1. Desktop Automation
   1. KeyBoard ShortCuts
   2. Drag And Drop
   3. Open App
   4. Close App
   5. Switch App
2. Web Automation
3. Open Webpage
4. Show
5. Snop
6. Dom
7. General
   1. Enter Keystrokes
   2. Echo
   3. Message Box
   4. Get Secret
   5. Get Credentials
   6. Read
   7. Wait
   8. HTTP API
   9. Assign

4) Module

1. Add External Flow

5) Mouse

1. Click
2. Right Click
3. Double Click
4. Hover
5. Select

6)Conditions

1. If
2. IfElese
3. ElseIf

7) Exception handling

A) Try Catch

8) Excel

1. Open Spreadsheet
2. B)Get SheetNames
3. Get Column
4. Delete Column
5. Get Row
6. Delete Row
7. G)Get Cell
8. H)Write Cell
9. Get Range
10. Write Range
11. Append Range
12. Delete Range
13. Get Cell Color
14. Set Range Color
15. Get Cell Formula
16. Create pivot Table
17. Convert Xls to Xlsx

9) Loops

1. For

10) Jump Statements

1. Statements

11) OCR

1. Techforce OCR

12) Files And Folders

1. Create Files /Folders
2. Write To File
3. Copy File/Folder
4. Get Folder Contents
5. Delete File/Folder
6. Read File

13) Secure FTP

1. Connect
2. Get File
3. Put File
4. Get File
5. Put Files
6. Delete

14) CSV

1. CSV Header
2. CSV Row

**Programming**

15) String

1. Matches

16) Technologies

1. JavaScript
2. Python
3. R
4. AutoHotkey

**Mail Integrtion**

**17 )** Outlook

1. Config
2. Read Mail
3. Send Mail

18) Imap

1. Read Mail

19) POP3

1. Read Mail

20 ) Outlook 2016

1. Read Mail
2. Send Mail

21 ) Save Attachments

22 ) Cognitive

1. Super Resolution
2. Image Classification
3. Face Similarity

**Desktop Automation**:

**Keyboard Shortcuts**

Used to send keyboard shortcuts to active application in which functions like CTRL, WIN, ALT, and SHIFT have been assorted in the IDE.

**Properties**

* **Custom keys:** Provides a text box in which user need to type the function keys to be passed to active application.
* **Keys:** Gives out a list of function keys and all the functions available on keyboard click on keys and select the required function to be used on active application.

**Example:**

Requirement: Need to send (Control +Copy) and enter key using custom keys and keys.

**Control+ copy:** check in the ctrl box and click on Custom keys and type “c” in the text box **Enter key:** check in the Keys and click on drop down box where you will find drop down list of the entire keyboard functions choose enter and click on it.

**Drag and Drop**

Used to Drag/change the location of file/folder from one location to another by using image selector or location selector.

**Properties**

From

Input starting point

* **Image Selector:** Where bot enables image capturing mode to select the image of required file/folder to change.
* **Location selector:** Where bot enables location capturing mode to select the x and y coordinates of required file/folder.

To

Output destination

* **Image Selector:** bot allows user to select the required location in the image form where bot recognize the to file/folder or place in the image format.
* **Location selector:** similarly bot allows user to select the destination of the selected file/folder by using the x and y coordinates.

**Example**

Requirement: Need to change the location .docx file present on desktop from one location to another by using image selector in “FROM” and by using location selector in “TO”.

**From:** Check in theImage Selector and click on the insert screen shot where you will give access to select the image of required .docx file.

**To:** Check in the location selector and click on the find location where bot allows you to select the required location of destination by using x and y coordinates.

**Open App**

Opens Windows applications.

**Properties**

**File Path:** It is used to open windows applications by giving the file path of the application with extension like notepad.exe .Copy the path and paste it in the Open app properties window.

**Example**

Requirement: Need to open Google by using Open App.

Copy the full path of the Google and paste it in the properties window.

**Close App**

Closes the active applications.

**Properties**

**File Path:** Similar to the Open App, Close App also works on same formulae but in order to the file path.Just give the header of the existing app.

**Example**

Requirement: Need to close amazon website using Close App.

