Avaya Aura® Experience Portal 7.2.3
Google Dialogflow Native Support White Paper

Issue 1.08

Date: 16th December 2021

Abstract

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

Contents

Overview

The Avaya Aura® 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. AAEP streams audio from a caller to Google Dialogflow via Googles 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 is currently Beta. This API defines the interaction between Experience Portal and Dialogflow and is accessible via https://dialogflow.com/ .

Note: This API is currently restricted and requires both your google account and Dialogflow Agent (bot) to be **"enable listed"** by Google to access these APIs. See section:

1.1. AAEP default VXML Dialogflow Application

AAEP 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 AAEP (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.

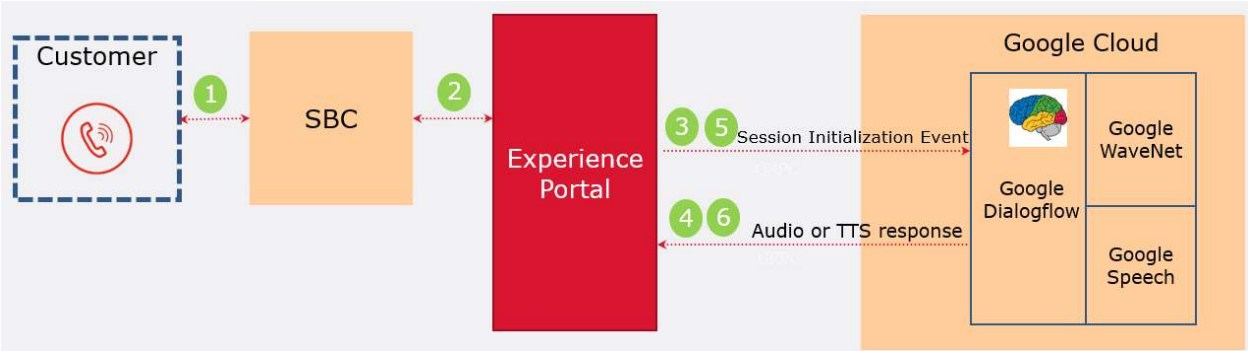
1.1.2. **Default VXML Dialogflow Application directory replaced during upgrades**

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. AAEP 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 AAEP 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:

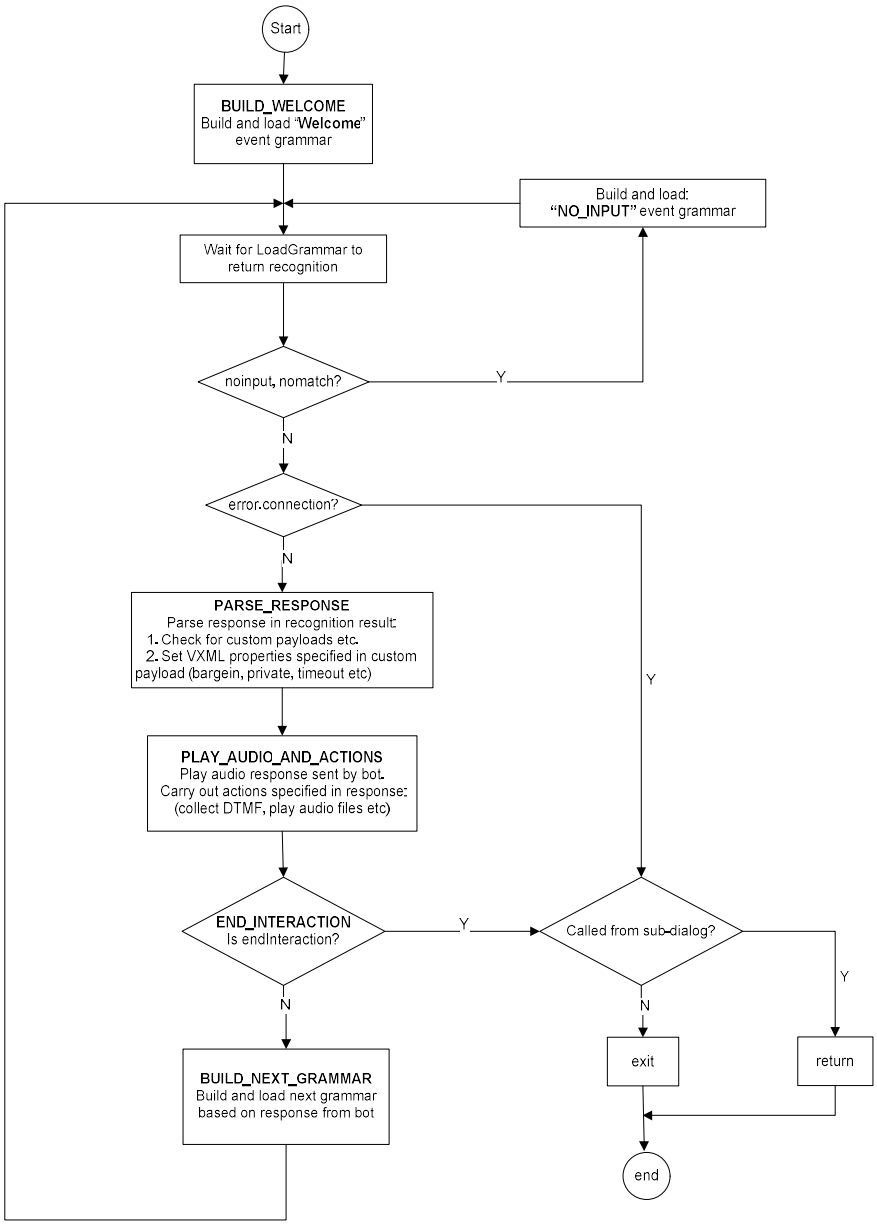
```
"event_input":{
    "name":"TELEPHONY_DTMF",
    "parameters":{
        "telephony_dtmf_digits":"590227"
    }
}
```

