Avaya Experience Portal 8.1.2
Google Dialogflow Native Support White Paper

Abstract

This paper provides information about the support for Connecting AEP self-service applications to Google Dialogflow. It provides details on configuration/licensing etc. to help customers with Dialogflow integration.

Issue 1.01
May 2023

Contents

Overview - 4 -

1 Feature Implementation and Description Overview - 5 -

1.1 AEP default VXML Dialogflow Application - 5 -

1.1.1 Invoking default VXML application as a sub-dialog - 5 -

1.1.1.1 Configuration Param - 6 -

1.1.1.2 Credentials Param - 7 -

1.1.1.3 Ringback Prompt URL Param - 7 -

1.1.2 Default VXML Dialogflow Application directory replaced during upgrades - 8 -

1.1.3 Default VXML Dialogflow Application interaction with Dialogflow - 9 -

1.1.4 Description of simple bot Interactions - 9 -

1.2 Termination of Interactions with Dialogflow - 10 -

1.3 Google Dialogflow bot - 11 -

2 Configure Google Dialogflow - 11 -

2.1 Google Dialogflow Virtual Agent (bot) - 11 -

2.2 GCP Project Entitlement - 14 -

2.2.1 GCP Project Setup by Avaya - 14 -

2.2.2 Exceptions - 14 -

2.3 Authorization & Authentication - 14 -

2.4 Dialogflow ES Agents – Enable BETA FEATURES - 14 -

2.5 GCP Project ID - 14 -

2.6 Dialogflow CX Agent ID - 15 -

3 Experience Portal Dialogflow Configuration - 15 -

3.1 AEP and MPP NTP and Internet access - 15 -

3.1.1 Firewall requirements for MPP - 15 -

3.2 Configure Faster Detection of Dead TCP Connections - 15 -

3.3 Accessing Google Dialogflow using http proxy - 16 -

3.3.1 Operating System HTTP Proxy Configuration - 16 -

3.3.2 Dialogflow HTTP Proxy Configuration - 16 -

3.4 Google Dialogflow licensing - 16 -

3.5 Add Dialogflow ASR Server - 16 -

3.6 Add Dialogflow Application to EP - 17 -

3.6.1 VAD Parameters in Dialogflow Application - 18 -

3.6.2 Vendor Parameters in Dialogflow Application - 22 -

3.6.3 Raw parameters - 25 -

3.7 Launch Dialogflow application - 25 -

3.8 Refreshing Dialogflow credentials using REST API - 25 -

5 Experience Portal - Dialogflow Interaction Definitions - 27 -

5.1 Welcome Event Sent from AEP to Dialogflow - 27 -

5.1.1 sip-hdrs Dialogflow context - 27 -

5.1.2 avaya-session-telephone Dialogflow context - 28 -

5.2 AEP detection of End of Conversation signalled by Dialogflow bot - 29 -

5.3 Hangup Event Sent from AEP to Dialogflow - 30 -

5.4 Dialpad entered DTMF collection using telephony_read_dtmf custom payload - 30 -

5.5 TELEPHONY_DTMF event to pass back collected DTMF to bot - 31 -

5.6 Barge-in Control - 31 -

5.6.1 Bargein usage in AEP Dialogflow - 31 -

5.7 VXML privacy feature control - 32 -

5.7.1 Intent Privacy - 32 -

5.7.2 Application Privacy (recommended) - 32 -

5.8 Playing of Pre-Recorded prompts - 32 -

5.9 DTMF Transfer (Play DTMF into call for Feature Access Code Transfer) - 33 -

5.10 VXML Transfer ..... - 33 -
5.10.1 Support for reporting Failed or Completed Bridge Transfers ..... - *34* -
5.11 Timer Support – No Input Timeout and Speech Complete Timeout ..... - 35 -
5.11.1 No input timeout ..... - *35* -
5.11.2 Speech complete timeout ..... - *35* -
5.12 Number of Attempts support DF_SYSTEM_NO_INPUT event ..... - 35 -
5.13 Play Audio until Avaya telephony Follow Up Event Received - Fetch Audio ..... - 36 -
5.14 Playing consecutive prompts using Follow Up Event ..... - 36 -
5.15 Playing mix of pre-recorded audio and Text to Speech (TTS) ..... - 38 -
5.16 Multilingual Support ..... - 38 -
5.17 Changing SynthesizeSpeechConfig at Runtime ..... - 39 -
5.18 Partial response Support ..... - 39 -
6 Experience Portal Voice Activity Detector ..... - 40 -
6.1 VAD Configuration ..... - 41 -
6.1.1 Application Specific VAD Configuration ..... - *41* -
6.1.1.1 **VAD Mode** ..... - 41 -
6.1.1.2 **VAD Type** ..... - 42 -
6.1.1.3 **GMM VAD Tuning Parameters** ..... - 42 -
6.1.1.3.1 Starting Minimum Energy ..... - 42 -
6.1.1.3.2 Starting Minimum Energy Scale ..... - 42 -
6.1.1.3.3 Starting Minimum Energy Frames ..... - 42 -
6.1.1.3.4 Interim Minimum Energy ..... - 42 -
6.1.1.3.5 Interim Minimum Energy Scale ..... - 42 -
6.1.1.3.6 Interim Minimum Energy Frames ..... - 42 -
6.1.1.4 **Inqa VAD Tuning Parameters** ..... - 43 -
6.1.1.4.1 Starting Noise Floor ..... - 43 -
6.1.1.4.2 Starting Noise Offset ..... - 43 -
6.1.1.4.3 Interim Noise Floor ..... - 43 -
6.1.1.4.4 Interim Noise Offset ..... - 43 -
6.1.2 Global System wide VAD Configuration ..... - *43* -
6.1.2.1 **Global** system wide **VAD Selection** ..... - 43 -
6.1.2.2 **Global system wide VAD Tuning for GMM VAD** ..... - 43 -
6.1.2.2.1 GMM VAD Aggressiveness ..... - 44 -
6.1.2.2.2 GMM VAD Minimum Energy ..... - 44 -
6.1.2.2.3 GMM VAD Minimum Speech Frames ..... - 44 -
6.1.2.2.4 GMM VAD Initial Minimum Energy ..... - 44 -
6.1.2.2.5 GMM VAD Initial Minimum Speech Frames ..... - 44 -

Overview

The Avaya Experience Portal now supports full native integration with Google Dialogflow.

Google Dialogflow allows customers to access Cloud AI based automation for voice calls via Experience Portal.

Dialogflow speech recognition extract **intents** from customer conversations. These intents are then used to drive process automation (room booking, reservations etc.) and find relevant answers from FAQs. These "bots" are developed on the Google cloud platform via https://dialogflow.com/.

Experience portal provides the telephony gateway and call management functionality to compliment Dialogflow. AEP streams audio from a caller to Google Dialogflow via Google's gRPC and acts on responses from Dialogflow i.e. transfer call, collect DTMF, play audio file/DTMF etc.

Experience portal provides out of the box integration with Google Dialogflow for voice applications via a default VXML application on MPP. This is responsible as the main interface for integrating with any Dialogflow bot. No changes are needed to the default Dialogflow application on MPP.

1 Feature Implementation and Description Overview

Experience Portal will extend the existing Google Speech interface to support the Google CC AI API set (Google Dialogflow). This API defines the interaction between Experience Portal and Dialogflow and is accessible via https://dialogflow.com/ .

Note: Google's commercial model is to sell the telephony enabled variant of CCAI via telephony partners. The telephony enabled CCAI API can be purchased via Avaya or an Avaya partner (see GCP Project Entitlement).

1.1 AEP default VXML Dialogflow Application

AEP supplies a default VXML application called **def_dialogflow.vxml** on MPP which is responsible as the main application for integrating with Dialogflow. This application interprets responses and handles all of the interactions supported by AEP (transfer, collect DTMF etc).

This VXML application sends the initial Welcome event and then loops handling responses from Dialogflow that are returned in the recognition result. For example, playing audio responses from Dialogflow, collecting DTMF locally, DTMF transfer and so on. It also checks for endInteraction key/value pair being set to true in the response whereby the def_dialogflow VXML application will exit or return, if it is called as a sub-dialog.

This application is named def_dialogflow.vxml and is located on MPP at directory: **$MPP/web/misc/dialogflowapp**. It is accessed using the following URL: http://127.0.0.1/mpp/misc/dialogflowapp/def_dialogflow.vxml

Note: Use 127.0.0.1 so that each MPP uses its own httpd to access the VXML.

1.1.1 Invoking default VXML application as a sub-dialog

The def_dialogflow.vxml application can be invoked directly or invoked as a VXML sub-dialog. A sample application is provided on MPP mentioning how this is done. The sample application is called invoke_def_dialogflow.vxml and is located on MPP at: **$MPP/web/misc/dialogflowapp/test**

The following parameters must be passed to the default VXML application by the calling VXML application:

```
<param name="calledAsSubDialog" value="true"/>
<param name="sipInfoFromParent" value="session.connection.protocol.sip"/>
```

The def_dialogflow.vxml application returns an array of responses received from Google Dialogflow.

In addition to the above subdialog <param/> elements, there are the following optional subdialog <param/> elements.

- <param name="configuration"/>

- <param name=”credentials”/>
- <param name=”ringbackPromptUrl”/>

The InvokeDefaultDFVxml OD sample application also demonstrates how to use these parameters when calling def_dialogflow.vxml as a subdialog from an OD application.

1.1.1.1 Configuration Param

This configuration <param/> allows the calling VXML application to specify the configuration parameters that are used to create a conversation with a Dialogflow Agent. The param carries a JSON string representation of the configuration object.

The configuration JSON object can contain Dialogflow parameters that can be configured in EP web admin > System Configuration > Applications, including:

- Project ID
- VAD Parameters in Dialogflow Application
- Vendor Parameters in Dialogflow Application
- Raw parameters

Note: The Location field in the Application cannot be specified in the configuration subdialog parameter. Any Dialogflow parameters that require the Google region to be specified (such as "project-id"), must have the region built into the locations/ path of the parameter.

When the configuration subdialog param is defined, these configuration parameters will take precedence over any corresponding configuration parameters that are provisioned in EP web admin > System Configuration > Applications or Speech Servers.

If the calling application has defined a configuration JS object assigned to a <var/>, then the ‘expr’ attribute of the <param/> can be used with the JSON.stringify() method to populate the param with the string representation of the JS object. For example.

```
<var name="config"/>
<script>
    <![CDATA[
        config = new Object();
        config.project-id = “aep-test-project”;
        config.conversationProfile = new Object();
        config.conversationProfile.automatedAgentConfig = new Object();
        config.conversationProfile.automatedAgentConfig.agent = “projects/aep-
test-project/locations/global/agents/b74b92bc-ce9b-4752-83a9-1a1d658a1221”;
    ]]>
</script>
<subdialog name="dialogflow-subdialog"
src="http://localhost/mpp/misc/dialogflowapp/def_dialogflow.vxml">
    <param name="configuration" expr="JSON.stringify(config)"/>
</subdialog>
```

The calling application can also pass the configuration JSON as a HTML encoded string. In this case the ’value’ attribute of the <param/> element is used to pass the JSON string representation of the configuration object. For example.

```
<param name="configuration" value="{&quot;project-id&quot;:&quot;aep-sample-project&quot;,&quot;conversationProfile&quot;:{&quot;automatedAgentConfig&quot;:{&quot;agent&quot;:&quot;projects/aep-sample-project/locations/global/agents/b74b92bc-ce9b-4752-83a9-1a1d658a1221&quot;}}}"/>
```

Below is an example of the configuration JSON with sample Dialogflow parameters. This JSON would request AEP to launch a Dialogflow CX Agent (referenced by the agent ID) that is in the aep-test-project GCP project:

```
{
    "project-id": "aep-test-project",
    "provider": "dialogflow_ccaiv2b1",
    "conversationProfile": {
        "automated_agent_config": {
            "agent": "projects/aep-test-project/locations/global/agents/b74b92bc-ce9b-4752-43b8-1a1d658a1221"
        }
    }
}
```

1.1.1.2 Credentials Param

The credentials <param/> allows the calling VXML application to specify the Dialogflow credentials that will be used by Experience Portal to authorize, authenticate and provide access to the Dialogflow Agent through the Dialogflow APIs. The param carries a JSON string representation of the credentials object – this is the content of the Service Account Key JSON file.

