



Avaya Experience Portal 8.1.2 Nuance Mix DLGaaS Native Support White Paper

Abstract

This paper provides information about the support for Connecting AEP self-service applications to Nuance Mix DLGaaS omnichannel conversation engine. It provides details on configuration and licensing etc. to help customers with DLGaaS integration.

**Issue 1.0
Oct 2022**

Contents

Overview.....	3 -
1. Feature Implementation and Description Overview	4 -
1.1. AEP default VXML Nuance Mix DLGaaS Application	4 -
1.1.1. Invoking default Nuance VXML application as a sub-dialog.....	4 -
1.1.2. Overriding default VXML application during upgrade / patching.....	4 -
1.1.3. Overall Flow of EP integration with Nuance Mix DLGaaS.....	5 -
1.1.4. Default VXML Application interaction with Nuance DLGaaS	5 -
1.1.5. Interactions Between Application and a Simple DLGaaS Bot	6 -
1.2. Termination of Interactions with Nuance Mix DLGaaS	7 -
1.3. Nuance Mix Bot	7 -
2. Experience Portal Nuance Mix DLGaaS Configuration	8 -
2.1. AEP and MPP NTP and Internet access	8 -
2.1.1. Firewall requirements for MPP.....	8 -
2.2. Configure Faster Detection of Dead TCP Connections.....	8 -
2.3. Accessing Nuance Mix Using HTTP Proxy	8 -
2.3.1. Operating System HTTP Proxy Configuration.....	8 -
2.3.2. Nuance Mix HTTP Proxy Configuration.....	9 -
2.4. Nuance Mix Licensing.....	9 -
2.5. Add Nuance Mix DLGaaS Speech Server.....	9 -
2.6. Add Nuance Mix DLGaaS Application to EP	10 -
2.6.1. VAD Parameters in Nuance Application.....	12 -
2.7. Launch Nuance Mix DLGaaS application.....	15 -
3. Configure Nuance Mix Application	16 -
3.1. Create Nuance Mix Project.....	16 -
3.2. Annotate User Input in NLU.....	17 -
3.3. Collect ASR input.....	19 -
3.4. Configure Barge-In.....	19 -
3.5. Configure DTMF Collection.....	19 -
3.6. Configure Transfer Call.....	21 -
• “bridge”: Connection with Nuance maintained even after transfer disconnects.	23 -
3.7. Transfer Call Failed	23 -
3.8. Deploy/Update Application	23 -
3.9. Debug Application.....	24 -
4. Experience Portal Voice Activity Detector	27 -
4.1. VAD Selection	27 -
4.2. VAD Tuning.....	27 -
4.2.1. VAD Aggressiveness.....	27 -
4.2.2. Minimum Energy.....	27 -
4.2.3. Minimum Speech Frames	28 -
4.2.4. Initial Minimum Energy	28 -
4.2.5. Initial Minimum Speech Frames.....	28 -

Overview

The Avaya Experience Portal now supports full native integration with Nuance Mix DLGaaS.

Nuance Mix DLGaaS allows customers to access Cloud AI-based automation for voice calls via Experience Portal.

Nuance Mix DLGaaS speech recognition extracts 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 Nuance cloud platform via <https://www.nuance.com/omni-channel-customer-engagement/ai-for-developers/nuance-mix.html>.

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

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

1. Feature Implementation and Description Overview

Experience Portal will use gRPC framework-based integration to support the Nuance Mix Dialog as a Service (DLGaaS). This API defines the interaction between Experience Portal and Nuance and is accessible via <https://docs.mix.nuance.com/>.

1.1. AEP default VXML Nuance Mix DLGaaS Application

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

This VXML application sends the initial Welcome event and then loops handling responses from Nuance that are returned in the recognition result. For example, playing audio responses from Nuance, 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_nuancemix VXML application will exit or return if it is called as a sub-dialog.

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

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

1.1.1. Invoking default Nuance VXML application as a sub-dialog

The def_nuancemix.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_nuancemix.vxml and is located on MPP at: **\$MPP/web/misc/nuancemix/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_nuancemix.vxml application returns an array of responses received from nuance.

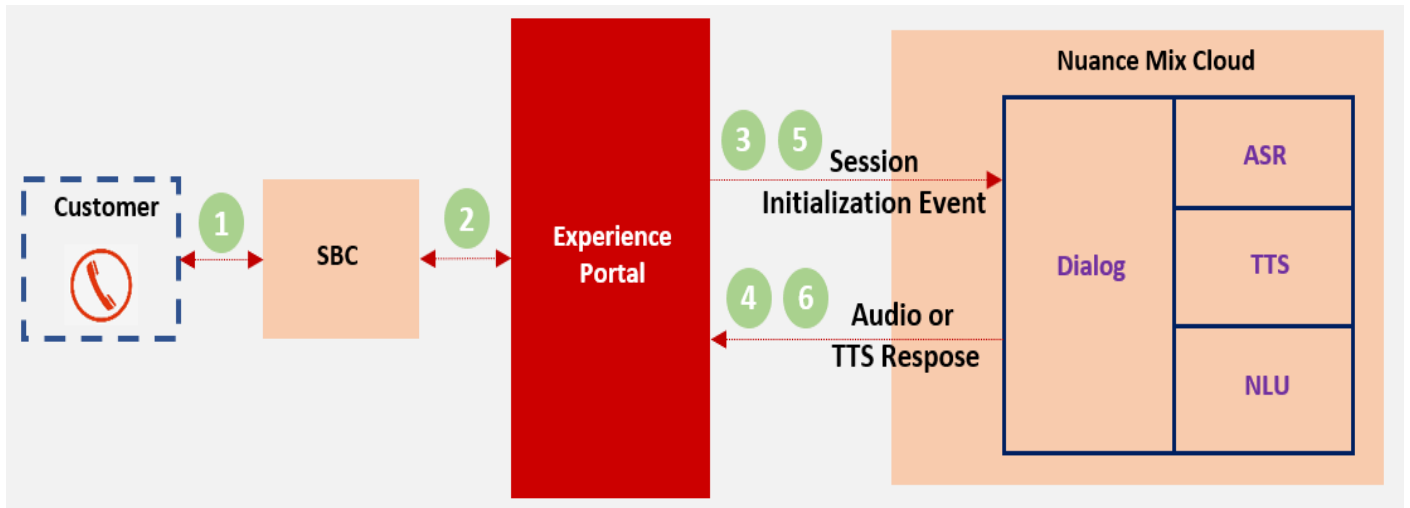
1.1.2. Overriding default VXML application during upgrade / patching

The directory \$MPP/web/misc/nuancemixapp 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 nuancemixapp 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 the default nuancemixapp directory to the new directory after an upgrade.

Figure 1 shows how Experience Portal interacts with Nuance Mix Dialog to deliver rich self-service applications.

