4_11_Synthesizer

Table of Contents

1. Document Versioning

Version	Date	Purpose	Author
4.11.1-SM	09-Feb-2021	First Draft	Saurabh Goyal

2. Introduction

Ameyo Synthesizer is an easy-to-use and Web-based tool for creating the Nodeflows. It allows the users to create or copy the Child Nodeflows, export the Nodeflow as a ".ANFX" file, and open the existing Nodeflows.

2.1 Licensable Feature

Synthesizer is a licensable feature that can be accessed from the Administrator Console after the login. If it's license is not obtained, "Synthesizer" button is not displayed.

2.2 Backward Compatibility

Synthesizer has succeeded the Nodeflow Designer. As of now, the Nodeflows created in Nodeflow Designer cannot be migrated to Synthesizer. Ameyo Server allows the usage of the already created Nodeflows of Nodeflow Designer (mostly they had ".Nodeflow" extension). For future, you can create the Nodeflows in Ameyo Synthesizer.

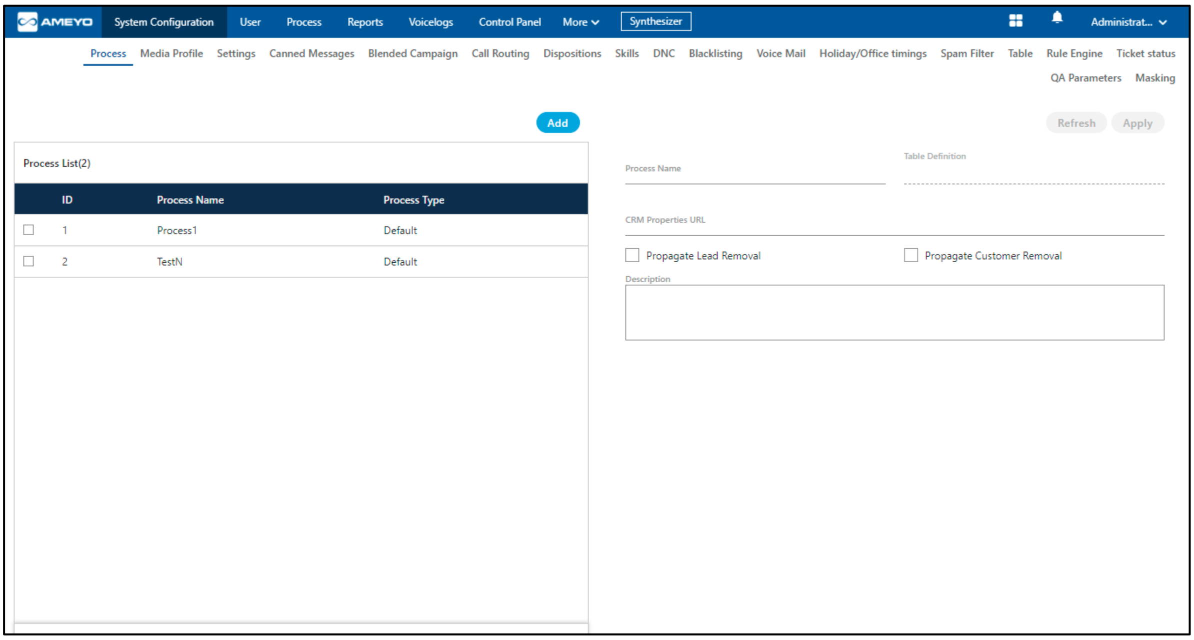
2.3 Nodeflow

The Nodeflows can be designed and customized to map the business process for both Outbound or Inbound Calls. The Nodeflow concept provides the users with boundless capabilities for mapping their business process with the contact center calls.

3. Getting Started

Ameyo Synthesizer comes preinstalled with the Ameyo Server. No separate installation or configuration is required. However, you need a separate license to use it.

After logon at the Administrator Console, Synthesizer button is displayed on the main navigation panel only when it's license has been obtained. If the license is not available, this button will not be visible.



Process List(2)

	ID	Process Name	Process Type
□	1	Process1	Default
□	2	TestN	Default

Process Name

Table Definition

CRM Properties URL

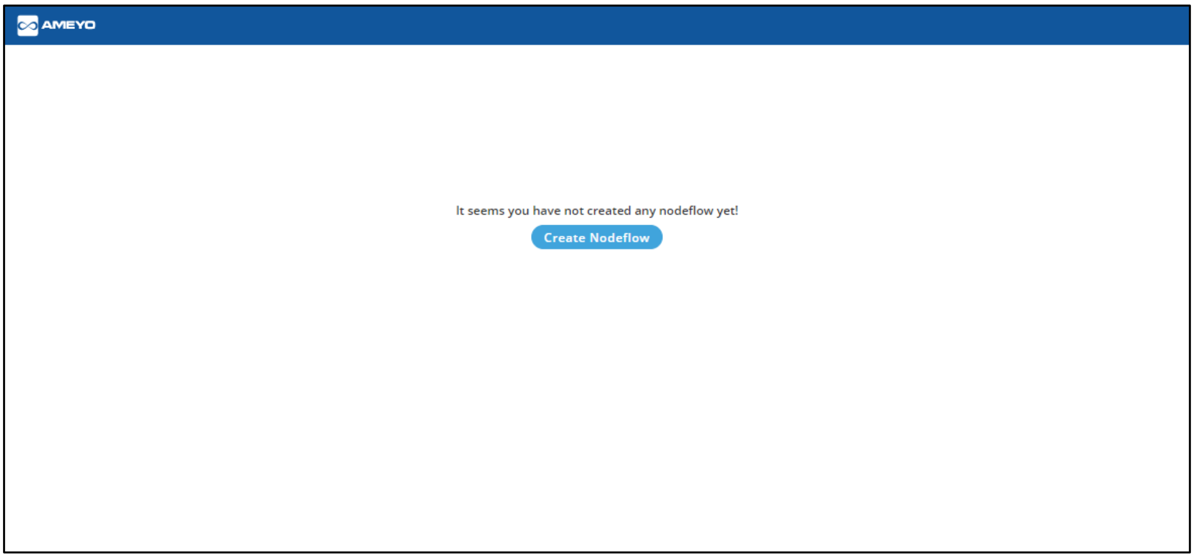
□ Propagate Lead Removal

□ Propagate Customer Removal

Description

Figure: Administrator Console

Click "Synthesizer" link to open its interface in a new tab.



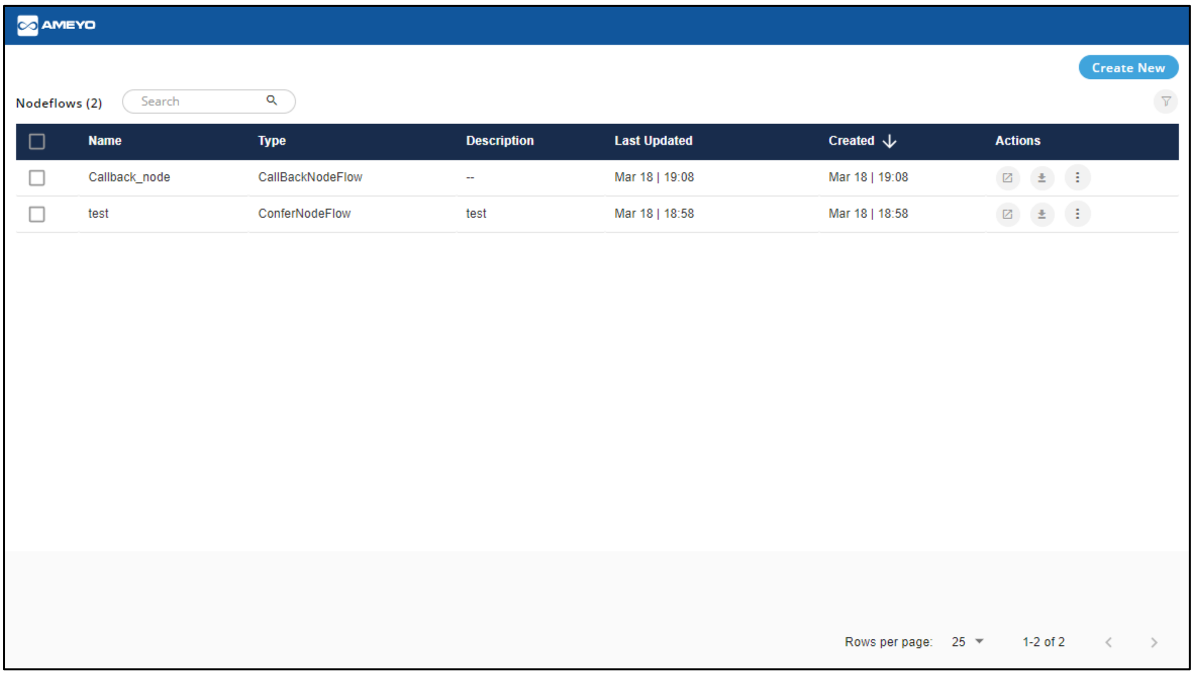
It seems you have not created any nodeflow yet!

Figure: First Page

Synthesizer does not need any separate login. It works with the session ID obtained from the Administrator logon session.

The User can click "Create New" button to create a new nodeflow. It redirects the user to "Create New Node" Page. Know more...

If the nodeflows exist in Synthesizer, then this first page will show their list and options to manage them.



Nodeflows (2)

□	Name	Type	Description	Last Updated	Created	Actions
□	Callback_node	CallBackNodeFlow	--	Mar 18 | 19:08	Mar 18 | 19:08	
□	test	ConferNodeFlow	test	Mar 18 | 18:58	Mar 18 | 18:58	

Rows per page: 25 1-2 of 2

Figure: First Page of Synthesizer

3.1 Operations

The User can perform the following operations on this page.

- Create Nodeflow
- Open Nodeflow
- Download the Nodeflow
- Options:
    - Create a copy of the Nodeflow
    - Rename the Nodeflow
    - Delete the Nodeflows
- Filter the List of Nodeflows
- Page Navigation

3.1.1 Create Nodeflow

Click "Create New" to create the new nodeflow. It redirects the user to the following page.

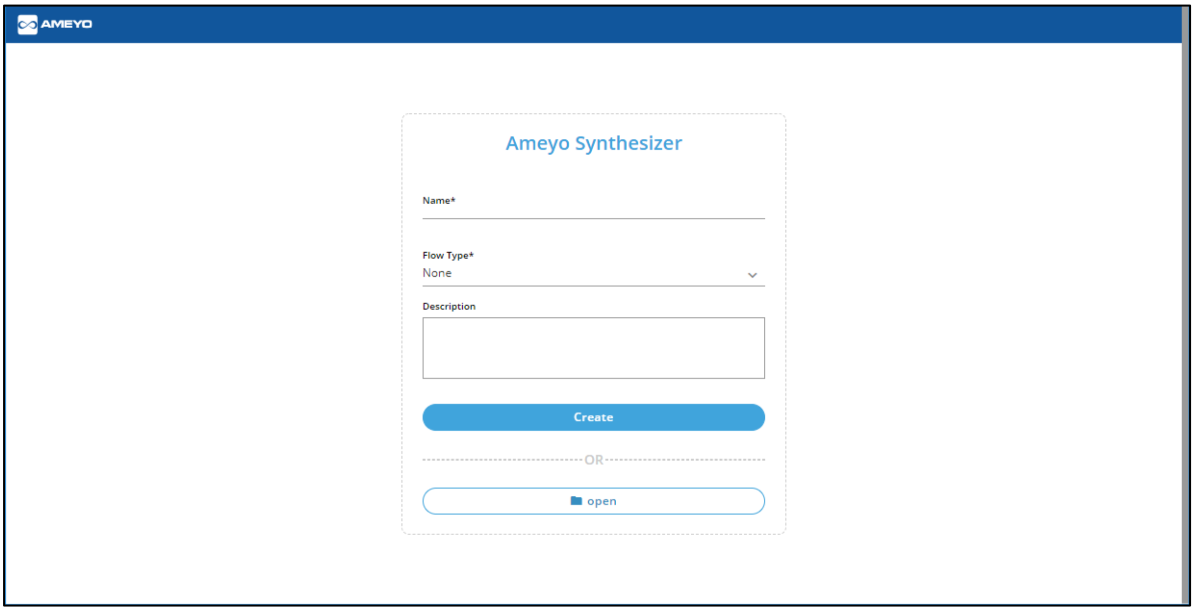


Figure: Synthesizer Page

Perform the following steps to create the Nodeflow.

1. Provide a name of the Nodeflow.
2. Select any of the following Nodeflow types.
    - **CallBackNodeFlow:** Select it to create a call back Nodeflow.
    - **ConferNodeFlow:** Select it to create a conference Nodeflow.
    - **CustomerBasedInboundNodeFlow:** Select it to create an Inbound Nodeflow based upon the customers.
    - **DialingNodeFlow:** Select it to create a dialing Nodeflow.
    - **DispositionNodeFlow:** Select it to create a disposition Nodeflow.
    - **InboundNodeFlow:** Select it to create an inbound Nodeflow.
    - **InteractionNodeFlow:** Select it to create an interaction Nodeflow.
    - **ManualDialNodeFlow:** Select it to create a manual dial Nodeflow.

- **PostCallProcessingNodeFlow:** Select it to create a Nodeflow that works after the call.
- **PostChatProcessingNodeFlow:** Select it to create a Nodeflow that works after the chat.

The definitions of these Nodeflow types is mentioned in "Nodeflow Types" section.

3. Describe the Nodeflow.
4. Click "Create". It shows the main area of Synthesizer where you can create the Nodeflows.

When the user saves the nodeflow, it will be displayed in the Nodeflow Management that is First Page of Synthesizer.

3.1.2 Open Nodeflow

For a nodeflow, the user can click ↗ icon to open the nodeflow in a new tab of the same window of the Browser.

3.1.3 Export Nodeflow

Click ⤓ icon to export the nodeflow. Know more...

3.1.4 Options

Click ⋮ to access its menu.

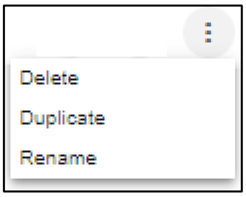


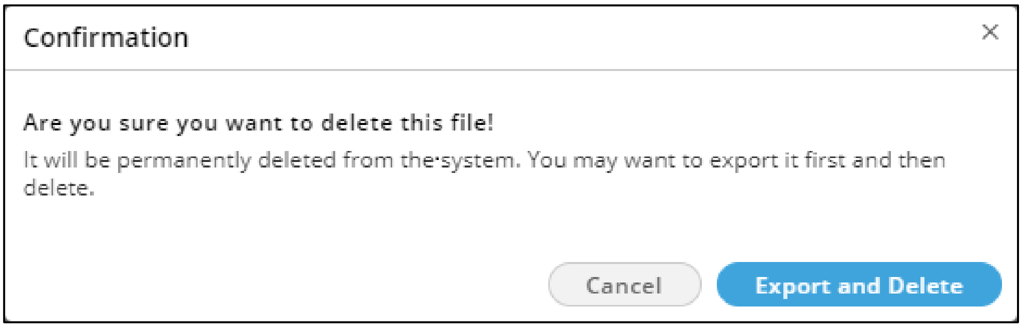
Figure: Options

It contains the following options.

- **Delete:** Perform the following steps to delete a nodeflow.

  If the nodeflow selected for deletion is not exported to the file system, then it is not possible to restore the deleted nodeflow.

  1. Click ⋮ to access its menu.
  2. Click "Delete" option. It shows the following modal.


Confirmation

Are you sure you want to delete this file!

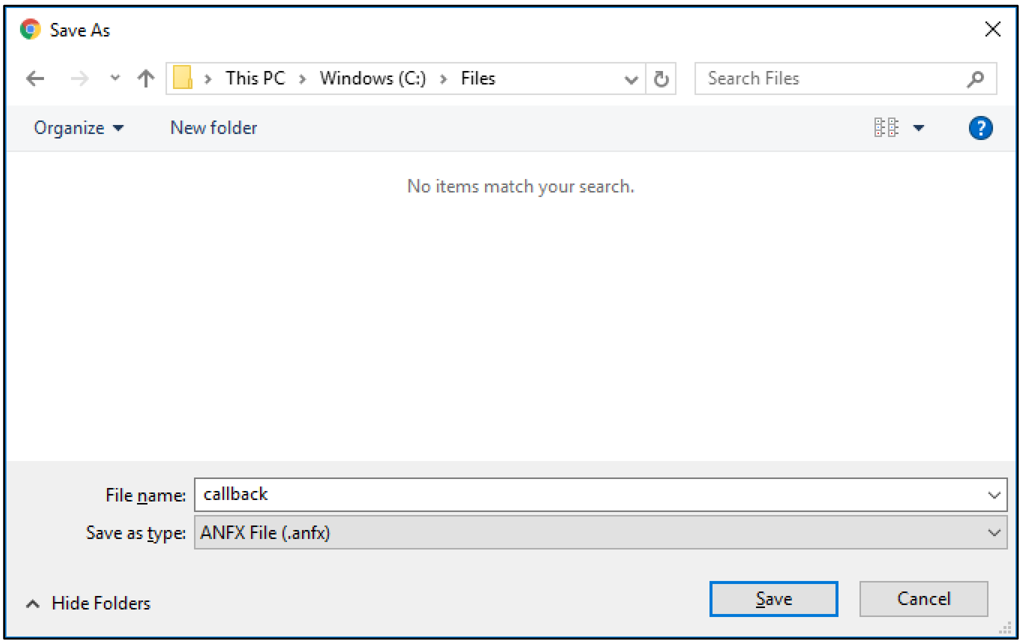
It will be permanently deleted from the system. You may want to export it first and then delete.

Figure: Warning Message

  3. Click "Export and Delete" to export the nodeflow to the disk and delete it from the Synthesizer.
  4. "Export" option is based upon "Save As" functionality of the Web browser. If "Ask where to save each file before downloading" option is not checked, the Nodeflow will be exported as ".anfx" file at the default download location of the Web browser, which will have the name provided while creating the Nodeflow. In this case, the Administrator does not get the option to change the name and path of the file.

It is recommended to keep "Ask where to save each file before downloading" or similar option checked in your Web Browser so that you can specify the name and path to save the files.

If "Ask where to save each file before downloading" or similar option is enabled in your Web Browser, then the following dialog box is displayed while exporting the Nodeflow.



File name: callback

Save as type: ANFX File (.anfx)

Figure: Export File

Select the location and provide a name for the file. Click "Save" to save the file.

5. As soon as either "Save As" dialog box is displayed or the nodeflow is downloaded automatically, the selected nodeflow is deleted from the system.

- **Duplicate:** Perform the following steps to create a duplicate copy of a nodeflow.

1. Click ⋮ to access its menu.

2. Click "Duplicate" option. A copy of the nodeflow is created with the name <nodeflow_name.1> and the nodeflow is opened in a new tab of the same Browser Window.

   If the user is creating the duplicate of an already duplicated nodeflow, then its name will be <nodeflow_name.1.1>. Another ".1" will be suffixed if another duplicate copy is created.

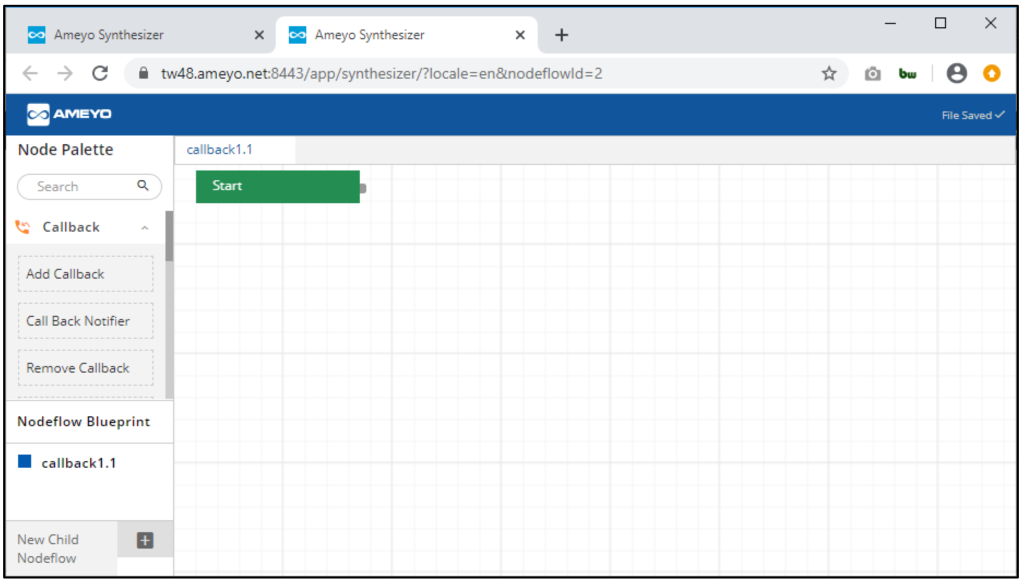
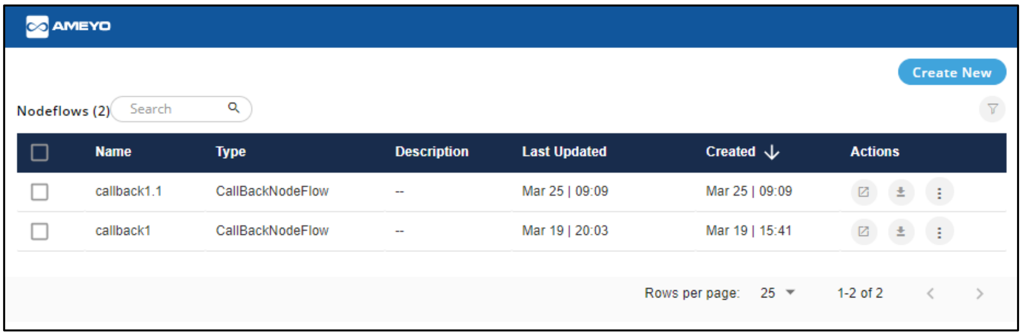


Figure: Duplicate Nodeflow in the new Tab

With the creation, the duplicate nodeflow is saved automatically and a message is displayed in the top right corner of the window.

3. Switch to the first page of Synthesizer, which shows the duplicate nodeflow.



Nodeflows (2)

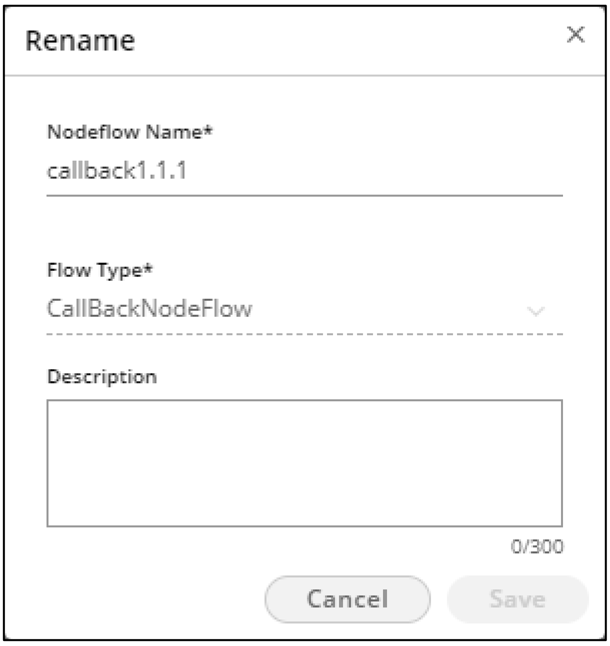
□	Name	Type	Description	Last Updated	Created	Actions
□	callback1.1	CallBackNodeFlow	--	Mar 25 | 09:09	Mar 25 | 09:09	
□	callback1	CallBackNodeFlow	--	Mar 19 | 20:03	Mar 19 | 15:41	

Rows per page: 25 1-2 of 2

Figure: Duplicate Nodeflow in the area

- **Rename**: Perform the following steps to rename a nodeflow.

  1. Click ⋮ to access its menu.
  2. Click "Rename" option. It shows the following modal.



Rename

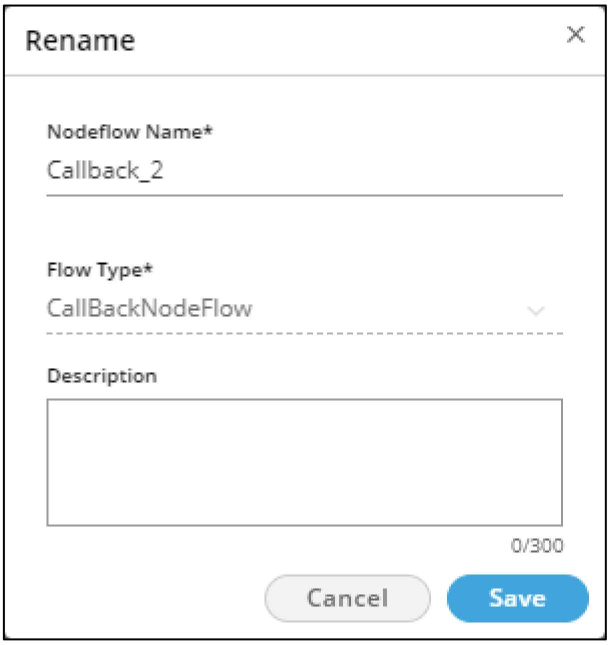
Nodeflow Name*
callback1.1.1

Flow Type*
CallBackNodeFlow

Description

Figure: "Rename Nodeflow" Modal

  3. Enter the new name of the nodeflow.



Rename

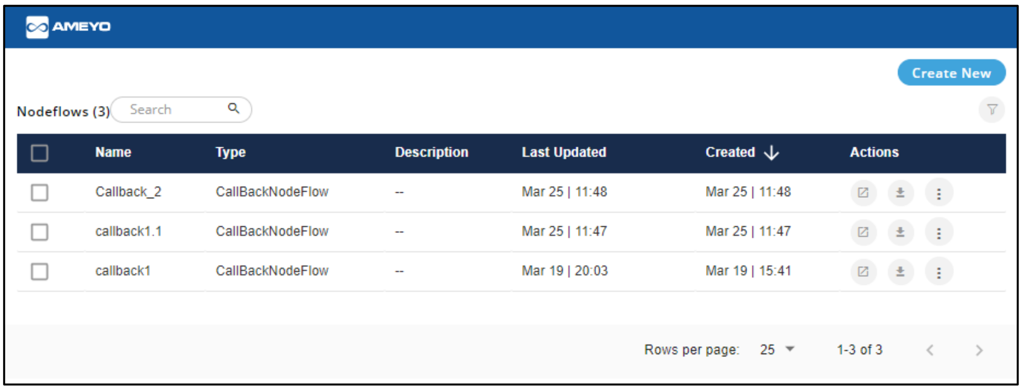
Nodeflow Name*
Callback_2

Flow Type*
CallBackNodeFlow

Description

Figure: Renaming the Nodeflow

4. Click "Save" button. The selected nodeflow is renamed.



Nodeflows (3) Search

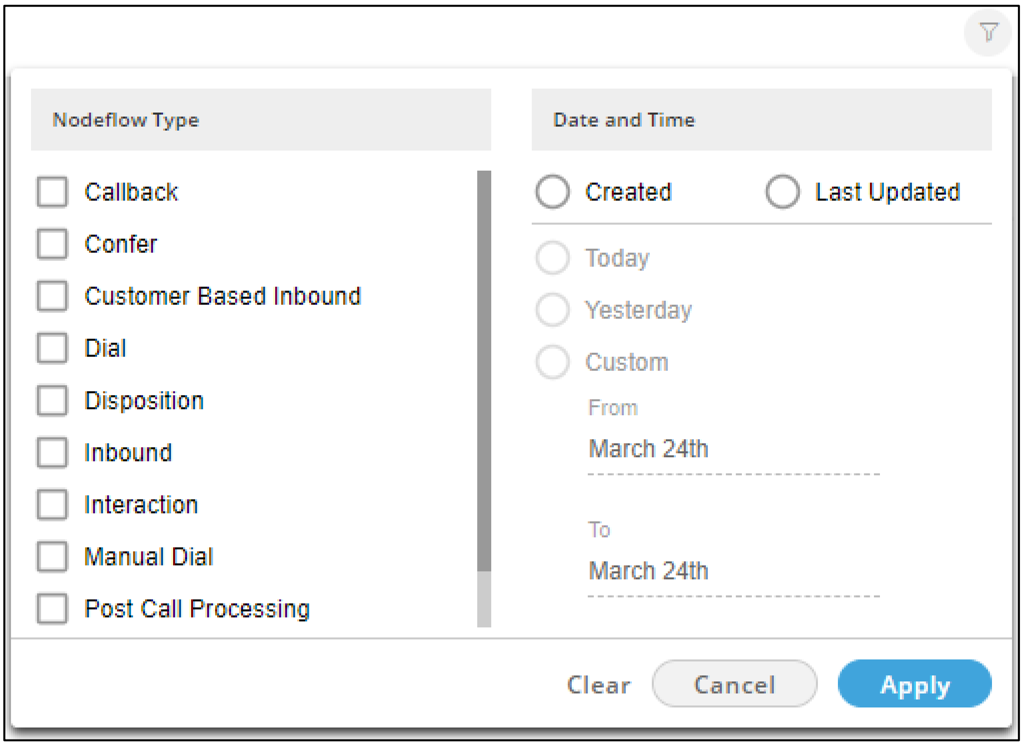
□	Name	Type	Description	Last Updated	Created	Actions
□	Callback_2	CallBackNodeFlow	--	Mar 25 | 11:48	Mar 25 | 11:48	
□	callback1.1	CallBackNodeFlow	--	Mar 25 | 11:47	Mar 25 | 11:47	
□	callback1	CallBackNodeFlow	--	Mar 19 | 20:03	Mar 19 | 15:41	

Rows per page: 25 1-3 of 3

Figure: Renamed the Nodeflow

3.1.5 Filter the Nodeflows

Click ▽ icon to filter the list of nodeflows. It shows the following modal.



Nodeflow Type

- □ Callback
- □ Confer
- □ Customer Based Inbound
- □ Dial
- □ Disposition
- □ Inbound
- □ Interaction
- □ Manual Dial
- □ Post Call Processing

Date and Time

○ Created ○ Last Updated

○ Today
○ Yesterday
○ Custom

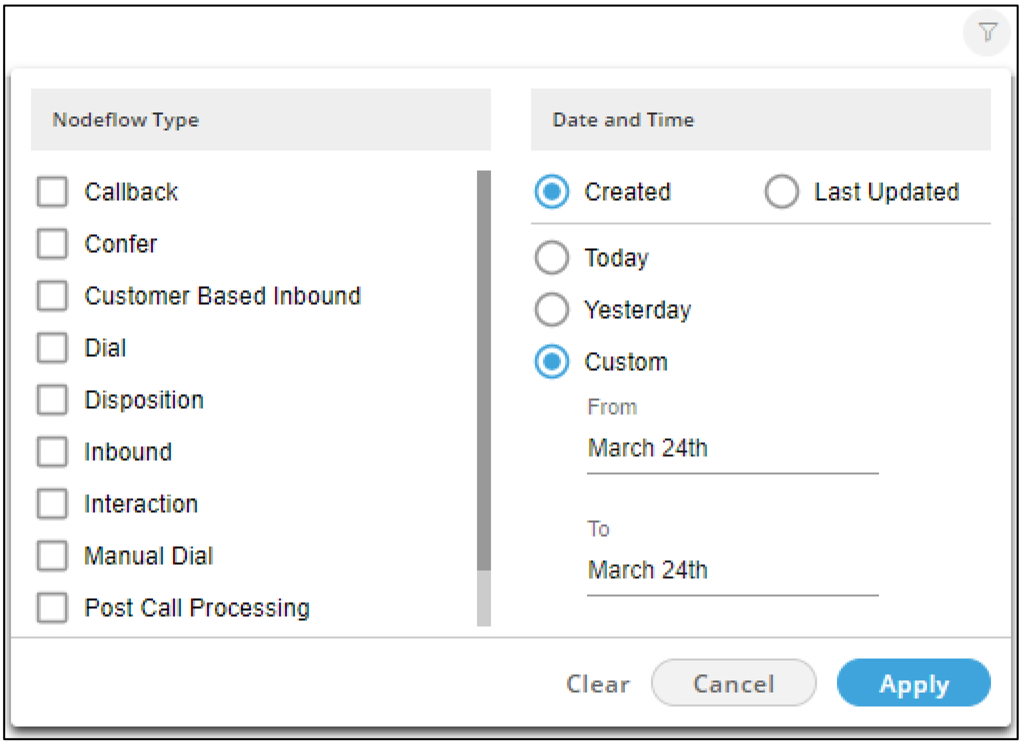
From
March 24th

To
March 24th

Figure: Filter Options

It contains the following filter options.

- **Nodeflow Types:** In this section, you can select the types of Nodeflows that has to be displayed in the list. It contains the following options.
    - Callback
    - Confer
    - Customer Based Inbound
    - Dial
    - Disposition
    - Inbound
    - Interaction
    - Manual Dial
    - Post Call Processing
    - Post Chat Processing
- **Date and Time:** It allows to filter the nodeflows based upon the created or last updated date and time. It contains the following options.
    - **Created:** Select it to filter the nodeflows based upon the created date and time.



Nodeflow Type

□ Callback
□ Confer
□ Customer Based Inbound
□ Dial
□ Disposition
□ Inbound
□ Interaction
□ Manual Dial
□ Post Call Processing

Date and Time

● Created ○ Last Updated

○ Today
○ Yesterday
● Custom

From
March 24th

To
March 24th

Figure: Enabled "Created" Date Filter

Select it to enable its following options.

- Today: Select it to view the nodeflows that are created today.
- Yesterday: Select it to view the nodeflows that are created yesterday.
- Custom: Select it to specify the date range within which the nodeflows are created. Click "From" to select the date from the calendar.

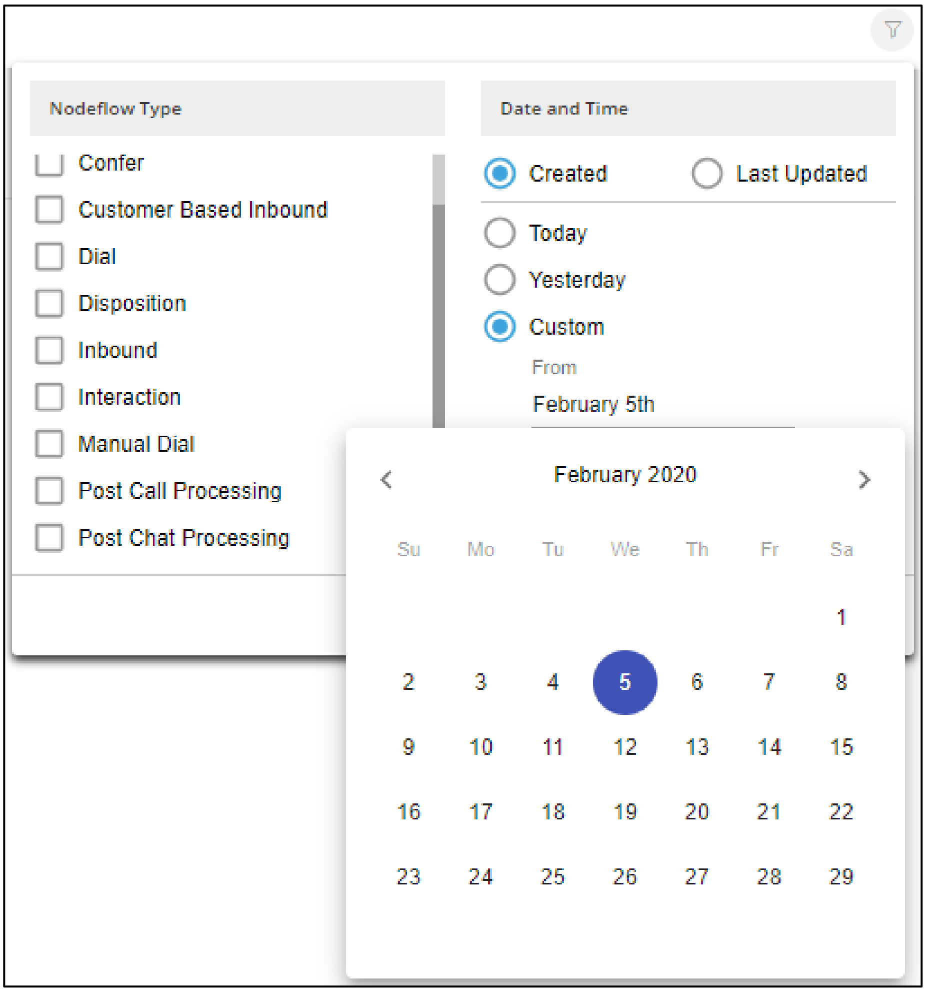
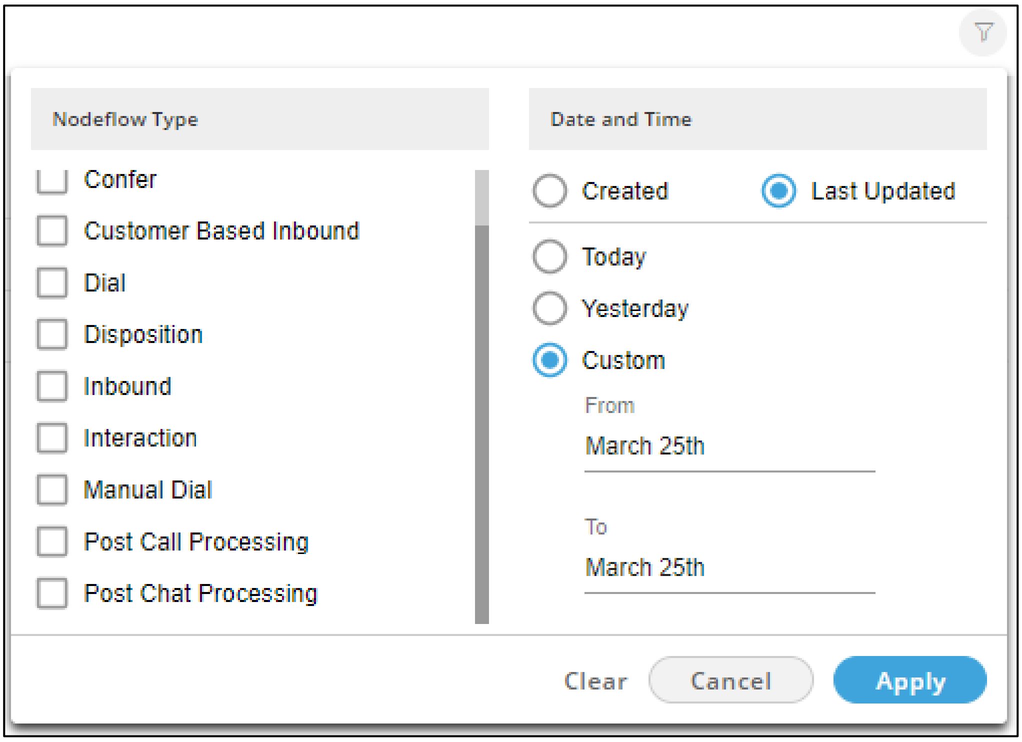


Figure: Calendar to select "From" date

Similarly, select "To" date using the calendar.

"From" date cannot be greater than "To" date.

- **Last Updated:** Select it to filter the nodeflows based upon the last date and time.



Nodeflow Type

□ Confer
□ Customer Based Inbound
□ Dial
□ Disposition
□ Inbound
□ Interaction
□ Manual Dial
□ Post Call Processing
□ Post Chat Processing

Date and Time

○ Created ● Last Updated

○ Today
○ Yesterday
● Custom

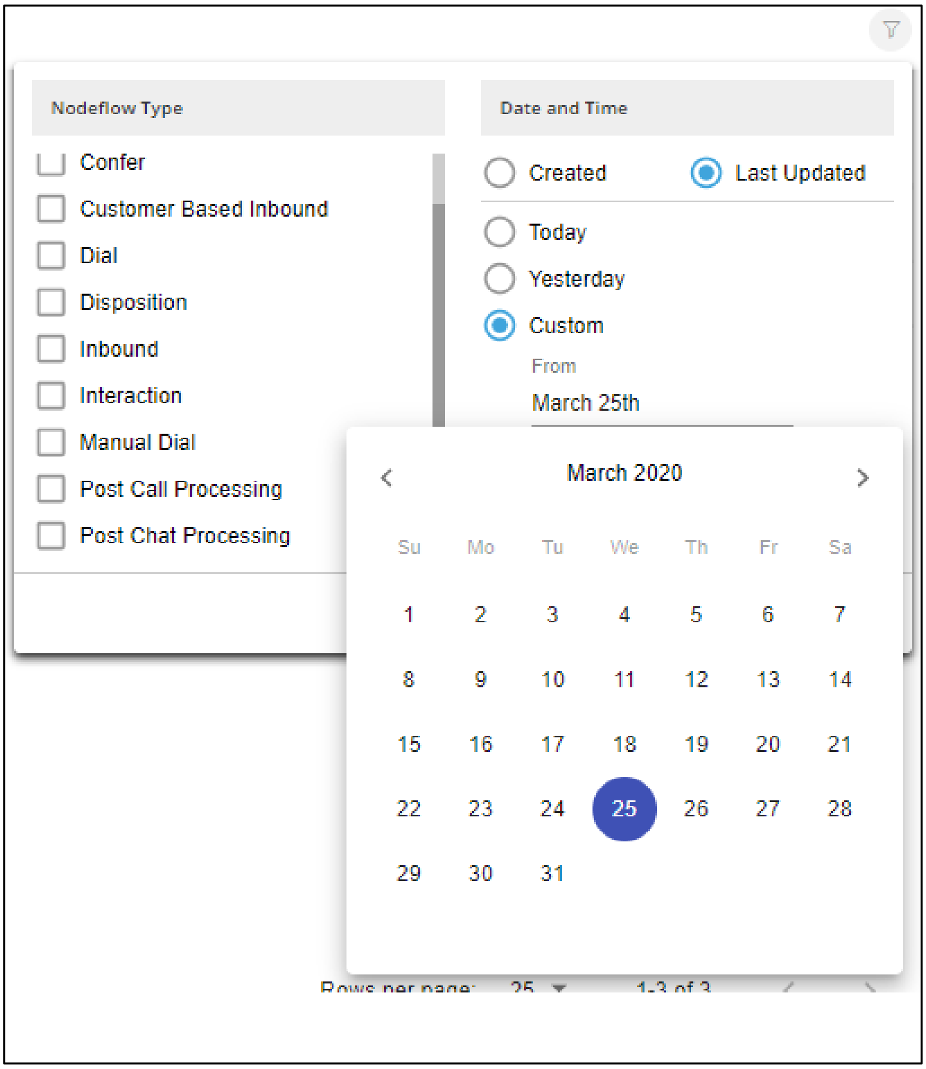
From
March 25th

To
March 25th

Figure: Enabled "Created" Date Filter

Select it to enable its following options.

- Today: Select it to view the nodeflows that are last updated today.
- Yesterday: Select it to view the nodeflows that are last updated yesterday.
- Custom: Select it to specify the date range within which the nodeflows are last updated. Click "From" to select the date from the calendar.



Nodeflow Type

- □ Confer
- □ Customer Based Inbound
- □ Dial
- □ Disposition
- □ Inbound
- □ Interaction
- □ Manual Dial
- □ Post Call Processing
- □ Post Chat Processing

Date and Time

○ Created ● Last Updated

○ Today

○ Yesterday

● Custom

From

March 25th

March 2020

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Rows per page: 25 1-3 of 3

Figure: Calendar to select "From" date

Similarly, select "To" date using the calendar.

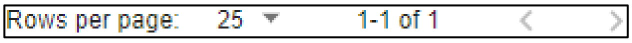
"From" date cannot be greater than "To" date.

After selecting a filter, click "Apply" to apply the filter. Rather, cick "Cancel" to not apply it.

To remove the filter, cick ▽ icon and click "Clear" link.

3.2 Page Navigation

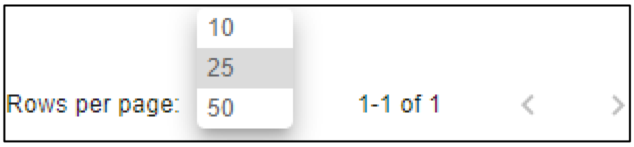
Following navigation options are available at the bottom of the page.



Rows per page: 25 1-1 of 1

Figure: Page Navigation Options

The User can click > to navigate to the next page, whereas click < to navigate to the previous page. Click the drop-down menu to access its options, which allows to define the number of nodeflows on a page.



Rows per page: 1-1 of 1

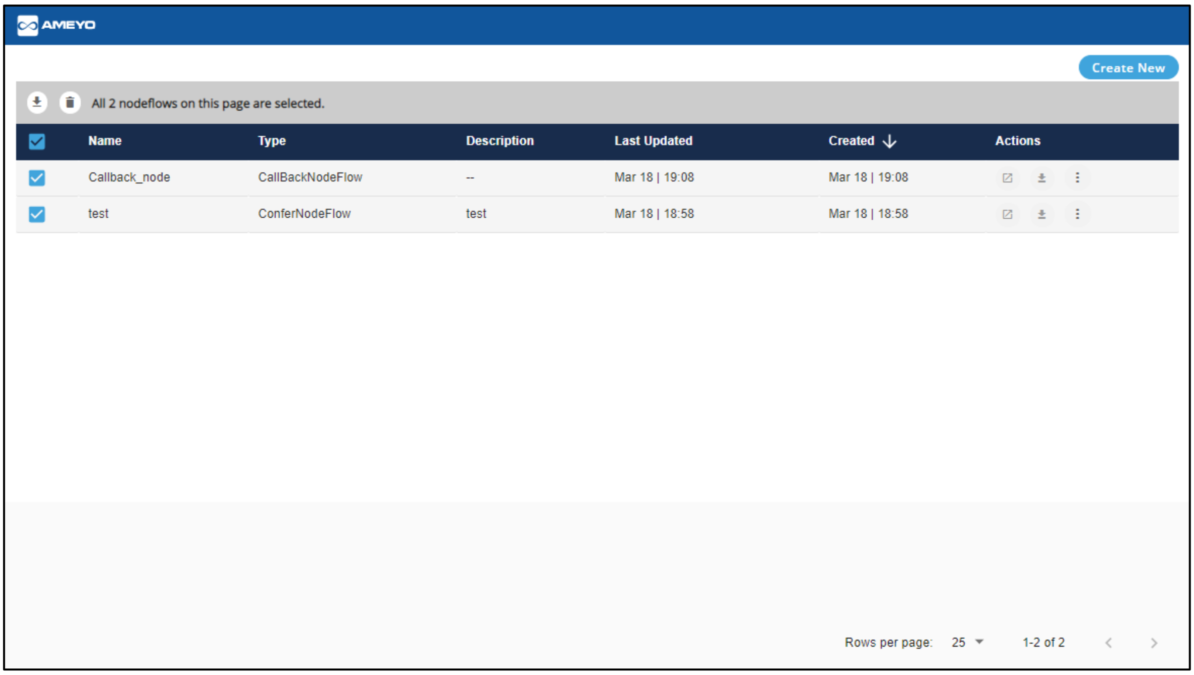
Figure: Page Navigation Options

It contains the following options.

- **10:** Select it to show 10 nodeflows in a page.
- **25:** Select it to show 25 nodeflows in a page.
- **50:** Select it to show 50 nodeflows in a page.

3.3 Bulk Operations on Nodeflows

The User can select all or multiple nodeflows in the list on a page. The User can also click checkbox in the header row to select all nodeflows in a page.



All 2 nodeflows on this page are selected.

☑	Name	Type	Description	Last Updated	Created	Actions
☑	Callback_node	CallBackNodeFlow	--	Mar 18 | 19:08	Mar 18 | 19:08	
☑	test	ConferNodeFlow	test	Mar 18 | 18:58	Mar 18 | 18:58	

Rows per page: 25 1-2 of 2

Figure: All Nodeflows are selected.

Here, the User can perform the following bulk operations.

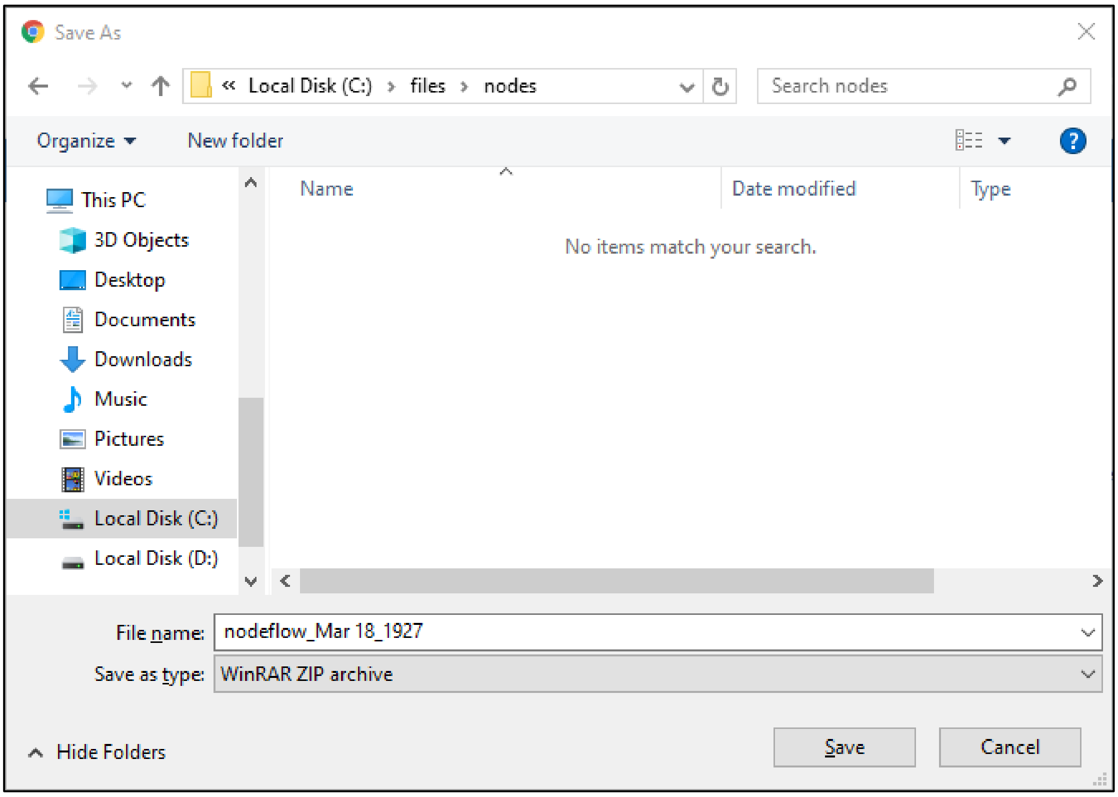
3.3.1 Download All Nodeflows

Perform the following steps to download all or multiple selected nodeflows.

