Genesys Intelligent Automation Help

Natural Language AI Integration

12/19/2025

Contents

Natural Language AI Integration

Intelligent Automation can integrate with external artificial intelligence (AI) services to provide natural language understanding (NLU) capabilities.

Important

The Natural Language Menu App and Custom Natural Language Menu are shipped separately and only allowed in conjunction with the purchase of either the bot bundle/orchestration either Voicebot or Chatbot bundle. Get in touch with your AE or Sales.

When Intelligent Automation connects to a natural language engine, Intelligent Automation allows a user to use natural language when responding, even entering multiple pieces of information in a single response. The natural language engine infers the relevant information contained in a user's response, enabling some subsequent Menu and Question blocks in the call flow to be skipped, because the information has already been captured. The result is a successful conversation that is shorter than that of the directed-dialog approach.

Warning

Please note that the customer is responsible for ensuring that the environment and bot applications they build are properly configured and secured according to PII and HIPAA requirements.

How it works

The natural language engine works in the background while Intelligent Automation communicates directly with the user. Here's how:

1. Intelligent Automation asks an open-ended question (For example, *What can I help you with?*)
2. The user provides a natural language response (For example, *What will the weather be like in London on Saturday*?).
3. Intelligent Automation receives the response and passes it along to the natural language engine.
4. The natural language engine responds, as follows:
    - If the natural language engine needs more information to understand the request, it responds to Intelligent Automation with follow-up questions. (For example, *For what city?*).
    - If the natural language engine has all the information it needs, it sends Intelligent Automation the Intent and associated Slots (In the Weather example, the Intent could be Weather and the Slots could be weatherLocation and weatherDate).

5. When Intelligent Automation receives and reads the Intents and Slots, it directs the interaction according to the configured application.

The following demonstrates a chat session with a natural language engine running in the background:

Link to video

Intelligent Automation currently has built-in connections to Genesys Dialog Engine and Google Dialogflow (the v2 API) . If you want to use other natural language AI services, contact your Genesys representative for help with a custom integration.

How to integrate with natural language AI services

To integrate with a natural language AI service, complete the following steps:

1. Configure Default Server Settings (applies to Genesys Dialog Engine only)
2. Configure NLU Settings
3. Map Intents to modules
4. Map Slots to questions

Configure Default Server Settings

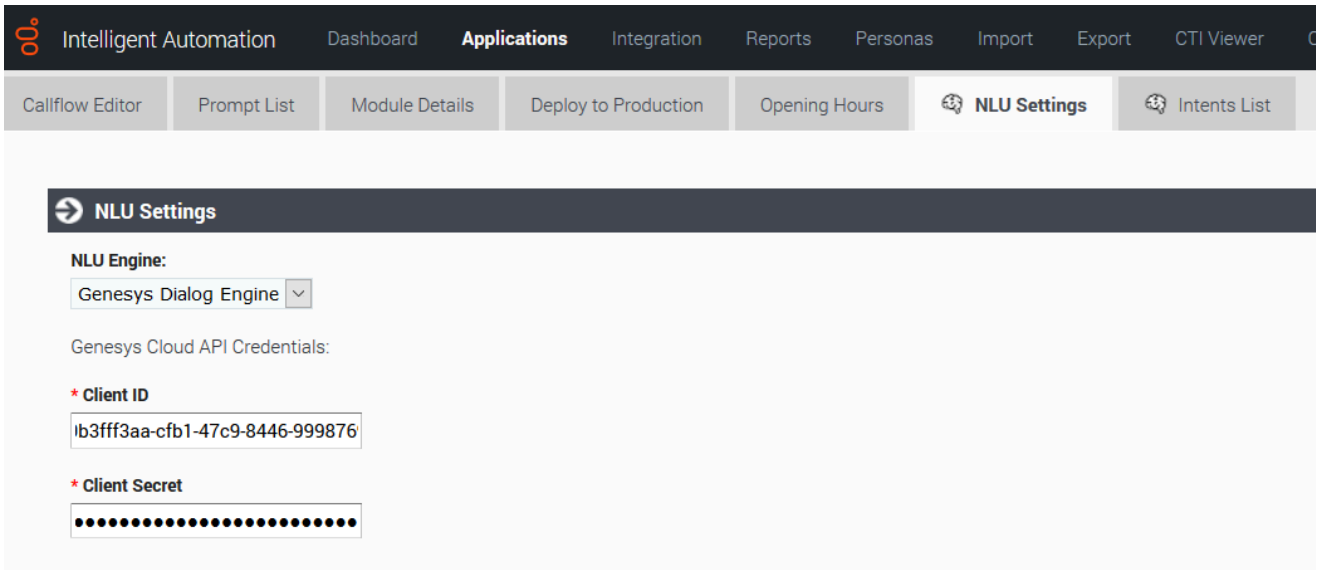
If you're using Genesys Dialog Engine, configure the following server settings:

- **DialogEngine.JOPv3.AuthBaseURL** - The URL to the Genesys Cloud Authentication API.
- **DialogEngine.JOPv3.FetchTimeoutMillis** - The length of time (in milliseconds) that Intelligent Automation waits for a response from Dialog Engine before throwing an error.
- **DialogEngine.JOPv3.ClientID** - The Dialog Engine Client Identifier.
- **DialogEngine.JOPv3.ClientSecret** - The Dialog Engine Client Secret password.

