User's Manual

AudioCodes Voice.AI Solutions

Voca Interaction Center Flow Designer

Cloud-Based & On-Premises Applications

Version 10

Notice

Date Published: June-19-2022

WEEE EU Directive

Pursuant to the WEEE EU Directive, electronic and electrical waste must not be disposed of with unsorted waste. Please contact your local recycling authority for disposal of this product.

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 Administrator's Guide
Voca Installation Manual
Voca Release Notes

Document Revision Record

LTRT	Description
12990	Initial document release for Version 10

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

Table of Contents

1 Introduction

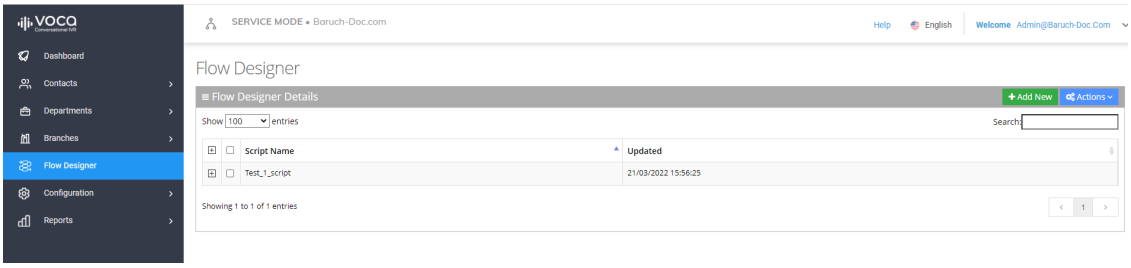
The purpose of this document is to get Voca Administrators familiar with the new Voca Flow Designer, that offers a new way to configure, design and manage complex call flows. The Flow Designer provides a rich and powerful set of building blocks that Administrators can use to create their own call flow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

2 Accessing Flow Designer

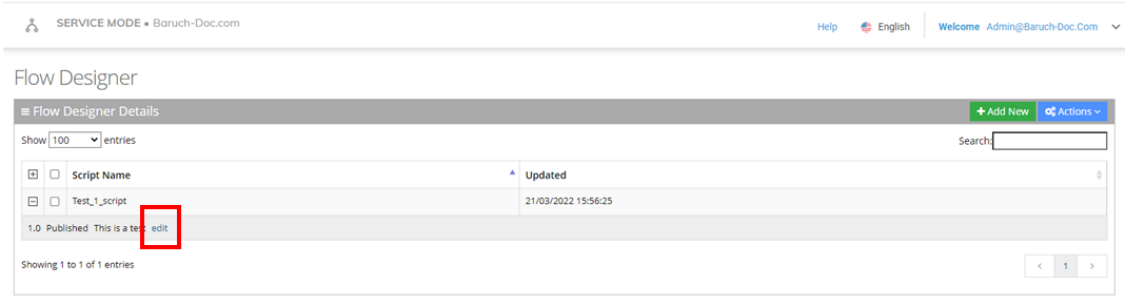
The Flow Designer page in Voca's user interface offers a way to configure, design and manage complex call flows using a powerful set of building blocks.

➢ To access the Flow Designer:

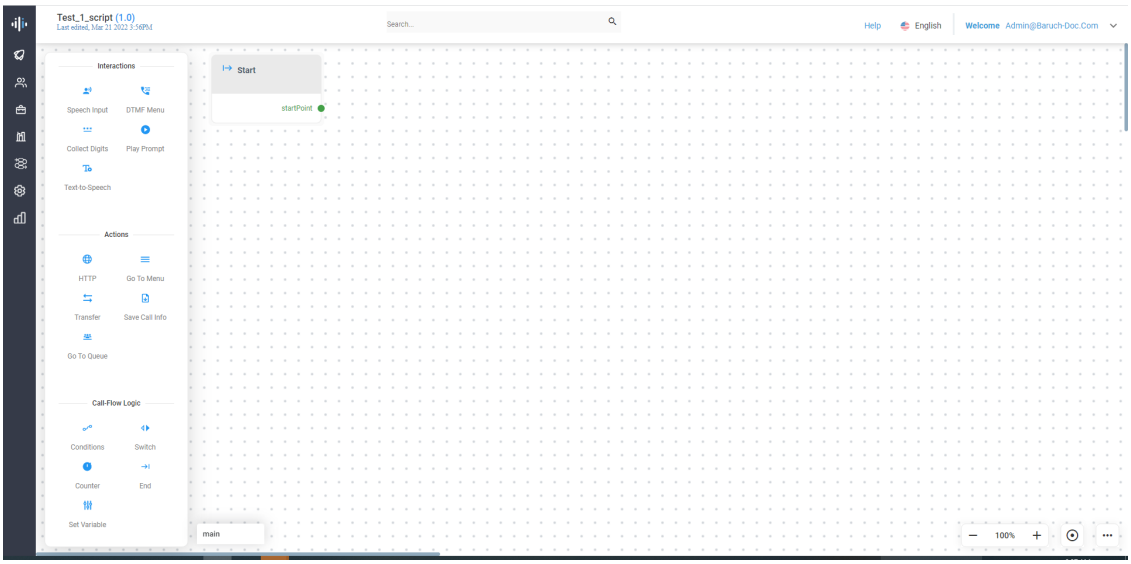
1. Log in to the Voca user interface.
2. From the Navigation pane, click **Flow Designer**; the Flow Designer page opens:



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



4. Click **edit**; the main flow builder workspace appears:

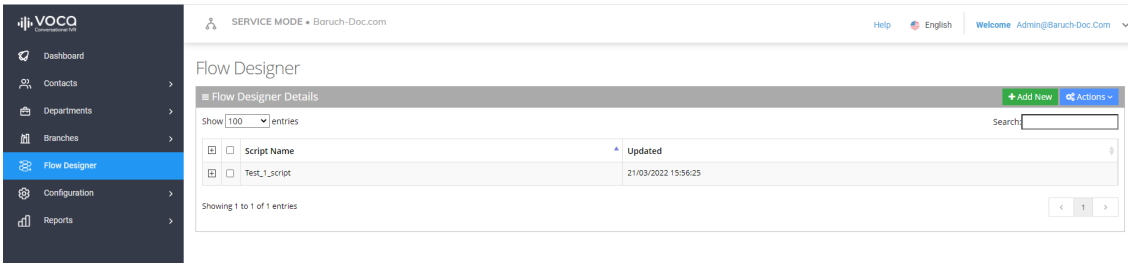


Adding a New Script

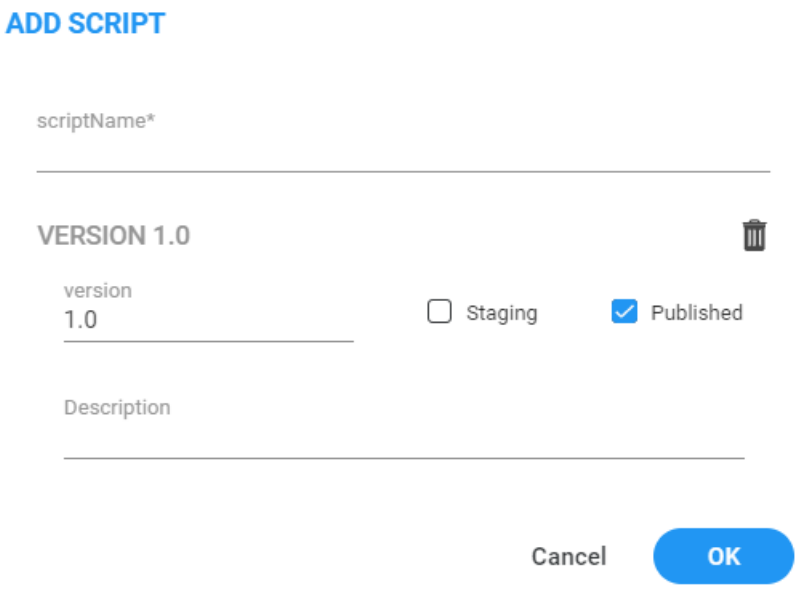
The procedure below describes how to add a new script.

➢ **To add a new script:**

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



2. Click **Add New**; the following appears:



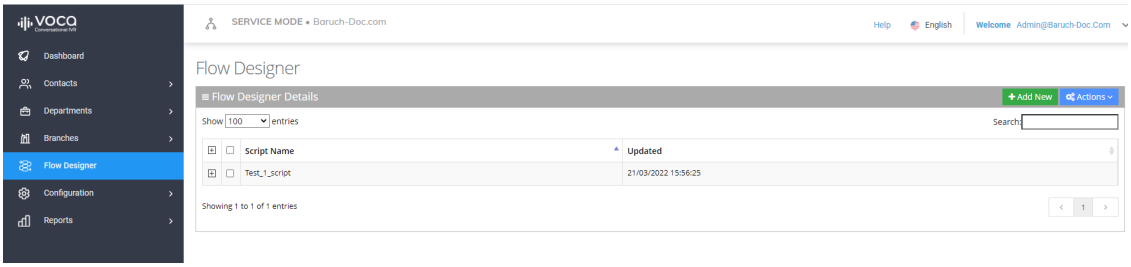
3. In the 'scriptName' field, enter the name of the script.
4. In the 'Version' field, enter the name of the version applicable to the script.
5. Select the 'Staging' check box, if the script is still being developed.
6. Select the 'Published' check box, if the script has been completed and published.
7. In the 'Description' field, enter a description of the script.
8. Click **OK**.

Editing a Script

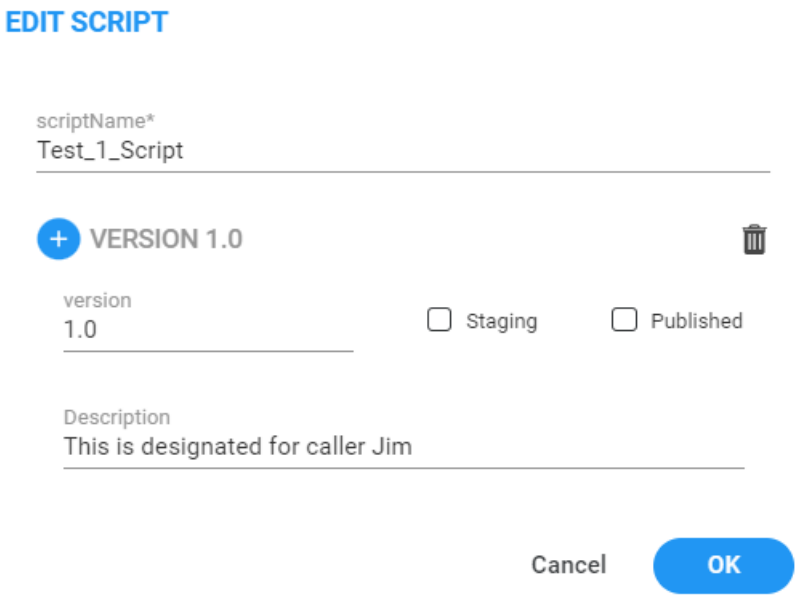
The procedure below describes how to edit a script.