When the credentials subdialog param is defined, these credentials take precedence over any Dialogflow credentials that are provisioned in EP web admin > System Configuration > Applications or Speech Servers.

Like the configuration param:

- the ‘expr’ attribute of the credentials <param/> can be used with JSON.stringify() if the JSON credentials are defined as a JS object in a <var/> in the calling application or
- the ‘value’ attribute of the credentials <param/> can be used to carry the credentials JSON as a HTML encoded string

This param also allows a calling application to implement credentials key rotation on a per Application basis.

1.1.1.3 Ringback Prompt URL Param

The ringbackPromptUrl <param/> allows the calling application to control the ringback that def_dialogflow.vxml plays to the caller while waiting for the conversation to be established with the Dialogflow Agent.

This param carries a string value.

If a URL is supplied as the ‘value’ attribute of the <param/>, def_dialogflow.vxml will download the audio file from this URL and play it to the caller instead of the default

ringback (en-US). This allows the calling application to provide locale specific ringback to the caller. For example:

```
<param name="ringbackPromptUrl" value="https://web-server/ringback.wav"/>
```

If the 'value' attribute of the <param/> is populated with two single quotes (empty string), def_dialogflow.vxml will disable the default ringback. The caller will hear silence while waiting for the conversation to be established with the Dialogflow Agent. For example:

```
<param name="ringbackPromptUrl" value="''"/>
```

1.1.2 Default VXML Dialogflow Application directory replaced during upgrades

The directory $MPP/web/misc/dialogflowapp is deleted and re-created during Experience Portal upgrades. This means that all new files and changes to existing files in this directory are lost. If custom changes are required, then the dialogflowapp directory must be copied and changes must be made within this copied directory. The Application can then be accessed from this new directory. Customers must then manually merge changes from default dialogflow directory to the new directory after upgrade.

Figure 1 shows how Experience Portal interacts with Google Dialogflow to deliver rich self-service applications.

Figure 1: Experience Portal – Dialogflow Native Integration – High-level

2. Call arrives on Experience Portal (MPP) via SBC
3. MPP will trigger Dialogflow default VXML application (**def_dialogflow.vxml**) which will connect with Dialogflow using configured project ID and credentials.
4. Dialogflow sends response (containing audio prompt and custom payload if defined) to MPP
5. MPP plays audio to customer call. MPP interprets custom payload sent in response from bot and acts on it (collects DTMF, transfer etc) and sends back details to bot. MPP will stream audio to Dialogflow bot.
6. Dialogflow sends response to MPP again with audio and custom payload (if defined)
7. MPP def_dialogflow.vxml will loop back to step 5 until the interaction is over.

Experience Portal provides DTMF detection and Voice Activity detection services for Dialogflow. AEP also provides SIP header and normal call parameters to Dialogflow in the initial so the "bot" can handle the call more effectively.

1.1.3 Default VXML Dialogflow Application interaction with Dialogflow

The flowchart in Figure 2 shows how def_dialogflow.vxml interacts with the Dialogflow bot. This flowchart shows that the VXML simply starts recognition (load grammar) with Dialogflow and then waits for a response from the bot on what to do next.

1.1.4 Description of simple bot Interactions

The following simple bot will now be used to explain the interactions between AEP and Dialogflow bot. The simple bot does the following:

1. Welcome intent “Hello, welcome. Please use dial pad to enter five digits”
2. Instruct Experience portal to collect five DTMF digits
3. Play the collected digits back to the user “The digits entered are: 12345” and sets this as the end of conversation

The interactions are as described:

1. **BUILD_WELCOME**: EP sends the “Welcome” event to Dialogflow bot.
2. bot receives event and triggers Welcome intent. The bot responds with audio: “Hello, welcome, please use dial pad to enter five digits”. bot also adds the custom payload:

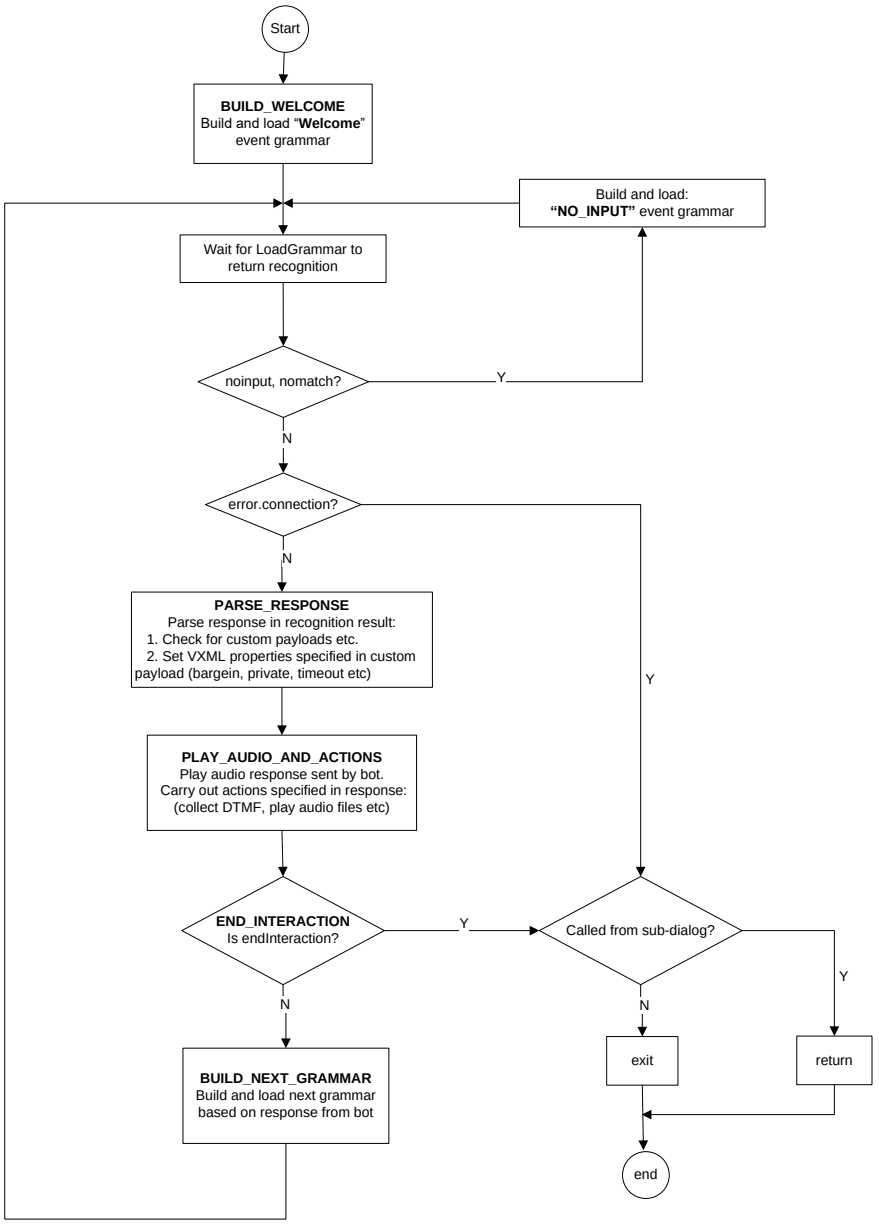
```
"telephony_read_dtmf": {
    "max_duration": "10s",
    "max_digits": 5,
    "listen_to_speech": false,
    "finish_digit": "DTMF_STAR"
}
```

3. **PARSE_RESPONSE**: EP parses the response and determines that the bot has instructed EP to collect five DTMF digits.
4. **PLAY_AUDIO_AND_ACTIONS**: EP plays the audio “Hello, Welcome. Please…” sent back from bot and then sets up local DTMF collection for five digits. Note: EP will also stop sending voice to Dialogflow as “listen_to_speech” is false.
5. User enters five digits: 590227 followed by *
6. **BUILD_NEXT_GRAMMAR**: EP will detect DTMF and on completion will send “TELEPHONY_DTMF” event to bot:

```
"event_input": {
    "name": "TELEPHONY_DTMF",
    "parameters": {
        "telephony_dtmf_digits": "590227"
    }
}
```

7. Bot receives “TELEPHONY_DTMF” event and sends audio to EP:
   `"Got DTMF via dialpad: #TELEPHONY_DTMF.telephony_dtmf_digits"`
   Note: This intent is also marked as “end of conversation”
8. **PARSE_RESPONSE and PLAY_AUDIO_AND_ACTIONS**: EP will play the audio and “Got DTMF via dialpad 590227”.
9. **END_INTERACTION**: EP detects that “endInteraction” is set to true. VXML will then finish (calling return if called from sub-dialog or exit)

Figure 2: Default VXML flowchart



1.2 Termination of Interactions with Dialogflow

Interactions between AEP and DialogFlow are terminated when:

- The customer caller hangs-up.
- The Dialogflow "bot" finished its flow and instructs AEP to end the call.
- The Dialogflow "bot" instructs AEP to complete a blind or consultative transfer of the call. Note: Bridge transfer keeps Dialogflow bot in the call.
- The Dialogflow "bot" instructs AEP that this is "**end of interaction**" intent in the response. AEP hangs up call.

The Default Dialogflow VXML Application purposefully leaves the Dialogflow CCAI Conversation in the **IN_PROGRESS** lifecycle state such that further phases of an application (e.g., CCAI agent assist) can continue to use the Conversation state. Consequently, should an application using the Default Dialogflow VXML prefer that the Dialogflow Conversation move to the **COMPLETE** lifecycle state then the application must

be designed to execute the Dialogflow CompleteConversation REST API method when the VXML application terminates. The execution of the CompleteConversation API method is left to the application as the AEP platform does not provide any facility to call this method.

1.3 Google Dialogflow bot

Successful integration with Dialogflow requires the development of a Google Dialogflow "bot" coordinated with the use of predefined custom payloads which are specified in the Dialogflow intent. These custom payloads pass data between Dialogflow and AEP.

2 Configure Google Dialogflow

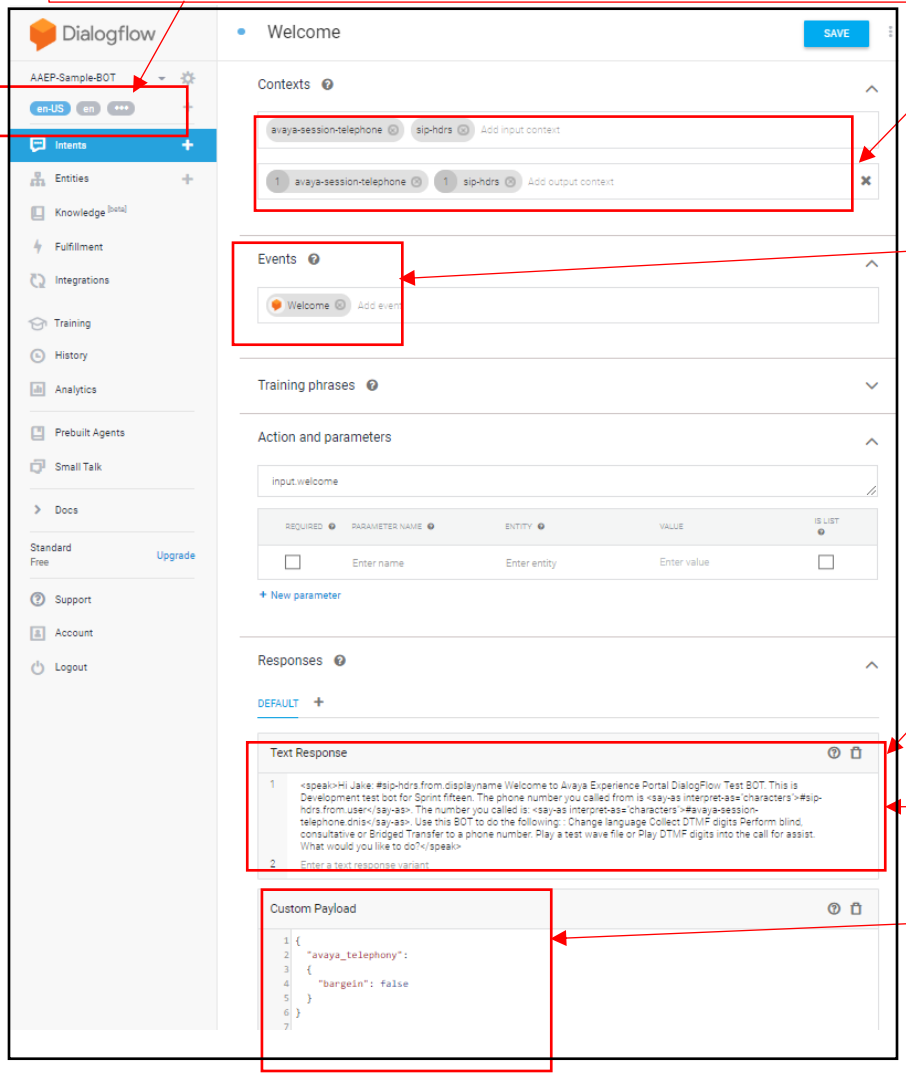
Experience Portal Dialogflow integration with the Google CCAI API set (Google Dialogflow) requires the development of a Google Dialogflow Agent. Google have tutorials, documentation and examples available at: https://dialogflow.com.