Copy the header of Amazon.in and paste it in the properties window.

**Switch App**

Switches between the active applications.

**Properties**

**File Path:** Switch app it used to a particular window or app when multiple windows are open. Just write down the header of the required app/window or variable and paste it in the properties window.

**Example**

Requirement: Need to switch app when two apps/windows are active.

**Input:** Just write down the heading of the required app/window or variable and paste it in the properties window.

**Web Automation**

Used to automate browser based applications.

**Open webpage**

Open website by passing the URL in the properties window as input withBack quote or else by creating the variable for URL in the variables section and pass the variable name in the properties.

**Properties:**

Google Chrome

**Page address:** Check in the Google box Pass the URL with back quote symbol (``) example: (`www.amazon.com`) in the page address properties window or else directly pass the variable.

Headless

**Page address:** Used to run a program in the background without any extension. Pass the URL with back quote symbol (``) example: (`www.flipkart.com`) in the page address properties window or else directly pass the variable.

**Example**

Requirement: Need to open webpage www.irctc.co.in by creating a variable.

**Input:** create a variable in the variables and select string for variable type and place the [www.irctc.co.in](http://www.irctc.co.in) in value.

Pass that variable into the page address property window.

.**Show**

Show action is used to display the text value in the logs.

**Properties**

**Element:** Pass the string or else create a variable in the variables and pass the variable name in back quote symbol(``) .Results will be displayed by the bot in the system logs.

**Example**

Requirement: Show hello world in the logs.

**Input:** Pass the variable/string in the property window.

**NOTE:** variables should be passed with back quote (``) like `apple`.

**Snap:**

Used to take screenshot of whole page or can customize to get specific image by passing the x path of the required html of windows

**Properties:**

Page

* **Image Name:** provide the image name in the property window and the image will be saved by the given name.

Custom

* **Xpath:** copy and paste the required image html path in the xpath properties window or pass the variable.
* **Image Name:** provide an image name in the property window and the image will be saved by the given name.

**Note**: By default images are saved in .png format with default name (snap1 snap2 like versa) .If user want to save the image to given name need to mention with the format also i.e.: .png or else each time bot over writes the existing image and save it in the same field.

Example: image.png

**Example**

Requirement: Capture the whole page of google.com.

**Page**

**Input:** Click on the page and pass the image name in the properties window as the captured image will be placed under the given file name.

**Custom**

**Input:** click on the page and paste the xpath of the variable to be captured and pass the image name as string/variable in the properties window.

**DOM:**

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page

When you want to access HTML elements with JavaScript, you have to find the elements first.

There are a couple of ways to do this:

**Changing HTML Elements:**

**Method Description**

document.getElementById (id) Find an element by element id

document.getElementsByTagName (name) Find elements by tag name

document.getElementsByClassName (name) Find elements by class name

**GENERAL AUTOMATION**

The General Activities pack contains all the basic activities used for creating automation projects. These activities enable our robots to:

* Getting the input to robots
* Printing the output in console
* Giving alert message to the user
* Getting credentials to communicate with orchestrator
* Read data from web
* Assigning the values to variables

**Enter Keystrokes**

Send keystrokes to a UI Element. Enter by using X-Path and by selecting Image.

**Properties**

To Web Element

* **Xpath:** The Xpath of the corresponding web element.
* **Text:** The text to be written in the specified UI element.

To Image

* **Insert Screenshot**: Capture the corresponding image which we need to enter our text.
* **Text:** The text to be written in the specified UI element.

**Note:**

If we give variable name instead of direct text in the text field. The variable name have to be placed between left quote (` `) command.

**Echo**

Prints a string or the value of a string variable to the Output Panel. Strings have to be placed between quotation marks.

**Properties**

Custom

* **Text/Variable:** The content to be written to the Output panel. This field only accepts Stings and String variables. Strings have to placed between quotation marks.

**Message Box**

Displays a message box with a given text with the button options. We can also set the time to present the message box.

**Properties**

* **Message**: The text to be displayed in the message box. We have to enter our own text in this box or we can pass variable
* **Time out**: The time to be display the message box. We can directly pass the variable.