1.1.3. Overall Flow of EP integration with Nuance Mix DLGaaS

Figure 1: Experience Portal – Nuance Mix Dialog Native Integration – High-level



1. Customer calls to Experience Portal
2. The Call arrives on Experience Portal (MPP) via SBC (Session Border Controller).
3. MPP will trigger Nuance default VXML application (**def_nuancemix.vxml**) which will connect with Nuance Mix using configured URN and credentials
4. Nuance Mix sends a response (containing audio prompt and custom payload if defined) to MPP
5. MPP plays audio to a customer call. MPP interprets custom payload sent in response from bot and acts on it (collects DTMF, transfer etc) and sends back details to the bot. MPP will stream audio to the Nuance Mix bot.
6. Nuance sends a response to MPP again with audio and custom payload (if defined)

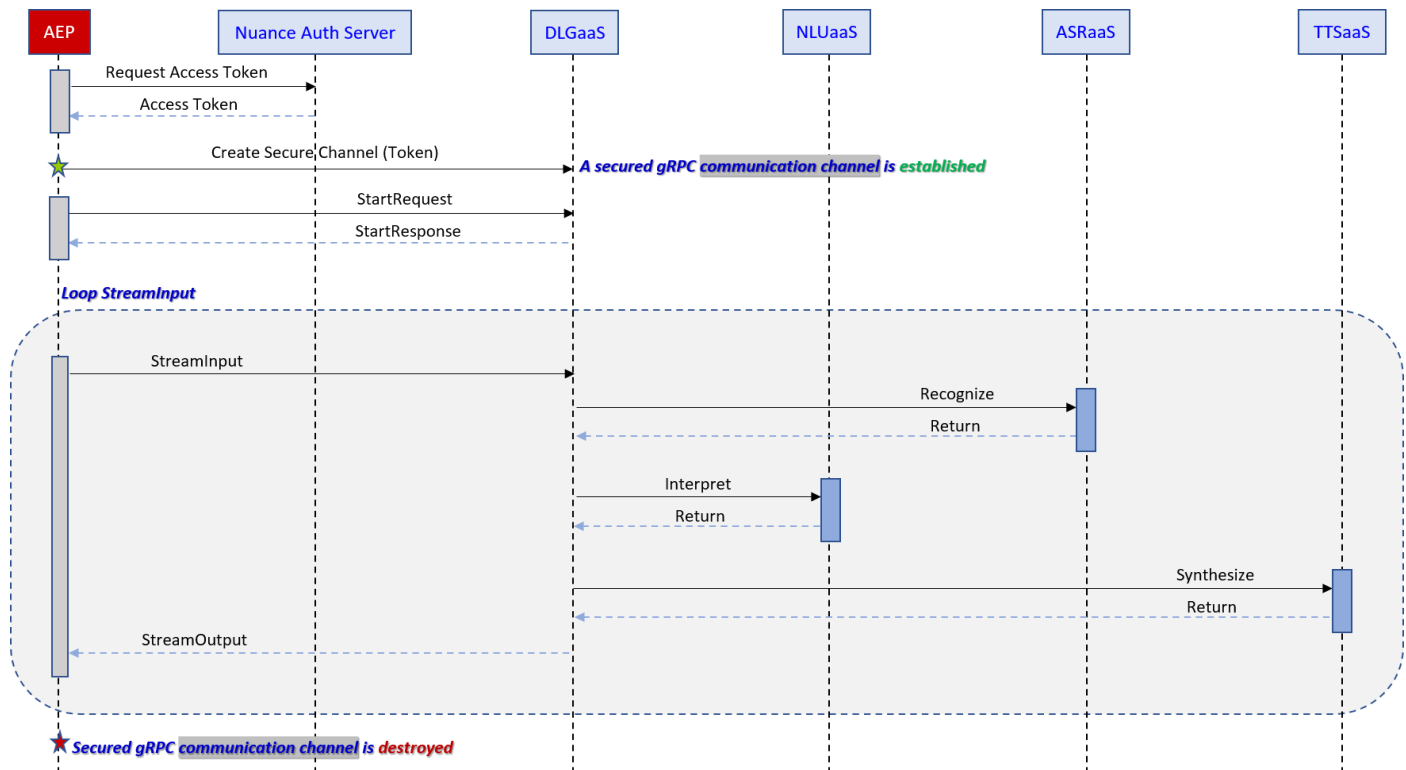
MPP **def_nuancemix.vxml** will loop back to step 5 until the interaction is over

Experience Portal provides service-menu style DTMF and Voice Activity detection services for Nuance Mix.

1.1.4. Default VXML Application interaction with Nuance DLGaaS

The high-level sequence in Figure 2 shows how **def_nuancemix.vxml** interacts with the Nuance Mix DLGaaS bot. This workflow shows that the VXML simply starts recognition (load grammar) with DLGaaS and then waits for a response from the bot on what to do next.

Figure 2: Default VXML workflow



1.1.5. Interactions Between Application and a Simple DLGaaS Bot

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

1. The welcome message "Welcome to the Coffee Shop!" with QA action "How can I help you today?"
2. Instruct Experience portal to collect user responses. Suppose the user says "I would like a large latte"
3. The bot generates responses "Your large latte is coming right up!" and sets this as the end of a conversation with a goodbye greeting "Thanks for visiting the Coffee Shop. Goodbye."

The interactions are as described:

1. **StartRequest.** The first StartRequest message, which is sent from Experience Portal to Nuance Mix DLGaaS, is used to start a session for conversation. This message would provide URN, channel type, language settings to let nuance initialize the applications on their cloud side.
2. **StartResponse.** This is a response to the StartRequest message from the Nuance application which indicates the beginning of a conversation and it might contain a conversation id.
3. **StreamInput.** When Experience Portal is required to send any data (as a stream) to the remote application, a StreamInput message will be fired. It can trigger ASR/DTMF response/event from nuance Mix DLGaaS.

5. **StreamOutput.** This is a response to the StreamInput request, and it contains what Nuance requires Experience Portal to do next. This message can contain audio/responses. For one StreamInput request, multiple StreamOutput might be generated as per VXML protocol specification.

1.2. Termination of Interactions with Nuance Mix DLGaaS

Interactions between AEP and Nuance Mix DLGaaS are terminated when:

- The customer caller hangs up.
- The Nuance Mix DLGaaS "bot" finished its flow and instructs AEP to end the call.
- The Nuance Mix DLGaaS "bot" instructs AEP to complete a blind or consultative transfer of the call.
- The Nuance Mix DLGaaS "bot" instructs AEP that this is "**endAction**" intent in the response. AEP hangs up call.

1.3. Nuance Mix Bot

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

2. Experience Portal Nuance Mix DLGaaS Configuration

This section details the AEP configuration required to use Nuance Mix DLGaaS native integration.

2.1. AEP and MPP NTP and Internet access

As Nuance Mix DLGaaS is an cloud based online only service, 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 the following addresses
 - a. **dlg.api.nuance.com**
 - b. **auth.crt.nuance.com**

2.1.1. Firewall requirements for MPP

The MPP connects to Nuance Mix DLGaaS using a TLS connection on port **443** to FQDN: **dlg.api.nuance.com**. Ensure that the Firewall allows this outbound connection to **dlg.api.nuance.com**, and **auth.crt.nuance.com** over port 443.

2.2. Configure Faster Detection of Dead TCP Connections

AEP uses Google gRPC libraries for communication with Nuance Mix. 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.

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.

2.3. Accessing Nuance Mix Using HTTP Proxy

The following configuration is required if the HTTP proxy server is used to access the internet in the deployment environment. The following sections detail how to configure Experience Portal MPP to use the 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 a 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. Nuance Mix HTTP Proxy Configuration

Nuance Mix does not use normal Red Hat Linux HTTP proxy configuration (using `http_proxy` environment variables). To configure Nuance Mix to use an 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. Nuance Mix Licensing

Nuance Mix DLGaaS server connections are licensed on AEP through the “Conversation Speech Connections” license feature. Obtain a new license with the Conversation Speech Connections feature and apply the license to the WebLM server:

```
<Feature type="counted">  
  <Name>ConversationSpeechConnections</Name>  
  <DisplayName>Maximum number of Conversation Speech  
Connections</DisplayName>  
  <Value>100</Value>  
</Feature>
```

2.5. Add Nuance Mix DLGaaS Speech Server

1. Logon to EP and go to: *Home > System Configuration > Speech Servers > Add ASR Server*.
2. Select Engine Type: Nuance DLGaaS.
3. On Nuance Mix, generate the client credentials (i.e. ID and Secret) and use the "Client ID" and "Client Secret" credentials as shown below:

Figure 3: Creating New ASR Speech Server on AEP

Add ASR Server

Use this page to configure Experience Portal to communicate with a new ASR server.