1. After selecting the nodeflows, click ⤓ icon to download the nodeflows.
2. This option is based upon "Save As" functionality of the Web browser. If "Ask where to save each file before downloading" option is not checked, the Nodeflow will be downloaded as ".zip" file at the default download location of the Web browser, which will have the name provided while creating the Nodeflow. In this case, the User does not get the option to change the name and path of the file.

It is recommended to keep "Ask where to save each file before downloading" or similar option checked in your Web Browser so that you can specify the name and path to save the files.

If "Ask where to save each file before downloading" or similar option is enabled in your Web Browser, then the following dialog box is displayed while exporting the Nodeflow.



No items match your search.

File name: nodeflow_Mar 18_1927

Save as type: WinRAR ZIP archive

Figure: Save zip file

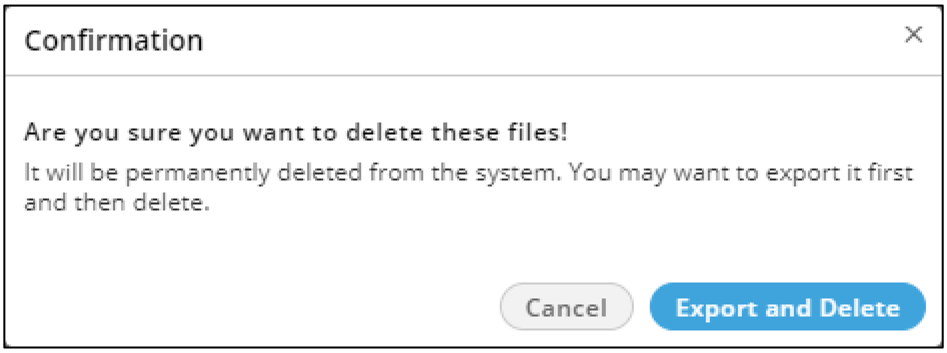
Select the location and provide a name for the file. Click "Save" to save the zip file.

3.3.2 Delete All Nodeflows

Perform the following steps to delete all or multiple selected nodeflows.

If the nodeflows selected for deletion are not exported to the file system, then it is not possible to restore the deleted nodeflows.

1. After selecting the nodeflows, click 🗑 icon to delete all the selected nodeflows. It shows the following modal.



Confirmation

Are you sure you want to delete these files!

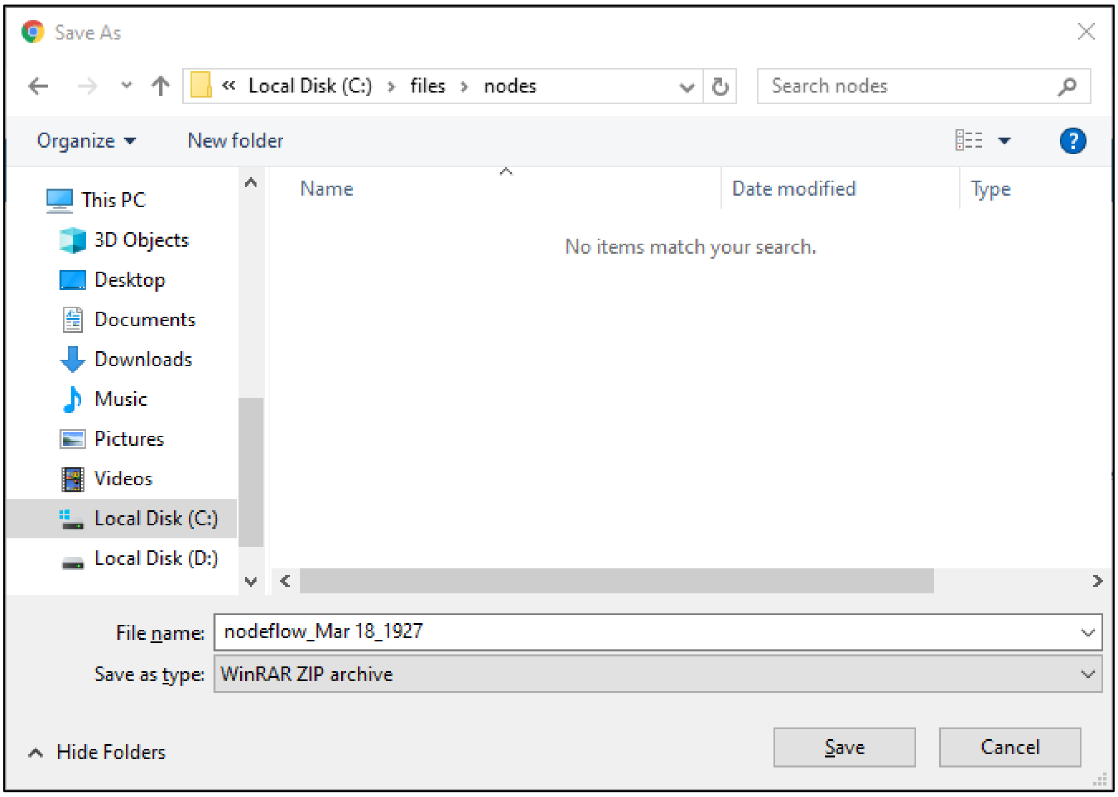
It will be permanently deleted from the system. You may want to export it first and then delete.

Figure: Warning Message

2. Click "Export and Delete" to export the nodeflows to the disk as a zip file and delete them from the Synthesizer.
3. "Export" option is based upon "Save As" functionality of the Web browser. If "Ask where to save each file before downloading" option is not checked, the Nodeflow will be exported as ".zip" file at the default download location of the Web browser, which will have the name provided while creating the Nodeflow. In this case, the Administrator does not get the option to change the name and path of the file.

   It is recommended to keep "Ask where to save each file before downloading" or similar option checked in your Web Browser so that you can specify the name and path to save the files.

   If "Ask where to save each file before downloading" or similar option is enabled in your Web Browser, then the following dialog box is displayed while exporting the Nodeflow.



File name: nodeflow_Mar 18_1927

Save as type: WinRAR ZIP archive

Figure: Export File

Select the location and provide a name for the file. Click "Save" to save the file.

4. As soon as either "Save As" dialog box is displayed or the nodeflows are downloaded automatically, the selected nodeflows are deleted from the system.

4. Nodeflow Types

Following types of Nodeflows are available in the Synthesizer.

4.1 Callback Nodeflow Type

It is used to create the Nodeflow for callbacks for voice campaigns.

4.2 Conference Nodeflow Type

It is used to create the Nodeflow for the conference for voice campaigns.

4.3 Customer-based Inbound Nodeflow Type

It is used to create the Nodeflow for customer-based Inbound Campaigns.

4.4 Dialing Nodeflow Type

It is used to create the Nodeflow for dialers to dial the calls. It works for both auto-dial and manual dial.

4.5 Disposition Nodeflow Type

It is used to create the Nodeflow for the dispositions. It is used to synchronize the disposition between the third-party CRMs and Ameyo.

4.6 Inbound Nodeflow Type

It is used to create the Nodeflow for inbound campaigns.

4.7 Manual Dial Nodeflow Type

It is used to create the Nodeflow for Manual Dialing.

4.8 Post Call Processing Nodeflow Type

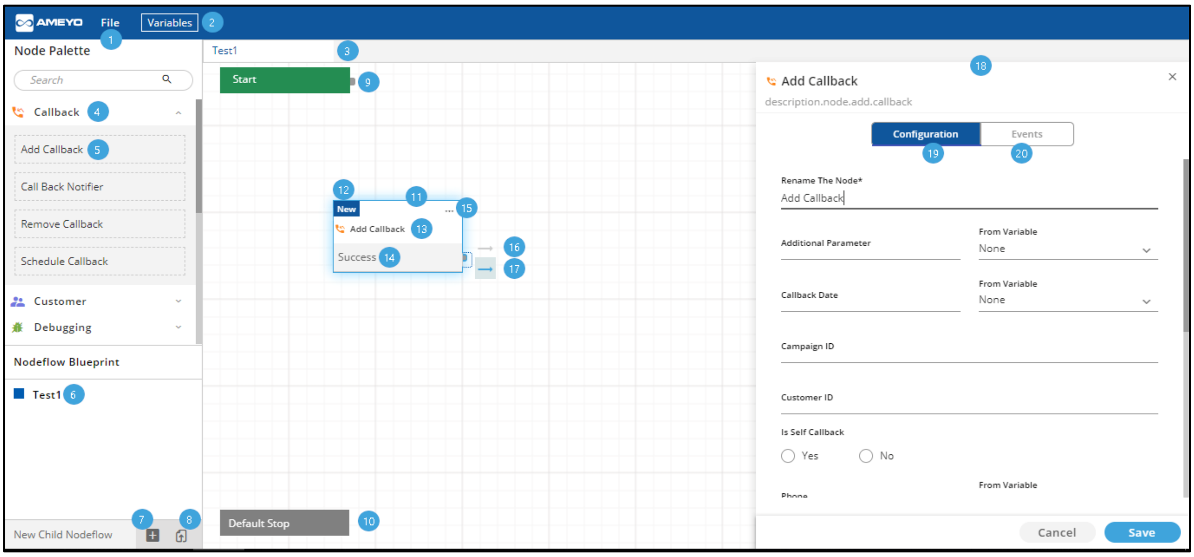
It is used to create the Nodeflow to perform processing after hang up and disposing a call.

4.9 Post Chat Processing Nodeflow Type

It is used to create the Nodeflow to perform processing after disposing a chat session.

5. Nodeflow Interface

When a new nodeflow is created or an existing nodeflow from Nodeflow Management is opened, then the following page is displayed. It is a screenshot of the Nodeflow Interface.



Add Callback
description.node.add.callback

Rename The Node*: Add Callback

Additional Parameter — From Variable: None

Callback Date — From Variable: None

Campaign ID

Customer ID

Is Self Callback: ○ Yes ○ No

Phone — From Variable

Figure: Synthesizer Interface

Following list explains the elements numbered in the above screenshot of Synthesizer Interface.

1. File Menu
2. Button to create and manage Variables
3. Tabs for a Nodeflow and its Child Nodeflows
4. Category of Nodes
5. Nodes in a category
6. List of Nodeflow and its Child Nodeflows
7. Icon to create a new Child Nodeflow

8. Icon to import an existing ".nodefflow" or ".anfx" nodeflow
9. Start Node
10. Default Stop Node
11. Node in the Canvas Area
12. Indication of a new Node in the Canvas
13. Name of the node
14. Events in the node
15. Menu to see options
16. Unconditional Transition to another node
17. Conditional Transition to another node
18. Right Pane to show the details of the selected node
19. Configuration Tab
20. Events Tab

The different elements of the user interface are explained hereinbelow.

- **Node Palette:** It contains the list of nodes for the selected Nodeflow type. These are divided into the categories. You can click any category to expand it and access its member nodes. Click it again to collapse. You can use "Search" box to search for any particular node.
- **Nodeflow Blueprint:** It contains the list of Nodeflows. Here, you can create the Child Nodeflows also.
- **Tabs:** A new tab is created for each Nodeflow. Different tabs will be there for the Child Nodeflows.

- **Canvas Area:** It is the work area. It contains "Start" and "Default Stop" nodes by default. You can drag any node from the node palette and drop it here.
- **Start Node:** It starts the execution of a Nodeflow.
- **Default Stop Node:** It is the default stop. The Event.System.Error events of all nodes and any failure occurred while processing the nodes directs the Nodeflow to this node directly.

5.1 File Menu

Click "File" on the top to access the File menu.

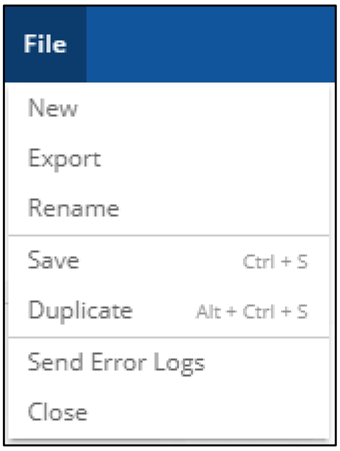


Figure: File Menu

It contains the following options.

- **New:** Click it to create a new Nodeflow. It starts a new session in a new browser tab and shows the first screen of Synthesizer. The Administrator can use this option to create multiple Nodeflows simultaneously.

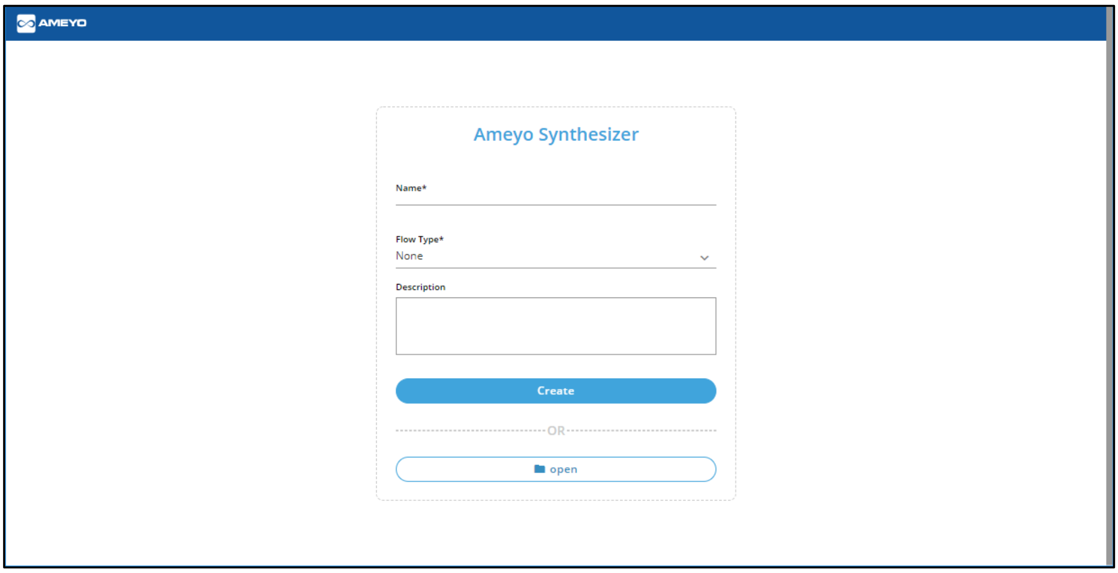
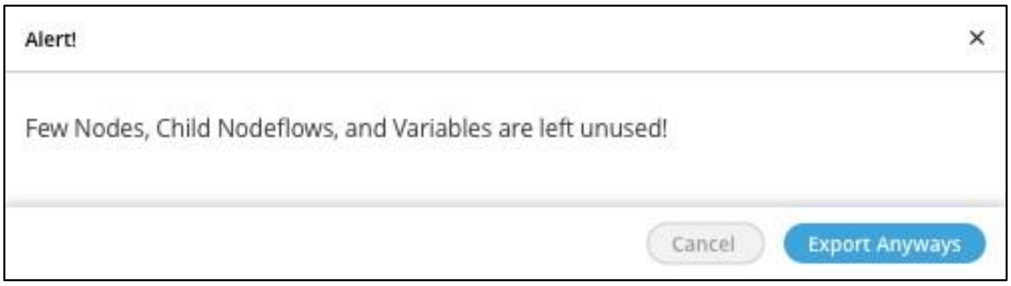


Figure: Synthesizer Page

- **Export:** Click "Export" to save the created Nodeflow to a ".anfx" file on the disk. If some nodes are unused, it shows the following warning message.



Alert!

Few Nodes, Child Nodeflows, and Variables are left unused!

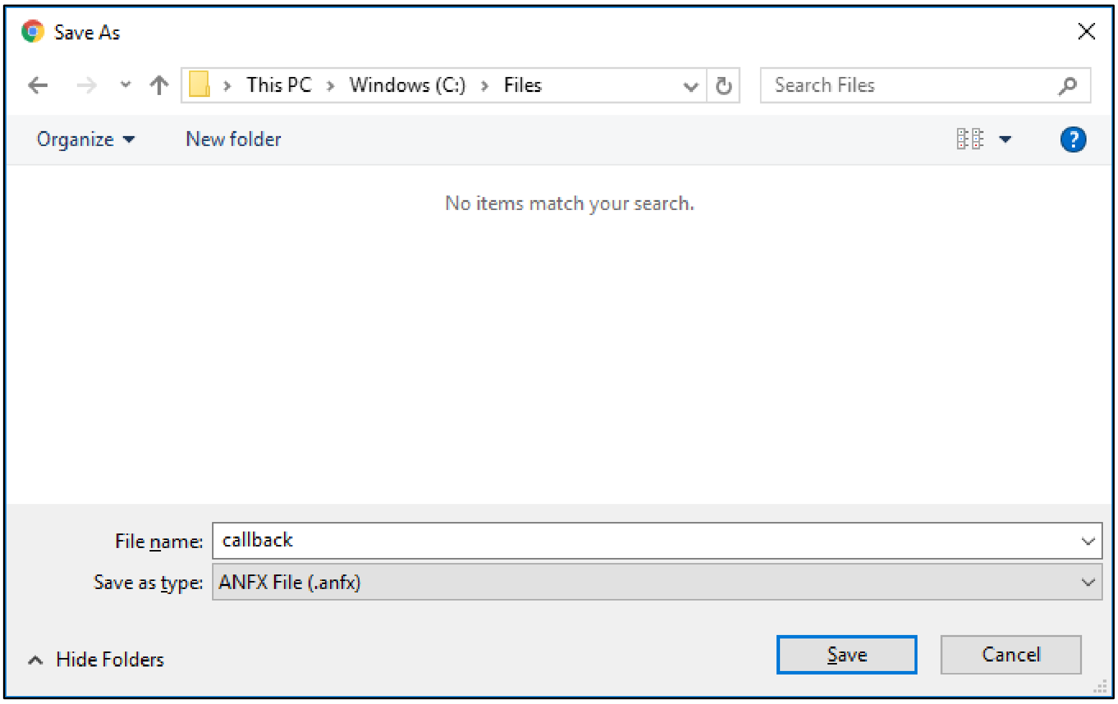
Figure: Error while exporting the nodeflow

If the error is displayed, then you can either fix the issue or click "Export Anyways" to continue exporting the nodeflow.

"Export" option is based upon "Save As" functionality of the Web browser. If "Ask where to save each file before downloading" option is not checked, the Nodeflow will be exported as ".anfx" file at the default download location of the Web browser, which will have the name provided while creating the Nodeflow. In this case, the Administrator does not get the option to change the name and path of the file.

It is recommended to keep "Ask where to save each file before downloading" or similar option checked in your Web Browser so that you can specify the name and path to save the files.

If "Ask where to save each file before downloading" or similar option is enabled in your Web Browser, then the following dialog box is displayed while exporting the Nodeflow.



File name: callback

Save as type: ANFX File (.anfx)

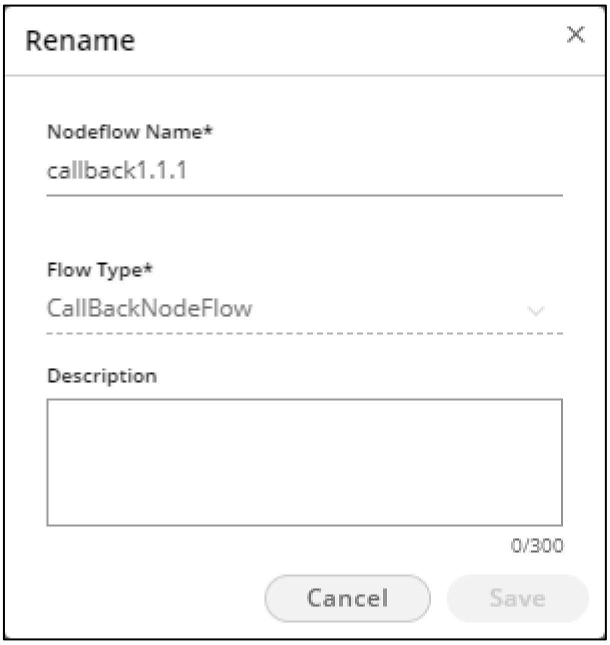
Figure: Export File

Whenever the changes are made in the nodeflow, Ameyo Synthesizer will write the nodeflow when the User saves it. However, the changes are not written to the exported files. Therefore, the User have to export the file each time after making the changes.

Select the location and provide a name for the file. Click "Save" to save the file on the disk.

- **Rename**: Perform the following steps to rename a nodeflow.

  1. Click "Rename" option. It shows the following modal.



Rename

Nodeflow Name*: callback1.1.1

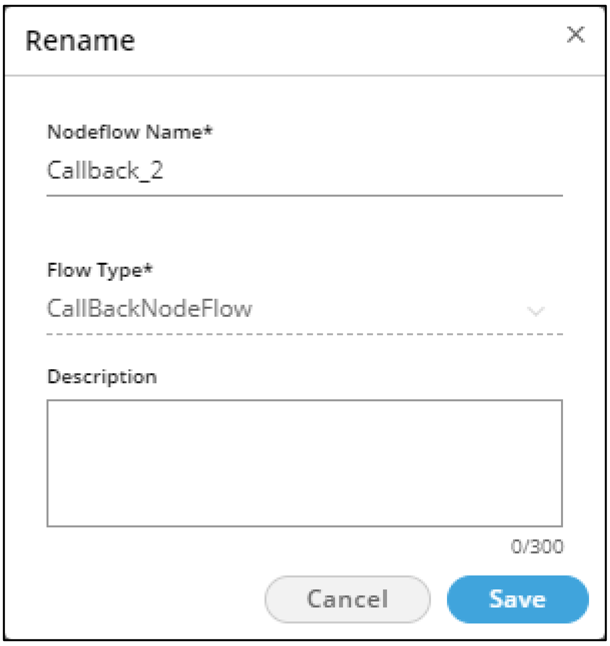
Flow Type*: CallBackNodeFlow

Description

0/300

Figure: "Rename Nodeflow" Modal

2. Enter the new name of the nodeflow.



Rename

Nodeflow Name*: Callback_2

Flow Type*: CallBackNodeFlow

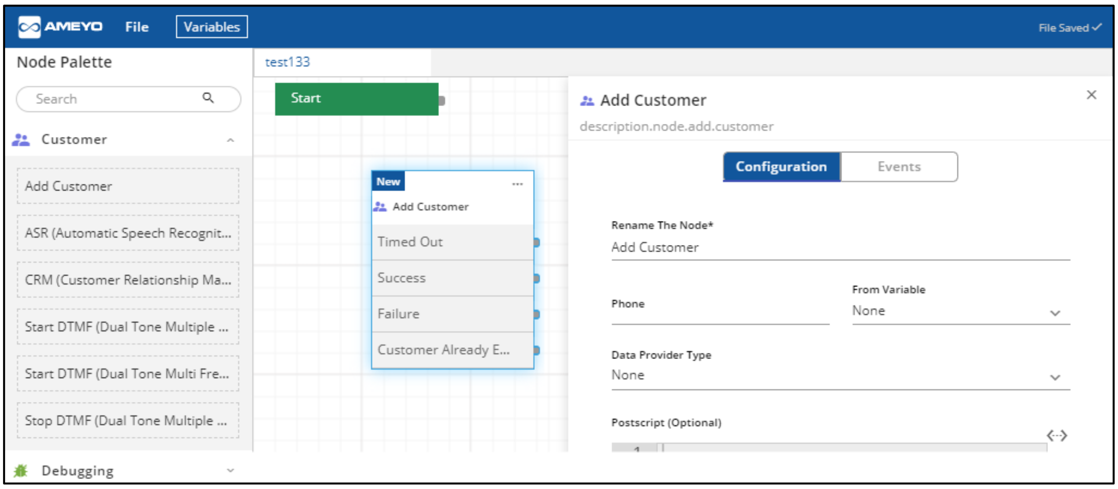
Description

0/300

Figure: Renaming the Nodeflow

3. Click "Save" button. The selected nodeflow is renamed.

- **Save:** After making the changes in nodeflow, it is recommended to save the nodeflow. Click "Save" option in the file menu. After saving the nodeflow, a small notification message "File is saved" is displayed in the top right corner.



Add Customer
description.node.add.customer

Rename The Node* Add Customer

Phone

From Variable None

Data Provider Type None

Postscript (Optional)

Figure: Nodeflow is saved.

- **Duplicate:** Perform the following steps to create a duplicate copy of a nodeflow.

1. Click ⋮ to access its menu.
2. Click "Duplicate" option. A copy of the nodeflow is created with the name <nodeflow_name.1> and the nodeflow is opened in a new tab of the same Browser Window.

If the user is creating the duplicate of an already duplicated nodeflow, then its name will be <nodeflow_name.1.1>. Another ".1" will be suffixed if another duplicate copy is created.

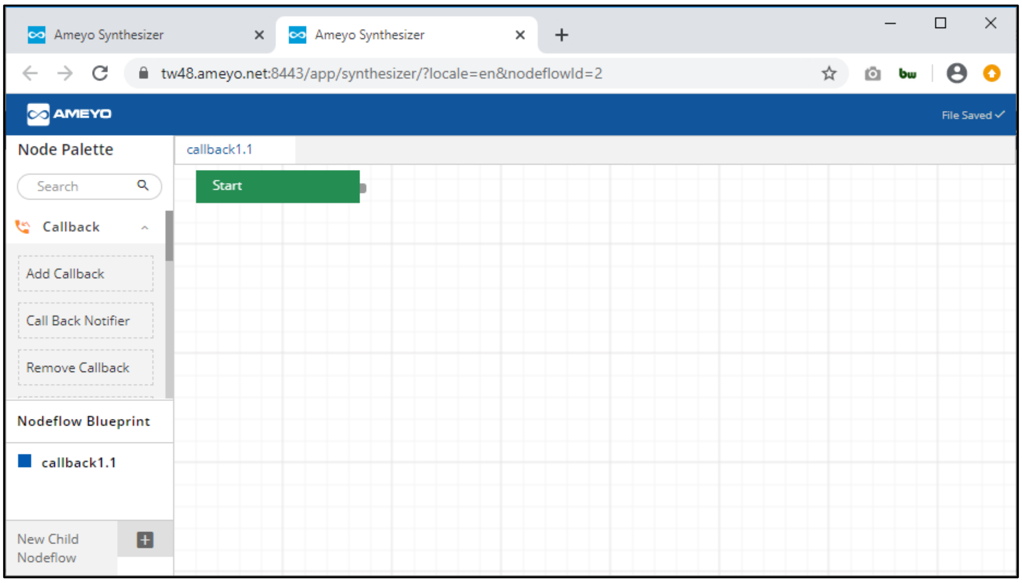


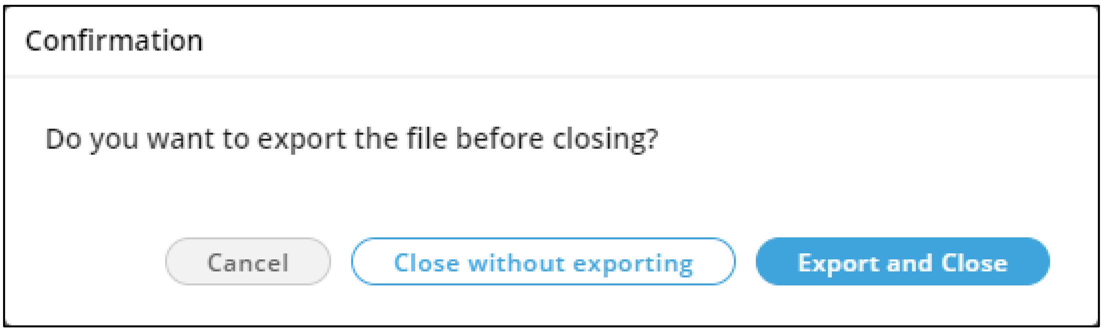
Figure: Duplicate Nodeflow in the new Tab

With the creation, the duplicate nodeflow is saved automatically and a message is displayed in the top right corner of the window.

3. Switch to the first page of Synthesizer, which shows the duplicate nodeflow.

- **Send Error Logs:** Click it to send the error logs to our Internal Team. It sends the logs related to the current Nodeflow only.
- **Close:** Click it to close the current instance of Ameyo Synthesizer. Other active session instances of Synthesizer opened in other Browser Windows or Browser Tabs will continue to function.

  If the Nodeflow has not been exported earlier, the following pop-up is displayed.


Confirmation

Do you want to export the file before closing?

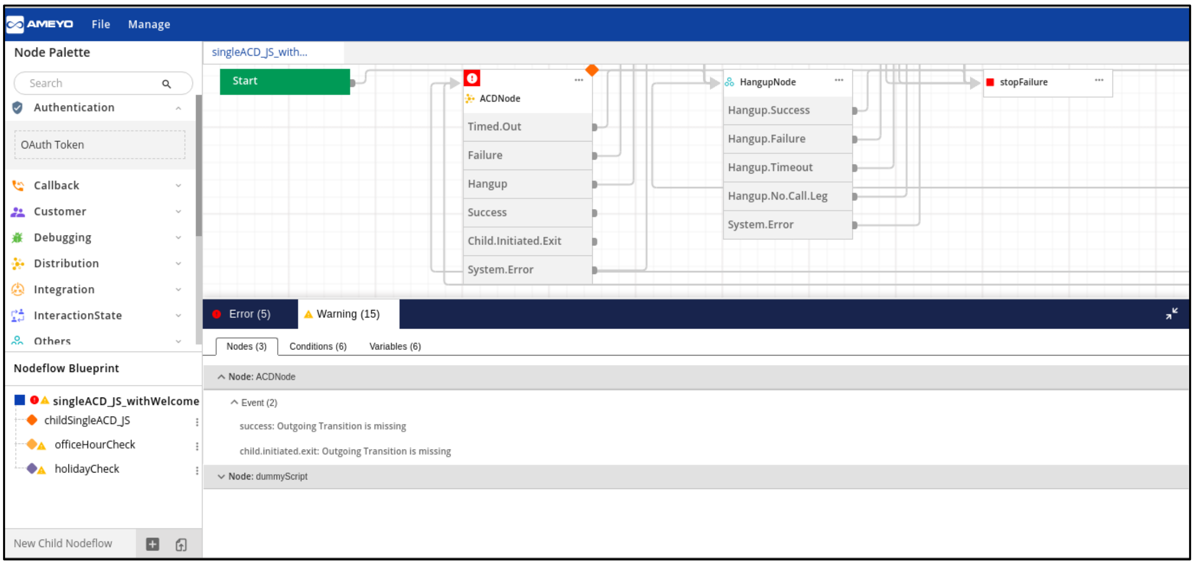
Figure: Close Synthesizer

It contains the following options.

-
    - **Cancel:** Click it to cancel the closing of Synthesizer.
    - **Close without Exporting:** Click it to not save the Nodeflow file and close Synthesizer.
    - **Export and Close:** Click it to save the Nodeflow file and close Synthesizer. Synthesizer will be closed, and there is an option to save the Nodeflow as ".anfx" file.

5.2 Introduced the Error Palette

The User has to go through the different nodeflows while designing a nodeflow. Every page in Synthesizer can have an error because of invalid user actions or user inputs. If the User is getting the errors on multiple pages, there is no common list of errors displayed in Synthesizer. To help the User, Ameyo introduced the Error Palette in Synthesizer at the bottom of the page. It shows all errors and warnings in a single place.



Node: ACDNode

Event (2)

success: Outgoing Transition is missing

child.initiated.exit: Outgoing Transition is missing

Node: dummyScript

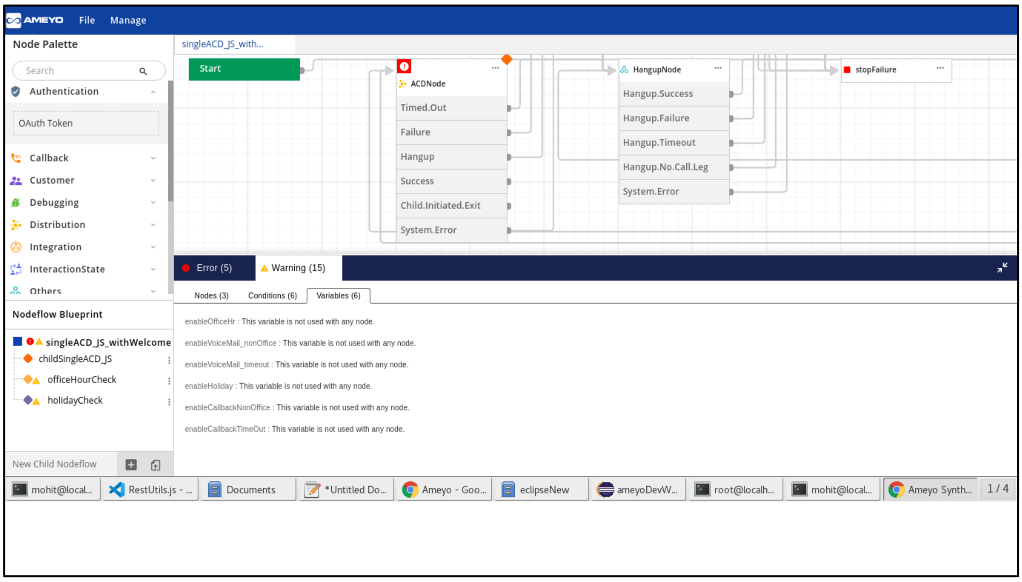
Figure: Error Palette

It contains the following tabs.

- **Errors:** It contains the error messages. A red-colored circle denotes its icon. It contains the following sub-tabs.
  - **Nodes:** It lists the error messages being displayed in the Nodes.
  - **Conditions:** It lists the error messages being displayed in the Conditions.
  - **Variables:** It lists the error messages being displayed in the Variables.

  If any element such as Node, Condition, or Variable does not have any error, then its tab will not be displayed.

- **Warnings:** It contains the warnings. A yellow colored triangle denotes its icon. It contains the following sub-tabs.
  - **Nodes:** It lists the warnings being displayed in the Nodes.
  - **Conditions:** It lists the warnings being displayed in the Conditions.
  - **Variables:** It lists the warnings being displayed in the Variables.



Error (5) | Warning (15)

Nodes (3) | Conditions (6) | Variables (6)

enableOfficeHr : This variable is not used with any node.

enableVoiceMail_nonOffice : This variable is not used with any node.

enableVoiceMail_timeout : This variable is not used with any node.

enableHoliday : This variable is not used with any node.

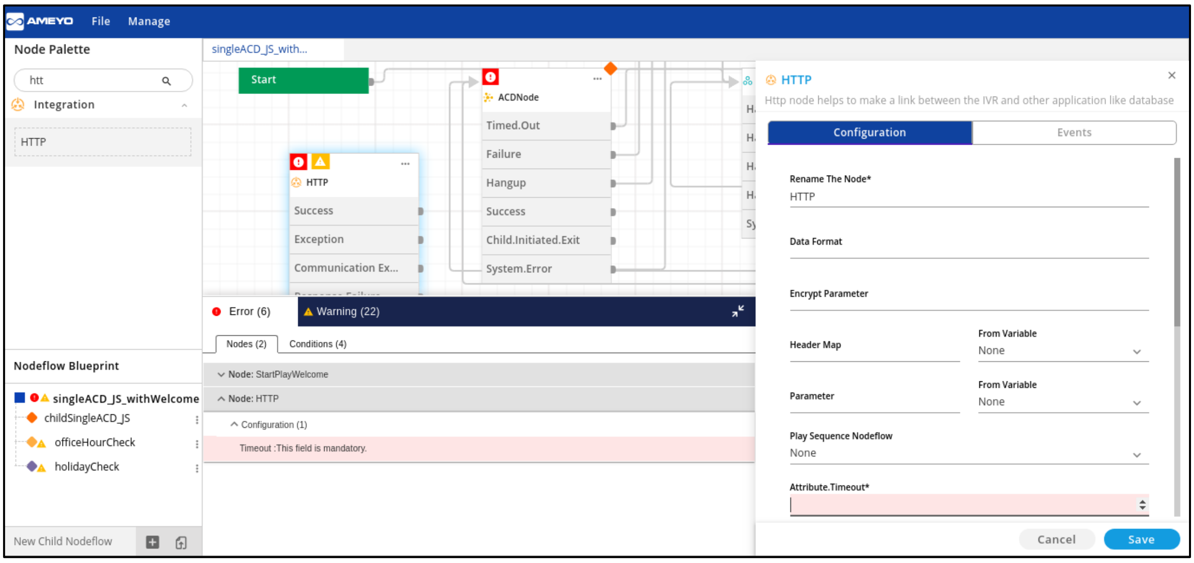
enableCallbackNonOffice : This variable is not used with any node.

enableCallbackTimeOut : This variable is not used with any node.

Figure: Warnings in Variables

If any element such as Node, Condition, or Variable does not have any warning, then its tab will not be displayed.

The User can go through these tabs to view the information about errors and warnings. The User can click any error message to visit the place where the error has occurred.



Error (6) | Warning (22)

Nodes (2) | Conditions (4)

Node: StartPlayWelcome

Node: HTTP

Configuration (1)

Timeout :This field is mandatory.

HTTP

Http node helps to make a link between the IVR and other application like database

Rename The Node*: HTTP

Data Format

Encrypt Parameter

Header Map — From Variable: None

Parameter — From Variable: None

Play Sequence Nodeflow: None

Attribute.Timeout*

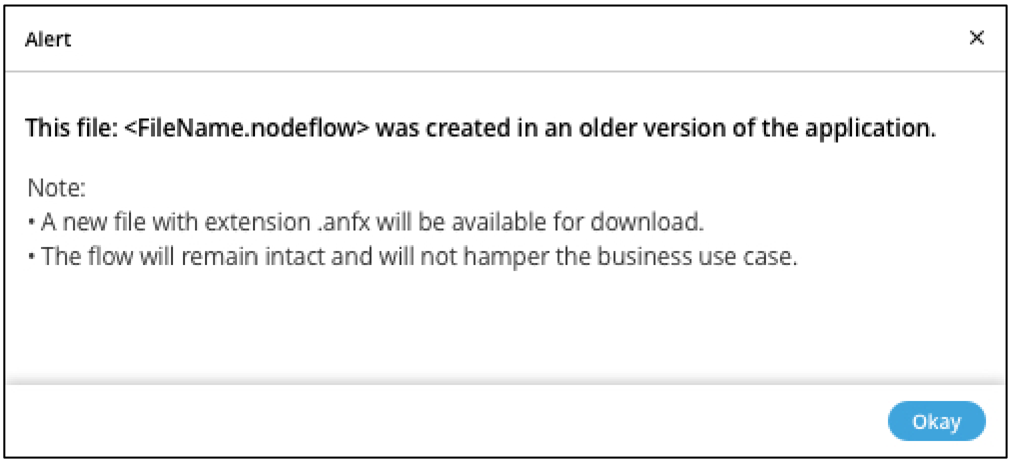
Figure: Redirect to the location from the Error Message

5.3 Import Nodeflow File of Nodeflow Designer

Nodeflow Designer is the predecessor of Synthesizer. The nodeflows created in it are saved with ".nodeflow" extension. Synthesizer now provides a limited period functionality to migrate a ".nodeflow" file to ".anfx". In Ameyo 4.5, Synthesizer can open ".nodeflow" file in its interface. After opening the nodeflow, the user can export it to save as a ".anfx" file.

The feature to open ".nodeflow" file in Ameyo Synthesizer is available for the limited period and can be removed in future.

The following message is displayed when a ".nodeflow" is tried to open in Ameyo Synthesizer.



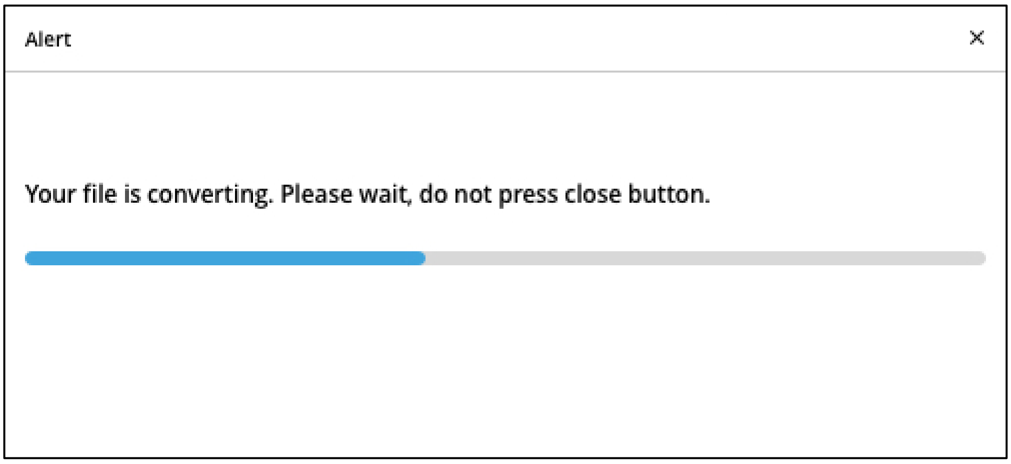
Alert

This file: <FileName.nodeflow> was created in an older version of the application.

Note:
- A new file with extension .anfx will be available for download.
- The flow will remain intact and will not hamper the business use case.

Figure: Message while trying to open a ".nodeflow" file

The user can click "Okay" to continue to open ".nodeflow" file. The following progress bar shows the progress for opening the file.

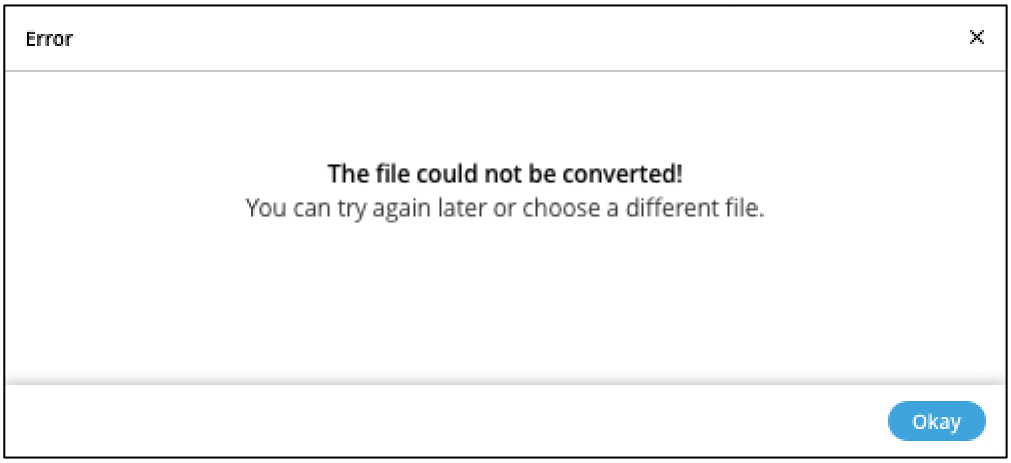


Your file is converting. Please wait, do not press close button.

Figure: Opening a ".nodeflow" file

5.4 Error while opening a .nodeflow File

If any issue occurs while opening ".nodeflow" file, the following error message is displayed on the screen.



The file could not be converted!
You can try again later or choose a different file.

Figure: Error while opening a ".nodeflow" file

5.5 Variables will now remain shared between the Parent and Child Nodeflow while importing ".nodeflow" to ".anfx" format

The variables are shared in the Parent and Child Nodeflows in .nodeflow file format in Nodeflow Designer. When a .nodeflow file containing such Parent and Child Nodeflows is uploaded Synthesizer for migration to ".anfx" format, the Variables were not being shared between Parent and Child Nodeflows automatically.

Whereas in ".anfx" nodeflow in Synthesizer, a variable of parent nodeflow with the same name has to be created in Child Nodeflow manually to share that variable between Parent and Child Nodeflows.

However, this behavior has to be handled automatically while migrating ".nodeflow" files to ".anfx" files. It has been handled now. When any .nodeflow file having Parent and Child Nodeflows with shared variables is uploaded on Synthesizer, then the variables will be created in both Parent and Child Nodeflows in the migrated ".anfx" file and these variables will be shared between them automatically.

6. Open Nodeflow

Synthesizer allows the user to open the Nodeflows from the first page that manages the nodeflows. The user can click "Create New" to either create a new nodeflow or open the existing Nodeflows saved in ".anfx" formats.

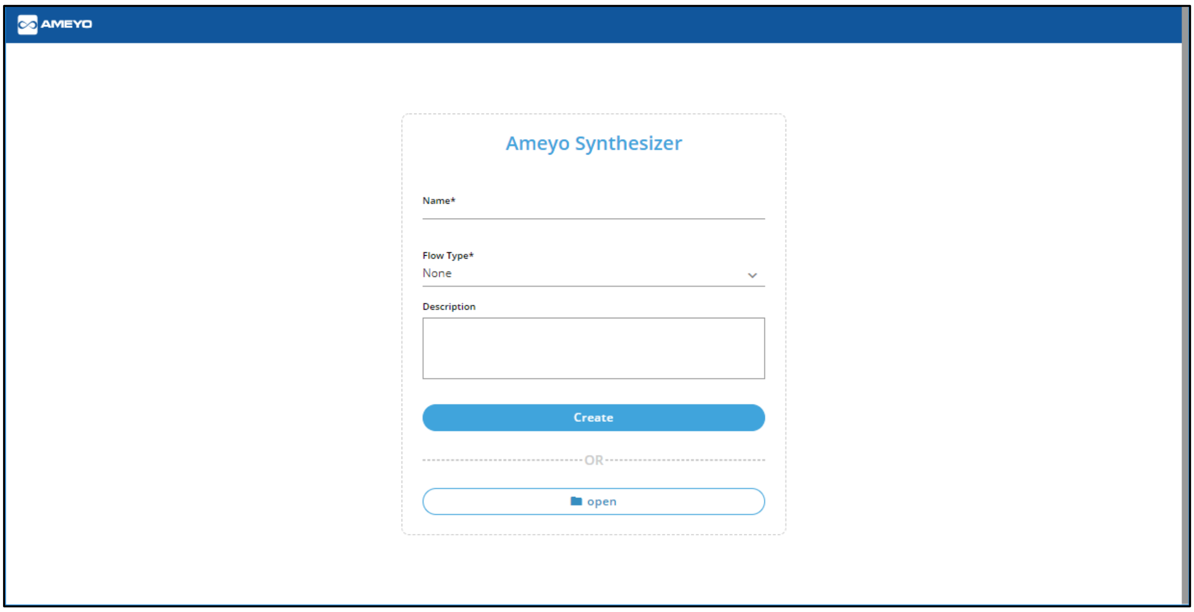
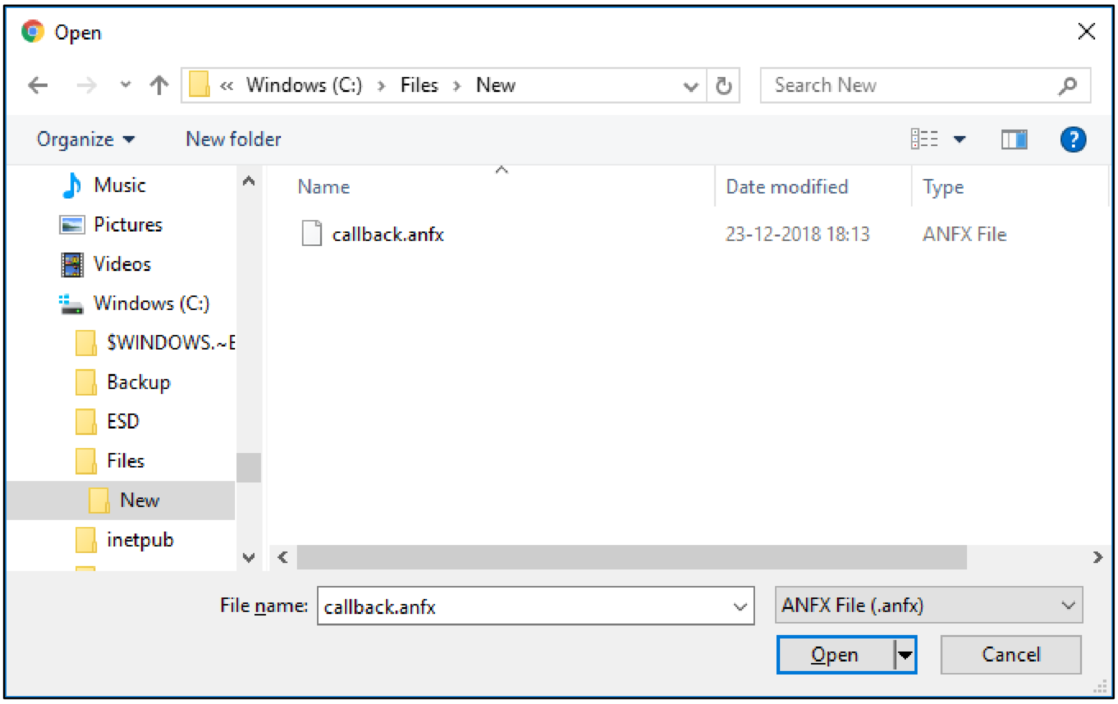


Figure: Synthesizer Page

Perform the following steps.