**Note**

We have to pass the variables between the left quote command. Ex..(` `)

**Read**

Read the data’s from web by using x-path value of that data. It is only used to read values from web.

**Properties**

* **Html Element**: The x-path of that data which we need to read from web
* **Variable:** The variable that used to contain red value from web.

**Wait**

Waits for a specified amount of time before continuing the workflow.

**Properties**

* **Time in sec:** The amount of time(in seconds)you want the following activity to be delayed.

**Assign**

Allocates any values to a variable. It can be used to increment the value of a variable in a loop. For example, Sum up the value of two or more variables and assign the result to a different variable.

**Properties**

* **Custom/Variable**: The name of the variable to be assigned a value.
* **Custom/Variable (second field)**: The value you want to assign to the variable. It may be string value or we can pass variable.

**Get Secret**

Gets a specified secret value from secret vault in Orchestrator by using provided Asset Name and return a secret value which can be Text, Boolean, Integer.

**Properties**

Configuration

* **User Name:** User Name of the your orchestrator account.
* **Password:** Password of the your orchestrator account.
* **IP:** The orchestrator IP which you need to access.

After Configuration we can access our secret vault in Orchestrator by using Given Asset name and Can store the values in variable.

* **Asset Name:** The asset name which created in the orchestrator to contain secret value.
* **Variable:** The variable which use to store returned value from secret vault.

**Note**

Once you configured you with orchestrator, it will not ask configure again. Directly activity shows asset name and variable field

**Get Credential**

Gets a specified credential by using provided Asset Name and returns a username and a secure password.

**Properties**

Configuration

* **User Name:** User Name of the your orchestrator account.
* **Password:** Password of the your orchestrator account.
* **IP:** The orchestrator IP which you need to access.

After Configuration we can access our secret vault in Orchestrator by using Given Asset name and user name, password.

* **Asset Name:** The asset name which created in the orchestrator to contain secret value.
* **User Name:** Type the variable name here, the username which is stored in orchestrator secret vault will come to this variable automatically.
* **Password:** Type the variable name here, the password which is stored in orchestrator secret vault will come to this variable automatically.

**Http API**

Enables you to perform HTTP operations to a specified web API. When first dragging this activity to the Designer panel, the HTTP Request Wizard window is displayed, which provides an easier way of building requests and previewing server responses.

**Properties**

* **URL:** URL to hit the corresponding api.
* **Get:** one of the method to hit api. This will get response from corresponding api. Here we can add only headers.
* **Post:** one of the method to hit api. This will send json object and get response from corresponding api. Here we will add body(payload) along with header.
* **Put:** This request will create a new resource or replaces a representation of the target resource with request payload.
* **Delete**: This request use to delete the specified resource.

**Add External Flow**

* Drag the add external flow into the flow.
* This action will help to add sub flows into the main flow. For this we need to give the exact path to call the sub flow into the main flow**.**

**Mouse Actions**

**Click**

This click action we use to point the particular placed element or image.

Specifies the type of mouse click (single, double, up, down) used when simulating the click event. The mouse button (left, right, middle) used for the click action. By default, the left mouse button is selected.

**Properties**

In Click Action we have three different types of selectors.

HTML Selector

* **Selector:** HTML selector is nothing but the element X-Path.

Image Selector

* **Selector:**Image selector is nothing but the Image as indicator.

Location Selector

* **Selector:**Location selector Is nothing but the location of the element which is x and y axis.

**Right Click**

Drag Right Click Action Into the flow.

**Properties**

In Click Action we have three different types of Selectors.

HTML Selector

* **Selector:** HTML selector is nothing but the element X-Path.

Image Selector

* **Selector:**Image selector is nothing but the Image as indicator.

Location Selector

* **Selector:**Location selector Is nothing but the location of the element which is x and y axis.

**Double Click**

Drag Double click action into the flow.

Specifies the kind of click action and it’s used for open direct applications like SAP & Outlook etc.

**Properties**

In Click Action we have three different types of Selectors.

HTML Selector

* **Selector:** HTML selector is nothing but the element X-Path.

Image Selector

* **Selector:**Image selector is nothing but the Image as indicator.