➢ To edit a script:

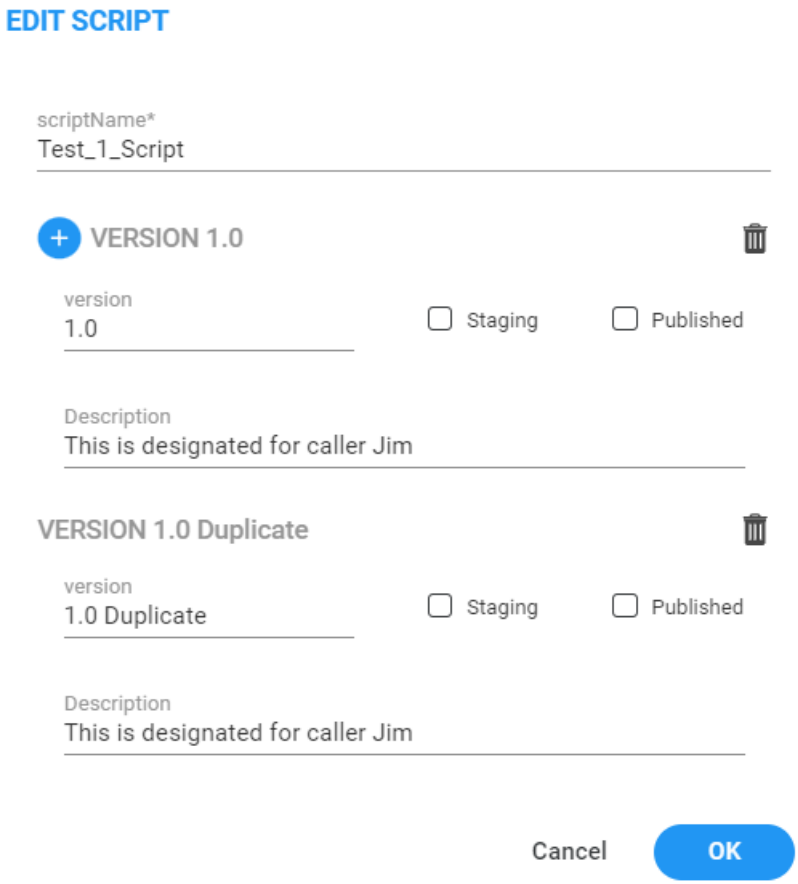
1. From the Navigation pane, click **Flow Designer**; the Flow Designer page opens:



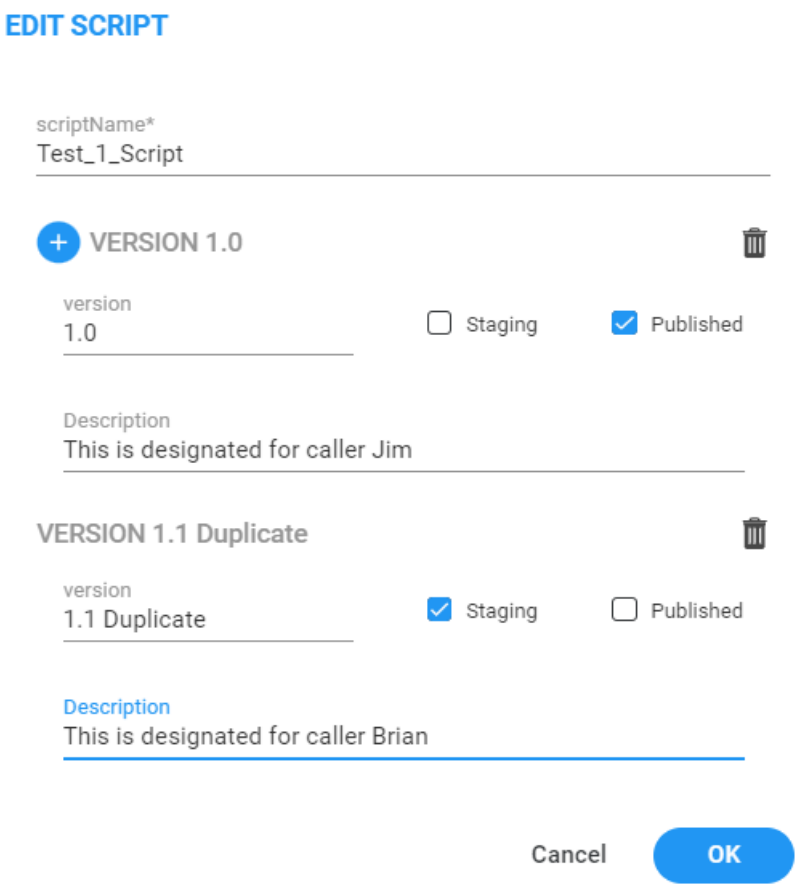
2. Select the script you wish to edit by enabling the script name check box.
3. From the 'Actions' drop-down menu, choose **Edit**; the following appears:



4. It is possible to have more than one script - i.e., you can have one script for **Staging** (in development) and one script for **Published** (completed script). To duplicate the script, click the + plus button; the following appears:

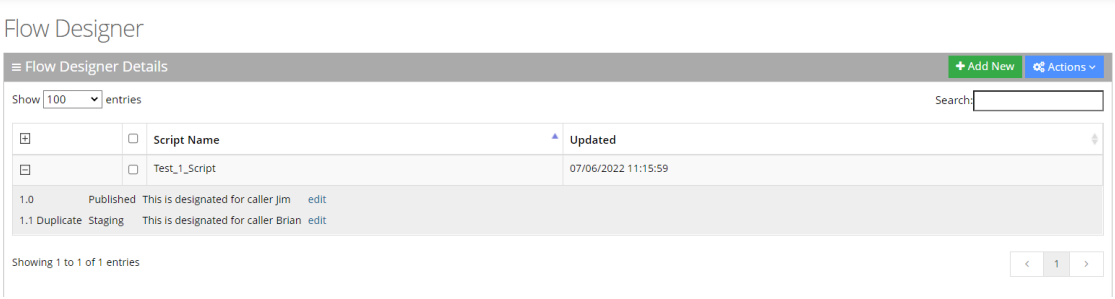


5. Click the garbage icon to remove a duplicate script.
6. Make your necessary changes; for example:



Only one script can be set to **Published**.

7. Click **OK**.
8. On the Flow Designer page, select the script entry you wish to expand by clicking ⊞ plus.



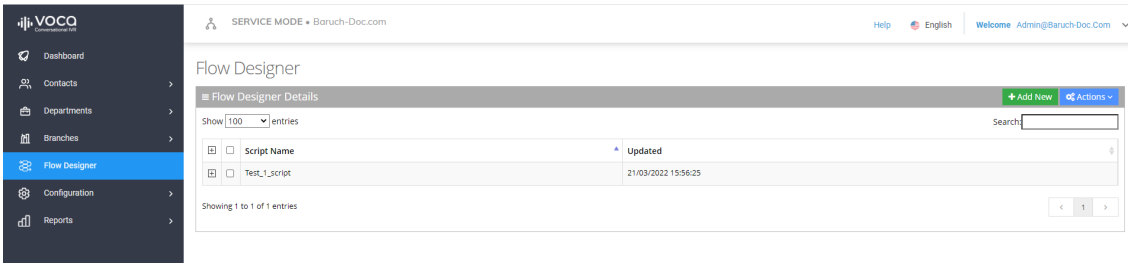
9. Select the script you wish to edit.

Deleting a Script

The procedure below describes how to delete a script.

➢ To delete a script:

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



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

3 Variable Syntax

The Flow Builder uses variables to store values collected or calculated during the call flow. Variable syntax across the Flow Builder 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

4 Expressions

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

- Arithmetic - see Arithmetic below
- String - see String below
- Boolean - see Boolean below
- Functions - see Functions on the next page

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 to create the following text-to-speech prompt:

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

- You can store two Speech input results as one:

  `${speechResult1} + " " + ${speechResult2}`

  The above expression can be used in SetVariable (for example) as the value of the variable.

Boolean

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

For example:

- `${ticketPrice} > 30`
- `${selectedShow} == "Friends"`

Voca supports the following Boolean operators:

- Greater / less than:
  - >
  - <
- Greater / less than or equals to:
  - >=

- <=

- Equals / not equals:
  - ==
  - !=

Functions

You can also use any supported function (see Supported Functions on page 11).

For example:

- `Trim(${selectedShow}) + " " + MakeUpper(${showDayOfTheWeek})`
- `MakeUpper(${selectedShow}) == "FRIENDS"`

5 Supported Functions

The following are different types of supported functions that you can use:

Contains

The Contains function checks to see if the provided text contains the searchText and returns 'True' if the text is contained. Otherwise, it returns 'False'.

Syntax

```
Contains(string text,string textToSearch)
```

Example

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

Date

The Date function receives a text string and returns a date object after trying to parse the text. It returns a NULL if the text provided is not string type or if the parse has failed for any reason (e.g., the text is not in date format).

Syntax

```
Date(string text)
```

Examples

```
Date("08/24/1981 16:22:53")
Date("1981-08-24T04:22:53.53")
Date("1981-08-25T00:00:00")
Date("Monday, 24 August 1981")
Date("16:33:44")
```

DateConvert

The DateConvert function receives two text objects - one representing a date, and the other is the requested output date format. The function then converts from one format to the requested one.

Syntax

```
DateConvert(string date, string outputDateFormat)
```

Example Usage

```
DateConvert("08/24/1981","MM/dd/yyyy")
```

DateParse

The DateParse function receives a date object and a date format as a string and returns the date object as a string in the provided format.

Syntax

```
DateParse(Date date, string dateFormat)
```

Example

```
DateParse(${date},"MM/dd/yyyy") -> when date is a date object saved in the context
```

GetJsonValue

The GetJsonValue function receives a string representation of a JSON and a string of the path within the JSON. This function parses the JSON into an object and extracts the value from the representing path. If the path cannot be found, the function returns a NULL.

Syntax

```
GetJsonValue(string json, string path)
```

Example

Assuming we have the following JSON that is saved in the context under the variable name 'ticket':

```
{
"ticket" :{
"price" : 30,
"currency" : "ILS",
"row" : 3,
"seat": 24
}
}
```

To get the ticket price, use the function as follows:

`GetJsonValue(${ticket},"ticket.price")`.

This returns '30'.

The same applies for currency:

`GetJsonValue(${ticket},"ticket.currency")`.

This return "ILS".

Length

This function provides the length of a given string ('-1' if the string is null or empty).

Syntax

```
Length(string text)
```

Example

```
Length("ido") -> returns '3'
Length(".net5.0") -> returns '7'
```

Lower

The Lower function receives a string and returns it all in lower case.

Syntax

```
Lower(string text)
```

Example

```
Lower ("HERSHKOVITZ") -> hershkovitz
```

```
Lower ("iDo") -> ido
```

```
Lower ("HellO WoRlD") -> hello world
```

Now

The Now function returns a date representing the present time in the tenant time zone.

Syntax

```
Now()
```

Example Usage

```
Now()
```

NowUtc

The NowUtc function returns a date representing the current time in the UTC time zone.

Syntax

```
NowUtc()
```

Example

```
NowUtc()
```

SubString

The SubString function has two overrides:

- It receives a text and integer startIndex. It returns the substring of the text provided from the starting index till the end of the string.
- It receives a text, integer startIndex and an integer length. It returns the substring of the text provided from the starting index with the length provided.

The SubString function returns a null if the text provided is not a string and if the text provided is null. This function returns the provided text input in the following cases:

- startIndex is equal or greater than the text length
- The length requested is equal or greater than the provided text length
- The sum of the startIndex and the length requested are equal or greater than the provided text length

---

Syntax

SubString(string text, int startIndex)

SubString(string text, int startIndex, int length)

Example

```
SubString("Ido Hershkovitz",4)" -> "Hershkovitz"
SubString("Ido Hershkovitz",4,3)" -> "Her"
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 "Ido Hershkovitz", you can also use the function as follows:

```
SubString(${text},4)" -> "Hershkovitz"
SubString(${text},4,3)" -> "Her"
SubString(${text},8)" -> "you?"
SubString(${text},4,3)" -> "are"
```

Trim

The Trim function receives a text string text and a direction. It returns a trimmed text, based on direction. The direction is optional and can be one of the following options. If no direction is provided, 'ALL' is used:

- LEFT
- RIGHT
- ALL

Syntax

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

Examples

```
Trim("My name is " , "RIGHT")
Trim(" My name is" , "LEFT")
Trim(" My name is " , "ALL")
Trim(" My name is " , "")
Trim(" My name is ")
```

All the above options provide the "My name is" string, without any spaces at the beginning or end of the provided string.

Upper

The Upper function receives a string and returns it all in upper case.

Syntax

```
Upper(string text)
```

Examples

```
Upper("hershkovitz") -> HERSHKOVITZ
```

```
Upper("iDo") -> IDO
```

```
Upper("HellO WoRlD") -> HELLO WORLD
```

WeekDay

The WeekDay function receives a date and returns the day of the week as an integer (0-6):

- '0' refers to Sunday
- '6' refers to Saturday

Syntax