1. Click "Open" to open the Nodeflow. It shows the following dialog box.



Name	Date modified	Type
callback.anfx	23-12-2018 18:13	ANFX File

File name: callback.anfx

ANFX File (.anfx)

Figure: Open ANFX File

2. Select the location where ".anfx" file is saved.

3. Select the file and click "Open". It opens the file in the Synthesizer.

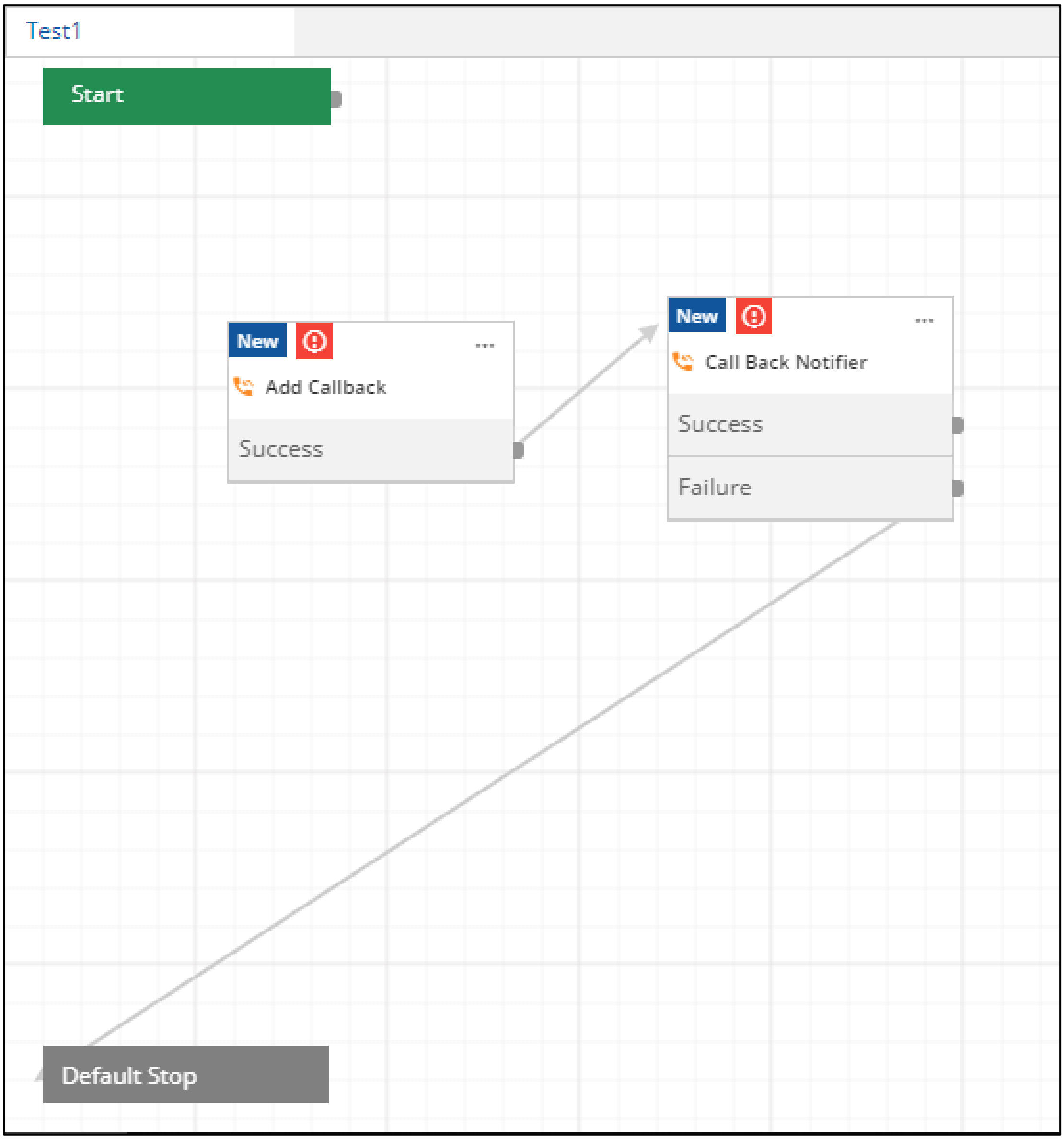


Figure: Opened the Nodeflow

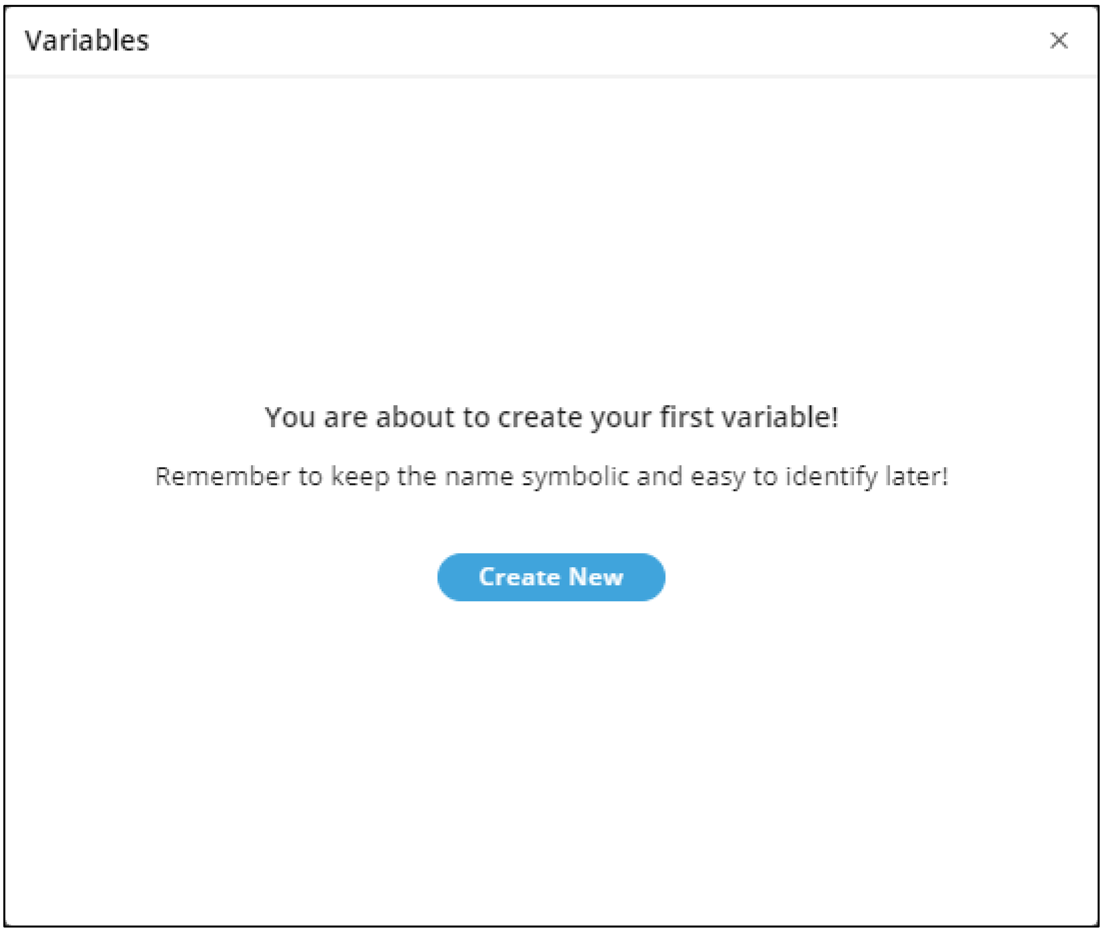
Now, the user can continue to modify the Nodeflow.

While exporting this modified Nodeflow (after opening ".anfx file), a new and different file will be created.

7. Manage Variables

In Synthesizer, the user can create the variables in the Nodeflow, which can be used while configuring the attributes in the configuration of a node.

Click "Variable" button on top to access and manage the variables in Synthesizer.



Variables

You are about to create your first variable!

Remember to keep the name symbolic and easy to identify later!

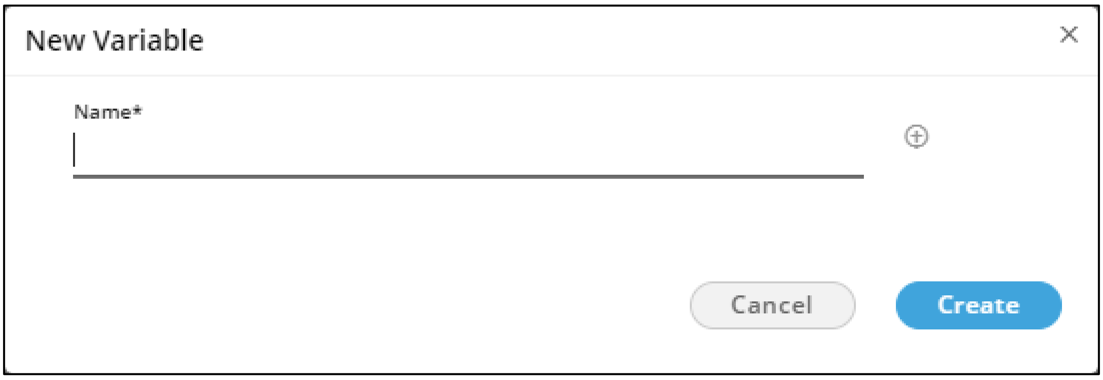
Figure: Variable

This message is displayed because no variable is created as of now.

7.1 Create Variable

Perform the following steps here to create a variable.

1. Click "Create New" button. It shows the following pop-up.

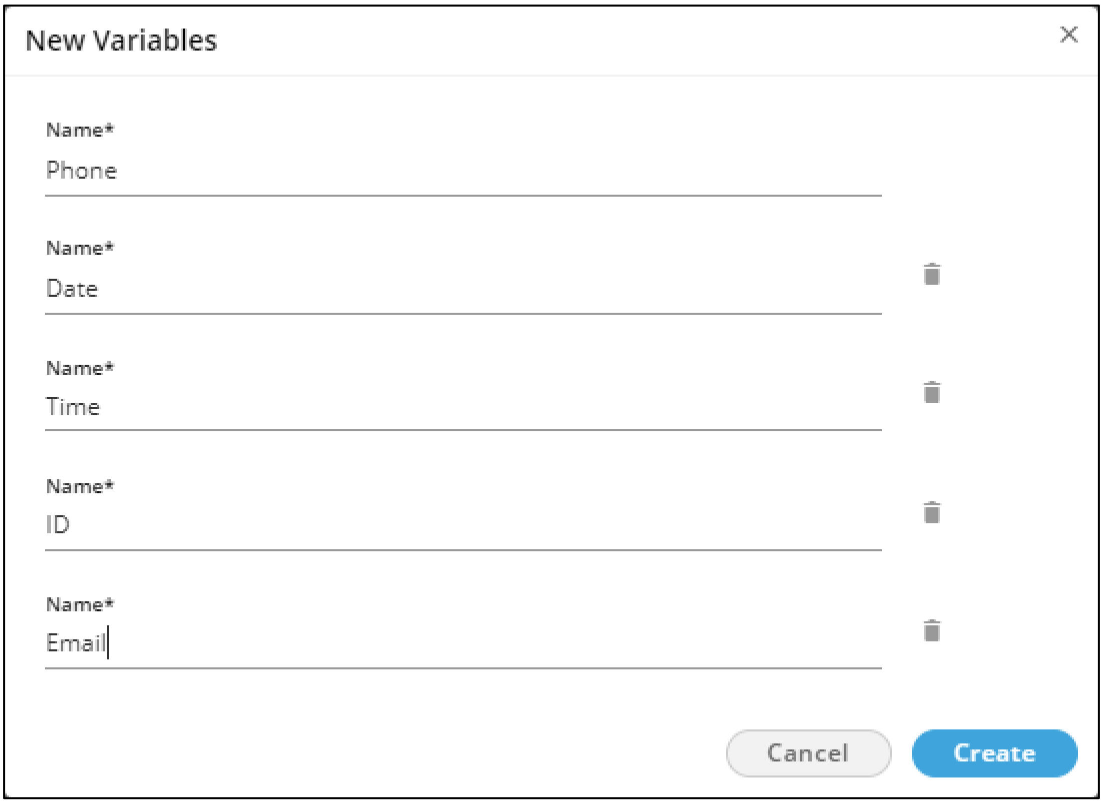


New Variable

Name*

Figure: Create New Variable

2. Provide a unique and easy-to-recall name to the variable. The variable can be used as a value for different parameters (such as phone, date, additional parameter, and others) in the configurations of nodes.
3. Click ⊕ icon to create a new variable. It adds a new row in the same pop-up. You can add up to 5 variables at a time.



Name*
Phone

Name*
Date

Name*
Time

Name*
ID

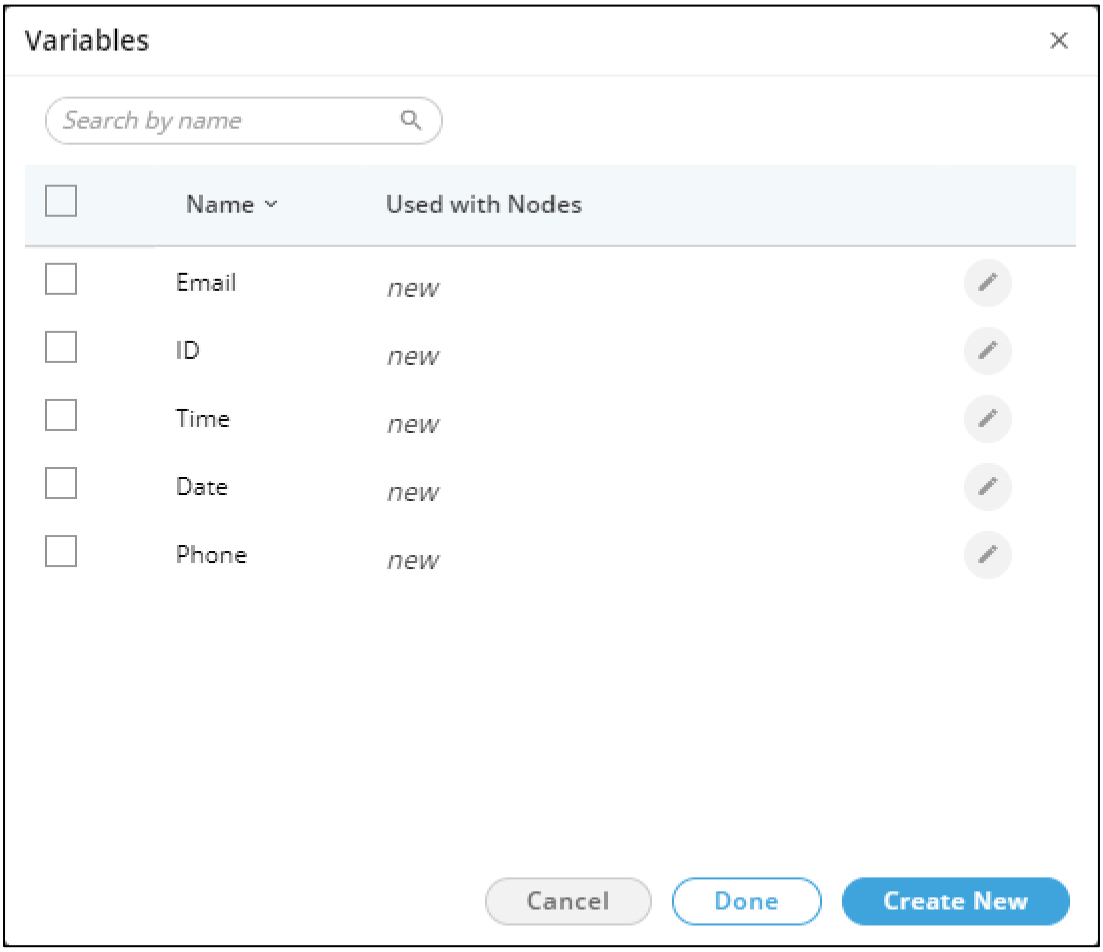
Name*
Email

Figure: Creating Two New Variables

Name of every variable should be unique. The error will be displayed if you are creating more than one variables with the same name.

You cannot create more than 5 variables at a time.

4. Click "Create". It creates the variable and shows the following pop-up.



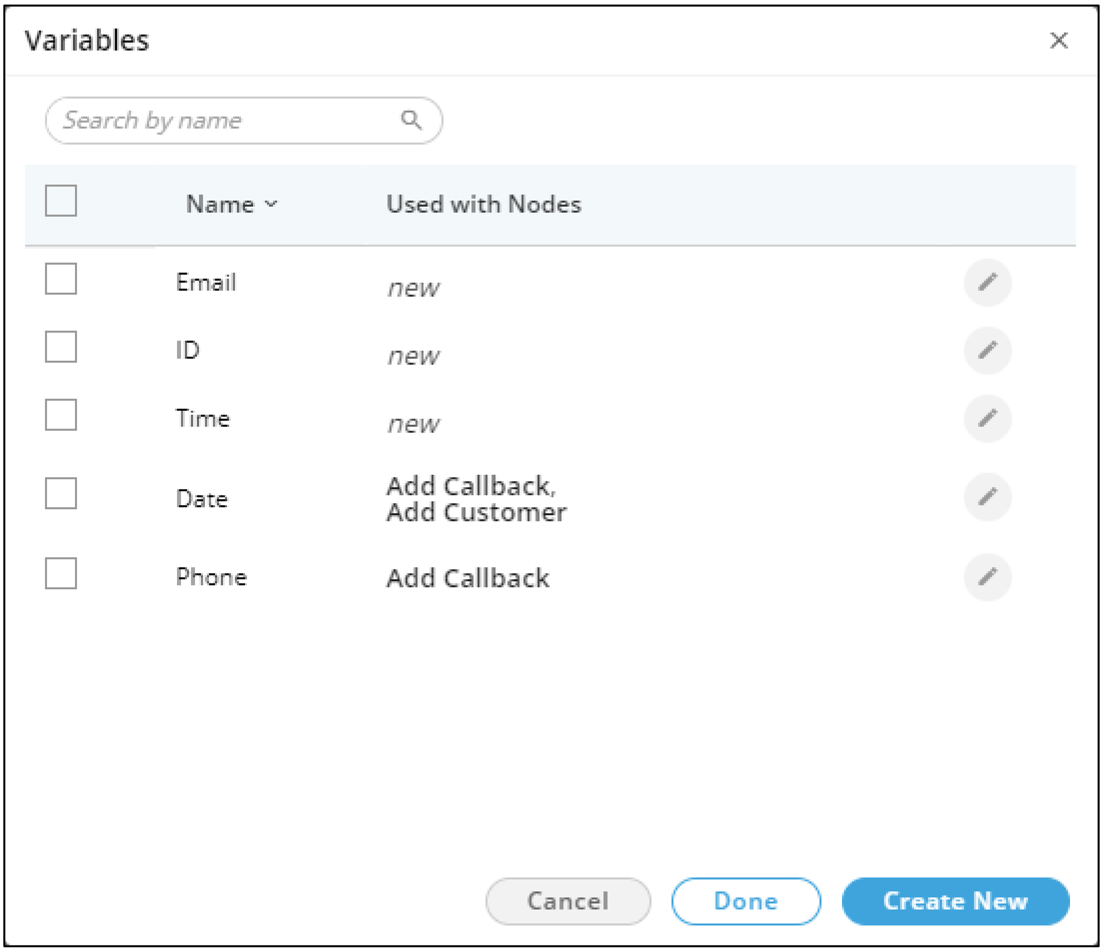
□	Name	Used with Nodes
□	Email	new
□	ID	new
□	Time	new
□	Date	new
□	Phone	new

Figure: Created Variables

5. Click "Create New" button here to repeat the process of creating more variables.
6. The Administrator can use the search box, which is located on top, to search for the particular variables.
7. Click ☒ icon to close this pop-up.
8. Click "Done" to close the pop-up.

7.2 Used with Nodes

"Used with Nodes" column shows the name of nodes in which a variable is being used.



□	Name	Used with Nodes
□	Email	new
□	ID	new
□	Time	new
□	Date	Add Callback, Add Customer
□	Phone	Add Callback

Figure: Usage in Nodes

The above screenshot shows the variables that are being used by the nodes.

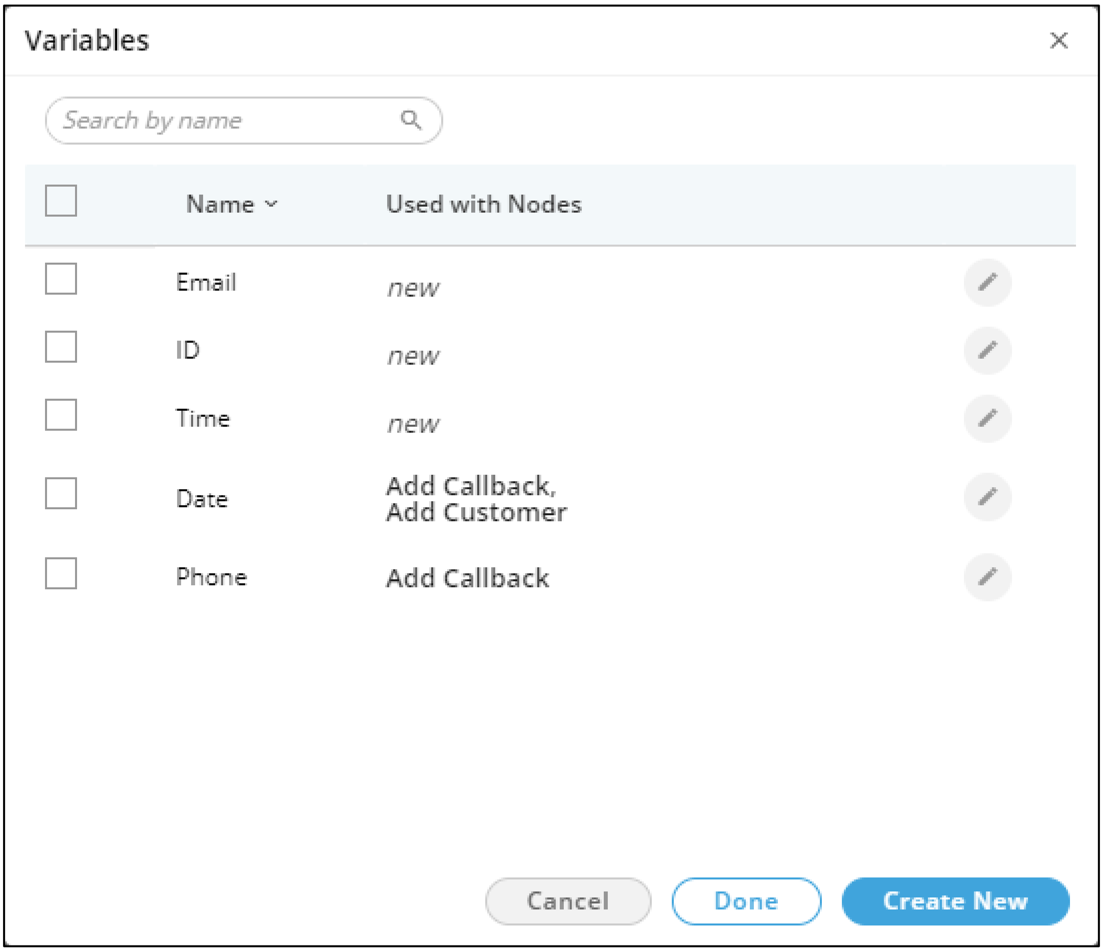
7.3 Share Variables with Child Nodeflow

To share a variable of a parent Nodeflow with the Child Nodeflow, the Administrator has to create the variable with the same name in Child Nodeflow. <<br /> For example, "P1" is the parent Nodeflow that has a variable named "Test" and "C1" is the Child Nodeflow of "P1". To share the variable, the user has to create "Test" variable in "C1".

7.4 Modify the Variable

Perform the following steps to modify the variable.

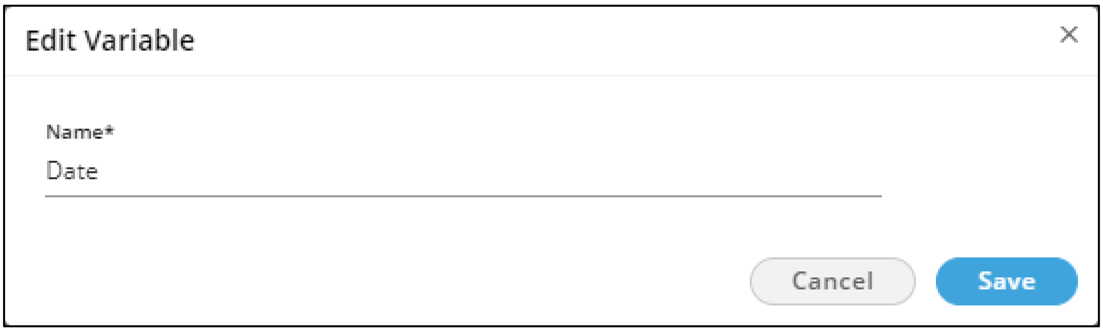
1. You can click "Variables" button to access the pop-up, which lists the variables.



□	Name ˅	Used with Nodes
□	Email	*new*
□	ID	*new*
□	Time	*new*
□	Date	Add Callback, Add Customer
□	Phone	Add Callback

Figure: List of Variables

2. Click ✎ icon to modify a variable. It shows the following pop-up.



Name*
Date

Figure: Modify Variables

3. You can change the name of the variable.

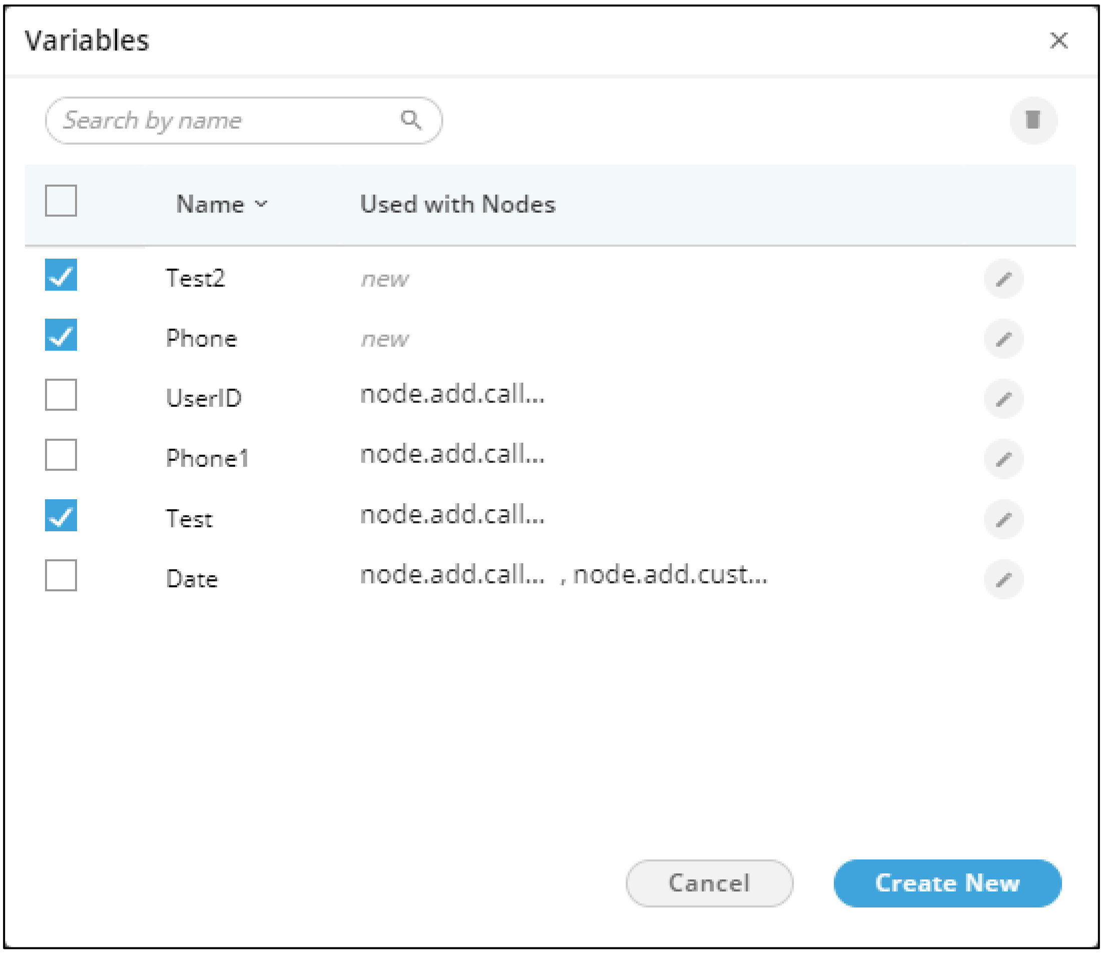
4. Click "Save" to save the variable.

7.5 Delete Variable

If the variables (that are selected for deletion) are being used in the nodes, the overall functionality of the Nodeflow might break. Also, the deleted variables cannot be restored.

Perform the following steps to delete the variables.

1. Select the variables that you want to delete.



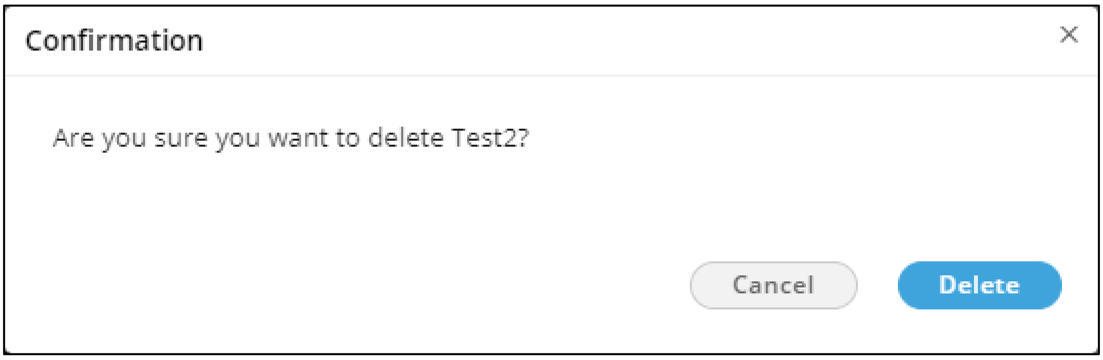
Variables

□	Name	Used with Nodes
☑	Test2	*new*
☑	Phone	*new*
□	UserID	node.add.call...
□	Phone1	node.add.call...
☑	Test	node.add.call...
□	Date	node.add.call... , node.add.cust...

Figure: Select the Variables

2. Click 🗑 icon to delete the selected variables.

Case 1 (Deleting Single Variable): The following message is displayed if a single variable is being deleted.

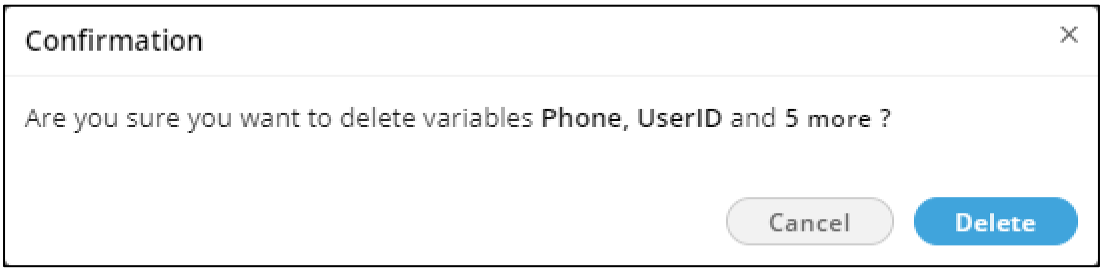


Confirmation

Are you sure you want to delete Test2?

Figure: Warning before deleting a single variable

Case 2 (Deleting Multiple Variables): The following message is displayed if multiple variables are being deleted at a time.

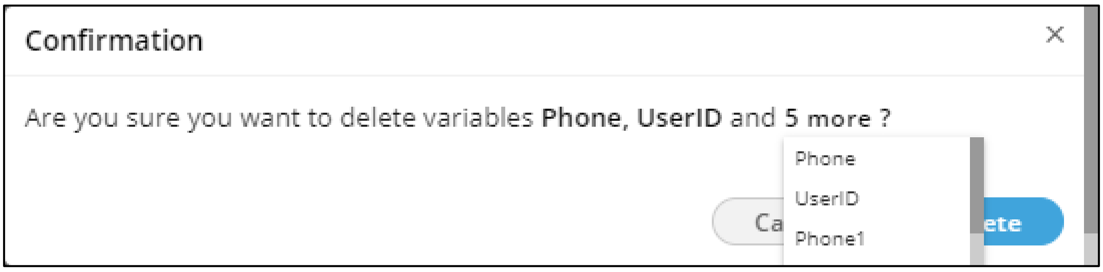


Confirmation

Are you sure you want to delete variables **Phone, UserID** and **5 more** ?

Figure: Warning before deleting the variables

In the case of multiple variables, the user can click "more" word to see the list of variables being deleted.

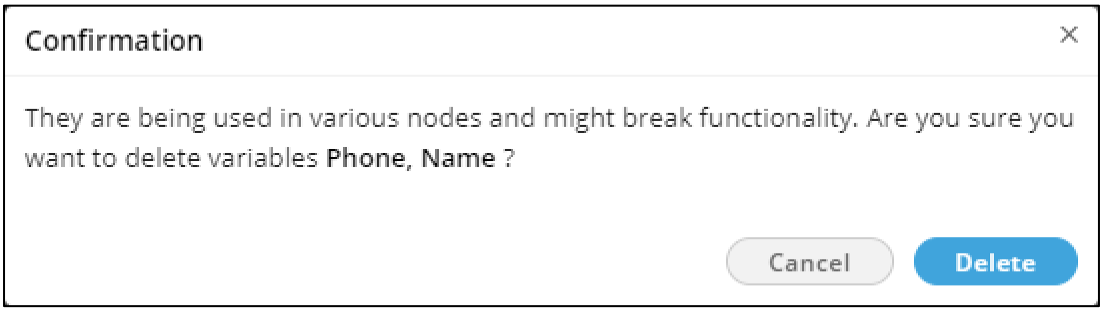


Confirmation

Are you sure you want to delete variables **Phone, UserID** and **5 more ?**

Figure: List of variables being deleted in the Warning Message

Case 3 (Deleting Variables that are being used in Nodes): The following message is displayed if you are trying to delete those variables which are being used in the nodes.



Confirmation

They are being used in various nodes and might break functionality. Are you sure you want to delete variables **Phone, Name** ?

Figure: Showing the variables are in use

3. Click "Delete" to delete the selected variables.

8. Design Nodeflow

Perform the following steps to design a Nodeflow.

1. Browse the categories and select a node.
2. Drag this node and drop it to the canvas. For example, "Add Callback" node is dragged in the canvas. It will be the first node after "Start".

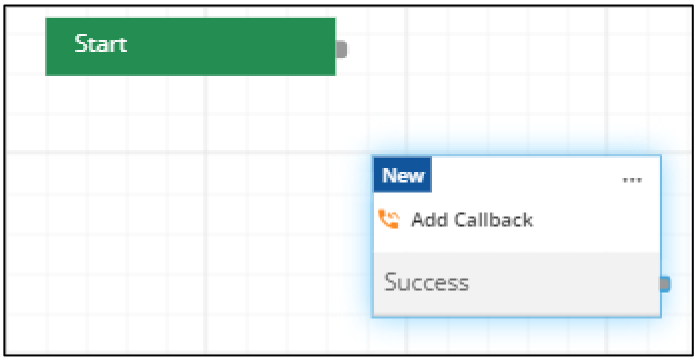


Figure: Adding a Node

> The user can again drag-and-drop the same node to the Canvas Area. However, the other copy of the same node will be named differently.

3. The node shows the events. The nodes that are meant to stop the execution will be directed to "Default Stop" by default.
4. Click "..." icon at the right corner of a node to open the Node Menu, which contains the following options.

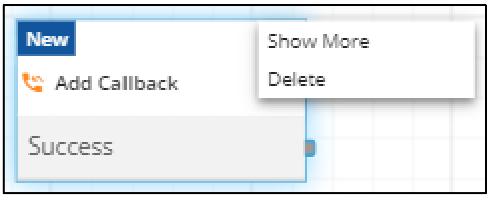
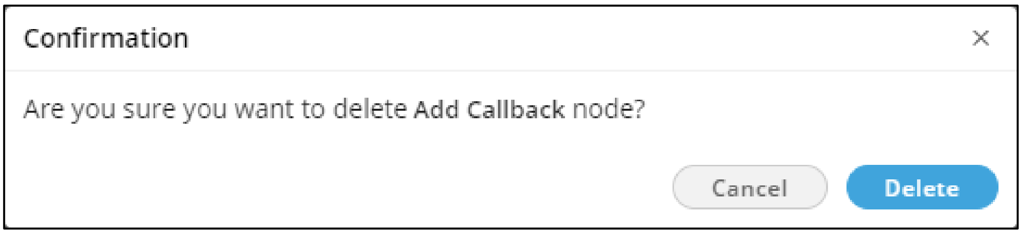


Figure: Node Menu

It contains the following options.

- **Show More:** Click it to show all events.
- **Delete:** Click it to delete the node.

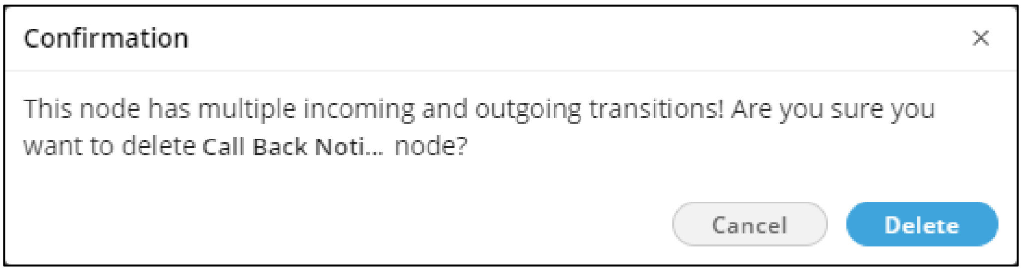
Case 1 (Deleting Single Node): It shows the following message.


Confirmation

Are you sure you want to delete **Add Callback** node?

Figure: Warning before deleting a Node

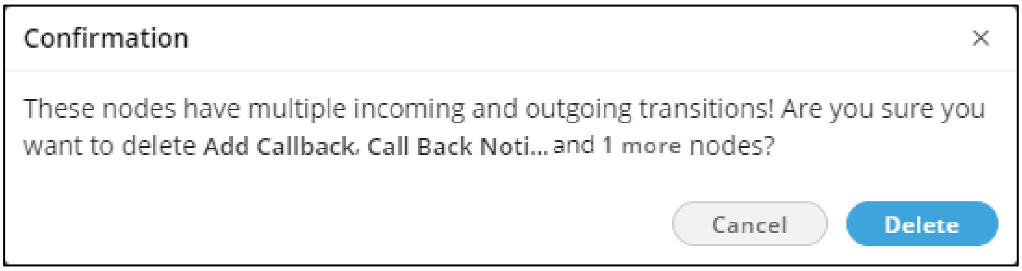
Case 2 (Deleting Single Node that has transitions): If the node has transitions to other nodes, the following message is displayed while trying to delete it.


Confirmation

This node has multiple incoming and outgoing transitions! Are you sure you want to delete **Call Back Noti...** node?

Figure: Warning before deleting a Node that has transitions

Case 3 (Deleting Multiple Nodes):The user can select multiple nodes clicking each node after pressing "CTRL" node. Now, press "DEL" key on the keyboard. It shows the following warning message.



Confirmation

These nodes have multiple incoming and outgoing transitions! Are you sure you want to delete **Add Callback, Call Back Noti...** and 1 more nodes?

Figure: Warning before deleting Multiple Nodes

You can click "more" button to view the name of the hidden node in the above pop-up.

If the nodes (selected for deletion) are a part of a Nodeflow, the overall functionality of the Nodeflow might break. Also, the deleted nodes cannot be restored.

Click "Delete" to delete the selected nodes. The user can also press "DEL" key.

When the node is showing all events, "Show More" is replaced with "Show Less" that means click it to show fewer events.

5. Drag more nodes to the canvas.

- At the canvas, scroll down continuously to zoom in and scroll up to zoom out.
- Click anywhere in the canvas (at the blank area) and move the mouse to any direction to move through the canvas.

6. Click the end point of "Start" node and bring the arrow to the node with which Nodeflow will start.

7. Click the end point of the node to see the two lines, which lets the Administrator decide the transition of the Nodeflow.

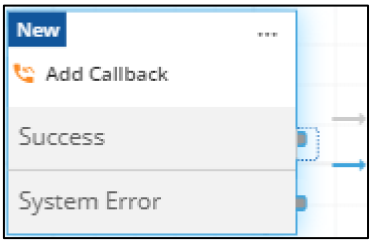
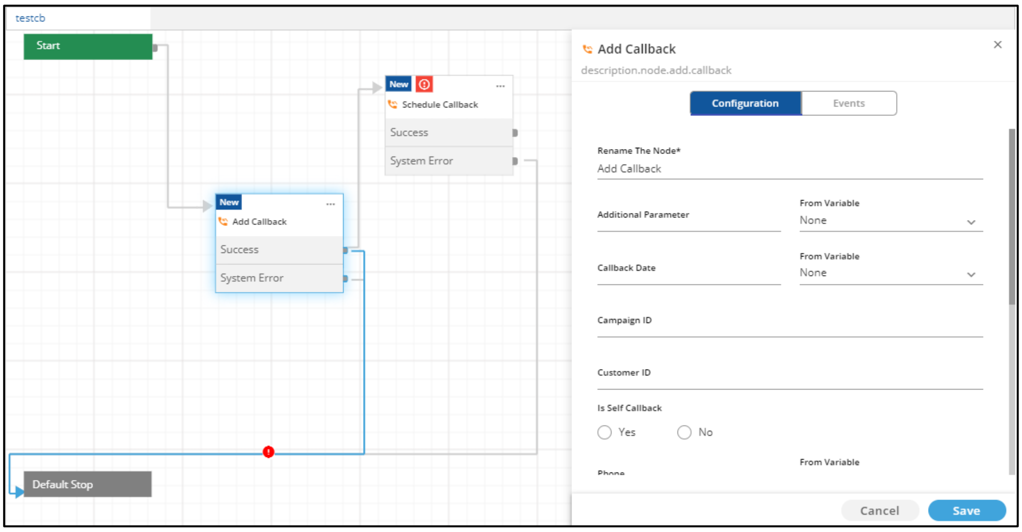


Figure: Transition Options

It contains the following two types of transition lines.

- **Unconditional Transition:** Select the grey line for the unconditional transition. Unconditional Transition is simple. Once an event of a node is generated, the event of another node will start executing. The process will stop at "Default Stop" node.
- **Conditional Transition:** Select the blue line for a conditional transition. If the administrator is connecting an event of a node to the event of another node with a blue directed line, the condition is displayed in the targeted event in its Node Configuration.



Add Callback

description.node.add.callback

Rename The Node*: Add Callback

Additional Parameter — From Variable: None

Callback Date — From Variable: None

Campaign ID

Customer ID

Is Self Callback: ○ Yes ○ No

Phone — From Variable

Figure: Conditional Transition

Specify the condition in a JavaScript Code. Click ⟨⋯⟩ to open the inbuilt JavaScript Editor to write the code.

Click "Save". Now, the transition will happen only when the condition specified in the JavaScript code is satisfied.

8. Connect one node to another using these transitions. Click anywhere between the lines to create a break point and then redirect it to another node.

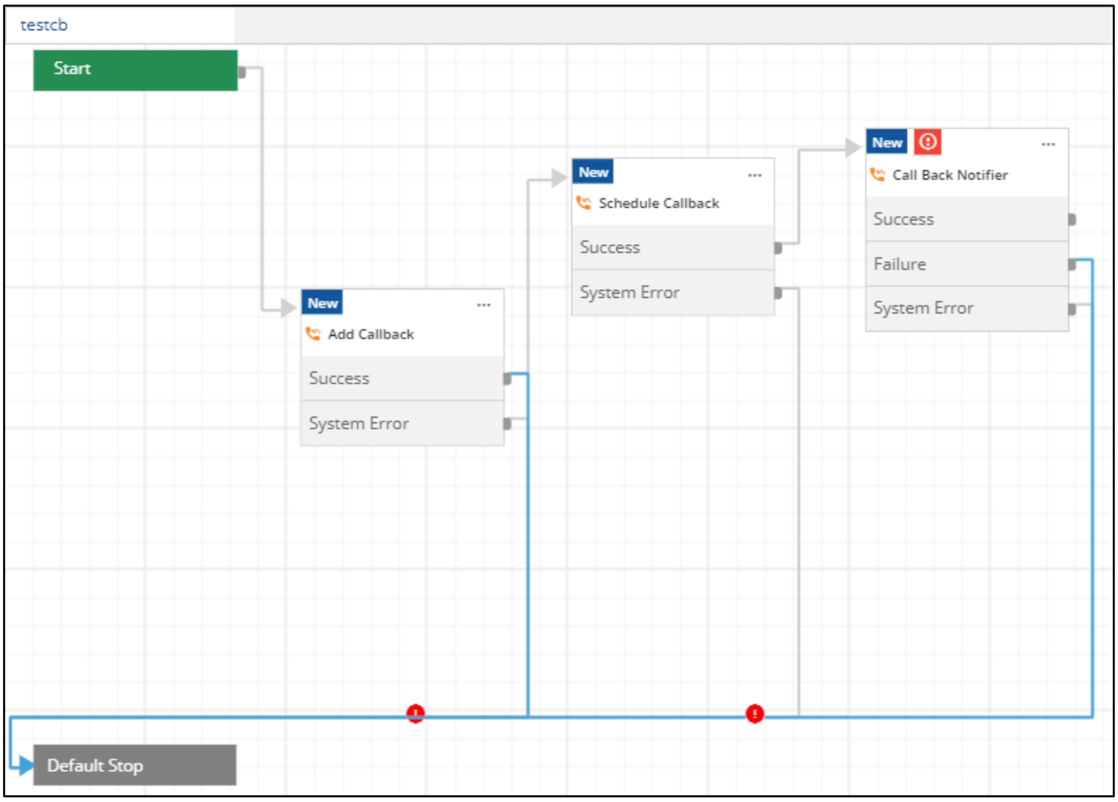
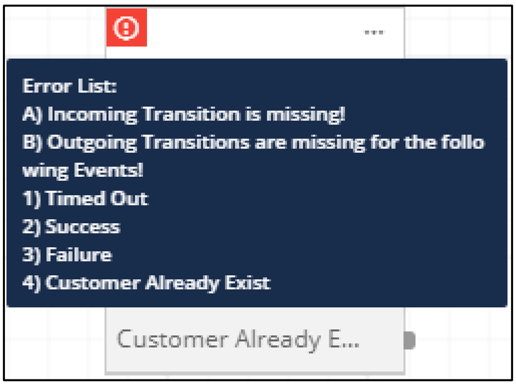


Figure: Sample Nodeflow

There are two arrows in transition from "Add Callback" and "Schedule Callback" nodes. Use the middle arrow to add another node here.

There can be multiple transition arrow-based lines from one node. Press and hold "CTRL+SHIFT" keys and do a left click of the pointing device (touchpad or mouse) on a transition to select it. Once selected, press "DEL" key to delete it.

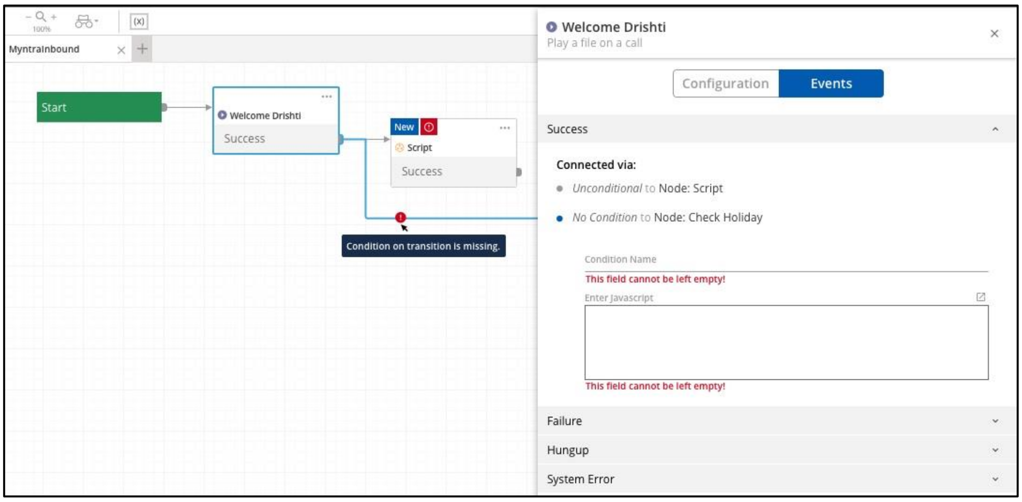
9. **<u>Validations and Errors</u>:** An error icon ❗ is displayed in the node if

- any mandatory attribute is missing.
- there is no "In-Transition" except the Start Node.
- there is no "Out-Transition" except the Default-Stop Node.
- there is a syntax error in Prescript or PostScript codes.
- 

Error List:
A) Incoming Transition is missing!
B) Outgoing Transitions are missing for the following Events!
1) Timed Out
2) Success
3) Failure
4) Customer Already Exist

Figure: Error Message List

- the node does not have any transition for an event.
- there is a syntax error in the Conditional Transition.
- the name or condition is not specified in the Conditional Transition. Any error related to an element on the interface is also listed on the interface in the red color.