Location Selector

* **Selector:**Location selector Is nothing but the location of the element which is x and y axis.

**HOVER**

Drag the Hover action into the flow. It can places the mouse cursor at particular place wherever we can select.

**Properties**

In Click Action we have three different types of Selectors.

HTML Selector

* **Selector:** HTML selector is nothing but the element X-Path.

Image Selector

* **Selector:**Image selector is nothing but the Image as indicator.

Location Selector

* **Selector:**Location selector Is nothing but the location of the element which is x and y axis.

**SELECT**

Drag select option into the flow.

In this select option we have only x-path of the selected value and option value (here option will be variable also).

**Select Variable:** Create one list variableand select that variable from dropdown. Here only list variable is allowed to select from dropdown.

**Exception Handling**

**Try Catch**

Drag the try catch block into the flow.

Whatever the actions involved in the flow we need add each and every step or action in the try block. Once we run the flow if any exception will come those are handled by catch block.

**Jump Statements**

We have two jump statements in our Techforce-IDE. Those are Break and Continue**.**

**Break**

Drag the statements action into the flow.

We use this statement only in for loop. If the iteration will stop for certain number of executions, then we will use this Break statement**.**

**Continue**

Drag the statements action into the flow.

We use this continue action in for loop for to do number of iterations will execute continuously.

**Files & Folders**

**Create File/Folder**

Creates a file/folder in the specified location.

**Properties**

File

* **File Path:** The full path of the file to be created with extension. The path can be given by assigning it to a variable using double slashes. For ex: path = C:\\Desktop\\*filename.Extension*. The variable should be included within `` while passing to file path.

Folder

* **Folder Path:** The full path of the folder to be created. The path can be given by assigning it to a variable using double slashes. For ex: path = C:\\Desktop\\*foldername.* The variable should be included within `` while passing to file path.
* **Overwrite:** The folder will be overwriting if there is any folder exists with the same name in the given path.
* **Skip:** The creation of folder will be skip if there is any folder exists with the same name in the given path.

**Write to File**

Appends/Overwrites the specified string or the data assigned to a variable to the specified file.

**Properties**

* **File Path:** The full path of the file. The path can be given by assigning it to a variable using double slashes. For ex: path = C:\\Desktop\\*filename.Extension*. The variable should be included within `` while passing to file path.
* **Content to Write:** The string/variable to be write to the file specified. If is a string, it can be given directly. If it is a variable, it should be included in ``.
* **Overwrites:** The file data will be overwriting with the string/variable data if there is any data exists in the file.
* **Append:** The file data will be appending with the string/variable data if there is any data exists in the file.

**Copy File/Folder**

Copy/Move a file from one location to another location as specified.

**Properties**

File

* **Source Path:** The full path of the file to be copied/moved including the extension. The path can be given by assigning it to a variable using double slashes. For ex: path = C:\\Desktop\\*filename.Extension*. The variable should be included within `` while passing to file path.
* **Destination Path:** The full path of the location where the file to be copied/moved including the extension. The path can be given by assigning it to a variable using double slashes. For ex: path = C:\\Desktop\\*filename.Extension*. The variable should be included within `` while passing to file path.
* **Copy:** Make a copy of the file specified from the source path to destination path.
* **Move:** Moves the file specified in the source path to destination path.
* **Overwrite:** Overwrites the file while moving/coping the file specified if there is a file exists with the same name in the destination.
* **Skip:** Skips the coping/moving the file specified if there exists a file with the same name in the destination.

Folder

* **Source Path:** The full path of the folder to be copied/moved. The path can be given by assigning it to a variable using double slashes. For ex: path = C:\\Desktop\\*foldername.* The variable should be included within `` while passing to file path.
* **Destination Path:** The full path of the location where the folder to be copied/moved. The path can be given by assigning it to a variable using double slashes. For ex: path = C:\\Desktop\\*foldername.* The variable should be included within `` while passing to file path.
* **Copy:** Make a copy of the folder specified in the source path to destination path.
* **Move:** Moves the folder specified in the source path to destination path.
* **Overwrite:** Overwrites the folder while moving/coping the folder specified if there is a folder exists with the same name in the destination.
* **Skip:** Skips the coping/moving the folder specified if there exists a folder with the same name in the destination.