Important

If you do not have access to these settings, contact your Genesys representative.

Configure NLU Settings



NLU Engine: Genesys Dialog Engine

Genesys Cloud API Credentials:

* Client ID: lb3fff3aa-cfb1-47c9-8446-999876

* Client Secret: ●●●●●●●●●●●●●●●●●●●●●●●●●

Once you have configured the Default Server Settings, open a **Natural Language Menu** module and click the **NLU Settings** tab.

From the **NLU Engine** menu, select the natural language engine you're using - either Genesys Dialog Engine, Google Dialogflow, or Microsoft Bot Framework.

- If you select Genesys Dialog Engine, the **Client ID** and **Client Secret** password configured on the **Default Server Settings** page will display. Leave the **Use default credentials** box checked to use these credentials or uncheck the box to override them for this particular application.
- If you select Google Dialogflow, you'll need to enter the Google API Service Account JSON. Google provides this JSON when you set up your Google services account.

Important

Ensure that you have the Dialogflow API Admin role configured to your Google services account.

- Ensure that the value of the **DialogEngine.Provider** setting (**Administration** > **Default Server Settings**) is set to `SlotBucket`.

Support for Proxies

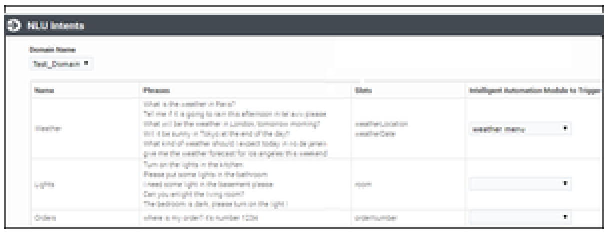
To use Google DialogFlow through a proxy, create an environment variable called `GRPC_PROXY_EXP` and set the value in the *host:port* format.

```
GRPC_PROXY_EXP=localhost:31138
```

Map Intents to modules

Important

The Map Intents to modules is supported for Genesys Dialog Engine, Google Dialogflow, and Microsoft Bot Framework.



NLU Intents

Domain Name: Test_Domain

Name	Phrases	Slots	Intelligent Automation Module to Trigger
Weather	What is the weather in Paris? Tell me if it is going to rain this afternoon in tel aviv please What will be the weather in London, tomorrow morning? Will it be sunny in Tokyo at the end of the day? What kind of weather should I expect today in rio de janeiro give me the weather forecast for los angeles this weekend	weatherLocation weatherDate	weather menu
Lights	Turn on the lights in the kitchen Please put some lights in the bathroom I need some light in the basement please Can you enlight the living room? The bedroom is dark, please turn on the light !	room	
Orders	where is my order? its number 1234	orderNumber	