Condition on transition is missing.

Connected via:

- *Unconditional* to Node: Script
- *No Condition* to Node: Check Holiday

Condition Name

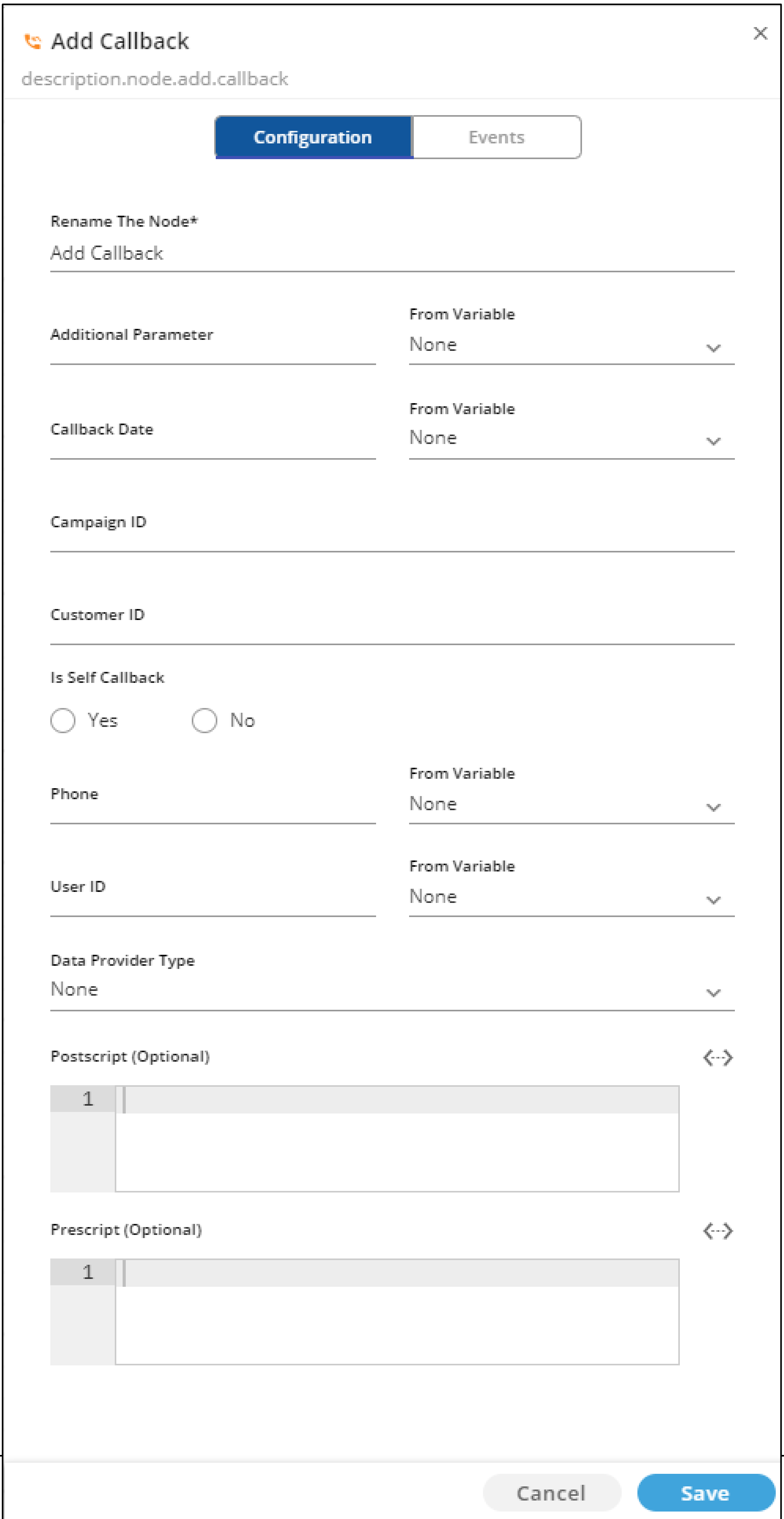
This field cannot be left empty!

Enter Javascript

This field cannot be left empty!

Figure: Validation error related to Conditional Transition of the node

10. The user can hover the mouse over the error icon to see the complete error message.
11. Click a node header, and it will show its properties on the right side of the canvas.



Add Callback

description.node.add.callback

Rename The Node*
Add Callback

Additional Parameter
From Variable
None

Callback Date
From Variable
None

Campaign ID

Customer ID

Is Self Callback
○ Yes ○ No

Phone
From Variable
None

User ID
From Variable
None

Data Provider Type
None

Postscript (Optional)

```
1
```

Prescript (Optional)

```
1
```

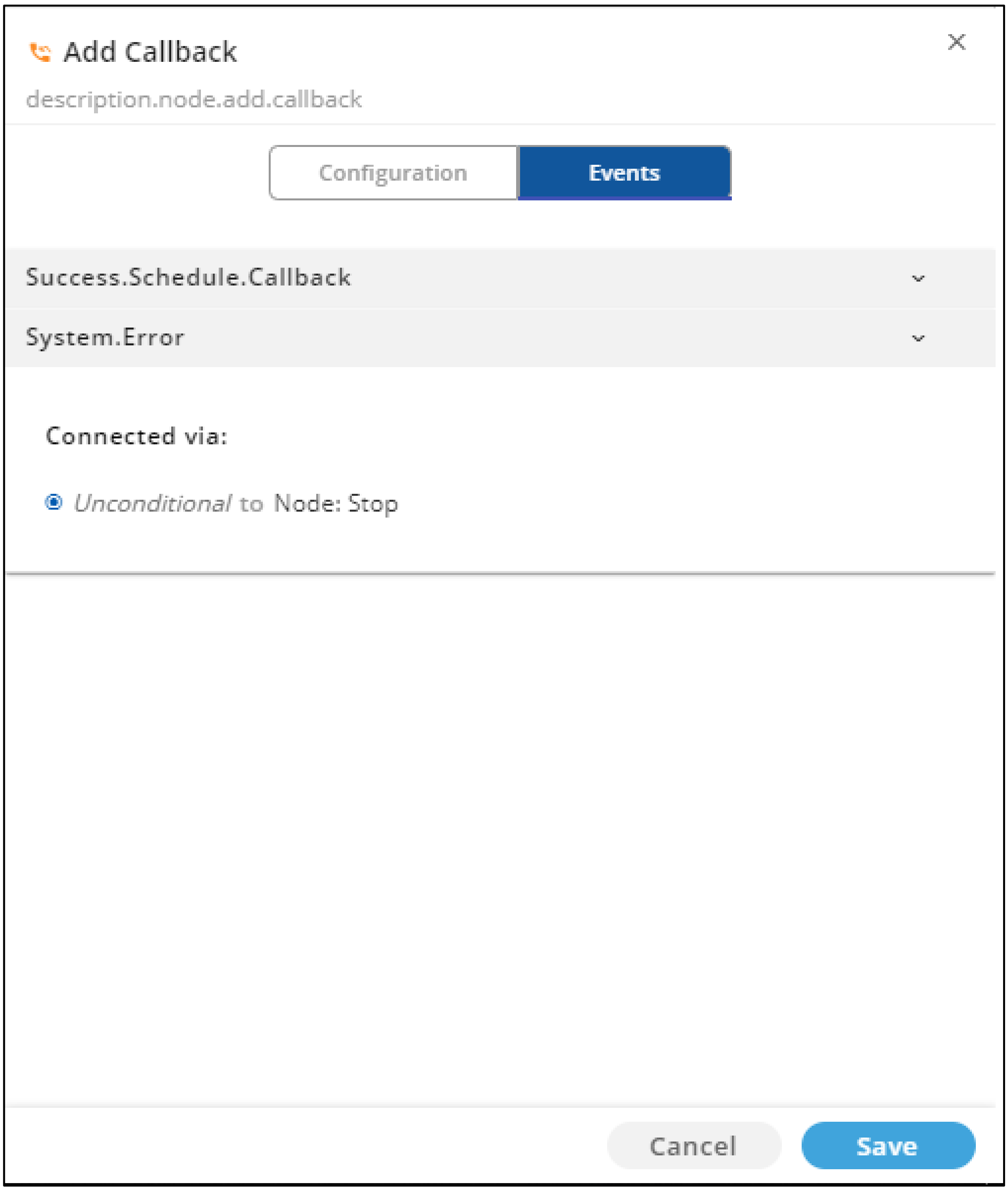
Figure: Node Properties

It contains the following two tabs.

- **Configuration:** This tab allows the user to configure the attributes, select the variables, and provide the custom JavaScript code that can run before and after the execution of the node.

  Make any change in the configuration and click "Save".

- **Events:** This tab contains the list of events available in this node.



Add Callback

description.node.add.callback

Success.Schedule.Callback

System.Error

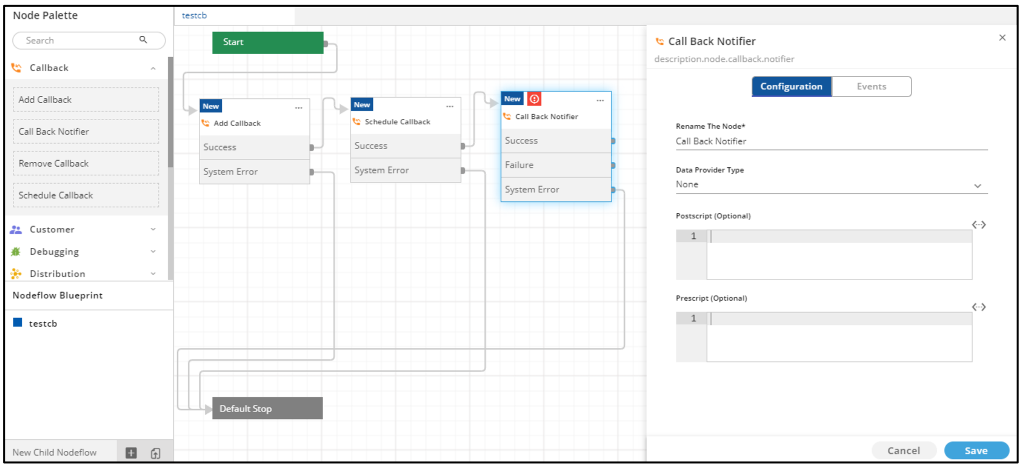
Connected via:

● *Unconditional* to Node: Stop

Figure: Events

Click any event to see to which node it is linked and using which transition (conditional or unconditional).

- Following is a screenshot of sample Nodeflow.



Rename The Node*: Call Back Notifier

Data Provider Type: None

Figure: Sample Nodeflow

Go to "File" Menu and click "Export" to save the Nodeflow.

- **Undo and Redo Actions:** The user can now undo (reverse the last action) and redo (reversing the last Undo operation) in Ameyo Synthesizer. The user can press **CTRL+Z** to undo the last actions, whereas, use **CTRL+Y** or **SHIFT+CTRL+Z** key combinations can be used to redo the action that has been undone.

  25 States of the following actions can be undone or redone.

  - Placement of nodes in the Canvas
  - Deletion of any element
  - **Saved Node Configuration Changes:** These changes are made in Node Configuration that has been saved by clicking "Save" button.

- **Saved Node Events Changes:** These changes are made in Events of Nodes that have been saved by clicking "Save" button.
- **Saved Transition Conditions:** These Transition Conditions have been saved by clicking "Save" button.
- **Saved Script Changes:** These changes are made in the Prescript and Postscript fields that have been saved by clicking "Save" button.

Similarly, the last 25 Undo Operations can be redone.

In this document, we have discussed the configuration of "Add Callback Node". The Configuration of all nodes is quite similar as the General Configuration discussed here and Configuration of "Add Callback Node". Know more...

8.1.1.1 Modal to help the User Experience for Quick Designing of Nodeflow

If the number of nodes is more than expected (for example, 30), then the nodes will be scattered over the large area in Synthesizer known as "canvas". It would not be possible for the user to visualize all nodes at once and connect them easily using the transition lines.

To solve this issue, "Connect to" modal has been introduced in Ameyo Application Server. When the user drags a transition from a node and drop it on the canvas, then "Connect to" modal is displayed.

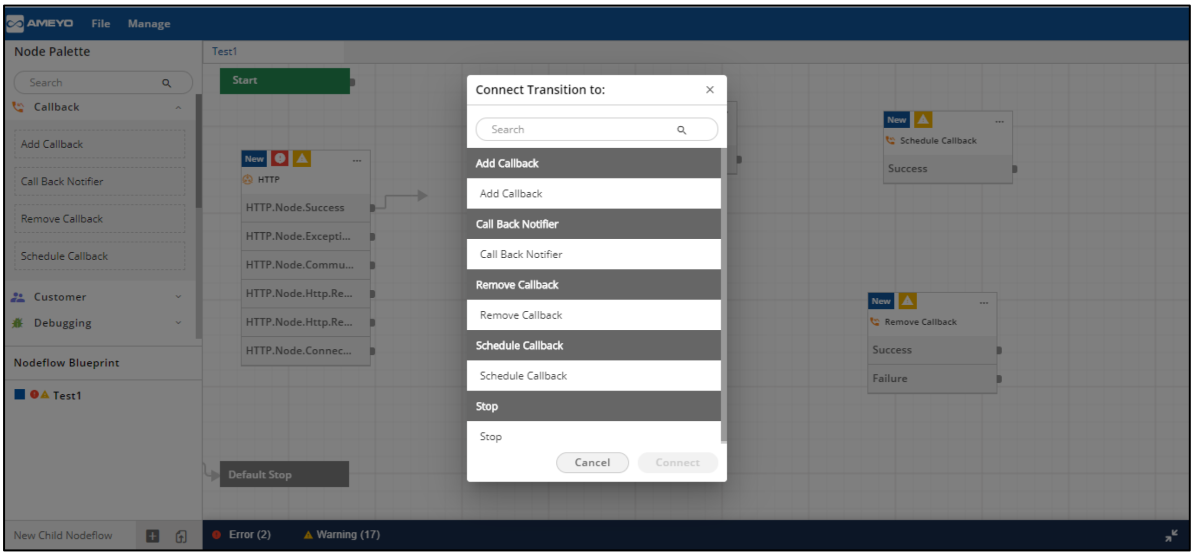


Figure: "Connect to" Modal

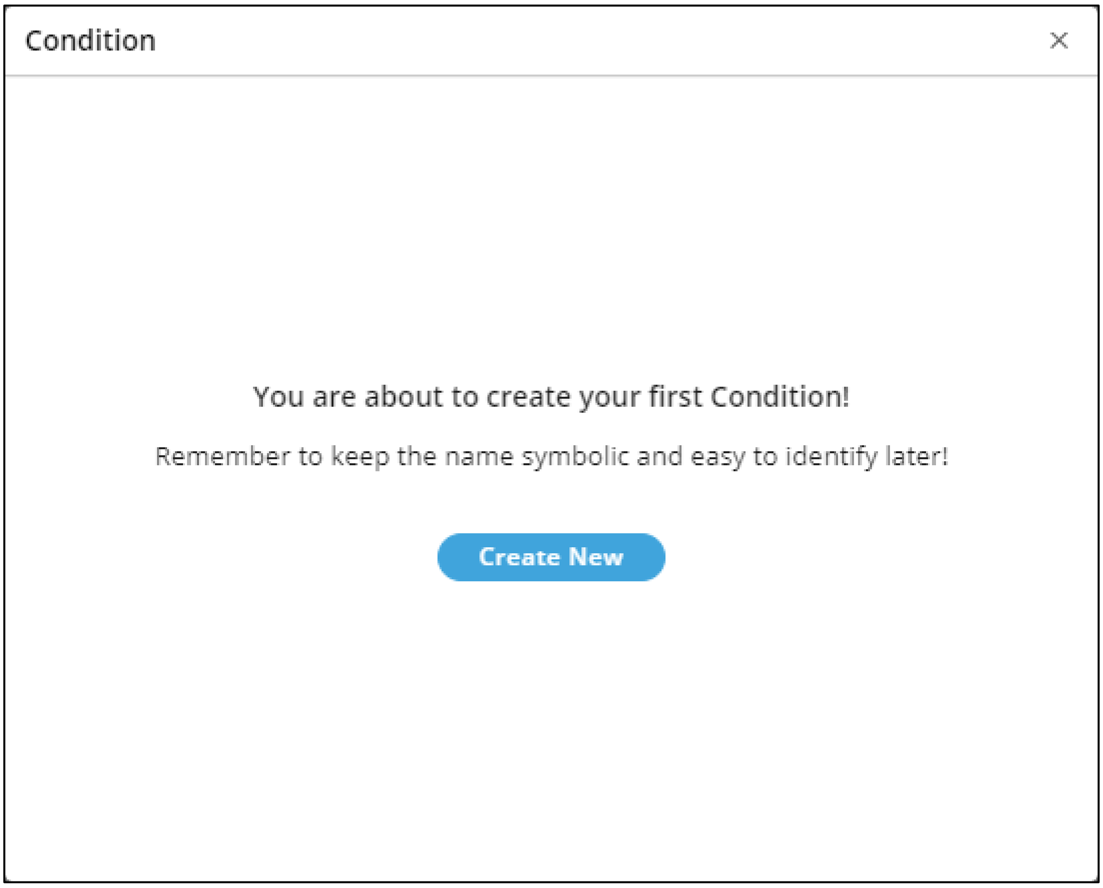
The User can select the target node to which the selected node has to be connected.

8.1.2 Reusable Conditions

If a condition has to be used in a different node, the Administrator has to perform the manual steps for doing the same. To make it easy, Ameyo introduced the reusability of conditions. Now, a Condition can be saved and reused anywhere in the Nodeflow like a Variable.

8.1.2.1.1 Reusable Conditions

In "Manage" menu, a new option, "Condition" has been added. Click this command to access "Condition" modal where the Administrator can create and manage the conditions.



Condition

You are about to create your first Condition!

Remember to keep the name symbolic and easy to identify later!

Figure: Condition Modal

If no condition is created, then it shows the message, "You are about to create your first Condition!".

The Administrator can click "Create New" to create a new condition. It shows the following modal.

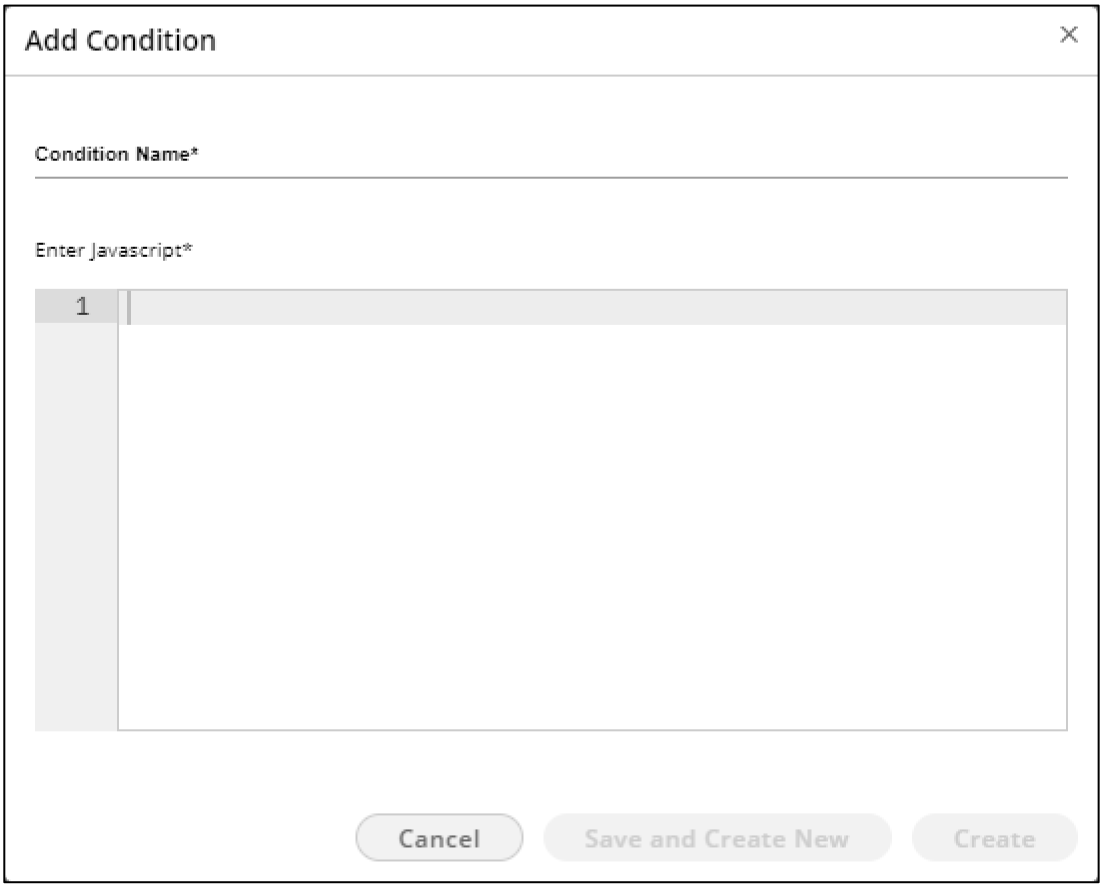
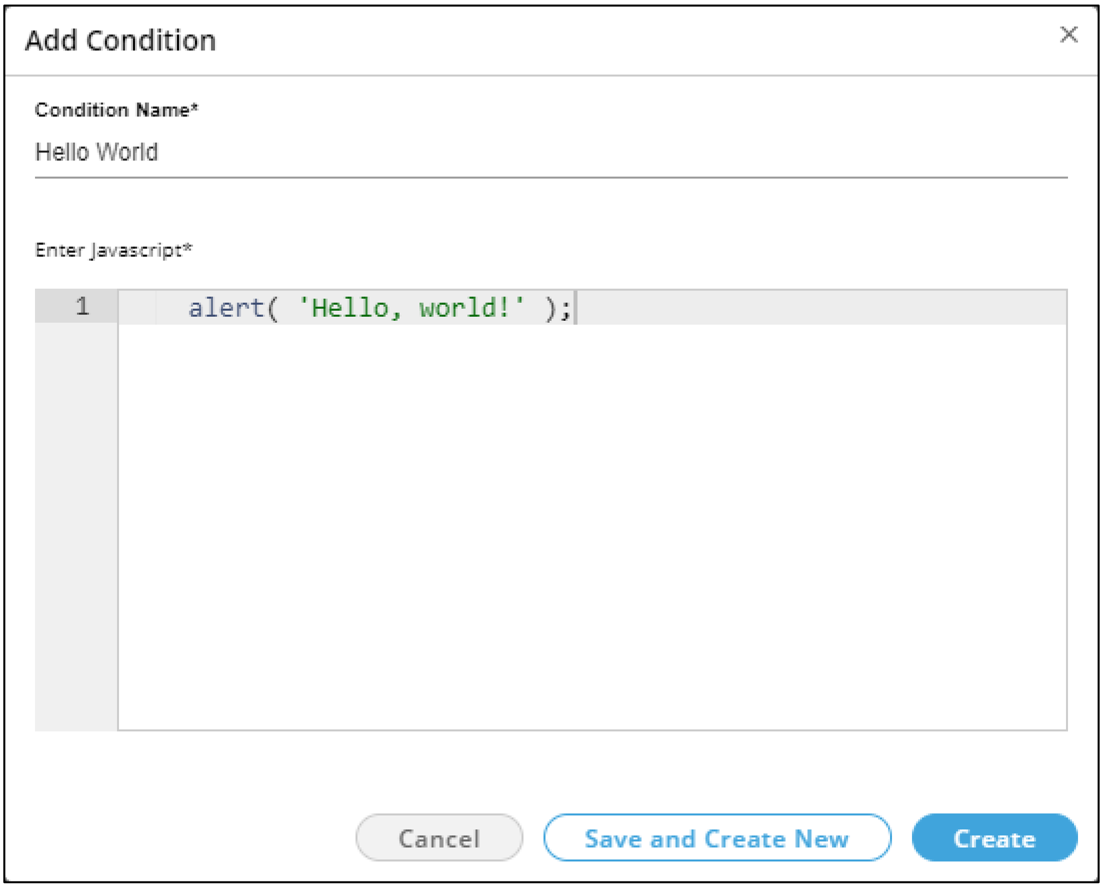


Figure: Create New Condition

Provide a name for the condition and add the JavaScript Code.



Condition Name*

Hello World

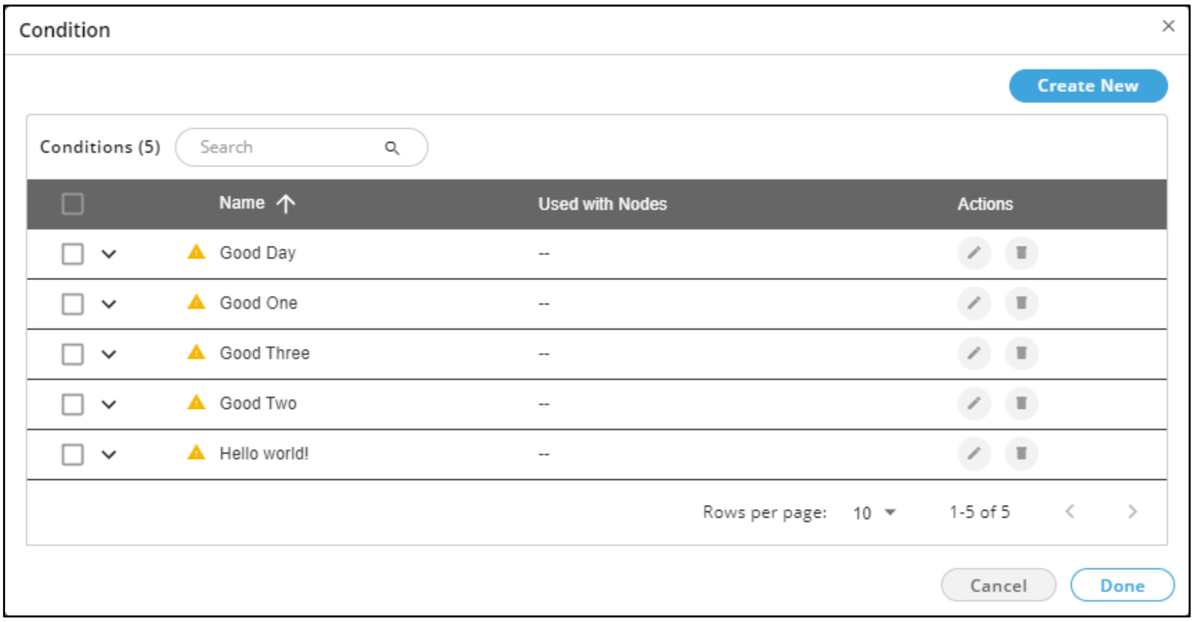
Enter Javascript*

```
alert( 'Hello, world!' );
```

Figure: Sample Condition

Click "Create New" button to create the node, whereas the Administrator can click "Save and Create New" to save this node and proceed to create another new condition.

After creating the nodes, "Condition" modal displays the created nodes.

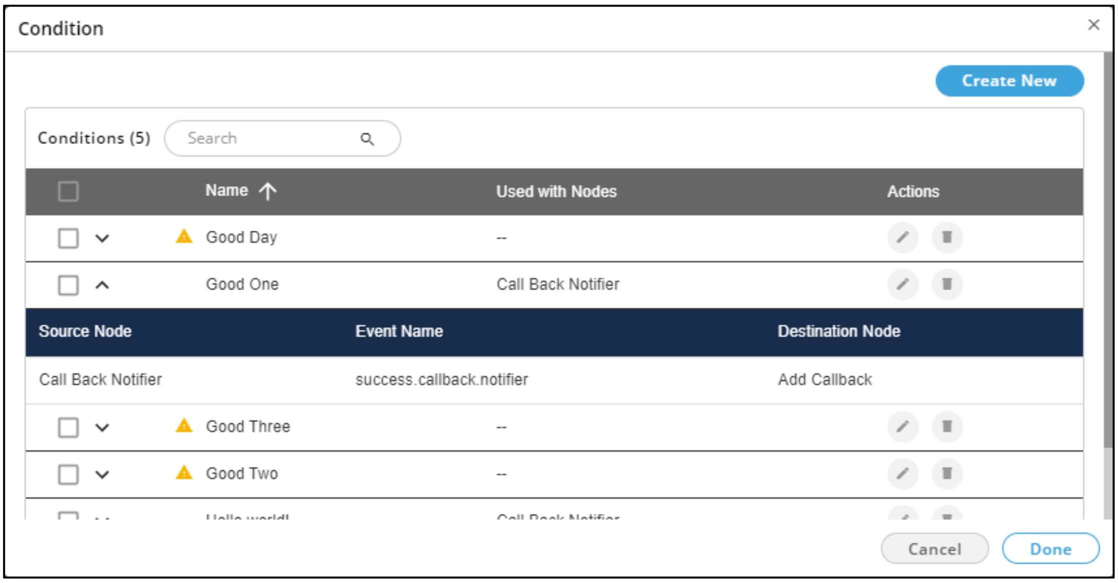


☐		Name ↑	Used with Nodes	Actions
☐	⌄	⚠ Good Day	--	✎ 🗑
☐	⌄	⚠ Good One	--	✎ 🗑
☐	⌄	⚠ Good Three	--	✎ 🗑
☐	⌄	⚠ Good Two	--	✎ 🗑
☐	⌄	⚠ Hello world!	--	✎ 🗑

Figure: List of Conditions

Here, the user can perform the following actions.

- **View the Details of Condition:** The User can expand a condition to view the list of nodes where this condition is being used.



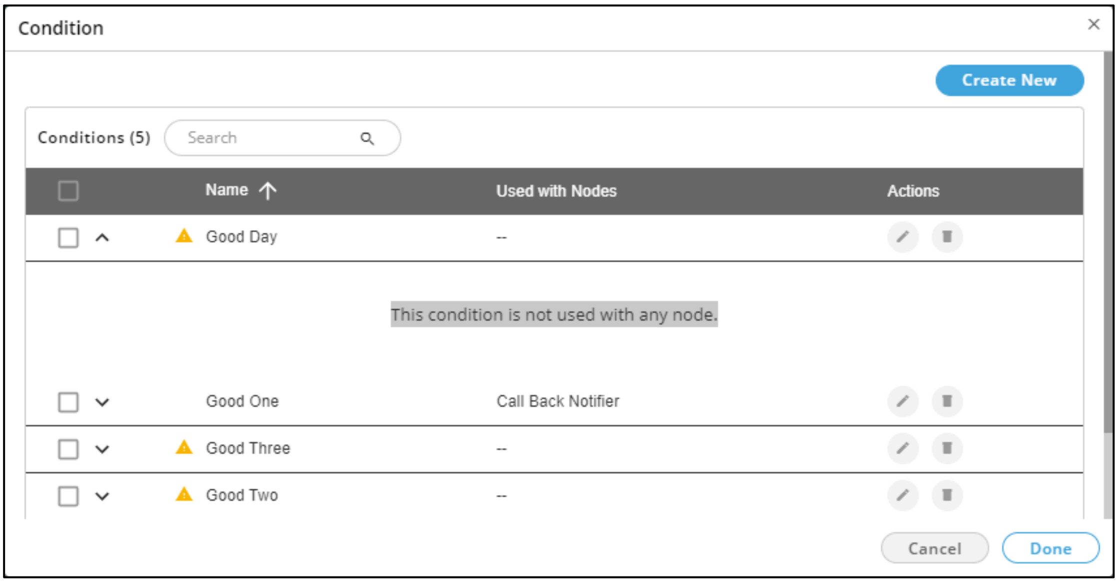
☐		Name ↑	Used with Nodes	Actions
☐	⌄	⚠ Good Day	--	✎ 🗑
☐	⌃	Good One	Call Back Notifier	✎ 🗑

Source Node	Event Name	Destination Node
Call Back Notifier	success.callback.notifier	Add Callback

☐		Name	Used with Nodes	Actions
☐	⌄	⚠ Good Three	--	✎ 🗑
☐	⌄	⚠ Good Two	--	✎ 🗑
☐	⌄	Hello world!	Call Back Notifier	✎ 🗑

Figure: Details of a Condition

If the condition is not being used in any node, then icon is displayed with that condition. When the User expands such a node, then a message is displayed.



☐	Name	Used with Nodes	Actions
☐	⚠ Good Day	--	✎ 🗑
	This condition is not used with any node.		
☐	Good One	Call Back Notifier	✎ 🗑
☐	⚠ Good Three	--	✎ 🗑
☐	⚠ Good Two	--	✎ 🗑

Figure: Condition is not being used.

- **Name on Transition:** The name of conditions is also displayed on the Transitions.

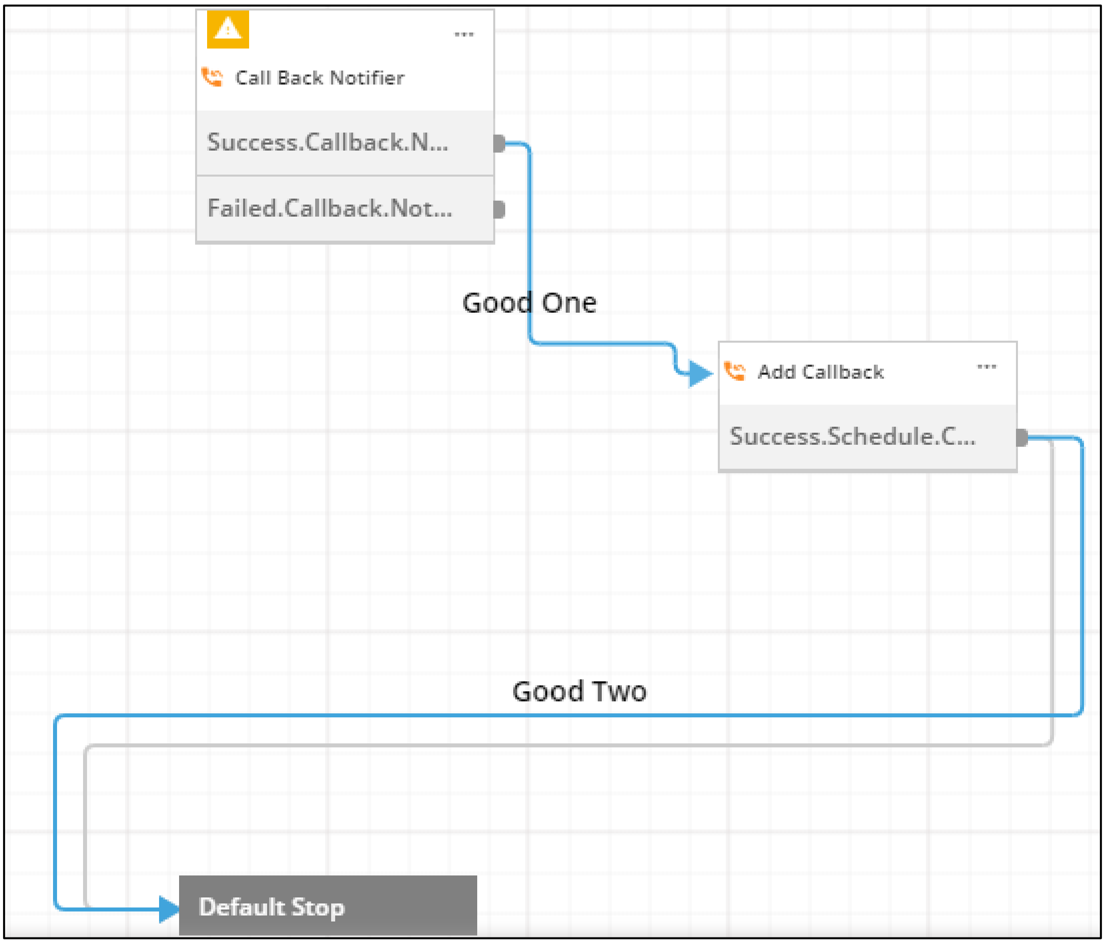
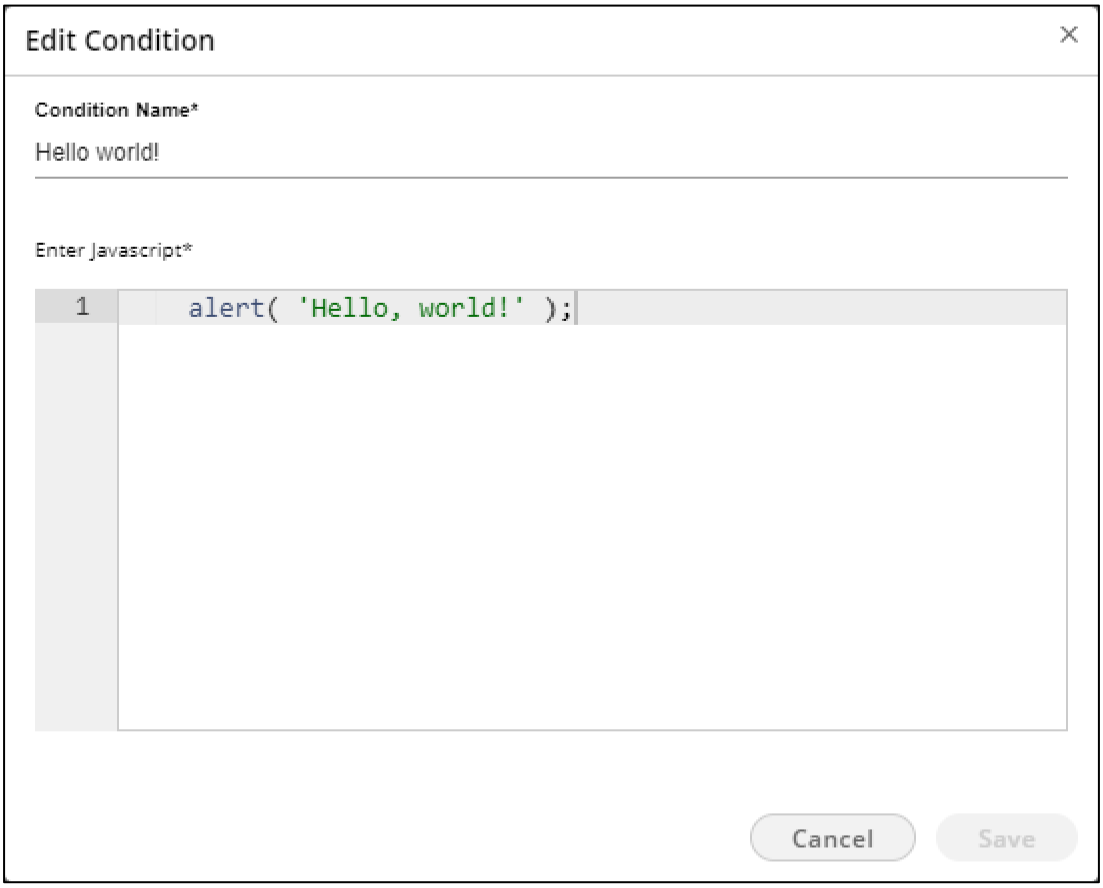


Figure: Conditions are being displayed on Transitions.

- **Pagination:** The User can select the number of conditions to be displayed on a page and navigate between the pages.
- **Edit the Condition:** Click icon to edit the condition using the following modal.



Condition Name*
Hello world!

Enter Javascript*

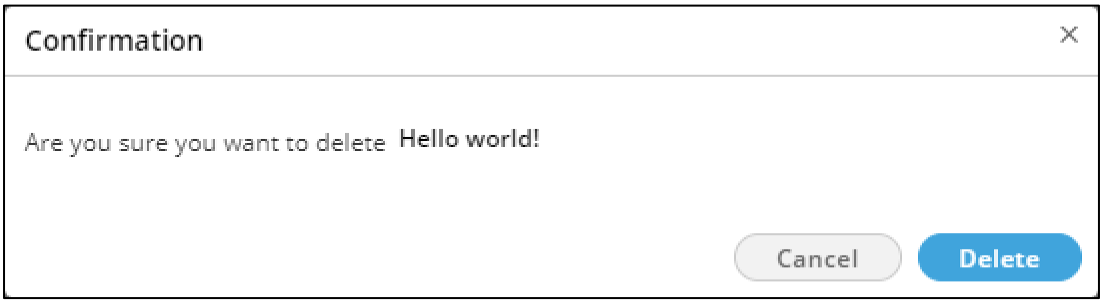
```
alert( 'Hello, world!' );
```

Figure: Edit a Condition

The User can edit the name and code. If the condition is being used with the transitions, then the following confirmation message is displayed.

Figure: Message to confirm the modification of a condition

- **Delete the Condition:** Click icon to delete the condition. It shows the following warning message before deleting the condition.



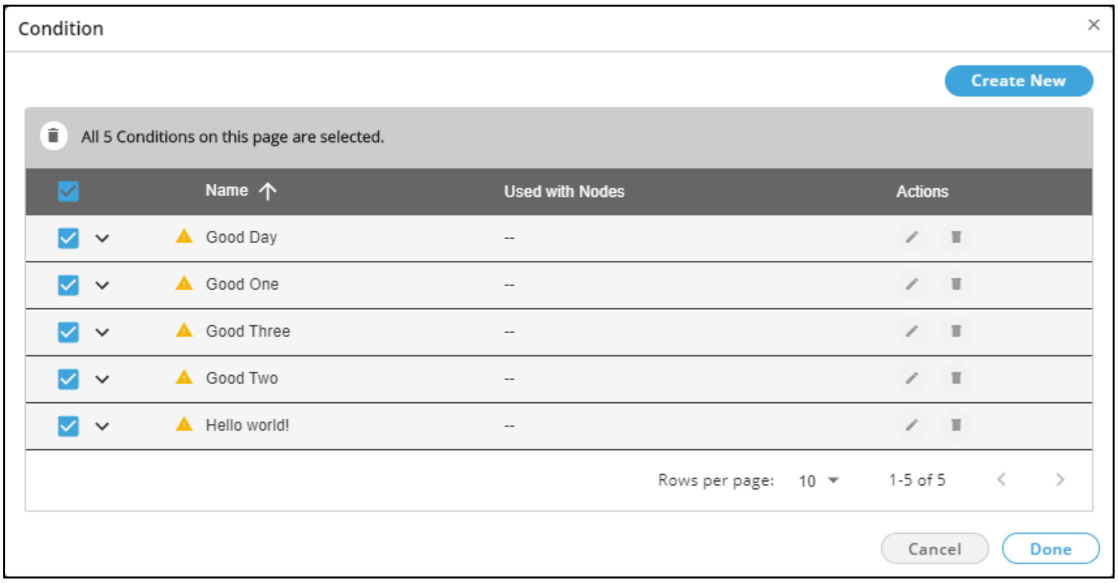
Confirmation

Are you sure you want to delete Hello world!

Figure: Warning before deleting a Condition

Click "Delete" to delete the condition.

- **Bulk Operation:** The User can click the checkbox in the header to select all conditions of a page collectively. The User can also select multiple conditions manually.



Condition

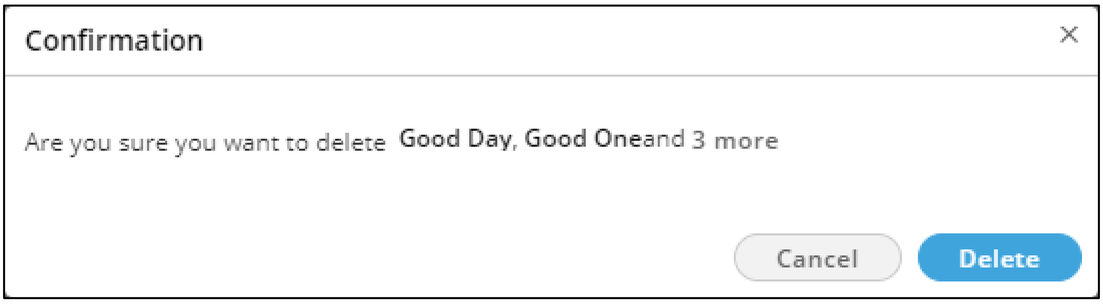
All 5 Conditions on this page are selected.

☑	Name ↑	Used with Nodes	Actions
☑	Good Day	--	
☑	Good One	--	
☑	Good Three	--	
☑	Good Two	--	
☑	Hello world!	--	

Rows per page: 10 1-5 of 5

Figure: Selecting all Conditions

The header now shows the option to delete all conditions collectively. The User can click icon to delete the conditions. It shows the following warning message.



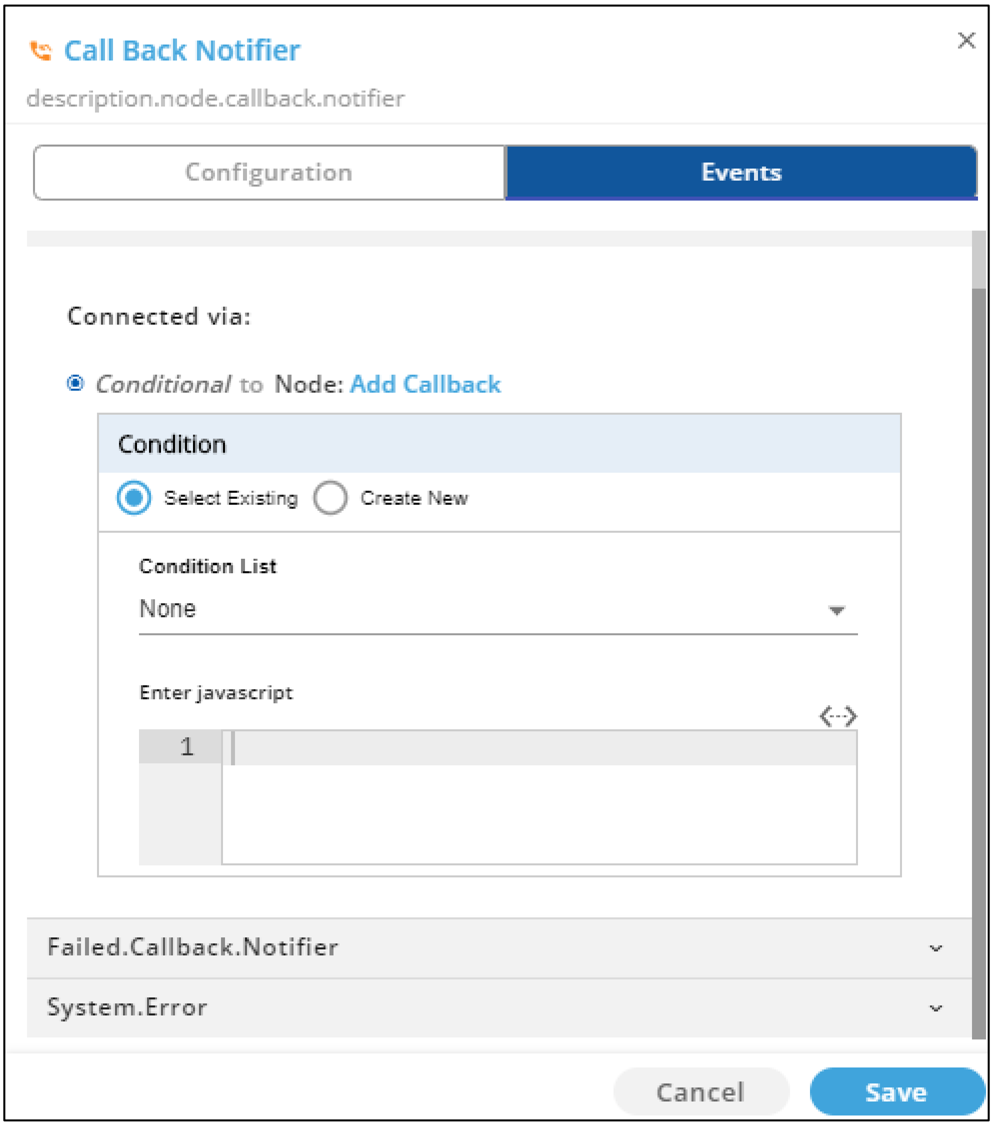
Confirmation

Are you sure you want to delete Good Day, Good Oneand 3 more

Figure: Warning before deleting Conditions

8.1.2.1.2 Conditions in Node Properties

"Events" tab of the Properties of a node is also modified. It provides two options: "Create New" or "Select Existing".



Call Back Notifier

description.node.callback.notifier

Connected via:

● *Conditional* to Node: Add Callback

Condition

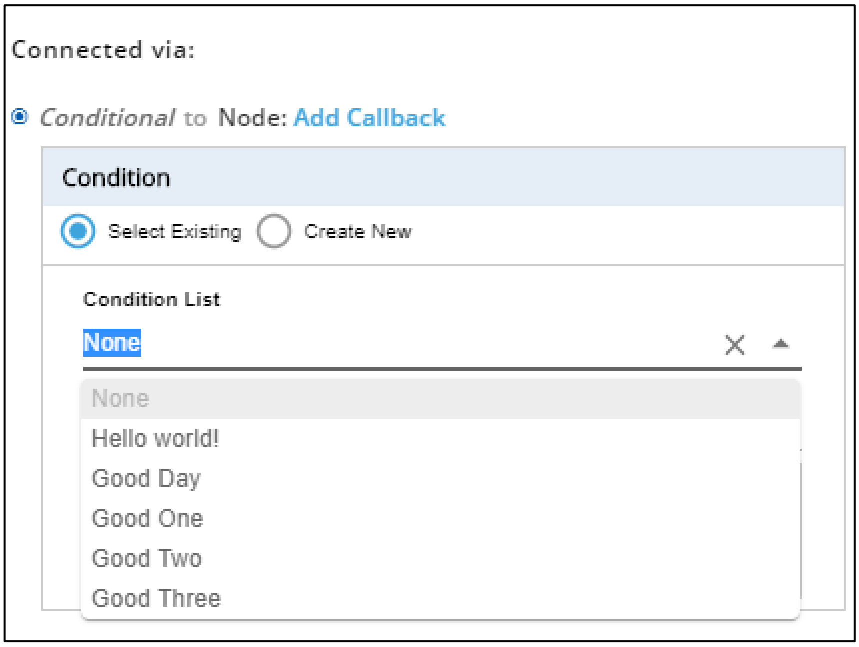
● Select Existing ○ Create New

Condition List: None

Enter javascript

Figure: "Events" Tab of the node properties

The User can select "Select Existing" to select a Condition in "Condition List" drop-down menu.



