "10.20.30.40",
    "caller": "12345678",
    "callerHost": "10.20.30.40"
  },
  "from": {
    "id": "12345678"
  },
  "locale": "en-US"
}</code></pre></td></tr>
<tr><td><b>Google Dialogflow</b></td><td>The message is sent as a <code>WELCOME</code> event, with the details as <code>event</code> parameters.
Example:
<pre><code>{
  "queryInput": {
    "event": {
      "languageCode": "en-US",
      "name": "WELCOME",
      "parameters": {
        "callee": "12345678",
        "calleeHost": "10.20.30.40",
        "caller": "12345678",
        "callerHost": "10.20.30.40"
      }
    }
  }
}</code></pre>
<b>Note:</b> These parameters can be used when generating the response text, by using a syntax such as this:
<pre><code>"#WELCOME.caller"</code></pre></td></tr>
</tbody>
</table>

3.2 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 Azure	The conversation is terminated by sending an `endOfConversation` 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.

3.3 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.

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

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

Bot Framework	Message Syntax

AudioCodes Bot API	The message is sent as a `message` activity. Additional details are sent in the `parameters` property.
	Example:
	`{`
	`    "type": "message",`
	`    "text": "Hi.",`
	`    "parameters": {`
	`       "confidence":0.6599681,`
	`    }`
	`}`

<table>
<tr><th>Bot Framework</th><th>Message Syntax</th></tr>
<tr><td><b>Microsoft Azure</b></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><b>Google Dialogflow</b></td><td>The message is sent as text input. Currently, additional details are not sent.
Example:
<pre>{
    "queryInput": {
        "text": {
            "languageCode": "en-US",
            "text": "Hi."
        }
    }
}</pre>
<b>Note:</b> 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.</td></tr>
</table>

3.4 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	This message is sent as a `DTMF` event with the digits as the event parameters.
	Example:
	`{`
	`  "queryInput": {`
	`    "event": {`
	`      "languageCode": "en-US",`
	`      "name": "DTMF",`
	`      "parameters": {`
	`        "digits": "3"`
	`      }`
	`    }`
	`  }`
	`}`
	Note: The digits can be used when generating the response text, by using a syntax such as this:
	`"#DTMF.digits"`

This page is intentionally left blank.

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.

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 by Bot.

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.

4.1 Basic Activity Syntax

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

Table 4-1: 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 Section event Activities.
`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 Section Parameters Controlled by Bot.

4.2 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>.
  ```

Note:

- 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.

4.3 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.

4.3.1 hangup

The `hangup` event disconnects the conversation.

The following table lists the parameters associated with this event.

Table 4-2: 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"
  }
}
```

4.3.2 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.

The following table lists the parameters associated with this event.

Table 4-3: 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`	Boolean	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 Section Adding SIP Headers on Call Transfer.

Example:

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

4.3.2.1 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 4-4: 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>
```

4.3.3 playUrl

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

Note: The format of the file must match the format specified by the `playUrlMediaFormat` parameter; otherwise, the audio will be played corruptly.

The following table lists the parameters associated with this event.

Table 4-5: 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"
  }
}
```

4.3.4 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
  }
}
```

4.4 Bot Framework Specific Details

4.4.1 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 Bot API* documentation.

4.4.2 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"
    }
  }
}
```

4.4.3 Google Dialogflow

For Google Dialogflow, the activities are derived from intent's response (the "Default" response, which is the response to PLATFORM_UNSPECIFIED platform).

The response's text is used to construct a `message` activity for playing the text to the user.

To send additional parameters or activities, Custom Payload must be added to the response (see https://cloud.google.com/dialogflow/docs/intents-rich-messages).

The Custom Payload can contain a JSON object with the following properties:

Table 4-6: Google Dialogflow Custom Payload Properties

Property	Description
`activityParams`	This is applied when playing the text of the response (i.e., of the `message` activity).
`sessionParams`	This is applied when playing the text of the response (i.e., of the `message` activity).
`activities`	Array of activities to be executed after playing the text of the response.