2.1 Google Dialogflow Virtual Agent (bot)

Dialogflow “bots” are developed in Google’s Dialogflow console:
https://dialogflow.cloud.google.com for ES agents or
https://dialogflow.cloud.google.com/cx/projects for CX agents.

Experience Portal Dialogflow starts by sending a “Welcome” event to the Google “bot” which contains the SIP Header information of the incoming call and other Avaya Telephony parameters. The “bot” then sends a response with audio content for Experience Portal to play to the customer caller which can also include predefined custom payloads specified in the matching Dialogflow intent. These custom payloads are used to pass data between Dialogflow and AEP. The below ES example shows the relevant fields:

Languages/Regions: Languages/Regions for Intent. This matches the language passed by AEP in the responses. AEP specifies language tags with region (i.e. en-us, es-es, en-gb or language only (en)).
CAVEAT: Intent fields are unique to each language. Make sure you have selected the correct language locale (**en-gb**, **en-us or en**) when editing intent, otherwise response will not contain the values set in the intent.



Input Contexts: containing SIP header and Avaya telephony parameters. These are accessed in TTS response below: **#sip-hdrs.from.displayname** and **#avaya-session-telephone.dnis**

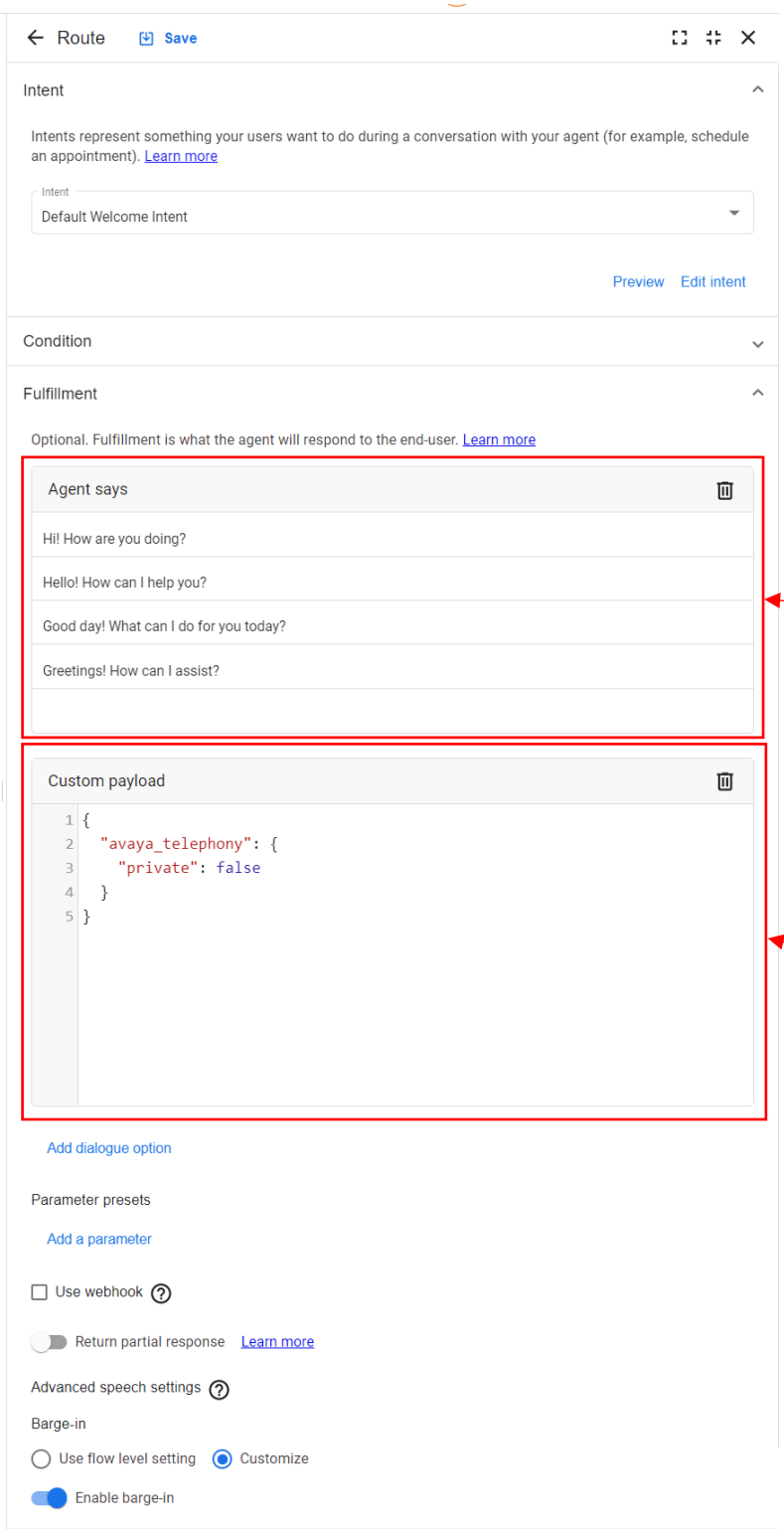
Events: Incoming Event that will trigger this "Welcome" Intent.

Text Response: The text that is converted to base64 encoded audio by Google and returned in the response to AEP. AEP plays this audio

Custom Payload: disabling barge in. This is sent to AEP in the response. AEP then disables barge in while playing the response audio.

Note: This field is language specific, if you add a custom payload when **en** language is selected, it will not be in the intent for **en-us** language.

The "Welcome" event in CX agents will trigger the Default Welcome Intent which has similar fields:



Text Response: The text that is converted to base64 encoded audio by Google and returned in the response to AEP. AEP plays this audio

Custom Payload: disabling barge in. This is sent to AEP in the response. AEP then disables barge in while playing the response audio.

Note: This field is language specific, if you add a custom payload when **en** language is selected, it will not be in the intent for **en-us** language.

2.2 GCP Project Entitlement

The Avaya Experience Portal integration with Google Dialogflow requires telephony enabled CCAI entitlements. These entitlements can be ordered via Avaya, or an Avaya business Partner.

Further details of the offer are described in the following link:
https://sales.avaya.com/en/pss/avaya-ai-virtual-agent

2.2.1 GCP Project Setup by Avaya

Once the Avaya order for telephony enabled CCAI entitlements is processed, a member from the Avaya NPI Operations team will contact the customer to:

- configure the customers GCP account to enable the CCAI entitlements
- provide the requested Dialogflow Agents
- provide the Service Account Key files (JSON)

2.2.2 Exceptions

There are certain exceptions where, as agreed by Avaya and Google, the Contact Center enabled CCAI Dialogflow entitlements can be purchased directly from Google.

In this case Google will need to be engaged to allow list the GCP project for CCAI and setup the GCP project for Dialogflow before it can be used.

2.3 Authorization & Authentication

GCP Service Accounts and Roles authorize Avaya Experience Portal to communicate with Dialogflow Agents through Dialogflow APIs.

GCP Service Account Keys authenticate Avaya Experience Portal when invoking Dialogflow APIs to allow communication with Dialogflow Agents.

Service Account Key files (JSON) are uploaded to Avaya Experience Portal as Dialogflow credentials for this purpose.

Note: GCP Service Account Key files must be stored securely by the customer.

2.4 Dialogflow ES Agents – Enable BETA FEATURES

1. Browse to https://dialogflow.cloud.google.com/#/agents and logon using your Google account
2. Click on the Configuration Wheel for your Dialogflow Agent to display the Settings page
3. Ensure **BETA FEATURES** is enabled for the Agent

2.5 GCP Project ID

The Project ID of the Google Cloud Platform Project that contains the customer's Dialogflow Agents can be accessed through the Google cloud console.

1. Browse to https://console.cloud.google.com/
2. Click the projects dropdown at the top of the page

3. The project ID for the GCP project is listed under the ID column of the displayed table.

2.6 Dialogflow CX Agent ID

Each Dialogflow CX Agent has a unique Agent ID. The Agent ID can be retrieved from https://dialogflow.cloud.google.com/.

1. From the Agent dropdown, select **View all agents**
2. On the relevant Agent, click **Actions** and then select **Copy name**

If using CX Agents with Experience Portal, the Agent ID must be populated as the Conversation Profile Automated Agent JSON parameter name as defined in Vendor Parameters in Dialogflow Application.

This can be populated in:

1. the Conversation Profile Automated Agent configuration field of the Application in EP web admin > System Configuration > Applications or
2. the conversationProfile/automatedAgentConfig/agent/ JSON pointer in the 'configuration' subdialog <param/> if you are calling def_dialogflow.vxml as a subdialog

3 Experience Portal Dialogflow Configuration

This section details the AEP configuration required to use Dialogflow native integration.

3.1 AEP and MPP NTP and Internet access

As Google Dialogflow is on the Internet, both the EP and MPP must be configured to use an NTP server and the MPP must be able to access the internet.

1. Check that NTP is configured on both EP and MPP and the times are correct.
2. Check that the MPP can resolve: dialogflow.googleapis.com

3.1.1 Firewall requirements for MPP

The MPP connects to Dialogflow using a TLS connection on port **443** to FQDN: **dialogflow.googleapis.com**. Ensure that Firewall allows this outbound connection.

3.2 Configure Faster Detection of Dead TCP Connections

AEP uses Google grpc libraries for communication with Google Dialogflow. It was observed during testing that these libraries could take up to fifteen minutes to detect a dead TCP connection. This would result in significant call disruption for a short network outage.

In order to speed up the detection of TCP dead connections to around eight seconds, the following Red Hat Linux configuration is required on the AEP MPP server:

1. Log on using a secure shell session (SSH) to the Avaya Enterprise Linux system as a user with root privileges
2. Open the file: /etc/sysctl.conf
3. Add the following lines to the end of this file

```
# Avaya MPP, Speed up detection of Dead TCP connection to approx. eight seconds
net.ipv4.tcp_retries2=6
```

4. Reboot the MPP server for settings to take effect.

3.3 Accessing Google Dialogflow using http proxy

The following configuration is required if http proxy server is used to access the internet in deployment environment. The following sections detail how to configure Experience Portal MPP to use http proxy.