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 AAEP and DialogFlow are terminated when:

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

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 AAEP.

2. Experience Portal Dialogflow Configuration

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

2.1. AAEP 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

2.1.1. **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.

2.2. Configure Faster Detection of Dead TCP Connections

AAEP 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 AAEP 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.

2.3. 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.

2.3.1. **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"
```

2.3.2. 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>
```

2.4. Google Dialogflow licensing

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

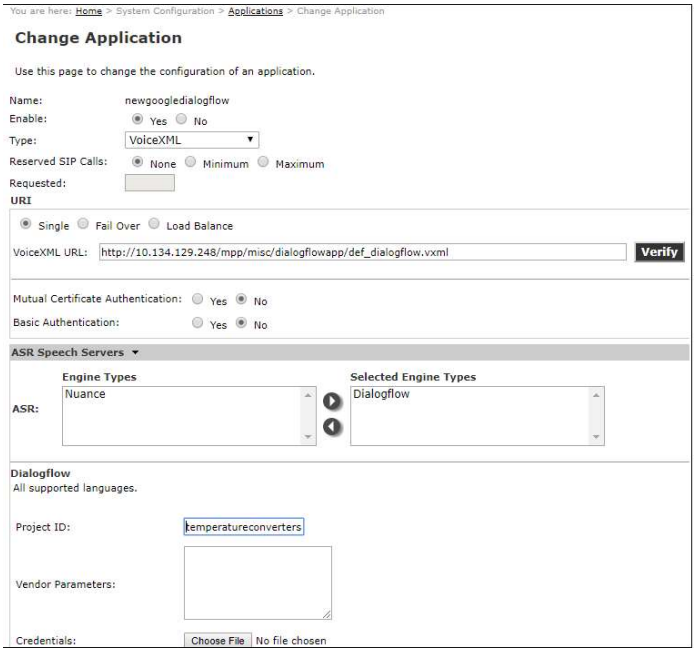
```
<Feature type="counted">
<Name> GoogleDialogflowConnections</Name>
<DisplayName>Maximum number of Google Dialogflow Connections</DisplayName>
<Value>100</Value>
</Feature>
```

2.5. 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:

2.6. 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://xx.xx.xx.xx/mpp/misc/dialogflowapp/def_dialogflow.vxml
5. Set the Project ID to the name of the Google Dialogflow project bot.
6. The Default VXML application has canned wav files to play for error conditions so a TTS server is not required.
7. Enter calling application launch number.
8. 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.



2.6.1. **Vendor Parameters in Dialogflow Application**

The Vendor Parameters field can be filled with any optional json parameters. The following website details all of the parameters that can be set using the json parameters:

OutputAudioConfig settings:
https://cloud.google.com/dialogflow/docs/reference/rpc/google.cloud.dialogflow.v2beta1#google.cloud.dialogflow.v2beta1.OutputAudioConfig

InputAudioConfig settings:
https://cloud.google.com/dialogflow/docs/reference/rpc/google.cloud.dialogflow.v2beta1#google.cloud.dialogflow.v2beta1.InputAudioConfig

QueryParams settings:
https://cloud.google.com/dialogflow/docs/reference/rpc/google.cloud.dialogflow.v2beta1#google.cloud.dialogflow.v2beta1.QueryParameters