Name:	<input type="text" value="DLGaaS"/>
Enable:	☒ Yes ☐ No
Engine Type:	<input type="text" value="Nuance DLGaaS"/>
Client ID:	<input type="text" value="appID%3ANMDPTRIAL_someone_avaya"/>
Client Secret:	<input type="text" value="....."/>
Audio Chunk Size:	<input type="text" value="8"/> kilobytes
Total Number of Licensed ASR Resources:	<input type="text" value="1000"/>
<input type="button" value="Save"/> <input type="button" value="Cancel"/> <input type="button" value="Help"/>	

4. The default value of audio chunk size is 8 KB
5. Populate the number of ASR licenses required

2.6. Add Nuance Mix DLGaaS 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_nuancemix.vxml application on your MPP server: http://localhost/mpp/misc/nuancemixapp/def_nuancemix.vxml
5. Need to set the Vendor Parameters for ASR Speech Server with the engine type as Nuance DLGaaS.
 - Default "Provider" is "Nuance DLGaaS" and it is non-editable.
 - Set the "Model Reference" as per the Nuance Mix project bot configuration.
 - The selector fields (i.e. "Channel", "Language" and "Library") are populated with default values as "default", "en-US", "default". For these fields, the user may specify configurations specific to the Nuance Mix project bot.
 - Any Nuance Mix project bot specific configurations can be populated in the "Raw Parameters" field.
 - **Note:** Application-specific credentials can also be added here. The fields "Client ID" and "Client Secret" can be left blank, however, if credentials are added at the application level, they will override the Nuance DLGaaS ASR credentials.

Figure 4: Creating New Application with DLGaaS Engine Type on AEP

Change Application

Use this page to change the configuration of an application.

Name:	6_DLGaaS_TravelApp_LoadTest		
Enable:	☒ Yes ☐ No		
Type:	VoiceXML <input type="button" value="v"/>		
Reserved SIP Calls:	☒ None ☐ Minimum ☐ Maximum		
Requested:	<input type="text"/>		
URI			
☒ Single ☐ Fail Over ☐ Load Balance			
VoiceXML URL:	<input type="text" value="http://10.134.142.36/mpp/misc/nuancemixapp/def_nuancemix.vxml"/>		<input type="button" value="Verify"/>
Mutual Certificate Authentication: ☐ Yes ☒ No			
Basic Authentication: ☐ Yes ☒ No			
ASR Speech Servers <input type="button" value="v"/>			
ASR:	<div>Engine Types Dialogflow Google Speech Nuance</div>	<div>Selected Engine Types Nuance DLGaaS</div>	
Nuance DLGaaS All supported languages.			
VAD Parameters <input type="button" value="v"/>			
VAD Mode:	Disabled <input type="button" value="v"/>		
Vendor Parameters <input type="button" value="v"/>			
Provider:	Nuance DLGaaS <input type="button" value="v"/>		
Model Reference:	<input type="text" value="nuance-mix:tag:model/TravelDemo/mix.dialog"/>		
Channel:	<input type="text" value="default"/>		
Language:	<input type="text" value="en-US"/>		
Library:	<input type="text" value="default"/>		
Raw Parameters:	<div></div>		
Client ID:	<input type="text" value="appID%3Aavaya_IntegrationTest_Prod00"/>		
Client Secret:	<input type="text" value="*****"/>		
TTS Speech Servers <input type="button" value="v"/>			
TTS:	No TTS <input type="button" value="v"/>		
Application Launch <input type="button" value="v"/>			
☒ Inbound ☐ Inbound Default ☐ Outbound			
☒ Number ☐ Number Range ☐ URI			
Called Number:	<input type="text"/>	<input type="button" value="Add"/>	
	<div>2140364</div>	<input type="button" value="Remove"/>	
SIP Header Source:	Any <input type="button" value="v"/>		
Speech Parameters <input type="button" value="v"/>			
Reporting Parameters <input type="button" value="v"/>			
Advanced Parameters <input type="button" value="v"/>			
<input type="button" value="Save"/> <input type="button" value="Apply"/> <input type="button" value="Cancel"/> <input type="button" value="Help"/>			

2.6.1. VAD Parameters in Nuance 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	<p>JSON Pointer: /vadMode Type: enum Valid values:</p> <ul style="list-style-type: none">“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 DLGaaS for analysis. Utterance endpoints are determined by DLGaaS. <p>Controls when the platform VAD is engaged to regulate the audio data streamed to DLGaaS.</p>
VAD Type	<p>JSON Pointer: /vadType Type: enum Valid values:</p> <ul style="list-style-type: none">“GMM” – (Default) Gaussian mixture model detection.“Inqa” – Speech detection using the call progress engine.“MeanSq” – Mean square energy detection. <p>Sets the VAD algorithm used in processing audio data for speech detection. Only applied if the “vadMode” parameter isn’t set to “disabled”.</p>
Aggressiveness	<p>JSON Pointer: /gmm_vad_conf_params/aggressiveness Type: uint32 Valid values:</p> <ul style="list-style-type: none">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	<p>JSON Pointer: /gmm_vad_conf_params/init/min_energy</p> <p>Type: uint32</p> <p>Valid values: [0, 16383], default 2048</p> <p>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”.</p>
Starting Minimum Energy Scale	<p>JSON Pointer: /gmm_vad_conf_params/init/min_energy_scl</p> <p>Type: uint32</p> <p>Valid values: [0, 32], default 12</p> <p>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”.</p>
Starting Minimum Energy Frames	<p>JSON Pointer: /gmm_vad_conf_params/init/min_speech_frms</p> <p>Type: uint32</p> <p>Valid values: [1, 100], default 5</p> <p>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”.</p>
Interim Minimum Energy	<p>JSON Pointer: /gmm_vad_conf_params/intr/min_energy</p> <p>Type: uint32</p> <p>Valid values: [0, 16383], default 2048</p> <p>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”.</p>
Interim Minimum Energy Scale	<p>JSON Pointer: /gmm_vad_conf_params/intr/min_energy_scl</p> <p>Type: uint32</p> <p>Valid values: [0, 32], default 10</p> <p>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</p>

	<p>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”.</p>
Interim Minimum Energy Frames	<p>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”.</p>
Silence Power	<p>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”.</p>
Silence Power Difference	<p>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”.</p>
Smoothing Factor	<p>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”.</p>
Holdover	<p>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”.</p>
Starting Noise Floor	<p>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”.</p>
Starting Noise Offset	<p>JSON Pointer: /inqa_vad_conf_params/init/noiseOffset Type: float Valid values: >=0.0, default 1.0</p>

	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”.