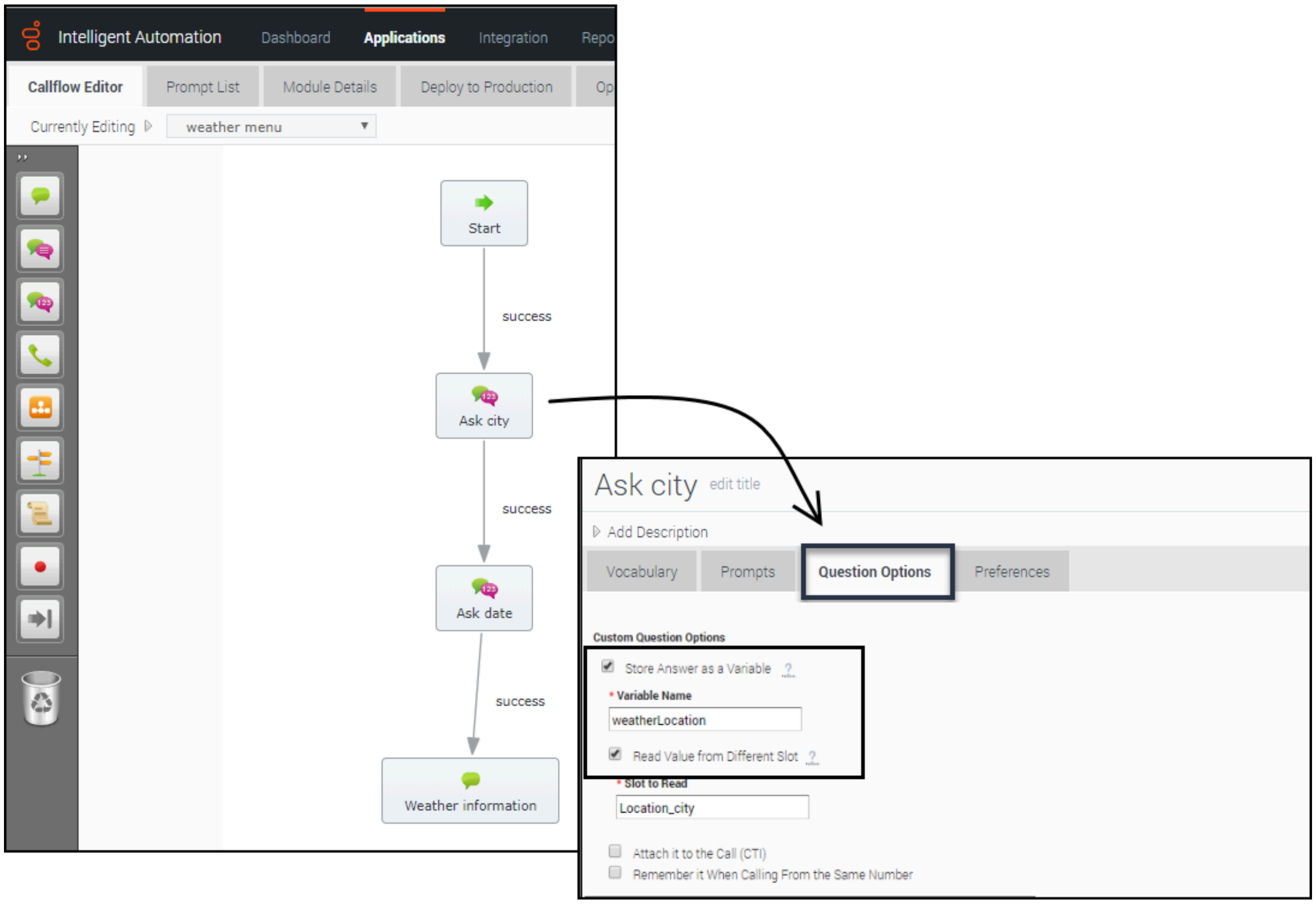
Intelligent Automation reads all Intents, Utterances, and Slots associated with a domain and then displays that information on the **Intents List** page for the **Natural Language Menu** module. This is where you'll map each Intent to a module.

To map an Intent to a module, simply select a module from the **Intelligent Automation Module to Trigger** menu for the Intent. Intelligent Automation will then automatically update the application's callflow accordingly.

Important

If the fulfilment is being performed by the bot, you can select *None* from the **Intelligent Automation Module to Trigger** menu and a generic “success” result will be returned after the intent has been completed.

Map Slots to questions



Custom Question Options

- [✓] Store Answer as a Variable
- * Variable Name: weatherLocation
- [✓] Read Value from Different Slot
- * Slot to Read: Location_city
- [ ] Attach it to the Call (CTI)
- [ ] Remember it When Calling From the Same Number

In the natural language engine, an Intent contains Slots, which are key pieces of information you need to extract from the end user to process a request. For example, if a user wants the weather forecast, you would need to know two key pieces of information before providing a weather forecast - city and date. These would be considered Slots.

For each Intent, you should have an associated module containing questions that will extract the right information from the customer. For the Weather example, you would have a Weather Questions module, which contains two questions: *For what city* and *For what date*? Both of these questions relate to the weatherLocation and weatherDate Slots.

Once you have created this module in Intelligent Automation, you need to map it to its associated Slots, as follows:

1. Open the module that is linked to Intent.
2. For each Question block in that module for which you would expect to have a slot, open the block and go to **Question Options**.
3. Check the **Store Answer as a Variable** checkbox and enter the Slot name (weatherLocation).

When a chat or voice session reaches one of those Question blocks, it first checks if that slot has been sent to Intelligent Automation from the natural language engine. In that case, Intelligent Automation uses that as the answer to the question. Otherwise, Intelligent Automation asks the question and waits for a response.

Support for Live Agent Hand-off

Important

This feature is supported only for Google Dialogflow.

When the live agent hand-off flag is returned as part of the DialogFlow intent, you can configure IA to follow a special path, usually a hand-off to a live agent.

When a value is provided in the **Path to take When Agent Requested** setting, the call is transferred to a live agent defined in the callflow.