Examples of json parameters are shown below:

1. Set the default voice talent (OutputAudioConfig) and pitch etc:

```
{ "synth_speech_cfg":
        { "voice": { "name": "en-US-Wavenet-F" },
          “speaking_rate”:2,
          “pitch”:2.0,
          “volume_gain_db”:-6.0
        }
 }
```

   Note: go to https://cloud.google.com/text-to-speech/ to get list of language/voices

2. Set InputAudioConfig model and model variant (Note: Google only supports phone_call model for the en-us locale. These are default values on Experience Portal if en-us locale is specified so should not need to be changed:

{ “model”:”phone_call”,
“modelVariant”:”USE_ENHANCED”
}

3. Set the default language:
{ "language": "en-gb" }

2.7. 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.

2.8. Refreshing Dialogflow credentials using REST API

Google recommends Dialogflow customers to change the credentials periodically. Avaya Aura 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 Aura® Experience Portal REST docs to prepare the environment.
- Customers have an administrator account for Google cloud.

Procedure

1. Generate a new key JSON file from Google Cloud control panel.

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/certs\",\"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.

3. Configure Google Dialogflow

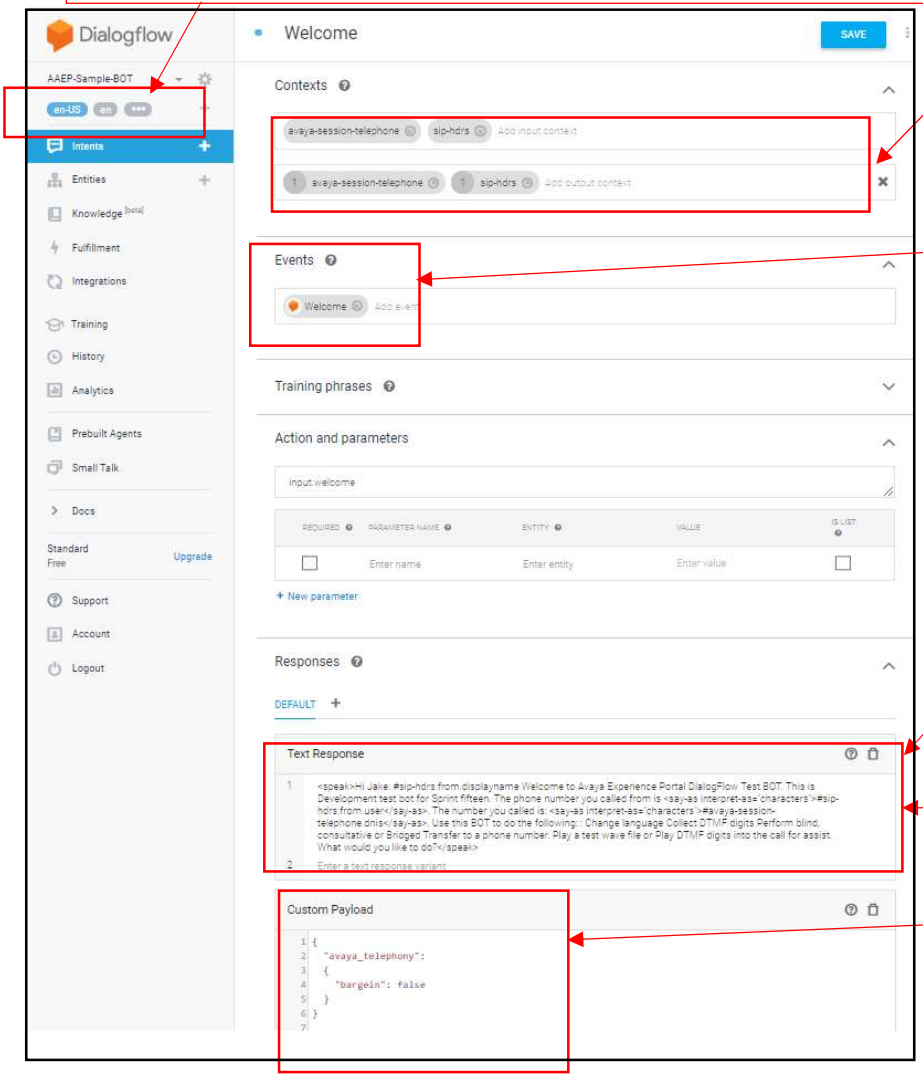
Experience Portal Dialogflow integration with the Google CC AI API set (Google Dialogflow) requires the development of a Google Dialogflow "bot". Google have Tutorials, documentation and examples available at: https://dialogflow.com.