2.7. Launch Nuance Mix DLGaaS application

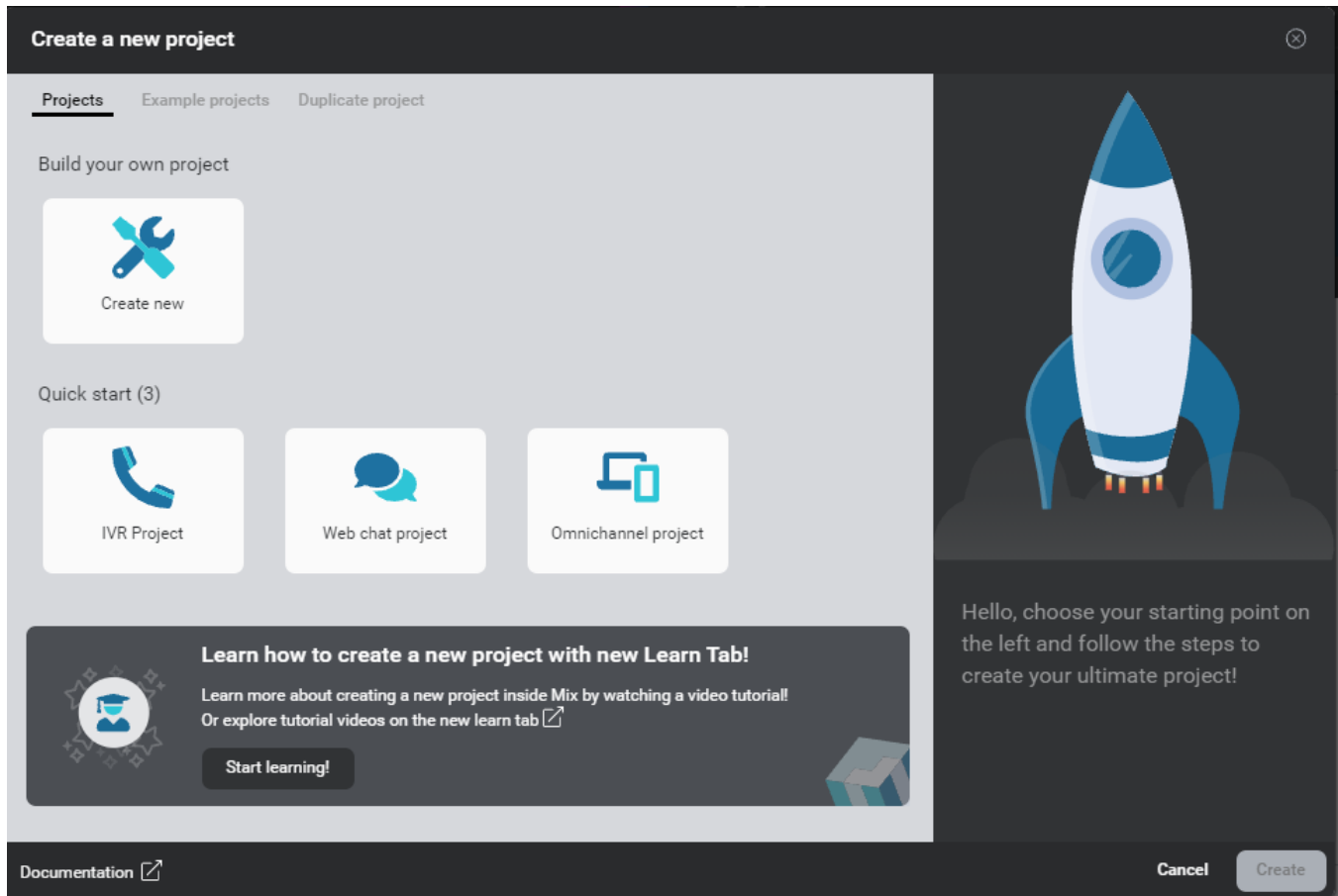
With all the above configurations in place, Nuance Mix DLGaaS is now ready to use. Simply dial the application launch number and you should hear the Welcome Intent response from your Nuance Mix bot.

3. Configure Nuance Mix Application

Experience Portal Nuance Mix integration use Nuance Mix DLGaaS API, which requires developers to build a corresponding Nuance Mix Application before used. A Nuance Mix Application can be treated as a deployment of a Nuance Mix Project. The documents, demo and tutorials are provided at <https://docs.mix.nuance.com/>.

3.1. Create Nuance Mix Project

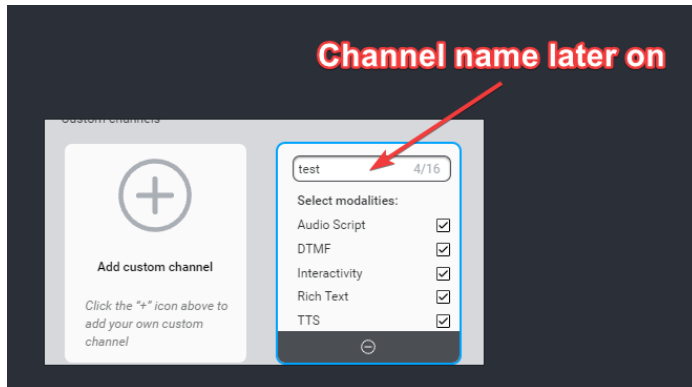
Figure 5: New Project Using Templates



The first step of creating a Nuance Application is to start with a new project. In one Nuance Mix Project developers can choose multiple channels, each channel representing one different solution, for example, calls, webchats. Nuance also supports multiple languages, however in each call currently only one language is supported.

For a traditional contact center, IVR Project is designed for the normal call. However, developers can also pick what they want themselves like below:

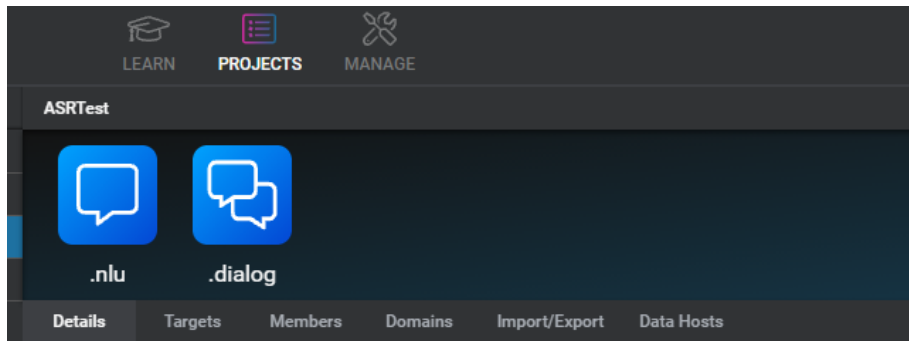
Figure 6: Naming a Channel



3.2. Annotate User Input in NLU

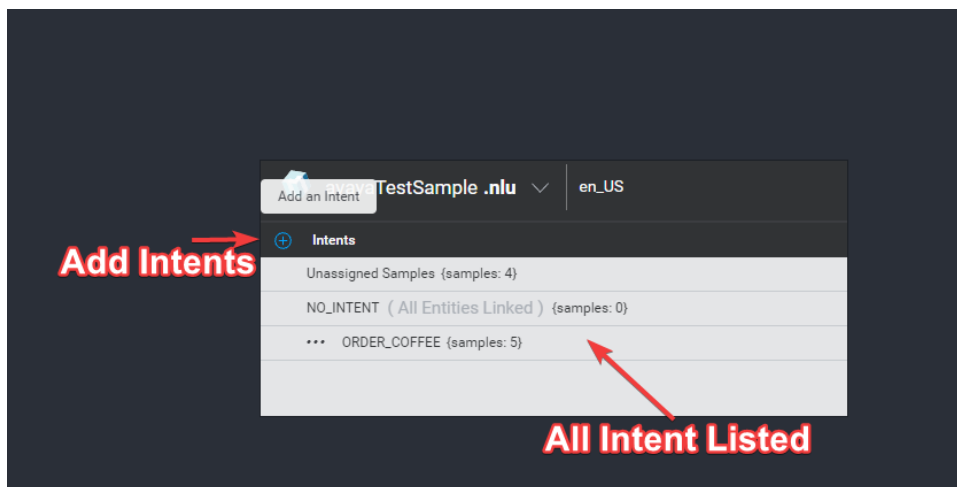
First click “.nlu” in the project page:

Figure 7: Selecting NLU Package



In “.nlu” the task is to tag input samples as intents used in the project, and, for each sample inputs, identify the corresponding entities.