3.3.1 Operating System HTTP Proxy Configuration

Red Hat Operating system uses the following environment variables to configure HTTP proxy. **http_proxy**, **https_proxy** and **no_proxy**. The **http_proxy** and **https_proxy** environment variables are configured to point at the HTTP proxy server. To avoid Experience Portal using proxy for normal communications between EP and MPP processes, **no_proxy** must be configured with the IP address of the EP, MPP and loopback addresses. An example of setting the environment variables is shown below (192.168.1.50 = EP, 192.168.1.51 = MPP):

```
export http_proxy=http://proxy.example.com:80
export https_proxy=http://proxy.example.com:80
export no_proxy=localhost,mpp-hostname,ep-hostname,192.168.1.50,192.168.1.51,127.0.0.1
```

3.3.2 Dialogflow HTTP Proxy Configuration

Dialogflow does not use normal Red Hat Linux http proxy configuration (using http_proxy environment variables). To configure Dialogflow to use a HTTP proxy:
Edit **$MPP/config/mppconfig.xml** and add Parameter:

```
<parameter name="mpp.voip.cloud.channel_args">{"http_proxy": "http://proxy.example.com:80"}</parameter>
```

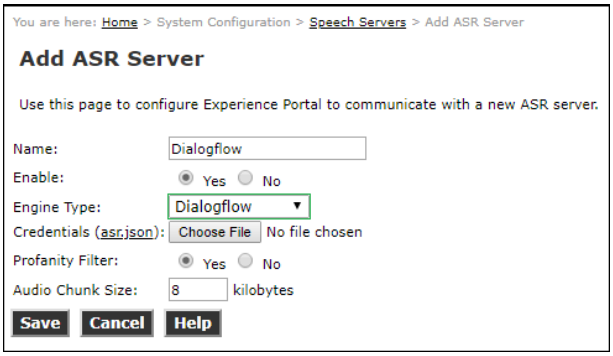
3.4 Google Dialogflow licensing

Google Dialogflow ASR connections are licensed on AEP. Obtain a new license with Google Dialogflow connections and apply license to the WebLM server:

```
<Feature type="counted">
<Name>GoogleDialogflowConnections</Name>
<DisplayName>Maximum number of Google Dialogflow Connections</DisplayName>
<Value>100</Value>
</Feature>
```

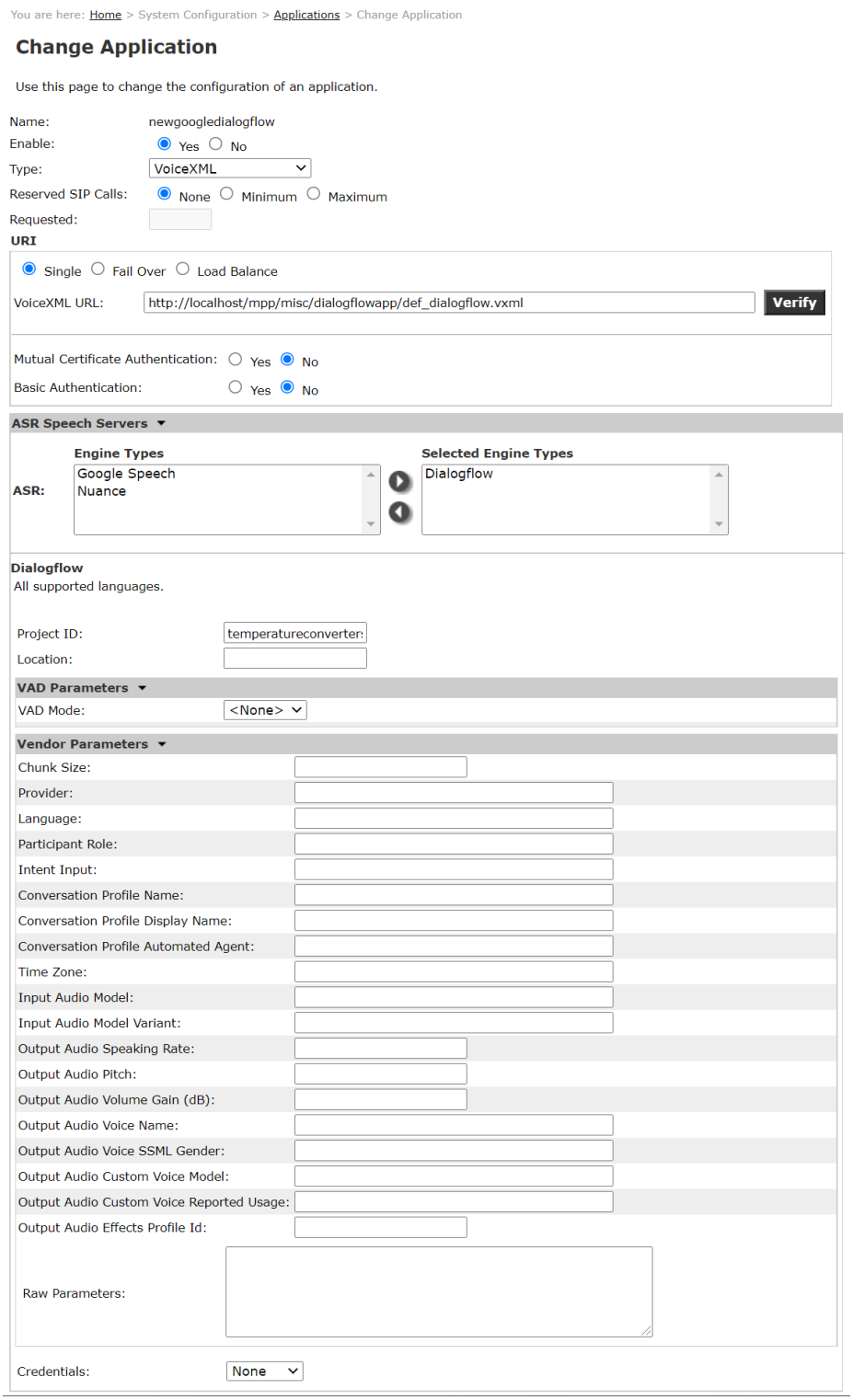
3.5 Add Dialogflow ASR Server

1. Logon to EP and go to: *Home > System Configuration > Speech Servers > Add ASR Server*.
2. Select Engine Type: DialogFlow.
3. On Dialogflow, generate the json credentials and Set Credentials to this license.json file as shown:



3.6 Add Dialogflow Application to EP

1. Logon to EP and go to: *Home > System Configuration > Applications > Add Application*.
2. Enter name.
3. Enter VoiceXML type
4. The URI to the application must point to the def_dialoggflow.vxml application on your MPP server: http://localhost/mpp/misc/dialogflowapp/def_dialogflow.vxml
5. Set the Project ID to the name of the Google Dialogflow project bot.
6. Optionally set the Location (Region ID) of the Google Dialogflow bot. If the field is left blank, the “global” region will be used by default.
7. For CX agents or ES agents using a non-draft environment, the Conversation Profile Automated Agent field must be set to the ID of the Dialogflow agent environment to use. If unset, the draft environment for the ES agent in the specified project is used.
8. The Default VXML application has canned wav files to play for error conditions so a TTS server is not required.
9. Enter calling application launch number.
10. Note: Credentials can also be added here for application specific credentials. This can be left blank, however if credentials are added at the application level, they will override the Dialogflow ASR credentials.



3.6.1 VAD Parameters in Dialogflow Application

The VAD Parameters group controls the Voice Activity Detector settings for the application. VXML applications can override any or all these settings, on a per utterance basis, by supplying the same fields (referenced by the provided JSON Pointer) in the JSON grammar that is executed for the given speech input.

JSON Parameter Name Description
VAD Mode
JSON Pointer: /vadMode Type: enum Valid values: “enabled” – (Default) Uses platform VAD to determine start and end of speech utterances. “hybrid” – Uses platform VAD to eliminate leading silence from speech utterances. Start and end of speech is determined from responses to the streaming request. “disabled” – Bypasses the platform VAD. All audio data, including silence, is sent to Dialogflow for analysis. Utterance endpoints are determined by Dialogflow. Controls when the platform VAD is engaged to regulate the audio data streamed to Dialogflow.
VAD Type
JSON Pointer: /vadType Type: enum Valid values: “GMM” – (Default) Gaussian mixture model detection. “Inqa” – Speech detection using the call progress engine. “MeanSq” – Mean square energy detection. Sets the VAD algorithm used in processing audio data for speech detection. Only applied if the “vadMode” parameter isn’t set to “disabled”.
Aggressiveness
JSON Pointer: /gmm_vad_conf_params/aggressiveness Type: uint32 Valid values: 0 – very relaxed, minimal thresholds for classifying energy as speech. 1 – relaxed, energy moderately matching speech profile is accepted. 2 – aggressive, energy nearly matching speech profile is accepted. 3 – (Default) very aggressive, energy is only classified as speech if it closely matches the calculated speech profile. Specifies tolerance for matching energy with the calculated speech profile. Only applied if “vadMode” isn’t “disabled” and “vadType” is “GMM”.
Starting Minimum Energy
JSON Pointer: /gmm_vad_conf_params/init/min_energy Type: uint32 Valid values: [0, 16383], default 2048 Sets the minimum energy threshold required for frame analysis, specified with 14 significant bits and a power of two scaling factor (gmm_vad_conf_params.init.min_energy_scl). Applied during the barge-in phase if “vadMode” isn’t “disabled” and “vadType” is “GMM”.

Starting Minimum Energy Scale
JSON Pointer: /gmm_vad_conf_params/init/min_energy_scl Type: uint32 Valid values: [0, 32], default 12 Sets the scale factor for the minimum energy threshold required for frame analysis as a power of two. The gmm_vad_conf_params.init.min_energy value is multiplied by 2 raised to the power of this value. Applied during the barge-in phase if “vadMode” isn’t “disabled” and “vadType” is “GMM”.
Starting Minimum Energy Frames
JSON Pointer: /gmm_vad_conf_params/init/min_speech_frms Type: uint32 Valid values: [1, 100], default 5 Sets the minimum number of consecutive frames with speech energy required to trigger the VAD. Applied during the barge-in phase if “vadMode” isn’t “disabled” and “vadType” is “GMM”.
Interim Minimum Energy
JSON Pointer: /gmm_vad_conf_params/intr/min_energy Type: uint32 Valid values: [0, 16383], default 2048 Sets the minimum energy threshold required for frame analysis, specified with 14 significant bits and a power of two scaling factor (gmm_vad_conf_params.init.min_energy_scl). Applied after prompts are finished playing (or if barge-in is disabled) if “vadMode” isn’t “disabled” and “vadType” is “GMM”.
Interim Minimum Energy Scale
JSON Pointer: /gmm_vad_conf_params/intr/min_energy_scl Type: uint32 Valid values: [0, 32], default 10 Sets the scale factor for the minimum energy threshold required for frame analysis as a power of two. The gmm_vad_conf_params.init.min_energy value is multiplied by 2 raised to the power of this value. Applied after prompts are finished playing (or if barge-in is disabled) if “vadMode” isn’t “disabled” and “vadType” is “GMM”.
Interim Minimum Energy Frames
JSON Pointer: /gmm_vad_conf_params/intr/min_speech_frms Type: uint32 Valid values: [1, 100], default 5 Sets the minimum number of consecutive frames with speech energy required to trigger the VAD. Applied after prompts are finished playing (or if barge-in is disabled) if “vadMode” isn’t “disabled” and “vadType” is “GMM”.
Silence Power
JSON Pointer: /msq_vad_conf_params/silence_power Type: float Valid values: [0.0, 120.0], default 40.0 Establishes the threshold for energy detection. Only applied if “vadMode” isn’t “disabled” and “vadType” is “MeanSq”.

Silence Power Difference
JSON Pointer: /msq_vad_conf_params/silence_power_diff Type: float Valid values: >0.0, default 5.0 Establishes the threshold difference for energy detection. Only applied if “vadMode” isn’t “disabled” and “vadType” is “MeanSq”.
Smoothing Factor
JSON Pointer: /msq_vad_conf_params/smoothing_factor Type: float Valid values: [0.0, 1.0], default 0.75 Determines the smoothing factor for the exponential moving average of the energy estimate. Only applied if “vadMode” isn’t “disabled” and “vadType” is “MeanSq”.
Holdover
JSON Pointer: /msq_vad_conf_params/holdover Type: uint32 Valid values: [0, 2000], default 300 Sets the number of milliseconds that energy will continue to be marked as speech after the moving average falls below the silence power. Only applied if “vadMode” isn’t “disabled” and “vadType” is “MeanSq”.
Starting Noise Floor
JSON Pointer: /inqa_vad_conf_params/init/noiseFloor Type: float Valid values: >=0.0, default 22.0 Establishes the initial absolute minimum value for the dynamic noise threshold. Only applied if “vadMode” isn’t “disabled” and “vadType” is “Inqa”.
Starting Noise Offset
JSON Pointer: /inqa_vad_conf_params/init/noiseOffset Type: float Valid values: >=0.0, default 1.0 Defines the starting amount of energy relative to the current dynamic noise threshold, which must be exceeded for an audio frame to be analyzed. Only applied if “vadMode” isn’t “disabled” and “vadType” is “Inqa”.
Interim Noise Floor
JSON Pointer: /inqa_vad_conf_params/intr/noiseFloor Type: float Valid values: >=0.0, default 22.0 Establishes the interim absolute minimum value for the dynamic noise threshold. Only applied if “vadMode” isn’t “disabled” and “vadType” is “Inqa”.
Interim Noise Offset
JSON Pointer: /inqa_vad_conf_params/intr/noiseOffset Type: float Valid values: >=0.0, default 1.0 Defines the interim amount of energy relative to the current dynamic noise threshold, which must be exceeded for an audio frame to be analyzed. Only applied if “vadMode” isn’t “disabled” and “vadType” is “Inqa”.