Connected via:

● *Conditional* to Node: Add Callback

Condition

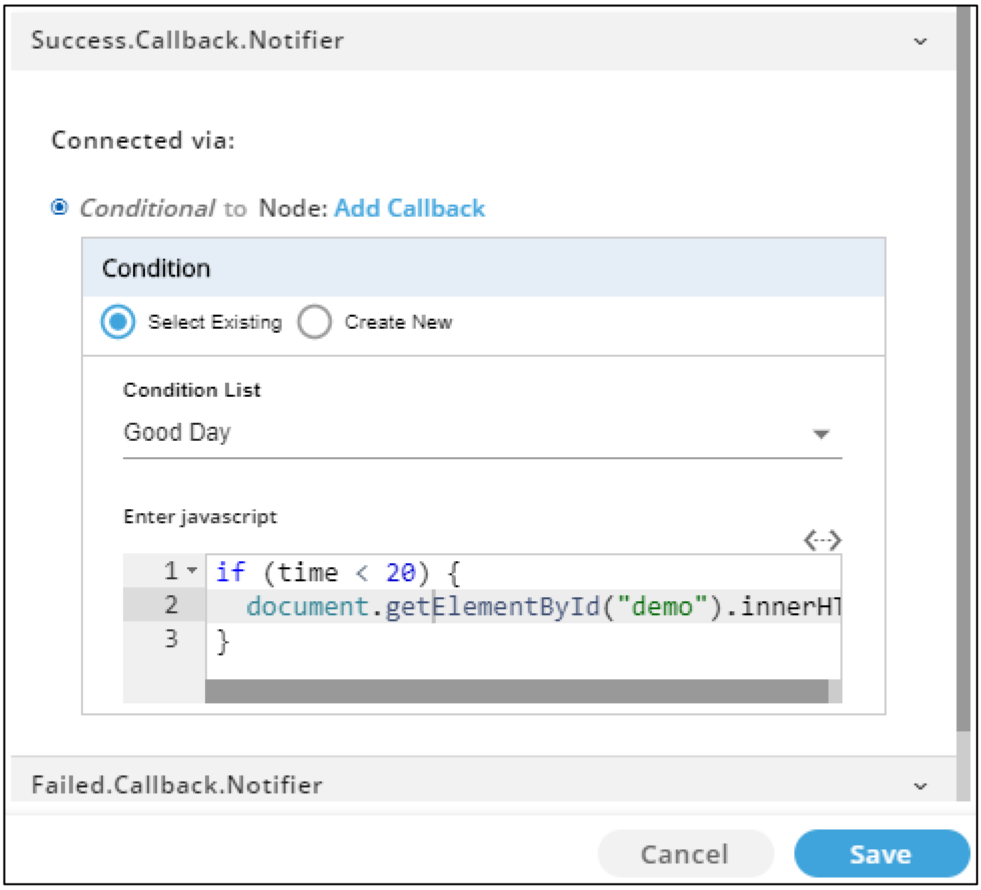
● Select Existing ○ Create New

Condition List

None

Figure: List of Conditions

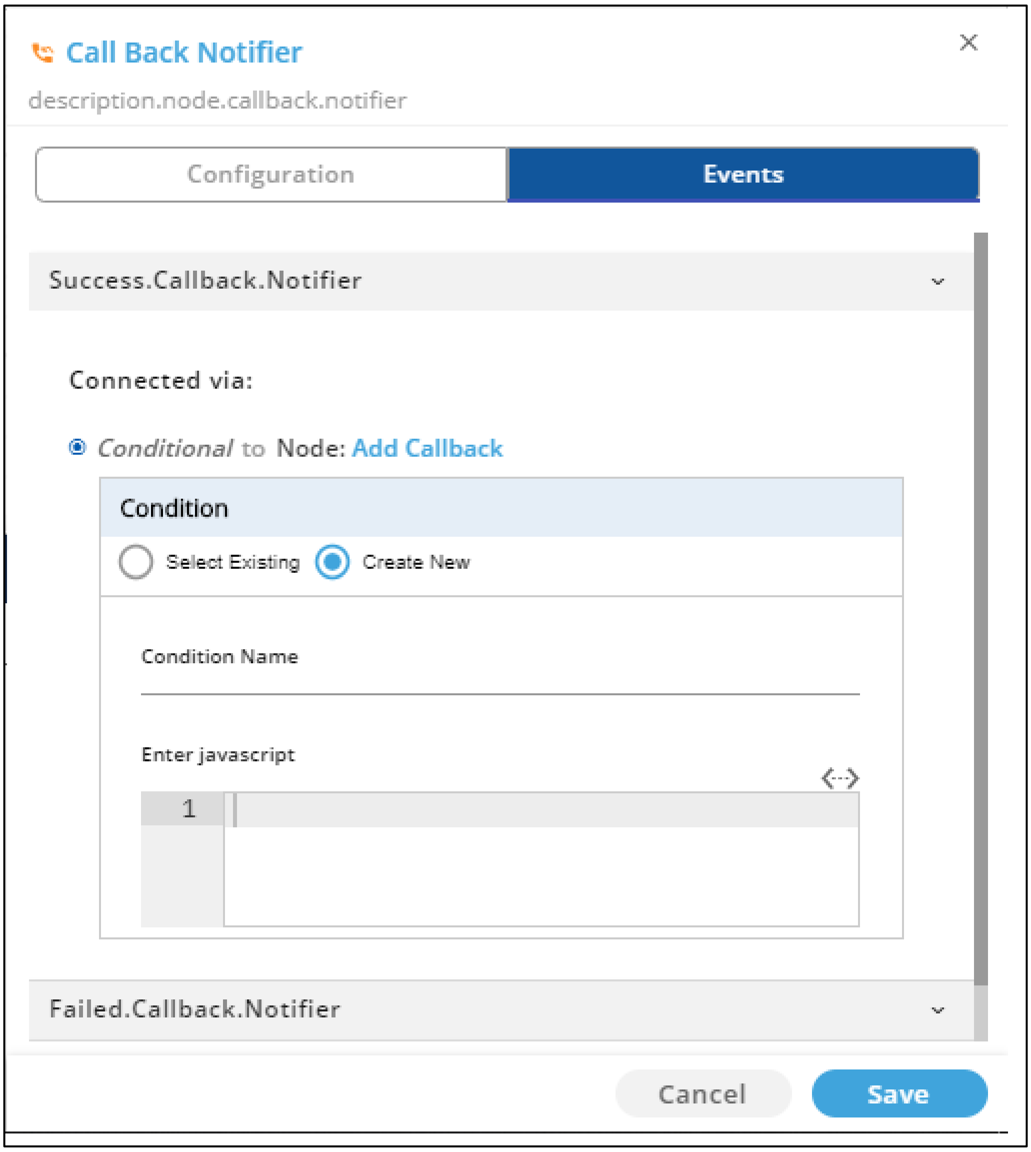
The code of the selected condition is displayed in "Enter JavaScript" text area where the user can edit it.



```
if (time < 20) {
  document.getElementById("demo").innerHT
}
```

Figure: Selected a Condition

The User can select "Create New" to create a new condition here. This newly created condition will also be available in "Manage Conditions" modal.



Call Back Notifier

description.node.callback.notifier

Success.Callback.Notifier

Connected via:

● *Conditional* to Node: Add Callback

Condition

○ Select Existing ● Create New

Condition Name

Enter javascript

Failed.Callback.Notifier

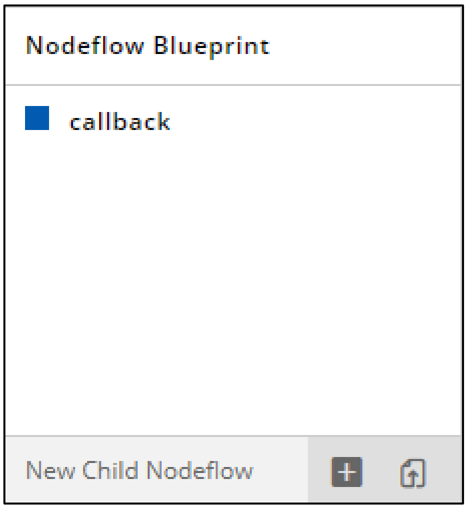
Figure: Create a new Condition

8.1.2.1.3 Conditions when the User exports the Nodeflow

When the user exports a nodeflow containing the conditions, all conditions and their links to transitions will also be saved and exported to ".anfx" file. The conditions will be available in "Manage Conditions" section on exporting, if a nodeflow has conditions.

9. Nodeflow Blueprint

Nodeflow Blueprint section is located under the Node Palette in the left pane.



Nodeflow Blueprint

🟦 callback

New Child Nodeflow

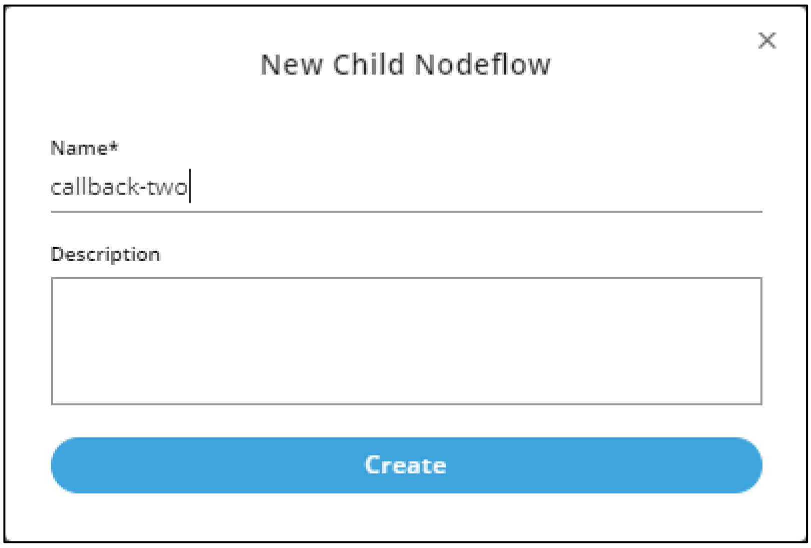
Figure: Nodeflow Blueprint

Here, the Administrator can see the name of the parent Nodeflow, create the Child Nodeflow, and manage the Child Nodeflows.

9.1 Create Child Nodeflow

Perform the following steps to create a Child Nodeflow.

1. In "Nodeflow Blueprint" section, click ⊞ icon for "New Childflow" to create a new Child Nodeflow. It shows the following pop-up.



New Child Nodeflow

Name*

callback-two

Description

Figure: Add Child Flow

2. Provide a name and description for the Child Nodeflow.

3. Click "Create" to create the Child Nodeflow. A new Child Nodeflow is created under the Nodeflow, and a new tab shows its canvas area.

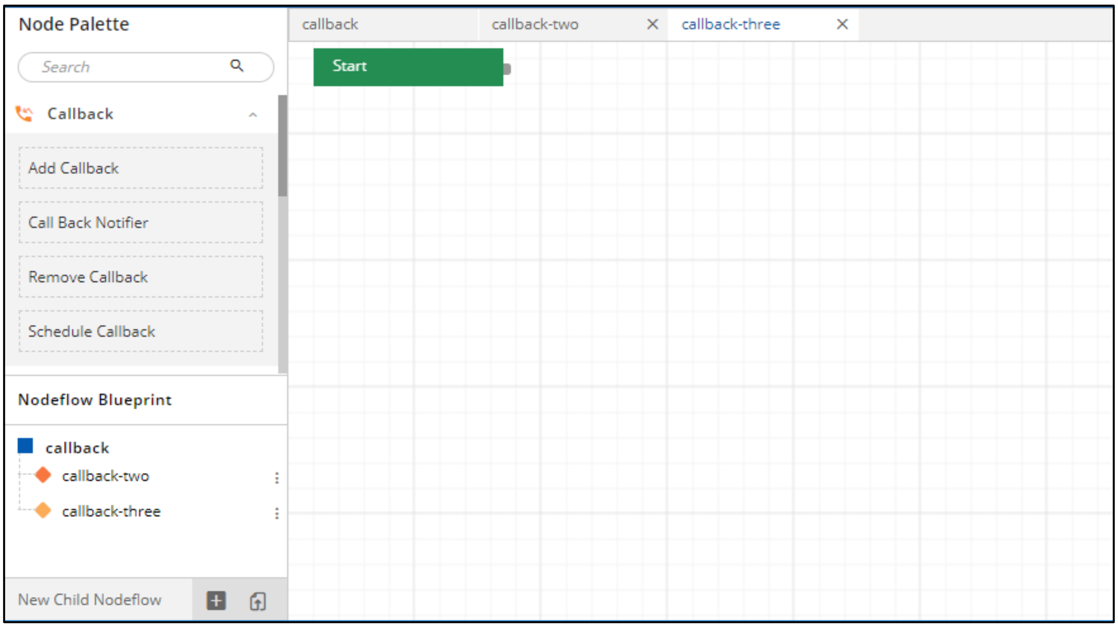


Figure: Created Child Nodeflow

9.2 Import Child Nodeflow

Now, the Child Nodeflows can now be imported from a ".anfx" or a ".nodeflow" file. If a user is importing a nodeflow that contains multiple child nodeflows, then the Synthesizer opens that nodeflows and import its child nodeflows.

Importing ".nodeflow" as child nodeflow will be a limited time functionality.

However, the functionality of importing ".anfx" files as child nodeflows will be available.

Perform the following steps.

1. The user has to click icon located on bottom in "Nodeflow Blueprint" section to import an existing ".anfx" or ".nodeflow" file as Child Nodeflow. It shows the following pop-up to open the file through the browser.

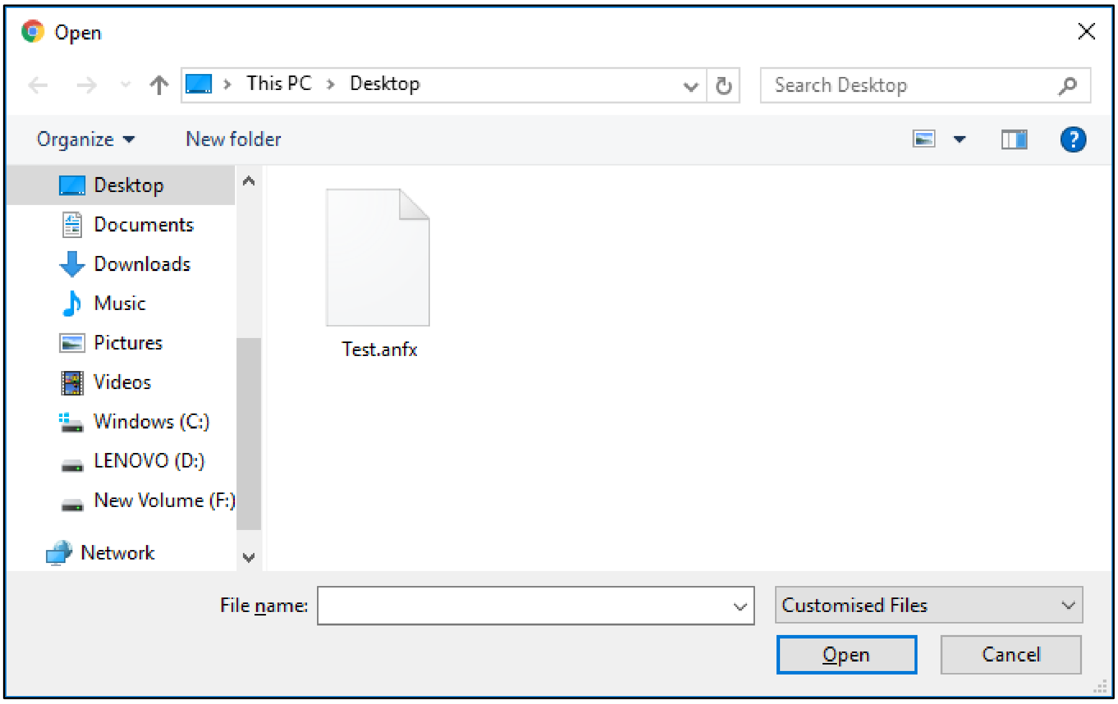
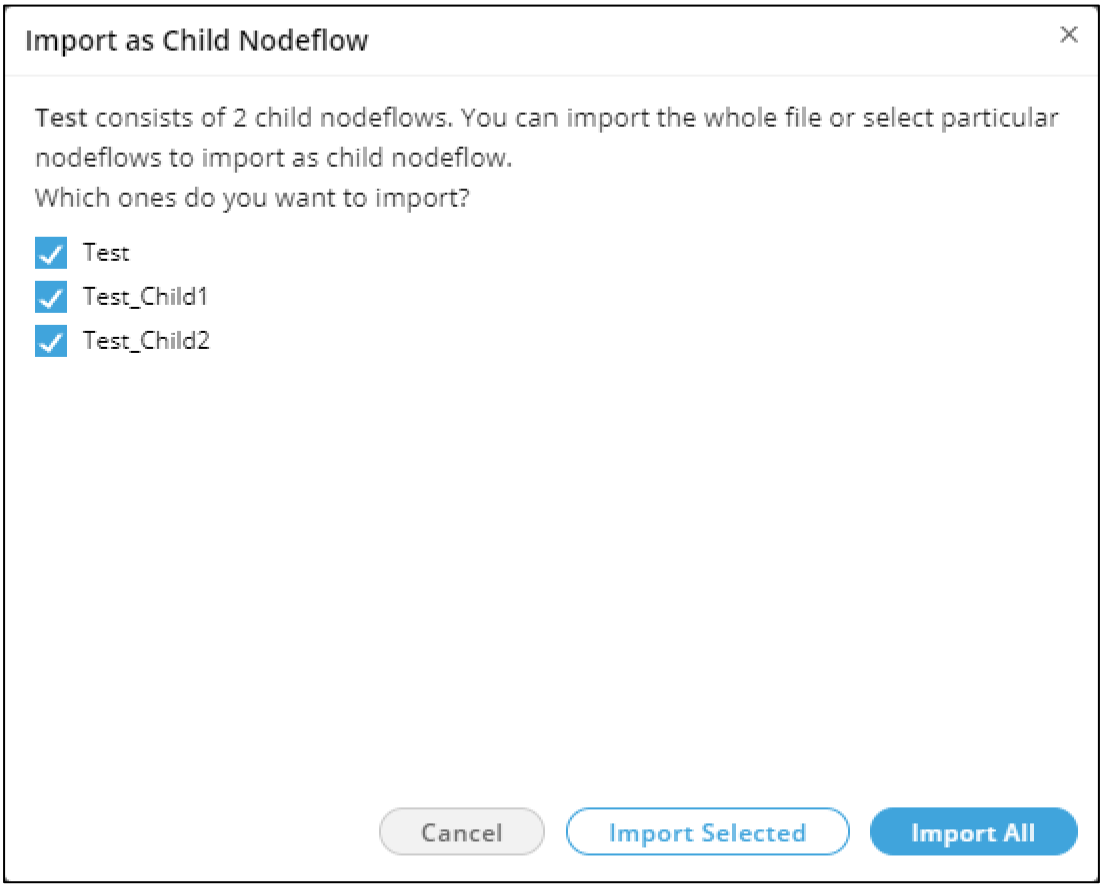


Figure: Import ANFX file

2. Select the location where the file is located, select the file, and click "Open". Synthesizer opens the file and displays the following pop-up.



Import as Child Nodeflow

Test consists of 2 child nodeflows. You can import the whole file or select particular nodeflows to import as child nodeflow.
Which ones do you want to import?

☑ Test
☑ Test_Child1
☑ Test_Child2

Figure: Select the Nodeflows

3. Select the nodeflows, which has to be imported, and click "Import Selected". If you want to import the nodeflow and its all child nodeflows from the selected file, click "Import All".

The Child Nodeflows in the selected file will also be listed as Child Nodeflow along with their parent after the import process.

After import, the child nodeflows will be listed in "Nodeflow Blueprint" section.

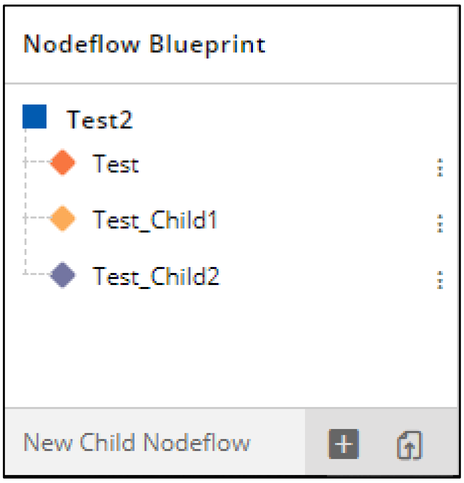


Figure: Imported the Nodeflows

If the child nodeflow, which is being imported, has the same name as that of an existing child nodeflow, then the Synthesizer will rename the target child nodeflow during import by adding "Renamed_" prefix.

9.3 Indicating Child Nodeflows

Child Nodeflows will now be indicated with a diamond-shaped icon. Each Child Nodeflow will have a unique colour for its icon. The Synthesizer manages the colour distribution for the child nodeflows.
If a nodeflow containing child nodeflows or an individual child nodeflow is exported to ".anfx" file, the color codes for the child nodeflows will not be stored in that ".anfx" file. If this exported file is imported in an existing nodeflow (containing child nodeflows), then the color codes for child nodeflows (from the imported file) may be different only if these colors are already taken by other child nodeflows.

9.4 Manage Child Nodeflows

Click ⋮ icon to access the menu.

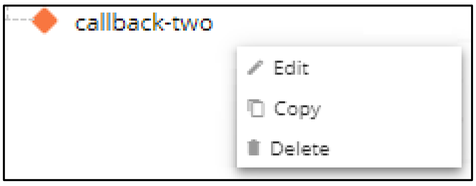


Figure: Menu Icon

It contains the following options.

- **Edit:** Click it to edit the Child Nodeflow. It opens the Child Nodeflow in a separate tab.
- **Copy:** Click it to create a copy of the selected Nodeflow. The copied Child Nodeflow is also created under the main Nodeflow.

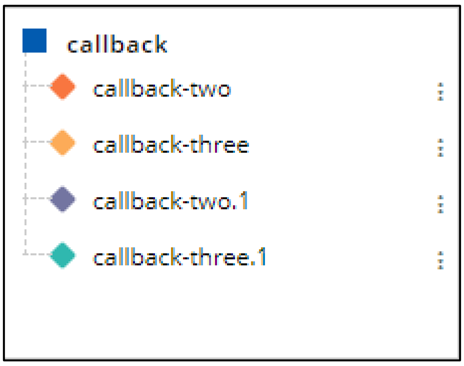
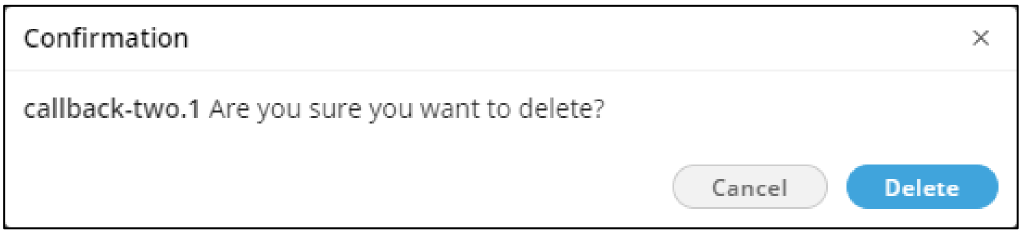


Figure: Copied Child Nodeflow

- **Delete:** Click it to delete the Child Nodeflow. It shows the following pop-up.


Confirmation

callback-two.1 Are you sure you want to delete?

Figure: Delete Child Nodeflow

The deleted Child Nodeflow cannot be restored.

Click "Delete" to delete the Child Nodeflow. Alternatively, you can click "Cancel" to not delete it.

10. List of All Nodes

10.1 List of All Nodes

Synthesizer contains the following nodes in different categories. While creating a Nodeflow, the Administrator has to configure the properties of these nodes. The configuration is simple. Follow the onscreen instructions to configure a node.

- Call Outcome Manipulation Node Category
  - Post Call Processing Node
- Authentication Node Category
  - OAuth Token Node
- Callback Node Category
  - Add Callback Node
  - Chained Interaction Node
  - Callback Notifier Node
  - Disposition Callback Node
  - Remove Callback Node
  - Schedule Callback
- Customer Node Category
  - Add Customer Node
  - Add Virtual Queue Request Node
  - Answering Machine Detection Node

    - Async Digit Collection Node
    - Automatic Speech Recognition Node
    - Busy Node
    - Check Time Zone Node
    - Check Virtual Queue Request Node
    - Click to Call Node
    - Customer Query Node
    - Customer Relationship Management Node
    - Digit Collection Node
    - Expected Automatic Call DistribSutor Wait Time Node
    - Music On Hold Node
    - Queue Position Node
    - Remove Virtual Queue Request Node
    - Start DTMF Node
    - Start DTMF and Logging Node
    - Stop DTMF Node
    - Upload and Churn Node
    - Voicemail Node
- Debugging Node Category
    - Debug Node
    - Nodeflow Info Node

    - Nodeflow Response Setter Node
    - Send Mail Node
- Distribution Node Category
    - Automatic Call Distributor Node
    - Automatic Call Distributor Web Service Node
    - Multi Automatic Call Distributor Node
    - Sync ACD (Automatic Call Distributor) Node
- Integration Node Category
    - Get Additional Meta Information Node
    - HTTP Node
    - Salesforce Node
    - Set Additional Meta Information for MRT Node
    - Simple DCOM Node
    - SQL Query Node
    - Web Service Node
- Interaction State Node Category
    - Barge Node
    - Resume Talk Node
    - Ringing Node
- Others Node Category
    - Aid Node

- Add Call Leg and Dial Node
- Add Chat Customer Node
- Block Node
- Chat Automatic Call Distributor Node
- Chat CRM Node
- Chat Hangup Node
- Chat Holiday Node
- Chat Office Hour Node
- Chat Send Message Node
- Chat Start Monitor Node
- Create Phone Call Runtime Node Model Node
- ChatC Customer Query Node
- Confer with Local IVR Node
- Conference with Bridge Application Node
- Control Playback Node
- Dispose Association Node
- Disposition CC Media Do Not Call Node
- Disposition CC Media Do Not Call Node Node
- Dial Node
- Echo Node
- Hangup Node

  - Holiday Node
  - Initialize Snoop Node
  - Make Call Node
  - Make Chat Node
  - Media Resource Control Protocol Playback Node
  - Modify Phone Node
  - Office Hour Node
  - Originate Call Runtime Node
  - Originate Chat Runtime Node
  - Pick Parked Call Node
  - Play Node
  - Play Text To Speech Node
  - Fetch Chat Logs Node
  - Script Node
  - Remove Phone Call Runtime Node Model Node
  - Snoop Node
- Recording Node Category
  - Listen Voicelog Node
  - Record to File Node
  - Record to Prompt Node
  - Start Media Resource Control Protocol Monitor Node

  - Start Monitor Node
  - Stop Monitor Node
  - Stop Media Resource Control Protocol Monitor Node
- Scrubbing Node Category
  - Disposition Campaign Customer Do Not Call Model Node
  - Disposition Campaign Media Do Not Call Model Node
  - Disposition Class Routing Node
  - Disposition Process Customer Do Not Call Node
  - Disposition Process Media Do Not Call Node
  - Do Not Call Node
- Storage Node Category
  - Additional Call History Node
  - Customer Data Node
  - Pack Node
  - Unpack Node
- System Node Category
  - Call Status Node
  - Modify Association Node
  - Notify Node
  - Send Notification Node
  - Sleep Node

    - Sync Child Flow Node
    - Wait Notify Node
- Terminate Node Category
    - Stop Node
- Transfer Node Category
    - Single Step Transfer Node
    - Transfer Node
    - Transfer to Campaign Node
    - Transfer to Phone Node
- User Node Category
    - Agent Availability Node
    - Answer Call Node
    - Associate User Prompt Node
    - Campaign Node
    - Find User to Supervise Node
    - Get Autocall Status Node
    - Get Channel-based Session Node
    - Get Ready Status Node
    - Login Node
    - Logout Node
    - Multi Originate Node

- Set Autocall Status Node
- Set Ready Status Node
- Whisper Call Node

10.2 Add Callback Node

It is used to add a scheduled callback (through IVR), which means when the callback will be made to the customer at the specified time from the already defined campaign and queue. The callback can be added in the same campaign, it cannot be done from the separate campaign. It is used to initiate a call back through the IVR itself.

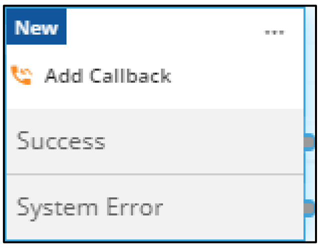
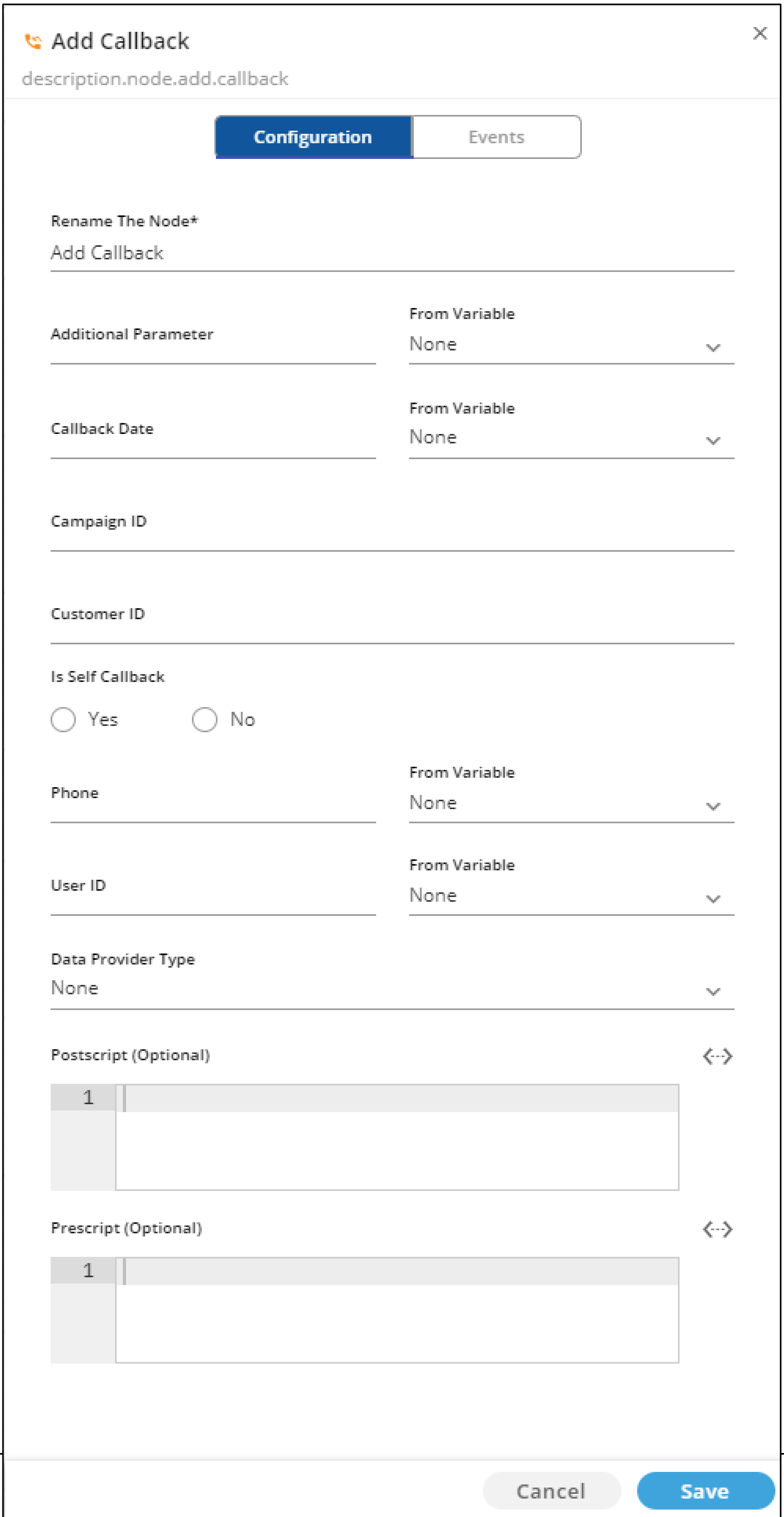


Figure: Add Callback Node

Following is a screenshot of its configuration.



Add Callback

description.node.add.callback

Rename The Node*
Add Callback

Additional Parameter
From Variable
None

Callback Date
From Variable
None

Campaign ID

Customer ID

Is Self Callback
○ Yes ○ No

Phone
From Variable
None

User ID
From Variable
None

Data Provider Type
None

Postscript (Optional)

```
1
```

Prescript (Optional)

```
1
```

Figure: Configuration of Add Callback Node

10.2.1 Availability in Nodeflow Types

This node is available in the following Nodeflow types.

- Callback
- Customer-based Inbound
- Dialing
- Inbound
- Post Call Processing

10.2.2 Configuration

Perform the following steps to configure this node.

1. **Rename Node:** Rename the node, if required.
   Provide an easy-to-recall name, which matches the default node name.

2. **Additional Parameter:** In some scenarios, a variable has to be fetched from the Ameyo System or it has to be declared in the JavaScript Code. To provide a value(7503908999) to the node attribute, either provide the value manually or select the value from the drop-down menu (that contains the list of the already defined variables) in Additional Parameter.

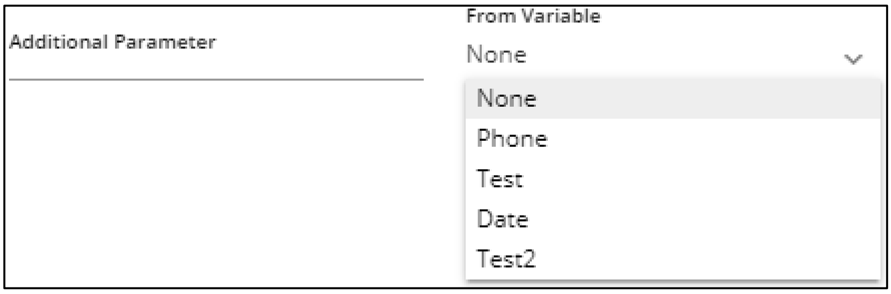
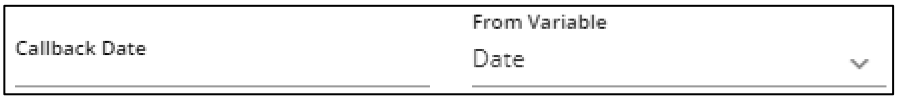


Figure: Additional Parameter

If the value of the additional parameter is being provided in both text field and drop-down menu, then the variable selected in the drop-down menu will have more priority and will be used by default.

3. **Callback Date:** To provide the callback date, either provide a name for its variable manually in the text field or select its variable name from the drop-down menu (that contains the list of variables). The value of Callback Date will be either fetched from the Ameyo System or the Administrator has to provide it manually in the JavaScript Code.



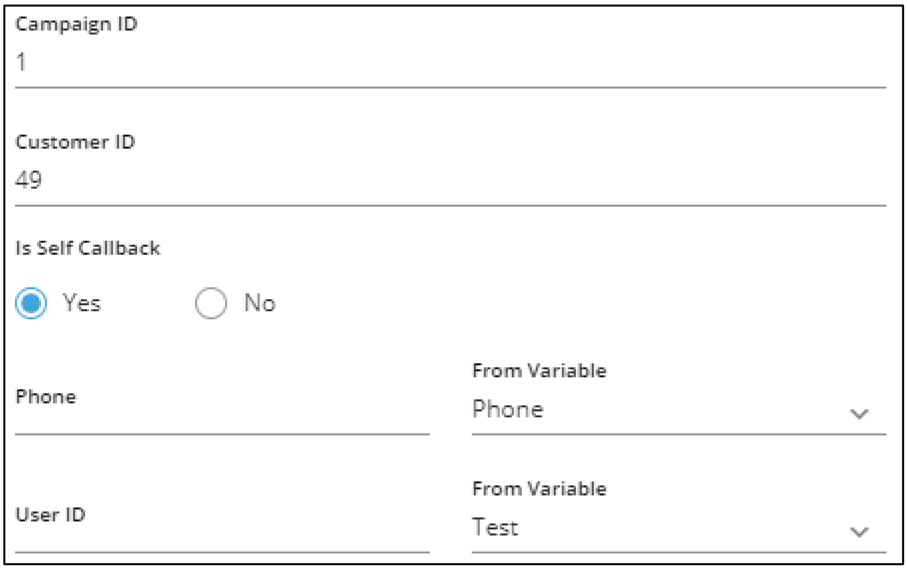
Callback Date

From Variable: Date

Figure: Select "Callback Date" Variable name

Even if the value of the callback data variable is being provided in both the text field and the drop-down menu, still the variable selected in the drop-down menu will have more priority and will be used by default.

4. **Campaign ID:** Enter the Campaign ID in numeric format.
5. **Customer ID:** Enter the Customer ID in numeric format.



Campaign ID
1

Customer ID
49

Is Self Callback
● Yes ○ No

Phone — From Variable: Phone

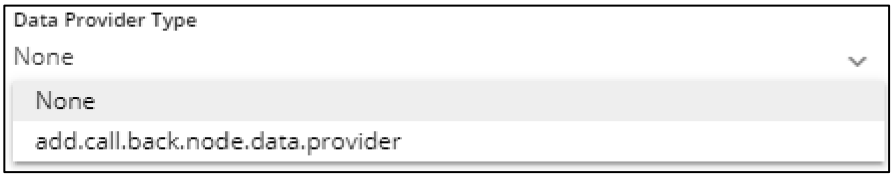
User ID — From Variable: Test

Figure: Other Options

6. **Is Self Callback:** Select "Yes" or "No" for the self callback option.
7. **Attribute.Phone:** To link "Phone" with the node, either provide a name for its variable manually in the text field or select its variable name from the drop-down menu (that contains the list of variables). The value of "Phone" will be either fetched from the Ameyo System or the Administrator has to provide it manually in the JavaScript Code.
8. **User ID:** To link "User ID" with the node, either provide a name for its variable manually in the text field or select its variable name from the drop-down menu (that contains the list of variables). The value of "User ID" will be either fetched from the Ameyo System or the Administrator has to provide it manually in the JavaScript Code.

   Even if the value of a variable is being provided in both the text field and the drop-down menu, still the variable selected in the drop-down menu will have more priority and will be used by default.

9. **Data Provider:** This node contains the following data provider. Select it.



Data Provider Type

None

None

add.call.back.node.data.provider

Figure: Add Callback Node Data Provider

- add.call.back.node.data.provider: It is the default data provider of this node.

10. **Postscript (Optional):** Synthesizer allows to run a JavaScript code after the processing of the node.

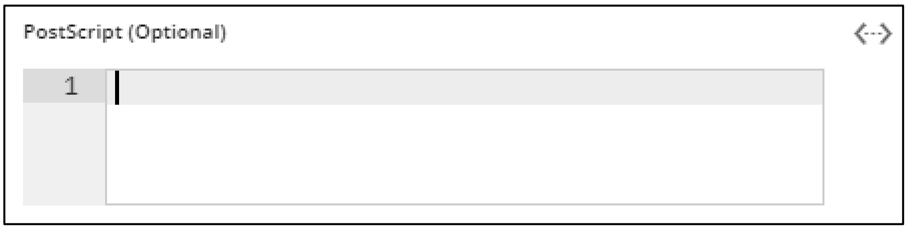
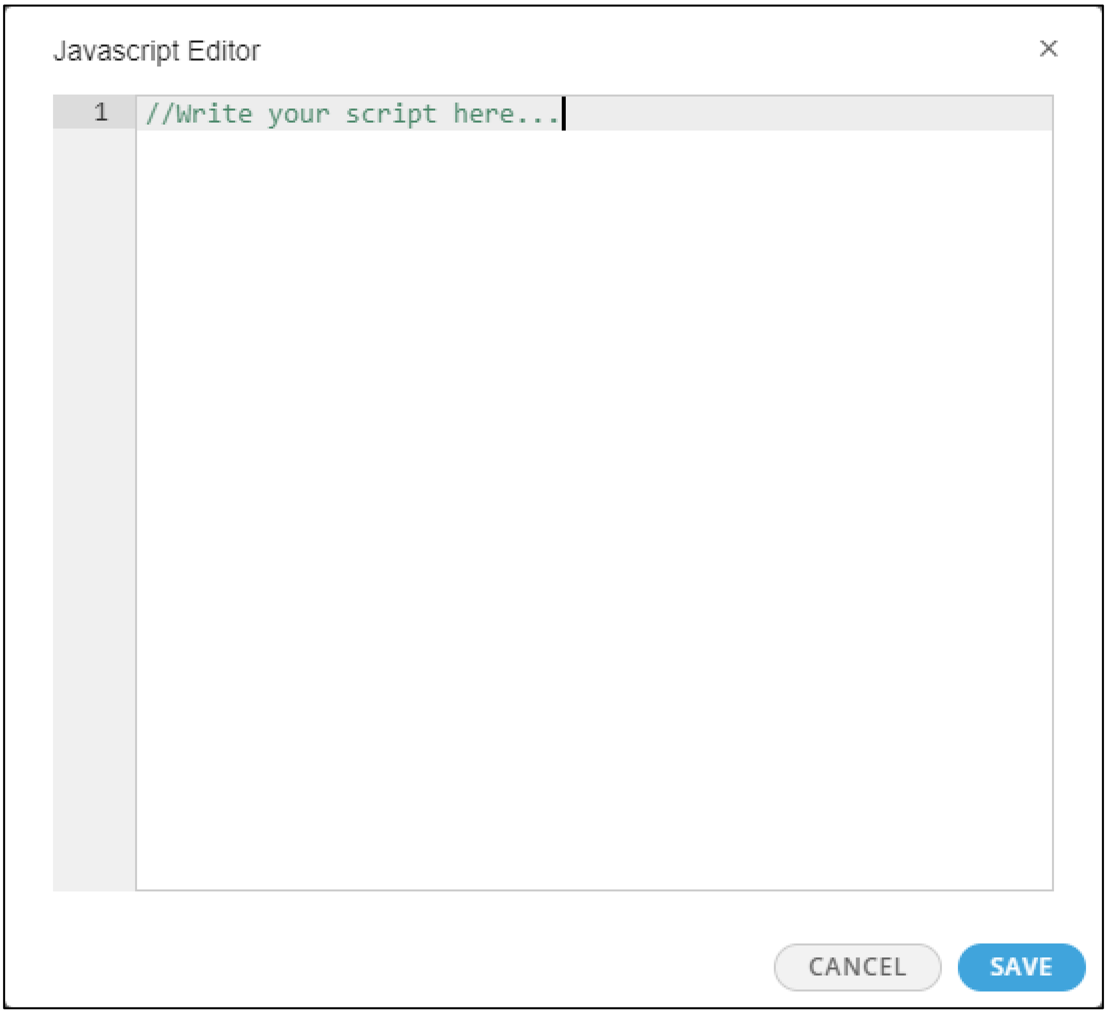


Figure: Script to run after processing "Add Callback" Node

Here, paste the JavaScript. Click <--> icon to open the following JavaScript Editor.



```
//Write your script here...
```

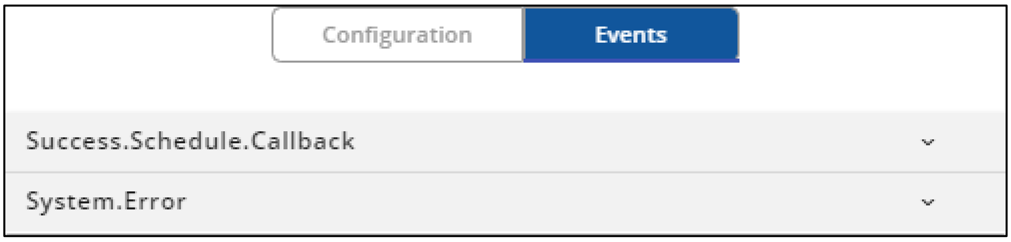
Figure: JavaScript Editor

The Administrator can write its own code here and click "Save". It returns to node configuration, which will show the saved code.

11. **PreScript (Optional):** Specify to run a custom JavaScript code before processing the node. Like PostScript, add the code in Prescript.
12. Click "Save" to save the Node Configuration.

10.2.3 Events

Switch to "Events" tab to see the events of this node.



Configuration | Events

Success.Schedule.Callback

System.Error

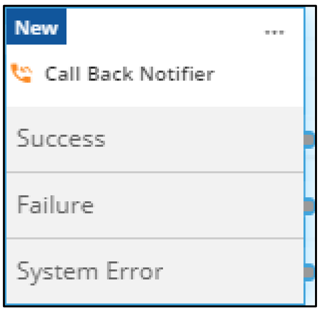
Figure: Events of Add Callback Node

These nodes are listed hereinbelow.

- **Event.Success.Schedule.Callback:** It is generated when the callback was successfully scheduled.
- **Event.System.Error:** It is generated when the system generates certain errors such as hardware, environmental, and others.

10.3 Callback Notifier Node

It is used to notify the callback to the assignee agent to whom the callback has been assigned.



New

Call Back Notifier

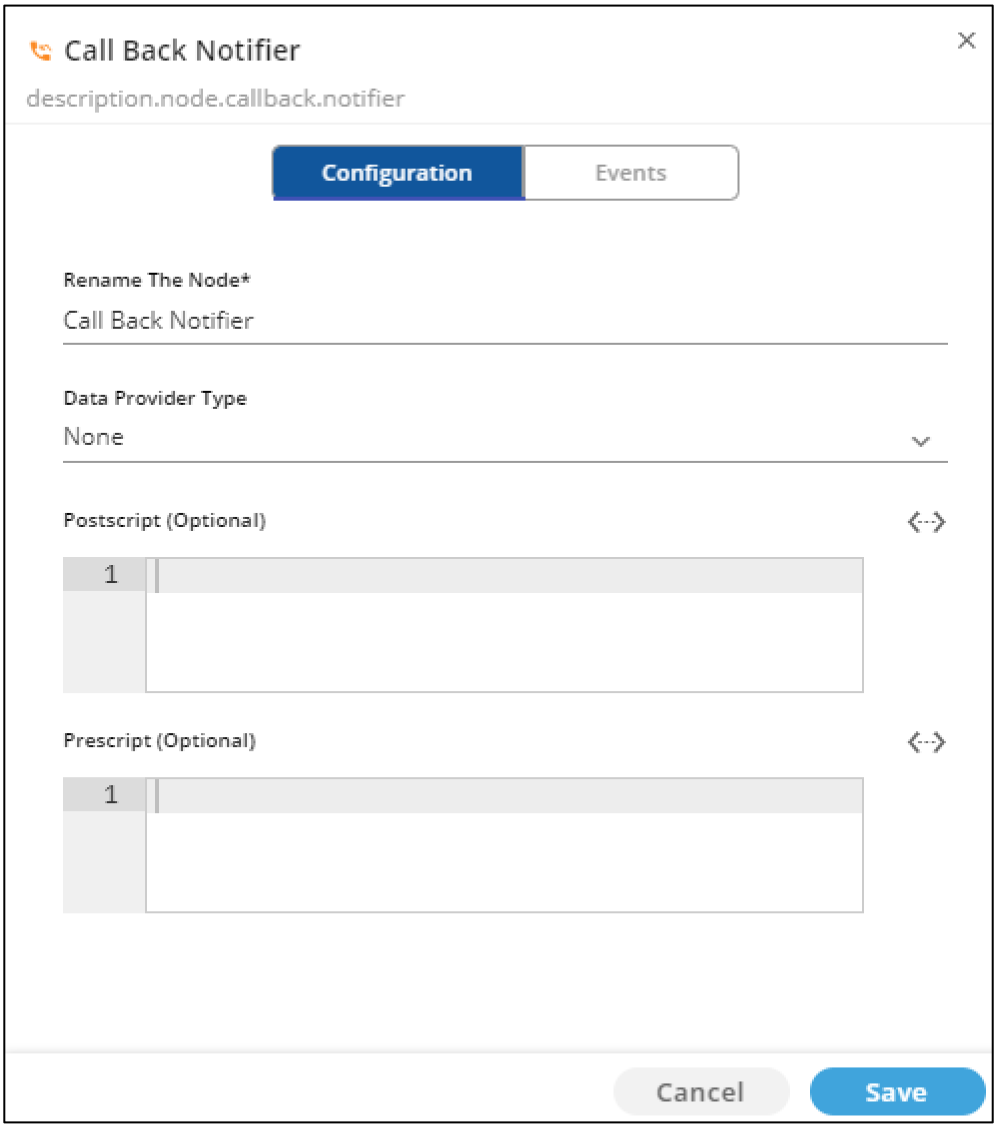
Success

Failure

System Error

Figure: "Callback Notifier" Node

Following is the screenshot of its configuration.



Call Back Notifier

description.node.callback.notifier

Rename The Node*: Call Back Notifier

Data Provider Type: None

Postscript (Optional)

Prescript (Optional)

Figure: Configuration of "Callback Notifier" Node

10.3.1 Availability in Nodeflow Types

This node is available in the following Nodeflow types.