Figure 8: Adding Intents



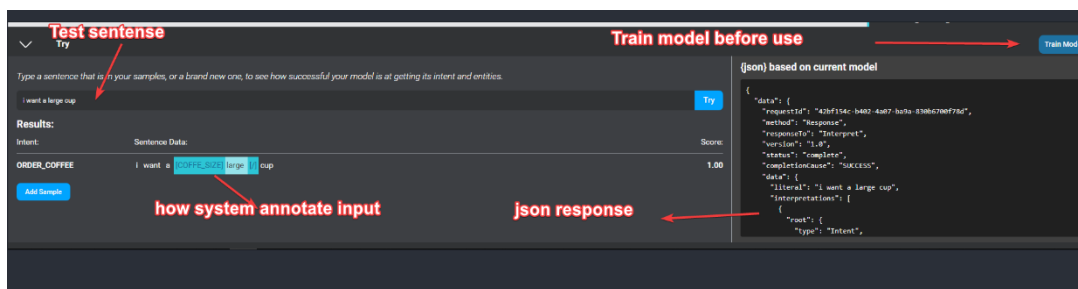
Create a bunch of intents that may possibly happen inside a specific dialogue. In Nuance Mix, for any input which cannot be tagged as any intent here, it would be recognized as NO_INTENT.

Figure 9: Adding Entities in the Intents



In each input, the dialogue processes only useful information termed as entities. Developers must first decide what kind of entities appear in the intent and then add some input samples. In the end, select the input samples to indicate the entities that is useful for the project.

Figure 10: Build and Train Intents



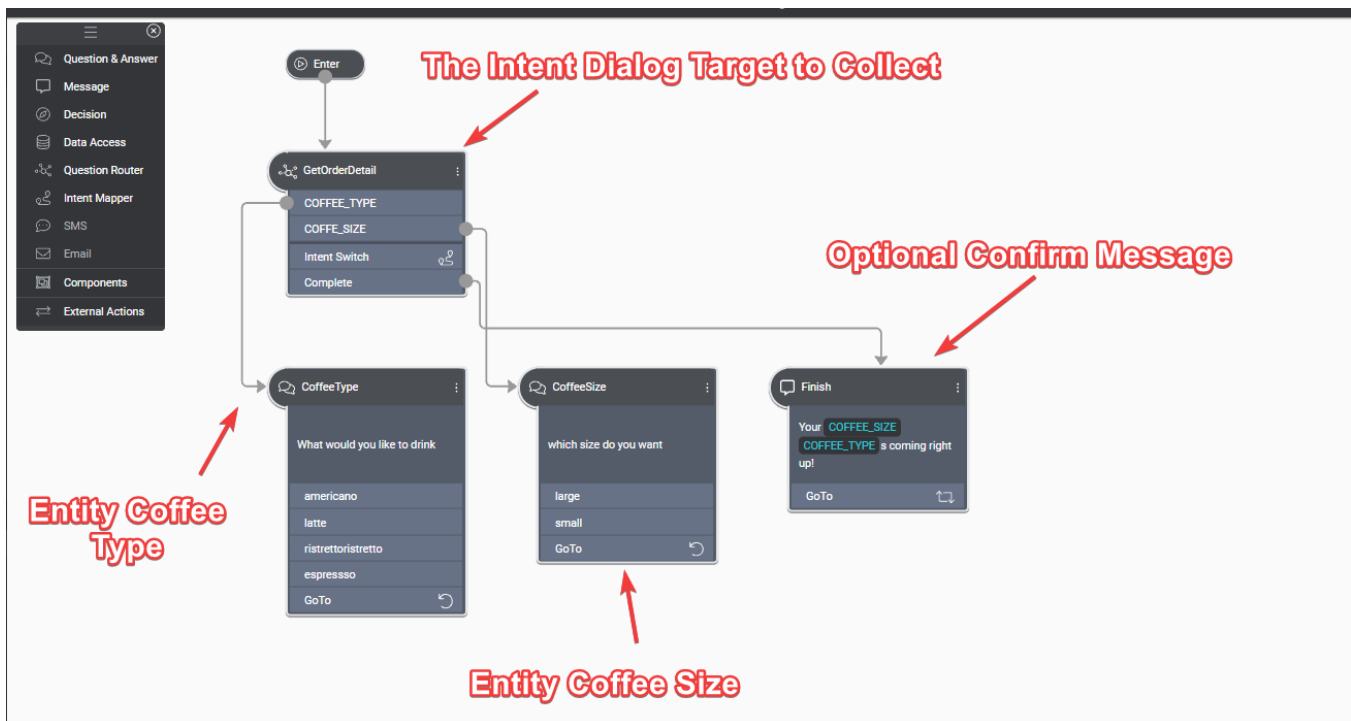
When finished configuring the intents a user should "train the model" and test with some test inputs. From this test, Nuance will give you a visual representation of the response and its JSON equivalent. If the test sentence can is not recognized correctly then we recommend adding it as a new sample to the intent.

3.3. Collect ASR input

When a developer tries to collect information from customers' voices, they need to define intent and create an intent mapper inside their dialog project.

Check <https://docs.mix.nuance.com/mix-dialog/#set-up-an-intent-mapper-node> to get full detail.

Figure 11: Sample Intents in a Dialog



3.4. Configure Barge-In

Experience Portal currently only support voice barge-in. “Hotword” is not supported.

Nuance provides turn off barge-in for the specific message. For details, please check

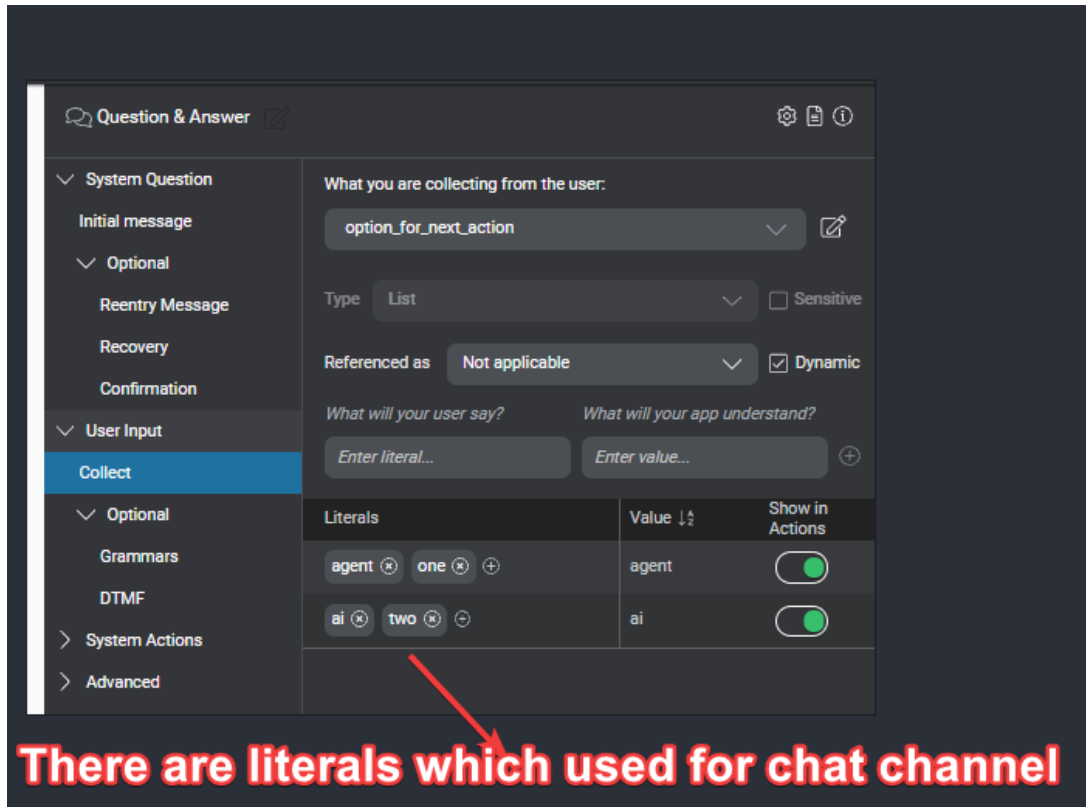
<https://docs.mix.nuance.com/mix-dialog/#view-channel-specific-dialog-flow>.