**Get Folder Contents**

Lists the contents of the folder based on the filter criteria.

**Properties**

* **Folder Path:** The full path of the folder where the contents to be retrieved. The path should be given using the forward slash (/). For ex: C:/Desktop/*FolderName*. If it is given by variable, it should be included in ``. for ex: `*variable name*`.
* **Variable:** Variable name to list the contents of the folder. The result will be stored in a list.

Which Contents to List

* **Folders:** Lists the folders in the folder specified in the **Folder Path.**
* **Files:** Lists the files in the folder specified in the **Folder Path.**
* **Filter:** Filters the files based on the given extension like *.png, .xls, .xlsx* etc.

Filter Criteria

* **Date Modified:** Filters the contents of the folder based on the date modified selected in the From Date and To Date fields.

**Delete File/Folder**

Deletes the file/folder specified.

**Properties**

File

* **File Path:** The full path of the file to be deleted including the extension. The path can be given by assigning it to a variable using double slashes. For ex: path = C:\\Desktop\\*filename.Extension*. The variable should be included within `` while passing to file path.

Folder

* **Folder Path:** The full path of the folder to be deleted. The path should be given using the forward slash (/). For ex: C:/Desktop/*FolderName*. If it is given by variable, it should be included in ``. for ex: `*variable name*`.

**Read File**

Read all the contents from the file specified and stores it into a variable specified.

**Properties**

* **File Path:** The full path of the file to be read including the extension. The path can be given by assigning it to a variable using double slashes. For ex: path = C:\\Desktop\\*filename.Extension*. The variable should be included within `` while passing to file path.
* **Variable:** Variable name to store the contents of the file specified.

**Conditions**

**If, If Else, elseif**

Provides an ability to route workflow execution to different step groups depending on conditions provided.

The If Else action compares values of two variables, or of a variable and a specific value and returns a Boolean result (True or False). If condition result is true, ‘Then’ block will execute. If condition result is false, ‘Else’ block will execute.

**Properties**

Custom

Checks the condition. If the condition met a block of code will execute. Otherwise else block will execute.

* **Variable:** Variable name that needs to compare with the value. The variable field also has a dropdown menu which contains url(), title(), text() to operate on a webpage.
* **Operator Dropdown:** Contains operators such as contains, not contains, equal to, not equal to, greater than, less than etc.
* **Value:** Value to compare with the variable. Value can be a string or a variable. If value is a string it should be included within double quotes (“”).

Present

* **HTML Selector:** Checks whether the given xpath exists on a webpage. If exists, a block of code will execute. Otherwise else block will execute.
* **Image selector:** Checks whether the given image exists on the current window. If exists, a block of code will execute. Otherwise else block will execute.

Visible

* **HTML Selector:** Checks whether the given xpath visible on a webpage. If visible, a block of code will execute. Otherwise else block will execute.
* **Image selector:** Checks whether the given image visible on the current window. If visible, a block of code will execute. Otherwise else block will execute.

Count

Checks the count of the xpath based on the condition. If the condition met, a block of code will execute. Otherwise else block will execute.

**Loops**

**For**

Loops are an efficient way to complete work via multiple number of iterations until the targeted result is achieved.

**Properties**

* **Element:** The variable that user wants to iterate.
* **Initial value:** The index value of the element to start with. In general, the index starts with zero.
* **Range:** The index value of the element to end with. To loop indefinitely use range value as infinity. This loops 1024 times as infinity variable is pre-set to 1024.

**Excel Actions**

**Note**: If we are passing the excel path by using variable the variable type should be string we need to mention if it is backward slash use double backward slashes (Eg: C:\\test\\excel123.xlsx), if it is forward slash then use single slash (Eg: C:/test/excel123.xlsx) for every excel Action then pass that variable with in tilt symbol (Eg: `VaribleName1`).

**Open spreadsheet**

Opens an Excel workbook in background mode and provides a scope for Excel Activities. When the execution of this activity ends, the specified workbook and the Excel application are closed. If a Workbook Application variable is provided in the Output. This activity can only be used if the Microsoft Excel application is installed on your machine.