- Callback
- Customer-based Inbound
- Dialing
- Inbound

10.3.2 Configuration

Perform the following steps to configure this node.

1. **Rename The Node:** Rename the node, if required.

   Provide an easy-to-recall name, which matches the default node name.

2. **Data Provider:** This node does not contain any data provider.

3. **Postscript (Optional):** Synthesizer allows to run a JavaScript code after the processing of the node.

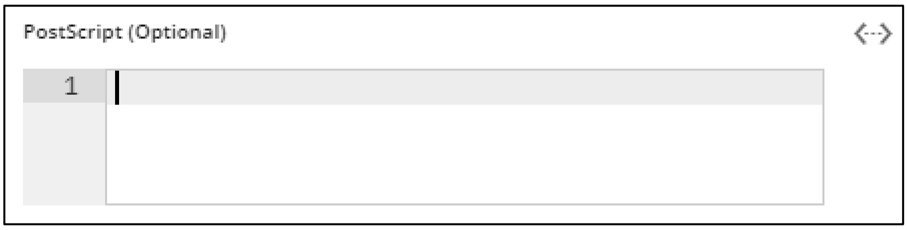
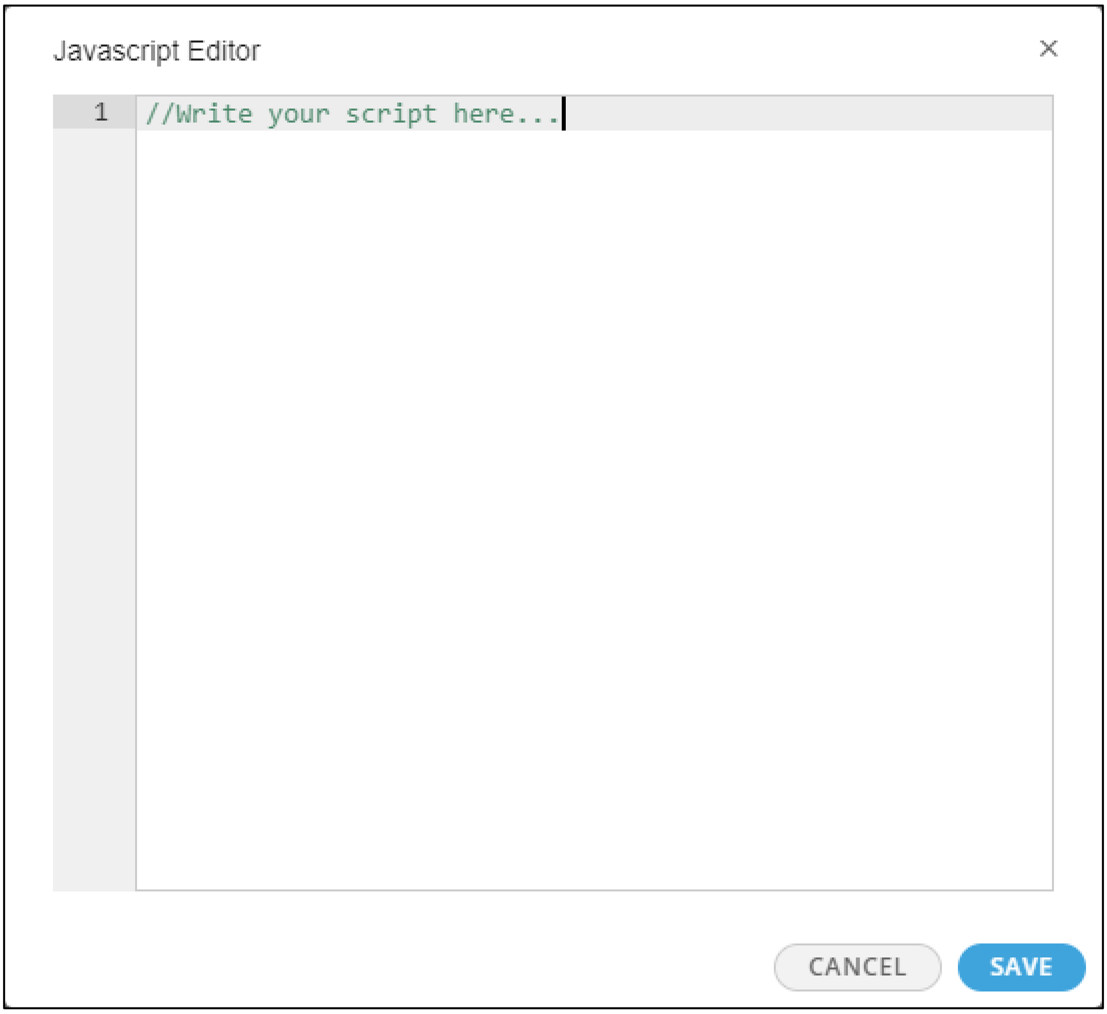


Figure: Script to run after processing "Callback Notifier" Node

Here, paste the JavaScript. Click <--> icon to open the following JavaScript Editor.



```
//Write your script here...
```

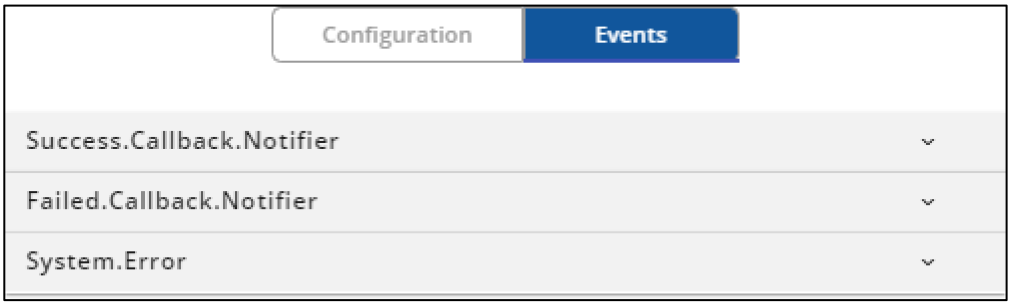
Figure: JavaScript Editor

The Administrator can write its own code here and click "Save". It returns to node configuration, which will show the saved code.

4. **PreScript (Optional):** Specify to run a custom JavaScript code before processing the node. Like PostScript, add the code in Prescript.
5. Click "Save" to save the Node Configuration.

10.3.3 Events

Switch to "Events" tab to see the events of this node.



Success.Callback.Notifier

Failed.Callback.Notifier

System.Error

Figure: Events of Callback Notifier Node

These events are listed hereinbelow.

- **Event.Success.Callback.Notifier:** It is generated when the callback has been notified to the assignee agent to whom the callback has been assigned, that is, sendUnprivilegedPushToSession is successful.
- **Event.Failed.Callback.Notifier:** It is generated when the callback cannot be notified to the assignee agent to whom the call back has been assigned because of any of the following reasons.
  - Not able to get the USER_SESSION_ID or USER_CRT_OBJECT_ID, callBackRequestId (as NODE_FLOW_REQUEST_ID beacuse nodeflowid not set)in node from the nodeflow variables.
  - Unsuccessful Push due to Can not find adapter for Entity,
- **Event.System.Error:** It is generated in any of the following cases.
  - Invalid node model
  - Invalid data in the Node Model
  - When the system generate certain errors such as hardware, environmental, and others.

10.4 Customer Query Node

It is used to fetch the customer information from the Ameyo database. It finds out if the customer exists in the Ameyo database or not. If found, it will add customer info to Call Run Time (CRT) Object.

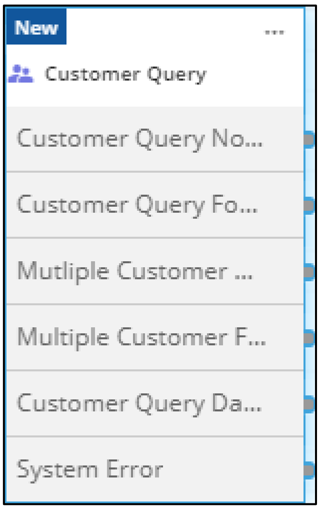
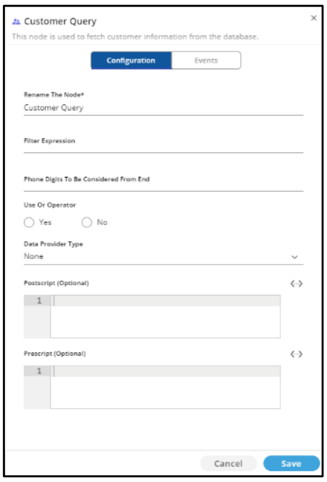


Figure: "Customer Query" Node

Following is a screenshot of its configuration.



This node is used to fetch customer information from the database.

Rename The Node*
Customer Query

Filter Expression

Phone Digits To Be Considered From End

Use Or Operator
○ Yes ○ No

Data Provider Type
None

Postscript (Optional)

Prescript (Optional)

Figure: Configuration of "Customer Query" Node

10.4.1 Availability in Nodeflow Types

This node is available in the following Nodeflow types.

- Callback
- Customer-based Inbound
- Conference
- Dialing

- Inbound
- Interaction
- Manual Dial
- Post Call Processing

10.4.2 Configuration

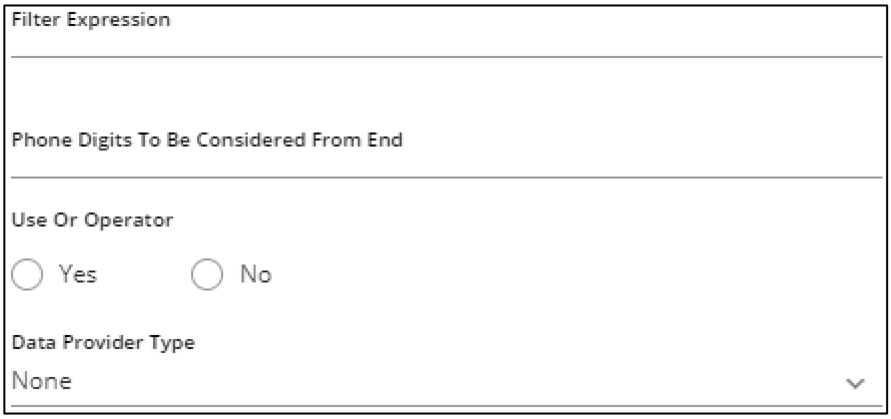
Perform the following steps to configure this node.

1. **Rename The Node:** Rename the node, if required.

   Provide an easy-to-recall name, which matches the default node name.

2. **Filter Expression (Optional):** It is an optional variable, whose acceptable value type is "String". It gives the expression based upon which the customers are retrieved from the database. Upon processing, the provided "String" type value is converted into a map having type Map<String,String>.

   Sample Values: "key1=value1, key2=value2, key3=value3"



Filter Expression

Phone Digits To Be Considered From End

Use Or Operator

○ Yes ○ No

Data Provider Type

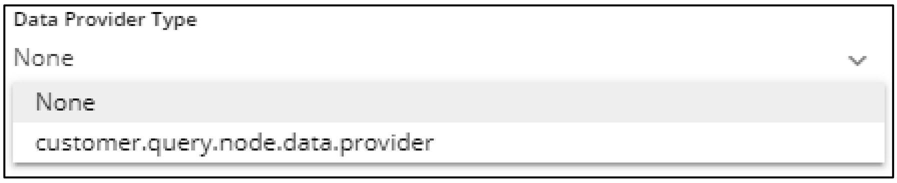
None

Figure: Attributes of Customer Query Node

3. **Phone Digits to be considered from End (Optional):** It gives the number of digits of the customer phone that have to be considered from the end to be taken as customer contact number.

4. **Use "OR" Operator (Optional):** It gives the information if "OR" operator has to be used or not. If its value is false or NULL, then "AND" operator would be used. If its value is true, then "OR" operator would be used.

5. **Data Provider:** This node contains the following data provider. Select it.



Data Provider Type

None

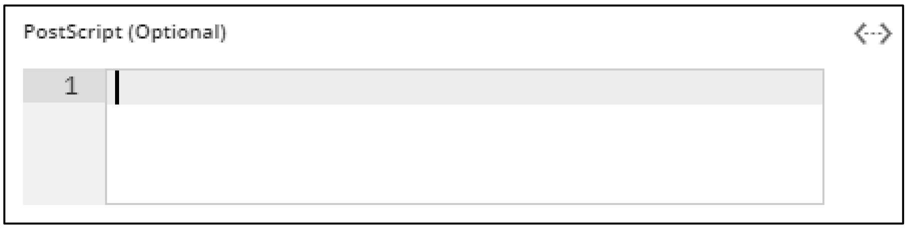
None

customer.query.node.data.provider

Figure: Data Provider of Customer Query Node

- customer.query.node.data.provider:

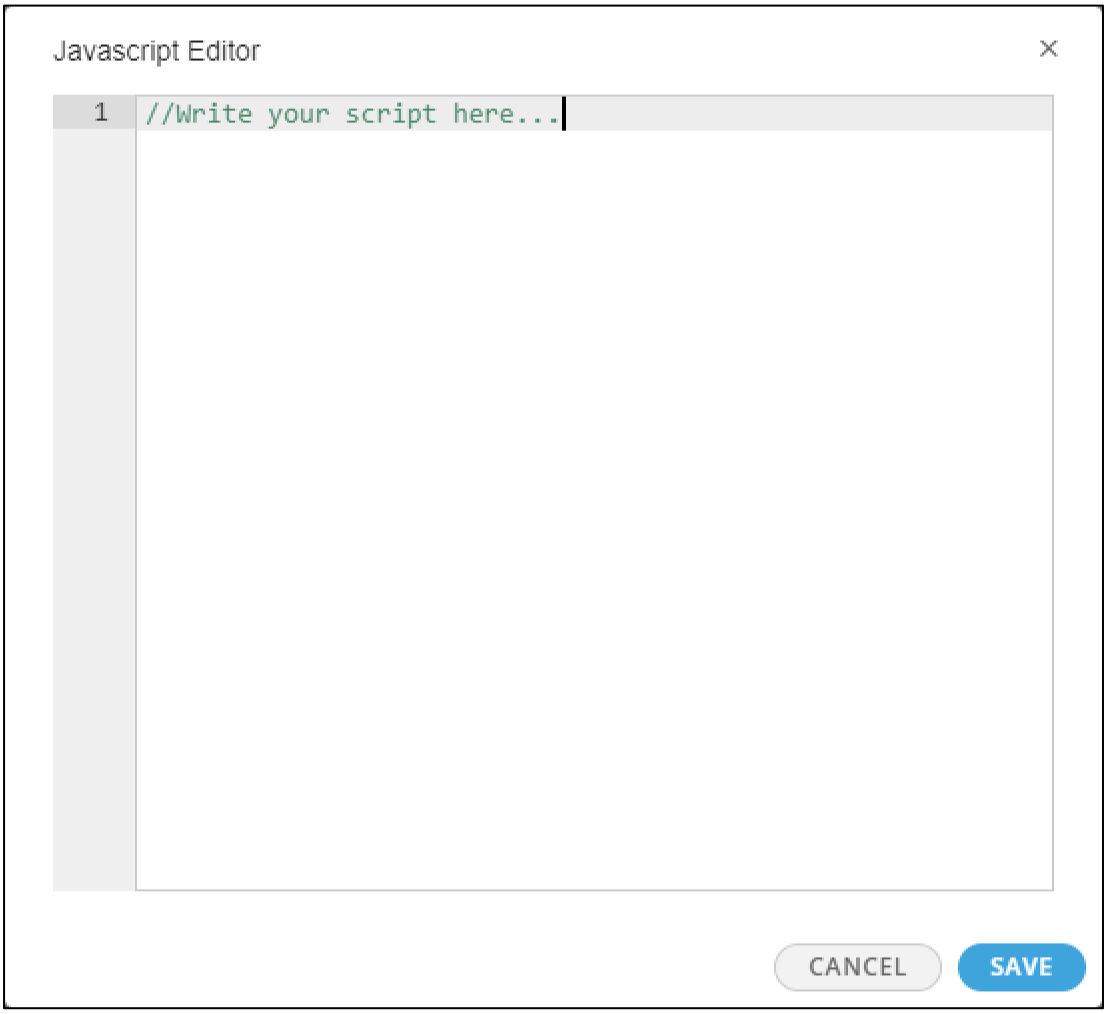
6. **Postscript (Optional):** Synthesizer allows to run a JavaScript code after the processing of the node.



PostScript (Optional)

Figure: Script to run after processing "Customer Query" Node

Here, paste the JavaScript. Click <--> icon to open the following JavaScript Editor.



```
//Write your script here...
```

Figure: JavaScript Editor

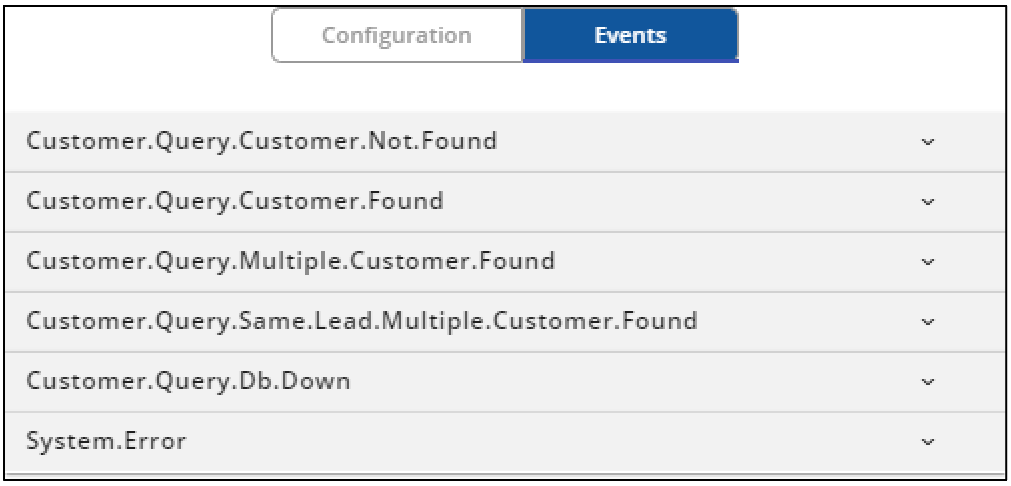
The Administrator can write its own code here and click "Save". It returns to node configuration, which will show the saved code.

7. **PreScript (Optional):** Specify to run a custom JavaScript code before processing the node. Like PostScript, add the code in Prescript.

8. Click "Save" to save the Node Configuration.

10.4.3 Events

Switch to "Events" tab to see the events of this node.



Customer.Query.Customer.Not.Found

Customer.Query.Customer.Found

Customer.Query.Multiple.Customer.Found

Customer.Query.Same.Lead.Multiple.Customer.Found

Customer.Query.Db.Down

System.Error

Figure: Events of Customer Query Node

These events are listed hereinbelow.

- **Event.Customer.Query.Customer.Not.Found:** It is generated in either of the following cases.
    - Data Provider is not an instance of CustomerQueryNodeDataProvider.
    - Contact Number in the Data Provider is NULL.
    - Campaign ID in the Data Provider is NULL.
    - CustomerCRTObjectId provided by the Data Provider is NULL.
    - The customer is not found in the database.
- **Event.Customer.Query.Customer.Found:** It is generated when a single customer has been retrieved for the provided contact number.
- **Event.Customer.Query.Same.Lead.Multiple.Customer.Found:** It is generated when the multiple customers have been retrieved for the provided contact number and they all belong to the same lead.
- **Event.System.Error:** It is generated in the following cases.
    - When we get the invalid nodemodel that is nodeModel is not an instance of CustomerQueryNodeModel.

  - When there is an exception while retrieving the VoiceCampaignInformation. It can occur when the default attribute for a particular component, for which Voice Campaign Information is to be retrieved, is not an instance of VoiceCampaignInfo.
  - When VoiceCampaignEntity is found null for the corresponding customer CRTObjectId, that is retrieved eventually from VoiceCampaignInformation. The customer CRTObjectId is said to be invalid.
  - When VoiceCampaignEntity is not instance of VoiceCallEntity for the corresponding customer CRTObjectId. The customer CRTObjectId is said to be invalid.

10.4.4 Output Script Variables

Following is the output variable, which is generated after the processing of this node.

- **customersCount:** It stores the number of customers retrieved for the provided contact number.
- **lastUserId:** It stores the last user's ID with whom the customer had interacted in that particular campaign.
- **customerId:** It stores the customer Id of the customer retrieved for the provided contact number.
- **lastStatus_[i]:** It stores the last 10 (i =1 to i =10) call statuses for the retrieved customer in that particular campaign.
- **lastDisposition_[i]:** It stores the last 10 (i =1 to i =10) call dispositions in the particular campaign for the retrieved customer.

10.4.5 Other Variables

Following is the list of other variables.

- **lastDialedNumber_[i]:** It stores the last 10 (i =1 to i =10) numbers that were dialed in that particular campaign for the retrieved customer.
- **lastDialedTime_[i]:** It stores the last 10 (i =1 to i =10) timestamps at which the customer was last dialed in that particular campaign.
- **leadOwner:** It stores the name of the owner of the lead to which the retrieved customer belongs.

10.5 Customer Relationship Management Node

It is used to specify a different CRM URL for the users. It appends call related and customer related data to the CRM URL as query parameters.

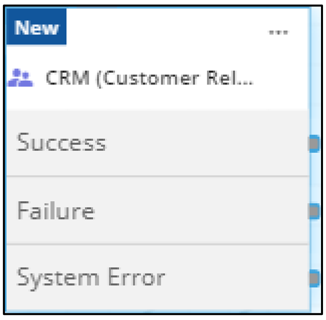
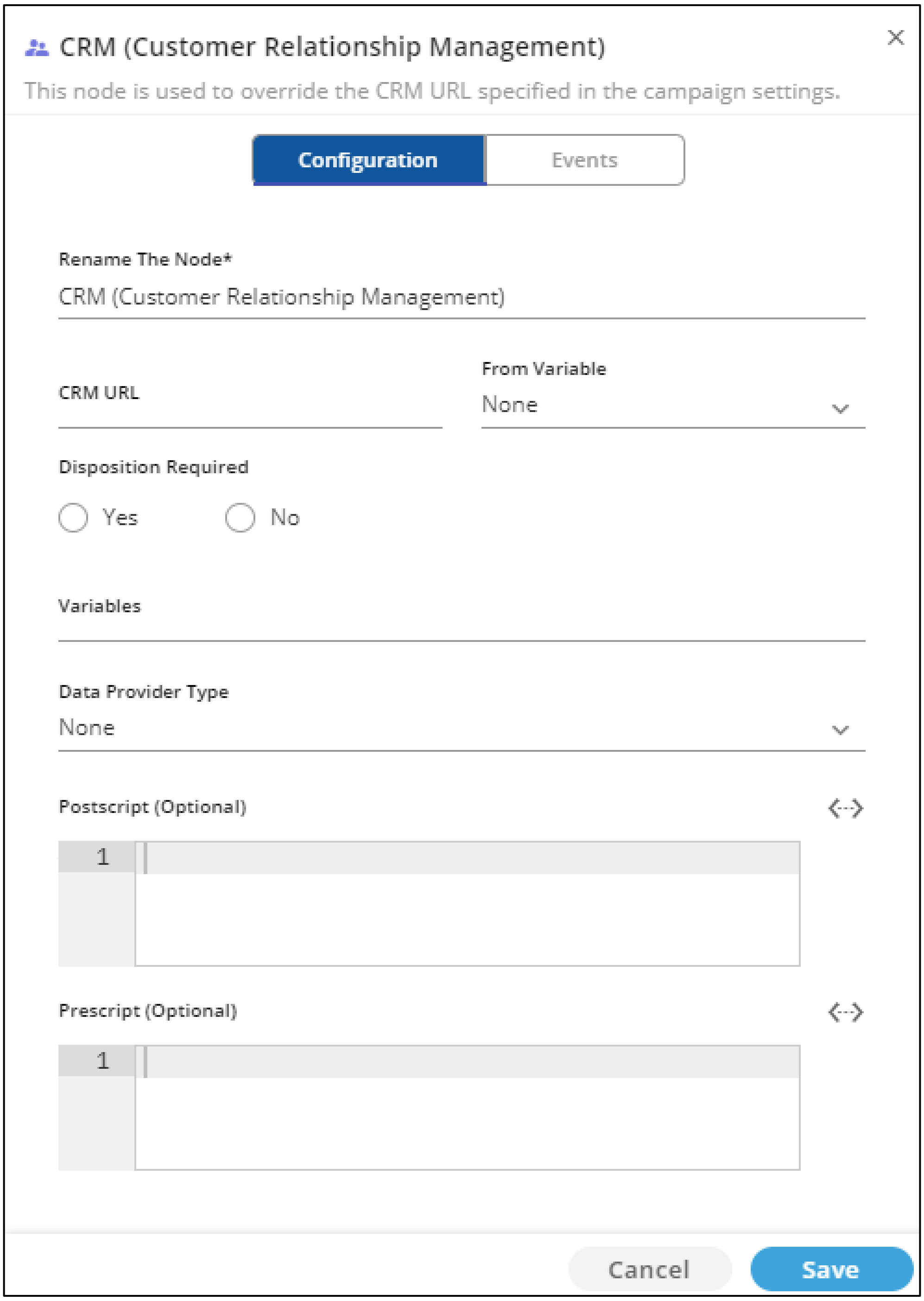


Figure: "Customer Relationship Management" Node

Following is a screenshot of its configuration.



CRM (Customer Relationship Management)

This node is used to override the CRM URL specified in the campaign settings.

Rename The Node*
CRM (Customer Relationship Management)

CRM URL

From Variable
None

Disposition Required
○ Yes ○ No

Variables

Data Provider Type
None

Postscript (Optional)

Prescript (Optional)

Figure: Configuration of "Customer Relationship Management" Node

10.5.1 Availability in Nodeflow Types

This node is available in the following Nodeflow types.

- Callback

- Customer-based Inbound
- Conference
- Dialing
- Inbound
- Interaction
- Manual Dial

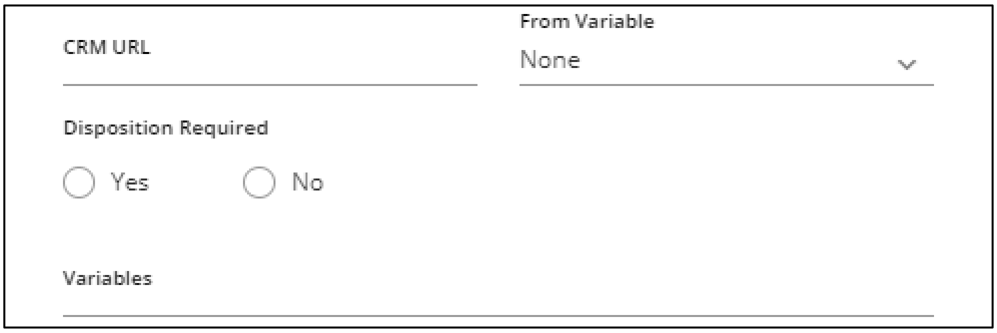
10.5.2 Configuration

Perform the following steps to configure this node.

1. **Rename The Node:** Rename the node, if required.

   Provide an easy-to-recall name, which matches the default node name.

2. **CRM URL (Optional, but Required):** It is the basic URL that is obtained from nodeflow. Data is fetched from the data provider and attributes and appended to this URL.



CRM URL

From Variable: None

Disposition Required: ○ Yes ○ No

Variables

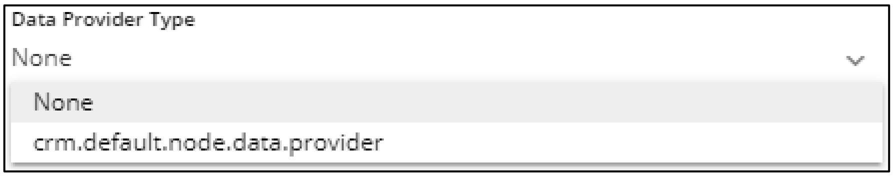
Figure: Attributes of "Customer Relationship Management" Node

3. **Disposition Required (Optional):** If the call needs to take the disposition from CRM, then this attribute should be set to true. The Application Response is determined and mapped to crtObjectId. It's default value is true. If it is NULL or selected as "NO", then the call will be auto-disposed.

4. **Variable (Optional):** Here, you can provide the semicolon (;) separated names of all variables, which are used to create the CRM URL. Variable names are taken from

variablesString and its value is get from scriptInterpreter that is used to construct the CRM URL.

5. **Data Provider:** This node contains the following one data provider. It is mandatory to select it else the node may fail to run.



Data Provider Type

None

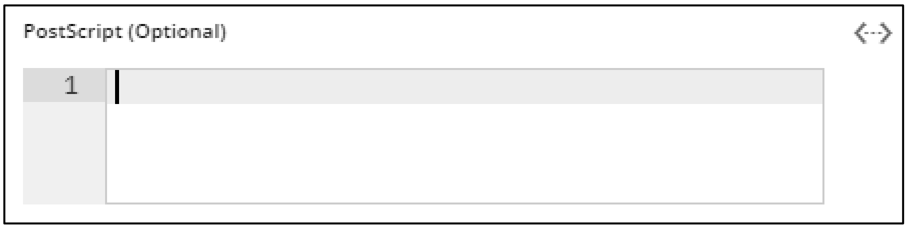
None

crm.default.node.data.provider

Figure: Data Provider of "Customer Relationship Management" Node

- crm.default.node.data.provider: It is the default data provider of this node. It provides CRM URL, crtObjectId, associationIds, callRuntime, and voiceCampaign Services.

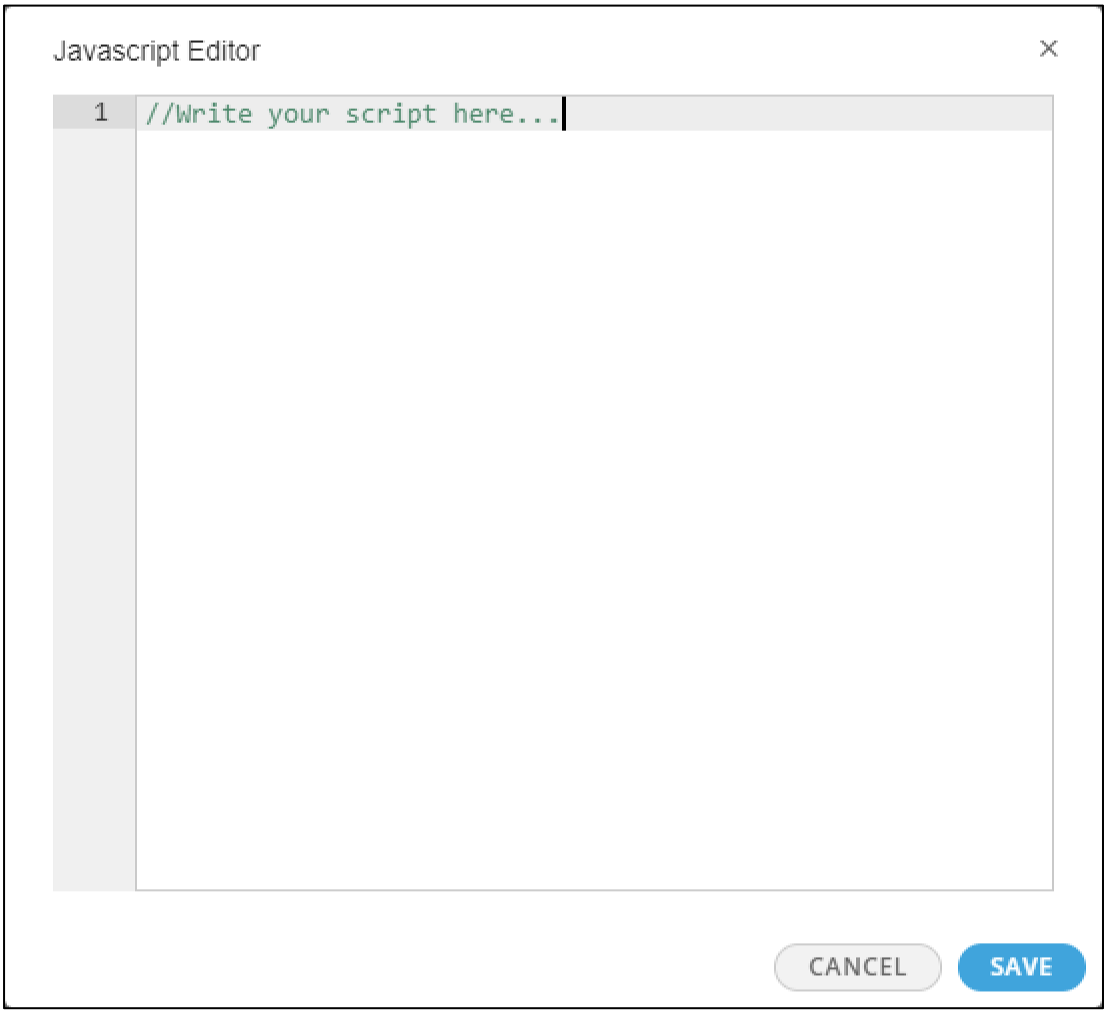
6. **Postscript (Optional):** Synthesizer allows to run a JavaScript code after the processing of the node.



PostScript (Optional)

Figure: Script to run after processing "Customer Relationship Management" Node

Here, paste the JavaScript. Click <--> icon to open the following JavaScript Editor.



```
//Write your script here...
```

Figure: JavaScript Editor

The Administrator can write its own code here and click "Save". It returns to node configuration, which will show the saved code.

7. **PreScript (Optional):** Specify to run a custom JavaScript code before processing the node. Like PostScript, add the code in Prescript.
8. Click "Save" to save the Node Configuration.

10.5.3 Events

Switch to "Events" tab to see the events of this node.

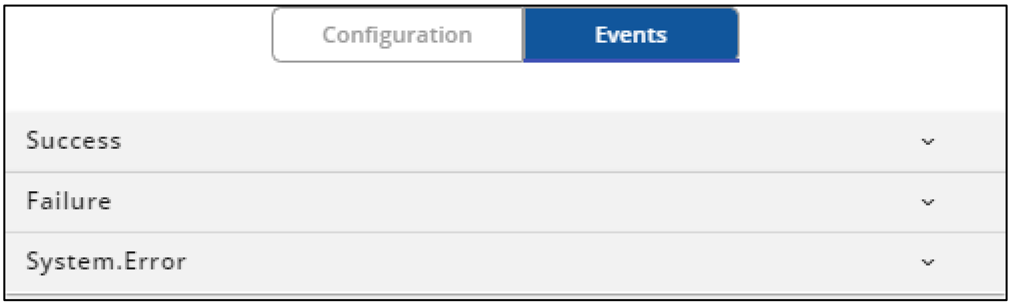


Figure: Events of "Customer Relationship Management" Node

These events are listed hereinbelow.

- **Event.Success:** It is generated when the node is successfully processed, but, it does not guarantee that CRM URL has been opened. When CRM url is modified, then the push is sent from the server, and it is the client's responsibility to open CRM. Also, in case CRM URL in nodeflow is not present (that is NULL), then campaign's configured CRM URL is used and "Customer Relationship Management" Node is executed giving Success as transition, where no CRM URL is created and is pushed to CRM.
- **Event.Failure:** It is generated when the node cannot be processed because of any reason such as incorrect attributes, no call leg, and others. This event is generated when data provider or nodeflow does not contain certain data.
    - **Cases:**
        - Data Provider returning:
            - crtObjectIds as NULL
            - VoiceCampaignInternal as NULL
            - associationIds as null
        - Nodeflow returning attribute:
            - sessionId as null
            - crtObjectId as null
- **Event.System.Error:** It is generated because of the reasons.
    - When received node model is not an instance of CRMNodeModel.

- Nodeflow does not have any attribute with name 'campaignId'
- There is problem sending crm url push.
- The system generate certain errors such as hardware, environmental, and others.

10.5.4 Output Script Variables

After running, "Customer Relationship Management" Node generates the following Output Script Variables.

- **variableStrings:** It contains all the semicolon(;) separated variables.

10.6 Custom Configuration Node

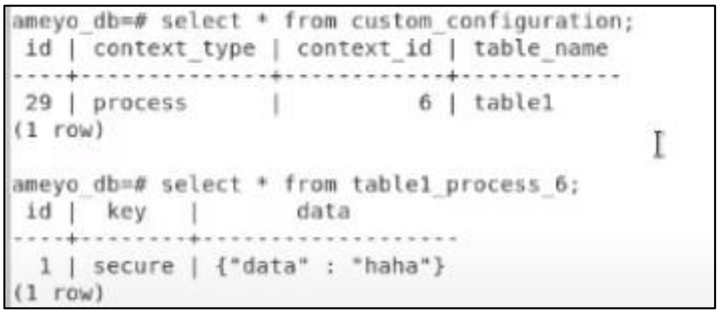
10.6.1 Introduction

Suppose a case of the Agent-based Preferred Routing where the call of a specific customer is routing to a Relationship Manager (RM). In the absence of RM, this call will be transferred to a Backup Relationship Manager. Also, take the examples of playing a prompt message for either advertisement or information about heavy call flow when the customer is waiting in the queue. In such cases, the backend configurations related to the nodeflow customizations have to be performed in the database. The Professional Services Team is storing such configurations in a custom table. However, this table was not standardized due to which the team members used to create different types of tables. Also, whenever a call is made or received, then the query is made to the database. If there is a large call volume, then multiple queries will be made to the database that will impact the database performance.

To help in such cases, Ameyo introduces the feature to make and access these customized configurations in the Cache instead of the database. This feature will improve the database performance and help in the faster turn out of the database queries. The following items have been introduced to meet this requirement.

- Standardized tables will be created for every context, including system, contact center, process, campaign, queue, and user to contain the custom configuration data in Cache.
- A new table named "custom_configuration" has been created, which will contain the names of these tables. Each table will be named as "<tablename_column_value>_<context_type>_<context_id>"; for example, "table1_process_6."

  The following screenshot displays these tables in the command line.



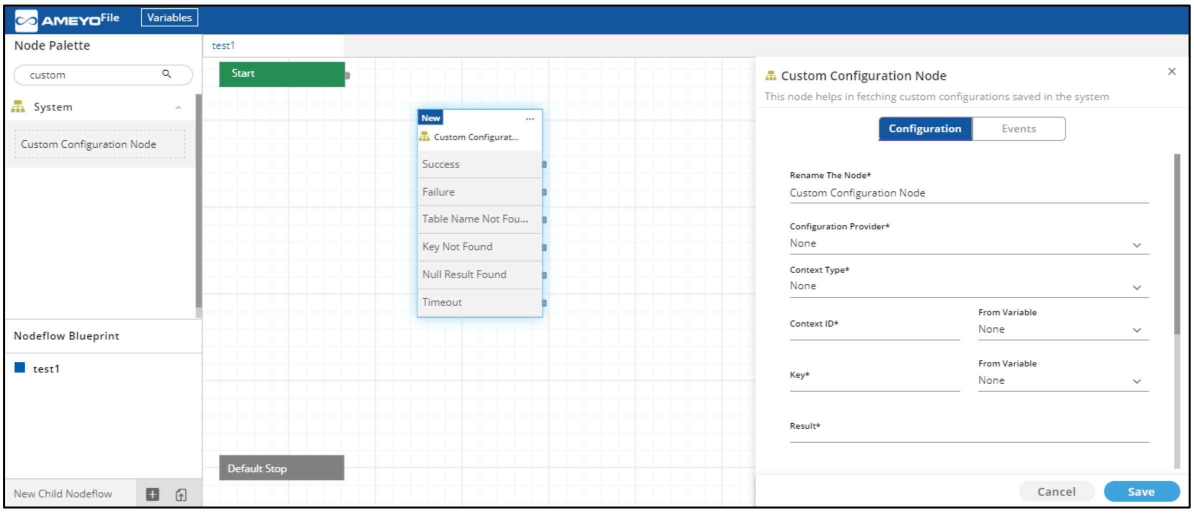
```
ameyo_db=# select * from custom_configuration;
 id | context_type | context_id | table_name
----+--------------+------------+------------
 29 | process      |          6 | table1
(1 row)

ameyo_db=# select * from table1_process_6;
 id |  key   |        data
----+--------+-------------------
  1 | secure | {"data" : "haha"}
(1 row)
```

Figure: Custom Configuration Tables

- APIs have been introduced that will add the data in these tables in the Cache.
- A new service named "CustomConfigurationService" has been introduced in the voice plugin.

10.6.2 About Custom Configuration Node

A new node named "Custom Configuration" has been introduced in Synthesizer. It accepts the following data providers for the services to fetch the custom configurations from the context tables. Refer to the following screenshot.



Custom Configuration Node

This node helps in fetching custom configurations saved in the system

Rename The Node*: Custom Configuration Node

Configuration Provider*: None

Context Type*: None

Context ID* — From Variable: None

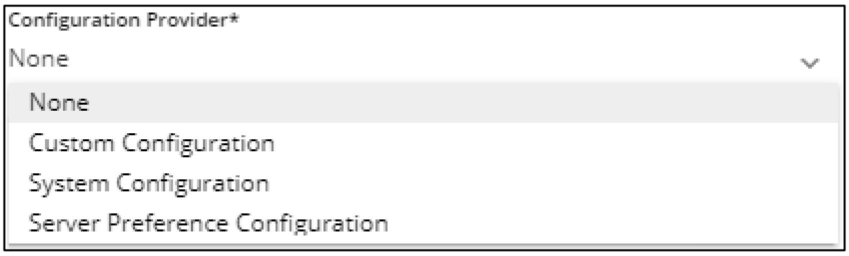
Key* — From Variable: None

Result*

Figure: Custom Configuration Node

The following input has to be provided in this node.

- **Configuration Providers**: Select any of the following configuration providers.
  - **Custom Configuration:** Use this data provider to fetch the configuration data from "custom_configuration" table.



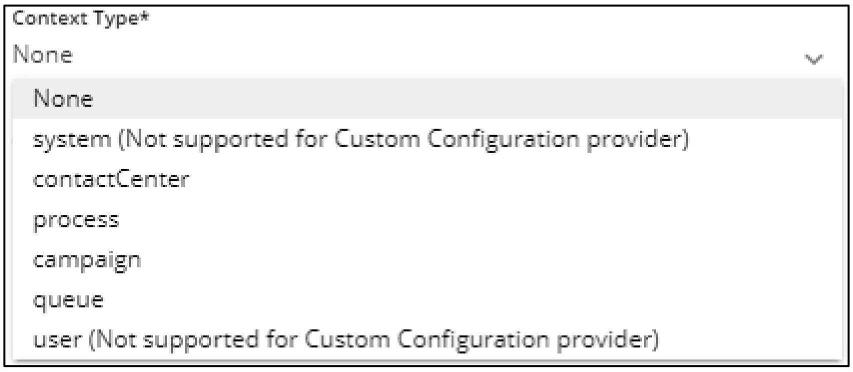
Configuration Provider*

None

- None
- Custom Configuration
- System Configuration
- Server Preference Configuration

Figure: Configuration Providers

  - **System Configuration:** Use this data provider to fetch the configuration data from "system_configuration_parameter" table.
  - **Server Preference Configuration:** Use this data provider to fetch the configuration data from "server_preference_store" table.
- **Context Type**: Select any of the following context types.
  - System (Not supported for "Custom Configuration" configuration provider)



Context Type*

None

None

system (Not supported for Custom Configuration provider)

contactCenter

process

campaign

queue

user (Not supported for Custom Configuration provider)

Figure: Context Types

  - contactCenter
  - process
  - campaign
  - queue
  - user (Not supported for "Custom Configuration" configuration provider)

- **Context ID:** Provide the context ID of which the context table has to be accessed.
- **Key:** Provide the key to read the specific configuration. It is a dynamic key value that can be passed through the variable.
- **Result:** It is the attribute where the result is set based on the input data filled in node attributes.

10.7 Do Not Call Node

It is used to verify if the given contact exist in the DNC of either in the campaign or the process or the contact center.

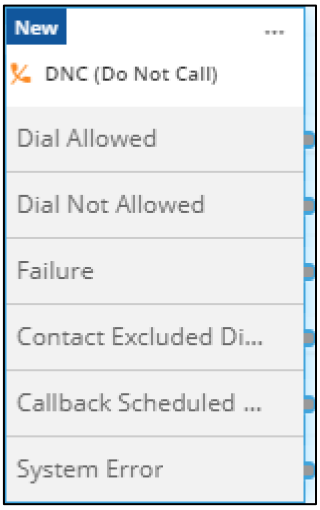
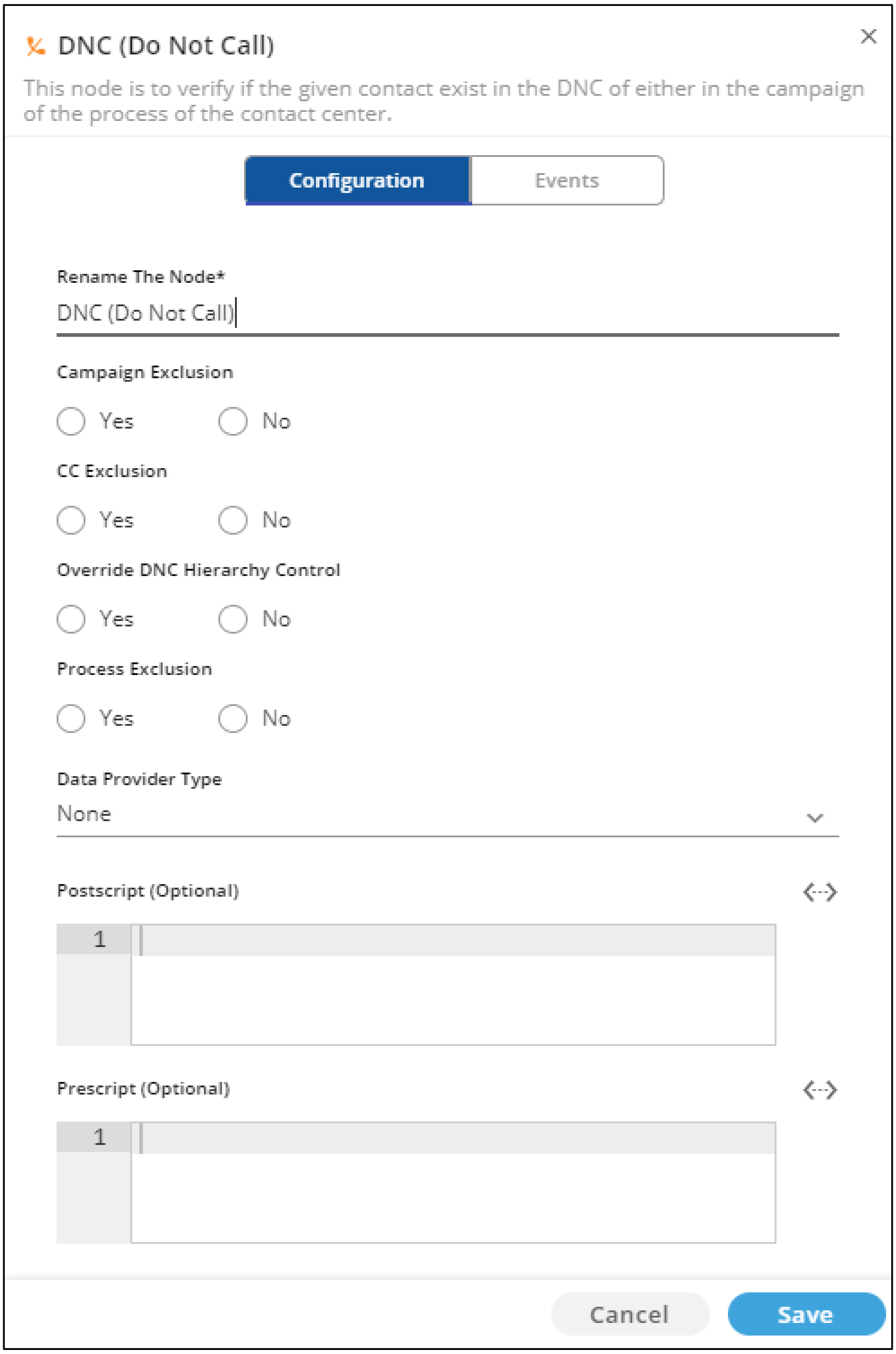


Figure: "Do Not Call" Node

Following is a screenshot of its configuration.



DNC (Do Not Call)

This node is to verify if the given contact exist in the DNC of either in the campaign of the process of the contact center.

Rename The Node*

DNC (Do Not Call)

Campaign Exclusion

○ Yes ○ No

CC Exclusion

○ Yes ○ No

Override DNC Hierarchy Control

○ Yes ○ No

Process Exclusion

○ Yes ○ No

Data Provider Type

None

Postscript (Optional)

```
1
```

Prescript (Optional)

```
1
```

Figure: Configuration of "Do Not Call" Node

10.7.1 Availability in Nodeflow Types

This node is available in the following Nodeflow types.

- Callback
- Customer-based Inbound
- Dialing
- Inbound
- Manual Dial

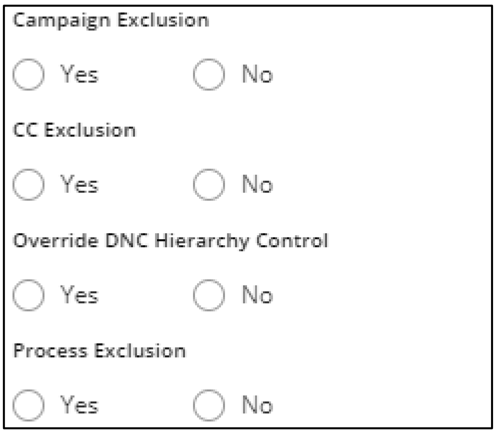
10.7.2 Configuration

Perform the following steps to configure this node.

