User's Manual

AudioCodes Voice.AI Solutions

Voca CIC Flow Designer

Cloud-Based & On-Premises

Version 12.2

Notice

Date Published: May-11-2026

Security Vulnerabilities

All security vulnerabilities should be reported to vulnerability@audiocodes.com.

Customer Support

Customer technical support and services are provided by AudioCodes or by an authorized AudioCodes Service Partner. For more information on how to buy technical support for AudioCodes products and for contact information, please visit our website at https://www.audiocodes.com/services-support/maintenance-and-support.

Documentation Feedback

AudioCodes continually strives to produce high quality documentation. If you have any comments (suggestions or errors) regarding this document, please fill out the Documentation Feedback form on our website at https://online.audiocodes.com/documentation-feedback.

Stay in the Loop with AudioCodes

Related Documentation

Document Name
Voca CIC Release Notes
Voca CIC Administrator's Guide
Voca CIC Worker & Supervisor Application User's Manual

Document Revision Record

LTRT	Description
12996	Updated for Version 12.2

The latest software versions can be downloaded from AudioCodes' Services Portal (registered Customers only) at https://services.audiocodes.com.

Table of Contents

1 Introduction

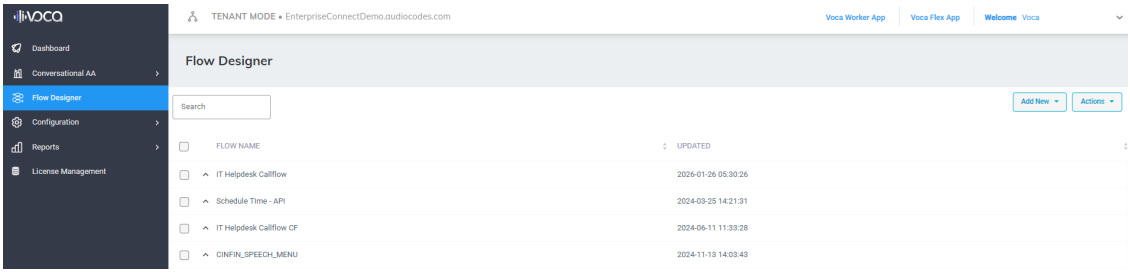
The purpose of this document is to familiarize Voca Administrators with the new Voca Flow Designer, which offers a new way to configure, design, and manage complex call flows. The Flow Designer provides a rich and powerful set of building blocks empowering users to create their own call flow scenarios.

2 Accessing Flow Designer

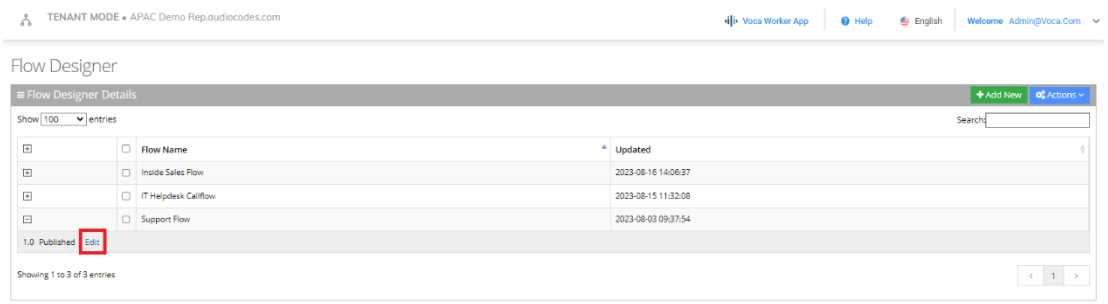
The Flow Designer page in the Voca application offers a way to configure, design, and manage complex call flows using a powerful set of building blocks.

➢ To access the Flow Designer:

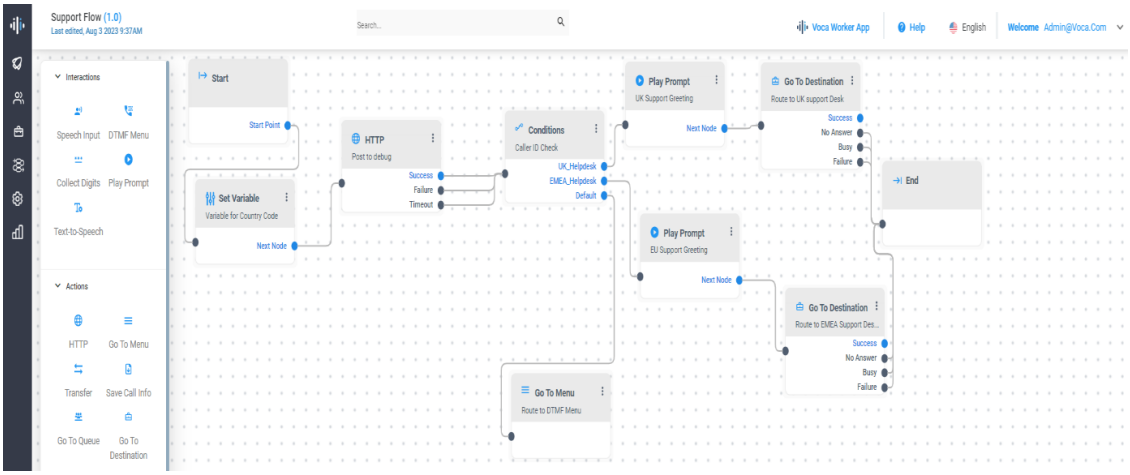
1. Log in to the Voca application.
2. From the Navigation pane, click **Flow Designer**; the Flow Designer page opens:



3. Select the flow that you want to edit, by clicking the corresponding plus box; the Edit link appears under the selected script:



4. Click **Edit**; the main flow designer workspace appears:



Add a new flow

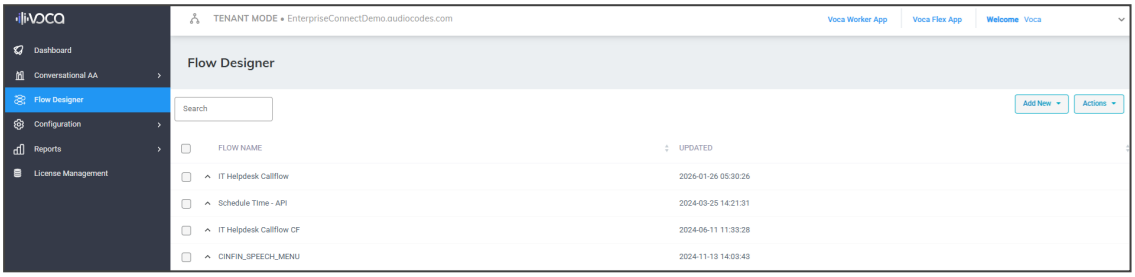
You can add a new flow from scratch or by using an existing template:

- Add a flow from scratch below
- Add a new flow from a template on the next page

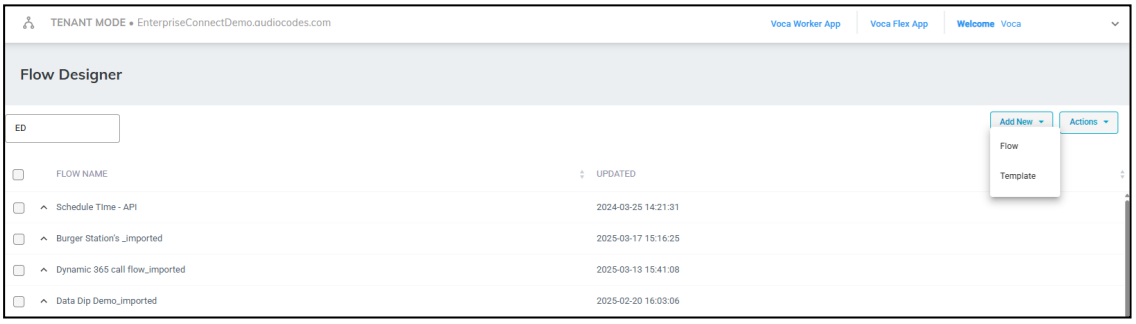
Add a flow from scratch

➢ To add a new flow from scratch:

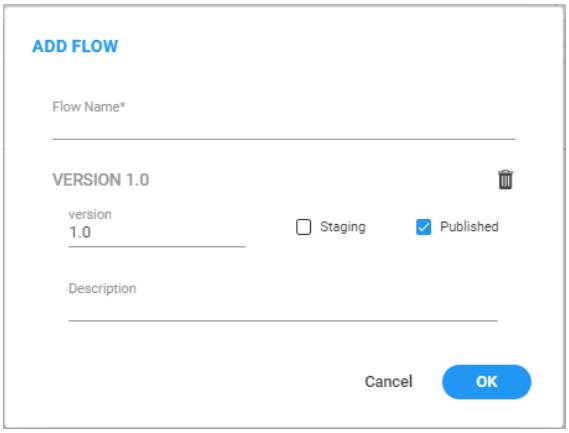
1. From the Navigation pane, click **Flow Designer**; the Flow Designer page opens:



2. Click **Add New**.



3. Select **Flow**.



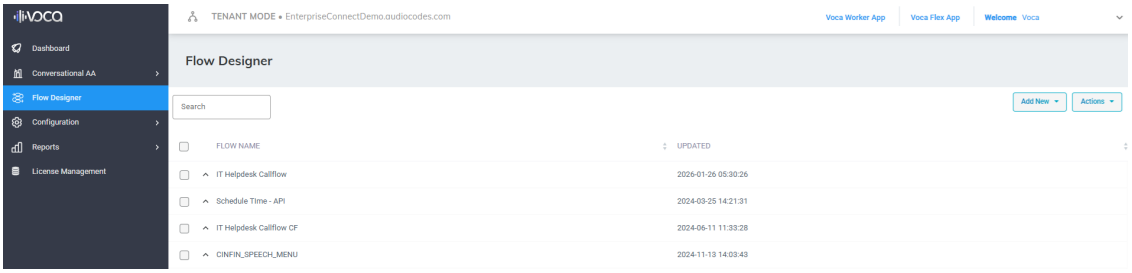
4. In the 'Flow Name' field, enter the name of the flow.
5. In the 'Version' field, enter the number of the version applicable to the flow.
6. Select the 'Staging' check box if the flow is still being developed.
7. Select the 'Published' check box if the flow has been completed and published.
8. In the 'Description' field, enter a description of the flow.

9. Click **OK**.

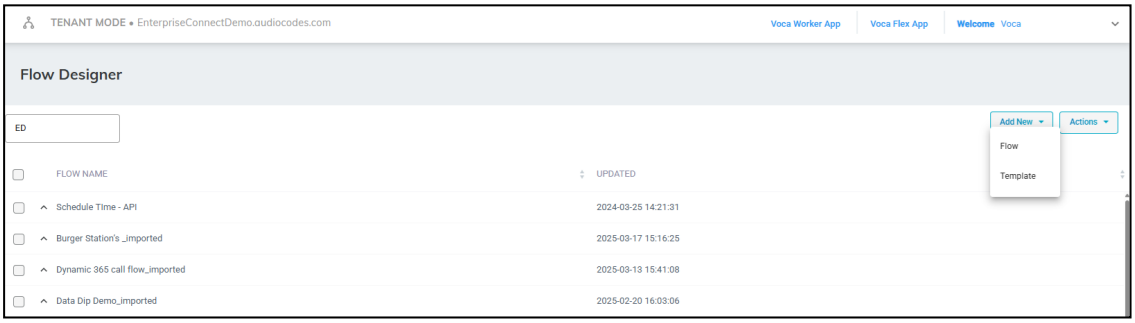
Add a new flow from a template

➢ To add a new flow from a template:

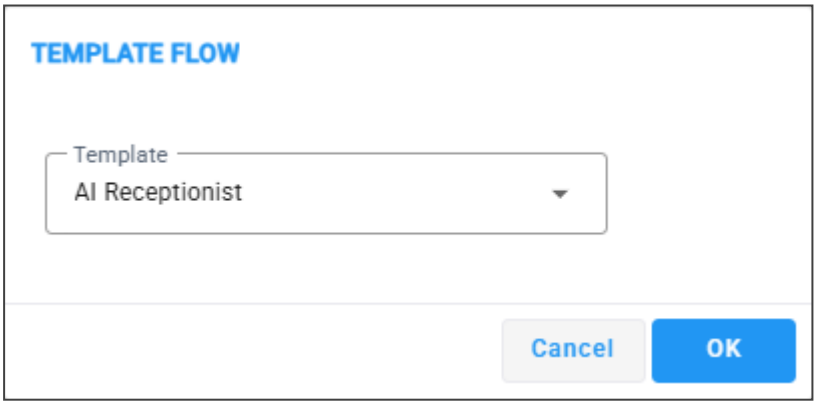
1. From the Navigation pane, click **Flow Designer**; the Flow Designer page opens:



2. Click **Add New**.



3. Select **Select Template**; the following appears:



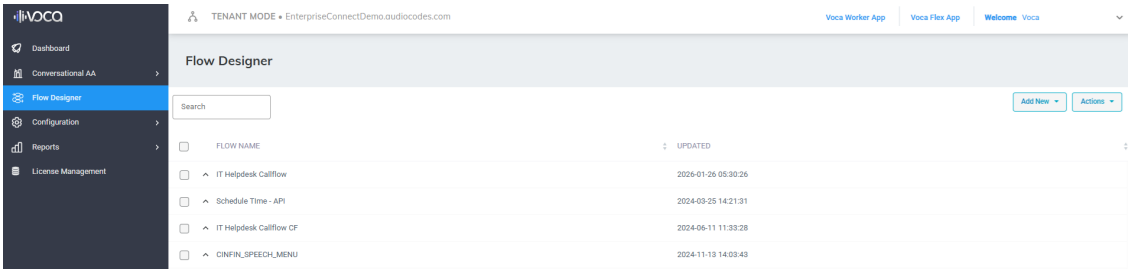
4. Select a template and click **OK**.

Edit a flow

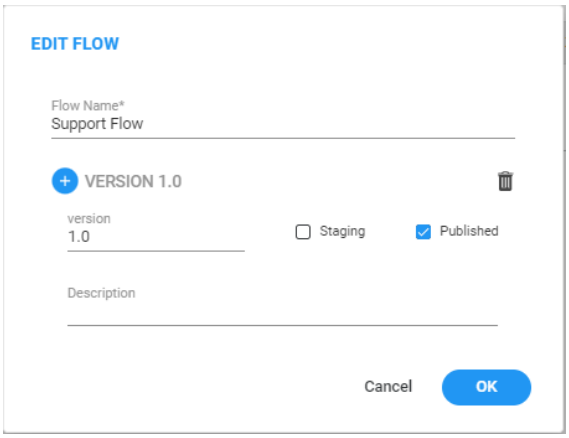
The procedure below describes how to edit a flow.

➢ To edit a flow:

1. From the Navigation pane, click **Flow Designer**; the Flow Designer page opens:

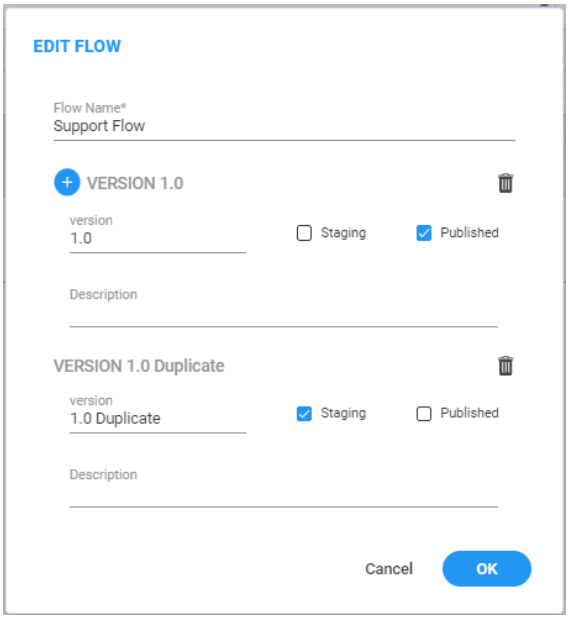


2. Select the flow you want to edit by enabling the Flow Name check box.
3. From the 'Actions' drop-down menu, choose **Edit**; the following appears:



4. It is possible to have more than one flow – for example, you can have one flow for **Staging** (in development), and one flow for **Published** (completed flow).

   To duplicate the flow, click the + button; the following appears:



Only one flow can be set to Published.

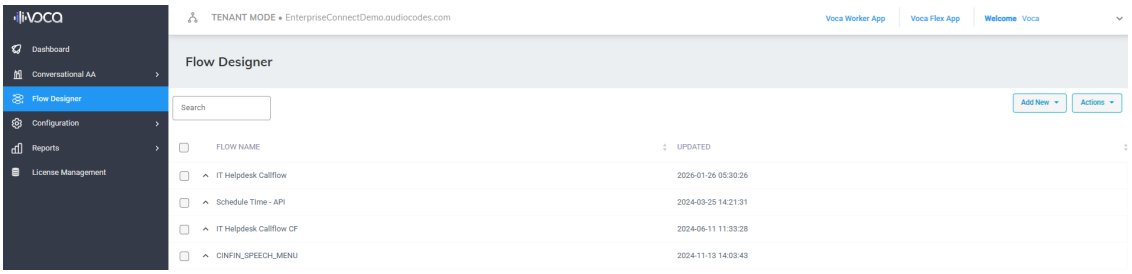
5. Click the delete icon to remove a duplicate flow.
6. Click **OK**.
7. On the Flow Designer page, select the flow entry you want to expand by clicking plus.
8. Select the flow you want to edit.

Delete a flow

The procedure below describes how to delete a flow.

To delete a flow:

1. From the Navigation pane, click **Flow Designer**; the Flow Designer page opens:



2. Select the flow you want to delete by enabling the 'Flow Name' check box.
3. From the 'Actions' drop-down menu, choose **Delete**; a message appears to confirm that you want to delete the selected flow.
4. Click **OK**.

3 Variable Syntax

The Flow Designer uses variables to store values collected or calculated during the call flow. Variable syntax across the Flow Designer should be in the following format:

${var_name}

The above syntax should be used when setting and reading from a variable and when using functions.

Variable name rules:

- Alphanumeric
- Underscore '_'
- Must start with a letter
- Case sensitive
- Maximum length of 24 characters

Predefined variables

The following are predefined variables. If specific attributes are passed to the script, they can be accessed in the same way.

- ${CLI} – Caller Line Identification
- ${DNIS} – Incoming call DNIS
- ${Call_ID} – Call Identification
- ${asrSessionId} – Stores the content of the header called X-ASRCall-Session-ID from the upcoming invite to Voca.

For variables that you set to control system behavior, see Reserved Variables below.

Reserved Variables

Reserved variables are fixed variables that control specific call behaviors when used within the **Set Variable** building block. By setting them in your flow using the **Set Variable** building block, you can override specific default behaviors during a call.

DISABLE_REC

Controls whether a call is recorded, regardless of the assigned agent's recording profile.

Example use case

During an IVR flow, the caller can be prompted via DTMF to consent to call recording. If the caller does not consent, use this variable to prevent recording when the call is routed to an agent.

Values	Behavior
“true”, “True”, “1”, “yes” or any non-empty string	Disables call recording, even if the caller reaches an agent whose profile is set to record calls.
"false", "FALSE", "0", or any empty string	Allows call recording if the caller reaches an agent whose profile is set to record calls.

- To apply this variable during the call, the flow must include a **Display Data** block after setting this variable.
- If this variable is not set, call recording follows the agent's recording profile by default.
- This variable does not persist if the call is transferred.

4 Expressions

The Flow Designer uses expressions as part of the building blocks. The following are different types of expressions that you can use:

- Arithmetic
- String
- Boolean

Arithmetic

You can enter expressions like:

- Adding numbers: `30 + 25.4`
- Context variables: `${ticketPrice} * ${tax}`

Arithmetic expressions can be applied for any arithmetic operation.

String

- You can use strings in expressions, for example, by using a Play Prompt\ Text-to-Speech to create the following text-to-speech prompt:

  ```
  "The name of the show you have selected is " + ${selectedShow}
  ```

- You can use strings in expressions, for example, by using a Play Prompt\Text-to-Speech to create the following text-to-speech prompt with two results:

  ```
  "The name of the show you have selected is " + ${selectedShow}
  + "and the price for the ticket will be " + ${ticketPrice}
  ```

Boolean

In the Conditions block, you need to enter an expression that produces a Boolean result. For example:

- `${ticketPrice} > 30`
- `${selectedShow} == "Friends"`
- `${selectedShow} == "Friends" && ${ticketType} == "VIP"`
- `${selectedShow} == "Friends" || ${selectedShow} == "Chernobyl"`
- `${ticketPrice} >= 30 || ${ticketPrice} <= 30`

Voca supports the following Boolean operators:

- Greater / less than:
  - >

- <

- Greater / less than or equals to:
  - >=
  - <=
- Equals / not equals:
  - ==
  - !=

5 Supported Functions

The following are different types of supported functions that can be used:

Contains

The Contains function is used to determine the presence of a specific search text within a source text string. If the search text is found within the source text, the function returns 'True'; otherwise, it returns 'False.' The Contains function is case-sensitive. This means the search treats upper and lowercase letters differently. Be aware of case sensitivity when utilizing this function as it can impact the results based on the letter casing of the text being examined.