**Properties**

* **Select an Excel File:** The full path of the Excel spreadsheet that you want to use. Only String variables and strings are supported. If you want to pass the excel workbook path through variable, then mention that variable name within tilt symbol in that field (Eg: `VaribleName`), otherwise click on File symbol and select that file.

**Note**: If we are passing the excel path by using variable the variable type should be string we need to mention if it is backward slash use double backward slashes (Eg: C:\\test\\excel123.xlsx), if it is forward slash then use single slash (Eg: C:/test/excel123.xlsx) for every excel Action then pass that variable with in tilt symbol (Eg: `VaribleName1`).

**Get Sheet Names**

Returns a list of all the sheet names in a workbook as String variables, ordered by their index (index value start from zero).

**Properties**

* **Enter Full Path:** Need to give the full path of the Excel spreadsheet that you want to use. This option is shows only you are taking the Get Sheet Names action directly. If we dragged into the open spreadsheet activity It will not ask the path, then no need to mention the path. .
* **Select Variable:** Create one list variableand select that variable from dropdown. Here only list variable is allowed to select from dropdown. The result value will be stored in list variable.

**Get column**

Reads the values from a column beginning with the cell specified in the Starting Cell property field, and stores them in a list variable.

**Properties**

* **Enter Full Path:** Need to give the full path of the Excel spreadsheet that you want to use. This option is shows only you are taking the Get Column action directly. If we dragged into the open spreadsheet activity It will not ask the path, then no need to mention the path.
* **Enter Sheet Name:** The name of the sheet in which the column that you want to read is. Only String variables and strings are supported (Eg: Sheet1).
* **Enter Cell Address:** The cell from which to start extracting the column data. Mention Column name A or B.
* **Select Variable:** Need to create the variable as list Stores the information from the specified spreadsheet column in a variable. Only list variables are supported.

**Delete Column**

Deletes a table column from a spreadsheet based on its name.

**Properties**

* **Enter Full path:** The full path of the Excel spreadsheet that you want to use. This option is shows only you are taking the Delete Column action directly. If we dragged into the open spreadsheet activity It will not ask the path, then no need to mention the path.
* **Enter Sheet Name:** The name of the sheet in which the column that you want to delete is. Only String variables and strings are supported (Eg: Sheet2).
* **Delete Column:** The exact number of the column that you want to delete. Only String variables and number values are supported.

**Get Row**

Reads the values from a row beginning with the cell specified in the Starting Cell field and stores it in a list variable.

**Properties**

* **Enter Full path:** The full path of the Excel spreadsheet that you want to use. This option is shows only you are taking the Delete Column action directly. If we dragged into the open spreadsheet activity It will not ask the path, then no need to mention the path.
* **Enter Sheet Name:** The name of the sheet in which the column that you want to delete is. Only String variables and strings are supported (Eg: Sheet1).
* **Get Row:** The exact row number that you want to read. Only String variables and number values are supported.
* **Select Variable:** Need to create the variable as list Stores the information from the specified spreadsheet row in a variable. Only list variables are supported.

**Delete Row**

Remove a specified row at a certain position.

**Properties**

* **Enter Full path:** The full path of the Excel spreadsheet that you want to use. This option is shows only you are taking the Delete Row action directly. If we dragged into the open spreadsheet activity It will not ask the path, then no need to mention the path.
* **Enter Sheet Name:** The name of the sheet in which the Row that you want to delete is. Only String variables and strings are supported (Eg: Sheet1).
* **Delete Row:** The exact number of the row that you want to delete. Only String variables and number values are supported.

**Get Cell**

Reads the values from a row beginning with the cell specified in the Starting Cell field and stores it in a string variable.

**Properties**

* **Enter Full Path:** The full path of the Excel spreadsheet that you want to use. This option is shows only you are taking the Get Cell action directly. If we dragged into the open spreadsheet activity It will not ask the path, then no need to mention the path.
* **Enter Sheet Name:** The name of the sheet in which the Cell that you want to read is. Only String variables and strings are supported (Eg: Sheet1).
* **Enter Cell Address:** The exact Cell address that you want to read. Only String variables and number values are supported.
* **Select variable:** Create the variable as string Stores the information from the specified spreadsheet Cell in a variable. Only string variables are supported.