3.1. Google Dialogflow Virtual Agent (bot)

Dialogflow “bots” are developed in Google’s Dialogflow console: https://console.dialogflow.com.

Experience Portal Dialogflow starts by sending a “Welcome” event to the Google “bot”. This Welcome Event message can also contain the SIP Header information of the incoming call and other Avaya Telephony parameters in the context. The “bot” then sends a response with audio content for Experience Portal to play to the customer caller. This response can also include predefined custom payloads which are specified in the Dialogflow intent. These custom payloads are used to pass data between Dialogflow and AAEP. An example of a Welcome intent is shown below along with notes on the relevant fields:

Languages/Regions: Languages/Regions for Intent. This matches the language passed by AAEP in the responses. AAEP 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 AAEP. AAEP plays this audio

Custom Payload: disabling barge in. This is sent to AAEP in the response. AAEP 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.

3.2. Enablelist the bot Application

Avaya acts as a Google Telephony Partner. This means Avaya is a reseller of Google CCAI capabilities including Dialogflow. When customer contracts with Avaya for the Google Dialogflow consumption Avaya will take care of the necessary allow listing process with Google. This is the default mechanism for activation.

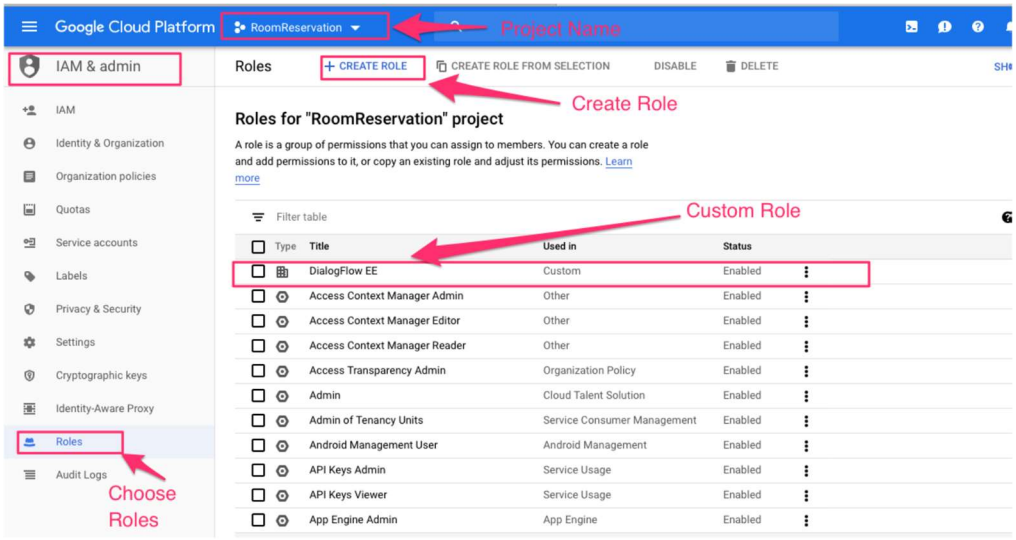
There are certain exceptions where as agreed by Avaya and Google and under the Advanced Agent Model (AAM) the customer will contract CCAI directly with Google in which case the allow listing will be provided by Google using the following form: https://docs.google.com/forms/d/e/1FAIpQLSe2xMdpg_L2bhvl4EoNTUC4Nctzc3Qlw54uQ1_JNFgr0VhOfg/viewform

3.3. Permissions and Roles in Identity Access Management (IAM)

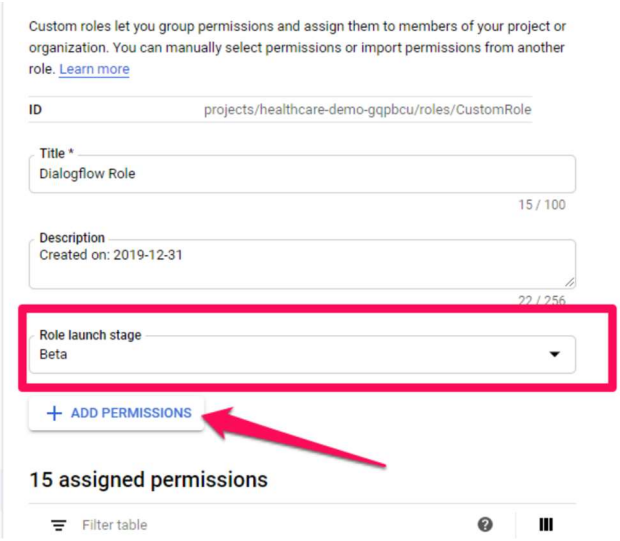
Google Dialogflow bots require IAM roles that allows access to the Dialogflow APIs. Go to http://console.cloud.google.com and follow the procedure below:

3.3.1. Create a custom Role, Assign Permissions and Assign Role to Member

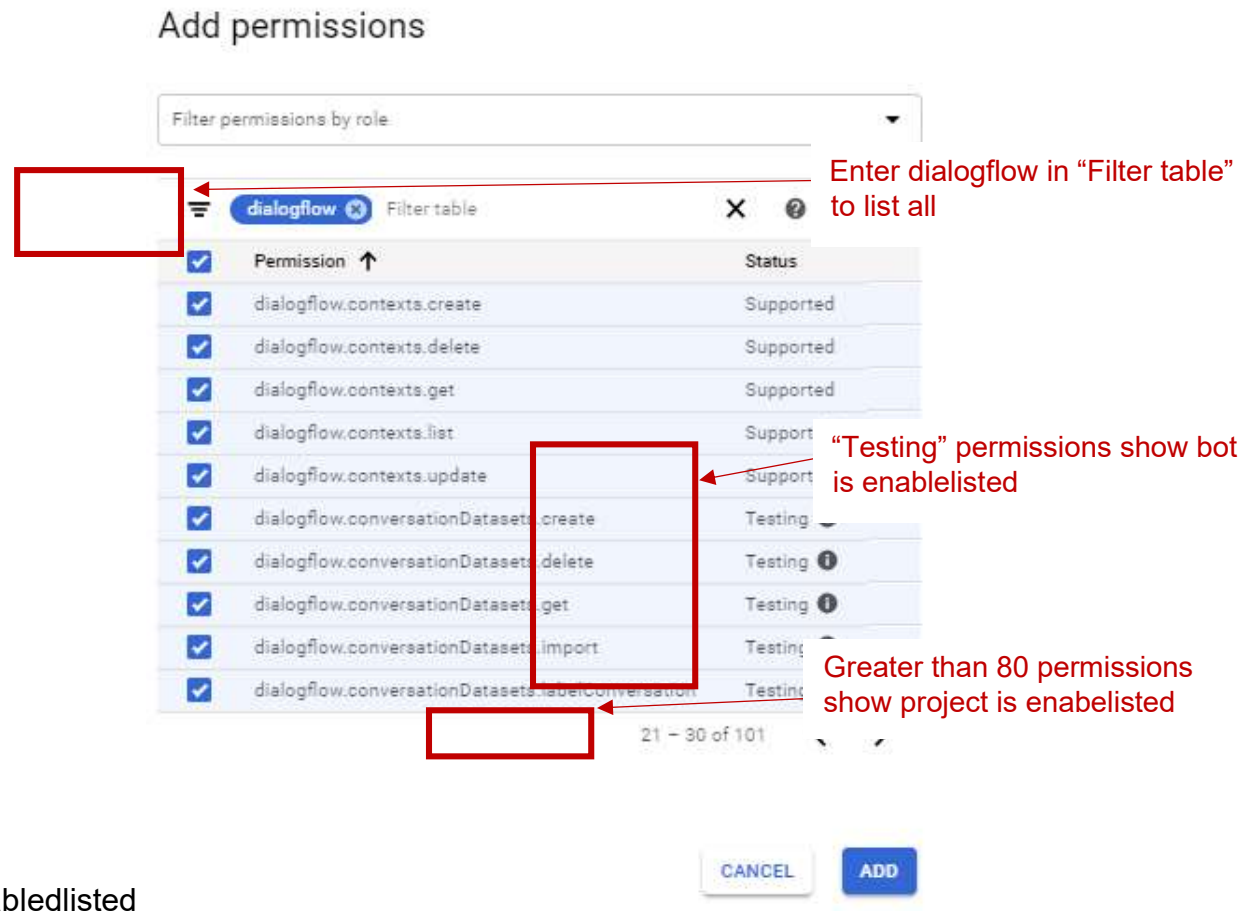
1. Click on Navigation menu and select **IAM & admin**.
2. Choose bot project Name from the drop down on the top of the screen.
3. Click on **Roles** under the **IAM & admin** on the left navigation panel.