```
WeekDay(Date date)
```

Example Usage

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

6 Building Blocks

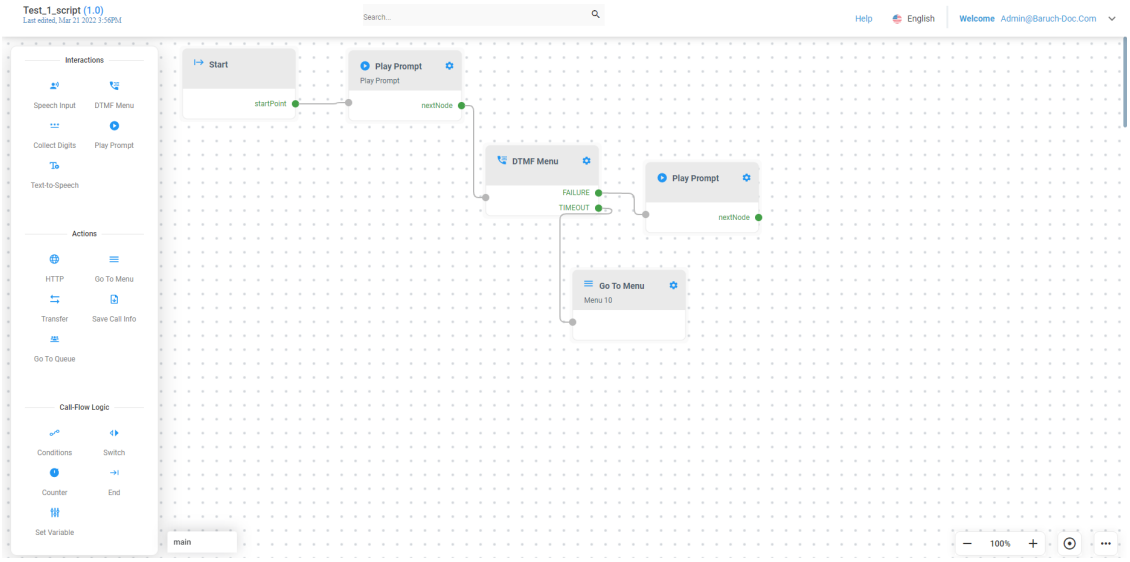
Use the following building blocks to create the call flow logic:

- Interactions
- Actions
- Call-Flow

You can connect the building blocks by placing the cursor on the green 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**.

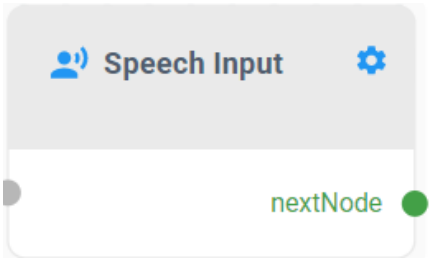
- Speech Input - see Speech Input on the next page
- DTMF Menu - see DTMF Menu on page 19
- Collect Digits - see Collect Digits on page 21
- Play Prompt - see Play Prompt on page 23
- Text-to-Speech - see Text-to-Speech on page 25

Speech Input

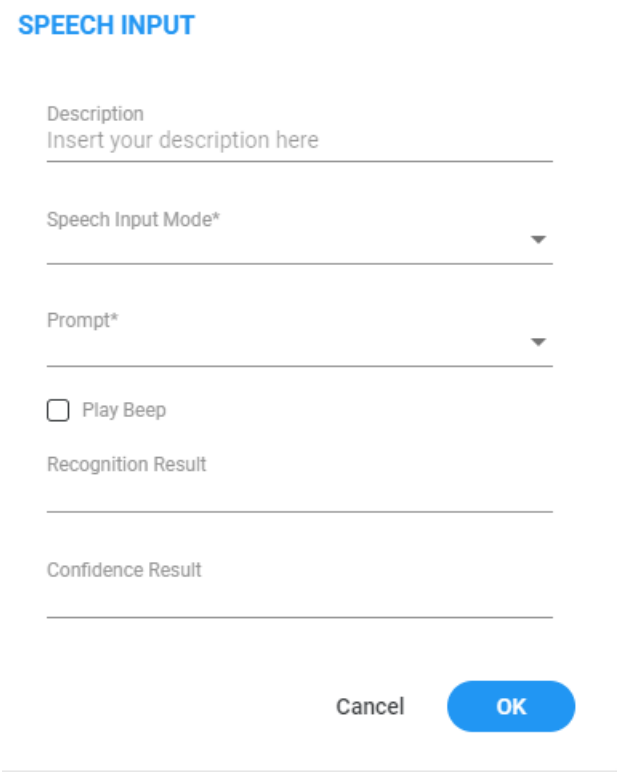
This section describes how to use the Speech Input building block.

➢ **To use the Speech Input building block:**

1. Click the **Speech Input** option under the **Interactions** group; the following Speech Input building block appears:

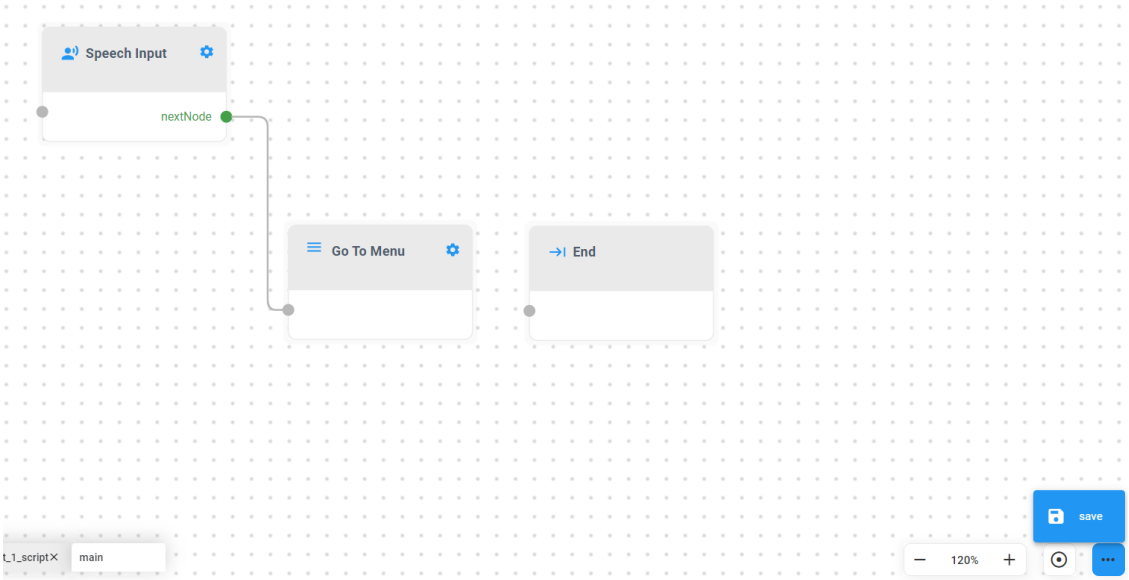


2. Click the icon; the following appears:



3. In the 'Description' field, enter a description of this building block (up to 50 characters).
4. From the 'Speech Input Mode' drop-down list, select the appropriate mode:
   - **Free Speech** returns what the caller said.
   - **Keywords** provides a list of up to 50 phrases. The caller speech input is checked against this list. The matching phrase with the highest confidence is returned. This option appears only if 'Keywords' was selected for ASR mode.
5. From the 'Prompt' drop-down list, select the prompt to be played before detection.

6. (Optional) Select the 'Play Beep' check box to play the beep sound before recognition.
7. (Optional) In the 'Recognition Result' field, enter the variable for holding the speech input result.
8. (Optional) In the 'Confidence Result' field, enter a variable for holding the confidence result.
9. Click **OK**.

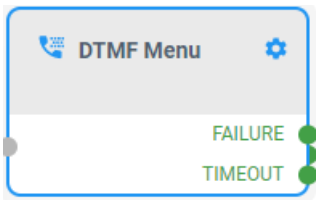


DTMF Menu

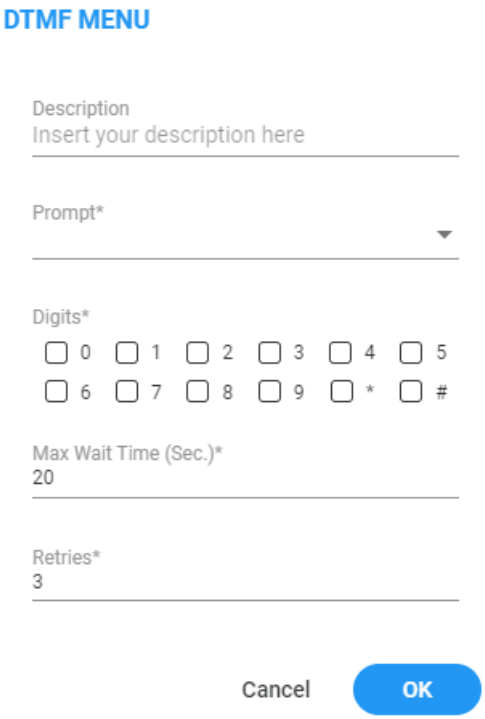
This section describes how to use the DTMF Menu building block.

➢ To use the DTMF Menu building block:

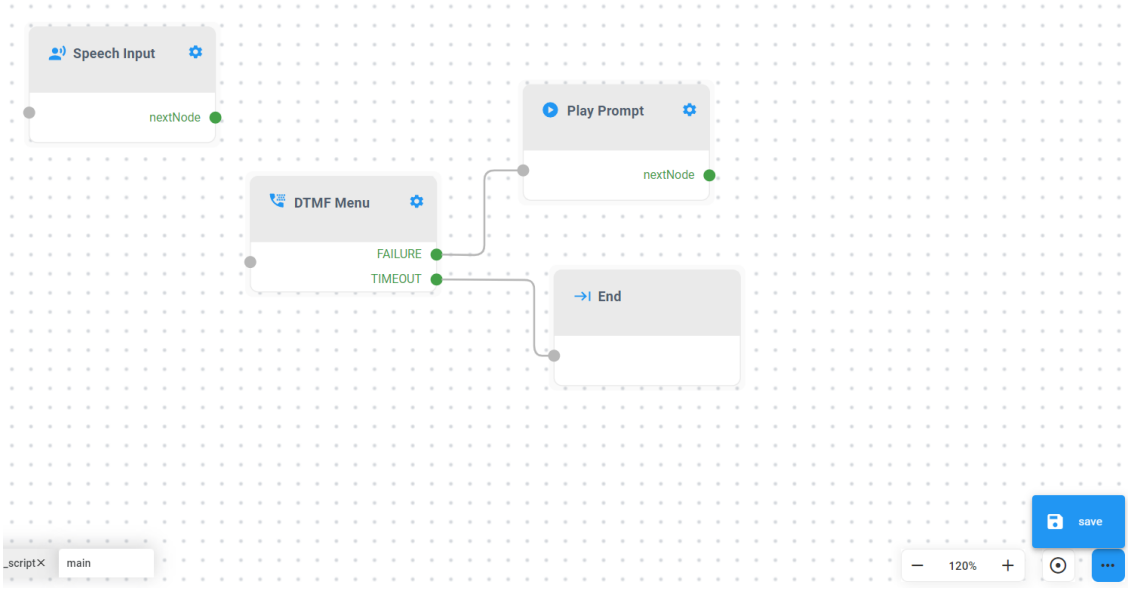
1. Click the **DTMF Menu** option under the **Interactions** group; the following DTMF Menu building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description of this building block (up to 50 characters).
4. From the 'Prompt' drop-down list, select the appropriate prompt:
5. Under the **Digits** group, select the digits that you want to be included in the DTMF menu (only 0-9, *, #).
6. In the 'Max Wait Time (Sec)' field, enter the maximum waiting time for input (1-30, default 10).