3.5. Configure DTMF Collection

Notice: When connecting with Nuance Mix DLGaaS, Experience Portal supports collecting one digit at a time as a valid DTMF terminal character to jump between parts of dialogue.

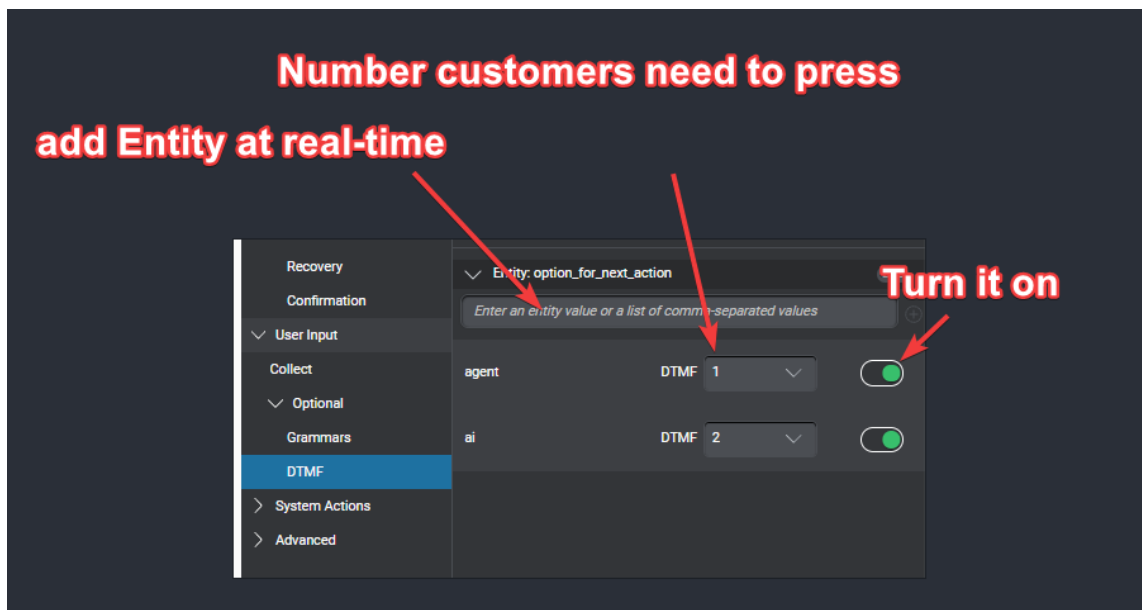
Ensure your channel has DTMF selected when you create your project. DTMF in Nuance Mix DLGaaS is considered as another mode of user input other than voice. The first step is to define those input literals and values.

Figure 12: Defining Literals for DTMF Processing



Then in the DTMF tab pick up the keys representing a specific input.

Figure 13: Associating Entities with DTMF Literals

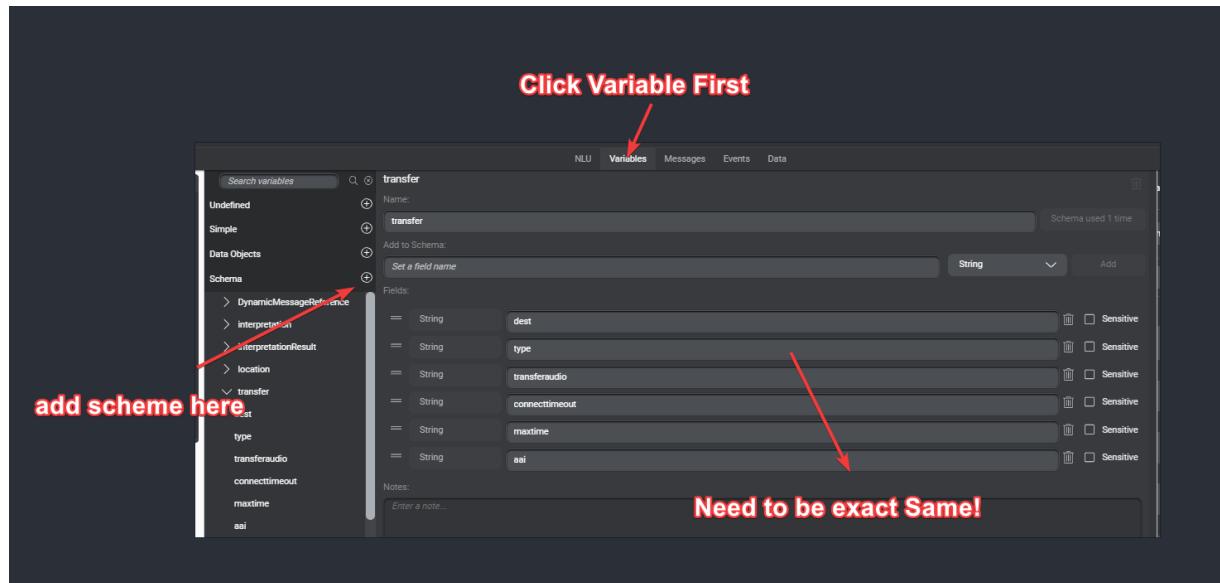


For more details please check <https://docs.mix.nuance.com/mix-dialog/#set-up-a-question-and-answer-node>.

3.6. Configure Transfer Call

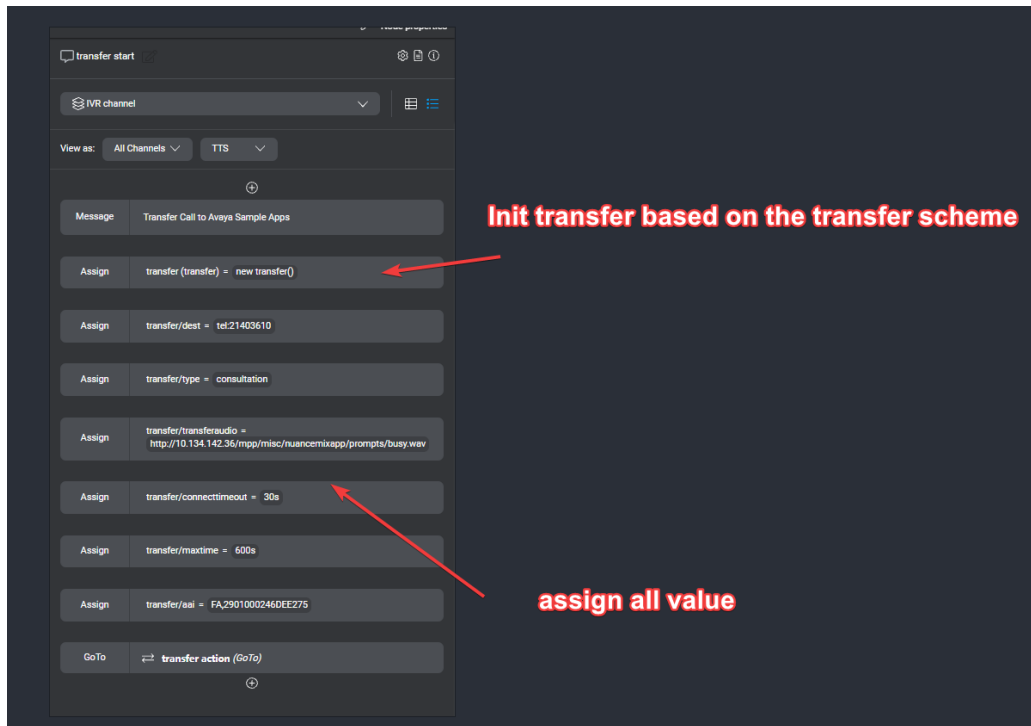
Nuance Mix uses a transfer action to trigger a transfer process. Document <https://docs.mix.nuance.com/dialog-grpc/v1/#transfer-actions>. To make a successful transfer Experience Portal will require additional information. The developer will first need to create a variable scheme - see below.