4. Click on **Create Role** and enter Name Role (Dialogflow Role)
5. Select **Beta** in **Role launch stage**

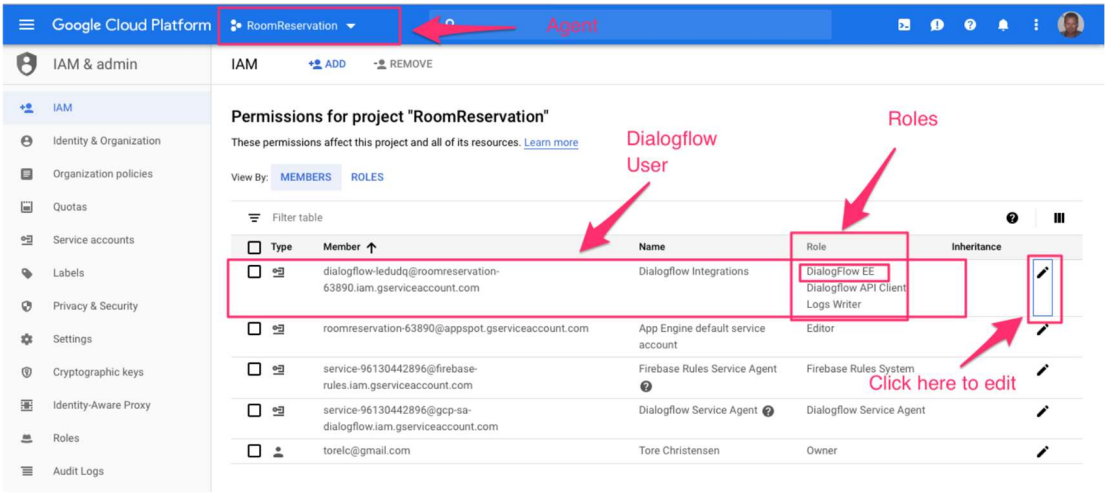


6. Click on **ADD PERMISSIONS**
7. Click on **Filter table** and enter "**dialogflow**" There should be approximately 80-100 permissions. If you only see around 45 or so, then your project has not been



enabledlisted

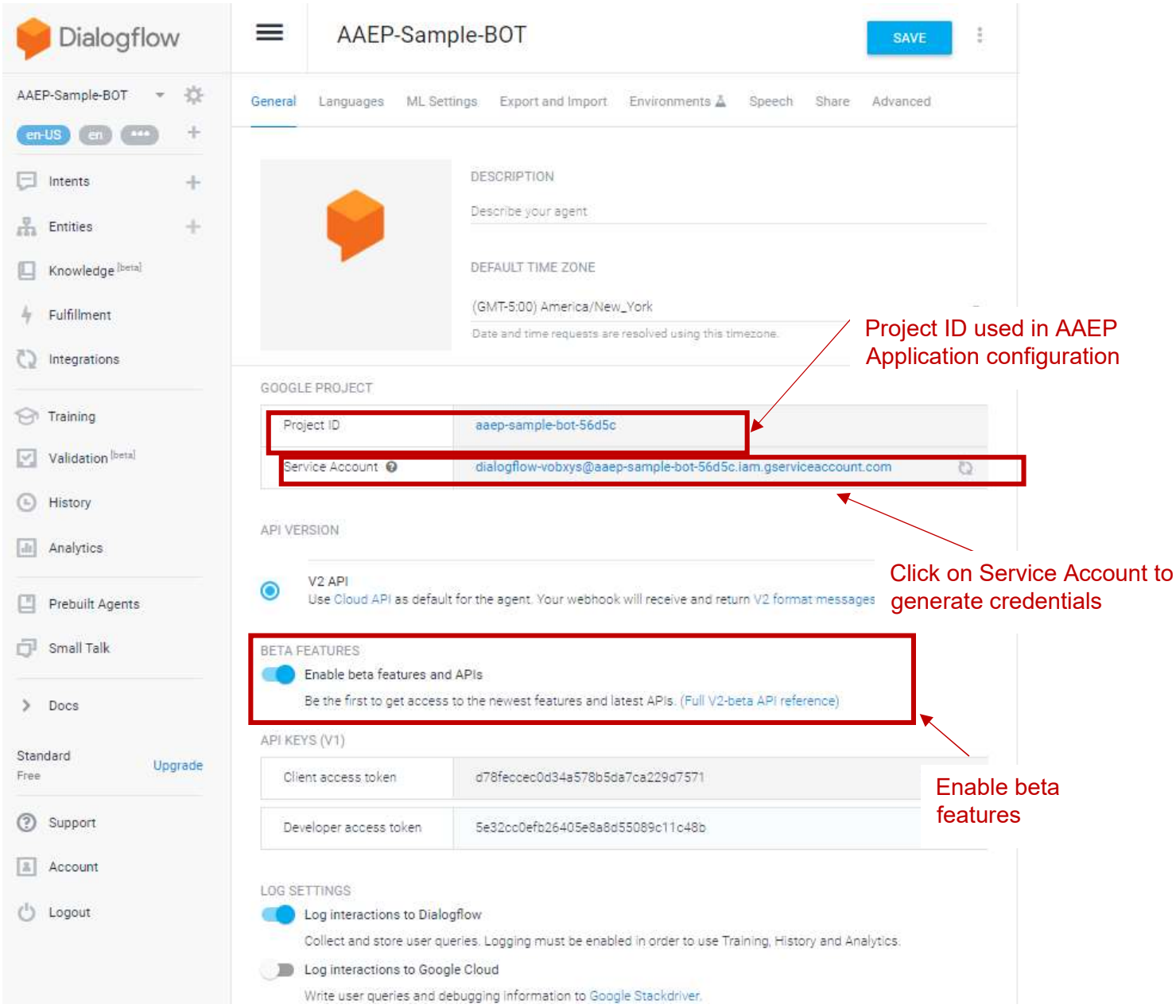
8. Click **Add** and the **Create**
9. Assign Role to Member: Click on **IAM** on the left menu to display the Members page
10. On the **Dialogflow Integrations** member, add the newly created role by clicking the pencil icon and add role to this member.