3.6.2 Vendor Parameters in Dialogflow Application

The Vendor Parameters group can used to set optional parameters used by the platform to set values in the various gRPC messages sent to Dialogflow. The values supplied through the Vendor Parameters fields are applied in the scope of the application. VXML applications can override any or all these settings, on a per utterance basis, by supplying the same fields (referenced by the provided JSON Pointer) in the JSON grammar that is executed for the given speech input.

JSON Parameter Name	Valid Providers
Description	
Provider	
JSON Pointer: /provider Type: enum Valid values: "dialogflowv2" – Utilizes the v2 version of the Dialogflow StreamingDetectIntent method and corresponds with v2 JSON parameters. "dialogflow_ccaiv2" – (Default) Utilizes the v2 versions of the Dialogflow AnalyzeContent/StreamingAnalyzeContent (and associated methods). Corresponds to JSON parameters with the ccaiv2 tag. "dialogflow_ccaiv2b1" – Utilizes the v2beta1 versions of the Dialogflow AnalyzeContent/StreamingAnalyzeContent (and associated methods). Corresponds to JSON parameters with the ccaiv2b1 tag. Determines which version of the Dialogflow API methods are used and, subsequently, which JSON parameters are applied.	
Chunk Size	v2 ccaiv2 ccaiv2b1
JSON Pointer: /chunkSize Type: uint32 Valid values: [400, 32000], default 8192 Controls the number of bytes of audio data per request message in the streaming request.	
Conversation Profile Name	ccaiv2 ccaiv2b1
JSON Pointer: /conversationProfile/name Type: string Format: projects/<Project ID>/locations/<Location ID>/conversationProfiles/<Conversation Profile ID> Can be set to use a specific conversation profile during initialization of the ASR resource or specify an explicit ID to be used in creating a new Conversation Profile. Only applied for the first recognition request on the resource and if the "participant-id" and "conversation-id" fields aren't set. When set, the specified conversation profile will be created if it doesn't already exist and used when creating a new Conversation.	

Conversation Profile Display Name		ccaiv2	ccaiv2b1
JSON Pointer: /conversationProfile/displayName Type: string Determines the human readable name to be used in creating a new Conversation Profile. Only applied for the first recognition request on the resource, if the "participant-id" and "conversation-id" fields aren't set and if the Conversation Profile isn't already created.			
Conversation Profile Automated Agent		ccaiv2	ccaiv2b1
JSON Pointer: /conversationProfile/automatedAgentConfig/agent Type: string Format: ES agents: projects/<Project ID>/locations/<Location ID>/agent/environments/<Environment ID or '-'> CX agents: projects/<Project ID>/locations/<Location ID>/agents/<Agent ID>/environments/<Environment ID or '-'>, only valid with dialogflow_ccaiv2b1 provider Specifies the Dialogflow agent setting for the Conversation Profie. Only applied for the first recognition request on the resource, if the "participant-id" and "conversation-id" fields aren't set and if the Conversation Profile isn't already created.			
Participant Role		ccaiv2	ccaiv2b1
JSON Pointer: /participant/role Type: enum Valid values: "HUMAN_AGENT" – Participant is a human agent. "AUTOMATED_AGENT" – Participant is an automate agent. "END_USER" – (Default) Participant is and end user. Sets the role of the Participant for the resource. Only applied for the first recognition request on the resource and if the "participant-id" field isn't set.			
Language	v2	ccaiv2	ccaiv2b1
JSON Pointer: /language Type: string Format: Tag for one of Dialogflow's supported language. Determines the default language code sent in input requests to Dialogflow. Can be overridden by the VXML application on a per utterance basis.			
Intent Input	v2	ccaiv2	ccaiv2b1
JSON Pointer: /init/intentInput/intent Type: string Format: projects/<Project ID>/locations/<Location ID>/agents/<Agent ID>/intents/<Intent ID> Represents the intent to trigger in the agent in place of the normal "Welcome" event.			
Time Zone	v2	ccaiv2	ccaiv2b1
JSON Pointer: /queryParams/timeZone Type: string Format: A time zone from the time zone database. Sets the time zone for conversational requests.			

Text Sentiment Analysis	v2	ccaiv2	ccaiv2b1
JSON Pointer: /queryParams/sentimentAnalysisRequestConfig/analyzeQueryTextSentiment Type: bool Enables or disables sentiment analysis on query text.			
Enable Word Info	v2	ccaiv2	ccaiv2b1
JSON Pointer: /audioConfig/enableWordInfo Type: bool When true recognition results will include information about the recognized speech words.			
Input Audio Model	v2	ccaiv2	ccaiv2b1
JSON Pointer: /audioConfig/model Type: string Format: Tag for one of Dialogflow's speech to text models. Controls the speech to text model used for streaming audio requests.			
Input Audio Model Variant	v2	ccaiv2	ccaiv2b1
JSON Pointer: /audioConfig/modelVariant Type: enum Valid values: "USE_BEST_AVAILABLE" – (Default) Uses the best variant available for the chosen model. "USE_STANDARD" – Use only the standard model variant. "USE_ENHANCED" – Use the enhanced model variant, if available. Will return an error if enhanced models aren't enabled on the project. Determines the speech to text model variant used for streaming audio requests.			
Output Audio Speaking Rate	v2	ccaiv2	ccaiv2b1
JSON Pointer: /outputAudioConfig/synthesizeSpeechConfig/speakingRate Type: float Valid values: [0.25, 4.0], default 1.0 Sets the speaking rate for generated TTS audio.			
Output Audio Pitch	v2	ccaiv2	ccaiv2b1
JSON Pointer: /outputAudioConfig/synthesizeSpeechConfig/pitch Type: float Valid values: [-20.0, 20.0], default 0.0 Sets the speaking pitch for generated TTS audio.			
Output Audio Volume Gain	v2	ccaiv2	ccaiv2b1
JSON Pointer: /outputAudioConfig/synthesizeSpeechConfig/volumeGainDb Type: float Valid values: [-96.0, 16.0], default 0.0 Adjusts the volume for generated TTS audio.			
Output Audio Effects Profile Id	v2	ccaiv2	ccaiv2b1
JSON Pointer: /outputAudioConfig/synthesizeSpeechConfig/effectsProfileId[] Type: array of strings Format: Comma separated list of audio profile IDs. Selects the audio effects profiles that are applied to generated TTS audio.			

Output Audio Voice Name	v2	ccaiv2	ccaiv2b1
JSON Pointer: /outputAudioConfig/synthesizeSpeechConfig/voice/name Type: string Format: Voice name for one of Google's TTS voices Determines the voice used to generate TTS audio. Must correspond with the default language setting.			
Output Audio Voice Gender	v2	ccaiv2	ccaiv2b1
JSON Pointer: /outputAudioConfig/synthesizeSpeechConfig/voice/ssmlGender Type: enum Valid values: "SSML_VOICE_GENDER_MALE" – Uses a male voice. "SSML_VOICE_GENDER_FEMALE" – Uses a female voice. "SSML_VOICE_GENDER_NEUTRAL" – Uses a gender-neutral voice. Sets the preferred gender of the voice used to generated TTS audio. Only applied if the "outputAudioConfig.synthesizeSpeechConfig.voice.name" and "outputAudioConfig.synthesizeSpeechConfig.voice.customVoice.model" parameters aren't set and the specified gender is available in the selected language.			
Output Audio Custom Voice Model			ccaiv2b1
JSON Pointer: /outputAudioConfig/synthesizeSpeechConfig/voice/customVoice/model Type: string Format: Name of an AutoML model Sets the custom voice model used to generate TTS audio.			
Output Audio Custom Voice Usage			ccaiv2b1
JSON Pointer: /outputAudioConfig/synthesizeSpeechConfig/voice/customVoice/reportedUsage Type: enum Valid values: "REALTIME" – TTS audio is only played once. "OFFLINE" – TTS audio is stored and can be reused. Specifies the usage of the TTS audio for the custom voice.			

3.6.3 **Raw parameters**

The Raw Parameters field can specify additional JSON fields for settings that don't have UI elements in the VAD Parameters or Vendor Parameters groups.

3.7 Launch Dialogflow application

With all the above configuration in place, Google Dialogflow is now ready to use. Simply dial the application launch number and you should hear the Welcome Intent response from your Dialogflow bot.

3.8 Refreshing Dialogflow credentials using REST API

Google recommends Dialogflow customers to change the credentials periodically. Avaya Experience Portal provides a new REST API to change Dialogflow Speech Server credentials.

Use this procedure to rotate the credentials through the new REST API.

Before you begin

Ensure the following:

- Customers follow the Avaya Experience Portal REST documentation to prepare the environment.

Procedure

1. Request new Service Account Key files (JSON) from the Avaya NPI Operations team.

2. Go to the following URL to complete the REST authorization process:
   https://BaseURL/EPWebServices/rest/management/asrservers/credentials/<asrName>
   Where,
   - BaseURL is the default IP address of your main EPM.
   - <asrName> ASR server for which you want to replace the keys.

3. Send a PUT call request to the above address with the json request body.
   For example,


```
{
"credentials": "{\"type\":\"service_account\",\"project_id\":\"aaep-sample-
bot25sv\",\"private_key_id\":\"36b87603a5abd6764463422c20adfbc43efeb564\",\"private_key\":\"-----BEGIN PRIVATE KEY-----
\\nKEYContent\n-----END PRIVATE KEY-----\\n\",\"client_email\":\"dialogflow-vcotni@aaep-sample-bot-
sv.iam.gserviceaccount.com\",\"client_id\":\"107673866867627177364\",\"auth_uri\":\"https://accounts.google.com/o/oauth2/auth\
",\"token_uri\":\"https://oauth2.googleapis.com/token\",\"auth_provider_x509_cert_url\":\"https://www.googleapis.com/oauth2/v1/c
erts\",\"client_x509_cert_url\":\"https://www.googleapis.com/robot/v1/metadata/x509/dialogflow-vcotni%40aaep-sample-bot-
sv.iam.gserviceaccount.com\"}",
"credentialFileName": "test2.json"
}
```


The json request body should contain the following mandatory parameters:

- *credentials*: Should contain the whole json credential file content downloaded from Google.
  ***CAVEAT*:** Ensure that credentials are escaped so that it can be treated as a string value in json format (Replace " with \", newline with \n, backslash with \\ etc) (see example)
- *credentialFileName*: Any string. This is used to identify the key
  .

4. At the prompt, enter your user name and password for authentication.
   You must use the credentials that you use to login to the EPM.

Result

On successful completion, customers receive a 200 Response containing the request body. If the process fails, customers receive a 400 or 500 response containing the error message used for debugging.

5 Experience Portal - Dialogflow Interaction Definitions

This section details all the supported interactions between AEP and Dialogflow. AEP and Dialogflow use both defined events and defined Dialogflow custom payloads to communicate for AEP to pass information back to Dialogflow bot and for the Dialogflow bot to instruct AEP to carry out various tasks. Each interaction is called out in the following sub-sections.

5.1 Welcome Event Sent from AEP to Dialogflow

When the AEP **def_dialogflow.vxml** applications starts, it sends a "**Welcome**" event to the Dialogflow bot. The Dialogflow bot must define an intent to trigger on this event. EP sends two Dialogflow contexts to Dialogflow: "**sip-hdrs"** and "**avaya-session-telephone**". These contexts are accessible by the Dialogflow Welcome intent.

```
"event_input": {
    "name": "Welcome",
    "contexts": [
        {
            "name": "sip-hdrs",
            "lifespanCount": 1,
            "parameters": {...}
        },
        {
            "name": "avaya-session-telephone",
            "lifespanCount": 1,
            "parameters": {...}
        }
    ]
}
```

When a CX agent is configured with the CCAI v2beta1 provider setting, the platform will instead directly trigger the Default welcome intent for the agent using the StreamingAnaylzeContent API method, rather than sending a generic "Welcome" event. This allows for the partial response feature to be used for the initial prompt. However, the generated "inputIntent" request will still contain the two Dialogflow contexts detailed below.