7. In the 'Retries' field, enter the maximum number of retries, to repeat the block in case DTMF is not detected (1-7, default 3).
8. Click **OK**.

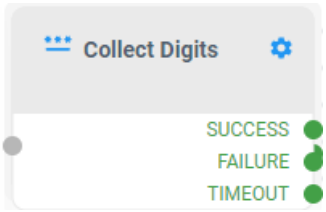


Collect Digits

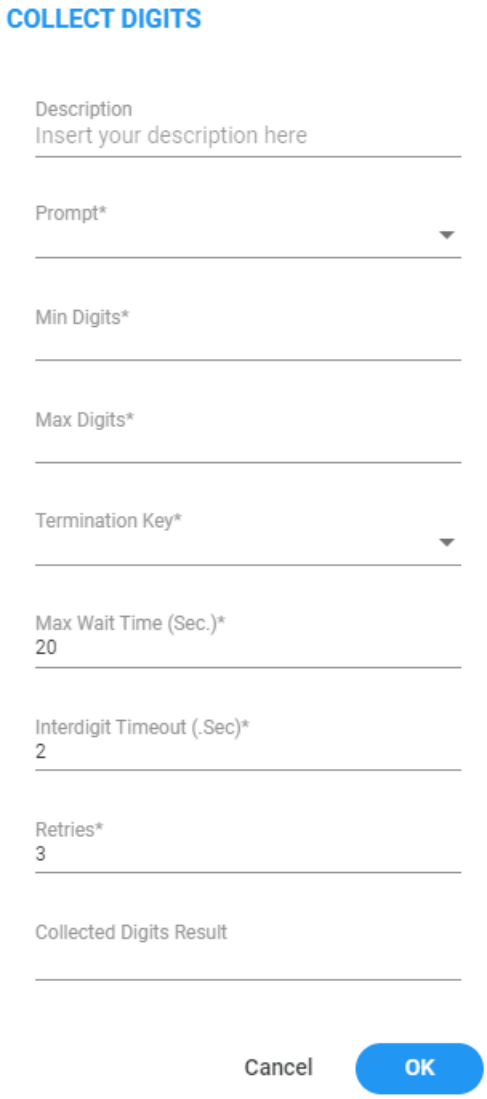
This section describes how to use the Collect Digits building block.

➢ **To use the Collect Digits building block:**

1. Click the **Collect Digits** option under the **Interactions** group; the following Collect Digits building block appears:



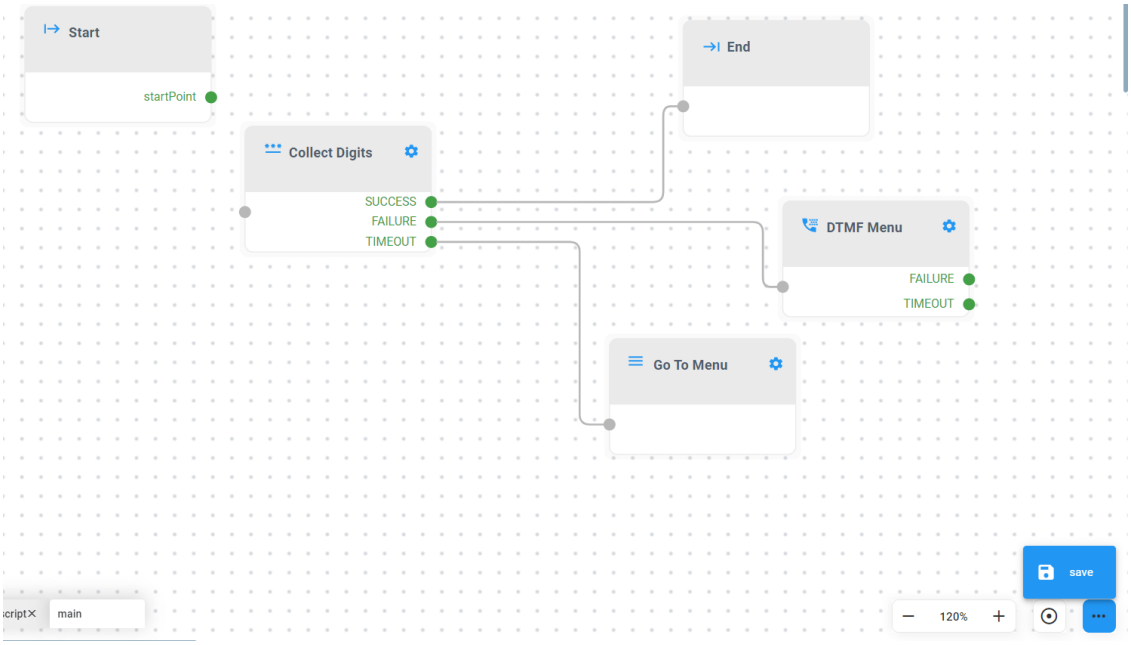
2. Click the icon; the following appears:



3. In the 'Description' field, enter a description of this building block (up to 50 characters).
4. From the 'Prompt' drop-down list, select the appropriate prompt.
5. In the 'Min Digits' field, enter the minimum number of digits (in seconds) to be collected (1-30).
6. In the 'Max Digits' field, enter the maximum number of digits (in seconds) to be collected (1-30). This number should be higher than 'Min Digits'.
7. (Mandatory) In the 'Max Wait Time (sec)' field, enter the maximum waiting time for input (1-30, default 10).
8. (Mandatory) In the 'Interdigit Timeout (sec)' field, enter the allowed waiting time between digits (1-30, default 10).
9. In the 'Retries' field, enter the maximum number of retries to repeat the block if the DTMF is not detected.

10. (Optional) In the 'Collected Digits Result' field, enter a variable for holding the collected digits.

11. Click **OK**.

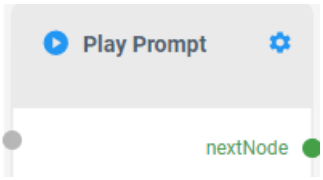


Play Prompt

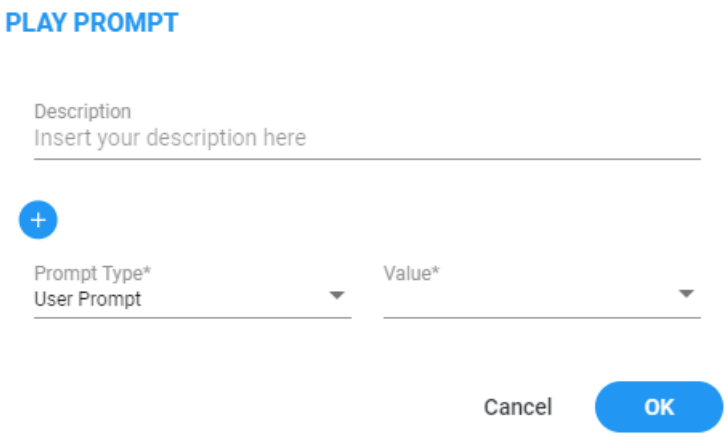
This section describes how to use the Play Prompt building block.

➢ **To use the Play Prompt building block:**

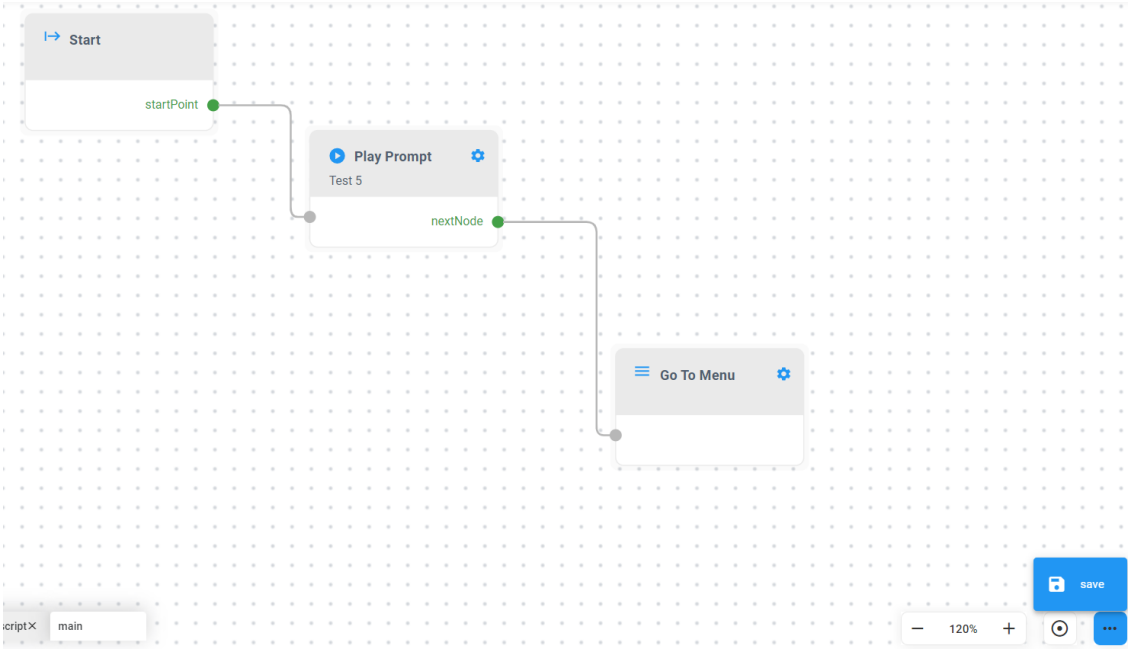
1. Click the **Play Prompt** option under the **Interactions** group; the following Play Prompt building block appears:



2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description of this building block (up to 50 characters).
4. From the 'Prompt Type' drop-down list, select the appropriate prompt:
   - **User Prompt:** Select from a drop-down with configured prompts.
   - **Play Date:** Play the full date. Expression resulting as a date (e.g., “2010-11-20”).
   - **Play Day of the Week:** Play day of the week. Expression resulting as a date (e.g., “2010-11-20”).
   - **Play Day in Month:** Play day in the month. Expression resulting as a date (e.g., “2010-11-20”).
   - **Play Day in Month:** Play day in the month. Expression resulting as a date (e.g., “2010-11-20”).
   - **Play Month:** Play month. Expression resulting as a date (e.g., “2010-11-20”).
   - **Play Year:** Play year. Expression resulting as a date (e.g., “2010-11-20”).