Syntax

```
Contains(source text string, search text string)
```

Example

```
Contains("Flight to New York","New York") -> returns 'True'
Contains("Flight to New York","NEW YORK") -> returns 'False'
```

Date

The Date function is used for manipulating dates and times. It receives a text string as input and returns it as a date object using the format 'mm/dd/yyyy hh:mm:ss tt', representing month/day/year, and time in a 12-hour clock format. The function also includes robust error handling, returning NULL for non-string inputs or when encountering parsing issues. This ensures reliable and predictable behavior even with unexpected input. If no date is provided, it defaults to returning the current time with the current date.

Syntax

```
Date(text string)
```

Example

```
Date("08/24/2023 16:22:53") -> returns '08/24/2023 4:22:53 PM'
Date("2023-08-24T04 :22 :53.53") -> returns '08/24/2023 4:22:53 AM'
Date("2023-08-25T00 :00 :00") -> returns '08/25/2023 12:00:00 AM'
Date("Monday, 24 August 2023") -> returns '08/24/2023 12:00:00 AM'
Date("08/24/2023") -> returns '08/24/2023 12:00:00 AM'
Date("16 :33 :44") -> returns '08/24/2023 4:33:44 PM'
```

DateConvert

The DateConvert function receives two inputs: a date/time string, and a the desired output date\time format. The function converts the input string to the requested format and returns the new string in the specified format.

If the provided input is not a valid string, or if the converting process fails (for example, the text is not in a valid date format), the function returns NULL.

Syntax

```
DateConvert(date/time text string, requested output date/time format text string)
```

Example

```
DateConvert("09/03/2023","dd/MM/yyyy")-> returns '03/09/2023'
DateConvert("09/03/2023","dddd, dd MMMM yyyy")-> returns 'Sunday, 03 September 2023'
DateConvert("09/03/2023 03:45:00 PM","yyyy-MM-ddTHH:mm:ss")-
```

```
> returns '2023-09-03T15:45:00
DateConvert("16:53:00","hh:mm tt")-> returns '04:53 PM'
```

DateParse

The DateParse function manipulates date objects. It receives two key parameters: a date object and a date format string. This function transforms the input date object to a customized date representation based on the provided format string, returning it as a string. If input errors occur, such as incompatible date formats, it returns NULL, ensuring dependable behavior.

Syntax

```
DateParse(date object, requested date format)
```

Example

If there is a ‘date‘ variable saved with the value of “08/24/2023 4:22:53 PM”, the function returns:

```
DateParse (${date},"dd/MM/yyyy") - > returns '24/08/2023'
```

GetJsonValue

The GetJsonValue function receives two input parameters: a JSON object, and a string specifying the desired path within the JSON. This function takes the provided JSON and attempts to retrieve the value located at the specified path. If the path exists and a valid value is found, the function returns that value. However, if the path cannot be found, or if any parsing errors occur during the process (e.g., invalid JSON syntax), the function returns NULL as a result. This ensures safe handling of input data and provides a consistent response when encountering issues with the JSON structure or path.

Syntax

```
GetJsonValue(JSON Object, JSON path string)
```

Example

Assuming we have the following JSON that is saved in the context under the variable name ‘studentsList’:

```
{
  "name": "John Doe",
  "age": 30,
  "still.student": false,
  "address": {
```

```
    "street": "123 Main Street",
    "city": "Anytown",
    "state": "CA"
  },
  "friends": [{
      "name": "Alice",
      "age": 28
    },
    {
      "name": "Bob",
      "age": 32
    }
  ]
}
```

To get the value from the key that called "name", use the function as follows:

`GetJsonValue(${studentsList},"name")-> returns 'Jhon Doe'`

To get the value from the key that called "city", use the function as follows:

`GetJsonValue(${studentsList},"address.city")-> returns 'Anytown'`

To get the value from the key that called "age" that associated to Alice, use the function as follows:

`GetJsonValue(${studentsList},"friends[0].age")-> returns '28'`

To get the value from the key that called "still.student", use the function as follows:

`GetJsonValue(${studentsList},"[still.student]")-> returns 'false'`

Length

The Length function receives a single text string as input and returns an integer value representing the length of that string. If the input string is empty, the function will return '0'. This ensures consistent behavior and provides a convenient way to determine the length of text while handling empty strings gracefully. If the input is not a string type, or if any issues arise during the length calculation, the function returns NULL, ensuring reliability and predictability in its output.

Syntax

`Length(string text)`

Example

`Length("Voca") -> returns '4'`

`Length("New York") -> returns '8'`

Lower

The Lower function works with text strings. It receives a single text string as input and returns all the characters within it to lowercase. If the provided input is not of string type or if any issues arise during the lowercase converting process, the function will return NULL. This robust error handling ensures a reliable and predictable response even in cases of unexpected input or converting challenges.

Syntax

```
Lower(text string)
```

Example

```
Lower("New York") -> returns "new york"
```

Now

The Now function returns the current date and time in a specific tenant time zone as a date object, represented as 'MM/dd/yyyy hh:mm:ss tt'. It provides accurate timestamping and handles unexpected errors by returning NULL when issues arise during the process.

Syntax

```
Now()
```

Example Usage

```
Now()
```

NowUtc

The ‘NowUtc’ function returns the current date and time in a UTC time zone as a date object, represented as 'MM/dd/yyyy hh:mm:ss tt.' It provides accurate timestamping and handles unexpected errors by returning NULL when issues arise during the process.

Syntax

```
NowUtc()
```

Example

```
NowUtc()
```

Replace

The Replace function manipulates text strings by allowing users to replace specific characters or substrings. The Replace function operates in a case-sensitive manner. With its intuitive syntax and dynamic input options, the Replace function empowers users to enhance productivity and streamline data processing workflows effectively.

Syntax

```
Replace(Source text string, Old text string, New text string)
```

Example

```
Replace("I like bananas", "bananas", "apple") -> returns "I like apples"
```

SubString

The SubString function allows you to manipulate and extract specific portions of text or characters from a larger string of content.

The substring function can work in two different ways:

- When provided with a string text, a start index (an integer), and an optional length (also an integer), it returns a portion of the string text that starts at the specified start index and extends for the given length, if provided.
- If the length is not provided, the function returns the portion of the string text from the start index to the end of the text.

The SubString function ensures reliability and predictability by returning NULL in the following scenarios:

- If the provided input text is not of string type.
- If the start index is equal to or greater than the length of the provided text, indicating an invalid starting point.

Syntax

```
SubString(string text, integer start index)
SubString(string text, integer start index, integer length)
```

Example

```
SubString("How are you?",8) -> "you?"
SubString("How are you?",4,3) -> "are"
```

Given that there is a ‘text’ variable saved in the context with the value of “How are you”, you can also use the function as follows:

```
SubString(${text},8) -> "you?"
```

```
SubString(${text},4,3) -> "are"
```

Trim

The Trim function is used for text manipulation, enabling the removal of spaces from a given text string. It accommodates both a text string and an optional direction as input parameters, allowing users to precisely control the trimming process. When executed, this function returns a modified version of the input text, with spaces eliminated according to the specified direction. If no direction is explicitly provided, the default behavior is to remove all spaces ('ALL'):

- LEFT
- RIGHT
- ALL

Syntax

```
Trim(string text, string direction)
```

Example

```
Trim(" My name is? ","LEFT")-> returns 'My name is? '
```

```
Trim(" My name is? ","RIGHT")-> returns ' My name is?'
```

```
Trim(" My name is? ","ALL")-> returns 'My name is?'
```

```
Trim(" My name is? ","")-> returns 'My name is?'
```

Upper

The Upper function works with text strings. It receives a single text string as input and transforms all the characters within it to uppercase. If the provided input is not of string type or if any issues arise during the uppercase converting process, the function will return NULL. This robust error handling ensures a reliable and predictable response even in cases of unexpected input or converting challenges.

Syntax

```
Upper(string text)
```

Example

```
Upper("New York") -> returns 'NEW YORK'
```

WeekDay

The WeekDay function is used to determine the day of the week corresponding to a given date. It accepts a date object as input and returns the corresponding day as an integer (0 for Sunday to 6 for Saturday). Robust error handling is in place to ensure reliability, with the function returning NULL if the input is not a date object or if any issues arise during the converting process.

Syntax

```
WeekDay(Date date)
```

Example

```
WeekDay(NowUtc())
```

```
WeekDay(Date("09/03/2023")) -> returns '0'
```

```
WeekDay(Date("09/15/2023")) -> returns '5'
```

6 Building Blocks

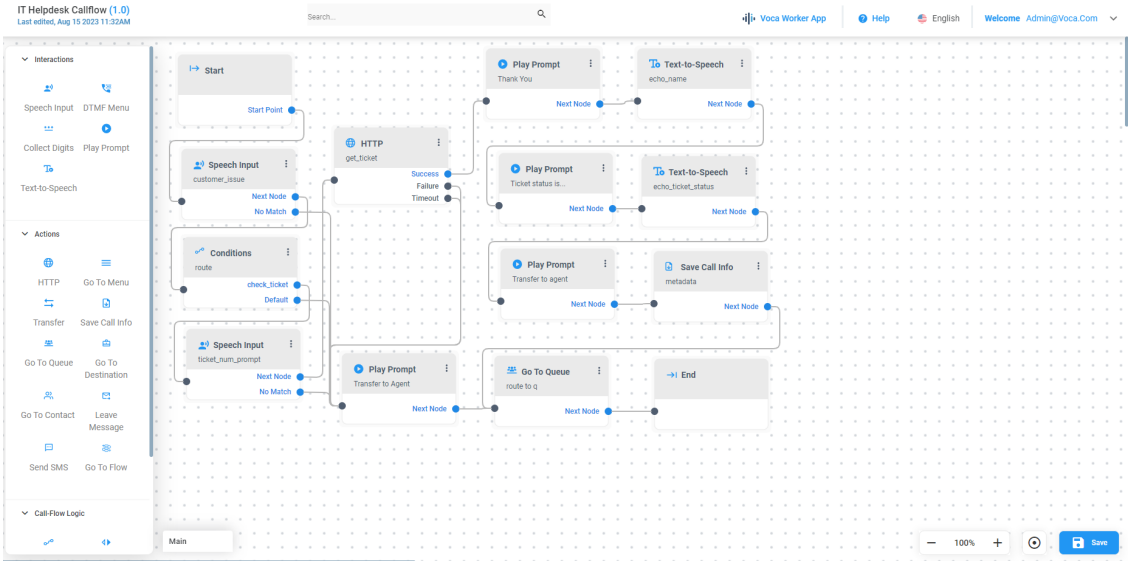
Use the following building blocks to create the call flow logic:

- Interactions below
- Actions on page 45
- Call-Flow Logic on page 66

You can connect the building blocks by placing the cursor on the output leg, and then dragging the connector to the appropriate actions.

➢ To add a building block to the flow designer:

1. From the left bar, click the appropriate building block; the selected building block appears on the flow designer workspace.
2. Drag the building block to the desired position on the flow designer workspace.
3. Drag the relevant nodes to each appropriate building block, to connect the flow.



Interactions

The following building blocks appear under **Interactions**.

- Speech Input
- DTMF Menu
- Collect Digits
- Play Prompt
- Text-to-Speech
- OpenAI

Speech Input

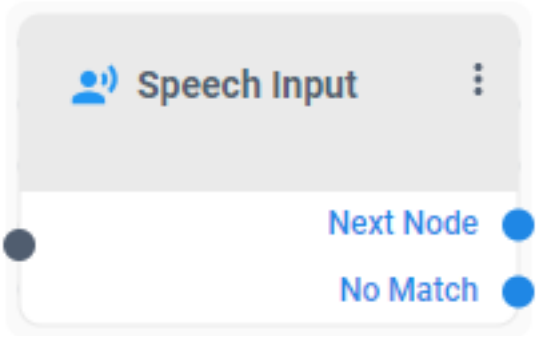
The 'Speech Input' building block seamlessly integrates speech recognition into your system, enabling callers to engage in natural interactions through spoken commands. During the Speech Recognition step, it not only interprets spoken input but also allows it to be mapped to variables. These variables can then later be leveraged to route calls based on their values. This significantly enhances user experience by facilitating intuitive voice-driven interactions, thereby improving the accessibility and efficiency of menu navigation and selections.

There are two results when routing a call with the 'Speech Input' building block, depending on how well it understands your voice command.

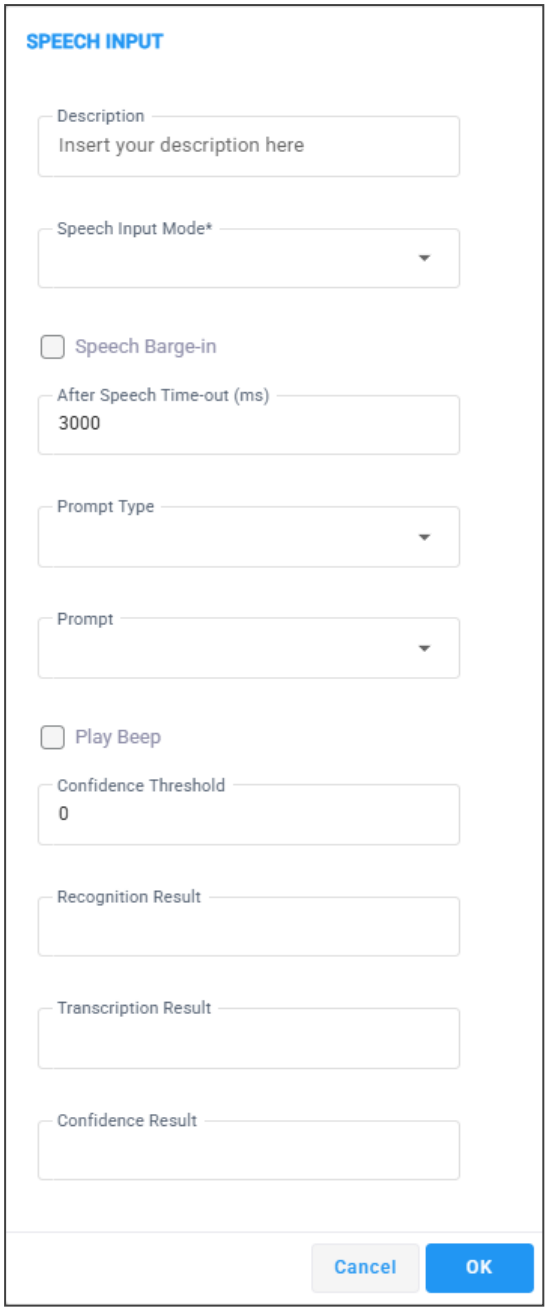
- **Next Node**: If the system understands the voice command enough (confidence is above the threshold), it will automatically send your call to the next step.
- **No Match**: If the system can't understand your voice command clearly (confidence is below the threshold), it will send your call to a different step called "No Match."

➢ To use the Speech Input building block:

1. On the left pane, under **Interactions**, click the **Speech Input** option; the following Speech Input building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. From the 'Speech Input Mode' drop-down list, select the preferred mode:
   - **Free Speech**: The system records the caller's input and then checks the confidence score. If the confidence score is below the preconfigured threshold (by default configured '0'), the system activates 'No Match'.
   - **Keywords**: The system records the caller's input and then cross-references it with a predefined list of up to 50 phrases that can be configured in the building block. The caller's speech is checked against this list, aiming to identify the phrase with the highest confidence score. If the confidence score is below the preconfigured threshold (by default configured '0'), the system activates 'No Match'.

     When selecting this mode, an additional field, ‘Keywords*’, becomes available.

- **Digits**: The system records the caller's input and subsequently attempts to convert this input into a numerical character. If the Automatic Speech Recognition (ASR) result cannot be successfully converted into a numerical character, or if the confidence score falls below the pre-configured threshold (by default configured '0'), the system activates 'No Match'.

  When selected, an additional field called ‘Min Digits’ and ‘Max Digits’ become available. Enter the minimum and the maximum length of the digits that the caller can input.

This mode is currently in the Beta phase and is exclusively available for the EN-US dialect.

- **Alpha-Numeric**: The system records the caller's input and subsequently attempts to convert this input into an alphabetical and numerical character. If the Automatic Speech Recognition (ASR) result cannot be successfully converted into an alphabetical and numerical character, or if the confidence score falls below the pre-configured threshold (by default configured '0'), the system activates 'No Match'.
  - When selected, an additional field called ‘Alpha-Numeric Pattern’ becomes available. Define both the length and sequence of the alphanumeric pattern you expect to receive. The syntax for available patterns includes:
    - '\s' – Indicates an expectation for an alphabetical character.
    - '\d' – Indicates an expectation for a numeric character.
    - '\a' – Indicates an expectation for an alphabetical or numerical character.

  Example:

  Note this pattern: \s\s\d\d\d\a. This signifies that you anticipate a response with a length of 6 characters, starting with 2 numeric characters, followed by 3 alphabetical characters, and ending with one alphabetical or numerical character. Any other response will be categorized as a "No Match". This allows you to set exactly what format you expect the input to be in.

This mode is currently in the Beta phase and is exclusively available for the EN-US dialect.

- **Date**: The system records the caller's input and subsequently attempts to convert this input into a date with the following format: 'yyyy-MM-dd'. If the Automatic Speech Recognition (ASR) result cannot be successfully converted into a date, or if the confidence score falls below the pre-configured threshold (by default configured '0'), the system activates 'No Match'.

When selected, an additional field called ‘Hint’ becomes available. This field allows you to specify whether the system-generated date should be in the past, or in the future, in the event that the customer does not mention the year.

This mode is currently in the Beta phase and is exclusively available for the EN-US dialect.

- **Credit Card Number**: The system records the caller's input and utilizes an algorithm to verify the accuracy of the identification number provided by the caller. If the system confirms the identification number as correct, it proceeds to the 'Next Node'. However, if the system detects an incorrect identification number, or if the confidence score falls below the pre-configured threshold (by default configured '0'), the system activates 'No Match'.

  The result destination information is returned in JSON format, encompassing all the relevant details associated with the selected destination.

- **Credit Card Expiration Date**: The system records the caller's input and utilizes an algorithm to verify the credit card expiration date provided by the caller. If the system confirms the credit card expiration date as correct, it proceeds to the 'Next Node'. However, if the system detects an incorrect identification number, or if the confidence score falls below the pre-configured threshold (by default configured '0'), the system activates 'No Match'.

- **Destination Dictionary**: The system records the caller's input and proceeds to compare it with a designated destination dictionary (contact/department list).

  It analyzes the caller's spoken input against this dictionary, aiming to identify the phrase with the highest confidence score. In cases where the confidence score falls below the pre-configured threshold (by default configured '0'), the system activates the 'No Match' state.

  When activated, an additional parameter called ‘Destination Dictionary’ becomes available. This parameter allows the users to specify the dictionary against which they want to perform the comparison.

The result destination information is returned in JSON format, encompassing all the relevant details associated with the selected destination.

JSON format (for Department):

```
{
 "EntityID": "",
 "Type": " DESTINATION",
 "Recording": "",
 "Destination": "",
 "AliasPrompt": "",
```

```
 "Extension1": "",
 "Extension2": "",
 "Extension3": ""
}
```

JSON format (for Contact):

```
{
 "EntityID": "",
 "Type": " CONTACT",
 "Recording": "",
 " FN": "",
 " LN": "",
 "Extension": "",
 "Mobile": "",
 "Dect": ""
}
```

5. Enable the 'Speech Barge-In' feature by selecting the checkbox. Activating this option grants callers the ability to interrupt the prompt mid-way using speech, eliminating the necessity to wait for the prompt to finish.
6. In the **After Speech Time out (ms)** field, define how long the system should wait, in milliseconds, after detecting the end of a callers speech before finalizing speech recognition. When the timeout expires without detecting additional speech, the system assumes the utterance is complete and proceeds to process the recognized input.
7. From the 'Prompt Type' drop-down list, select the appropriate prompt type:
   - **User Prompt**: Gives you the capability to designate a specific fixed prompt that will unfailingly play whenever a call is directed through the Speech Input building block.