5.1.1 sip-hdrs Dialogflow context

As part of the Welcome event, AEP includes a sip-hdrs Dialogflow context. This context contains all the SIP headers available for the VXML session. The sip-hdrs context must be added as a Input context in the Dialogflow intent. The contents of the context parameters will vary based on the actual headers received in the SIP INVITE request.

sip-hdrs context parameters can be accessed in Dialogflow BOT but must be added as an input context in the Dialogflow intent.

The example below shows a response in the Welcome Intent that accesses the sip-hdrs context:

*"Hi #**sip-hdrs.from.displayname** the phone you called from is: #**sip-hdrs.from.user**"*

This will say:

Hi PaulMc the phone you called from is 8140971

The JSON structure below is an example sip-hdrs Dialogflow context:

```
{
    "name":"sip-hdrs","lifespanCount":1,
    "parameters": {
        "from": {
            "displayname":"\"PaulMc\"",
            "host": "sipccgal.com",
            "tag": "3ffdc4b8555541e98158050568f34fb",
            "uri": "sip:8140971@sipccgal.com",
            "user": "8140971"
        },
        "to": {
            "displayname":"",
            "host":"sipccgal.com",
            "uri":"sip:2141280@sipccgal.com",
            "user":"2141280"
        },
        "unknownhdr": [
            { "name": "Max-Breadth", "value": "60" },
            { "name": "User-to-User", "value":
"00FA08000E04E35CA37458;encoding=hex" },
            { "name": "P-AV-Message-Id","value":" 1_1" },
            { "name": "Av-Global-Session-ID","value": "3ffdbbee-5555-41e9-8156-
0050568f34fb" },
            { "name": "P-Location", "value": "SM;origlocname=\"ContactCenter\"" }
        ],
        "callid": "3ffdc4d6555541e98159050568f34fb",
        "requestmethod": "INVITE",
        "requesturi": "sip:2141280@sipccgal.com",
        "requestversion": "SIP/2.0",
        "require": "",
        "supported": "100rel histinfo join replaces sdp-anat timer"
    }
}
```


5.1.2 **avaya-session-telephone Dialogflow context**

As part of the Welcome event, AEP includes avaya-session-telephone Dialogflow context. This context contains all the Avaya platform parameters available to the VXML session, including **ANI**, **DNIS**, **call tag**, **channel**, **AAI** and **Session ID**.

avaya-session-telephone context parameters can be accessed in Dialogflow BOT but must be added as an input context in the Dialogflow intent.

The example below shows a response in the Welcome Intent that accesses the avaya-session-telephone context:

*The number you called is: **#avaya-session-telephone.dnis***

This will say:
The number you called is 2141280

The JSON structure below is an example avaya-session-telephone Dialogflow context:

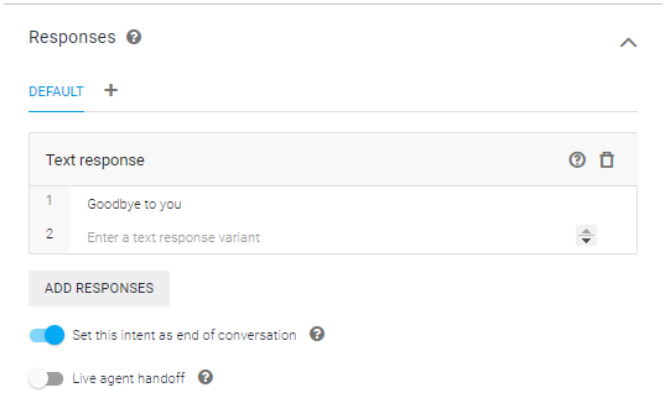

```
{
    "name": "avaya-session-telephone",
    "lifespanCount": 1,
    "parameters": {
        "aai": "00FA08000E04E35CA37458;encoding=hex",
```

```
        "ani": "8140971",
        "call_tag": "mpp248-2019092144217-11",
        "called_extension": "",
        "callid": "mpp248-EPMPP224SM-1-2019092144217",
        "ccxml": {
            "namelist": ["early_media"],
            "values": { "early_media" :"false" }
        },
        "channel": "11",
        "coverage_reason": "",
        "coverage_type": "",
        "dnis": "2141280",
        "early_media": "false",
        "iidigits": "",
        "rdnis": "",
        "startPage":
"http://xxx/mpp/misc/dialogflowapp/def_dialogflow.vxml?session___sessionid=",
        "teletype": "SIP",
        "uui": "00FA08000E04E35CA37458;encoding=hex"
    }
}
```


5.2 AEP detection of End of Conversation signalled by Dialogflow bot

In Dialogflow ES agents, an intent can be set as the “end of conversation”. This is set in the Intent Responses section by enabling “**Set this intent as end of conversation**” as shown in Figure 5.2:

Figure 5.2: Dialogflow Intent screenshot showing “end of conversation” field



The AEP def_dialogflow.vxml application checks responses for an “**intent”** object with field “**endInteraction**” set to true. AEP plays any response and disconnects the call.

For CX Agents the def_dialogflow.vxml application will detect the “**endInteraction**” response message automatically sent when the agent reaches the **END_SESSION** page, playing any response and disconnecting the call.

5.3 Hangup Event Sent from AEP to Dialogflow

When the AEP detects the customer call has disconnected, an event named "**Hangup**" is sent to the Dialogflow bot. The Dialogflow bot should handle this event to perform any post call steps when the caller hangs up.

This event could be used if there is there is a need to perform further actions to create CRM cases, wrap up call information etc.

```
"event_input":{
    "name":"Hangup"
}
```

In order for AEP to send a hangup event the following JSON needs to be added or part of the "Raw Parameters" JSON

```
{ "sendHangupEvent": true }
```

5.4 Dialpad entered DTMF collection using telephony_read_dtmf custom payload

Dialogflow can instruct AEP to use local DTMF detection and detect digits entered using a dial pad. The Dialogflow agent should send a **telephony_read_dtmf** custom payload and the **def_dialogflow.vxml** application will enable a local DTMF grammar. The **listen_to_speech** parameter is used to determine if speech is passed to Dialogflow to potentially collect DTMF by voice.

AEP will send back the collected DTMF using the TELEPHONY_DTMF event.

The telephony_read_dtmf custom payload is defined as follows:

```
"telephony_read_dtmf": {
    "max_duration": "10s",
    "max_digits": 5,
    "listen_to_speech": false,
    "finish_digit": "DTMF_STAR"
}
```

The following table lists all the fields supported in the telephony_read_dtmf payload:

Custom Payload transfer field	Description (See VXML specification for full definition)
max_digits	VXML **maxlength** parameter in DTMF grammar. Determines maximum number of digits to collect.
listen_to_speech	**true**: DTMF can be spoken. **false**: Speech ignored.
max_duration	Ignored as no equivalent VXML value exists
finish_digit	VXML **termchar** property name. Values shown in table below

The table below shows all values of finish_digit and the mapping to termchar value

finish_digit value	termchar property value
DTMF_STAR	*
DTMF_POUND	#
DTMF_ONE	1

DTMF_TWO	2
DTMF_THREE	3
DTMF_FOUR	4
DTMF_FIVE	5
DTMF_SIX	6
DTMF_SEVEN	7
DTMF_EIGHT	8
DTMF_NINE	9
DTMF_A	A
DTMF_B	B
DTMF_C	C
DTMF_D	D

5.5 TELEPHONY_DTMF event to pass back collected DTMF to bot

When AEP has collected DTMF for the Dialogflow bot, it uses the TELEPHONY_DTMF event to send the DTMF back to the bot.

```
"event_input": {
    "name": "TELEPHONY_DTMF",
    "parameters": {
        "telephony_dtmf_digits": "590227"
    }
}
```

5.6 Barge-in Control

Dialogflow can control whether the audio response from an intent being played by AEP, can be interrupted. For ES agents this is accomplished by sending a custom payload to AEP with a flag to instruct the AEP to allow barge-in of an audio response being played. AEP def_dialogflow.vxml does this by setting the VXML bargein property for the current audio response. The custom payload is named avaya_telephony and has a field named “bargein” which is set to false or true.

```
"avaya_telephony": {
    "bargein": false
}
```

CX agents can control the same flag without using custom payloads by simply configuring the appropriate “Enable barge-in” toggles in the Advanced speech settings.

5.6.1 Bargein usage in AEP Dialogflow

In VXML the bargein property normally defaults to enabled. If you do not set the bargein property in VXML, the caller will be able to barge in (voice or DTMF) and interrupt the audio. However, for Dialogflow, due to the noise/echo cancellation being calculated at the start of the call, it was decided that barge in would default to disabled for the first response (i.e. Welcome message), unless overridden using the bargein flag. If the user experiences any issues with barge in, it is recommended to disable it on a per intent/page basis.

5.7 VXML privacy feature control

Dialogflow Agents can enable privacy feature support. There are two privacy options available.

5.7.1 Intent Privacy

This is accomplished by the Dialogflow Agent sending a custom payload to AEP to set the private property. If the private property is enabled AEP does not write any speech recognition results, DTMF results, or TTS strings into the session transcription logs or into any alarms that may be generated during the processing of the next recognition response.

AEP def_dialogflow.vxml does this by setting the VXML private property when processing the current response. The custom payload is named avaya_telephony and has a field named “private” which is set to false or true.

```
"avaya_telephony": {
    "private": true
}
```

5.7.2 Application Privacy (recommended)

This is accomplished by the Dialogflow Agent sending a custom payload to AEP to set the applicationPrivate property. The applicationPrivate property allows privacy to be enabled at the start of the Dialogflow Agent. Once enabled, privacy will persist for the conversation with the Dialogflow Agent - unless application privacy is subsequently disabled in the Dialogflow Agent.

Once the applicationPrivate property is enabled AEP does not write any speech recognition results, DTMF results, or TTS strings into the session transcription logs or into any alarms that may be generated during the conversation with the Dialogflow Agent.

AEP def_dialogflow.vxml does this by caching the value of the applicationPrivate custom payload when processing each response from the Dialogflow Agent. This cached value is then used to set the VXML private property for all subsequent recognition iterations with the Dialogflow Agent. The custom payload is named avaya_telephony and has a field named “applicationPrivate” which is set to false or true.

```
"avaya_telephony": {
    "applicationPrivate": true
}
```

5.8 Playing of Pre-Recorded prompts

Dialogflow can instruct the AEP to override the generated audio response and play one or an array of supported audio files by specifying the URI(s) in the avaya_telephony custom payload. The URI must be accessible by the AEP MPP server. The **def_dialogflow.vxml** application will inspect the custom payload in a recognition response and use the specified URIs in place of the returned reply audio when playing the next prompt. The custom payload uses the “avaya_telephony” object with a “reply_audio_uri” field like one of the following examples:

```
"avaya_telephony": {
    "reply_audio_uri": "http://1.2.3.4/prompts/HelloRoomBookingAgent.wav"
}
OR
"avaya_telephony": {
    "reply_audio_uri": [
        "http://1.2.3.4/prompts/HelloRoomBookingAgent.wav",
        "http://1.2.3.4/prompts/HowCanIHelp.wav",

         "file:///opt/Avaya/ExperiencePortal/MPP/web/misc/dialogflowapp/prompts/t
         est/Emergency.wav"
    ]
}
```

CX agents can also simply use one or more Play pre-recorded audio dialogue options in the fulfilment instead of using the custom payload.