   - **Play Number:** Play number. Expression resulting as a number.
   - **Play Ordinal Number:** Play ordinal number. Expression resulting as a number (e.g., 1st, 2nd).
   - **Play Time:** Play Time. Expression resulting as a time (e.g., “22:12”).
   - **Play Time with Seconds:** Play Time with seconds. Expression resulting as time (e.g., “22:12:15”).
5. In the 'Value' field, enter the appropriate value.
6. Click **OK**.

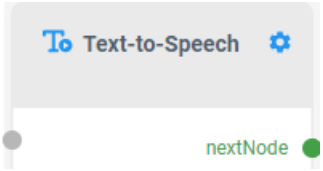


Text-to-Speech

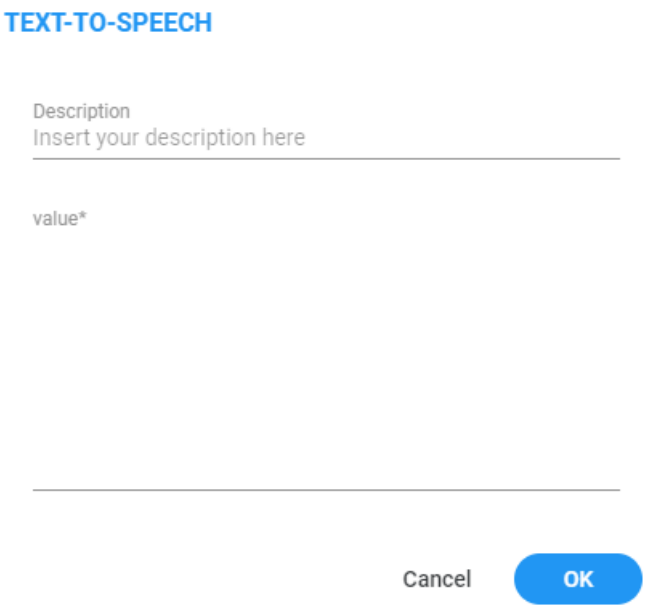
This section describes how to use the Text-to-Speech building block.

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

1. Click the Text-to-Speech option under the **Interactions** group; the following Text-to-Speech building block appears:



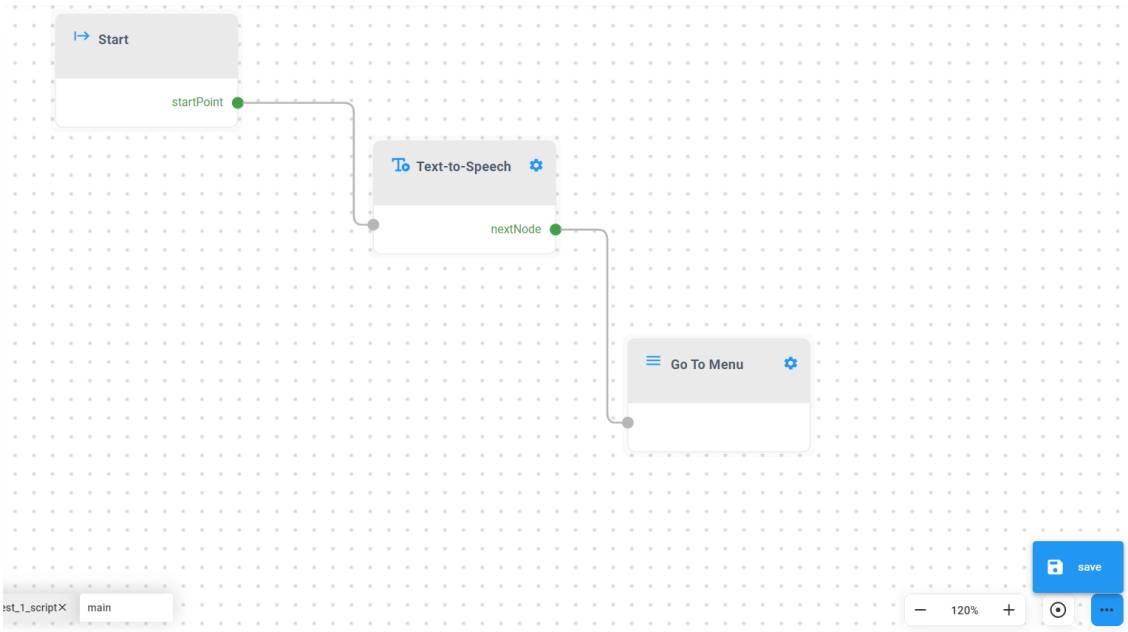
2. Click the  icon; the following appears:



3. In the 'Description' field, enter a description of this building block (up to 50 characters).
4. In the 'Value' field, enter the appropriate expression evaluated to a string.

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

5. Click **OK**.



Actions

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

- HTTP - see HTTP on the next page

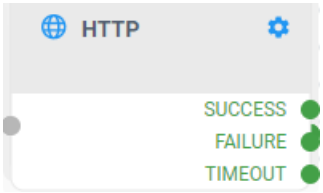
- Go To Menu - see Go To Menu on page 30
- Transfer - see Transfer on page 31
- Save Call Info - Save Call Info on page 32
- Go to Queue - Go To Queue on page 33

HTTP

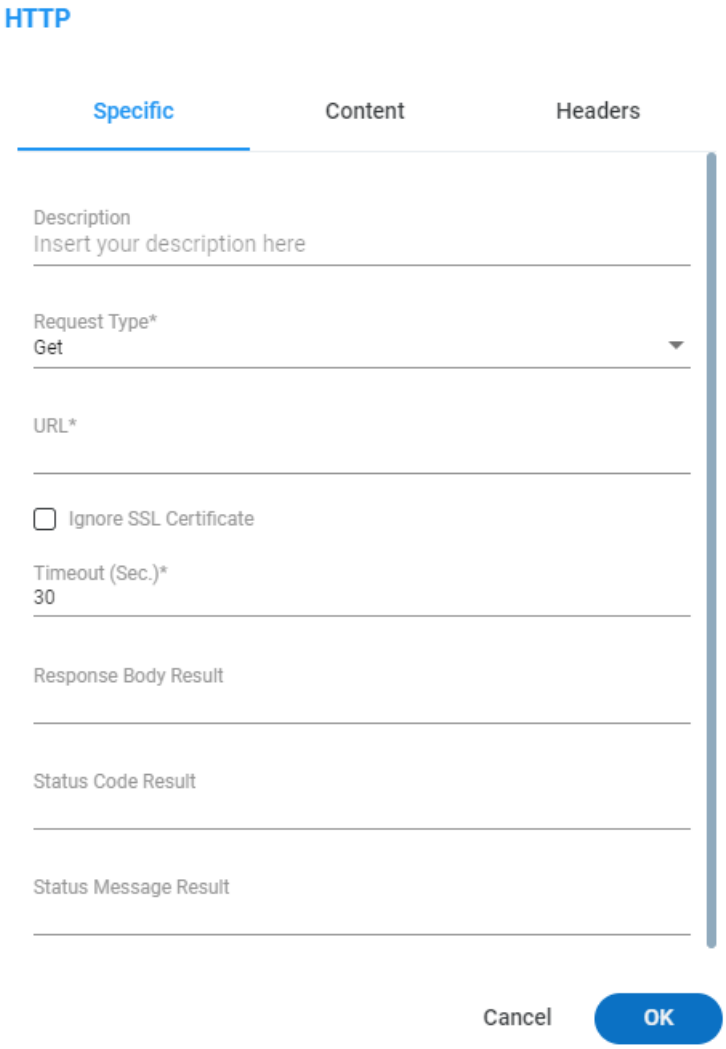
This section describes how to use the HTTP building block.

➢ **To use the HTTP building block:**

1. Click the **HTTP** option under the **Actions** group; 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 of this building block (up to 50 characters).
   b. From the 'Request Type' drop-down list, select the appropriate type of request:
      - **Get**: (Optional) Enter a list of parameters.
      - **Post:** Enter a variable containing the data.
   c. In the 'URL' field, enter the URL string.
   d. Select the 'Ignore SSL Certificate' check box to ignore the certificate. This is applicable to self-signed certificates. **Use with caution**.
   e. In the 'Timeout' field, enter the maximum timeout in seconds, for the request (1-60 seconds). The default value is 20.
   f. (Optional) In the 'Response Body Result' field, enter a variable for holding the response.

g. (Optional) In the 'Status Code Result ' field, enter a variable for holding the returned status code.

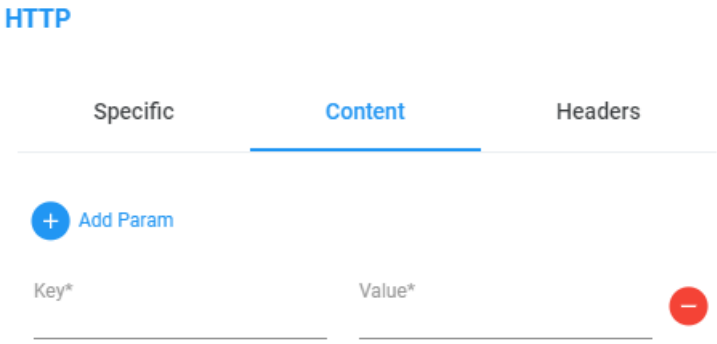
h. (Optional) In the 'Status Message Result ' field, enter a variable for holding the status message.

4. Select the **Content** tab, and then configure the following:

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

b. Enter the 'Key' and 'Value' fields.

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

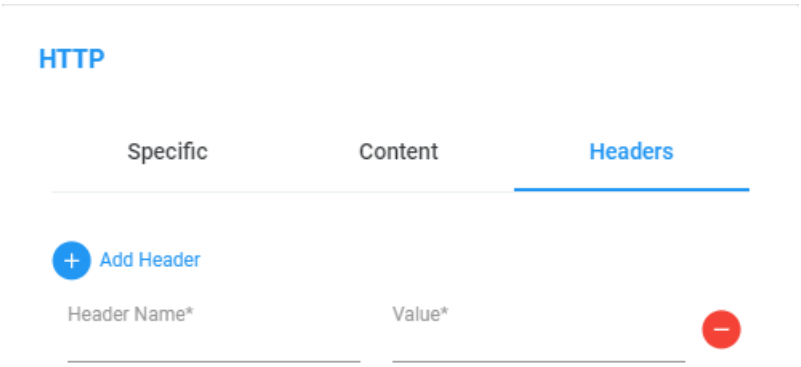


5. Select the **Headers** tab, and then configure the following:

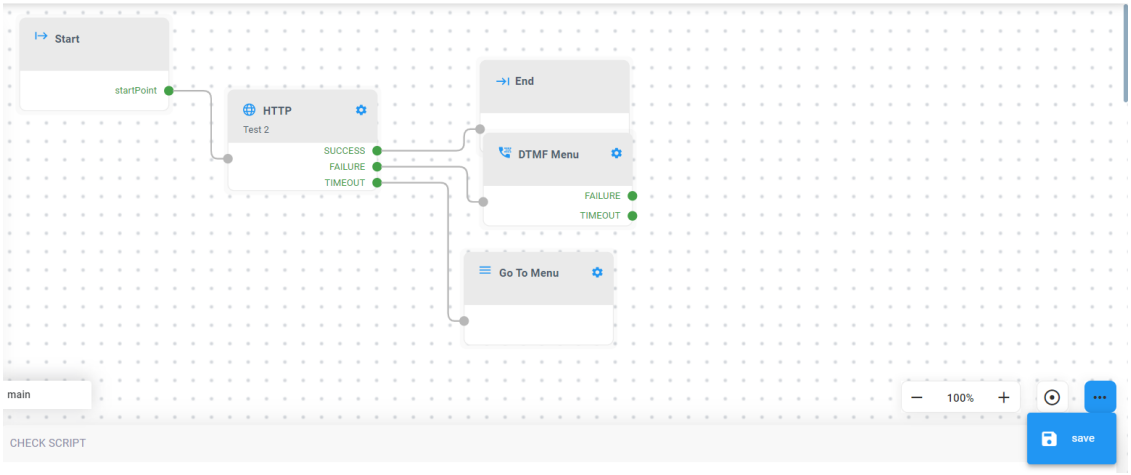
a. Click the **Add Header** icon to add a header.

b. In the 'Header Name', enter a string describing the header key.

c. In the 'Value' field, enter the values as an expression.



6. Click **OK**.

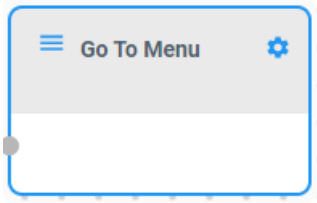


Go To Menu

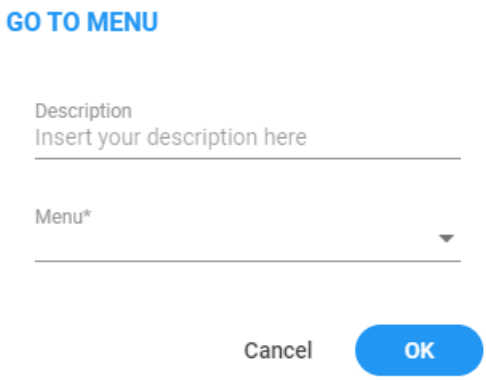
This section describes how to use the Go To Menu building block.

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

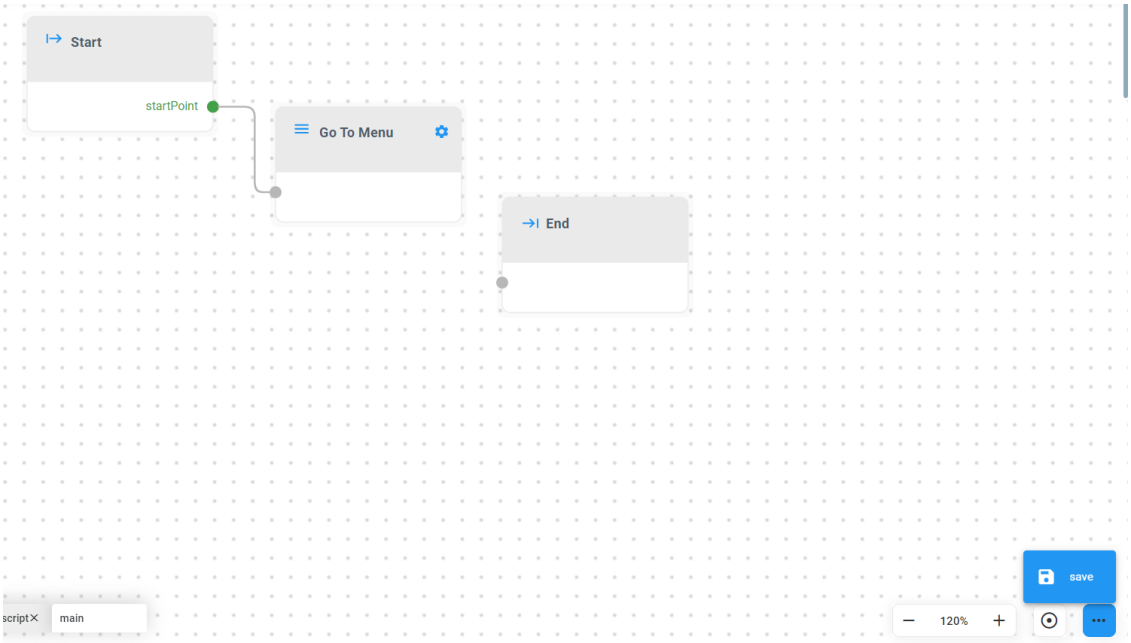
1. Click the **Go To Menu** option under the **Actions** group; the following Go To Menu building block appears:



2. Click the  icon; the following appears:



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

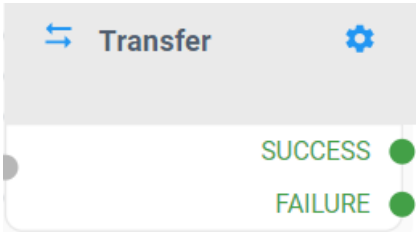


Transfer

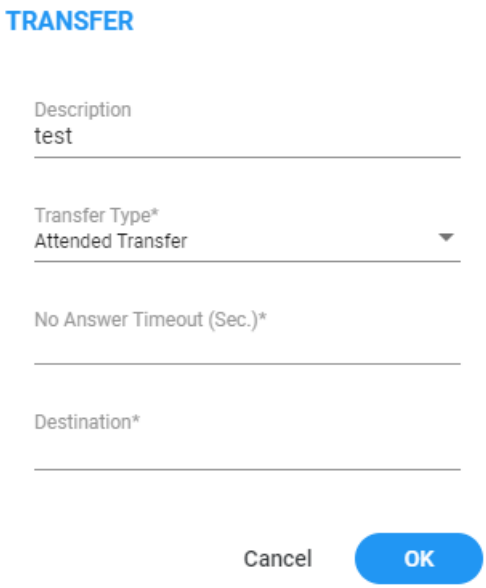
This section describes how to use the Transfer building block.

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

1. Click the **Transfer** option under the **Actions** group; the following Transfer building block appears:



2. Click the  icon; the following appears:



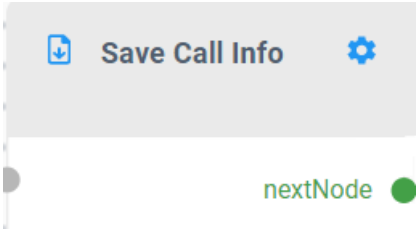
3. In the 'Description' field, enter a description of this building block (up to 50 characters).
4. From the 'Transfer Type' drop-down list, select the appropriate type of transfer:
   - Blind Transfer (default)
   - Attended
5. In the 'No Answer Timeout (Sec)' field, enter maximum time to wait for an answer (1-60 seconds). The default value is 20.
6. In the 'Destination' field, enter the destination as a string.
7. Click **OK**.

Save Call Info

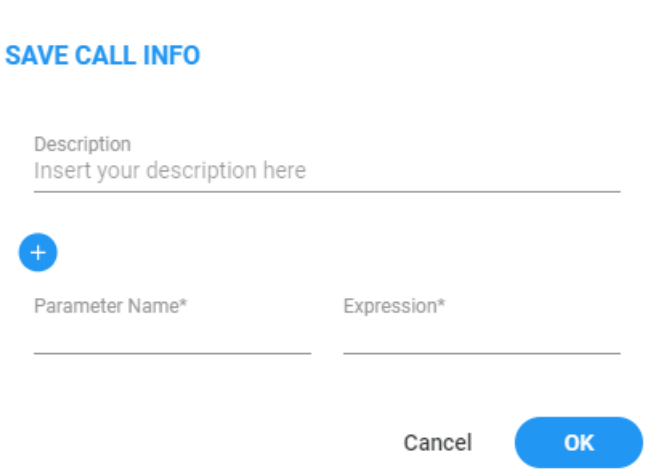
This section describes how to use the Save Call Info building block. This block is used to save collected data before transferring the call to an agent.

➢ **To use the Save Call Info building block:**

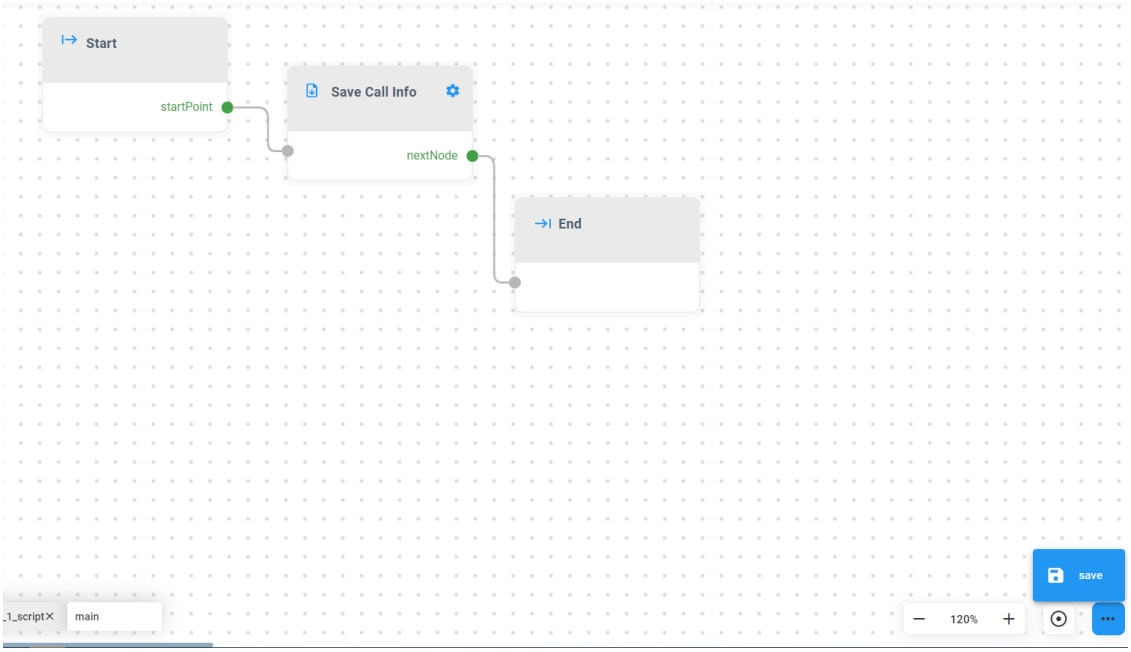
1. Click the **Save Call Info** option under the **Actions** group; the following Save Call Info building block appears:



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 'Parameter Name' field, enter the name of the parameter.
5. In the 'Expression' field, enter the expression.
6. Click **OK**.

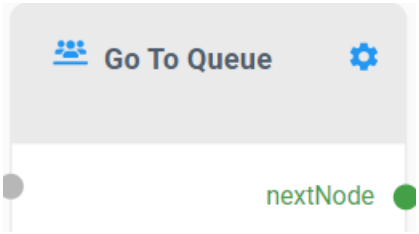


Go To Queue

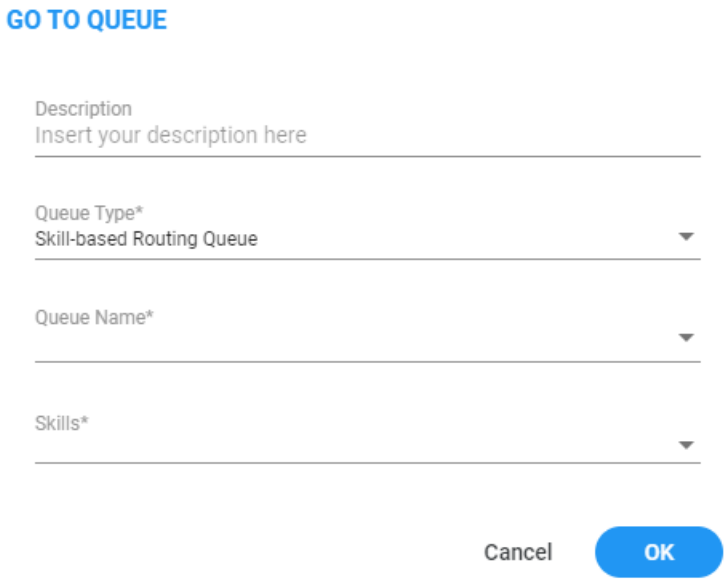
This section describes how to use the Go To Queue building block.

➢ To use the Go To Queue building block:

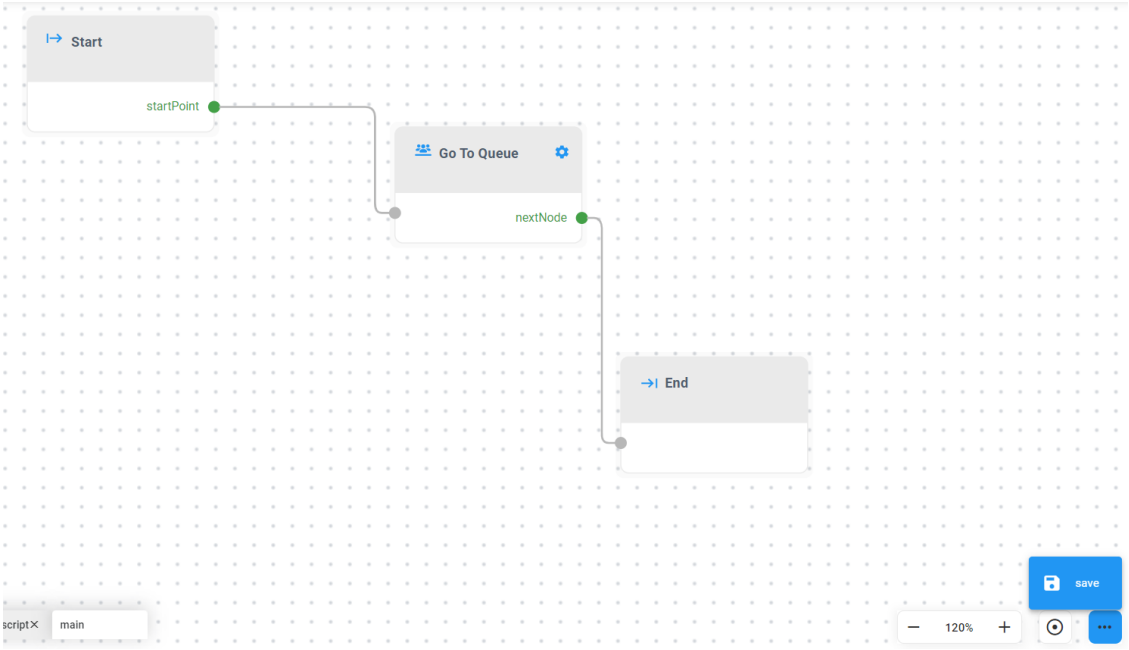