     When selected, an additional field called "Prompt" becomes available. This field allows you to specify the prompt you want to utilize when the call is routed through the Speech Input building block.
   - **Dynamic Prompt**: By using this prompt type, you gain the flexibility to dynamically adjust the prompt based on previous actions taken by the caller within the flow. When selected, an additional field called "Value" becomes available. This field allows you to specify the name of your prompt as it appears in the system prompt list, using a variable.

If the specified prompt is not found, the system pauses briefly before continuing to the next step.

Example:

If you have a single flow that can be triggered from different DID numbers, and you want to change the prompt based on the dialed number. To do this, configure a “Conditions” building block (see Conditions building block for more information) before running the Speech Input building block. This condition block checks the DID number and routes the call to the 'Set Variables' building block accordingly (see Set Variable building block for more information). The 'Set Variables' block stores the prompt name. This variable can then be used to play distinct prompts for different DID numbers, providing dynamic caller experiences.

- **Alphanumeric Phonetic**: The Alphanumeric Phonetic mode enables the system to interpret caller input spoken using phonetic representations of letters and numbers. In this mode, the system captures the caller’s spoken response and applies a normalization process that converts phonetic alphabet terms (for example “Alpha,” “Bravo,” “Charlie”) and spoken digit words (for example “Zero,” “One,” “Seven”) into their corresponding alphanumeric characters (A–Z, 0–9).

  This mode enhances recognition accuracy by supporting callers who naturally provide information using phonetic pronunciation. It improves input reliability without altering the behavior of the existing Alphanumeric (beta) mode.

  Example:

  A spoken input of “Alpha Bravo Three Seven Zulu” is normalized to “AB37Z.”

This mode is currently in the Beta phase and is exclusively available for the EN-US dialect.

8. In the 'Confidence Threshold' field, you can configure the threshold for confidence. Calls falling below this confidence threshold are routed to 'No Match'. By default, this field has the value of '0'.
9. Enable the 'Play Beep' option by selecting the corresponding check box. This triggers the playing of a beep sound before the system begins to collect the user's response.
10. In the 'Recognition Result' field, specify a variable in the format ${var_name} to store the speech input result. If the call is routed to 'No Match', the system automatically replaces the recognition result with "No Match".
11. In the 'Transcription Result' field, specify a variable in the format ${var_name} to store the original speech input result before the system processing takes place. This variable allows you to access the original speech input content, even if the call is routed to 'No Match'.
12. In the 'Confidence Result' field, define a variable in the format ${var_name} to capture the confidence score identified by the Automatic Speech Recognition (ASR) system. This score can be utilized later to inform decision-making based on the obtained result.
13. Click **OK**, and then **Save**.

DTMF Menu

The 'DTMF Menu' building block empowers you to offer callers a menu of choices. As an admin, you have the flexibility to select from two input methods tailored to your users' preferences and needs:

- **DTMF**: The DTMF method exclusively employs DTMF (Dual-Tone Multi-Frequency) input, providing callers with a specific interaction mode that does not include the option to utilize ASR (Automatic Speech Recognition).
- **DTMF with Speech**: The DTMF and Speech method offers callers a versatile communication experience, providing two distinct options for interaction. Callers can choose to use DTMF (Dual-Tone Multi-Frequency) input, allowing them to enter information or make selections using the keypad on their phone. Alternatively, callers can take advantage of the ASR (Automatic Speech Recognition) capability, enabling them to interact with the system by speaking naturally. This flexibility empowers callers to engage with the system in the manner that best suits their preferences and needs.

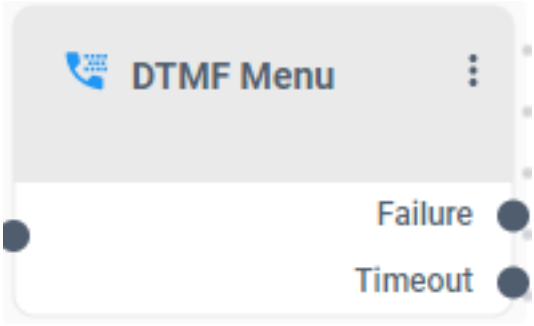
Without Speech

The 'DTMF Menu' building block has 2 exit legs:

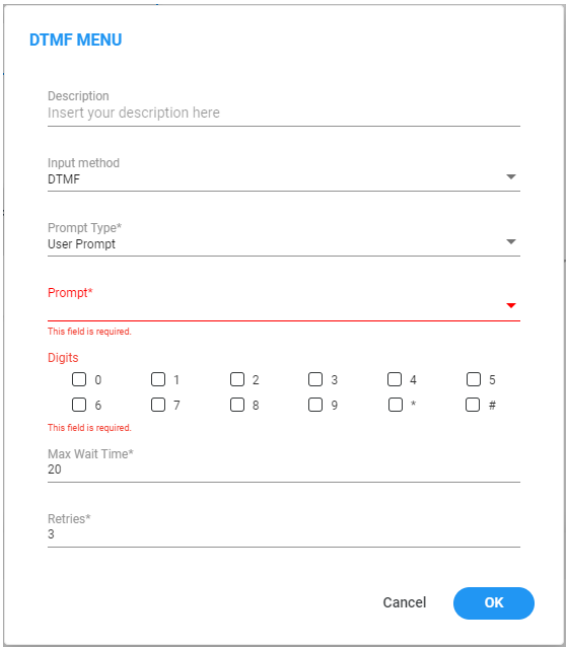
- Failure: Indicates when the system recognizes a DTMF key that is not configured as an option on the menu, and the maximum number of retries has been reached.
- Timeout: Indicates when the system fails to recognize any DTMF input and the maximum number of retries has been reached.

➢ **To use the DTMF Menu building block without speech method:**

1. On the left pane, under **Interactions**, click **DTMF Menu**; the following DTMF Menu building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description of this building block (up to 50 characters).
4. In the 'Input Method' select 'DTMF'.
5. From the 'Prompt Type' drop-down list, select the appropriate prompt type:

   - **User Prompt**: Gives you the capability to designate a specific fixed prompt that will unfailingly play whenever a call is directed through the Speech Input building block.

     When selected, an additional field named "Prompt" becomes available. This field allows you to specify the prompt you want to utilize when the call is routed through the Speech Input building block.

   - **Dynamic Prompt**: Gives you the flexibility to dynamically adjust the prompt based on previous actions taken by the caller within the flow. When selected, an additional field called "Value" becomes available. This field allows you to specify the name of your prompt as it appears in the system prompt list, using a variable.

If the specified prompt is not found, the following message appears: "We are experiencing system issues. Please call back later." The call will then be disconnected.

Example

If you have a single flow that can be triggered from different DID numbers, and you want to change the prompt based on the dialed number. To do this, configure a “Conditions” building block (see Conditions building block for more information) before running the Speech Input building block. This condition block will check the DID number and route the call to the 'Set Variables' building block accordingly (see Set Variable building block for more information). The 'Set Variables' block stores the prompt name. This variable can then be used to play distinct prompts for different DID numbers, providing dynamic caller experiences.

6. In the Digits section, select the specific digits you want to include in the DTMF menu. Available options range from 0 through 9, *, and #.

   Each digit you select is automatically added as an exit leg.

7. In the 'Max Wait Time' field, enter the maximum waiting time for user input, between 1 to 45 seconds. The default is 20 seconds.

8. In the 'Retries' field, specify the maximum number of retries to repeat the block if DTMF input is not detected. You can set this value between 1 and 10. The default is 3 retries.

9. Click **OK**, and then **Save**.

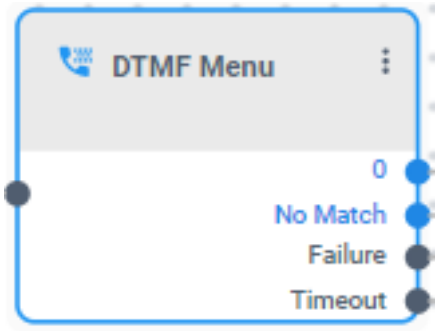
With Speech

The DTMF Menu building block has 3 exit legs:

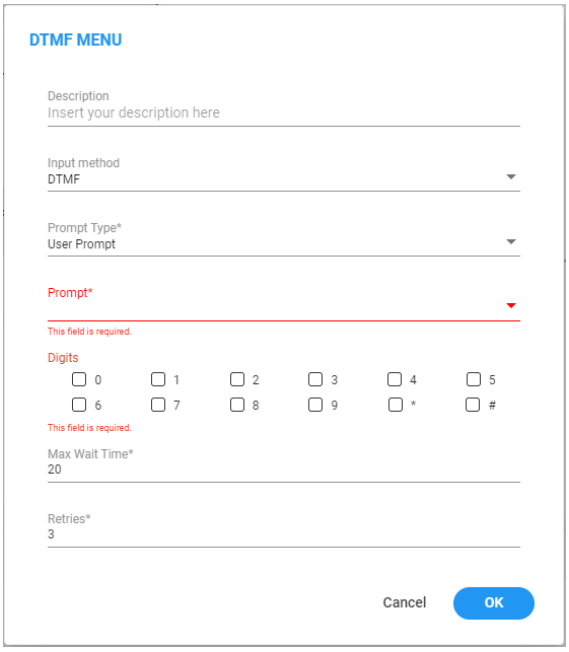
- **Failure**: When the system recognizes a DTMF key that is not configured as an option on the menu, and the maximum number of retries has been reached.
- **Timeout**: When the system fails to recognize input from the caller, whether it's DTMF or speech, and the maximum number of allowed retries has been reached.
- **No Match**: When the confidence result of the speech recognition is '0', the system will route the call to 'No Match'.

➢ **To use the DTMF Menu building block with speech method:**

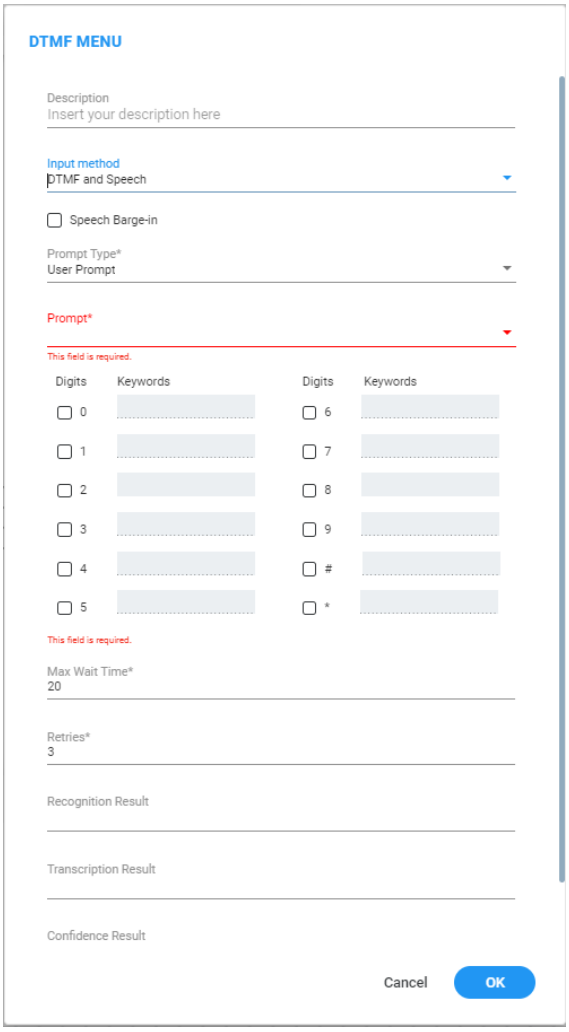
1. On the left pane, under **Interactions**, click **DTMF Menu**; the following DTMF Menu building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block, (up to 50 characters).
4. In the 'Input Method' change to 'DTMF and Speech'; the following appears:



5. You can enable 'Speech Barge-In' by selecting the corresponding checkbox. Activating this option grants callers the ability to interrupt the prompt mid-way using either speech or DTMF input, eliminating the necessity to wait for the prompt to finish.
6. From the 'Prompt Type' drop-down list, select the appropriate prompt type:
   - **User Prompt**: Gives you the capability to designate a specific fixed prompt that will unfailingly play whenever a call is directed through the Speech Input building block. When selected, an additional field called 'Prompt' becomes available. This field allows you to specify the prompt you want to utilize when the call is routed through the Speech Input building block.
   - **Dynamic Prompt**: Gives you the flexibility to dynamically adjust the prompt based on previous actions taken by the caller within the flow. When selected, an additional field called "Value" becomes available. This field allows you to specify the name of your prompt as it appears in the system prompt list, using a variable.

If the specified prompt is not found, the following message appears: "We are experiencing system issues. Please call back later." The call will then be disconnected.

Example

If a single flow that can be triggered from different DID numbers, and you want to change the prompt based on the dialed number. To do this, configure a "Conditions" building block (see Conditions building block for more information) before running the Speech Input building block. This condition block will check the DID number and route the call to the 'Set Variables' building block accordingly (see Set Variable building block for more information). The 'Set Variables' block stores the prompt name. This variable can then be used to play distinct prompts for different DID numbers, providing dynamic caller experiences.

7. In the Digits/Keywords section, customize the DTMF menu by selecting the digits (0-9, *, and #) and associating keywords with each digit. This feature enables callers to either press the specified digit or verbally say the associated keyword, directing them through the corresponding digit leg.

   To add a new keyword, enter the keyword name and then press enter.
8. In the 'Max Wait Time' field, enter the maximum waiting time for user input, between 1 to 45 seconds. The default is 20 seconds.

The Automatic Speech Recognition (ASR) system can record audio for a maximum of 1 minute. If the combined length of the prompt and the maximum wait time exceeds one minute, the system will proceed to the next building block after the one-minute mark.

9. In the 'Retries' field, specify the maximum number of retries to repeat the block if DTMF input is not detected. You can set this value between 1 and 10. The default is 3 retries.

10. In the 'Recognition Result' field, specify a variable in the format ${var_name} to store the speech input/DTMF key result. If the call is routed to 'No Match', the system automatically replaces the recognition result with "No Match."
11. In the 'Transcription Result' field, specify a variable in the format ${var_name} to store the original speech input result before any system processing takes place. This variable allows you to access the original speech input content, even if the call is routed to 'No Match'.
12. In the 'Confidence Result' field, define a variable in the format ${var_name} to capture the confidence score identified by the Automatic Speech Recognition (ASR) system. This score can be utilized later to inform decision-making based on the obtained result.
13. Enable the 'Play Beep' option by selecting the corresponding check box. This triggers the playing of a beep sound before the system begins to collect the user's response.
14. Click **OK**, and then **Save**.

Collect Digits

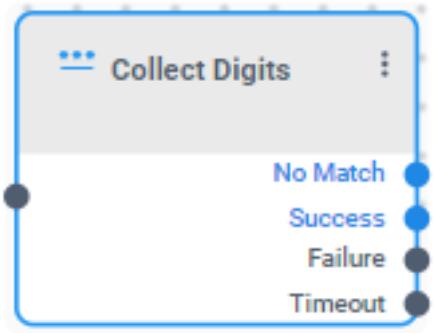
'Collect Digits' building blocks enable you to offer callers the option to enter multiple DTMF digits in response to a prompt. Additionally, as an admin, you can enable ASR (Automatic Speech Recognition), further enhancing the overall flexibility and experience for callers.

The 'Collect Digits' building block has 4 exit legs:

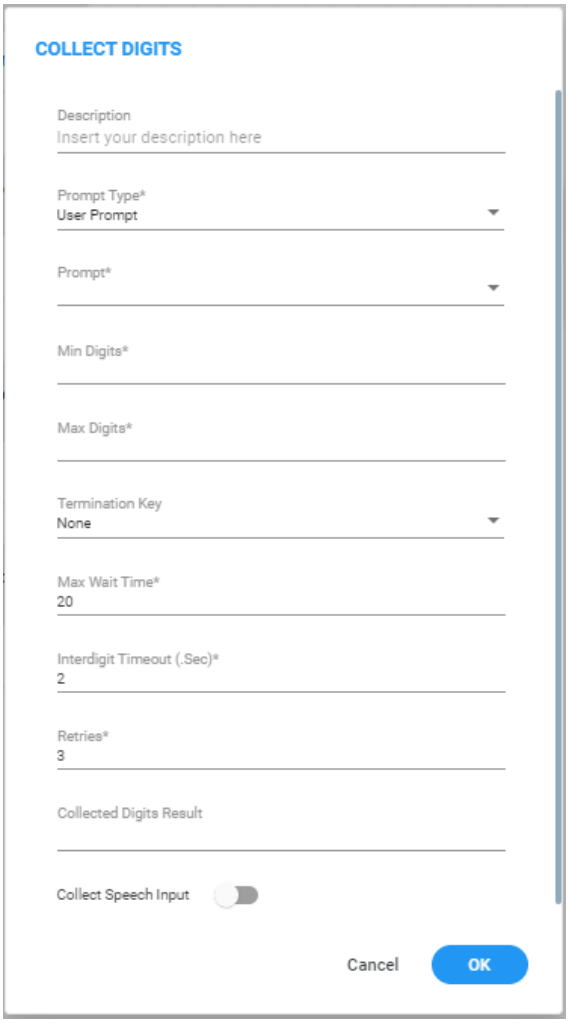
- **Success**: When the system successfully collects all the spoken or dialed digits from the caller, and the number of digits aligns with the criteria specified in the 'Min Digits' and 'Max Digits' settings within the building block, the call is automatically routed through this leg.
- **Failure**: If the system has gathered all the spoken or dialed digits from the caller but the number of digits does not align with the criteria defined in the 'Min Digits' and 'Max Digits' settings within the building block, the call is automatically directed through this leg.
- **Timeout**: If the system is unable to recognize input from the caller, whether it's DTMF or speech, and the maximum number of allowed retries has been reached, the call will be automatically directed through this leg.
- **No Match**: When the confidence result of the speech recognition is '0', the system will intelligently route the call to 'No Match'. This leg is only applicable after enabling the 'Collect Speech Input'.

➢ To use the Collect Digits building block:

1. On the left pane, under **Interactions**, click **Collect Digits**; the following Collect Digits building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. From the 'Prompt Type' drop-down list, select the appropriate prompt type:
   - **User Prompt**: Gives you the capability to designate a specific fixed prompt that will unfailingly play whenever a call is directed through the Speech Input building block.

When selected, an additional field called "Prompt" becomes available. This field empowers you to specify the prompt you want to utilize when the call is routed through the Speech Input building block.

- **Dynamic Prompt**: Gives you the flexibility to dynamically adjust the prompt based on previous actions taken by the caller within the flow. When selected, an additional field called "Value" becomes available. This field allows you to specify the name of your prompt as it appears in the system prompt list, using a variable.

If the specified prompt is not found, the following message appears: "We are experiencing system issues. Please call back later." The call will then be disconnected.

Example:

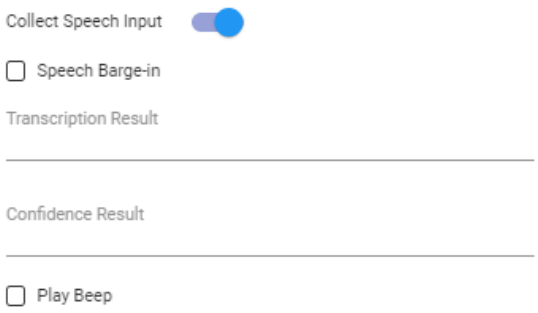
If you have a single flow that can be triggered from different DID numbers, and you want to change the prompt based on the dialed number. To achieve this, you can configure a “Conditions” building block (see Conditions building block for more information) before running the Speech Input building block. This condition block will check the DID number and route the call to the 'Set Variables' building block accordingly (refer to the Set Variable building block for more information). The 'Set Variables' block will store the prompt name. This variable can then be used to play distinct prompts for different DID numbers, providing dynamic caller experiences.

5. In the 'Min Digits' field, specify the minimum number of digits to be collected, ranging from 1 to 30 digits.
6. In the 'Max Digits' field, enter the maximum number of digits to be collected, between 1 to 30 digits. Verify that this number is higher than the 'Min Digits' setting.
7. From the 'Termination Key' drop-down list, select the appropriate termination key type:
   - **'None'**: The system will automatically stop collecting digits when the interdigit timeout parameter is reached, and no termination key is required.
   - **'*'**: To stop digit collection, the caller can simply press the '*' key.
   - **'#'**: To stop digit collection, the caller can simply press the '#' key.
8. In the 'Max Wait Time' setting, specify the maximum duration for awaiting user input. This duration can be set between 1 to 45 seconds. The default is 20 seconds.