5.9 DTMF Transfer (Play DTMF into call for Feature Access Code Transfer)

Dialogflow can instruct the AEP to play DTMF into a call. This might be used for DTMF assist or playing DTMF to initiate a transfer. The senddigit builtin URI can be used in the "**reply_audio_uri**" parameters from the pre-recorded prompts feature to play DTMF digits. The digits specified in the URI will be played out in the appropriate order. This uses the **avaya_telephony** custom payload"

```
"avaya_telephony": {
    "reply_audio_uri": [
        "http://1.2.3.4/prompts/TransferToAgent.wav",
        "builtin://senddigit/*991611"
    ]
}
```

5.10 VXML Transfer

Dialogflow can instruct the AEP to perform a VXML transfer using **avaya_telephony** custom payload. The **def_dialogflow.vxml** application will inspect the payload for the "transfer" object. It will then perform a transfer mapping the transfer field values to attributes of the VXML transfer tag. The custom payload is defined as:

```
"avaya_telephony": {
    "transfer": {
        "dest": "tel:1234",
        "type": "consultation",
        "transferaudio": "http://1.2.3.4/prompts/music.wav",
        "connecttimeout": "30s",
        "maxtime": "600s",
        "aai": "FA,2901000246DEE275",
        "sip_headers": {
            "unknownhdr[0].name": "Random",
            "unknownhdr[0].value": "This is an unknown header"
        }
    }
}
```

The following table lists all the fields supported in the transfer object of avaya_telephony:

Custom Payload transfer field	Description (See VXML specification for full definition)
dest	**Mandatory**: Valid **tel:** or **sip/s:** URI specifying destination of call transfer
type	**Optional**: Specifies the transfer type to use and must be one of: "**bridge**": Connection with Dialogflow maintained even after transfer disconnects. "**blind**": Connection with Dialogflow dropped regardless of transfer outcome "**consultation**": Connection with Dialogflow is maintained until transfer is successful.
connecttimeout	**Optional**: Bridge/Consultation only: The time to wait while trying to connect the call before returning.
maxtime	**Optional**: Bridge only: The maximum time the call is connected.
transferaudio	**Optional** field: Bridge/Consultation only: The time to wait while trying to connect the call before returning.
aai	**Optional**: String contain Application to Application (aai) date to be sent to far end application
sip_headers	**Optional**: Object that consists of name/value pairs using VXML Property AVAYA_SIPHEADER. The AEP documentation should be consulted for details on supported header names.

5.10.1 Support for reporting Failed or Completed Bridge Transfers

When control for a failed or completed bridged transfer is returned to the **default_dialogflow.vxml** application an event is sent to the Dialogflow bot to allow a follow-up intent to recover the conversation.
This event will include a parameter named "**reason**", indicating how the transfer failed or was completed. The events are:

- "**TELEPHONY_XFER_FAILED**": bridge transfer fails.
  reason parameter values for failed event are:
  - busy
  - network_busy
  - noanswer
  - unknown

Example:

```
"event_input": {
```

```
            "name": "TELEPHONY_XFER_FAILED",
            "parameters": {
                "reason": "noanswer"
            }
        }
```

- "**TELEPHONY_XFER_COMPLETE**": bridge transfer completed. reason parameter values for completed event are:
    - near_end_disconnect
    - maxtime_disconnect
    - network_disconnect
    - far_end_disconnect

Example:

```
        "event_input": {
            "name": "TELEPHONY_XFER_COMPLETE",
            "parameters": {
                "reason": "far_end_disconnect"
            }
        }
```

5.11 Timer Support – No Input Timeout and Speech Complete Timeout

Dialogflow can instruct the AEP to control the VXML no input and speech complete timeout. Dialogflow using the avaya_telephony custom payload can change these timers for the next recognition. The custom payload is defined as:

```
"avaya_telephony": {
    "no_input_timeout": "7s",
    "speech_complete_timeout": "1s"
}
```

The settings from this custom payload are only applied when the AEP VAD is enabled. If the AEP VAD is disabled, CX agents can control the no input timeout through the "No speech timeout" field in the Advanced speech settings.

5.11.1 No input timeout

The no_input_timeout field maps to the VXML property: "**timeout**". This specifies the amount of silence time after a prompt is played after which a no input event is thrown in VXML. The default value for timeout VMXL property is 7 seconds. A no input event will trigger the def_dialogflow.vxml application to send a "DF_SYSTEM_NO_INPUT" event to Dialogflow bot.

5.11.2 Speech complete timeout

The speech_complete_timeout maps to the VXML property: "**speechtimeout**". Specifies the period of silence required after user speech to determine that speaker has finished talking. The default value for speechtimeout VXML property is 0.25 seconds.

5.12 Number of Attempts support DF_SYSTEM_NO_INPUT event

When AEP detects no input (silence), it sends an event called: **DF_SYSTEM_NO_INPUT** to the Dialogflow bot. This event can be used by the Dialogflow bot to implement number of attempts functionality in the bot. The implementation of this is outside the realm of AEP Dialogflow integration and will not be detailed here.

5.13 Play Audio until Avaya telephony Follow Up Event Received - Fetch Audio

Dialogflow can instruct the AEP to perform the ability to play an audio WAV file for an unspecified amount of time until the bot sends a follow up intent response. This could be used to play music and carry out activities that take a non-deterministic amount of time. This is achieved by the Dialogflow bot sending a “followup_event” object in the **avaya_telephony** custom payload. The **def_dialogflow.vxml** application will inspect the payload for the “followup_event” object and, if defined will treat the received response as an interim result and proceed by sending the named event (WaitStart in the example below) with any optional parameters specified. It then will wait for the response to that event before continuing. While waiting, the specified audio file will be played. The audio in the response will work as it normally does. That is, if the reply_audio_uri field is also specified the listed prompt URI(s) will be played in order. If the reply_audio_uri field is omitted the generated TTS from the reply_audio field in the response will be played instead.

```
{
    "avaya_telephony": {
        "followup_event": {
            "name": "WaitStart",
            "parameters": {
                "Key1": "Value1",
                "Key2": "Value2",
                "ThirdKey": "Value3"
            }
        },
        "reply_audio_uri": "http://1.2.3.4/prompt/music.wav"
    }
}
```

The following table lists all the fields supported in the followup_event object of avaya_telephony payload:

Custom Payload followup_event field	Description (See VXML specification for full definition)
name	**Mandatory**: Name of the Event to send to dialogflow bot
parameters	**Optional**: A list of key value pairs which are sent with the event to dialogflow bot
reply_audio_uri	**Optional**: This is not a followup_event field but is used to specify the audio file(s) to play while waiting for the response to the event.

5.14 Playing consecutive prompts using Follow Up Event

In traditional IVR systems, a designer can play consecutive prompts without the requirement for user input. In Dialogflow, an intent must always play a prompt and collect user input to move to the next intent. Dialogflow provides consecutive prompts functionality of playing by using a followup_event defined in the first intent which will be sent by AEP to the Dialogflow bot when the response of the first intent audio has finished playing. The second intent triggered by the followup_event can then play another prompt. For example, a flow may want to prompt:

Example:

(Customer dials in)

bot: “Hello, welcome to the contact center what can I do for you?”

(Traditional IVR would switch to another form to send second prompt)

bot: “Would you like to hear our offers today?”

Dialogflow provides a custom payload to provide this functionality. The first intent would set a custom payload as shown:

```
{
    "intent": "FOLLOWUP_EVENT",
    "data": {
        "event_input": {
            "name": "event_name",
            "parameters": {
                "param1": "param1_value",
                "key2": 2
            }
        }
    }
}
```

The parameters in bold can be changed:

event_name: This is the event that triggers the next intent after the response from the first intent is played.

parameters: key value pair list that can be accessed in the followup intent response as shown:

*“Hi, this is the follow up intent response. The parameters passed are **#event_name.param1** and **#event_name.key2**”*

The response in the first intent is played to completion and AEP will only send the followup event (event_name) when this is completed. The bot will then match an intent waiting for this event which will send a response. This response will then be played.

Note: Each prompt can use the custom payloads to control the intent. i.e. you could add custom payload to control bargein of one of the prompts:

```
{
    "avaya_telephony": {
        "bargein": true
    }
}
```

This could be used if a flow wanted one portion of a prompt to not be interrupted (some legal information for example) and then the second part of the prompt could be interrupted (bargein set to true).

This feature can play any number of prompts consecutively where the previous prompt will trigger the next intent using follow up intent to play the next prompt.

This feature differs from the Play Audio until Avaya telephony Follow Up Event - Fetch Audio in that it plays the full response of the first intent before it sends the event to trigger the follow up

intent rather than sending the event before the audio starts playing to allow for music to be played while asynchronous activity takes place.

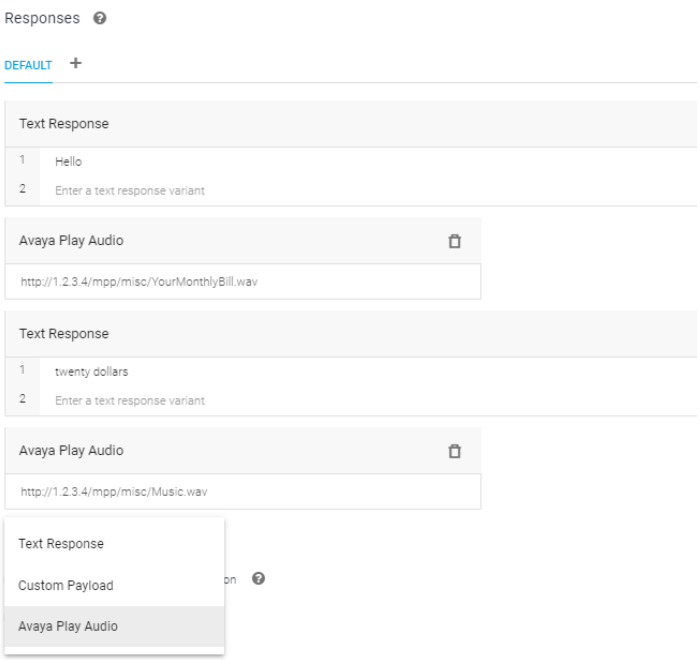
5.15 Playing mix of pre-recorded audio and Text to Speech (TTS)

A feature to allow a bot developer to play a mix of Text to Speech and pre-recorded audio files for intent responses. Note: For ES agents this feature is only available if the project has been allowlisted for Avaya extra features (this is an extra allowlist enablement not covered by the process in section **Error! Reference source not found.**).

Playing wav files for static prompts can provide cost savings over using Google Dialogflow Text to Speech which is charged on a per character basis.

Example: TTS: “Hello” + YourMonthlyBill.wav + TTS: “20 Dollars” + Music.wav

For ES agents with the Avaya extra features, the Dialogflow GUI will provide a mechanism for this in the Intent screen.
In the Responses section TTS and Avaya Play Audio responses can be added in the order played. Click on “Add Response button and select “Text Response” or “Avaya Play Audio” response. The screenshot below shows the example given in this section and the options for the Responses (Text Response, Avaya Play Audio, etc.)



For CX agents, no special allowlisting is necessary as the regular UI allows for fulfilments to specify a series of dialogue options that contain a mix of Text and Play pre-recorded audio dialogue options as needed.

5.16 Multilingual Support

Dialogflow can instruct the AEP to change the language of the bot. This change will affect all intents after it is changed. This is achieved by the Dialogflow bot sending a “**set_language_code**” element in the **avaya_telephony** custom payload. This element is

set to a IETF language tag (en-us, en-gb, es-es, es-419 etc). The full set of supported languages are defined by Dialogflow and Google are adding new languages with every new release.
The **def_dialogflow.vxml** application will inspect the payload for the “set_language_code” element and use this to set the language for the next, and all subsequent recognitions/generated audio.

```
{
    "avaya_telephony": {
        "set_language_code": "es-es"
    }
}
```

5.17 Changing SynthesizeSpeechConfig at Runtime

Dialogflow Agents can instruct AEP to change the OutputAudioConfig SynthesizeSpeechConfig object that is populated in streaming API requests that are sent to Dialogflow.

This is achieved by the Dialogflow Agent sending a “**set_speech_cfg**” element in an **avaya_telephony** custom payload to AEP. This element carries a Dialogflow API SynthesizeSpeechConfig object.

```
{
    "avaya_telephony": {
        "set_speech_cfg": {
            "pitch": 0.0,
            "volumeGainDb": 0.0,
            "effectsProfileId": [
                "telephony-class-application"
            ],
            "voice": {
                "name": "en-US-Standard-C",
                "ssmlGender": "SSML_VOICE_GENDER_FEMALE"
            }
        }
    }
}
```

This payload overrides the values for SynthesizeSpeechConfig that were set in Vendor Parameters in Dialogflow Application or in the Configuration Param.