1. Click the **Go To Queue** option under the **Actions** group; the following Go To Queue building block appears:



2. Click the icon; the following appears:



3. In the 'Description' field, enter a description of this building block (up to 50 characters).
4. From the 'Queue Type' drop-down list, select the appropriate type of queue:
   - Queue
   - Skill-based Routing 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 to the queue. This option is shown only if **Skill-Based Routing Queue** is selected.
7. Click **OK**.



Call-Flow Logic

The following building blocks appear under **Call-Flow Logic**:

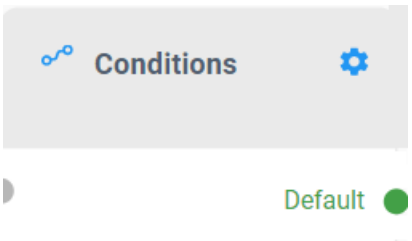
- Conditions - see Conditions below
- Switch - see Switch on the next page
- Counter - see Counter on page 38
- Set Variable - see Set Variable on page 39
- End - see End on page 40

Conditions

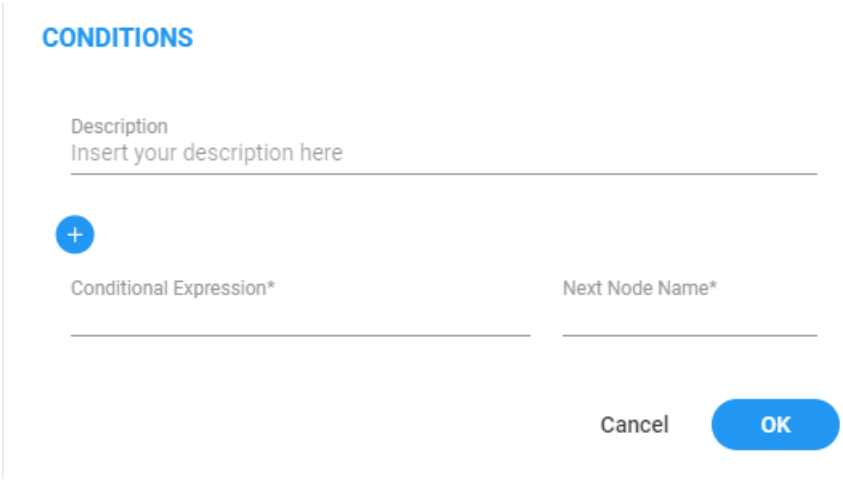
This section describes how to use the Conditions building block.

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

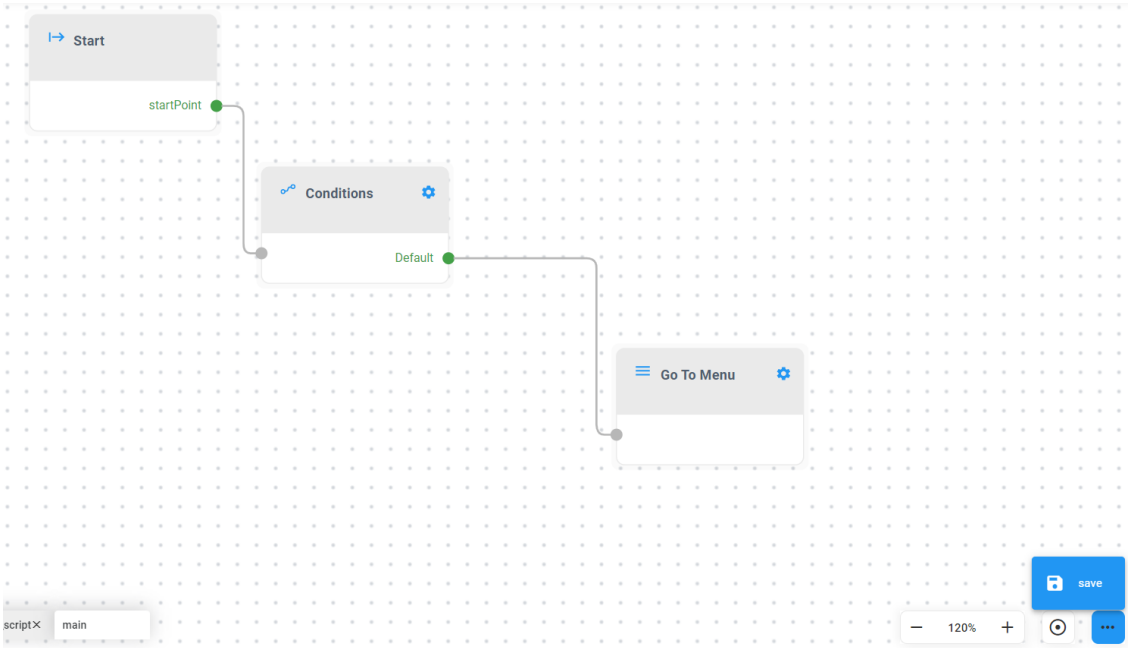
1. Click the **Conditions** option under the **Call-Flow Logic** group; the following Conditions building block appears:



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 'Conditional Expression' field, enter the expression to be evaluated. If the expression evaluated is 'True', the node exits out the corresponding leg. If 'False', it checks the next evaluated expression. If no expression is evaluated to 'True', the node exits out the default leg.
5. In the 'Next Node Name' field, enter the name of the output leg to be created. The name should be written without quotes. Use alphanumeric, underscore (_), hyphen (-), space. It must start with an alphanumeric. The maximum length is 24.
6. Click **OK**.

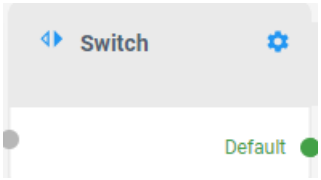


Switch

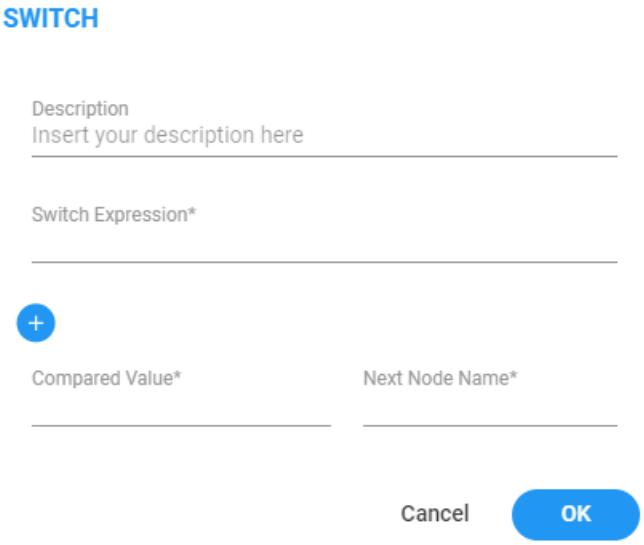
This section describes how to use the Switch building block.

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

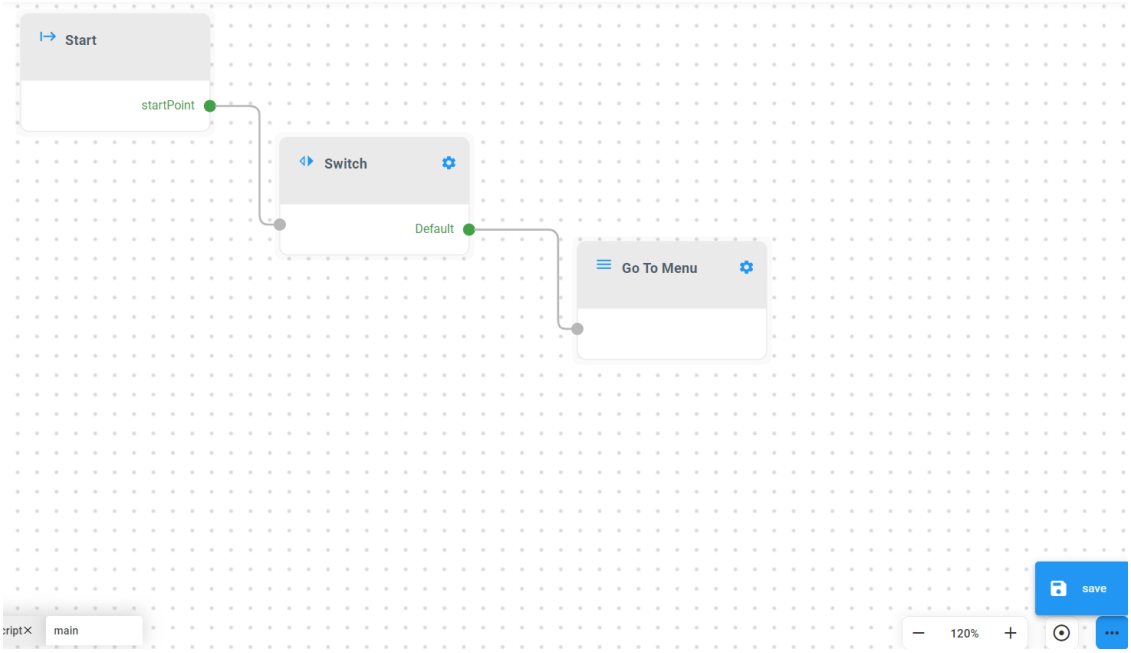
1. Click the **Switch** option under the **Call-Flow Logic** group; the following Switch building block appears:



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 'Switch Expression' field, enter the expression to be evaluated. The returned value is compared against the list of compared values. If a match is found the node exits out the corresponding leg. If no match is found, the node exits out the default leg. Compared value can be '0'.
5. In the 'Compared Value' field, enter the compared value.
6. In the 'Next Node Name' field, enter the next node name.