The Automatic Speech Recognition (ASR) system can record audio for a maximum of 1 minute. If the combined length of the prompt and the maximum wait time exceeds one minute, the system will proceed to the next building block after the one-minute mark.

9. In the 'Interdigit Timeout (sec)' field, specify the permissible waiting time between digits, from 1 to 30 seconds. The default is 2 seconds.

10. In the 'Retries' field, specify the maximum number of retries to repeat the block if DTMF input is not detected, between 1 and 10. The default is 3 retries.
11. In the 'Collected Digit Result' field, specify a variable in the format ${var_name} to store the speech input/DTMF keys result. If the call is routed through the 'No Match', the system will automatically replace the collected digit result with "No Match."



12. In the 'Collect Speech Input' toggle, choose whether to enable or disable the ASR capability for this block. When the toggle is enabled, two new fields and two additional checkboxes are added to the building block:
    - **'Speech Barge-In' checkbox**: Activating this checkbox grants your callers the ability to interrupt the prompt mid-way using either speech or DTMF input, eliminating the necessity to wait for the prompt to finish.
    - **'Transcription Result' field**: Designate a variable in the format ${var_name} to store the original speech input result before any system processing takes place. This variable allows you to access the original speech input content, even if the call is routed through 'No Match'.
    - **'Confidence Result' field**: Define a variable in the format ${var_name} to capture the confidence score identified by the Automatic Speech Recognition (ASR) system. This score can be utilized later to inform decision-making based on the obtained result.
    - **'Play Beep' checkbox**: Activating this checkbox will trigger the playing of a beep sound before the system begins to collect the user's response.
13. Click **OK**, and then **Save**.

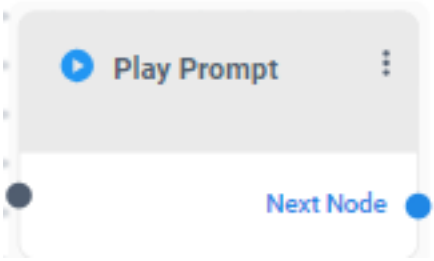
Play Prompt

'Play Prompt' allows you to integrate audio prompts into your workflows. It gives you the flexibility to play a single prompt or blend multiple prompts together, enabling the creation of dynamic and engaging interactions.

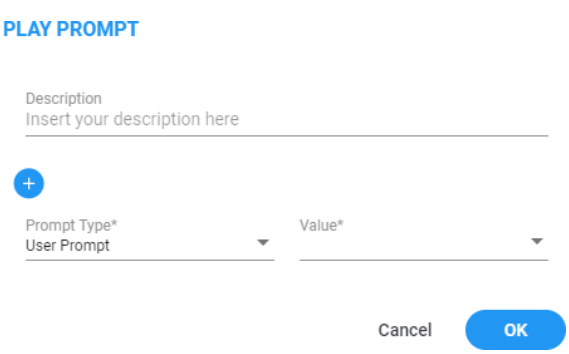
In the 'Play Prompt' building block, a single pathway, referred to as 'Next Node', is available. After the prompt’s playback, the system will automatically guide the call to 'Next Node'.

To use the Play Prompt building block:

1. On the left pane, under **Interactions**, click **Play Prompt**; the following Play Prompt building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block, (up to 50 characters).
4. From the 'Prompt Type' drop-down list, select the appropriate prompt type:
   - **User Prompt**: Select from a drop-down menu containing pre-configured prompts.
   - **Play Date**: The system will read the complete date you entered it out loud. The anticipated format for input in the value field is "yyyy-MM-dd.

     For example, if you entered the value: "2023-11-20" the system will play: 'November twenty, two thousand twenty-three. '
   - **Play Day of the Week**: The system will read out loud the day of the week corresponding to the date entered. Ensure that the date is provided in the expected format: "yyyy-MM-dd"

     For example, if you entered the following value: "2023-11-20" the system will play: 'Sunday. '
   - **Play Day in Month**: The system will read out loud the day of the month corresponding to the date entered. Ensure that the date is provided in the expected format: "yyyy-MM-dd"

     For example, if you entered the following value: "2023-11-20" the system will play: 'twenty. '

- **Play Month**: The system will read out loud the month corresponding to the date entered in the value field. Ensure that the date is provided in the expected format: "yyyy-MM-dd"

  For example, if you entered the following value: "2023-11-20" the system will play: 'November. '

- **Play Year**: The system will read out loud the year corresponding to the date entered in the value field. Please ensure that the date is provided in the expected format: "yyyy-MM-dd"

  For example, if you entered the following value: "2023-11-20" the system will play: 'two thousand twenty-three. '

- **Play Number**: The system will read out loud the number exactly as entered, treating it as a whole number.

  For example, if you entered the following value: "543" the system will play: 'Five hundred and forty-three. '

- **Digit by Digit**: The system will read out loud the number entered in the field digit by digit, rather than reading it as a whole number.

  For example, if you entered the following value: "543" the system will play: 'Five, four, three. '

- **Play Ordinal Number**: The system will read out loud the number entered as an ordinal number, such as "first," "second," "third," and so forth.

  For example, if you entered the following value: "2" the system will play: 'Second.'.

- **Play Time**: The system will read out loud the time in a 12-hour format without seconds. Make sure that the time provided is in 24hour format: "HH:mm".

  For example, if you enter the value "22:43", the system will announce it as 'Ten, forty-three PM.'

- **Play Time with Seconds**: The system will read out loud the time in a 12-hour format with seconds. Make sure that the time is provided is in a 24 hour format: "HH:mm:ss".

  For example, if you entered the value "22:43:53", the system will announce it as 'Ten, forty-three and fifty-three PM. '

- **Dynamic Prompt**: Gives you the flexibility to dynamically adjust the prompt based on previous actions taken by the caller within the flow. When selected, an additional field called "Value" becomes available. This field allows you to specify the name of your prompt as it appears in the system prompt list, using a variable.

  For example, if you have a single flow that can be triggered from different DID numbers, and you want to change the prompt based on the dialed number. To do this, configure a “Conditions” building block (see Conditions building block for more information) before running the Speech Input building block. This condition block will check the DID number and route the call to the 'Set Variables' building block accordingly (see Set Variable building block for more information). The 'Set Variables'

block will store the prompt name. This variable can then be used to play distinct prompts for different DID numbers, providing dynamic caller experiences.

5. In the 'Value' field, enter the value you want to play. This value can either be a variable collected earlier in the flow or a static string configured directly in the 'Value' field.
6. Click the  button to include additional prompts.
7. Click **OK**, and then **Save**.

Text-to-Speech

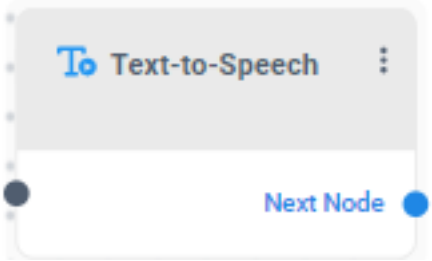
The 'Text-to-Speech' (TTS) building block enables the playback of text from a string that has been inserted under this specific building block. By default, the language used for Text-to-Speech is determined by the primary language associated with the tenant at the tenant level.

In this building block, you can combine a static string with variables that collected in the workflow. These elements can be played back as a single prompt.

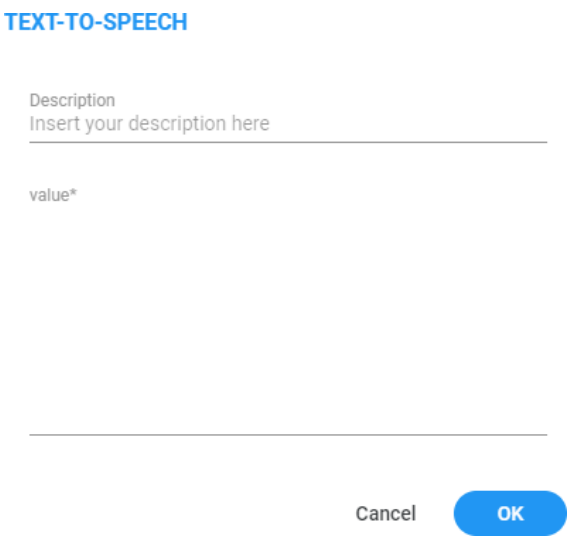
In the 'Text-to-Speech' building block, a single pathway, referred to as 'Next Node', is available. After selecting text to speech playback, the system will automatically guide the call to 'Next Node'.

➢ **To use the Text-to-Speech building block:**

1. On the left pane, under **Interactions**, click **Text-to-Speech**; the following Text-to-Speech building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block, (up to 50 characters).
4. In the 'Value' field, you can tailor the playback content according to your specific requirements. This content can be a combination of variable data collected earlier in the workflow, static strings configured directly within the 'Value' field, or a blend of both. This flexibility allows you to create dynamic prompts that adapt to your application's needs.

   For further details on creating dynamic content by combining variables and static strings, seethe section String Expression.

Text-to-Speech limits the text to 500 characters when playing.

5. Click **OK**, and then **Save**.

OpenAI

The 'OpenAI' building block seamlessly integrates generative AI capabilities into your contact center flows, enabling intelligent, conversational engagements with callers.

This block orchestrates a complete AI-driven interaction by combining speech input collection, real-time analysis via OpenAI models, and dynamic text-to-speech playback of the generated response. It captures the caller's spoken input, securely processes the request using an AI model to understand intent and generate an appropriate response, and then delivers that response back to the caller in natural-sounding speech.

By abstracting these steps into a single, easy-to-use block, the OpenAI Interaction building block simplifies flow design while empowering developers to create more responsive, contextual, and human- like conversations, ultimately enhancing customer experience and enabling more effective self-service interactions.

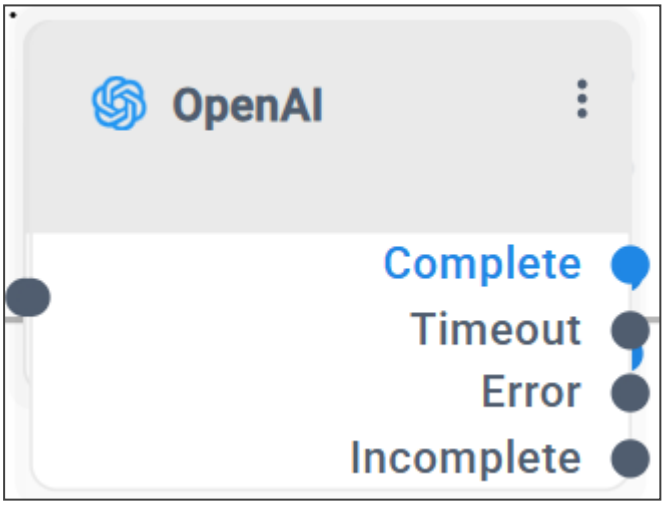
The 'OpenAI' building block has 4 exit legs:

- **Complete**: The Complete exit leg indicates a successful interaction between the customer and the 'OpenAI' block.

- When a **single interaction** is configured, the system successfully receives a response from the LLM and plays it back to the customer using TTS.
- When a **continuous interaction** is configured, the system routes to this leg once the LLM response contains the predefined completion indicator (e.g., **TASK_COMPLETE**), signaling that the conversation has successfully concluded.

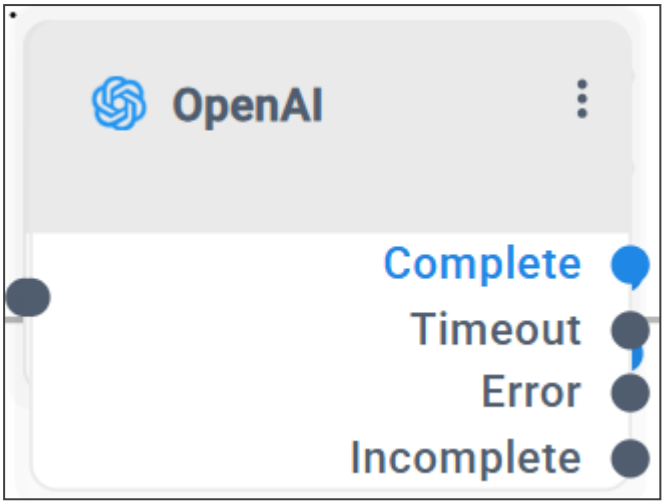
- **Timeout**: The Timeout exit leg is triggered when the block exceeds the configured LLM or Speech-To-Text (STT) timeout. In this case, the system is unable to receive a response within the defined time limits and routes the call accordingly.
- **Error**: The Error exit leg indicates that the **OpenAI** block encountered a failure during execution.

  This may occur due to configuration or processing issues, such as an invalid or malformed LLM prompt, missing or incorrect URLs, authentication problems, or other runtime errors that prevent the block from completing successfully.
- **Incomplete**: The Incomplete exit leg applies only to **continuous interactions**. It is triggered when the block reaches the configured **Max Interactions** limit before the LLM returns the TASK_COMPLETE indicator, indicating that the conversation did not conclude successfully within the allowed number of interaction cycles.

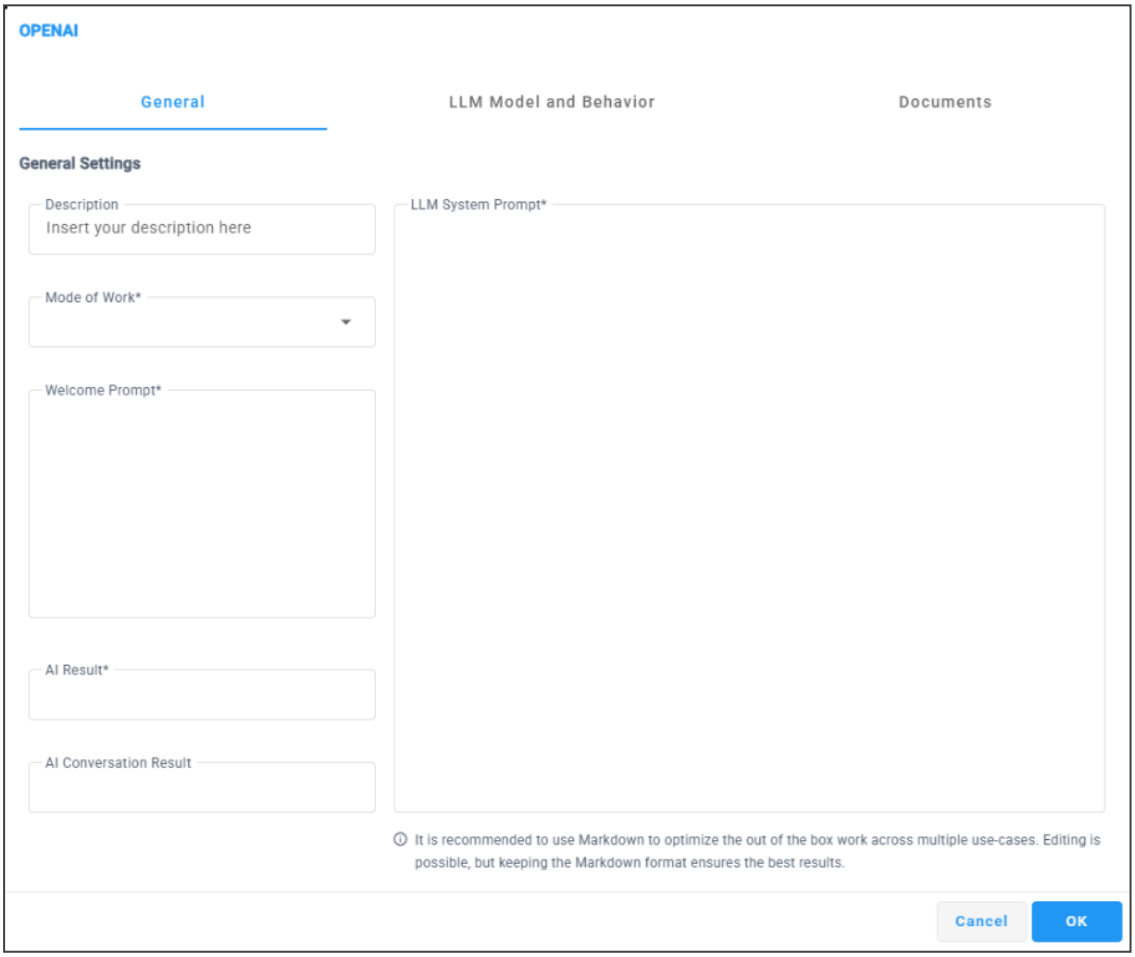


➢ **To use the HTTP building block:**

1. Click the HTTP option under the Interaction group; the following OpenAI building block appears:



2. Click the  icon; the following appears:



3. Select the General tab, and then configure the following:

   - Within the 'Description' field, provide a brief description for this building block, limited to 50 characters.
   - From the 'Mode of Work' drop-down list, select the appropriate type of request:

- Single Interaction - Performs a one-time interaction where the system collects the customer's speech input, sends it to the LLM for processing, and plays the generated response using TTS. The interaction ends immediately after the response is played.
- Continuous Interaction- Enables an ongoing, multi-turn conversation with the customer. The system repeatedly collects speech input and plays LLM responses until a completion indicator (such as TASK_COMPLETE) is returned or a configured interaction limit is reached.

- Within the 'Welcome Prompt' field define the welcome message that is played to the customer using TTS when entering the OpenAI block.

The welcome prompt may include static text, dynamic variables, or a combination of both. Static text and dynamic variables can be written directly one after the other, without using quotation marks, concatenation symbols, or any special separators.

For example:

- Hi Welcome to AudioCodes
- Hi ${companyName}
- Hi ${company Name} How can I help you

The system automatically resolves dynamic variables at runtime and converts the complete text into speech before the interaction begins.

- Within the 'AI Result' field, you can specify a variable in the format ${var_name} to store the response received from the AI. The AI response structure should be defined in the LLM system prompt to ensure the data is returned in the expected format.

It is recommended to use a **JSON-formatted response**, as this allows easy extraction of specific values using the GetJsonValue function in subsequent flow steps.

- Within the 'AI Conversation Result' field, you can specify a variable in the format ${var_name} to store the full conversation transcript between the customer and the LLM. The stored result is returned as a plain text conversation log, presenting each interaction turn sequentially, clearly indicating what the user said and how the LLM responded at each step.

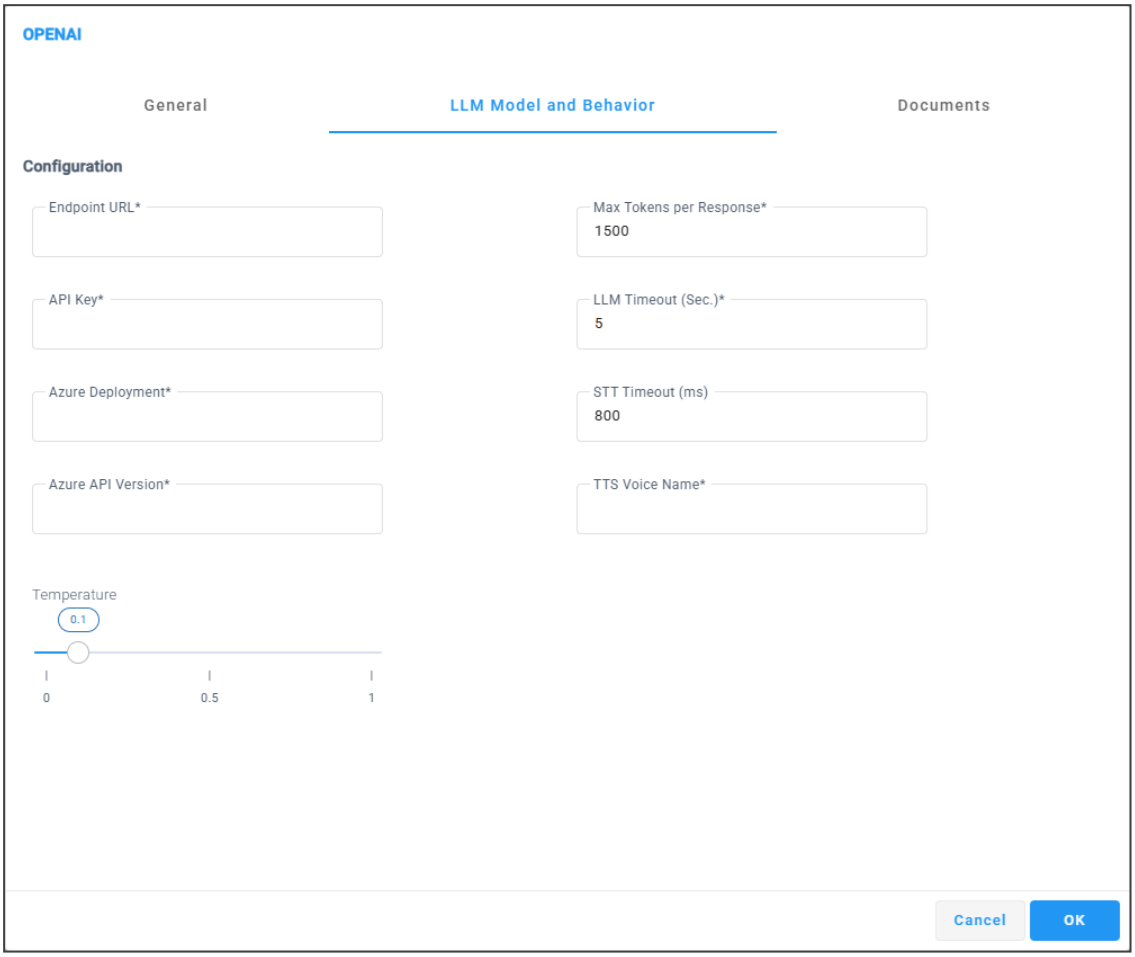
This field is intended for logging, auditing, or downstream processing of the complete conversational context rather than structured data extraction

- The "LLM System Prompt" field allows you to define the system-level instructions that guide how the LLM should interpret user input and generate responses throughout the interaction. The system prompt is used to set context, define rules, control response style, and specify the expected output structure returned by the AI.