1. **Rename The Node:** Rename the node, if required.

   Provide an easy-to-recall name, which matches the default node name.

2. **Campaign Exclusion (Optional):** If its value is "Yes", then it will enable the Number Exclusion configured at Campaign Level.
3. **CC Exclusion (Optional):** If its value is "Yes", then it will enable the Number Exclusion configured at Contact Center Level.
4. **Override DNC Hierarchy Control (Optional):** If its value is "Yes", then it will allow the node to override the system configured value of excluding the number at campaign level, process level, or contact center level. In other words, if configured as "Yes" the number will be dialed even if it is excluded at campaign level, process level, or contact center level.



Campaign Exclusion

○ Yes ○ No

CC Exclusion

○ Yes ○ No

Override DNC Hierarchy Control

○ Yes ○ No

Process Exclusion

○ Yes ○ No

Figure: Attributes of "Do Not Call" Node

5. **Proces Exclusion:** If its value is "Yes", then it will enable the Number Exclusion configured at the Process Level.
6. **Data Provider:** This node does not contain any data provider.
7. **Postscript (Optional):** Synthesizer allows to run a JavaScript code after the processing of the node.

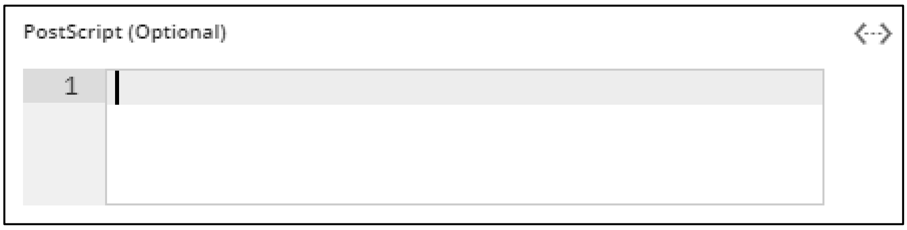
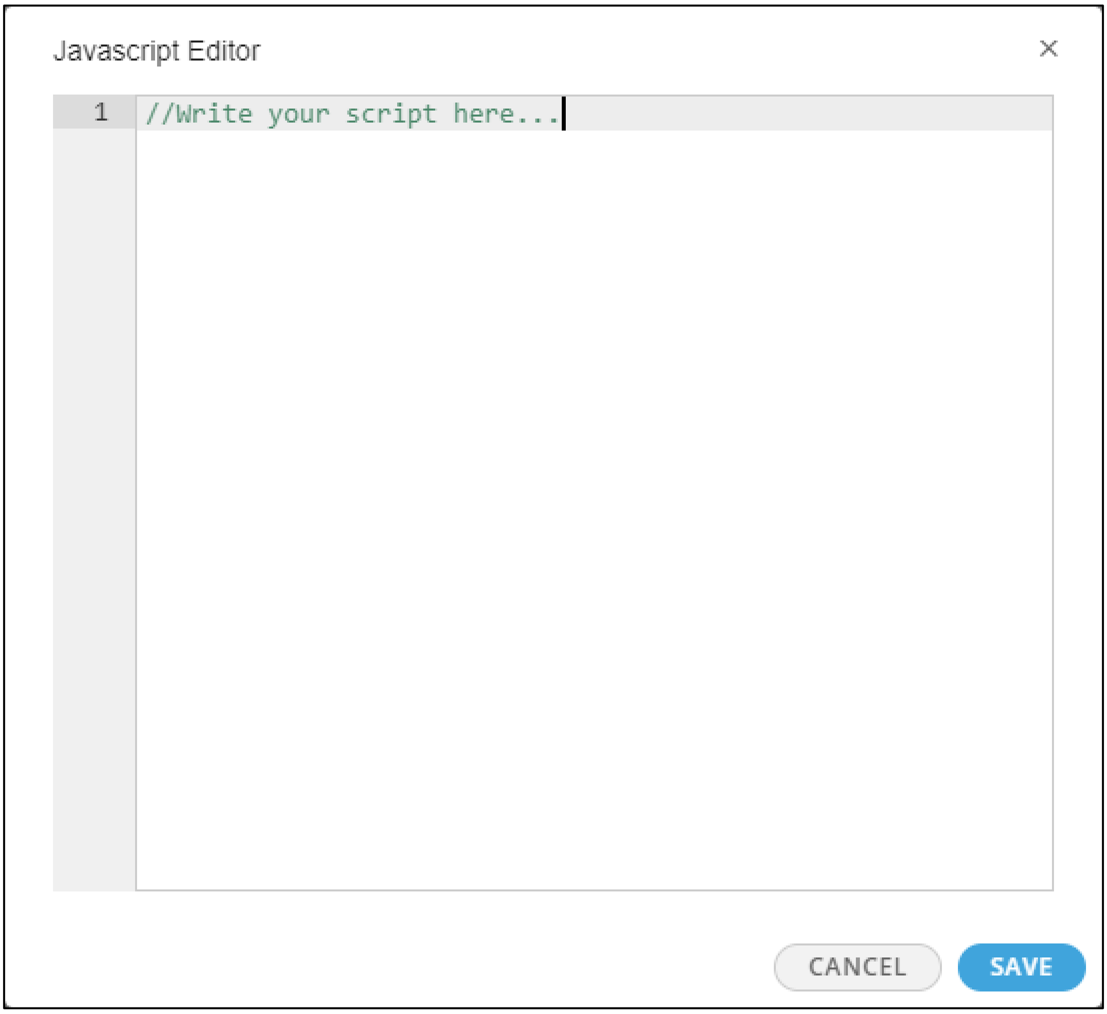


Figure: Script to run after processing "Do Not Call" Node

Here, paste the JavaScript. Click <--> icon to open the following JavaScript Editor.



```
//Write your script here...
```

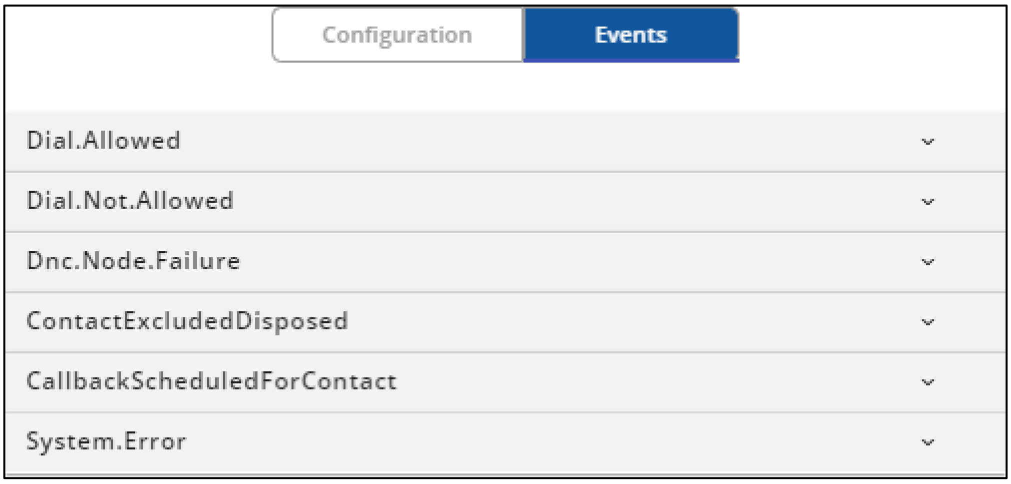
Figure: JavaScript Editor

The Administrator can write its own code here and click "Save". It returns to node configuration, which will show the saved code.

8. **PreScript (Optional):** Specify to run a custom JavaScript code before processing the node. Like PostScript, add the code in Prescript.
9. Click "Save" to save the Node Configuration.

10.7.3 Events

Switch to "Events" tab to see the events of this node.



Dial.Allowed

Dial.Not.Allowed

Dnc.Node.Failure

ContactExcludedDisposed

CallbackScheduledForContact

System.Error

Figure: Events of DNC Node

These events are listed hereinbelow.

- **Event.Dial.Allowed:** It is generated when the number is not mentioned in "Do Not Call" list and its dialing is allowed.
- **Event.Dial.Not.Allowed:** It is generated when the number is mentioned in "Do Not Call" list and its dialing is not allowed.
- **Event.Do.Not.Call.Node.Failure:** It is generated when this node fails because of any invalid value set in the attributes or other application failures.
- **Event.Contact.Excluded.Disposed:** It is generated when the customer's is_excluded_disposed flag is true (last disposition belongs to excluded disposition codes list) and shouldCheckExcludedOrCallbackDisposedCustomer flag is enabled in server preference store
- **Event.Callback.Scheduled.For.Contact:** It is generated when the callback is scheduled for the customer and shouldCheckExcludedOrCallbackDisposedCustomer flag is enabled in server preference store
- **Event.System.Error:** It is generated when the system generates certain errors such as hardware, environmental, and others.

10.8 Make Call Node

It is used to combine all call legs and Call Runtime Objects to make a call. For example, in an outbound process, when the call is picked up by the customer, then use this node to connect the call to the agent. It will be successful only when the agent is connected. It is also used during an existing call such as to make the conference or make the transfer.

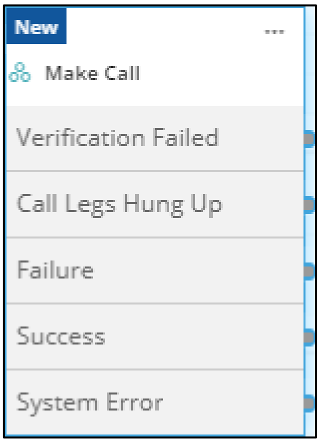
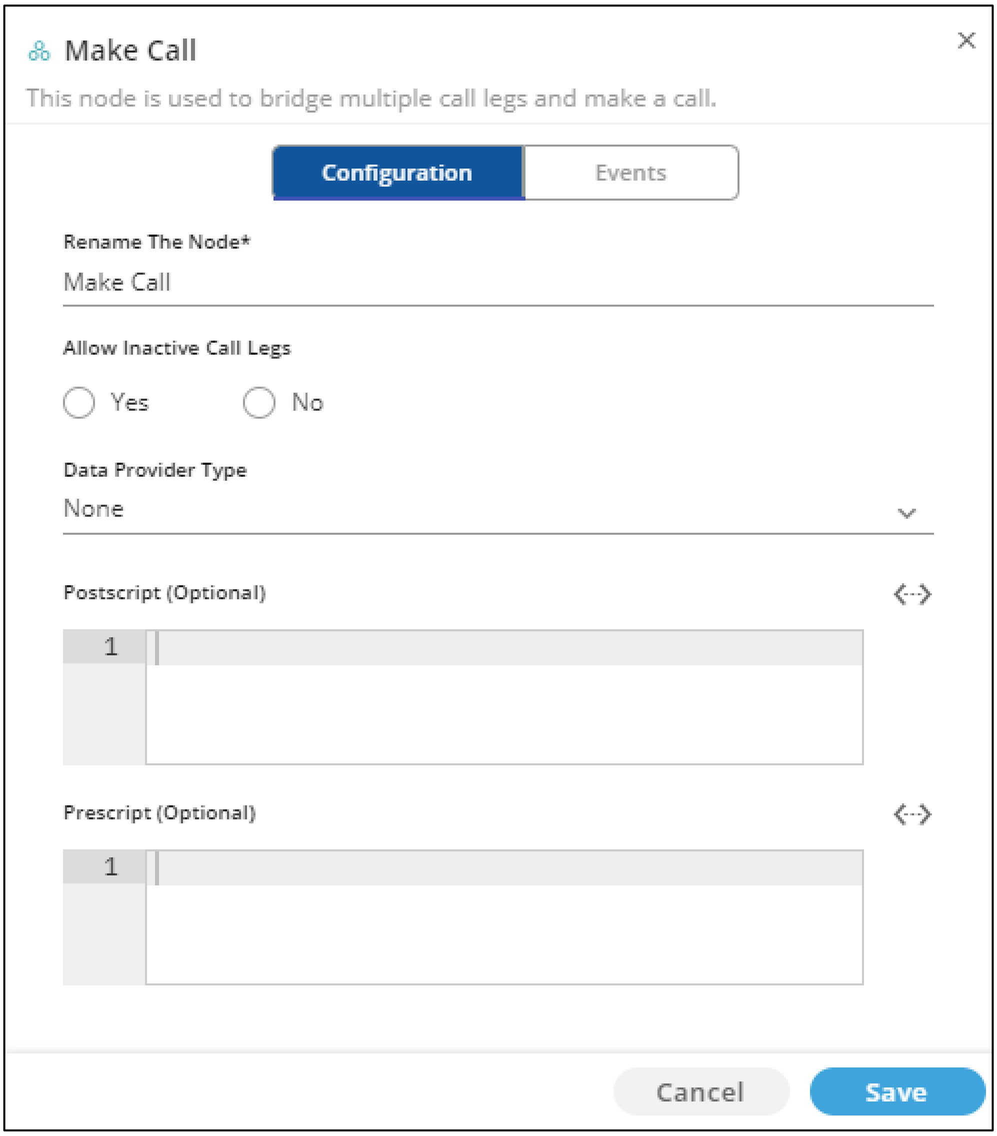


Figure: "Make Call" Node

Following is the screenshot of its configuration.



Make Call

This node is used to bridge multiple call legs and make a call.

Rename The Node*
Make Call

Allow Inactive Call Legs
○ Yes ○ No

Data Provider Type
None

Postscript (Optional)

Prescript (Optional)

Figure: Configuration of "Make Call" Node

10.8.1 Availability in Nodeflow Types

This node is available in the following Nodeflow types.

- Callback
- Customer-based Inbound
- Conference
- Dialing
- Inbound

- Manual Dial

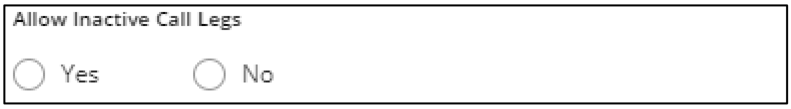
10.8.2 Configuration

Perform the following steps to configure this node.

1. **Rename The Node:** Rename the node, if required.

   Provide an easy-to-recall name, which matches the default node name.

2. **Allow Inactive Call Legs (Optional):** If you want to allow the inactive call legs, then give the value of this attribute in "Boolean" format - "TRUE". If set to "FALSE", then the inactive call legs will not be allowed.

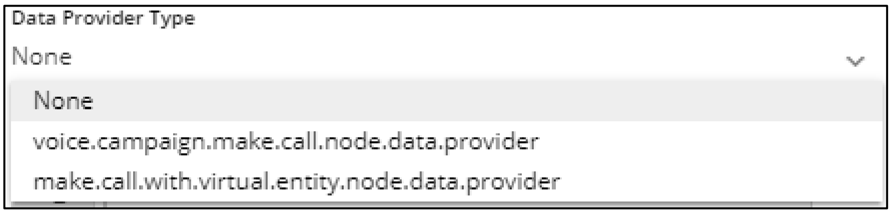


Allow Inactive Call Legs

○ Yes ○ No

Figure: Attributes of Make Call Node

3. **Data Provider:** This node contains the following data providers. Select any one of them.



Data Provider Type

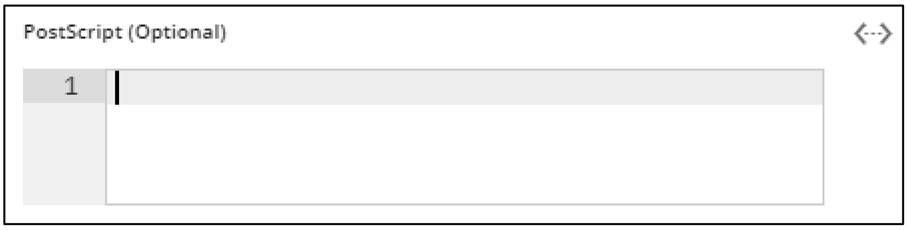
None

None
voice.campaign.make.call.node.data.provider
make.call.with.virtual.entity.node.data.provider

Figure: Data Provider of Make Call Node

   - voice.campaign.make.call.node.data.provider: It is used to bridge the live call leg originated by originate node.
   - make.call.with.virtual.entity.node.data.provider: It is used to bridge the virtual call leg (such as CRT call leg) in case of conference.

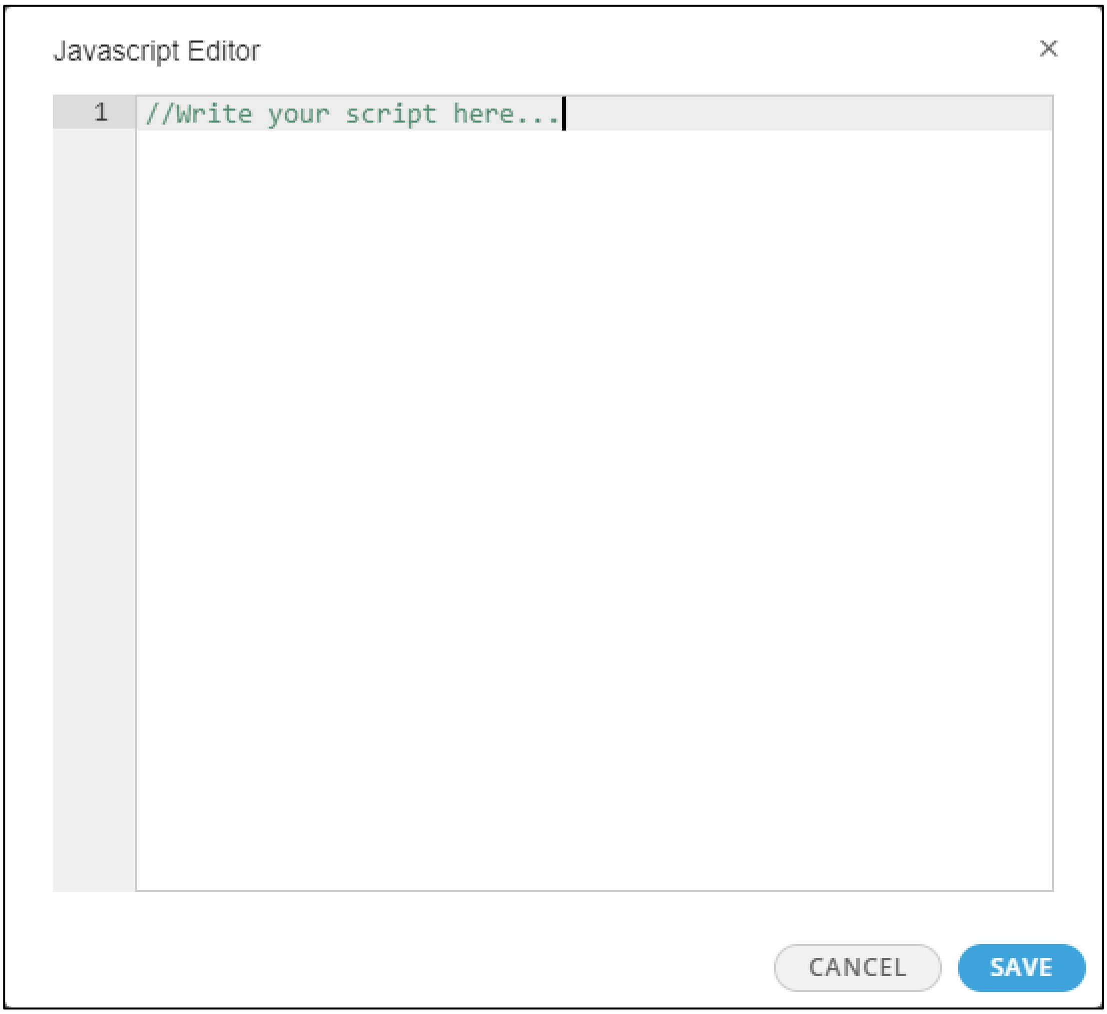
4. **Postscript (Optional):** Synthesizer allows to run a JavaScript code after the processing of the node.



PostScript (Optional)

Figure: Script to run after processing "Make Call" Node

Here, paste the JavaScript. Click <--> icon to open the following JavaScript Editor.



Javascript Editor

```
//Write your script here...
```

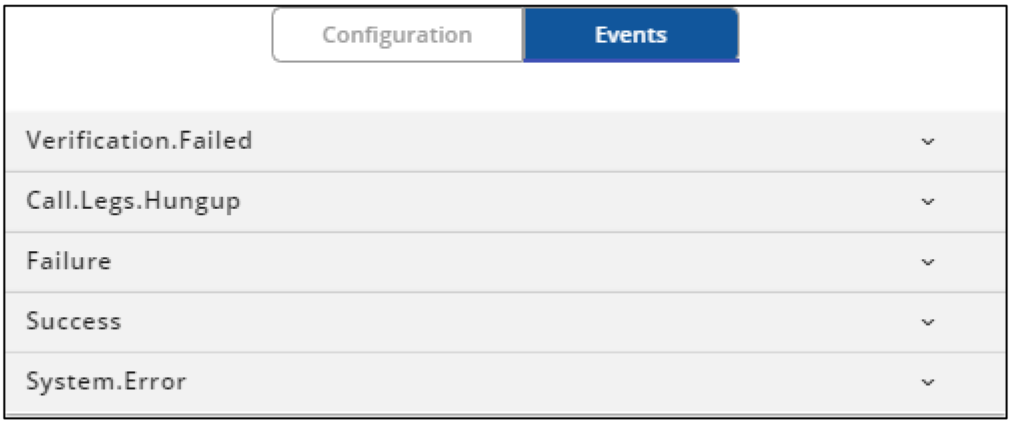
Figure: JavaScript Editor

The Administrator can write its own code here and click "Save". It returns to node configuration, which will show the saved code.

5. **PreScript (Optional):** Specify to run a custom JavaScript code before processing the node. Like PostScript, add the code in Prescript.
6. Click "Save" to save the Node Configuration.

10.8.3 Events

Switch to "Events" tab to see the events of this node.



Verification.Failed

Call.Legs.Hungup

Failure

Success

System.Error

Figure: Events of Make Call Node

These events are listed hereinbelow.

- **Event.Verification.Failed:**
- **Event.Call.Legs.Hungup:** It is generated when "Allow Inactive Call Leg" Attribute flag is set to "FALSE" and the system is not able to fetch any call leg or a valid media call leg from the object. CallLegActionResponse is unsuccessful, that is, Call Runtime Objects are failed to enter in the already initiated call because of any of the following reasons.
  - If CallLegActionResponse is timed out.
  - It is generated when one of the calls gets disconnected while the other is about to be connected.
- **Event.Failure:** It is generated because of any of the following reasons.
  - "Make Call Runtime Objects" information are null.
  - Exception occurred while adding the call legs to call.

    - Nodeflow already stopped when received response of any async (asynchronization) request like adding call legs to call.
    - Call Runtime object hung-up when call was created
    - While obtaining the call manager service.
    - Call Manager Voice Resource is not Registered or available while creating the call.
    - Call Context is not enabled.
    - Invalid data provider
    - Nodeflow returns error event

- **Event.Success:** It is generated in any of the following cases.
    - If CallLegActionResponse is successful, that is, Call Runtime Objects are successfully entered in the already existing initiated call.
    - It is generated when the node is successfully processed and both calls are bridged.
    - If request has been placed and all call legs enter call.
    - Some call legs might exit from the call before other call legs have entered the call.
- **Event.System.Error:** It is generated in any of the following cases.
    - Invalid node model for MakeCall Node
    - Invalid Component State of Call Runtime object while obtaining the call manager service
    - While obtaining crt object node service
    - Invalid Call Runtime object found in make call data
    - When the system generate certain errors such as hardware, environmental, and others.

10.9 Originate Call Runtime Node

It is used to setup connection between customers and agents. In an Outbound process, when the customer picks up a call, then this node generates a call for the agent to connect to the customer.

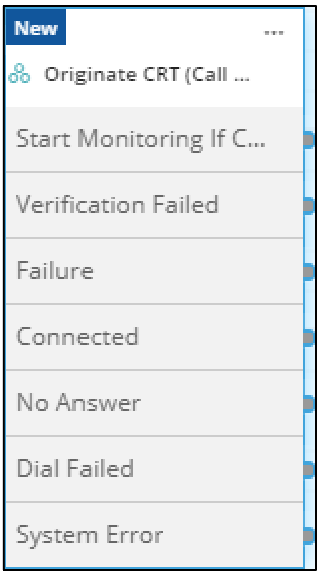
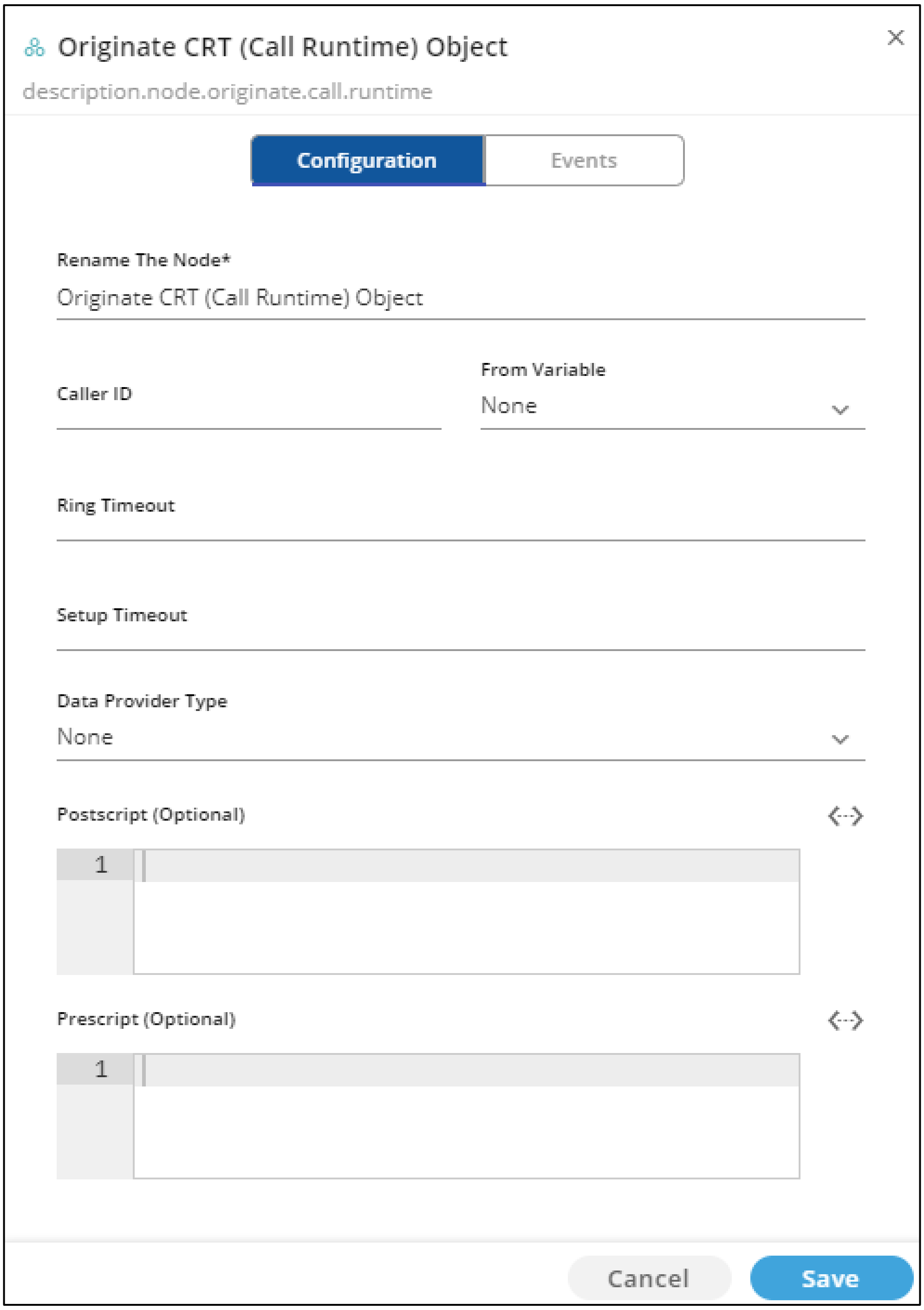


Figure: "Originate Call Runtime" Node

Following is a screenshot of its configuration.



Originate CRT (Call Runtime) Object

description.node.originate.call.runtime

Rename The Node*: Originate CRT (Call Runtime) Object

Caller ID

From Variable: None

Ring Timeout

Setup Timeout

Data Provider Type: None

Postscript (Optional)

Prescript (Optional)

Figure: Configuration of "Originate Call Runtime" Node

10.9.1 Availability in Nodeflow Types

This node is available in the following Nodeflow types.

- Callback

- Customer-based Inbound
- Conference
- Dialing
- Inbound
- Manual Dial

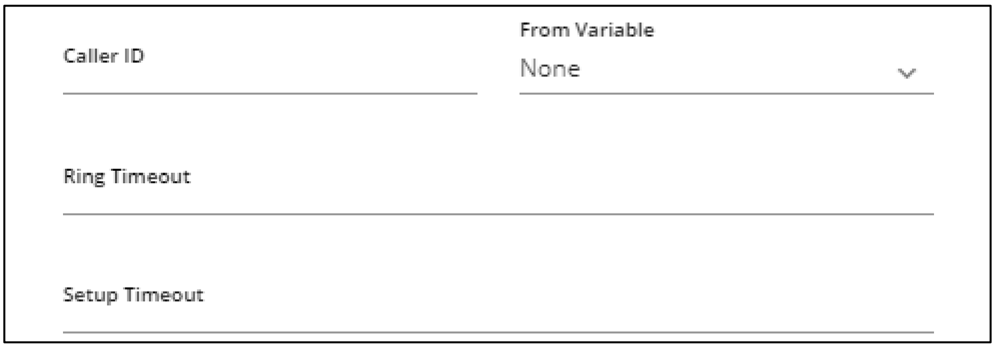
10.9.2 Configuration

Perform the following steps to configure this node.

1. **Rename The Node:** Rename the node, if required.

   Provide an easy-to-recall name, which matches the default node name.

2. **Caller ID (Optional):** If you want to override the Caller ID configured in the Campaign Settings, then you can provide the Caller ID here. It will be displayed to the agent who will receive the call. It's value format is "String".



Caller ID

From Variable

None

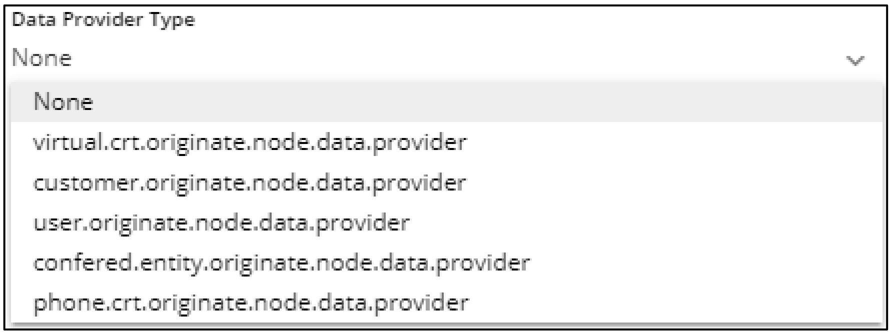
Ring Timeout

Setup Timeout

Figure: Attributes of "Originate Call Runtime" Node

3. **Ring Timeout (Optional):** Ring Timeout is actually defined in the Campaign Settings and if they are left blank at Campaign Level, then the default value from the database will be used.
   Here, you can provide the value of Ring Time Out in seconds, which will override the Ring Timeout given at the Campaign Level. The call will be disconnected after the ringing time for that call reaches this value.

4. **Setup Timeout (Optional):** Setup Time is the defined as the time duration which starts from the initiation of the call request till the time when phone starts ringing. It is actually defined in the Campaign Settings and if they are left blank at Campaign Level, then the default value from the database will be used. Here, you can provide the value of Setup Timeout in seconds, which will override the Setup Timeout given at the campaign level. The call will be disconnected after reaching this limit.
5. **Data Provider:** This node contains the following data providers. Select any one of them.



Data Provider Type

None

None

virtual.crt.originate.node.data.provider

customer.originate.node.data.provider

user.originate.node.data.provider

confered.entity.originate.node.data.provider

phone.crt.originate.node.data.provider

Figure: Data Provider of "Originate CRT" Node

- virtual.crt.originate.node.data.provider: It is used to originate Call Runtime Object for the virtual calls. The successful transitions of AssociateVirtualCRTObjectNode should originate for the virtually created entity on the user prompt.
- customer.originate.node.data.provider: It is used to originate Call Runtime object to a customer to take customer information according to the calll type such as Manual Dial or Auto Dial.
- user.originate.node.data.provider: It is used to originate Call Runtime object to a user to take information as per the call type. For example, it will originate the Call Runtime Object to the user obtained from ACD in case of an inbound call.

- confered.entity.originate.node.data.provider: It is used to originate Call Runtime Object to a conferred phone to take information from conferWithPhone feature.
- phone.crt.originate.node.data.provider: It is used originate Call Runtime Object to an agent's phone.

6. **Postscript (Optional):** Synthesizer allows to run a JavaScript code after the processing of the node.

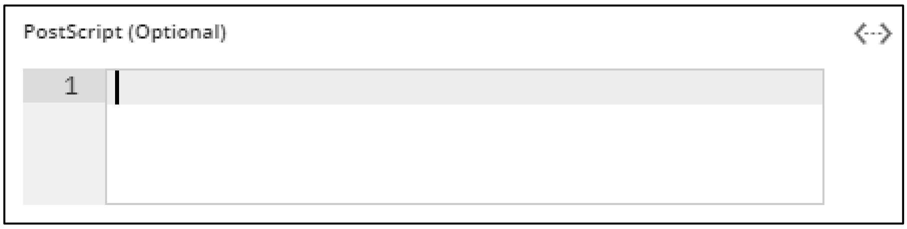
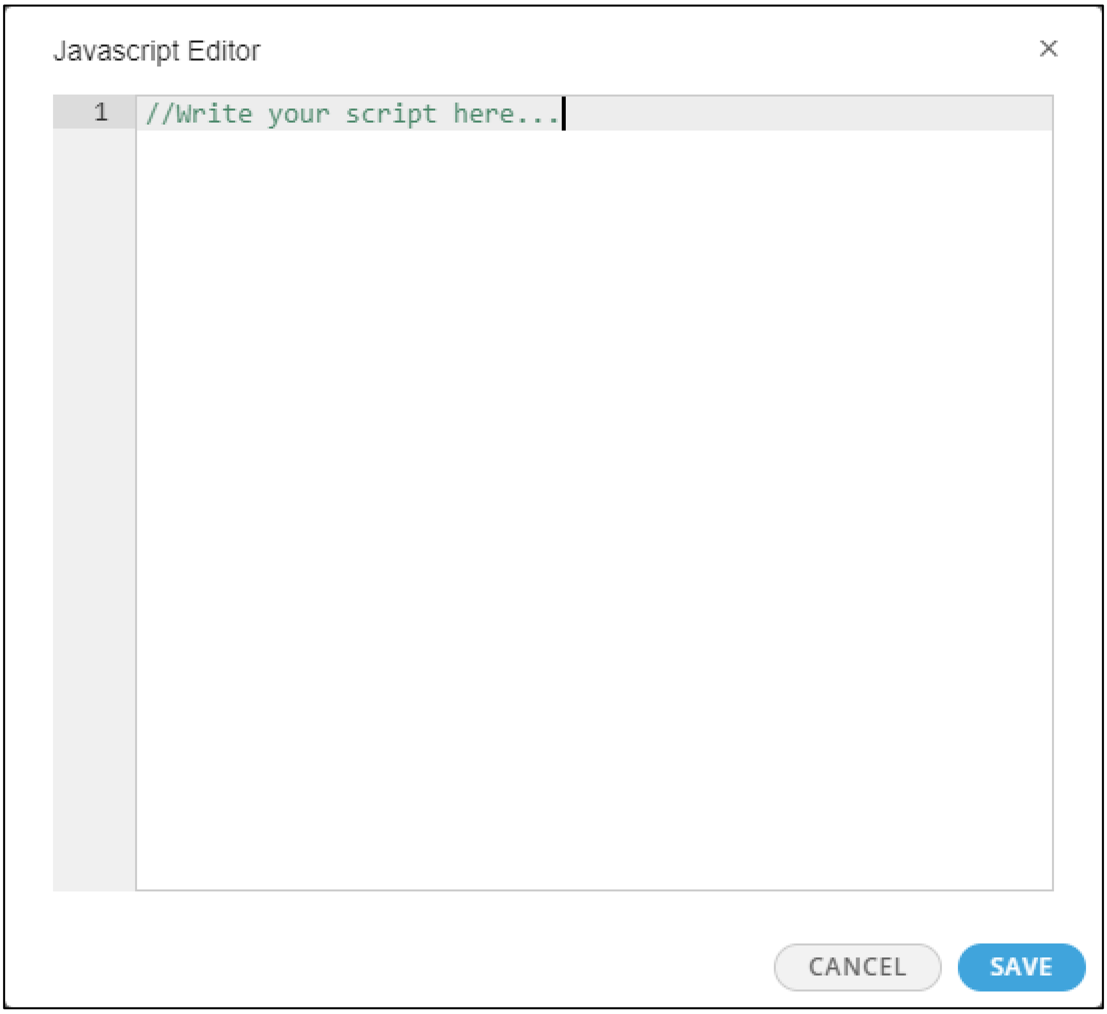


Figure: Script to run after processing "Originate Call Runtime" Node

Here, paste the JavaScript. Click <--> icon to open the following JavaScript Editor.



```
//Write your script here...
```

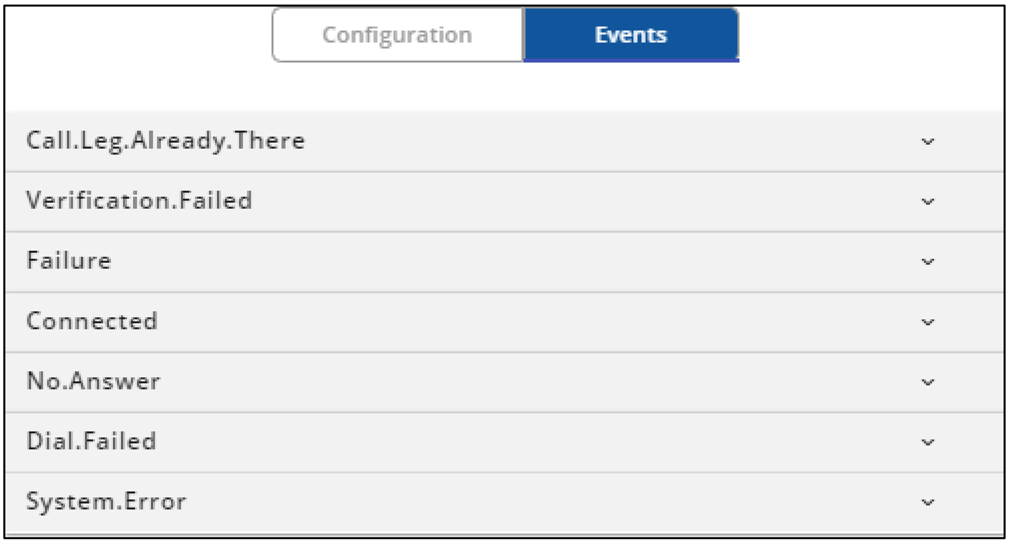
Figure: JavaScript Editor

The Administrator can write its own code here and click "Save". It returns to node configuration, which will show the saved code.

7. **PreScript (Optional):** Specify to run a custom JavaScript code before processing the node. Like PostScript, add the code in Prescript.
8. Click "Save" to save the Node Configuration.

10.9.3 Events

Switch to "Events" tab to see the events of this node.



Configuration | Events

- Call.Leg.Already.There
- Verification.Failed
- Failure
- Connected
- No.Answer
- Dial.Failed
- System.Error

Figure: Events of Originate CRT Node

These events are listed hereinbelow.

- **Event.Call.Leg.Already.There:** It is generated when the entity has a valid call leg, that is, contact with an entity which is already in the call.
- **Event.Verification.Failed:** It is generated because of the following reasons and only when customer.originate.node.data.provider is selected as the Data Provider.
  - phone_dial_order license is enabled, but dial_multiple flag is false.
  - The system failed to dial all phone numbers available in the Customer Information.
- **Event.Failure:** It is generated when the node setting is not correct. For example, data provider is not set or Call Runtime Object (crtObject) no longer exist in the system.
- **Event.Connected:** It is generated when the Call Leg is connected.
- **Event.No.Answer:** It is generated when we recieve the call leg state HANGUP after RINGING.
- **Event.Dial.Failed:** It is generated when we recieve the call_leg state HANGUP after INTIALIZED.
- **Event.System.Error:** It is generated because of any of the following reasons.
  - Failed to invoke API for getting the Call Runtime Object.

- If Call Runtime Object does not contain mediaCalllegdetails.
- Failed to create the call leg in call manager.

10.9.4 Output Script Variables

After running, "Originate Call Runtime" Node generates the following Output Script Variables.

- **channelHeaders:** It contains the media specific SIP details.
- **hangupCause:** It contains hang-up cause code, which is understood at Call Manager levels. It can be number.failure, provider.failure, and others. A single hang-up cause can correspond to multiple hang-up cause codes.
- **hangupCauseCode:** It contains hang-up cause code, which can give the detailed description about the Call Hang-up. It can correspond to different entity types such as ASTERISK_SIP, ASTERISK_ZAP_TRUNK, and ASTERISK_ZAP_CHANNEL.
- **hangupCauseDescription:** It contains the detailed hang-up cause description that corresponds to a Hang-up Cause Code.

10.10Pack Node

It saves the properties from nodeflow variables in a persistence service based on a unique ID.

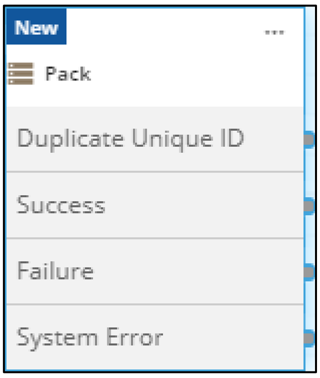


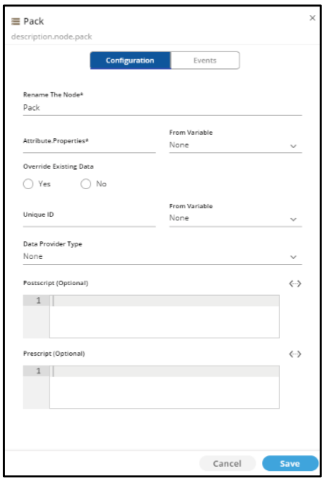
Figure: "Pack" Node

In other words, "Pack" Node is used to transmit the variables from a nodeflow to the same or different nodeflow. All the variables in persistence service are cleared after a defined cleanup interval. The default value of cleanup interval is 15 minutes, however, it can be changed from the backend.

Steps to change the interval from the backend: PENDING ON DEV TEAM

Unpack Node is required to receive and unpack the data in the same or different nodeflow.

Following is a screenshot of its configuration.



Rename The Node*
Pack

Attribute.Properties*

From Variable
None

Override Existing Data
○ Yes ○ No

Unique ID

From Variable
None

Data Provider Type
None

Postscript (Optional)

Prescript (Optional)

Figure: Configuration of "Pack" Node

10.10.1 Availability in Nodeflow Types

This node is available in the following Nodeflow types.

- Callback
- Customer-based Inbound
- Conference
- Dialing
- Inbound
- Manual Dial
- Post Call Processing

10.10.2 Configuration

Perform the following steps to configure this node.

1. **Rename The Node:** Rename the node, if required.

   Provide an easy-to-recall name, which matches the default node name.

2. **Attribute Properties (Mandatory):** It's acceptable value type is "String". It is the mandatory variable that has to be provided by the user while configuring this node. It can contain comma separated attributes or script variables. Following are the examples of its acceptable value.

   ```
   Attribute Properties =
   "originalCustomerPhone,originalCustomerCRTId,originalAgentId"

   Attribute Properties = "col1=val1,col2=val2"
   ```

   The user can either provide the value manually in the textbox or select a variable from the drop-down menu.

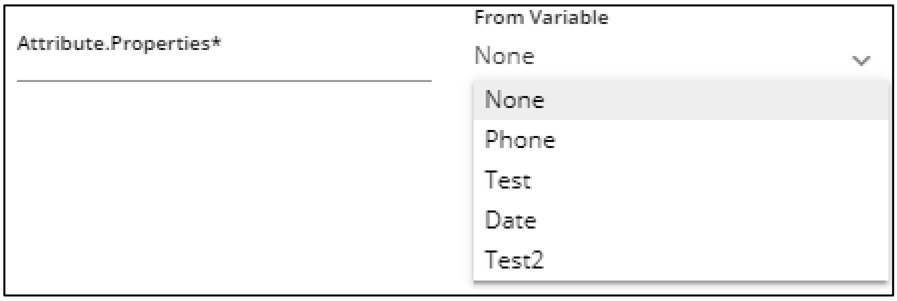
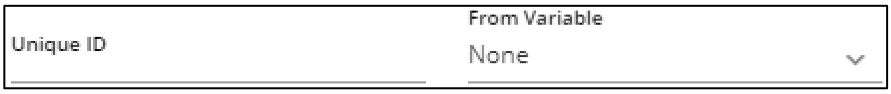


Figure: Attribute Properties

   If the value of the additional parameter is being provided in both text field and drop-down menu, then the variable selected in the drop-down menu will have more priority and will be used by default.

3. **Override Existing Data (Optional):** It lets you override the existing data that has been packed. You can select "Yes" to override the data, whereas select "No" to not override data.

4. **Unique ID (Optional):**.



Unique ID

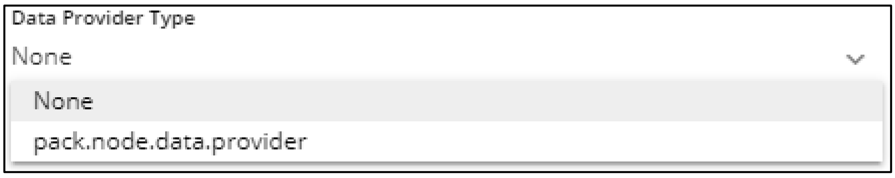
From Variable

None

Figure: Unique ID Attribute

Even if the value of a variable is being provided in both the text field and the drop-down menu, still the variable selected in the drop-down menu will have more priority and will be used by default.

5. **Data Provider:** This node contains the following data provider. Select it.



Data Provider Type

None

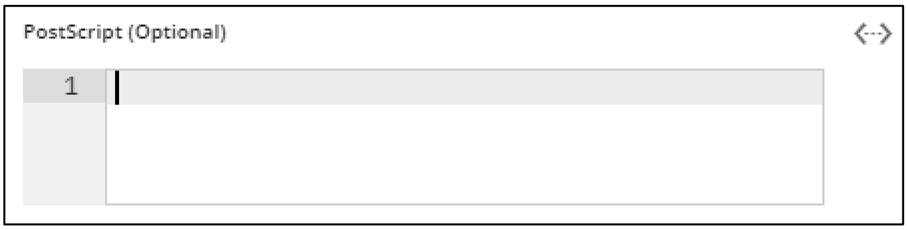
None

pack.node.data.provider

Figure: Data Provider of Pack Node

- pack.node.data.provider: It is the default data provider of this node. It is used to get the properties, uniqueId, shouldOverrideExistingData.

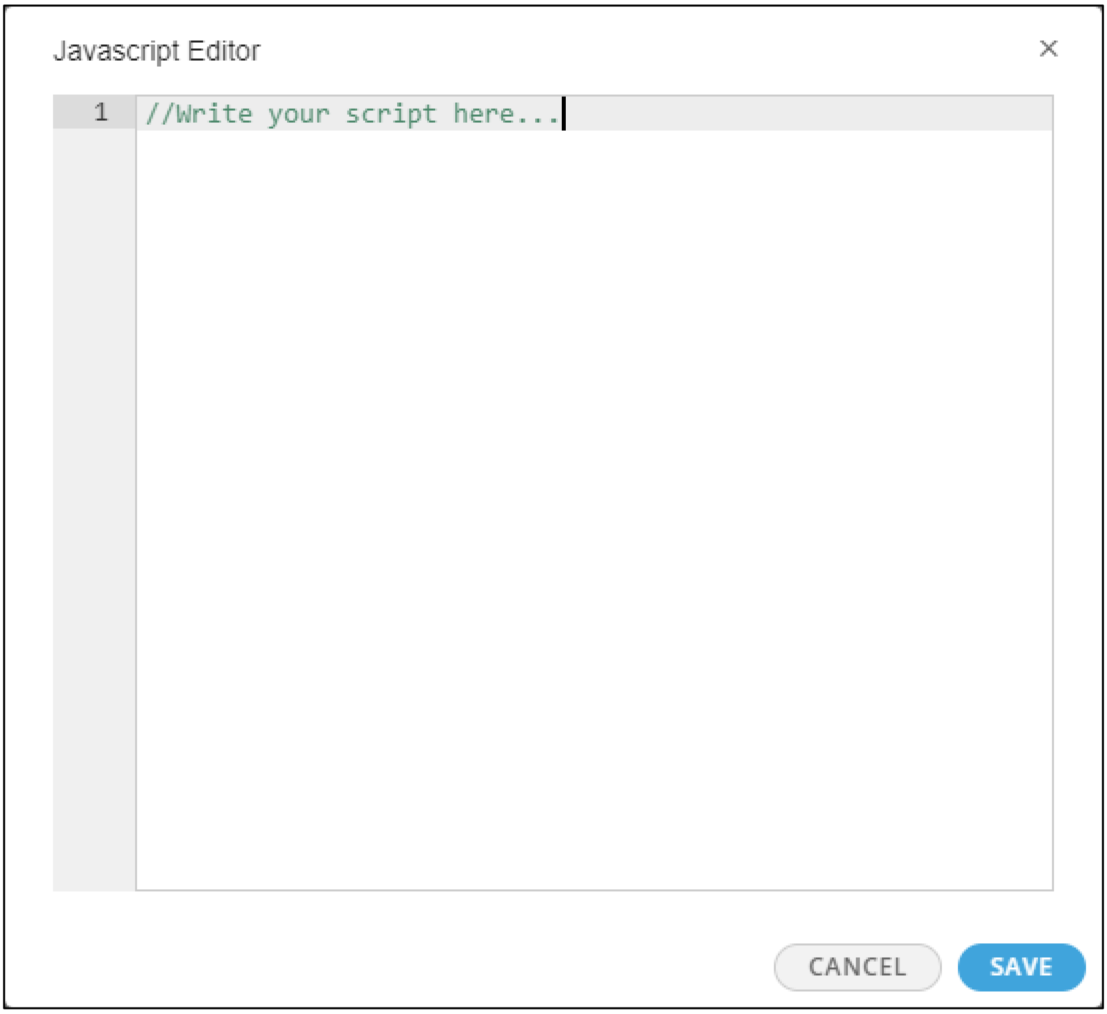
6. **Postscript (Optional):** Synthesizer allows to run a JavaScript code after the processing of the node.