3.4. Project ID and Generating Credentials

AAEP Dialogflow Application configuration require both the project ID and credentials to access the Dialogflow bot. This section gives details on how to access the Project ID and generate credentials.

1. Go to https://dialogflow.cloud.google.com/ and logon using you Google account, then select your bot.
2. Click on the Configuration Wheel to display Settings page
3. Make sure BETA FEATURES is enabled as shown in the diagram below
4. Project ID is shown in diagram below.



5. To generate JSON credentials, click on Service Account.
6. Click on **Dialogflow Integrations** service account **Actions** and select **Create key**
7. Select **JSON** key type and click Create. This json file is used by AAEP Dialogflow applications as the Credentials.

4. Experience Portal - Dialogflow Interaction Definitions

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

4.1. Welcome Event Sent from AAEP to Dialogflow

When the AAEP **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":{…}
}
```

4.1.1. sip-hdrs Dialogflow context

As part of the Welcome event, AAEP 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 "
  }
```

4.1.2. **avaya-session-telephone Dialogflow context**

As part of the Welcome event, AAEP 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

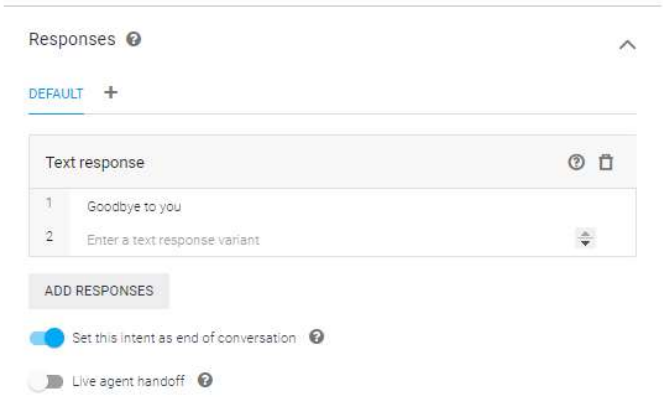
The JSON structure below is an example avaya-session-telephone Dialogflow context:

```
{
  "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"
  }
```

4.2. AAEP detection of End of Conversation signalled by Dialogflow bot

In Dialogflow, 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 AAEP def_dialogflow.vxml application checks responses for an "**intent"** object with field "**endInteraction**" set to true. AAEP plays a canned response and disconnects the call.

4.3. Hangup Event Sent from AAEP to Dialogflow

When the AAEP detects the customer call has disconnected, an event named "**Hangup**" is sent to the Dialogflow bot. The Dialogflow bot should define an intent to match this event to handle the case 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"
}
```

4.4. Dialpad entered DTMF collection using telephony_read_dtmf custom payload

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

AAEP 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

4.5. TELEPHONY_DTMF event to pass back collected DTMF to bot

When the AAEP 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"
    }
}
```

4.6. Barge-in Control

Dialogflow can control whether the audio response from an intent being played by AAEP, can be interrupted. This is accomplished by sending a custom payload to AAEP with a flag to instruct the AAEP to allow barge-in of an audio response being played. AAEP 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
  }
```

4.6.1. Bargein usage in AAEP Dialogflow

In VXML the bargein property normally defaults to enabled. i.e. 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 custom payload. If the user experiences any issues with barge in, it is recommended to disable it on a per intent basis.

4.7. VXML privacy feature control

Dialogflow can enable privacy feature support. This is accomplished by sending a custom payload to AAEP to set the private property. If the private property is enabled AAEP 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 execution. AAEP 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
  }
```