The prompt supports a combination of static text and dynamic variables in the format ${var_name}. As with the Welcome Prompt field, static text and variables can be written directly together, without quotation marks, concatenation characters, or special symbols. All variables are resolved at runtime before being sent to the LLM.

It is strongly recommended to write the system prompt using Markdown (MD) formatting, as structured formatting (such as headings, lists, and emphasis) helps the LLM better understand instructions, enforce rules, and return clearer and more consistent responses especially when defining structured outputs or multi-step behaviors.

4. Navigate to the LLM Model and Behavior tab to configure the following settings:



- Within the 'Endpoint URL' field, ꞇpecifies the base endpoint URL of the Azure OpenAI service that the OpenAI block will connect to.

  This URL defines the Azure region and OpenAI resource used to process all LLM requests from the flow.

  The endpoint must be provided in the following format:

  https://<resource-name>.openai.azure.com

  Ensure that the endpoint corresponds to a valid and accessible Azure OpenAI resource.

- In the 'Api Key' filed Specify the authentication key used to authorize requests to the Azure OpenAI service. The API key is required to securely connect the OpenAI block to the configured endpoint and must match the Azure OpenAI resource defined in the Endpoint URL.

- In the 'Azure Deployment' Specifies the name of the Azure OpenAI model deployment that will be used to process requests from the OpenAI block. This value must match the

deployment name configured in your Azure OpenAI resource, not the model name itself.

The selected deployment determines which LLM is used for analyzing user input and generating responses during the interaction. Ensure that the deployment exists, is active, and is supported by the configured endpoint and API key.

- In the 'Azure API Version' Specifies the Azure OpenAI API version that will be used to process requests from the OpenAI block. This value must match a supported API version exposed by the configured Azure OpenAI endpoint.

  The selected API version defines the request and response schema used during the interaction. Ensure that the specified version is supported by the Azure OpenAI resource and is compatible with the configured endpoint, deployment, and API key.

- In the 'Temperature' slider Controls the randomness and creativity of the LLM responses. Lower values make the AI responses more deterministic, focused, and consistent, while higher values allow more variation, creativity, and flexibility in the generated output.

  The default temperature value is **0.1**, providing stable and predictable responses that are recommended for most contact center and automation use cases. Increasing the value may be useful for more open-ended or conversational scenarios where flexibility is desired.

- In the 'Mat Token Per Response' filed Specifies the maximum number of tokens the LLM is allowed to generate in a single response. This parameter limits the length of the AI-generated output, including the actual response text.

Setting a lower value helps control response size, reduce latency, and prevent overly long answers. Higher values allow more detailed and comprehensive responses but may increase processing time and token consumption. Ensure the configured value aligns with the expected response complexity and the use case of the flow.

- In the 'LLM Timeout (Sec.)' filed Specifies the maximum amount of time, in seconds, that the system will wait for a response from the LLM before terminating the request. If the LLM does not return a response within this time limit, the OpenAI block will exit through the 'Timeout' leg.

  This parameter helps control responsiveness and prevent long waits during the interaction. Configuring an appropriate timeout value ensures a balance between allowing sufficient processing time for complex requests and maintaining a smooth customer experience

- In the 'STT Timeout (ms)' filed Specifies the maximum amount of time, in milliseconds, that the system will wait for the Speech to Text (STT) service to return a transcription of the customer's speech. If no transcription is received within this time limit, the OpenAI block will exit through the 'Timeout' leg.

  This parameter helps control responsiveness during speech collection and ensures that the flow does not wait indefinitely for user input. Configure the value according to expected speech length and network conditions.

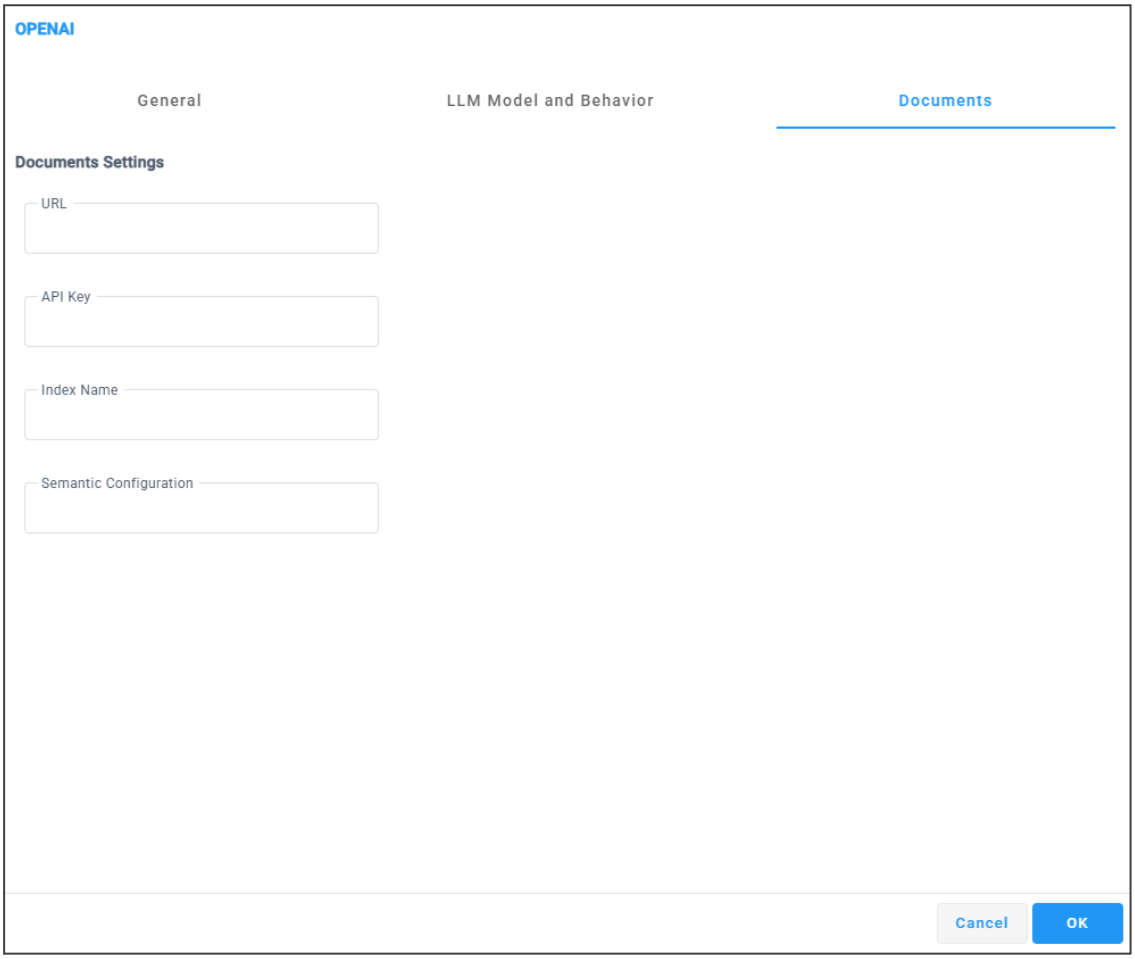
- In the ‘TTS Voice Name’ Specifies the Azure Text To Speech voice that will be used to play AI-generated responses and prompts to the customer. The value must be provided in the **Azure voice identifier format**, as defined by the Azure Speech service.

  The selected voice determines the language, accent, and speaking style of the synthesized speech. Ensure that the specified voice is supported by the Azure region configured for the service.

  For example:

  - en-US-JennyNeural

5. Navigate to the Document tab to configure the following settings:



The Documents section allows you to connect the OpenAI block to an external document index for knowledge-based responses. When configured, the LLM can retrieve and use relevant information from indexed documents to answer customer questions, enabling grounded, context-aware interactions based on your organization’s content.

This capability is typically used for FAQs, knowledge bases, manuals, or internal documentation.

- Within the 'URL' field, Specifies the endpoint URL of the document search service (for example, Azure Cognitive Search). This endpoint is used by the OpenAI block to query indexed documents during the interaction.

- Within the 'API Key' field, Specifies the API key used to authenticate access to the configured document search service. Ensure that the key is valid and has permission to query the specified index.
- Within the “Index Name” Specifies the name of the document index that contains the searchable content. This value must match an existing index in the configured document search service.
- Within the “Semantic Configuration Specifies the semantic configuration name used by the document index to improve search relevance and ranking. This configuration controls how document content is interpreted and matched against user queries.

Actions

The following building blocks appear under Actions.

- HTTP
- Go To Menu
- Transfer
- Save Call Info
- Go To Queue
- Go To Destination
- Go To Contact
- Leave Message
- Send SMS
- Go To Flow
- Select Language

HTTP

The 'HTTP' building block offers a seamless avenue for integration, process automation, and tailored customization, empowering you to engage with an array of systems and platforms. This versatile building block extends the capability to integrate with third-party applications, such as CRM systems, enabling real-time data retrieval and the execution of actions within your third-party applications. It provides a robust framework to enhance your system's connectivity and functionality.

The 'HTTP' building block has 3 exit legs:

- **Success:** Indicates a successful transmission of a request from Voca and the subsequent receipt of a response from the destination. Note that the success exit leg does not necessarily indicate that there are no issues with the response, it just confirms that Voca successfully received a response.

After a call resulted in success, you should utilize the Condition building block (see Condition building block for more information) to ascertain whether the status code received aligns with the expected status code. This step enables you to validate the outcome of the request and make informed decisions based on the response status code.

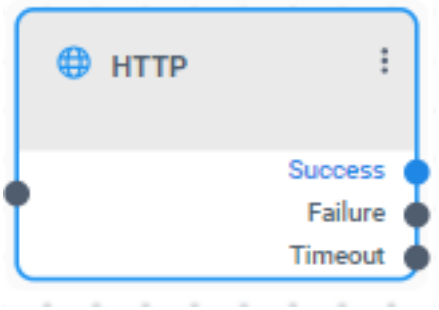
- **Failure:** Indicates an unsuccessful request sent from Voca. When a call is processed through this leg, it indicates that there are issues with the request itself, such as an invalid URL, an improperly formatted body, or invalid headers, etc...

  This serves as an indicator that the request could not be successfully initiated due to a specific issue on the request side.

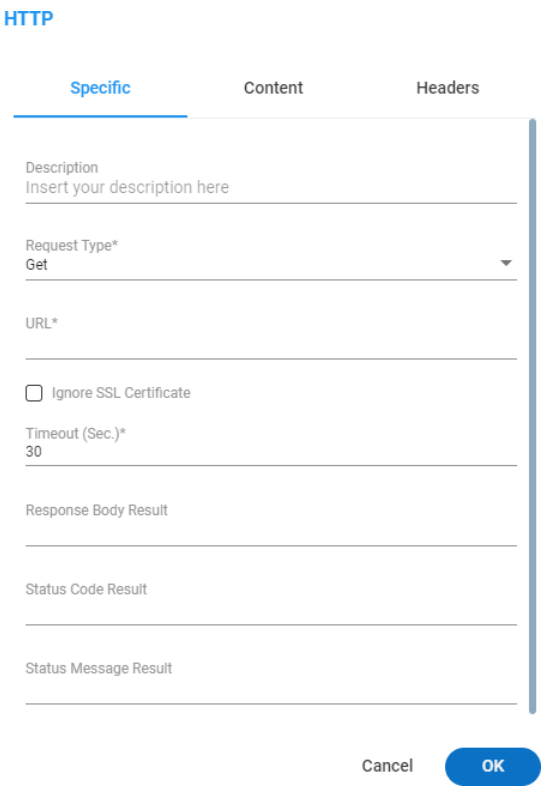
- **Timeout:** Indicates Voca successfully transmitted a request; however, it did not receive any response from the destination. This signifies a situation where the request was sent, but there was no corresponding response received within the expected time frame.

➢ To use the HTTP building block:

1. On the left pane, under **Actions**, click **HTTP**; the following HTTP building block appears:



2. Click the  icon; the following appears:



3. Select the **Specific** tab, and then configure the following:

   a. In the 'Description' field, enter a description for this building block (up to 50 characters).

   b. From the 'Request Type' drop-down list, select the appropriate type of request:

      - **Get**
      - **Post**

   c. In the 'URL' field, enter the destination URL. Note, this URL can take the form of a static string, a variable, or an expression that dynamically computes the appropriate REST address during runtime. When opting for a static string, there is no requirement to enclose it within quotation marks at both the start and end of the value.

   d. To disregard the SSL certificate, select the 'Ignore SSL Certificate' check box. This is relevant for self-signed certificates. However, exercise caution when utilizing this option.

   e. In the 'Timeout' field, specify the maximum timeout duration in seconds for the request, ranging from 1 to 60 seconds. Once this predetermined time limit is reached, the call will be routed to the 'Timeout'. The default is 20 seconds.

   f. In the 'Response Body Result' field, specify a variable in the format ${var_name} to store the received response. The response will be automatically converted into a string and assigned to the specified variable.

   g. In the 'Status Code Result' field, specify a variable in the format ${var_name} capture the returned status code.

h. In the 'Status Message Result' field, specify a variable in the format ${var_name} to capture the returned status message.

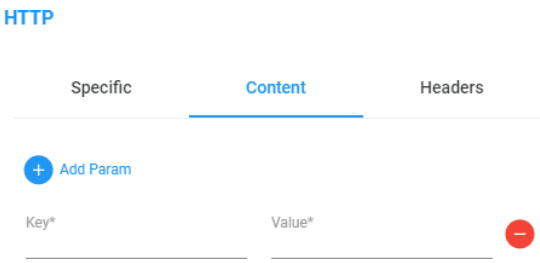
4. Navigate to the **Content** tab and configure the following settings. This screen will only appear if you are using a **GET request**:

a. Click the **Add Param** icon to add a parameter.

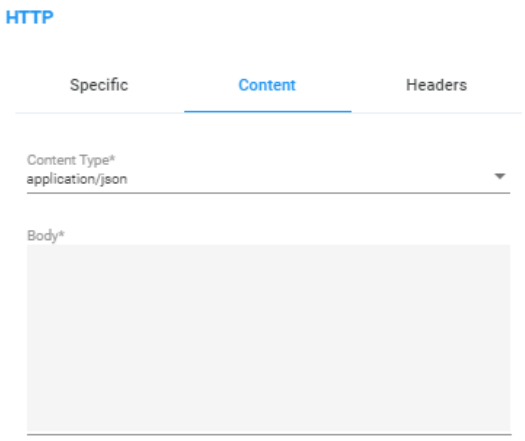
b. In the 'Key' field, enter the header key, excluding quotation marks.

c. In the 'Value', enter the values as a static string, a variable, or an expression that dynamically computes the appropriate header value during runtime.

d. Click the **Add Param** icon to add more parameters.



5. Navigate to the **Content** tab and configure the following settings. This screen will only appear if you are using a **POST request**:



a. From the 'Content Type' drop-down list, select the appropriate content type:

- Application/json
- Text/plain

b. In the 'Body' field, you have the flexibility to define the content you want to include in the HTTP requests. There are two primary options available for formatting the body content:

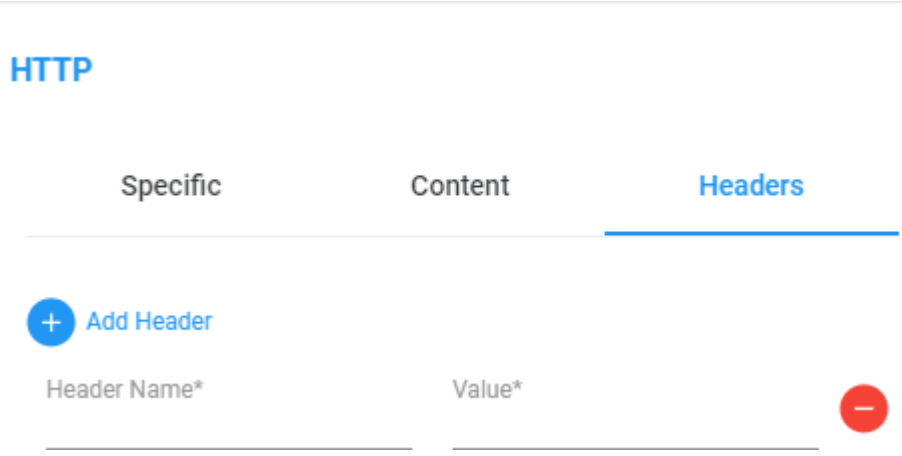
- **JSON Payload**: you can input a static JSON payload for transmission. This involves defining a fixed JSON structure with specific key-value pairs. The provided JSON remains constant in the request.

- **Expression (String Format)**: Alternatively, you can enter an expression in string format. This option allows you the insertion of dynamic content from the workflow into the HTTP request body.

> When incorporating variables into your JSON body, the JSON content should be provided as a string rather than as a structured JSON payload.

For details how to use the HTTP request body using these options, see Creating a body for HTTP Request.

6. Select the **Headers** tab, and then configure the following:
   a. Click the **Add Header** icon to add a header.
   b. In the 'Header Name' field, enter the header key, excluding quotation marks.
   c. In the 'Value' field, enter the values as a static string, a variable, or an expression that dynamically computes the appropriate header value during runtime.
   d. Click the **Add Header** icon to more headers.



7. Click **OK**, and then **Save**.

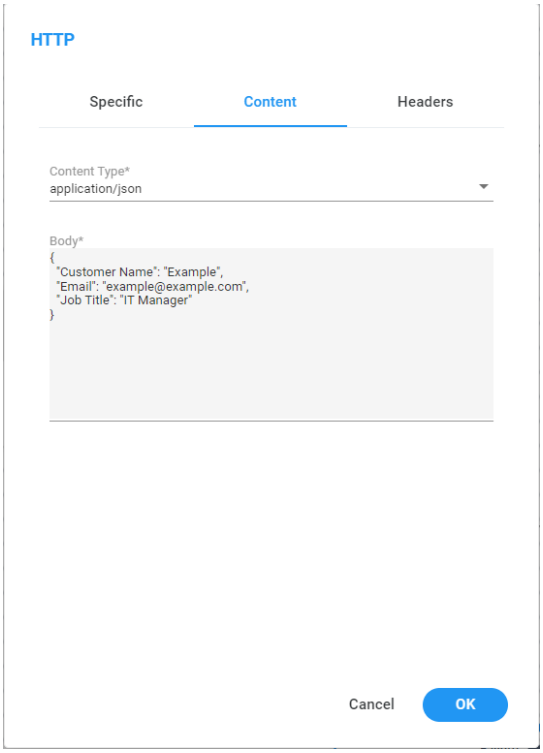
➢ **To create a body for HTTP Request:**

- **JSON Payload**: If you want to send a static JSON payload in the body of your HTTP request, follow this format:

```
{
    "Key1": "Value1",
      "Key2": "Value2",
    ...
}
```

In this scenario, define the JSON structure with specific key-value pairs, and these values remain constant in the request.

For example:



- **Expression (String Format)**: When you need to insert dynamic variables from your workflow into the HTTP request body, follow these steps:

  a. Convert the JSON structure you expect to receive into a string format. Enclose the entire JSON within quotation marks, like this:

```
"{
      \"Key1\": \"Value1\",
\"Key2\": \"Value2\",
        ...
}"
```

  You can use any online JSON-to-string converter for this purpose.

  Converting the JSON to a string format allows for additional manipulation and variable insertion.