In Dialogflow, the liveAgentHandoff flag should be enabled using a Custom Payload response within the intent.

```
{ “liveAgentHandoff” : true }
```

Support for Maximum Attempts

You can set the maximum number of retries to fill a slot and the maximum number of attempts to identify intents in a chat session.

The **Enable Maximum Attempts Counters** setting allows configuring the number of retry attempts and the action to be performed when the limit is reached.

The **Enforce Maximum Attempts for filling Mandatory Slot** will enable or disable the slot fill failure timeout feature. The number of unsuccessful tries can be configured after which the call is either transferred to a different module or return a result.

For example, if the number of retries is specified as 2, the bot will look if it can fill up any slots based on the user input two times. If the slot is not filled after both attempts, the call flow will be transferred to either a different module or return a special value.

When a value is provided in the **Maximum Attempts to Fill Mandatory Slot Values** option, the bot will try to slot-fill the user input. If the bot cannot perform a slot-fill successfully, the counter is increased. When the counter value exceeds the maximum retry values, Intelligent Automation either transfers the callflow to a different module (configured in the **Exit Module** field) or return a special result.

Important

When a slot is filled successfully or if the intent is switched, the counter is reset to zero.

When the **Enforce Maximum Attempts for Intent Disambiguation** option is configured, the bot will try to identify an intent from the conversation. When the maximum number of unsuccessful attempts is reached, Intelligent Automation either transfers the callflow to a different module (configured in the **Exit Module** field) or return a special result.

Pass Context Settings

Important

The Map Intents to modules is supported for Genesys Dialog Engine, Google Dialogflow, and Microsoft Bot Framework.

Intelligent Automation allows you to pass variables to an NLU engine. The NLU engine can use the variables as part of the slot filling capabilities when the **Send Intelligent Automation variables as context to NLU Engine** setting is enabled.

For DialogFlow, the parameter's default value must be mapped to the variable defined in Intelligent Automation. You can configure the Default Value option from the contextual menu for a parameter in DialogFlow and it should be use the following format: *#fish_context.<VariableName>*.

For Dialog Engine, all variables are passed and Dialog Engine will map the variables to slots if the variable names from Intelligent Automation match the slot names in Dialog Engine.

For Microsoft Bot Framework, the context settings are passed in the `slots` section and Microsoft LUIS can use this to extract information. The context is passed only from Intelligent Automation to Microsoft Bot Framework. Context is **not** passed from the Bot Framework to Intelligent Automation.

Bypass Opening Question

You can allow users to bypass the initial question from the bot by passing the utterance as a variable to the NLU. The NLU then skips the initial question and proceeds to the next question. You can use any previously collected data or create and define a variable in the Script block and pass the utterance as a parameter in the **Link** block to the Natural Language Menu. The utterance is then passed to the NLU menu as an input in the callflow.

Support for System entities

Important

This feature is supported only for Google Dialogflow.

DialogFlow supports composite entities and returns them as part of the response. To allow use of these entities in a callflow, Intelligent Automation can use the values as a whole object or split the object into simpler components.

For example, in DialogFlow, we have a slot called *Duration* and the data type is `@sys.duration`, DialogFlow returns the following object as a result:

```
{"amount":10,"unit":"min"}
```

Intelligent Automation will store the object as a slot which can be accessed using a **Script** block.

```
Duration  = {"amount":10,"unit":"min"}
```

Intelligent Automation will also split the object as separate sub-slots which can be referenced in a **Question** block (see section Map Slots to questions). You can access each entity as *`<slotName_type> = value`*.

```
Duration_amount = 10
Duration_unit = min
```

Sample script

This script will parse the value of the variable, NLSlots and log the slot name, value and confidence.

```
def slots = context.getVariable("NLSlots")

slots.entrySet().each
{
    entry ->
    sSlotName = entry.key
    sSlotValue = entry.value
    sSlotValue.entrySet().each
    {
        valueEntry ->
        value = valueEntry.key
        fConfidence = valueEntry.value
        context.log("Slot info: Name: " + sSlotName + " Value: " + value.getValue() + "
Confidence " + fConfidence)
    }
}
```

Support for Follow-up Intents

Important

This feature is supported only for Google Dialogflow.

Intelligent Automation now supports follow-ups intents from Google Dialogflow. A follow-up intent is an intent that is tied to another earlier intent. Think of it as a child intent of an existing parent intent. When you create a follow-up intents in Dialogflow, Intelligent Automation can read and work with contexts for the intents.

Set up a module with a **Question** block. The Question block stores the intent in the **Prompt Wording** field. Ensure that this field has `var:NLTextResponse` as its value. In the **Question Options** tab, enable the **Store Answer as a variable** option. This holds the values received from DialogFlow. Ensure that the variable name is `utterance`.