The value set in the “**set_speech_cfg**” element will persist for all subsequent streaming API requests in the conversation with the Dialogflow Agent (until it is changed).

The “**set_speech_cfg**” element can change the Google TTS voice that is used by the Dialogflow Agent at runtime. This element can be useful when used in conjunction with the “**set_language_code**” element to change the language and TTS voice for multilingual Dialogflow Agents at runtime.

5.18 Partial response Support

Dialogflow CX agents can return a partial response for long running webhook functions to allow one or more interim responses to be played before a final response is finally played

after the long running webhook completes. The **def_dialogflow.vxml** application automatically enables partial responses by signalling support in any streaming API requests, setting the enablePartialAutomatedAgentReply flag to true. Accordingly, any partial responses returned to the **def_dialogflow.vxml** application are handled by playing the returned audio to the caller and waiting for further interim responses or a final response. Speech input is disabled during partial response playback (as the previous utterance is still being processed) and once the final response is received, speech input enablement is controlled through the bargein settings in the response.

6 Experience Portal Voice Activity Detector

A Voice Activity Detector (VAD) allows AEP to detect the presence or absence of human speech, i.e. If sound is background noise or actual speech (speech energy).

VADs are used to determine when to send the audio stream to Cloud Speech Resources (Dialogflow, Google Speech or NuanceDaaS – Nuance Dialog as a Service). When the AEP VAD detects what it determines as speech energy, AEP will only then send audio stream to the Cloud Speech resource.

A properly tuned VAD reduces the incidence of announcements being falsely interrupted when barge in is enabled.

AEP VAD configuration also allows for the ability to enable and disable the local VAD and use the Cloud Speech Resource instead:
AEP VAD configuration also allows for:

1. Ability to disable the local VAD and to immediately stream all audio to the cloud speech resource (Dialogflow, Google Speech or NuanceDaaS) so as to use the clouds own VAD.

2. Hybrid option where the local VAD is used to detect start of speech, but the cloud speech resource is used for the rest of the utterance.

CAVEAT: Disabling AEP local VAD or using Hybrid option will lead to **increased network usage** as the audio is streamed immediately to the cloud speech resource. It will also lead to **increased charges** from the cloud speech resource provider as you are using the cloud speech resource VAD.

AEP implements the following VAD Algorithms:

- **Gaussian Mixture Model** - (GMM)
- **Inqa** - Speech detection using Call Progress Engine
- **MeanSquare** - Original VAD algorithm (deprecated VAD)

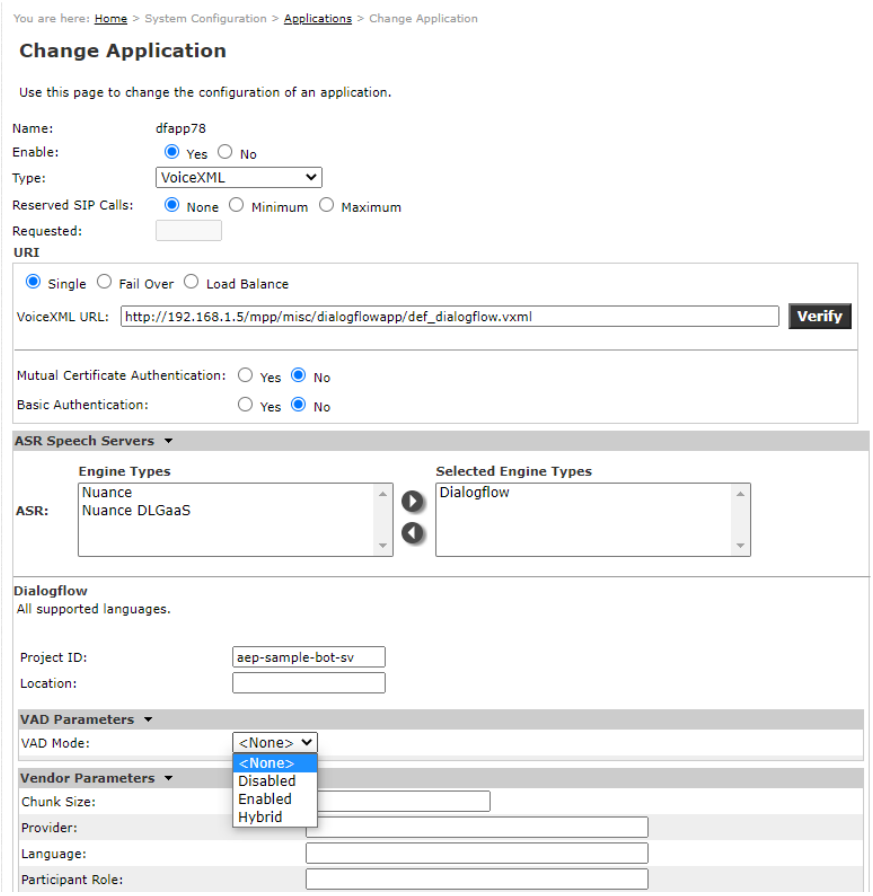
6.1 VAD Configuration

VAD configuration consists of both VAD selection and VAD tuning. VAD configuration can be applied on a global system wide basis or on a per application basis.

6.1.1 Application Specific VAD Configuration

AEP allows for separate VAD configuration for each application. This overrides the Global system wide VAD configuration for this application.

When configuring an application for Cloud Speech Resource Provider (Dialogflow, Google Speech or NuanceDaaS ASR) you will be presented with the ability to configure the VAD Parameters for this application. The screenshot below show Application configured for Dialogflow:



6.1.1.1 VAD Mode

VAD Mode determines whether local VAD is used, cloud speech provider's VAD is used or a combination of both are used. VAD mode values are:

- **<None>** - AEP Local VAD is enabled and uses default GMM VAD or system-wide VAD Type specified in mppconfig.xml

- **Disabled** - AEP local VAD is disabled. Audio stream is sent to cloud provider Immediately.

- **Enabled** - AEP local VAD is enabled. Local VAD uses selected VAD type algorithm

to detect speech energy. Once speech energy is detected, AEP will then send Audio stream to cloud provider.

- **Hybrid** - AEP local VAD is enabled for initial response in every call but is disabled for all other utterances for the rest of the call. This means that local VAD will be used first and then cloud provider VAD will be used for the rest of the call.

6.1.1.2 **VAD Type**

VAD Type determines which local VAD algorithm is used. VAD Type values are:

- **<None>** - Use default GMM VAD or system-wide VAD specified in mppconfig.xml
- **GMM** - Gausian Mixture Model VAD algorithm used.
- **Inqa** - Inqa Call Progress Engine VAD Algorithm used.

6.1.1.3 **GMM VAD Tuning Parameters**

When GMM VAD is selected the following GMM VAD tuning parameters are available to configure:

6.1.1.3.1 **Starting Minimum Energy**

Minimum energy threshold required for frame analysis during the barge-in phase of speech input. Enter a value in the range 0 to 16,383. Default value is 2048.

6.1.1.3.2 **Starting Minimum Energy Scale**

Power of two scale factor for the minimum energy threshold during the barge-in phase of speech input. Enter a value in the range 0 to 32. Default value is 6.

6.1.1.3.3 **Starting Minimum Energy Frames**

Minimum number of consecutive frames with speech energy required to trigger the VAD during the barge-in phase of speech input. Enter a value in the range 1 to 100. Default value is 3.

6.1.1.3.4 **Interim Minimum Energy**

Minimum energy threshold required for frame analysis after interruptible prompts are finished playing. Enter a value in the range 0 to 16,383. Default value is 2048.

6.1.1.3.5 **Interim Minimum Energy Scale**

Power of two scale factor for the minimum energy threshold after interruptible prompts are finished playing. Enter a value in the range 0 to 32. Default value is 6.

6.1.1.3.6 **Interim Minimum Energy Frames**

Minimum number of consecutive frames with speech energy required to trigger the VAD after interruptible prompts are finished playing. Enter a value in the range 1 to 100. Default value is 3.

6.1.1.4 **Inqa VAD Tuning Parameters**

When Inqa VAD is selected the following Inqa VAD tuning parameters are available to configure:

6.1.1.4.1 **Starting Noise Floor**

Minimum value for adaptive energy threshold during the barge-in phase of speech input. Enter a value in the range 0 to 100. Default value is 22.0.

6.1.1.4.2 **Starting Noise Offset**

Offset value, relative to the adaptive energy threshold, required for frame analysis during the barge-in phase of speech input. Enter a value in the range 0 to 100. Default value is 1.0.

6.1.1.4.3 **Interim Noise Floor**

Minimum value for adaptive energy threshold after interruptible prompts are finished playing. Enter a value in the range 0 to 100. Default value is 22.0

6.1.1.4.4 **Interim Noise Offset**

Offset value, relative to the adaptive energy threshold, required for frame analysis after interruptible prompts are finished playing. Enter a value in the range 0 to 100. Default value is 1.0.

6.1.2 **Global System wide VAD Configuration**

Configuration parameters can be added to the **$MPP/config/mppconfig.xml** file to change both global system wide VAD selection and VAD tuning.

Note: As editing the **mppconfig.xml** file is required to change global VAD configuration, then this file needs to be modified on all MPPs in the solution.

6.1.2.1 **Global** system wide **VAD Selection**

A new configuration parameter has been added to allow selection of the VAD type.
The following parameter can be added to **$MPP/config/mppconfig.xml** to change the VAD:

```
<parameter name="mpp.voip.cloud.vad.type">GMM</parameter>
```

Valid settings:

- GMM – Gaussian mixture model, new VAD (default setting)
- Inqa – Speech detection using the call progress engine
- MeanSquare – Old VAD

6.1.2.2 **Global system wide VAD Tuning for GMM VAD**

The new GMM VAD has been tuned by default and changing tuning parameters is not advised.

New configuration parameters have been added to AEP to manually tune the new GMM VAD on a system wide basis. These are configured in file: **$MPP/config/mppconfig.xml**

6.1.2.2.1 **GMM VAD Aggressiveness**

Sets system wide default aggressiveness for energy rejection.

<parameter name="mpp.voip.cloud.vad.aggressiveness">3</parameter>

Valid settings:

- "0" – very relaxed, minimal thresholds for classifying energy as speech
- "1" – relaxed, energy moderately matching speech profile is accepted
- "2" – aggressive, energy nearly matching speech profile is accepted
- "3" – very aggressive, energy is only classified as speech if it closely matches the calculated speech profile (default setting)

6.1.2.2.2 **GMM VAD Minimum Energy**

This controls systemwide default for minimum energy required for frame analysis, specified with 14 significant bits and a power of two scaling factor

<parameter name="mpp.voip.cloud.vad.min_energy">2048</parameter>

<parameter name="mpp.voip.cloud.vad.min_energy_scl">10</parameter>

Valid settings:

- min_energy – values from 0 to 16,383 (default value 2048)
- min_energy_scl – values from 0 to 32 (default value 10)

Note: See PSN005581u for Additional information related to optimal values.

6.1.2.2.3 **GMM VAD Minimum Speech Frames**

Controls system wide default for the minimum number of consecutive speech frames required to trigger GMM VAD

<parameter name="mpp.voip.cloud.vad.min_speech_frames">5</parameter>

Valid settings:

- Values from 1 to 20 (default value of 5)

6.1.2.2.4 **GMM VAD Initial Minimum Energy**

Determines system wide default for minimum energy required for frame analysis, specified with 14 significant bits and a power of two scaling factor, during the barge-in phase of speech input (i.e. while speech input is collected, but prompts are still playing). This allows the VAD to be tuned to be less sensitive to audio while prompts are playing.

<parameter name="mpp.voip.cloud.vad.init_min_energy">2048</parameter>

<parameter name="mpp.voip.cloud.vad.init_min_energy_scl">12</parameter>

Valid settings:

- init_min_energy – values from 0 to 16,383 (default value 2048)
- init_min_energy_scl – values from 0 to 32 (default value 12)

6.1.2.2.5 **GMM VAD Initial Minimum Speech Frames**

Determines system wide default for the minimum number of consecutive speech frames required to trigger VAD, during the barge-in phase of speech input (i.e. while speech input is collected, but prompts are still playing).

<parameter name="mpp.voip.cloud.vad.init_min_speech_frames">5</parameter>

Valid settings:

- Values from 1 to 20 (default value of 5)