PostScript (Optional)

Figure: Script to run after processing "Pack" Node

Here, paste the JavaScript. Click <--> icon to open the following JavaScript Editor.



```
//Write your script here...
```

Figure: JavaScript Editor

The Administrator can write its own code here and click "Save". It returns to node configuration, which will show the saved code.

7. **PreScript (Optional):** Specify to run a custom JavaScript code before processing the node. Like PostScript, add the code in Prescript.
8. Click "Save" to save the Node Configuration.

10.10.3 Events

Switch to "Events" tab to see the events of this node.

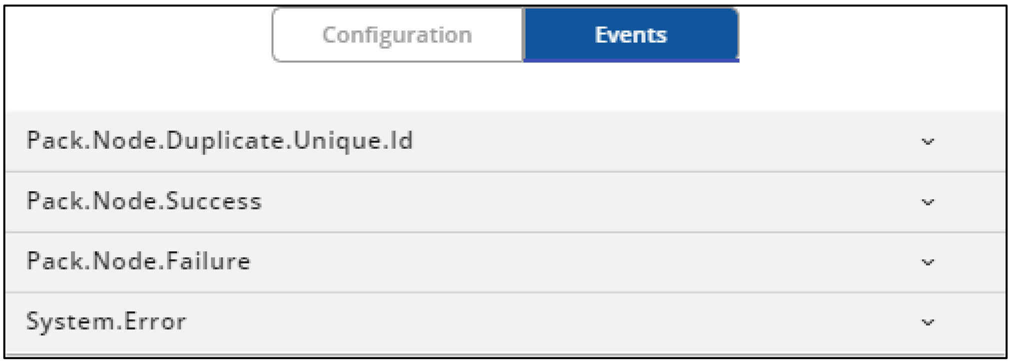


Figure: Events of Pack Node

These events are listed hereinbelow.

- **Event.Pack.Node.Duplicate.Unique.ID:** It is generated when shouldOverrideExistingData attribute is false and Unique ID supplied already exists in the persistence service.
- **Event.Pack.Node.Success:** It is generated when node executes successfully, that is, the provided properties attribute is successfully saved in persistence service corresponding to the Unique ID.
- **Event.Pack.Node.Failure:** It is generated when the node fails because of Application Service failures.
- **Event.System.Error:** It is generated when the system generates certain errors such as hardware, environmental, and others.

10.10.4 Output Script Variables

Following is the output variable, which is generated after the processing of this node.

- **packNodeUniqueId:** It is the unique ID, which corresponds to the packed and saved data.

10.11Start Monitor Node

It is used to start the recording of voice calls after the call is started. The node can be placed anywhere in the call stage when a call leg (Call Leg of agent or Call Leg of customer) is connected.

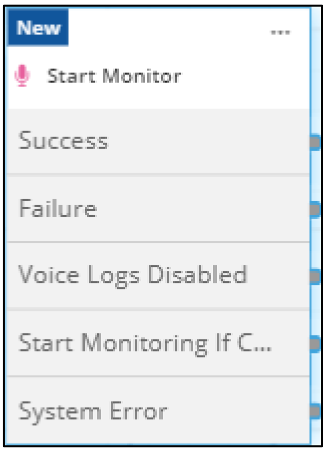
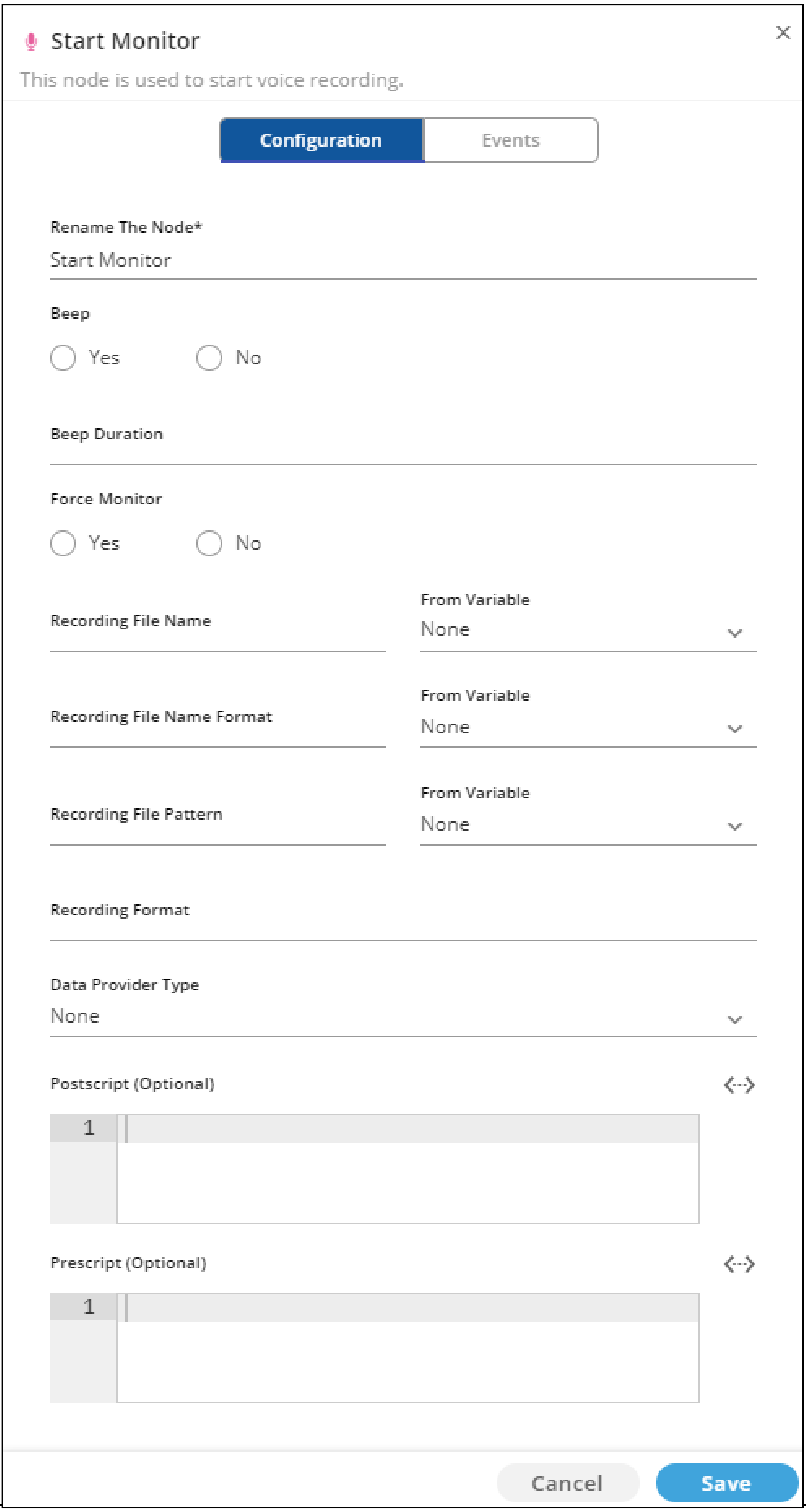


Figure: "Start Monitor" Node

Following is a screenshot of its configuration.



Start Monitor

This node is used to start voice recording.

Rename The Node*
Start Monitor

Beep
○ Yes ○ No

Beep Duration

Force Monitor
○ Yes ○ No

Recording File Name
From Variable
None

Recording File Name Format
From Variable
None

Recording File Pattern
From Variable
None

Recording Format

Data Provider Type
None

Postscript (Optional)

Prescript (Optional)

Figure: Configuration of "Start Monitor" Node

10.11.1 Availability in Nodeflow Types

This node is available in the following Nodeflow types.

- Callback
- Customer-based Inbound
- Conference
- Dialing
- Inbound
- Manual Dial

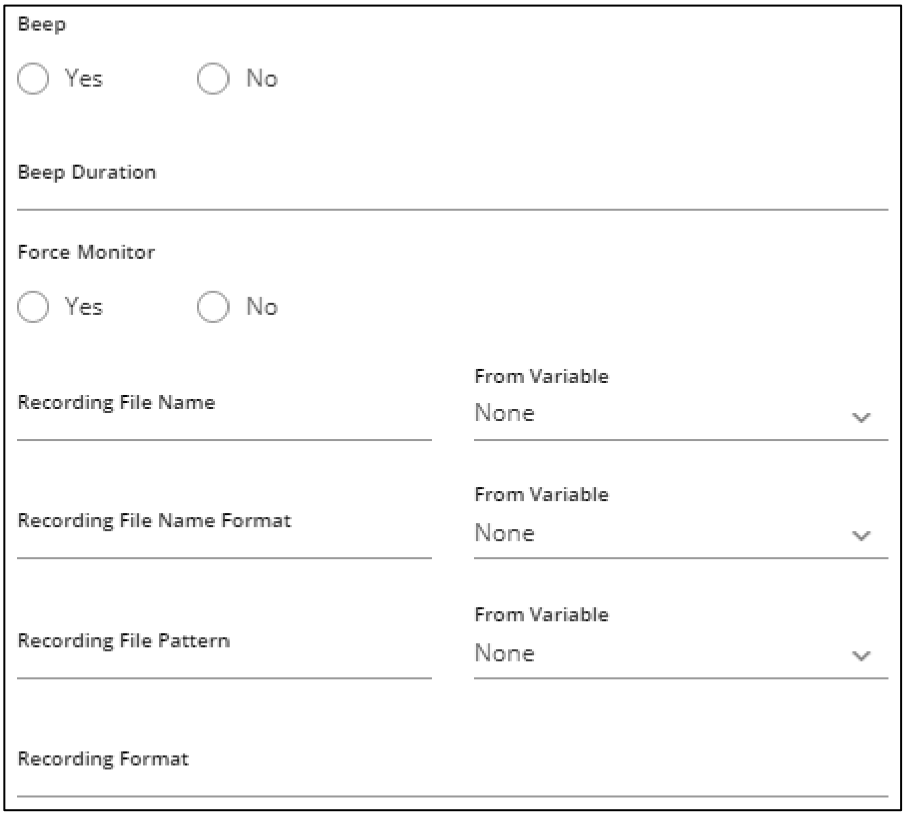
10.11.2 Configuration

Perform the following steps to configure this node.

1. **Rename The Node:** Rename the node, if required.

   Provide an easy-to-recall name, which matches the default node name.

2. **Beep (Optional):** Select "Yes" for this option, if you want to play the beep during the call to indicate the user and customer that this call is being recording.
3. **Beep Duration (Optional):** If you are playing "Beep", then this attribute allows you to configure its duration. You can enter the beep duration in seconds.
4. **Force Monitor (Optional):** Select "Yes" for this option to start the monitoring forcefully even if the monitoring of the current call is already going on.
5. **Recording File Name (Optional):** Here, you can provide the full path and name for the file that will store its recording.



Beep

○ Yes ○ No

Beep Duration

Force Monitor

○ Yes ○ No

Recording File Name — From Variable: None

Recording File Name Format — From Variable: None

Recording File Pattern — From Variable: None

Recording Format

Figure: Attributes of "Start Monitor" Node

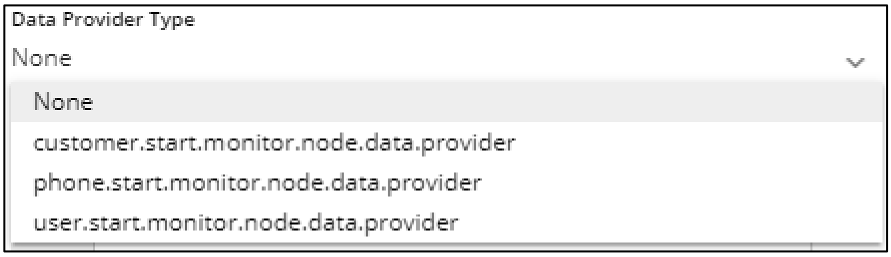
6. **Recording File Format:**
7. **Recording File Pattern (Optional):** It refers to the pattern that is placed as a tag on the voice records. For example, a voice log file can contain either of agent's name, agent's ID, campaign name, time, customer details, and call disposition.
8. **Recording Format (Optional):** Recording File Format is configured in the Campaign Settings at Campaign Level. By default, the recording file will be saved in "WAV" file format. Here, you can provide the name of file format in "String" to override the campaign level voicelog settings.

Following is the list of priority options for voicelog file name format. The items listed on top has more priority.

- Recording File Name
- Recording File Format
- Recording File Pattern

- System Level Configured Settings
- Default Values

9. **Data Provider:** This node contains the following data providers. Select anyone of them.

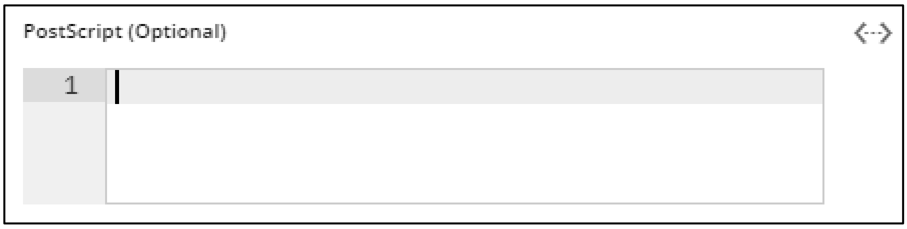


Data Provider Type
None
None
customer.start.monitor.node.data.provider
phone.start.monitor.node.data.provider
user.start.monitor.node.data.provider

Figure: Data Provider of "Start Monitor" Node

- customer.start.monitor.node.data.provider: It records the input and output voice streams of customer. During hold only customer voice and hold music will be captured. If during hold agent is in conference with third-party (user or phone) then neither agent's voice nor third-party voice is recorded.
- phone.start.monitor.node.data.provider: It is used when we want to record the Input and output stream of phone.
- user.start.monitor.node.data.provider: It records the input and output streams of user agent. During hold only agent's voice will be captured. If during hold the agent is in conference with third party (user or phone), then communication of agent with third party will also be captured.

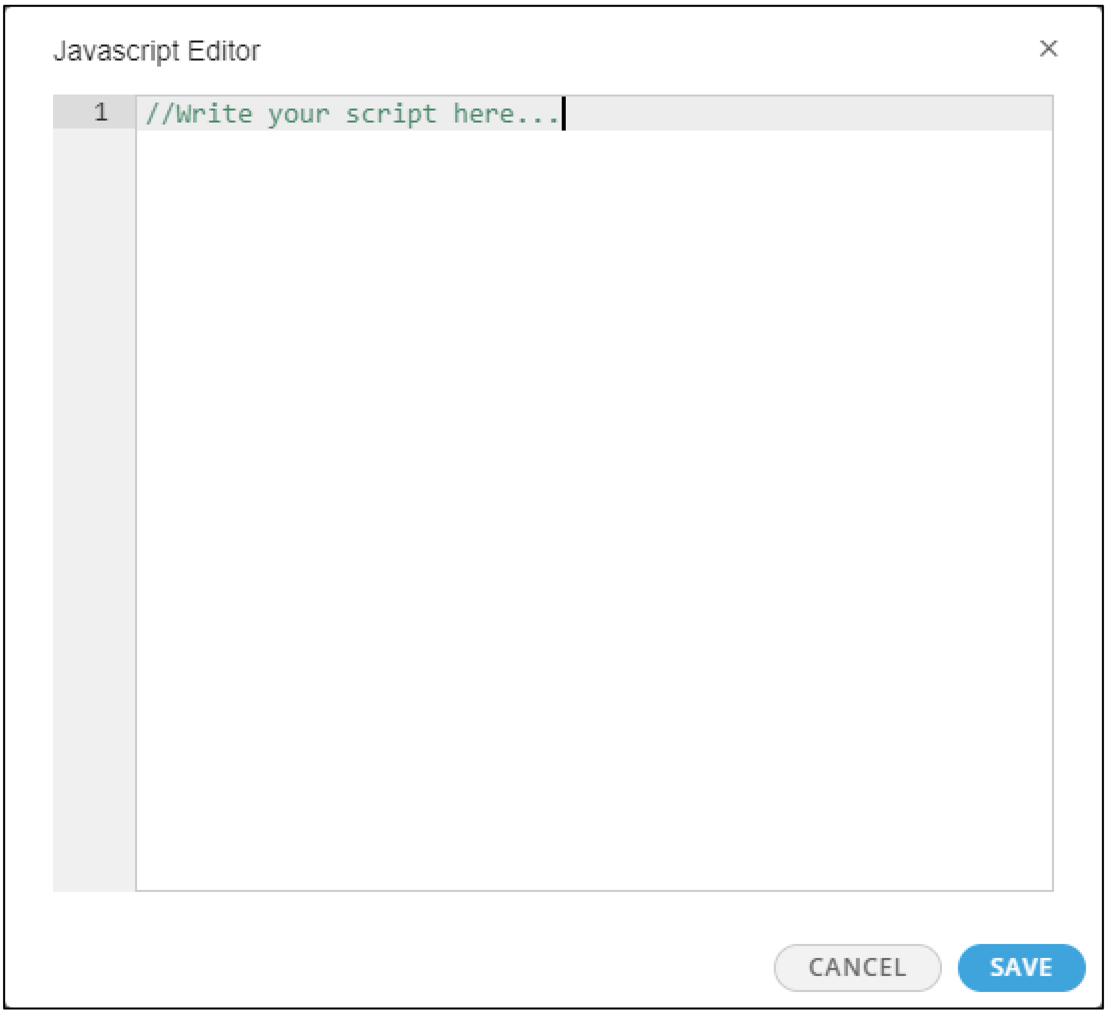
10. **Postscript (Optional):** Synthesizer allows to run a JavaScript code after the processing of the node.



PostScript (Optional)

Figure: Script to run after processing "Start Monitor" Node

Here, paste the JavaScript. Click <···> icon to open the following JavaScript Editor.



```
//Write your script here...
```

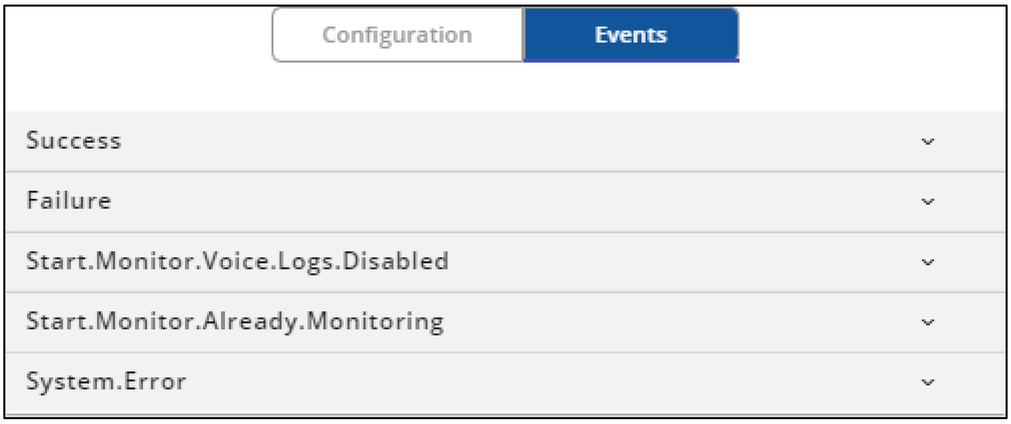
Figure: JavaScript Editor

The Administrator can write its own code here and click "Save". It returns to node configuration, which will show the saved code.

11. **PreScript (Optional):** Specify to run a custom JavaScript code before processing the node. Like PostScript, add the code in Prescript.
12. Click "Save" to save the Node Configuration.

10.11.3 Events

Switch to "Events" tab to see the events of this node.



Configuration | Events

Success

Failure

Start.Monitor.Voice.Logs.Disabled

Start.Monitor.Already.Monitoring

System.Error

Figure: Events of "Start Monitor" Node

These events are listed hereinbelow.

- **Event.Success:** It is generated when Start Monitor Node Successfully send the monitoring event to the Call server no matter recording has successfully done or not or even started on call server.
- **Event.Failure:** It is generated when the Call Leg for Call Runtime object is not connected because of any of the following reasons.
  - Call get disconnected before Start Monitor event being sent to call server.
  - Asterisk gets disconnected before Start Monitor event being sent to it.
- **Event.Start.Monitor.Voice.Logs.Disabled:** It is generated when the voicelogs in Ameyo are disabled.
- **Event.Start.Monitor.Already.Monitoring:** It is generated when the call is already being monitored.
- **Event.System.Error:** It is generated because of the reasons.
  - Invalid model for start monitor node
  - Invalid record file for start monitor node
  - Invalid Call Runtime Object to monitor
  - Invalid data provider for start monitor node

- The system generate certain errors such as hardware, environmental, and others.

10.11.4 Output Script Variables

After running, "Start Monitor" Node generates the following Output Script Variable.

- **recordingFileUrl:** It contains the path of the voicelog created on call server.

10.12Sync ACD Node

It distributes the call to the agents at the campaign level, not at queue level. Unlike ACD, it reserves the agent synchronously, that is, this node takes transition only after the request is served at the moment and does not wait for the agents to get free until timeout. This resource always reserves a specific resource determined by the attributes.

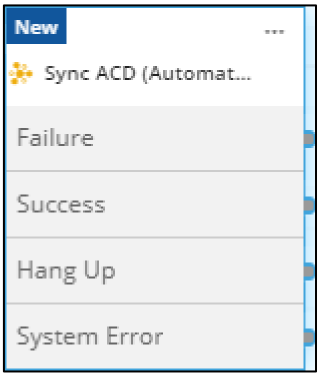
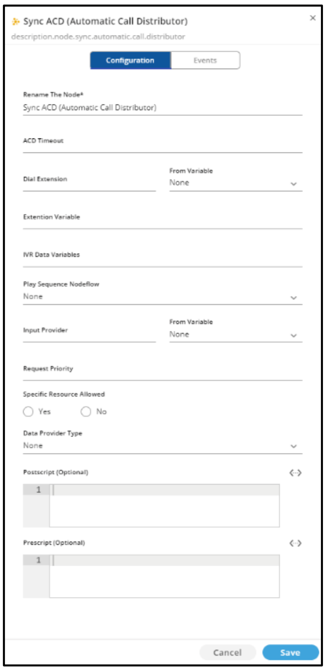


Figure: "Sync ACD" Node

With data provider, this node identifies the provided extension or agent and reserves it for the call.

Following is the screenshot of its configuration.



Sync ACD (Automatic Call Distributor)

description.node.sync.automatic.call.distributor

Rename The Node*: Sync ACD (Automatic Call Distributor)

ACD Timeout:

Dial Extension: | From Variable: None

Extention Variable:

IVR Data Variables:

Play Sequence Nodeflow: None

Input Provider: | From Variable: None

Request Priority:

Specific Resource Allowed: ○ Yes ○ No

Data Provider Type: None

Postscript (Optional)

Prescript (Optional)

Figure: Configuration of "Sync ACD" Node

10.12.1 Availability in Nodeflow Types

This node is available in the following Nodeflow types.

- Callback
- Dialing
- Inbound

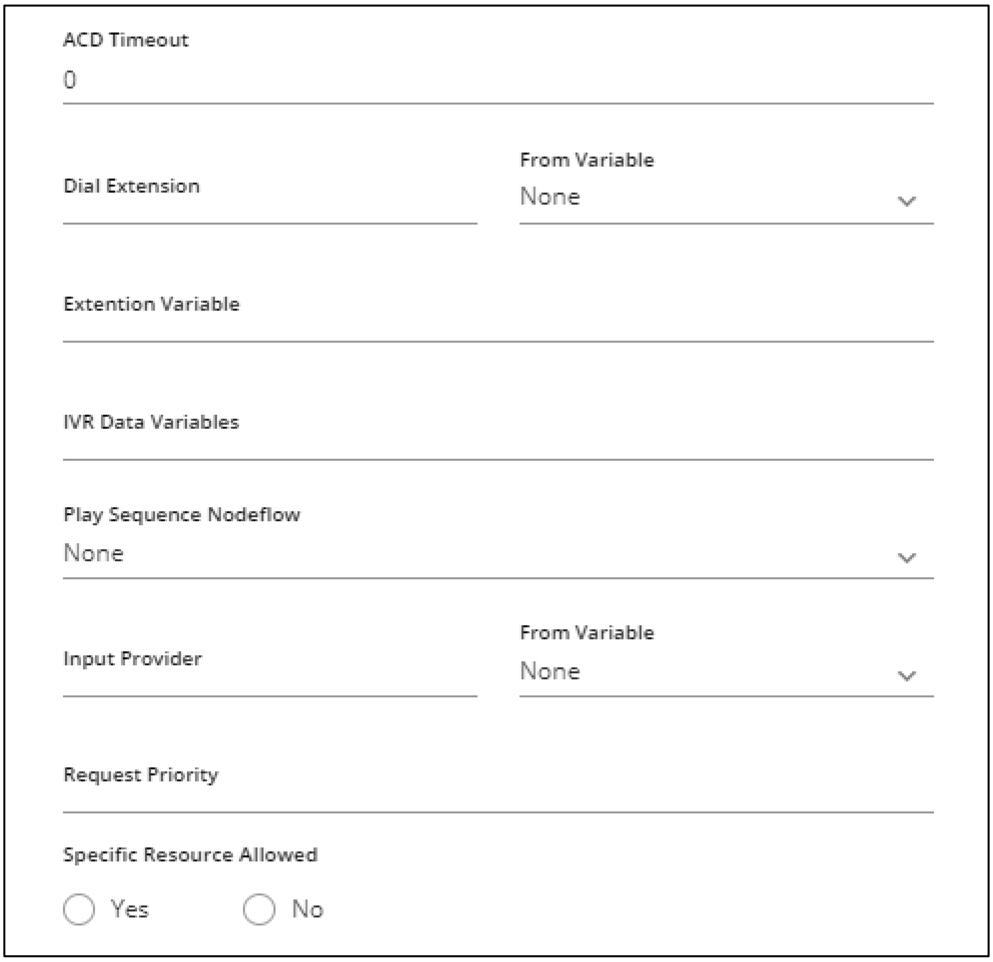
10.12.2 Configuration

Perform the following steps to configure this node.

1. **Rename The Node:** Rename the node, if required.

   Provide an easy-to-recall name, which matches the default node name.

2. **ACD Timeout (Optional, but Required):** It is the specified time(mandatory) after which ACD node will take transition whether resource is reserved or not. Enter its value in "String" format.



ACD Timeout

0

Dial Extension

From Variable

None

Extention Variable

IVR Data Variables

Play Sequence Nodeflow

None

Input Provider

From Variable

None

Request Priority

Specific Resource Allowed

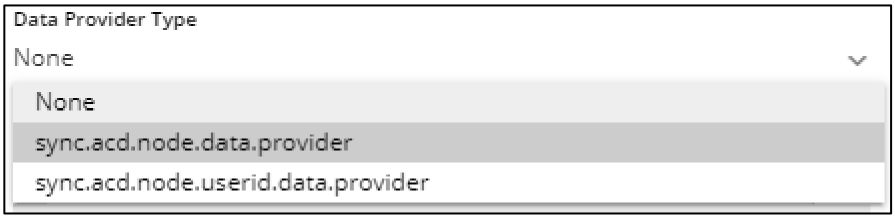
○ Yes ○ No

Figure: Attributes of Sync ACD Node

3. **Dial Extension (Optional):** It is used to determine the extension of a specific user. It should be same as that of the extension column corresponding to user in "contact_center_user_extension" table. Enter its value in "String" format.
4. **Extension Variable:**
5. **IVR Data Variables:**
6. **Play Sequence Nodeflow:**

   It is used to play the prompt while the resource is being served.
7. **Input Provider (Optional):** It accepts "String" type value. It is the User ID of the user, of which allocation is to be requested for this call. It is used when "SyncACDUserIdDataProvider" is used.

8. **Request Priority:** It is used to determine the request priority. It accepts "Integer" type value.
9. **Specific Resource Allowed (Mandatory):** t specifies whether a specific resource is to be requested or not. Enter its value in "boolean" format such as "TRUE" or "FALSE". Though, in "Sync ACD" Node specificeResourceAllowed is always true, irrespective of supplied value.
10. **Data Provider:** This node contains the following data providers, with which it identifies the provided extension or agent and reserves it for the call. Select any one of them.



Data Provider Type

None

None
sync.acd.node.data.provider
sync.acd.node.userid.data.provider

Figure: Data Provider of Sync ACD Node

- sync.acd.node.data.provider:
- sync.acd.node.userid.data.provider: It is used if call is to be served at campaign level and a specific user is to be requested for call, that user's id is to be provided in mandatory parameter "Input Provider".

11. **Postscript (Optional):** Synthesizer allows to run a JavaScript code after the processing of the node.

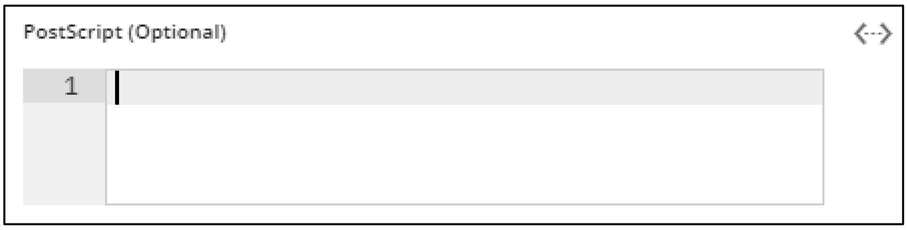
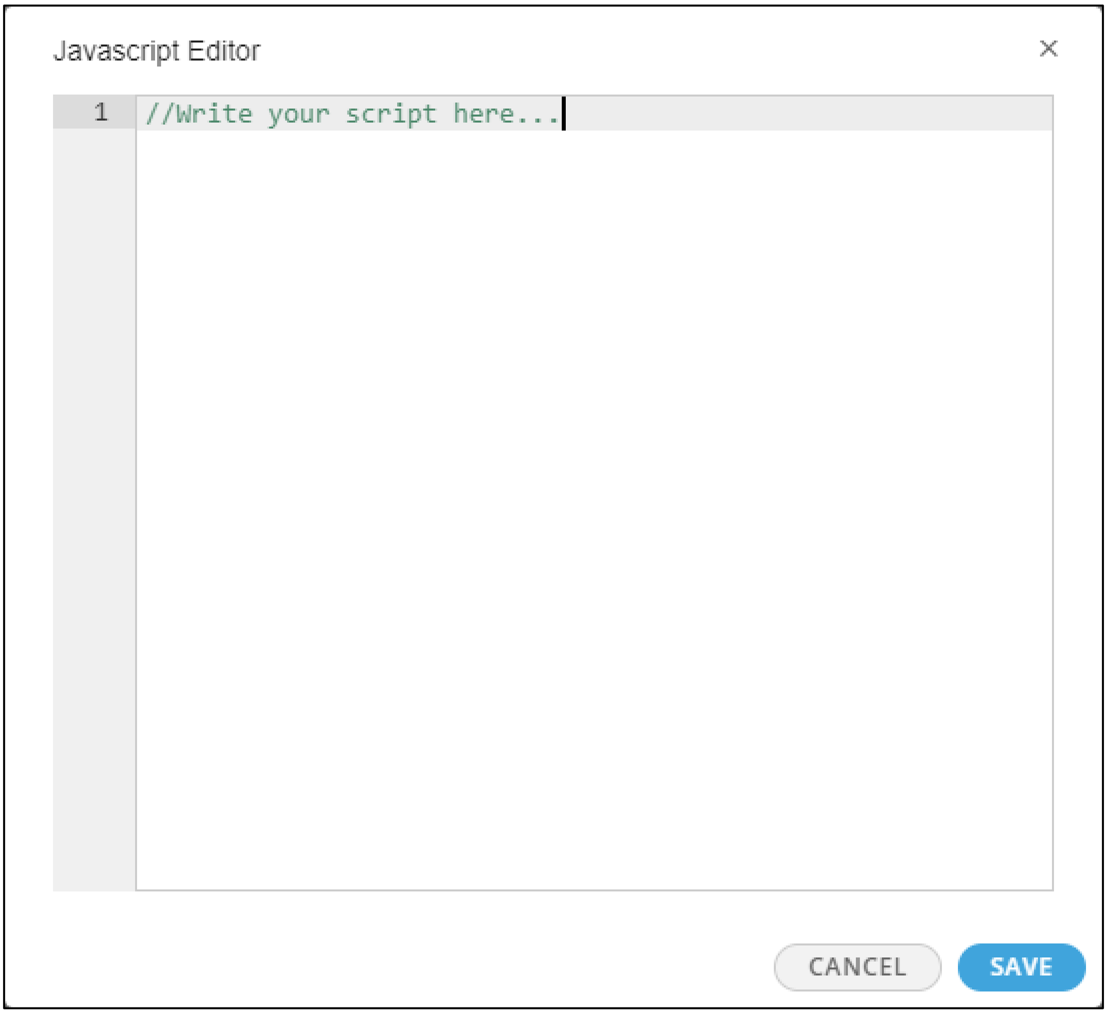


Figure: Script to run after processing "Sync ACD" Node

Here, paste the JavaScript. Click <--> icon to open the following JavaScript Editor.



```
//Write your script here...
```

Figure: JavaScript Editor

The Administrator can write its own code here and click "Save". It returns to node configuration, which will show the saved code.

12. **PreScript (Optional):** Specify to run a custom JavaScript code before processing the node. Like PostScript, add the code in Prescript.
13. Click "Save" to save the Node Configuration.

10.12.3 Events

Switch to "Events" tab to see the events of this node.

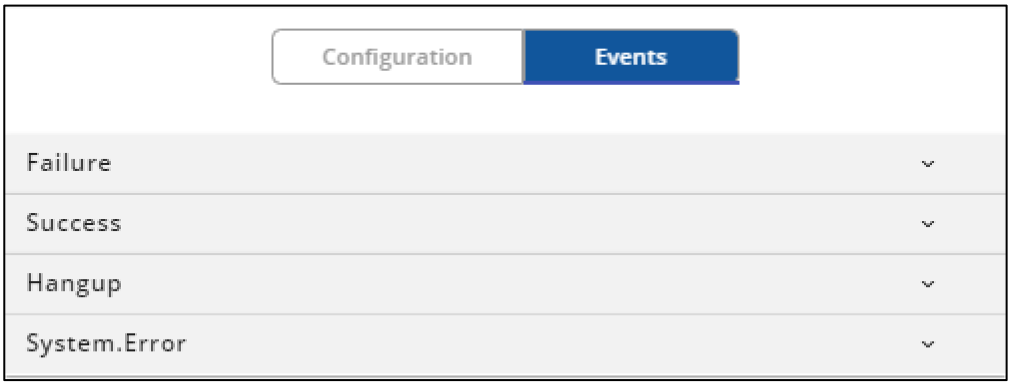


Figure: Events of Sync ACD Node

These events are listed hereinbelow.

- **Event.Timeout:**
- **Event.Failure:** It is generated when the node fails because of any invalid value set in the attributes or other application failures.
- **Event.Hangup:** It is generated when the call is disconnected while processing this node.
- **Event.Success:** It is generated when this node successfully processed, that is, the agent is successfully reserved for the call.
- **Event.Child.Initiated.Exit:** This event does not exist.
- **Event.System.Error:** It is generated when the system generate certain errors such as hardware, environmental, and others.

10.13Unpack Node

It fetches the packed and saved data from the persistence service such as Pack Node based on the Unique ID. The saved data can be from same nodeflow or another nodeflow.

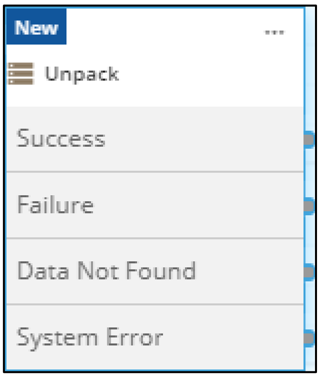
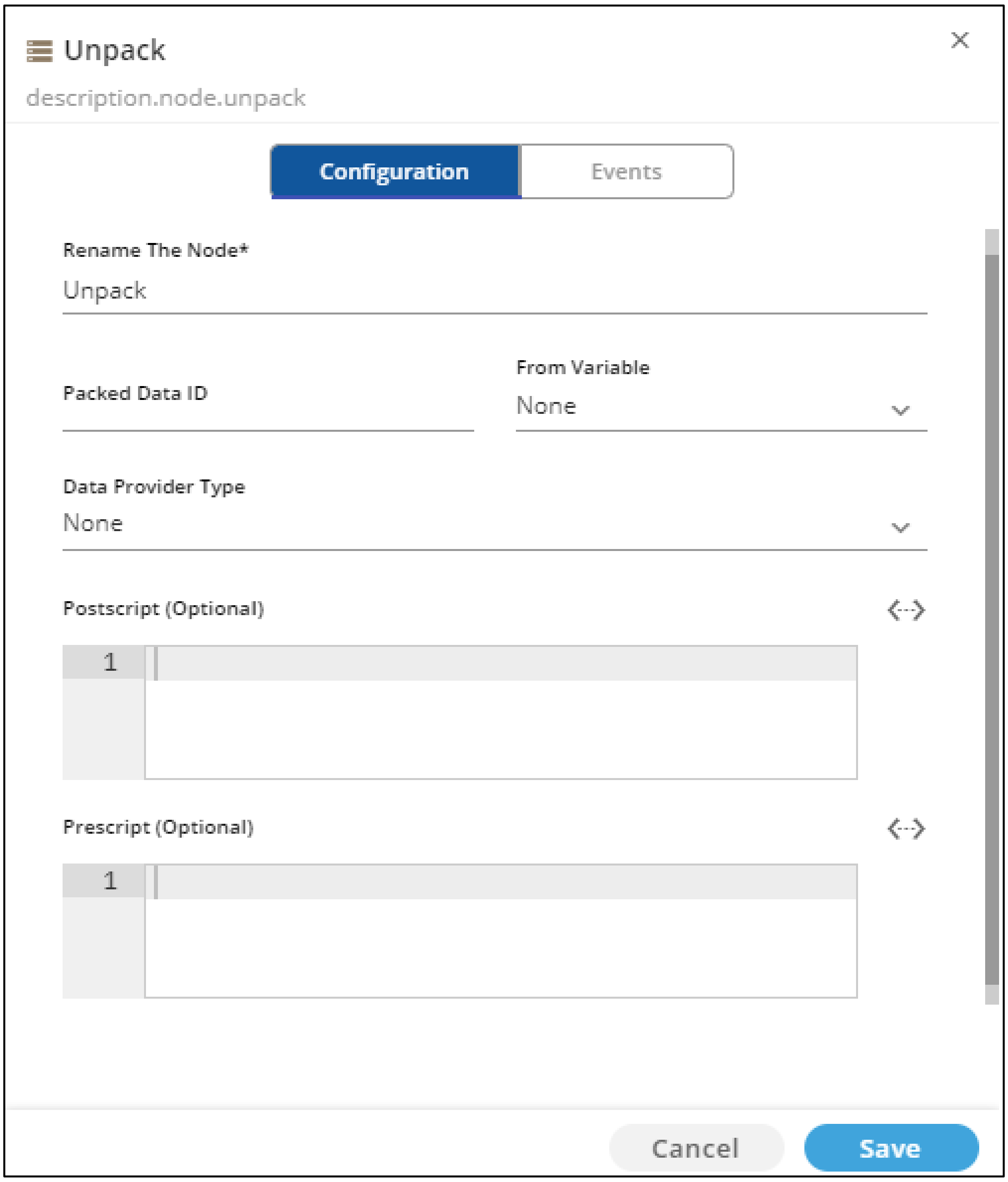


Figure: "Unpack" Node

Following is a screenshot of its configuration.



Unpack

description.node.unpack

Rename The Node*: Unpack

Packed Data ID:

From Variable: None

Data Provider Type: None

Postscript (Optional)

Prescript (Optional)

Figure: Configuration of "Unpack" Node

10.13.1 Availability in Nodeflow Types

This node is available in the following Nodeflow types.

- Callback
- Customer-based Inbound
- Conference
- Dialing

- Disposition
- Inbound
- Manual Dial
- Post Call Processing

10.13.2 Configuration

Perform the following steps to configure this node.

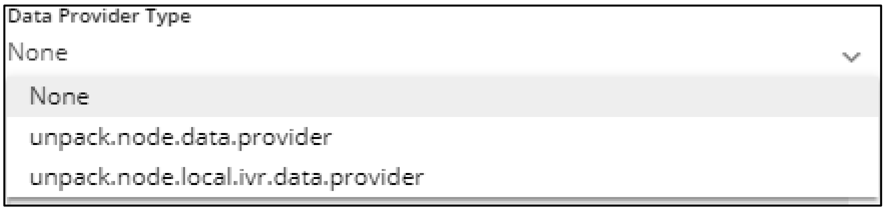
1. **Rename The Node:** Rename the node, if required.

   Provide an easy-to-recall name, which matches the default node name.

2. **Packed Data ID (Optional, but Required):** It's acceptable value type is "String". It is an optional variable, but required to unpack only that data which has been packed with a specific Unique ID. Provide its value.

   If the value of the additional parameter is being provided in both text field and drop-down menu, then the variable selected in the drop-down menu will have more priority and will be used by default.

3. **Data Provider:** This node contains the following data providers. Select anyone of them.



Data Provider Type: None

None
unpack.node.data.provider
unpack.node.local.ivr.data.provider

Figure: Data Providers of Unpack Node

- unpack.node.data.provider: It is used to get the unique ID for the requested data. If the unique ID is NULL, it returns srcPhone (Source Phone).
- unpack.node.local.ivr.data.provider: It is used to get the Unique ID for the requested data. The unique ID is retrieved from packNodeUniqueId key in the property map of calllegDetail.

4. **Postscript (Optional):** Synthesizer allows to run a JavaScript code after the processing of the node.

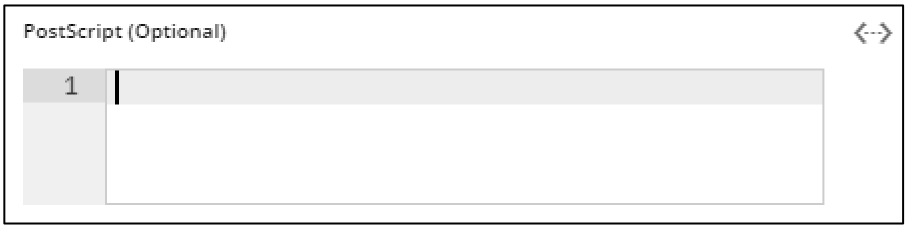
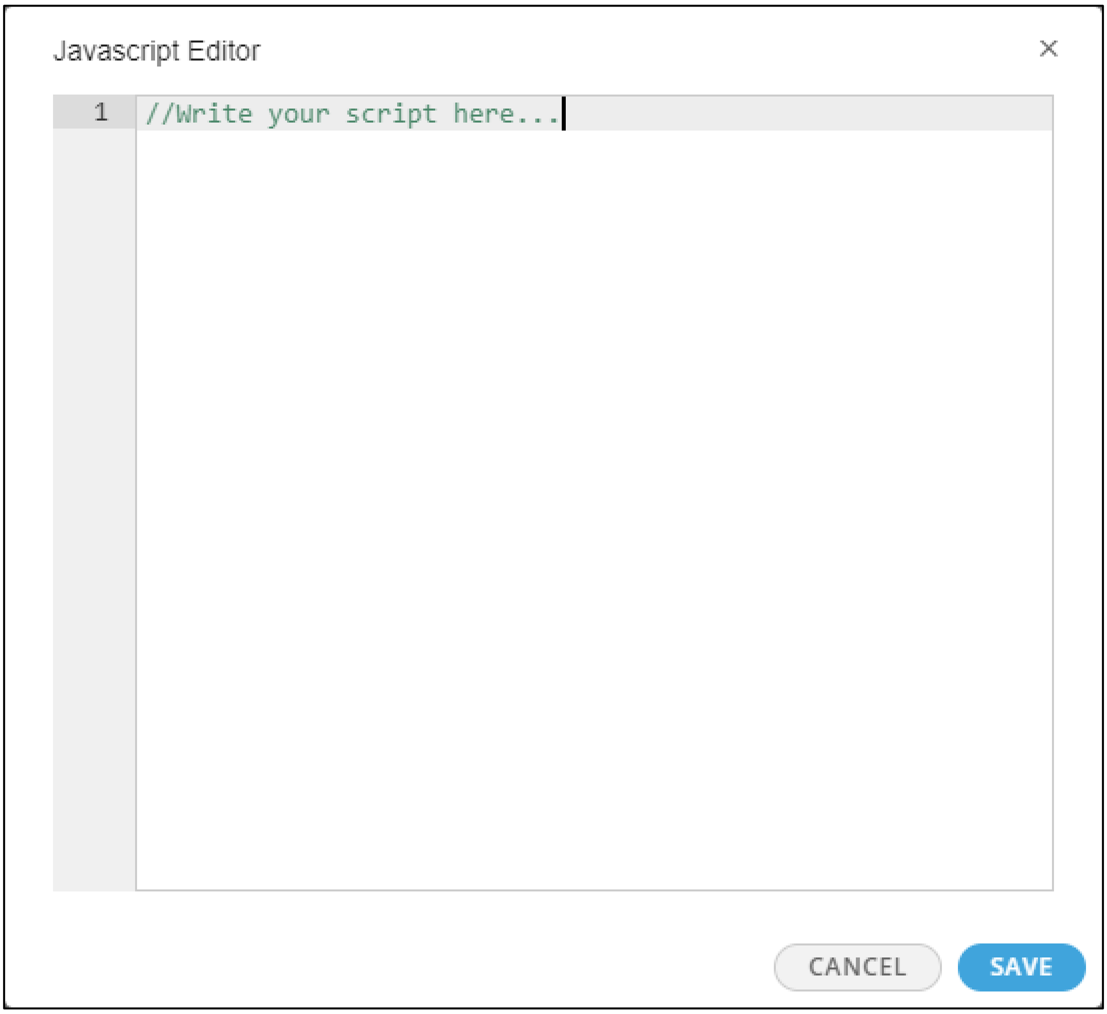


Figure: Script to run after processing "Unpack" Node

Here, paste the JavaScript. Click <-> icon to open the following JavaScript Editor.



```
//Write your script here...
```

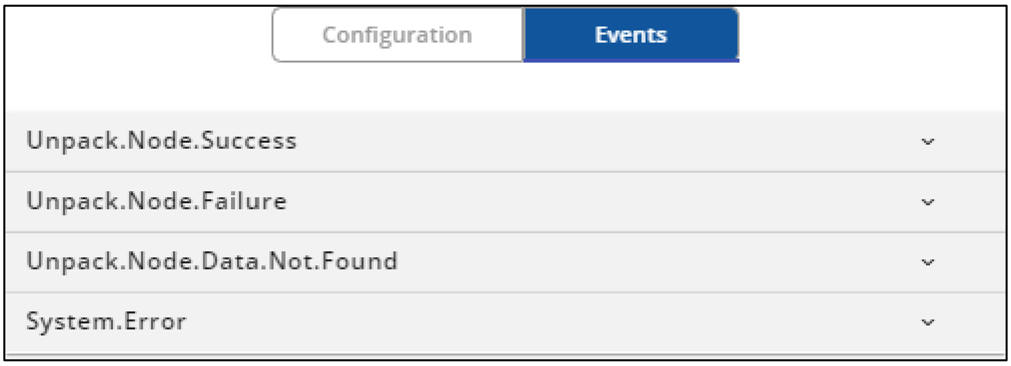
Figure: JavaScript Editor

The Administrator can write its own code here and click "Save". It returns to node configuration, which will show the saved code.

5. **PreScript (Optional):** Specify to run a custom JavaScript code before processing the node. Like PostScript, add the code in Prescript.
6. Click "Save" to save the Node Configuration.

10.13.3 Events

Switch to "Events" tab to see the events of this node.



Unpack.Node.Success

Unpack.Node.Failure

Unpack.Node.Data.Not.Found

System.Error

Figure: Events of "Unpack" Node

These events are listed hereinbelow.

- **Event.Unpack.Node.Success:** It is generated when the node executes successfully, that is, the data is sucessfully found and retrieved from the persistence service (like Pack Node) and is injected into the output script variables.
- **Event.Unpack.Node.Failure:** It is generated when the node fails because the provided unique ID is NULL or other application failure occurs.
- **Event.Unpack.Data.Not.Found:** It is generated when no data exists in persistence service corresponding to the specified unique ID.
- **Event.System.Error:** It is generated when the system generates certain errors such as hardware, environmental, and others.

10.13.4 Output Script Variables

Following is the output variable, which is generated after the processing of this node.

- **packNodeProperties:** It contains the retrieved JSON data as string. For example,

```
dataVar = '{"var1":"val1","var2":"val2"}'
```