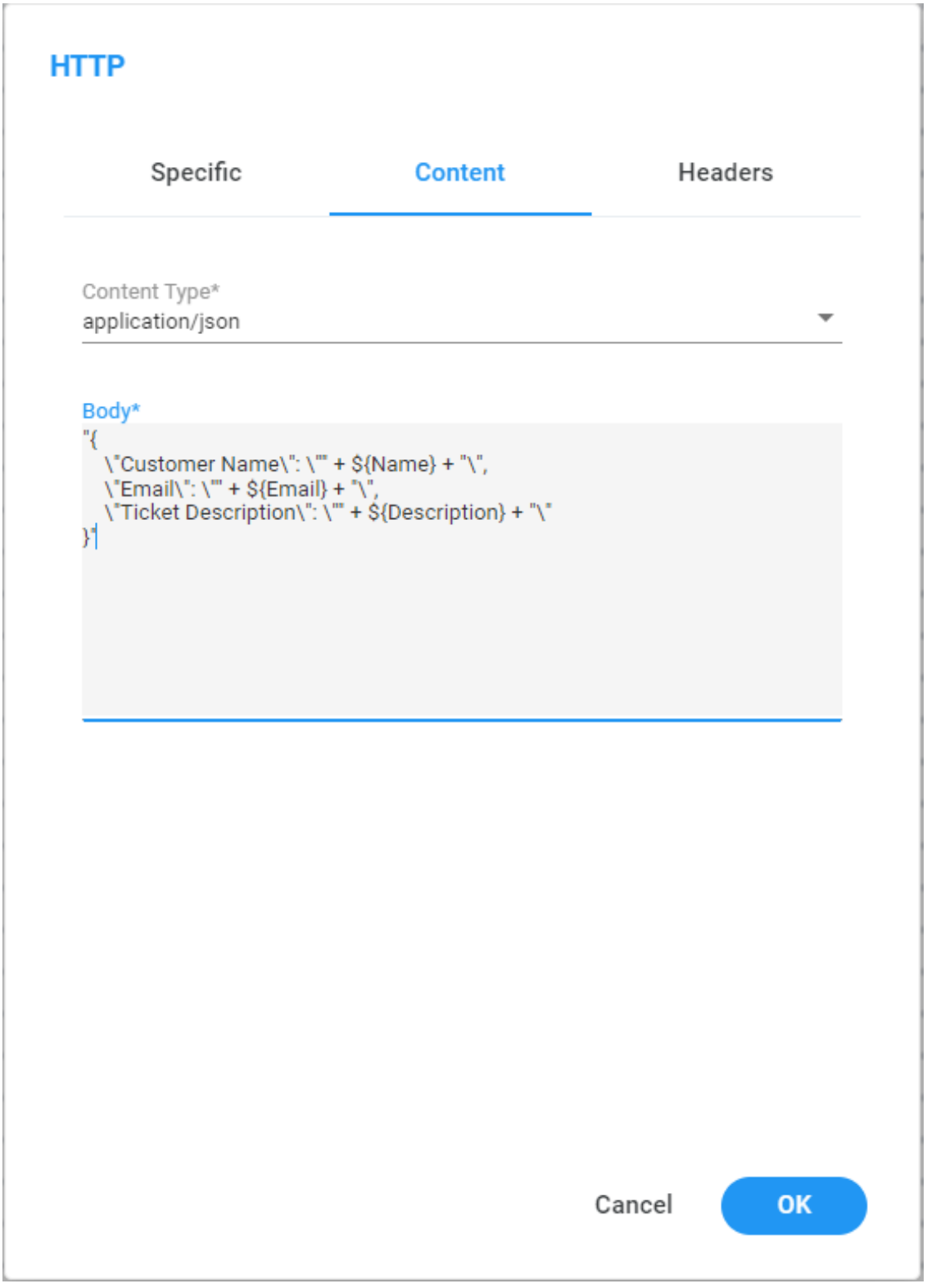
  b. Replace the value with the variable that you want to use, using the following format, including the quotation marks:

Original value	Formatted value
Value1	" + ${var_name1} + "
Value2	" + ${var_name2} + "

Replace ${ var_name1} with the actual variable name you want to include. This syntax ensures that the variable's value is dynamically inserted at runtime when the HTTP request is made.

```
"{
        \"Key1\": \"" + ${var_name1} + "\",
\"Key2\": \"" + ${var_name2} + "\",
          ...
}"
```

Example:

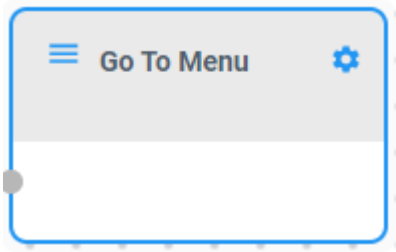


Go To Menu

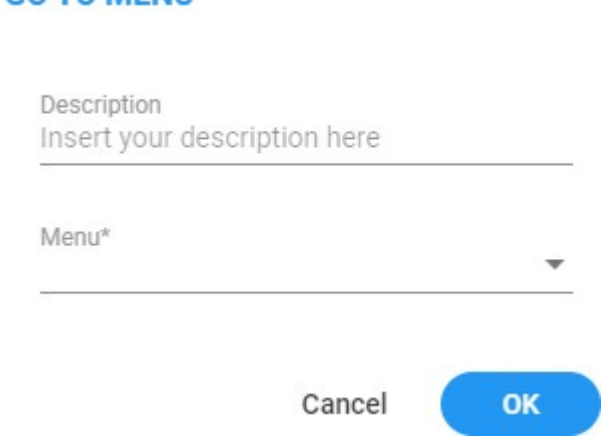
The 'Go To' Menu' building block provides the capability to seamlessly redirect a call directly from the call flow to a menu that is configured under 'Configuration -> Menu Setting' tab.

➢ **To use the Go To Menu building block:**

1. On the left pane, under **Actions**, click **Go To Menu**; the following Go To Menu building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. From the 'Menu' drop-down list, select the appropriate menu.
5. Click **OK**, and then **Save**.

Transfer

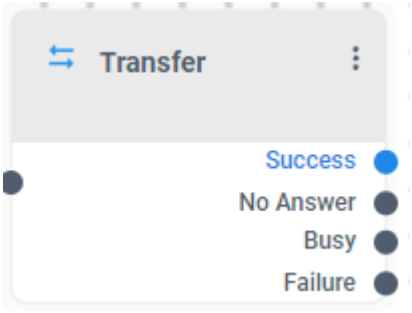
The 'Transfer' building block allows you to execute either a blind transfer or an attended transfer directly from the flow designer. Subsequently, you can initiate specific actions based on the response received following the transfer operation.

The ‘Transfer’ block has different exit points depending on the chosen "Transfer Type.

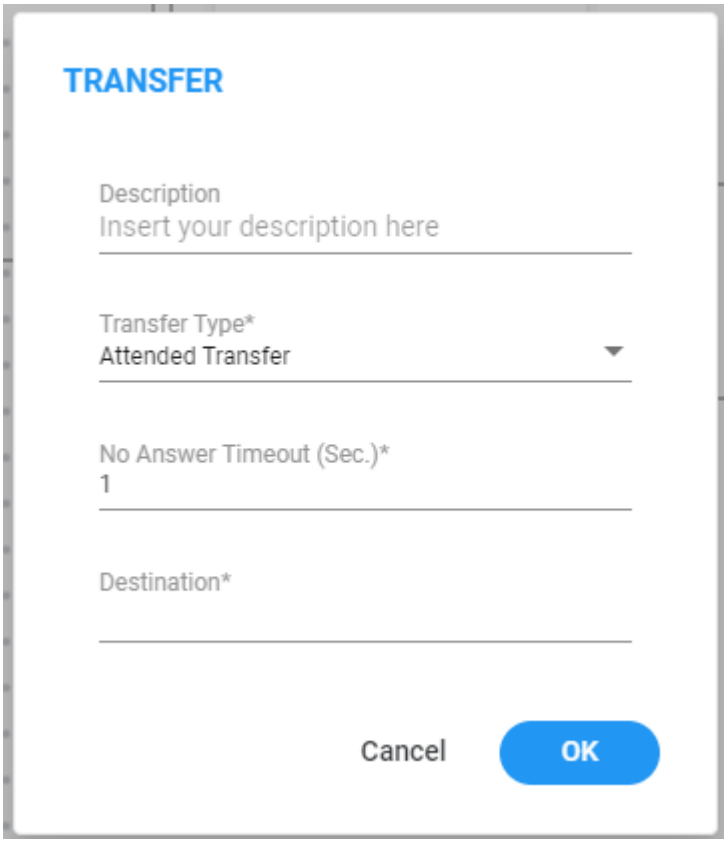
- **Success**: The transfer of the call to the designated destination was completed successfully.
- **No Answer**: The call directed to the assigned destination remained unanswered beyond the duration configured for the "No Answer Timeout" timer. This only appears when you use "Transfer Type" mode during an attended transfer.
- **Busy**: The designated destination was busy, and the call could not be transitioned. This exit leg only appears when you use "Transfer Type" mode.
- **Failure**: The attempt to transfer the call to the specified destination has failed.

➢ To use the Transfer building block:

1. On the left pane, under **Actions**, click **Transfer**; the following Transfer building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block, (up to 50 characters).
4. From the 'Transfer Type' drop-down list, select the appropriate type of transfer:
   - Blind Transfer
   - Attended
5. In the 'No Answer Timeout (Sec)' field, enter maximum time to wait for an answer (1 to 120 seconds). When the predetermined time limit is reached, the call is automatically redirected to 'No Answer'. This field only appears if ‘Attend Transfer’ is selected.

6. In the 'Destination' field, enter the destination number for the transfer operation. This destination can either be a variable collected earlier in the flow or a static string directly configured in the 'Destination' field.
7. Click **OK**, and then **Save**.

Display Data

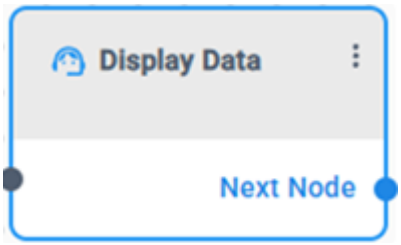
The 'Display Data' building block enables you to collect data from the call flow and seamlessly provide this information to your workers when they handle calls through the Flex and Worker queue. In addition, you can configure a CRM screen pop-up within the 'Display Data' block. This feature allows Voca workers to access and update CRM data in real-time within the Voca Flex App or Worker App, all within a unified interface.

To enable the CRM screen pop-up feature, your CRM system must support iframes. The CRM screen pop-up functionality relies on the ability to embed content within an iframe.

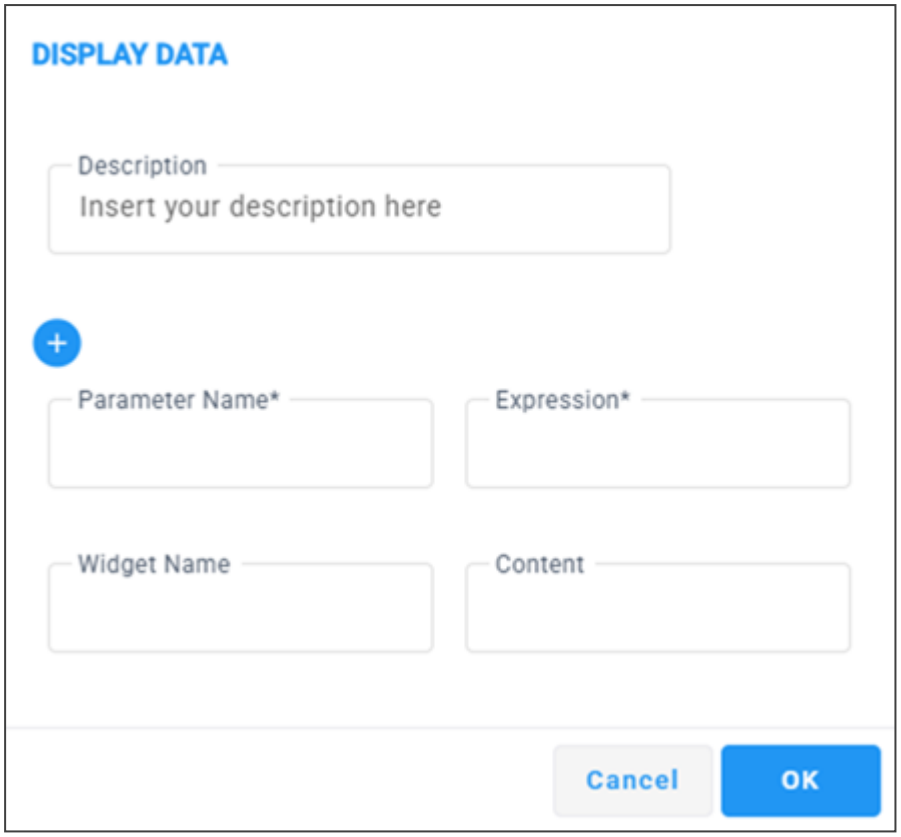
In the 'Display Data' building block, you have access to a single pathway called 'Next Node'. When the system has completed the process of saving all collected data, it will direct the call to 'Next Node'.

➢ **To use the Display Data Info building block:**

1. On the left pane, under **Actions**, click **Display Data**; the following Worker Application building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. In the 'Parameter Name' field, enter the name of the field that will be displayed to the worker after they answer the call.
5. In the 'Expression' field, enter the value corresponding to the field configured in the 'Parameter Name.' This value can either be a previously collected variable within the flow or a static string.
6. In the 'Widget Name' field, enter the name of your CRM system. This name will be displayed at the top of the CRM screen for the worker after they answer the call.
7. In the 'Content' field, enter the URL of your CRM system. Note, this URL can be either a static string, a variable, or an expression.

   Static strings don't need quotation marks.

- The entered URL must support display within an iframe.
- If the URL contains ".salesforce.com", the Worker app will automatically open the popup in a new browser tab.

8. Click the + button to include additional parameters.
9. Click **OK**, and then **Save**.

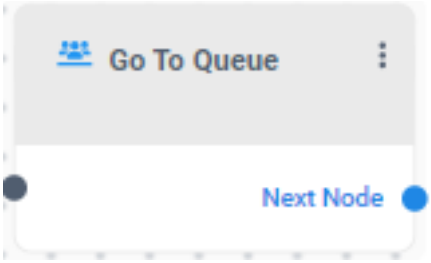
Go To Queue

The 'Go To Queue' building block allows you to redirect a call directly from the call flow to a queues that configured under the 'Configuration -> Routing -> Queues' tab.

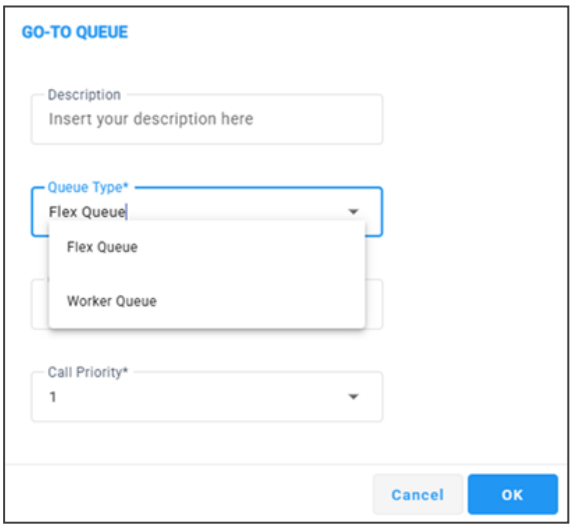
In the 'Go To Queue' building block, you are provided access to a singular route called 'Next Node'. Following the system's call transfer to the designated queue, you can configure supplementary actions that do not require user interaction, such as initiating an HTTP Request or sending an SMS message.

➢ To use the Go To Queue building block:

1. On the left pane, under **Actions**, click **Go To Queue**; the following Go To Queue building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. From the 'Queue Type' drop-down list, select the appropriate type of queue:
   - Flex Queue
   - Worker Queue
5. From the 'Queue Name' drop-down list, select the appropriate queue name.

6. From the 'Skills' drop-down list, select the appropriate skill when transferring a call to a queue. When the call reaches the queue, the system will automatically search for an available worker with all the specified skills from the building block. Note this option is only displayed when the 'Worker Queue' is selected, ensuring that calls are directed to the most qualified workers.
7. From the 'Call Priority' drop-down menu, administrators can designate the call priority for incoming calls. This allows administrators to implement a tiered priority system, guaranteeing the swift handling of critical or high-value calls. Prioritization operates on a numerical scale, where a higher priority value corresponds to increased precedence in the call queue. The default priority level is '1,' but administrators can adjust this priority level up to '10'.
8. Click **OK**, and then **Save**.

Go To Destination

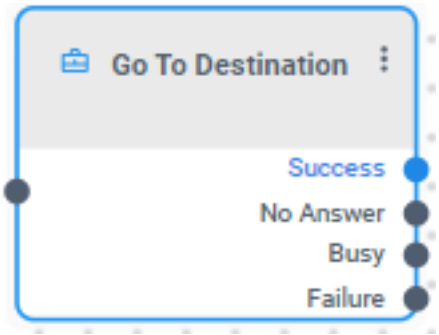
The 'Go To Destination' building block provides the capability to redirect a call directly from the call flow to a Departments/Contact on the system.

The Go To Destination" block's exit leg depends on ‘Transfer Type’ mode:

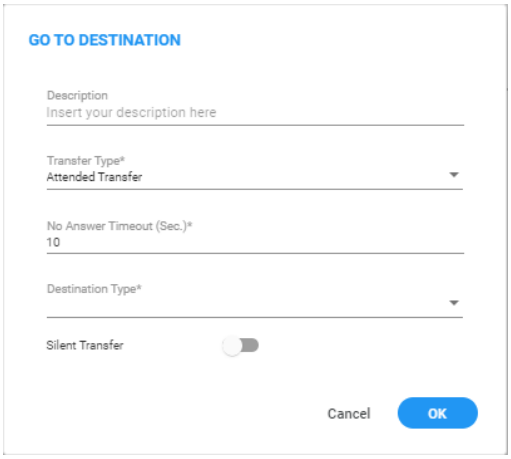
- **Success**: Indicates the transfer of the call to the designated destination was completed successfully.
- **No Answer**: Indicates the call directed to the assigned destination remained unanswered beyond the duration configured for the "No Answer Timeout" timer. This exit leg only appears when you use ‘Transfer Type’ mode during an attended transfer.
- **Busy**: Indicates the designated destination was busy, and the call could not be transitioned. This exit leg only appears when you use "Transfer Type" mode during an attended transfer.
- **Failure**: Indicates the attempt to transfer the call to the specified destination has failed.