4.8. Playing of Pre-Recorded prompts

Dialogflow can instruct the AAEP 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 AAEP 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/test/Emergency.wav"
  ]
}
```

4.9. DTMF Transfer (Play DTMF into call for Feature Access Code Transfer)

Dialogflow can instruct the AAEP 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"]
}
```

4.10. VXML Transfer

Dialogflow can instruct the AAEP 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 AAEP documentation should be consulted for details on supported header names.

4.10.1. 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”
    }
}
```

4.11. Timer Support – No Input Timeout and Speech Complete Timeout

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

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

The following two sub-sections detail how each timer is used:

4.11.1. 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.

4.11.2. 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.

4.12. Number of Attempts support DF_SYSTEM_NO_INPUT event

When AAEP 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 AAEP Dialogflow integration and will not be detailed here.

4.13. Play Audio until Avaya telephony Follow Up Event Received - Fetch Audio

Dialogflow can instruct the AAEP 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.

4.14. 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 AAEP 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 or 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 AAEP 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.

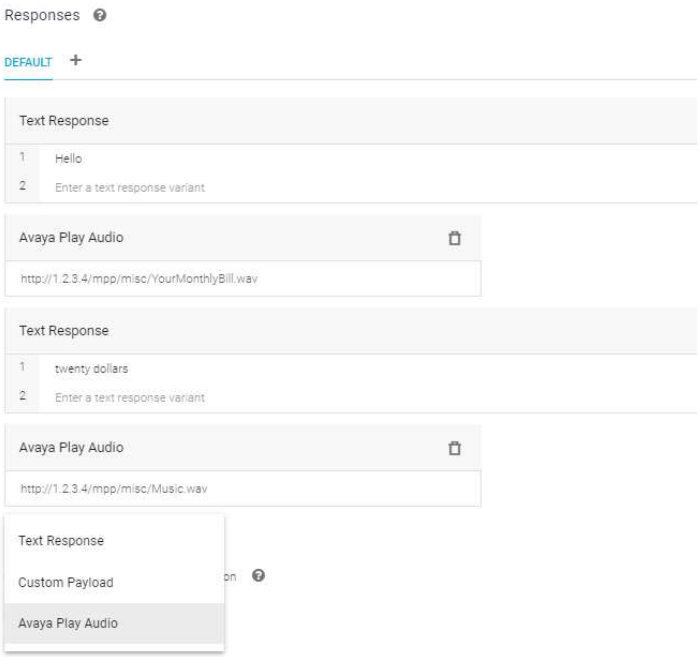
This feature differs from the Play Audio until Avaya telephony Follow Up Event - Fetch Audio in that it plays the full response of the first intent before it sends the event to trigger the follow up intent rather than sending the event before the audio starts playing to allow for music to be played while asynchronous activity takes place.

4.15. 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: This feature is only available If project has been enablelisted for Avaya extra features. Playing wav files for static prompts can be very cost saving over using Google Dialogflow Text To Speech which is charged on a per character basic.

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

Dialogflow GUI provides 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,



4.16. Multilingual Support

Dialogflow can instruct the AAEP 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. Experience Portal Voice Activity Detector

AAEP has updated the Voice Activity Detector (VAD) in 7.2.3 for Cloud speech resources (Dialogflow and Google Speech). A VAD is used to detect the presence or absence of human speech. This allows AAEP to detect if sound is background noise or speech energy. A properly tuned VAD reduces the incidence of announcements being falsely interrupted when barge in is enabled.

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

5.1. VAD Selection

A new configuration parameter has been added to allow selection of old VAD
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)
- MeanSquare – Old VAD

5.2. VAD Tuning

New configuration parameters have been added to AAEP to manually tune the new GMM VAD. These are added to the

5.2.1. **VAD Aggressiveness**

Controls 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)

5.2.2. **Minimum Energy**

Controls 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">10</parameter>
<parameter name="mpp.voip.cloud.vad.min_energy_scl">5</parameter>

Valid settings:

- min_energy – values from 0 to 16,383 (default value 2048)
- min_energy_scl – values from 0 to 32 (default value 6)

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

5.2.3. **Minimum Speech Frames**

Controls minimum number of consecutive speech frames required to trigger VAD

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

Valid settings:

- Values from 1 to 20 (default value of 3)