Figure 14: Adding a Transfer Scheme



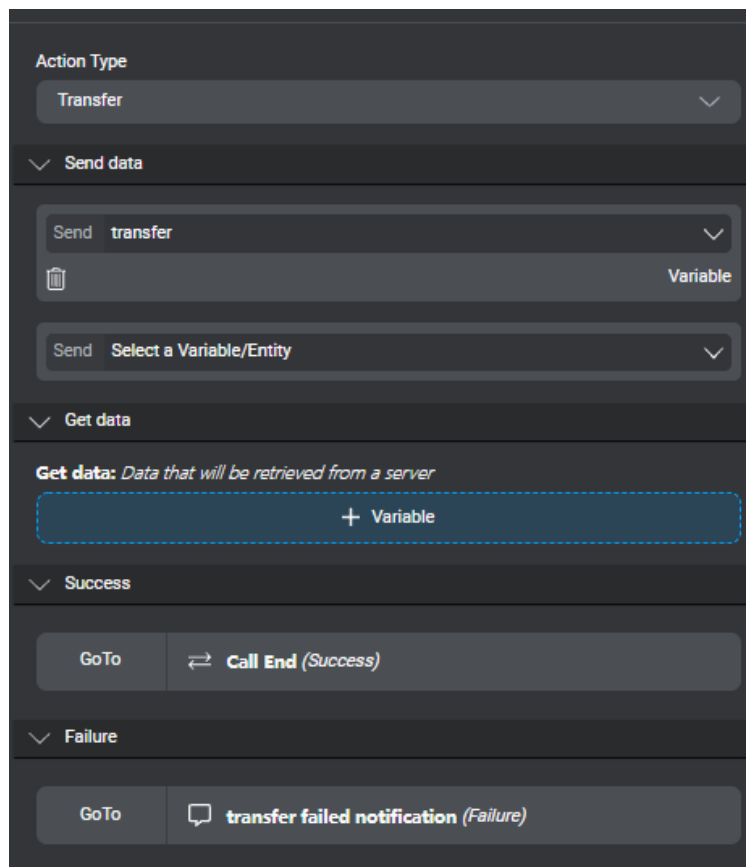
Attention: All the variables should be the same name as in the image above, otherwise the default_nuancemix.vxml would not work. After creating the transfer scheme, next is initialization of the variable based on the scheme.

Figure 15: Initializing a Transfer Scheme



This process must be done before the transfer action. In the end, send the data in the transfer action:

Figure 16: Defining Transfer Action



Attention: The variable sent to Experience Portal must be named "transfer", otherwise default_nuancemix.vxml would not work.

For all the variables in the settings above, here is detailed instructions:

Transfer Field	Description (Detail in VXML spec)
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: <ul style="list-style-type: none">• "bridge": Connection with Nuance maintained even after transfer disconnects.• "blind": Connection with Nuance dropped regardless of transfer outcome• "consultation": Connection with Nuance 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 Information (aai) to be sent to far end application

3.7. Transfer Call Failed

When control for a failed bridged transfer is returned to the default_nuancemix.vxml application an event is sent to the Nuance Application.

Four reasons would be returned:

- busy
- "network busy"
- "noanswer"
- unknown

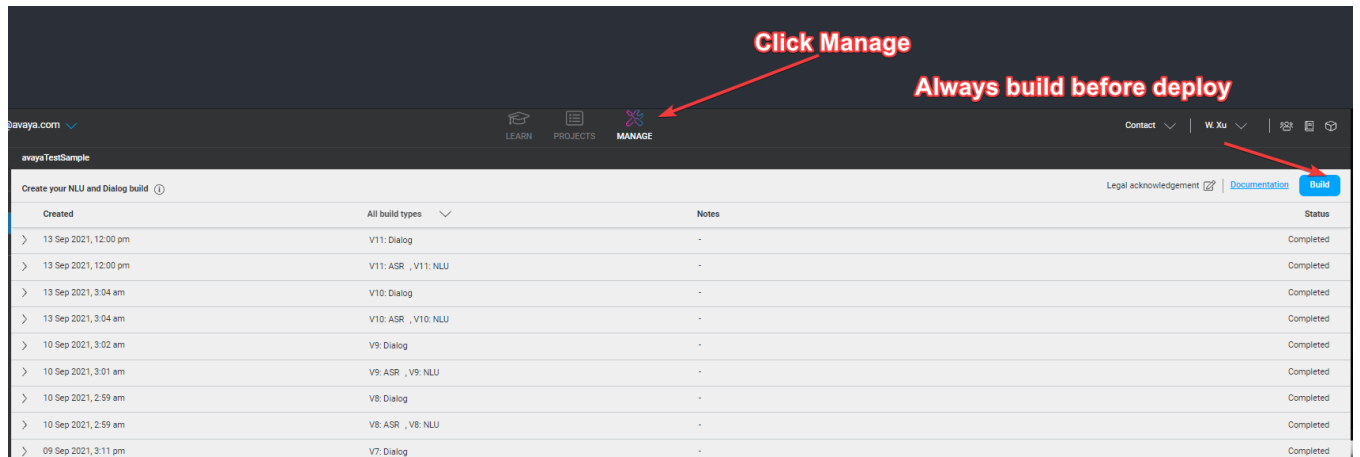
Reason code would be specified in the "returnMsg" field.

3.8. Deploy/Update Application

To deploy/redeploy an application, developers need to get a valid build of the project first.