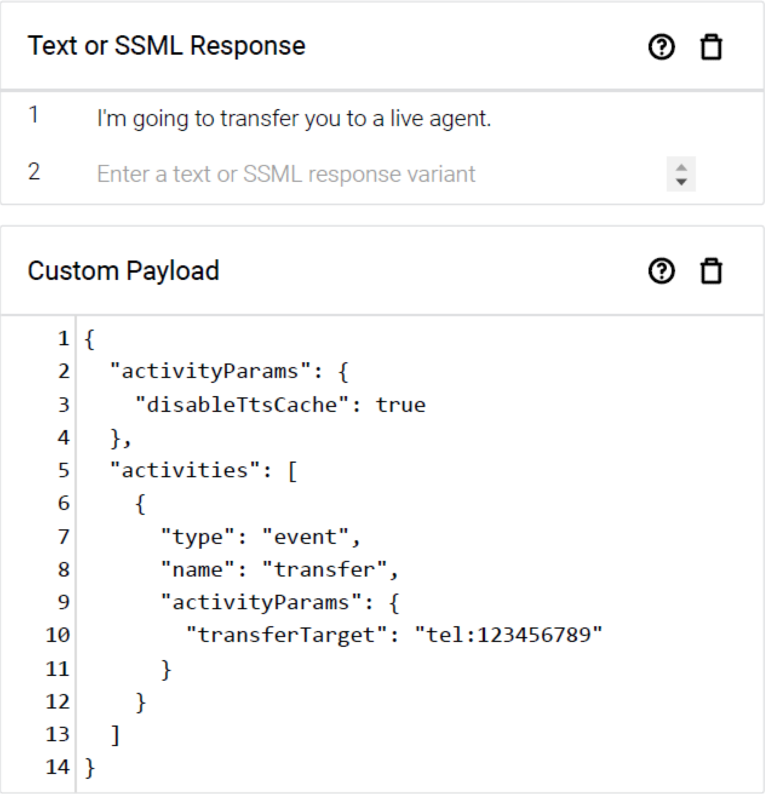
For example, if the text response is "I’m going to transfer you to a human agent" and the Custom Payload contains the following JSON object:

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

Then the audio of the text "I’m going to transfer you to a human agent." is played without caching (due to the `disableTtsCache` parameter). After it has finished playing, the `transfer` activity is executed.

The above example can be configured through the Dialogflow user interface, as follows:

Figure 4-1: Custom Payload Configuration Example through Dialogflow User Interface



Text or SSML Response

1 I'm going to transfer you to a live agent.

2 Enter a text or SSML response variant

Custom Payload

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

4.5 Parameters Controlled by Bot

The following table lists the parameters that can be updated dynamically by the bot. Parameters that are specific to a single event type are documented in Section event Activities. As explained in Section Basic Activity Syntax, these parameters can be included in the `activityParams` or the `sessionParams` of any activity sent by the bot.

Table 4-7: Parameters Controlled by Bot

Parameter	Type	Description
`language`	String	Defines the language of the conversation. For more information, refer to sections Required Information of STT Provider and Required Information of TTS Provider.
`voiceName`	String	Defines the voice name for text-to-speech. For more information, refer to Section Required Information of TTS Provider.
`sttDisablePunctuation`	Boolean	Prevents the speech-to-text response from the bot to include punctuation marks. ▪ `true`: Enabled - punctuation excluded ▪ `false`: (Default) Disabled - punctuation included **Note:** This requires support from the speech-to-text engine.
`sttEndpointID`	String	A synonym for the `sttContextId` parameter.
`bargeIn`	Boolean	Enables the Barge-In feature. ▪ `true`: Enabled - when the bot is playing a response to the user (playback of bot message), the user can "barge-in" (interrupt) and start speaking. This terminates the bot response, allowing the bot to listen to the new speech by 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.
`bargeInMinWordCount`	Number	Defines the minimum number of words that the user has to say in order 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.
`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 speech-to-text 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.

Parameter	Type	Description
`disableTtsCache`	Boolean	Disables caching of the text-to-speech (audio) result from the bot. - `true`: Enabled - `false`: (Default) Disabled
`continuousASR`	Boolean	Enables the Continuous ASR feature. Continuous ASR enables the Voice.AI Gateway to concatenate multiple speech-to-text 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 Product Description.
`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 speech-to-text 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.
`continuousASRDigits`	String	This parameter is applicable when the Continuous ASR feature is enabled. Defines a special DTMF key that 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 recognized. To overcome this delay, configure the parameter to a value that is appropriate to your environment.
`googleSendDTMF`	Boolean	Enables the sending of DTMF events to the Google Dialogflow bot. - `true`: Enabled - `false`: (Default) Disabled

Parameter	Type	Description
`sttContextId`	String	When using Azure's speech-to-text (STT) 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. **Note:** - The parameter can be used by all bot providers, as long as the STT engine is Azure. - The Custom Speech model must be deployed on the same subscription that is used for the Azure STT engine. - When using other STT engines, the parameter has no affect.
`sttContextPhrases`	Array of Strings	When using Google's Cloud Speech-to-Text (STT) engine, this parameter controls Speech Context phrases. The parameter can be set to a list of 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 can be used by all bot providers, as long as the STT engine is Google. - When using other STT engines, the parameter has no affect.
`sttContextBoost`	Number	When using Google's Cloud Speech-to-Text engine, this parameter controls the boost adaptation of the phrases defined by the `sttContextPhrases` parameter. For more information on boost adaptation, go to https://cloud.google.com/speech-to-text/docs/speech-adaptation. **Note:** - The parameter can be used by all bot providers, as long as the STT engine is Google. - When using other STT engines, the parameter has no affect.

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/