7. In the 'Exit Name', enter the name of the output leg to be created. The name should be written without quotes. AlphaNumeric, Underscore (_), hyphen (-), space, must start with Alpha. The maximum length is 24 characters.
8. Click **OK**.

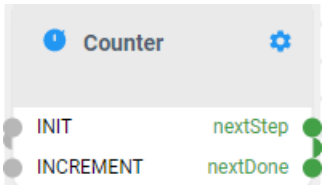


Counter

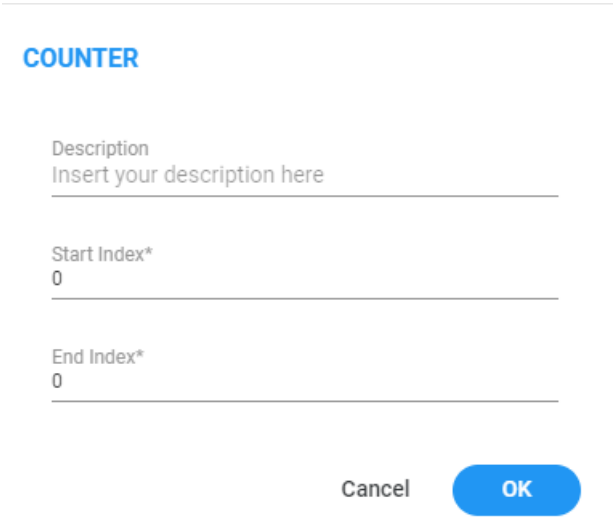
This section describes how to use the Counter building block.

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

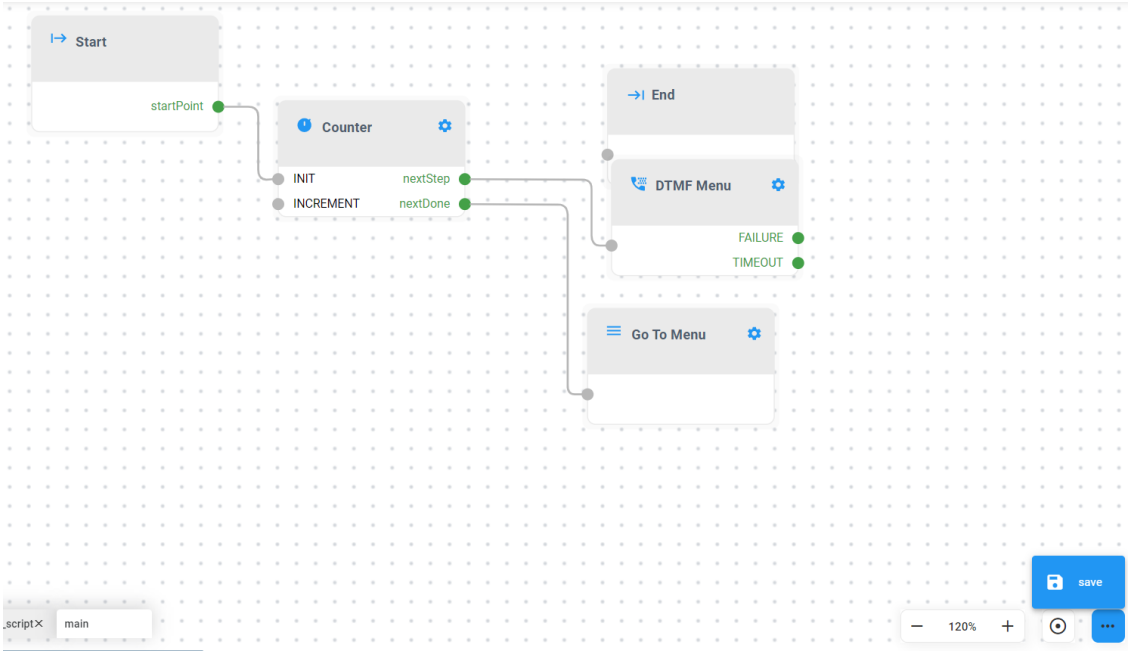
1. Click the **Counter** option under the **Call-Flow Logic** group; the following Counter building block appears:



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 'Start Index' field, enter the counter start index.
5. In the 'End Index' field, enter the Counter end index.
6. Click **OK**.

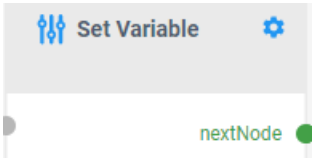


Set Variable

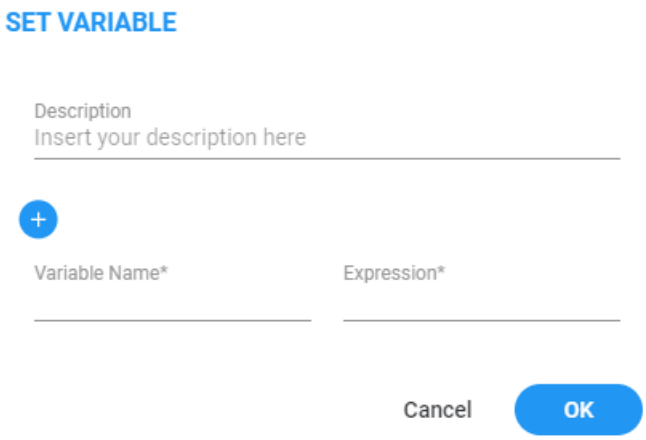
This section describes how to use the Set Variable building block.

➢ **To use the Set Variable building block:**

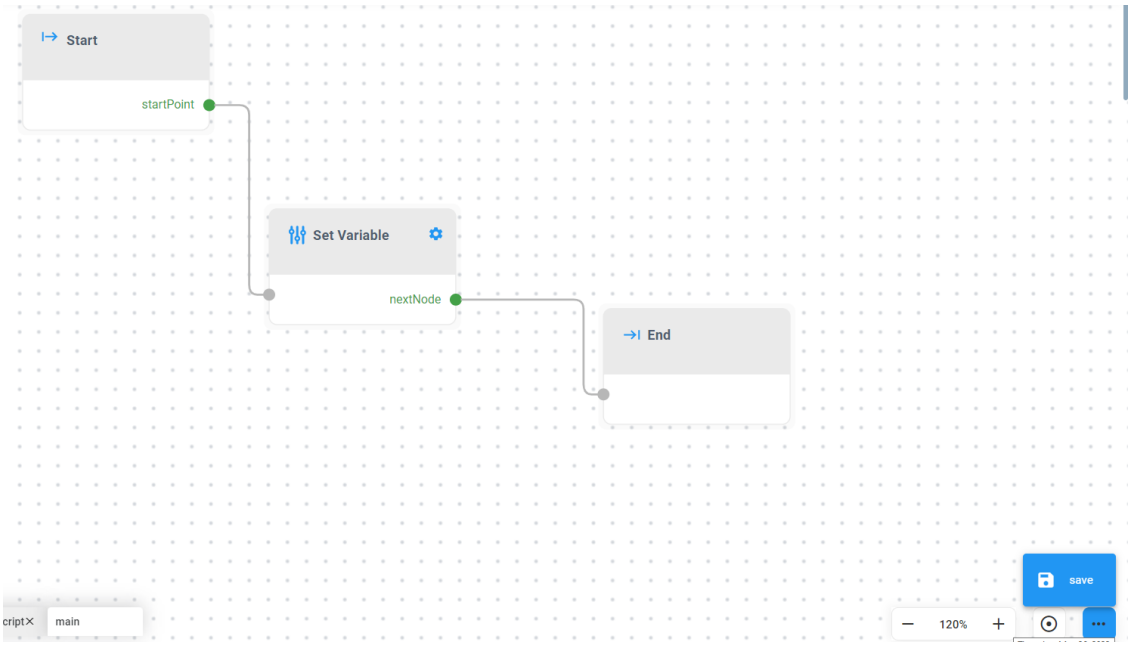
1. Click the **Set Variable** option under the **Call-Flow Logic** group; the following Set Variable building block appears:



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 'Variable Name' field, enter the name of the variable.
5. In the 'Expression' field, enter the expression to be evaluated.
6. Click **OK**.

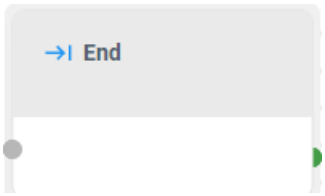


End

This section describes how to use the End building block.

➢ To use the End building block:

1. Click the **End** option under the **Call-Flow-Logic** group; the following End building block appears:



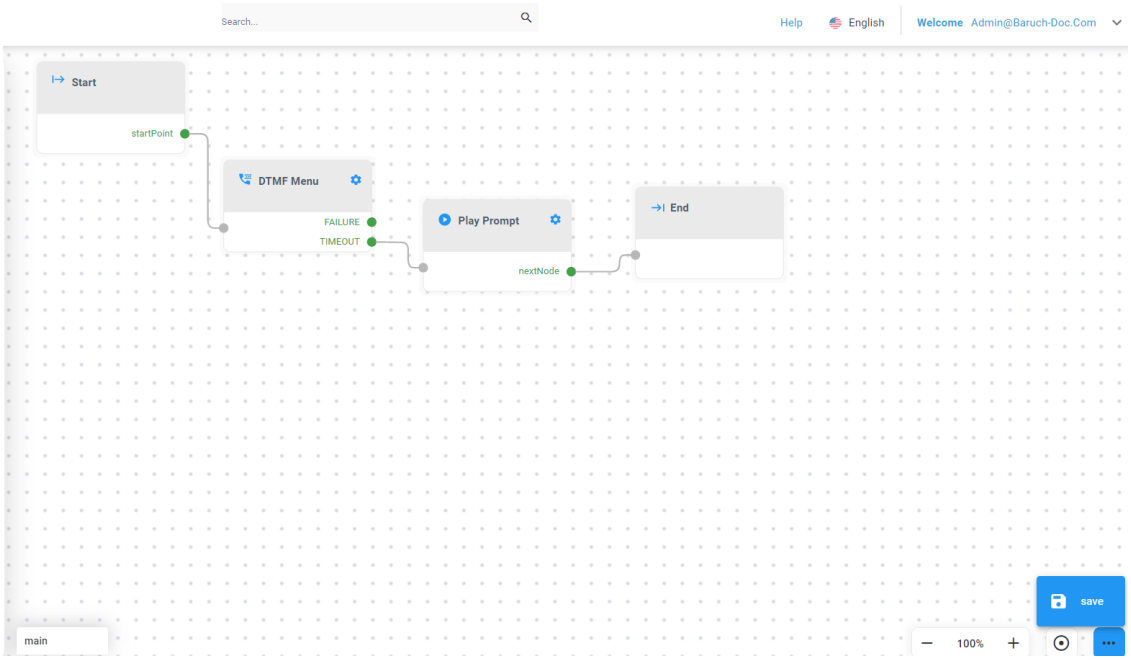
This is an End node for script ending.

7 Save

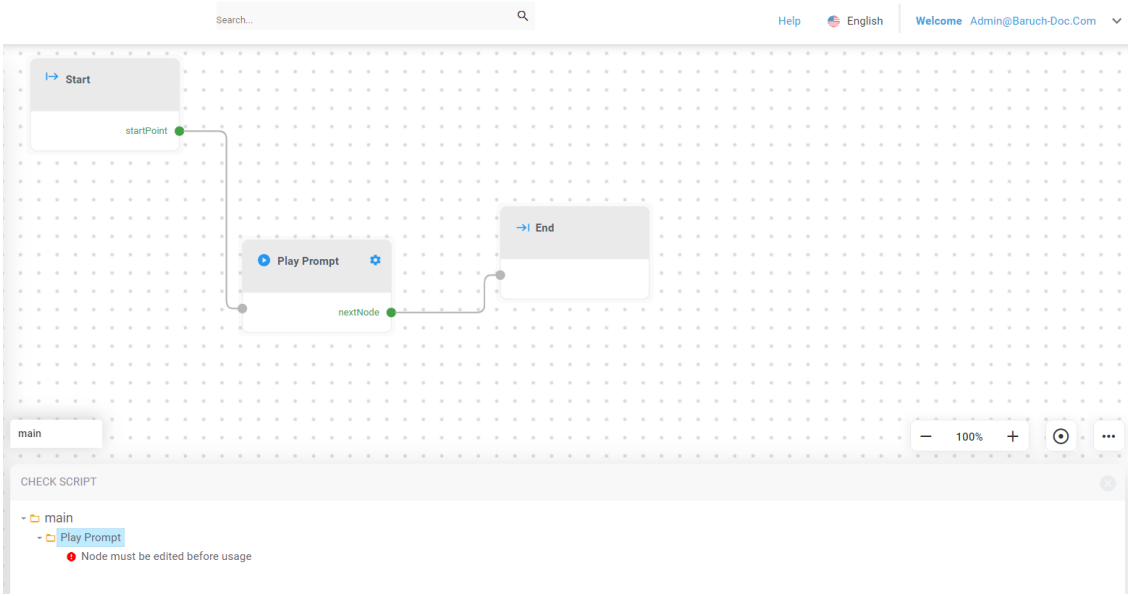
The procedure below describes how to save a script.

➢ To save a script:

1. On the lower-right part of the main flow builder 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.

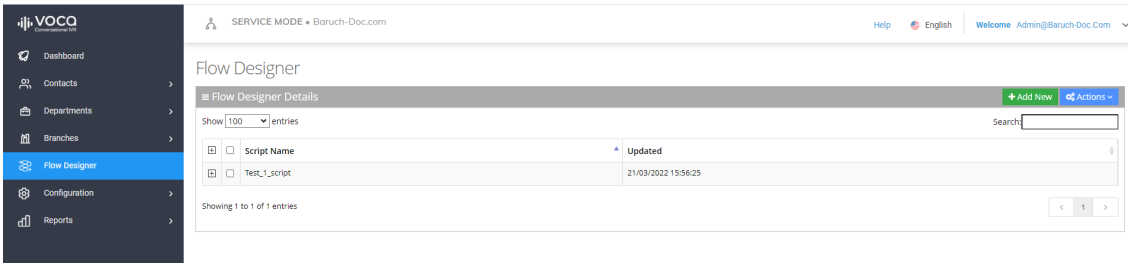


8 Search

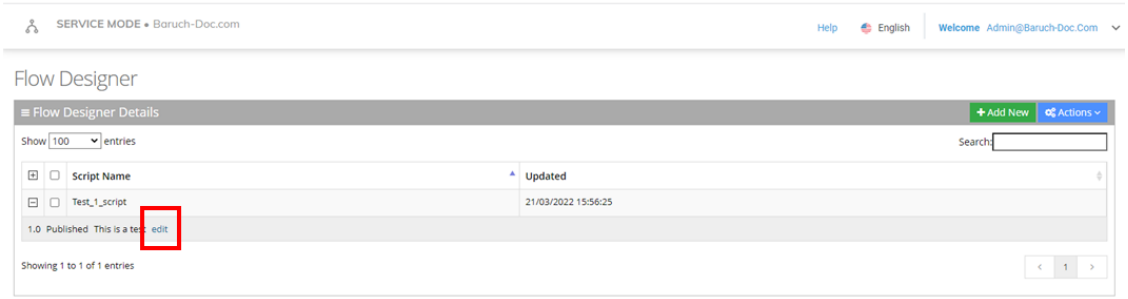
The procedure below describes how to use the Search engine.

➢ To use the Search engine:

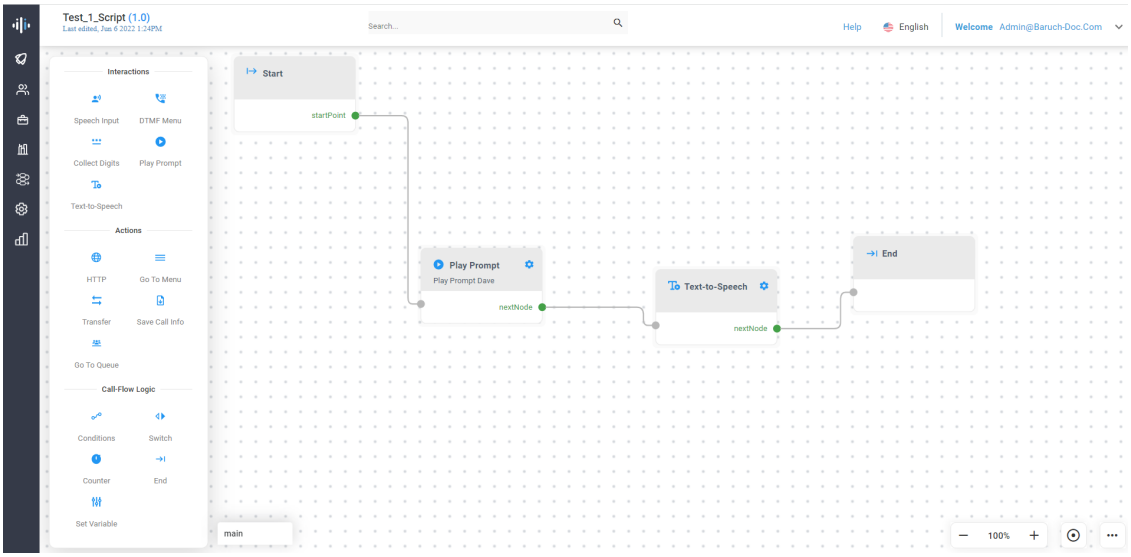
1. From the Navigation pane, click **Flow Designer**; the Flow Designer page opens:



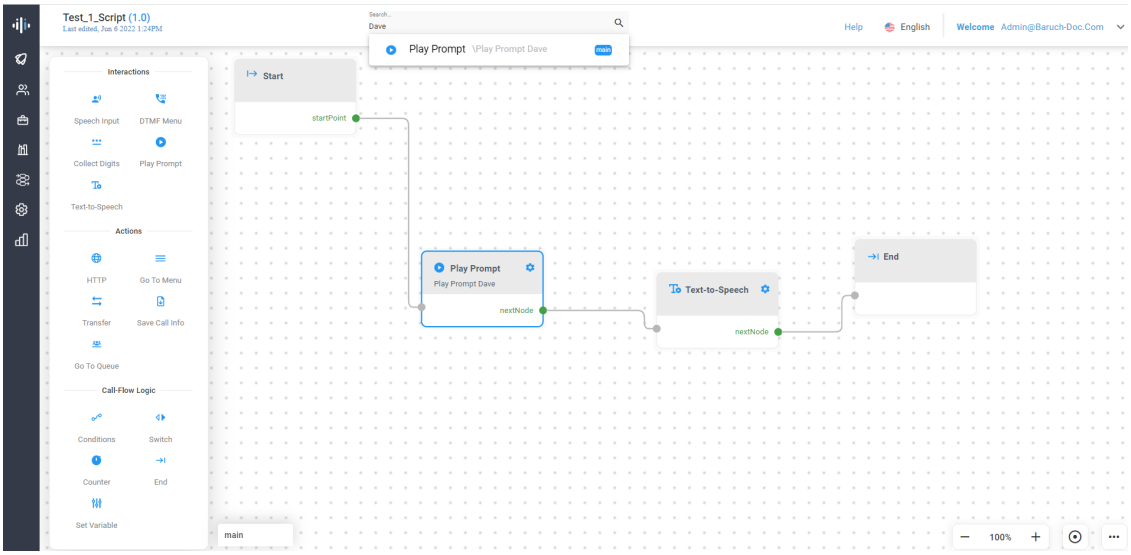
2. Select the script you wish to edit, by clicking the corresponding plus box; the **edit** link appears under the selected script:



3. Click **edit**; the main flow builder 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

1 Hayarden Street,

Airport City

Lod 7019900, Israel

Tel: +972-3-976-4000

Fax: +972-3-976-4040

AudioCodes Inc.

200 Cottontail Lane

Suite A101E

Somerset NJ 08873

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-12990