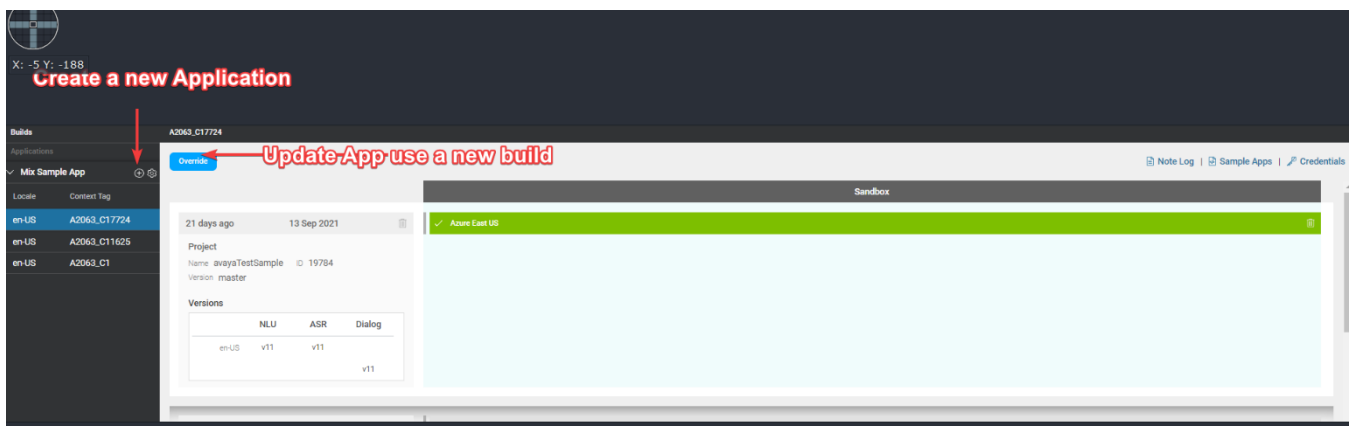
Attention: A valid build is mandatory before the deployment!

Figure 17: Build Projects



Once build is successful the management page of Nuance Mix will appear.

Figure 18: Deploy or Redeploy Application

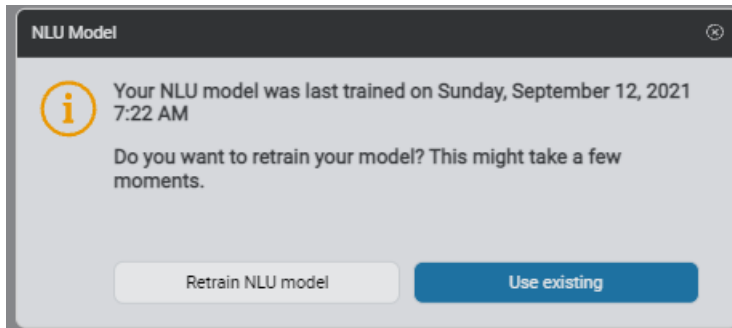


If you don't have any application settings, a new one should be created. If you want to update a current working application, use the override function to deploy a new build result to the same application. When developers use overwrites to update their applications, all credentials/URNs are kept unchanged.

3.9. Debug Application

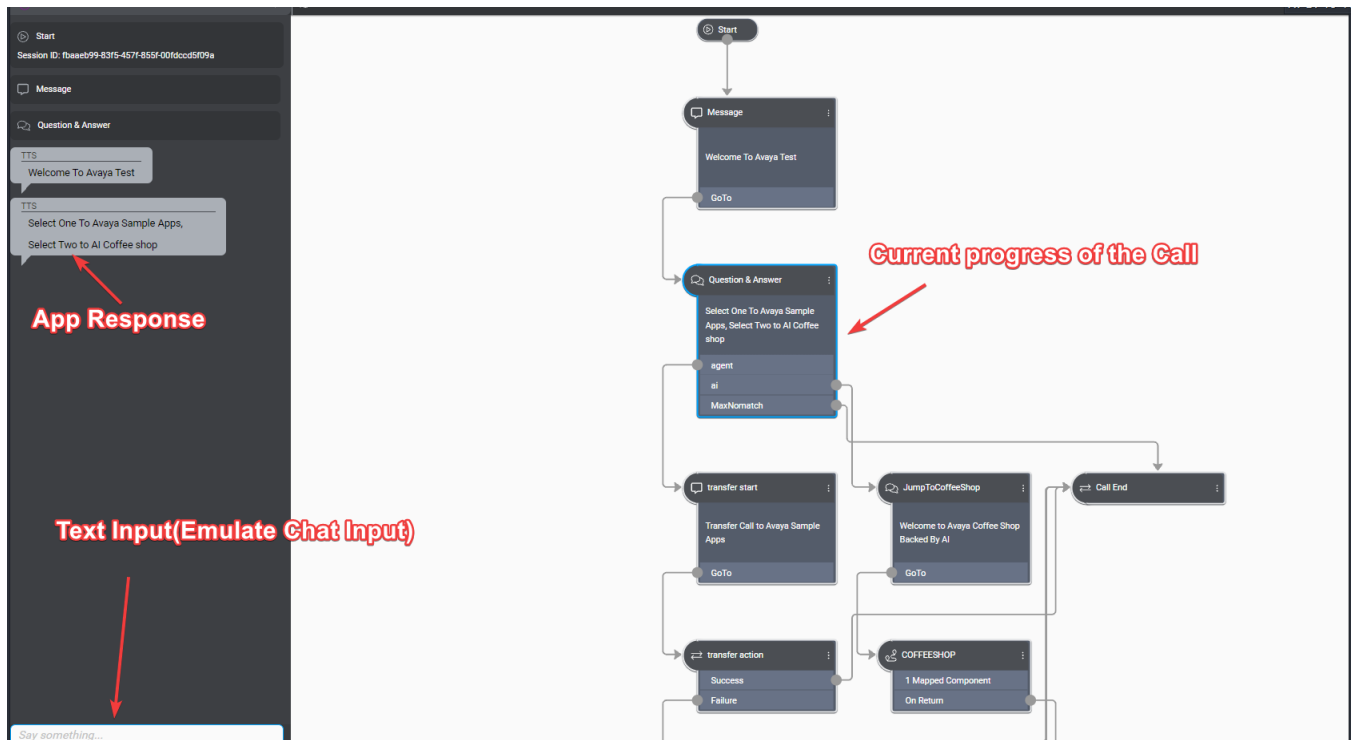
Nuance provides two ways to debug applications. The first solution is embedded when working on the dialogue in the project. First click test in the dialogue page:

Figure 19: Retraining NLU



After retraining NLU model developers would be introduced to debug options.

Figure 20: Local Debug



Attention: Using the local test embedded inside the Nuance Mix application, it is not possible to test all the functionalities (e.g., DTMF cannot be tested using local test environment).

Avaya recommends using the alternative web-based solution <https://demo.mix.nuance.com/app/> to test user created applications.

Figure 21: Nuance Mix Demo

Nuance Mix

[Profile](#) [DLGaaS](#) [NLUaaS](#) [TTSaaS](#)

Set up your configuration, and save it for a URL you can bookmark and share.

Client Configuration

Client ID

Client Secret ***
.....

DLGaaS Tab Configuration

Simulate Experience
Visual VA: Text Input with HTML Output

DLG Channel
default

DLG Language
en-US

DLG URN
urn:nuance-mix:tag:model/REPLACE_ME/mix.dialog

DLG Session Timeout
900

DLG Session ID (optional)

NLUaaS Tab Configuration

NLU URN
urn:nuance-mix:tag:model/REPLACE_ME/mix.nlu?=language=eng-USA

*** Keep your Client Secret handy (and safe and secure!)
– you need to provide it with every new browser session.

Save

This website provided by Nuance provides full functions on a call with DLGaaS and it would generate the same response with selected Application.

4. Experience Portal Voice Activity Detector

AEP has updated the Voice Activity Detector (VAD) in 8.1.0 for Cloud speech resources. A VAD is used to detect the presence or absence of human speech. This allows AEP to detect if the 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.

4.1. VAD Selection

A new configuration parameter has been added to allow the 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

4.2. VAD Tuning

New configuration parameters have been added to AEP to manually tune the new GMM VAD. These can be configured through the **\$MPP/config/mppconfig.xml** file.

4.2.1. 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)

4.2.2. Minimum Energy

Controls system-wide 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.

4.2.3. Minimum Speech Frames

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

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

Valid settings:

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

4.2.4. 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 factors, 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 the 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)

4.2.5. 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)