**Write Cell**

Writes a value into a specified spreadsheet cell. If the sheet does not exist, a new one is created with the Sheet Name value. If a value exists, it is overwritten. Changes are immediately saved.

**Properties**

* **Enter Full Path:** The full path of the Excel spreadsheet that you want to use. This option is shows only you are taking the Write Cell action directly. If we dragged into the open spreadsheet activity It will not ask the path, then no need to mention the path.
* **Enter Sheet Name:** The name of the sheet in which the Cell that you want to write is. Only String variables and strings are supported (Eg: Sheet1).
* **Enter Cell Address:** The exact Cell address that you want to write. Only String variables and number values are supported.
* **Enter Cell Value:** Enter the cell value which you want to write. It allows String and String variable only. If passing the value through variable mention variable within tilt symbol (Eg: `VaribleName`).

**Get Range**

Reads the value of an Excel range and stores it in a DataTable variable. If the range isn't specified, the whole spreadsheet is read. If the range is specified for particular range, It reads the values from that specific range only.

**Properties**

* **Enter Full Path:** The full path of the Excel spreadsheet that you want to use. This option is shows only you are taking the Get Range action directly. If we dragged into the open spreadsheet activity It will not ask the path, then no need to mention the path.
* **Enter Sheet Name:** The name of the sheet in which the Cell that you want to read. Only String variables and strings are supported (Eg: Sheet1).
* **From Range:** Specifies the range of cells to be read. If this value is not specified, the whole spreadsheet is read. Only String variables and strings are supported to mention that range (Eg: B5:G15).
* **Select variable:** Create the variable as DataTable Stores the information from the specified spreadsheet range values in a variable. Only dataTable variables are supported select.

**Write Range**

Writes the data from a DataTable variable in a spreadsheet starting with the cell indicated in the **Starting Cell** field. If the starting cell isn't specified, the data is written starting from the A1 cell. If the sheet does not exist, a new one is created with the value specified in the **Sheet Name**property. All cells within the specified range are overwritten. Changes are immediately saved.

**Properties**

* **Enter Full Path:** The full path of the Excel spreadsheet that you want to use. This option is shows only you are taking the Write Range action directly. If we dragged into the open spreadsheet activity It will not ask the path, then no need to mention the path.
* **Enter Sheet Name:** The name of the sheet in which the range that you want to write. Only String variables (Eg: `VaribleName`) and strings are supported (Eg: Sheet1).
* **Write Range:** The cell from which to start writing the data. Only string variables and strings are supported (Eg: A1 or B1).
* **Select variable:** Select the variablethe data that you want to write to the specified range, as a DataTable variable. Only DataTable variables are supported.

**Append Range**

Adds the information stored in a DataTable variable to the end of a specified Excel spreadsheet. If the sheet does not exist, a new one is created with the name indicated in the Sheet Name field.

**Properties**

* **Enter Full Path:** The full path of the Excel spreadsheet that you want to use. This option is shows only you are taking the Append Range action directly. If we dragged into the open spreadsheet activity It will not ask the path, then no need to mention the path.
* **Enter Sheet Name:** The name of the sheet in which the range that you want to append. Only String variables (Eg: `VaribleName`) and strings are supported (Eg: Sheet1).
* **Select variable:** Select the variablethe data that you want to append to the specified range, as a DataTable variable. Only DataTable variables are supported.

**Delete Range**

Deletes a specified range in an Excel workbook.

**Properties**

* **Enter Full Path:** The full path of the Excel spreadsheet that you want to use. This option is shows only you are taking the Delete Range action directly. If we dragged into the open spreadsheet activity It will not ask the path, then no need to mention the path.
* **Enter Sheet Name:** The name of the sheet in which the range that you want to delete. Only String variables (Eg: `VaribleName`) and strings are supported (Eg: Sheet1).
* **From Range:** Specifies the range of cells to be delete. If this value is not specified, the whole spreadsheet is read. Only String variables and strings are supported to mention that range (Eg: B5:G15).