➢ To use the Go To Destination building block:

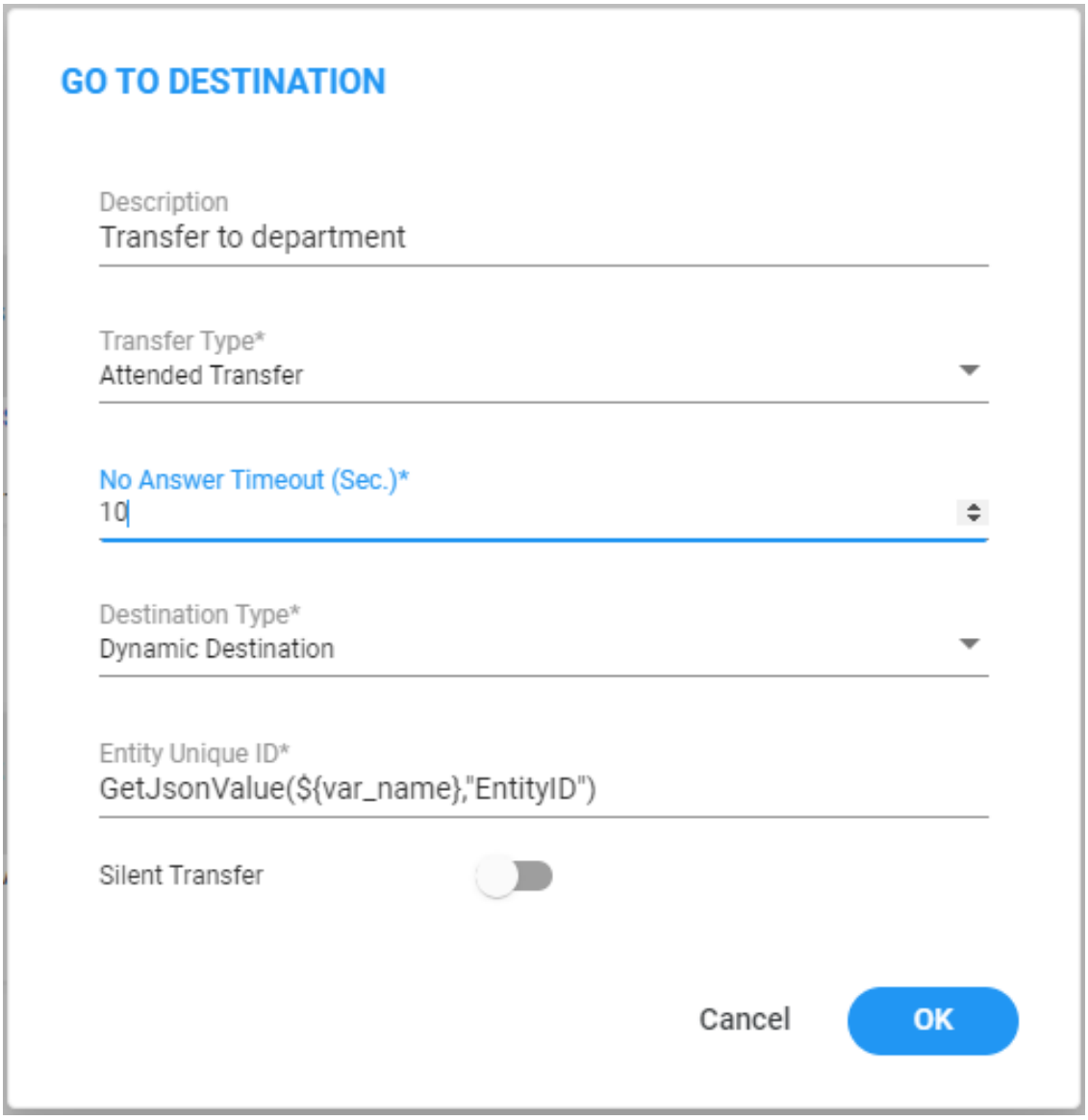
1. On the left pane, under **Actions**, click **Go To Destination**; the following Go To Destination building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. From the 'Transfer Type' drop-down list, select the appropriate type of transfer:
   - **Blind Transfer**: When the system has successfully routed the call to the designated department, it runs configured actions within the department (unless the transfer fails).
   - **Attended Transfer:** Select this option when the department's configured action falls into one of the following categories: 'Transfer to Phone,' 'Silent Transfer to Phone,' or 'Transfer to Extension.' As an administrator, you seek the capability to perform supplementary actions within the same call flow after the transfer process has concluded.
5. In the 'No Answer Timeout (Sec)' field, enter maximum time to wait for an answer (1 to 120 seconds). When the predetermined time limit is reached, the call is automatically redirected to 'No Answer'. This field only appears if ‘Attend Transfer’ is selected.
6. From the 'Destination Type' drop-down list, select the appropriate destination type:
   - **Destination List:** You can select a specific department from a predefined dictionary. When selected, two additional fields become available:
     - 'Dictionary'.
     - 'Department'
   - **Dynamic Destination**: Allows you to dynamically adjust the department destination based on caller interactions leading up to this stage. Enter a variable that holds the department ID from your tenant. When selected, an additional field becomes available:



7. In the 'Entity Unique ID' field, enter the department's unique identifier. This value can be derived from a variable gathered earlier in the workflow, such as through speech input, or it can be a fixed string configured directly in the 'Value' field.

   For example:

   If you've previously utilized the Speech Input building block in destination dictionary mode in your workflow and stored the result as a variable named ${var_name}, you can utilize the GetJsonValue function (for more details, see GetJsonValue function) to route the call to the relevant department.

8. Use the 'Silent Transfer' toggle, to enable or disable the default prompt, "Transferring the call to...", before initiating the call transfer to the destination.

9. Click **OK**, and then **Save**.

Go To Contact

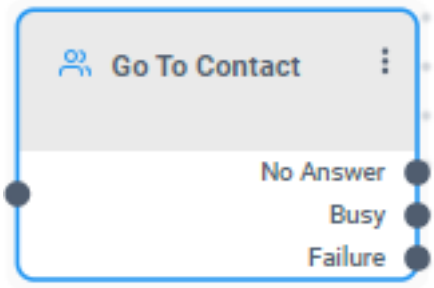
The 'Go To Contact' building block allows you to redirect a call directly from the call flow to a contacts that configured under the 'Contacts' tab.

The 'Go To Contact' building block have 3 exit legs:

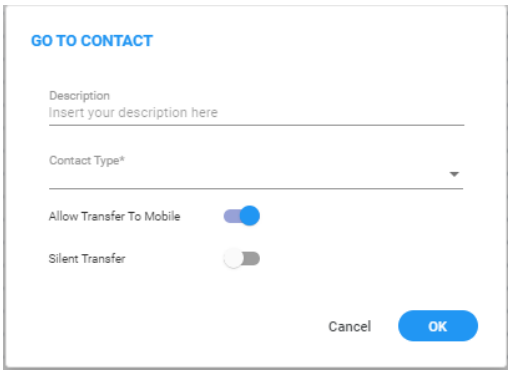
- **No Answer**: Indicates the call directed to the assigned destination remained unanswered beyond the duration configured for the "No Answer Timeout" timer. This exit leg only appears when you use "Transfer Type" mode during an attended transfer.
- **Busy**: Indicates the designated destination was busy, and the call could not be transitioned.
- **Failure**: Indicates the attempt to transfer the call to the specified destination has failed.

➢ To use the Go To Contact building block:

1. On the left pane, under **Actions**, click **Go To Contact**; the following Go To Contact building block appears:

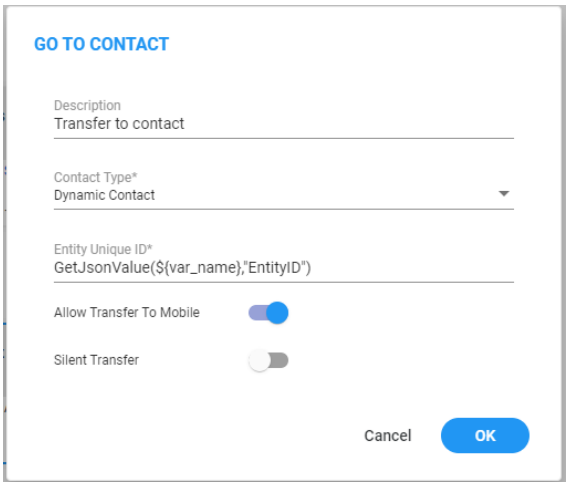


2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. From the Contact Type' drop-down list, select the appropriate contact type:
   - **Contact List**: The system allows you to select a specific contact from the contact list. When selected, an additional field becomes available.
   - **Dynamic Contact**: This Contact type provides the versatility to dynamically adjust the contact destination based on caller interactions leading up to this stage. In the value

field, you can input a variable that holds the contact ID from your tenant. When selected, an additional field becomes available.



5. In the 'Entity Unique ID' field, enter the contact's unique identifier. This value can be derived from a variable gathered earlier in the workflow, such as through speech input, or it can be a fixed string configured directly in the 'Value' field.

   For example:

   If you've previously utilized the Speech Input building block in destination dictionary mode within your workflow and stored the result as a variable named ${var_name}, you can utilize the GetJsonValue function (for more details, see GetJsonValue function) to route the call to the relevant department.

6. Use the 'Allow Transfer To Mobile' toggle, to activate or deactivate the capability to route calls to the mobile phones of your contacts.

7. Use the 'Silent Transfer' toggle, to enable or disable the default prompt, "Transferring the call to...s", before initiating the call transfer to the contact.

8. Click **OK**, and then **Save**.

Leave Message

The 'Leave Message' building block gives you the ability to offer your customers the option to leave a message that will be transmitted via email. The system permits you to play a prompt, followed by recording the customer's message and sending it as an email attachment.

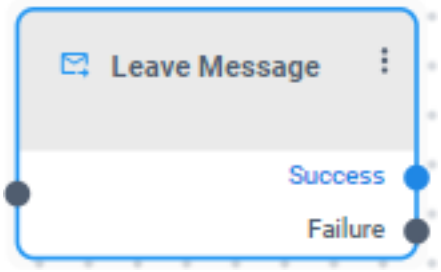
The 'Leave Message' building block requires an external email service—such as an SMTP server or Azure mail server—to be configured under Email Settings in Voca

The 'Leave Message' building block has two exit legs:

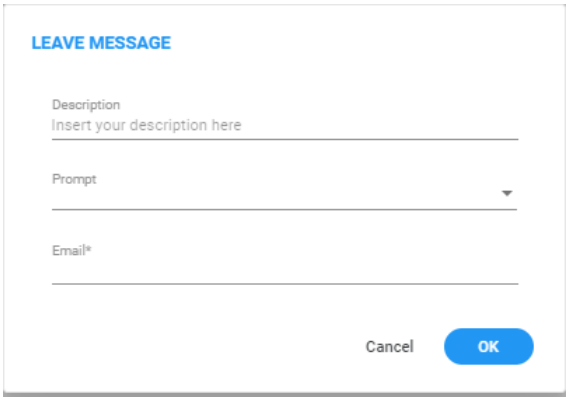
- **Success**: Indicates the email server has successfully received the request to send the email to the recipient.
- **Failure**: Indicates the email server did not successfully receive the request to send the email to the recipient.

➢ **To use the Leave Message building block:**

1. On the left pane, under **Actions**, click **Leave Message**; the following Leave Message building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. From the 'Prompt' drop-down list, select the appropriate prompt.
5. In the 'Email' field, enter the email address enclosed in quotation marks. This designated email address will be the recipient of the message along with the recording attachment.

   The 'Email' field can be a static string, a variable, or an expression that dynamically computes the appropriate recipient address during runtime.
6. Click **OK**, and then **Save**.

Send SMS

The 'Send SMS' building block grants you the capability to send SMS to your customers during the call flow.

The “Send SMS” building block requires Voca to be integrated with a telecom provider to enable outbound SMS functionality.

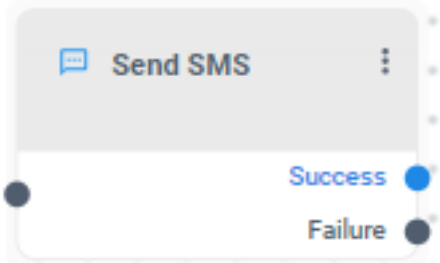
The 'Send SMS' building block have two exit legs:

- **Success**: Indicates the SMS server has successfully received the request to send the SMS message to the recipient.

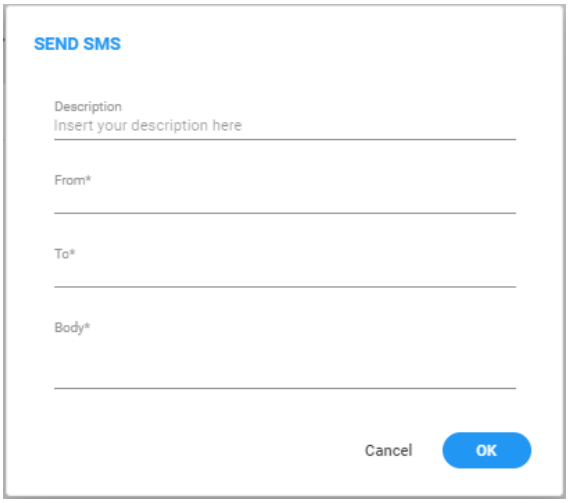
- **Failure**: Indicates the SMS server did not successfully receive the request to send the SMS message to the recipient.

➢ To use the Send SMS building block:

1. On the left pane, under **Actions**, click **Send SMS**; the following Send SMS building block appears:



2. Click the  icon; the following appears:



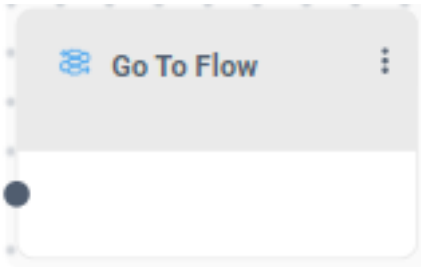
3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. In the 'From' field, enter the sender address from which the SMS will be sent. Make sure that the sender address contains 1 to 11 alphanumeric characters or hyphens (-), without quotes.
5. Enter the recipient's address in E164 format.
6. The 'To' field can be a static string, a variable, or an expression that dynamically computes the appropriate recipient address during runtime.
7. In the 'Body' field, enter the content of the message that will be sent. The message content must be surrounded by quotation marks.
8. Click **OK**, and then **Save**.

Go To Flow

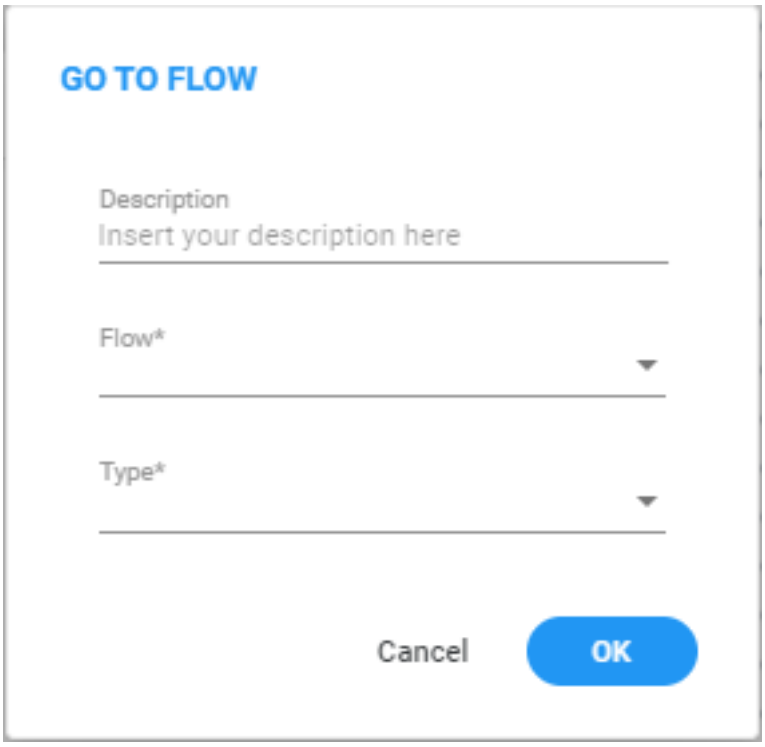
The 'Go To Flow' building block provides administrators with the ability to establish distinct boundaries between various call flows within the system while retaining all previously collected information within the call flow. This functionality ensures that essential data and context gathered earlier in the call process are seamlessly preserved as callers transition between different stages of the call flow, promoting a coherent and efficient call handling experience.

➢ **To use the Go To Flow building block:**

1. On the left pane, under **Actions**, click **Go To Flow**; the following Go To Flow building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. From the 'Flow' drop-down list, select the appropriate flow.
5. From the 'Type' drop-down list, select the appropriate Type.

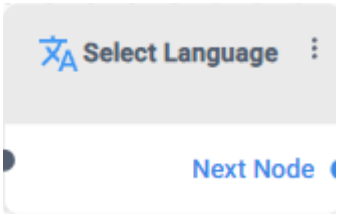
6. Click **OK**, and then **Save**.

Select Language

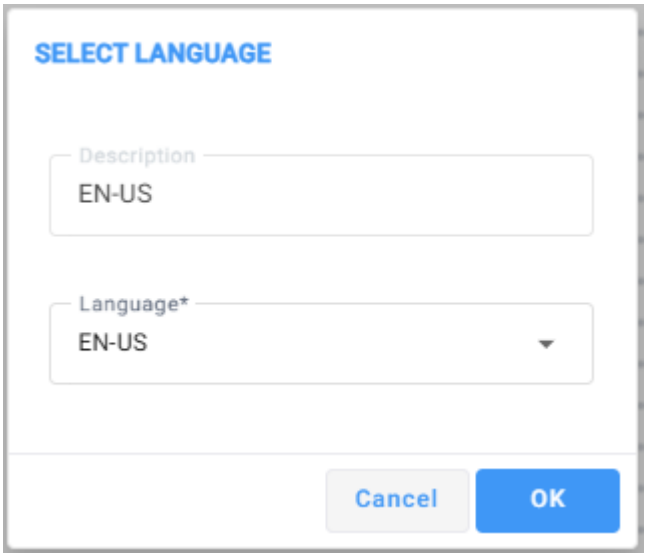
The 'Select Language' building block allows you to change the language of the flow. This affects Text-to-Speech (TTS), all prompts (including system prompts), and the speech recognition process, ensuring that all spoken messages play in the selected language. This enables the use of the same flow for multiple languages. The system will automatically direct the call to the 'Next Node,' and from there, it will operate in the selected language.

➢ **To use the Select Language building block:**

1. On the left pane, under **Actions**, click **Select Language**; the following Select Language building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. From the 'Language' drop-down list, select the appropriate language.
5. Click **OK**.

Call-Flow Logic

The following building blocks appear under Call-Flow Logic:

- Conditions building

- Switch building
- Counter building
- Set Variable building
- End building

Conditions

The 'Condition' building block serves as a fundamental element in our call flow system, affording administrators fine-grained control over call routing. This building block assesses conditions, yielding 'True' or 'False' results, and guides call routing accordingly. Administrators can establish multiple conditional paths and specify the exit legs for each condition, allowing for intricate routing configurations.

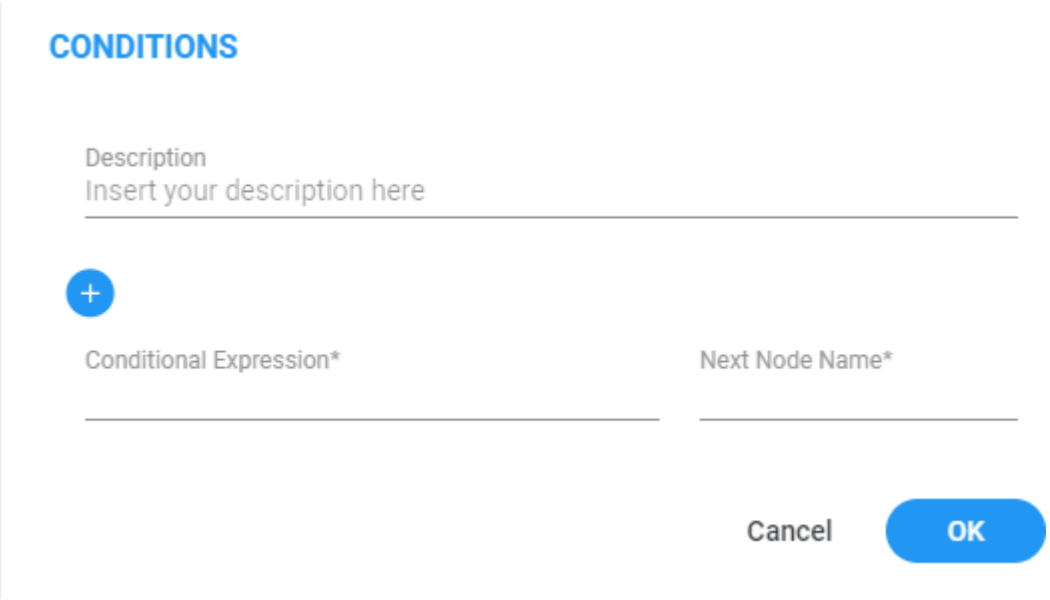
The 'Condition' building block has one exit leg:

- **Default**: When the system has evaluated all configured conditions, and the outcome of each condition is 'False,' the call is automatically routed to this option.

➢ **To use the Conditions building block:**

1. On the left pane, under **Call-Flow Logic**, click **Conditions**; the following Conditions building block appears:

2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. In the 'Conditional Expression' field, enter the expression for evaluation.
   - If the expression evaluated is 'True,' the node proceeds to the associated exit leg.
   - If the expression evaluated is 'False,' the node checks the next evaluated expression.
   - If no expression is evaluated to 'True', the node proceeds to the default exit leg.

   For further details on creating 'Conditional Expression ', see Boolean Expression.
5. In the 'Next Node Name' field, enter the name for the output leg. The name should be written without quotes. Use alphanumeric characters, underscores (_), and periods (.). It must commence with a letter. The maximum length is 24 characters.
6. Click the + button to include additional conditions.
7. Click **OK**, and then **Save**.

Switch

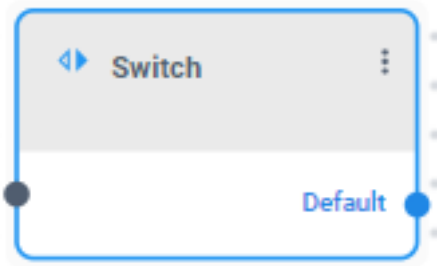
The 'Switch' building block allows you to route program logic to various cases by evaluating a specified expression. Administrators can define multiple values for comparison against the expression and specify the corresponding exit leg, enabling the creation of complex routing configurations.

The 'Switch' building block has one exit leg:

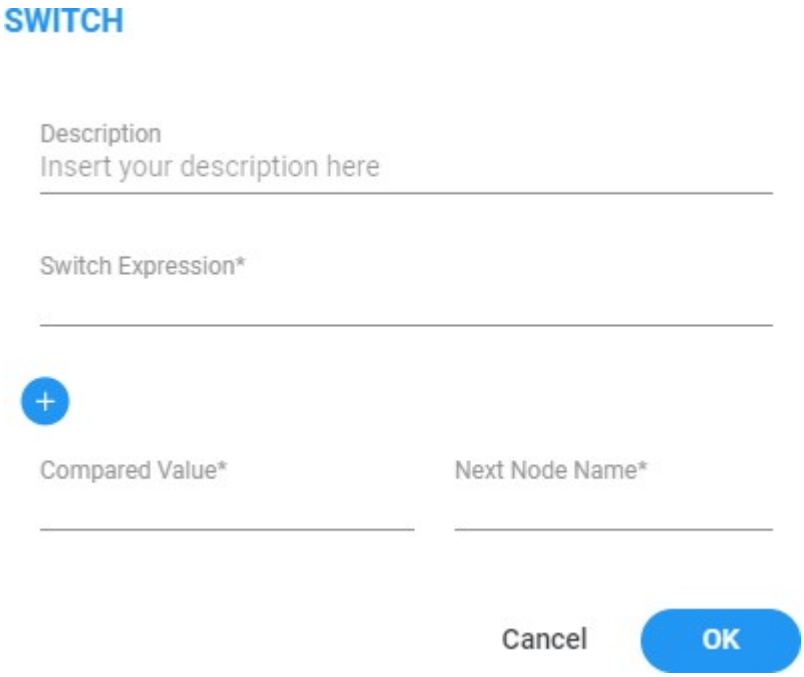
- **Default**: The system evaluates all values against the expression, and none of them yield a 'True' outcome, the call is automatically routed to this designated leg.

➢ **To use the Switch building block:**

1. On the left pane, under **Call-Flow Logic**, click **Switch**; the following Switch building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. In the "Switch Expression" field, enter the expression that requires evaluation. The resulting value is then compared with a predefined list of values.
   - If a match is identified, the node proceeds along the corresponding path.
   - If there is no match, the node proceeds to the default path.

   Note, '0' is a valid option.
5. In the 'Compared Value' field, enter the value to be compared against the result obtained from the 'Switch Expression.
6. In the 'Next Node Name' field, enter the name for the output leg. The name should be written without quotes. Use alphanumeric characters, underscores (_), and periods (.). It must commence with a letter. The maximum length is 24 characters.

7. Click the  button to add additional conditions.
8. Click **OK**, and then **Save**.

Counter

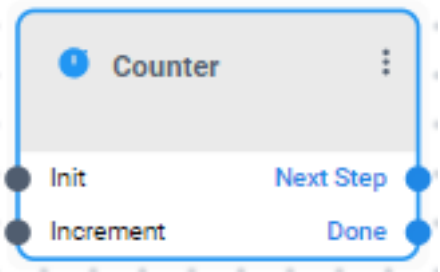
The 'Counter' building block lets you establish loops within your call flow. Loops start and end at defined points and can repeat specific actions multiple times. When the loop finishes, additional actions can be triggered.

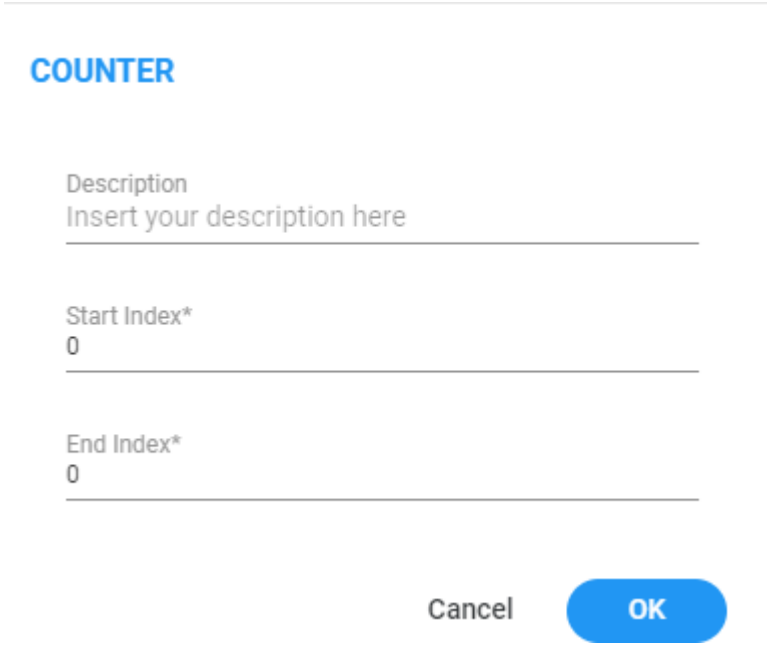
The 'Counter' building block has four exit legs:

- **Init**: Indicates the starting entry point to the building block. When you connect your building block to this leg, the system automatically assigns the start index. The call will then be directed to 'Next Step'.
- **Increment**: Indicates the entry point for incrementing the index. When you connect your building block to this leg, the system automatically retrieves the assigned index and increases it by 1.
- **Next Step**: Indicates an exit point when the current index is not yet equal to the end index.
- **Done**: Indicates an exit point when the current index is equal to the end index.

➢ To use the Counter building block:

1. On the left pane, under **Call-Flow Logic**, click **Counter**; the following Counter building block appears:





2. Click the icon; the following appears:
3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. In the 'Start Index' field, enter the initial index value to be assigned when the 'Init' path is activated within the counter building block.
5. In the 'End Index' field, enter the index that will be compared against the index assigned to the call when the call reaches the counter building block.
6. Click **OK**, and then **Save**.

Set Variable

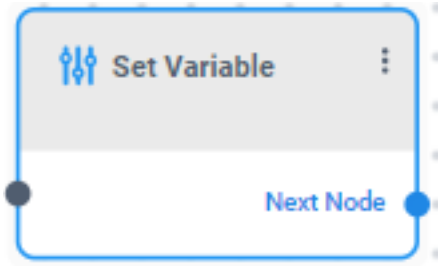
The 'Set Variable' building block allows you to configure variables for later use in your call flow.

The 'Set Variable' building block has one exit leg:

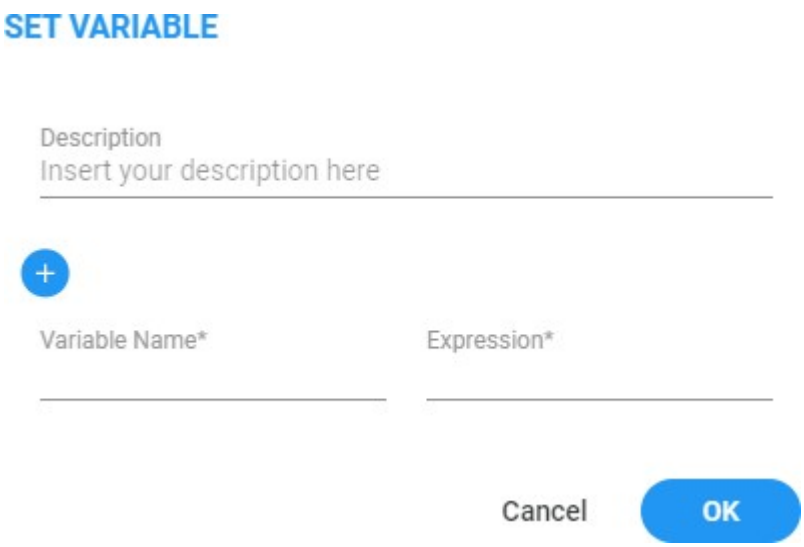
- **Next Node**: This is triggered when the system has completed the evaluation of all expressions and stored them as variables.

➢ To use the Set Variable building block:

1. On the left pane, under **Call-Flow Logic**, click **Set Variable**; the following Set Variable building block appears:



2. Click the icon; the following appears:



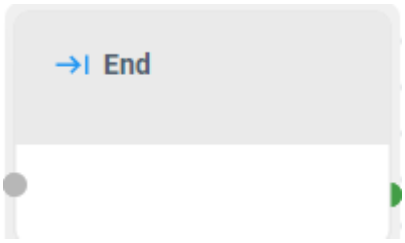
3. In the 'Description' field, enter a description for this building block (up to 50 characters).
4. In the 'Variable Name' field, enter the variable name in the format ${var_name}. Note, when using the variable later in your workflow, it is case-sensitive manner. This means the search treats upper and lowercase letters differently, and any disparity may result in potential issues.
5. In the 'Expression' field, enter the expression to be evaluated.
6. Click the button to add more variables.
7. Click **OK**, and then **Save**.

End

The 'End' building block represents the end point of the flow.

➢ **To use the End building block:**

1. On the left pane, under **Call-Flow Logic**, click **End**; the following End building block appears:

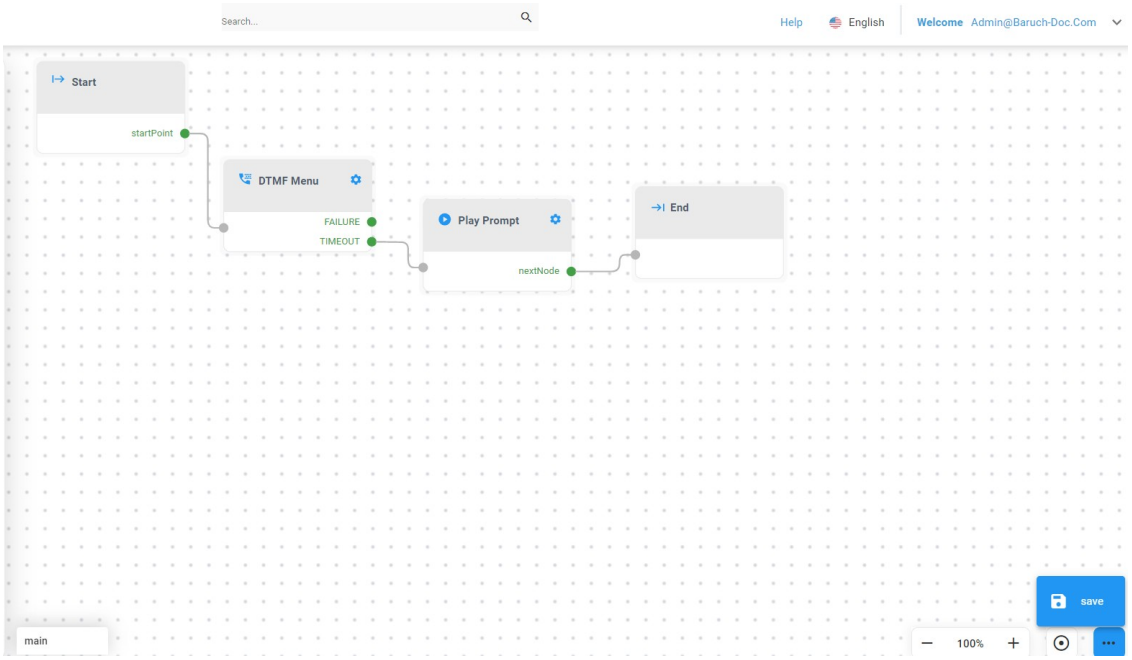

End

7 Save

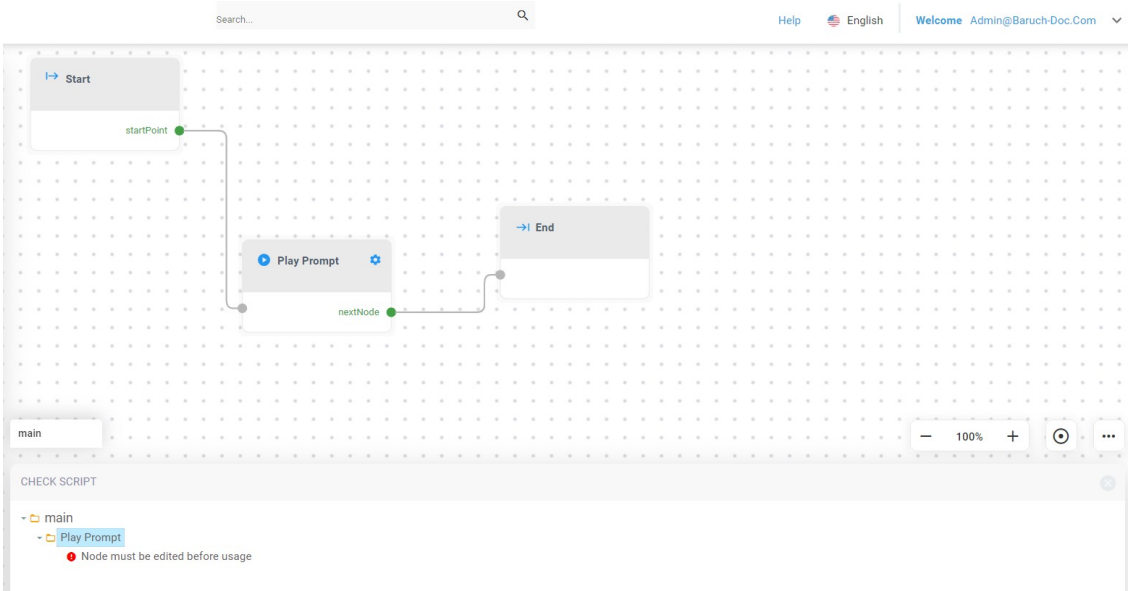
The procedure below describes how to save a script.

To save a script:

1. On the lower-right part of the main flow designer workspace, click the **ellipsis icon** (three dots), and then click **Save**.



A CHECK SCRIPT validation window appears on the lower part of the screen to display script errors (if any). This feature allows you to easily locate errors and fix them.

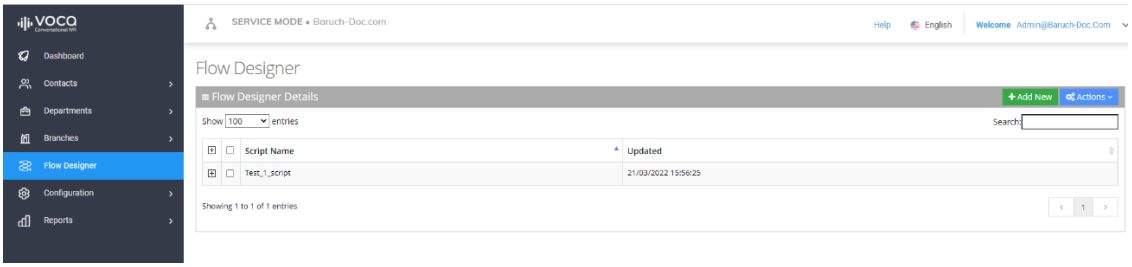


8 Search

The procedure below describes how to use the Search engine.

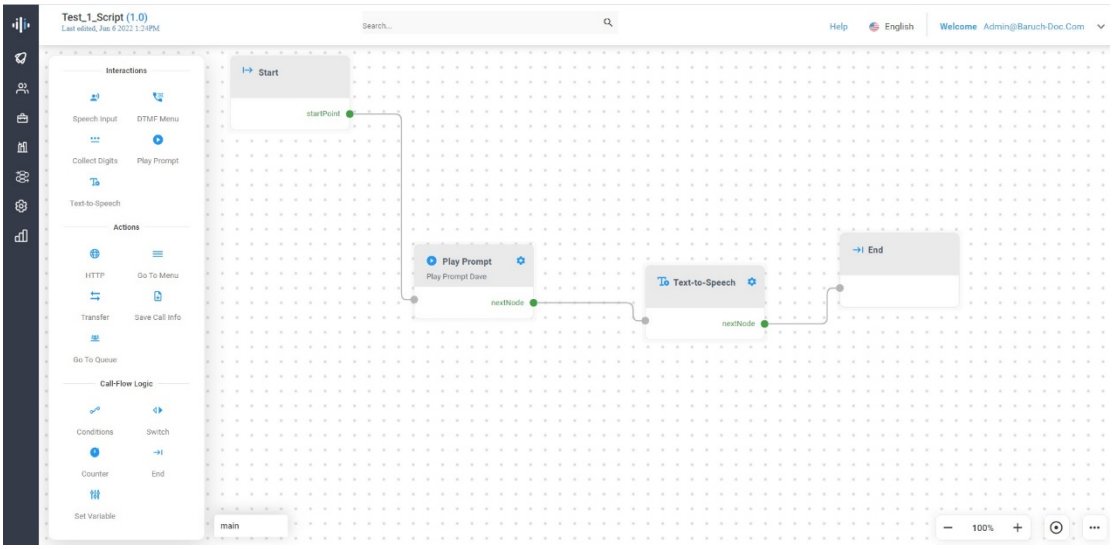
To use the Search engine:

1. From the Navigation pane, click **Flow Designer**; the Flow Designer page opens:

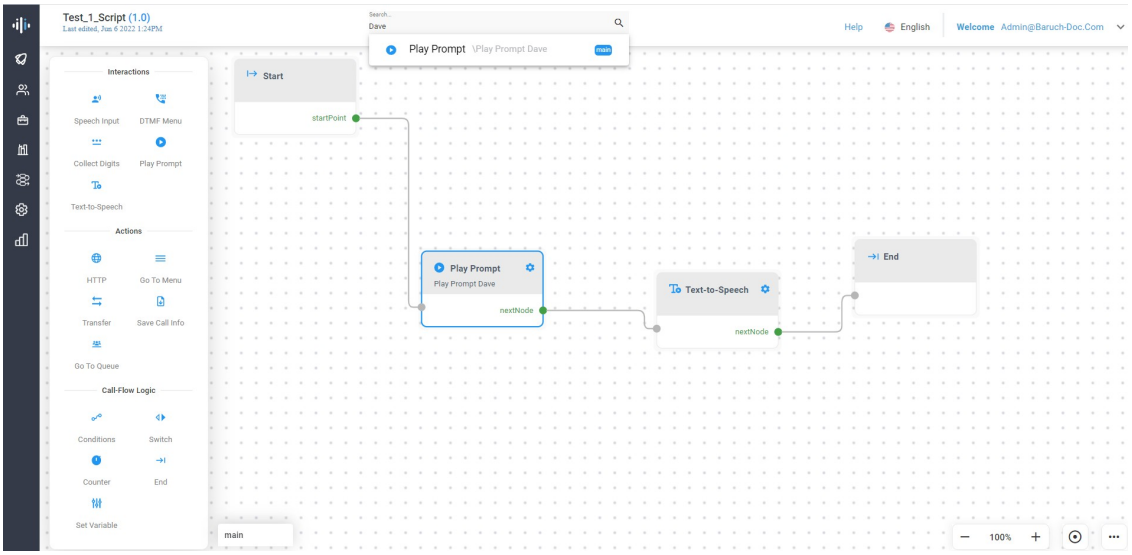


2. Select the script you want to edit, by clicking the corresponding plus box; the **edit** link appears under the selected script:

3. Click **edit**; the main flow designer workspace appears:



4. In the 'Search' field, enter the search string you are looking for, and then press **Enter**; the building block containing the search string is highlighted. In the example below, the building block containing the search string "Dave", is highlighted.


Test_1_Script (1.0)
Last edited, Jun 6 2022 1:24PM
Search...
Dave
Play Prompt \Play Prompt Dave
main
Help
English
Welcome Admin@Baruch-Doc.Com
Interactions
Speech Input
DTMF Menu
Collect Digits
Play Prompt
Text-to-Speech
Actions
HTTP
Go To Menu
Transfer
Save Call Info
Go To Queue
Call-Flow Logic
Conditions
Switch
Counter
End
Set Variable
Start
startPoint
Play Prompt
Play Prompt Dave
nextNode
Text-to-Speech
nextNode
End
main
100%

This page is intentionally left blank.

International Headquarters

6 Ofra Haza Street

Naimi Park

Or Yehuda, 6032303, Israel

Tel: +972-3-976-4000

Fax: +972-3-976-4040

AudioCodes Inc.

80 Kingsbridge Rd

Piscataway, NJ 08854, USA

Tel: +1-732-469-0880

Fax: +1-732-469-2298

Contact us: https://www.audiocodes.com/corporate/offices-worldwide

Website: https://www.audiocodes.com/

Documentation Feedback: https://online.audiocodes.com/documentation-feedback

Document #: LTRT-12996