

RE Framework in UiPath

Robotic Enterprise Framework is a project template based on State Machines.

Re-Framework is a Template which provides all the required **basic needs of any process automation.**

Like reading and **storing the config data.**

Close all the unnecessary applications.

Open the required applications.

Get the Transaction and Process it.

Retrying the transaction if required and logging the status of all processed transactions, Failed or successful.

The Features of Re-Framework:

Proper Exception handling

Recovery abilities

Effective Logging

Reusability...etc..

The Re-Framework has a **main.xaml** file, which uses the state machine workflow. It is used for a better presentation of the process flow.

The **Main.xaml** file is designed with the state machine. And it is **prepared with the three state activities and 7 Transition**. Each state has been connected with the transaction.

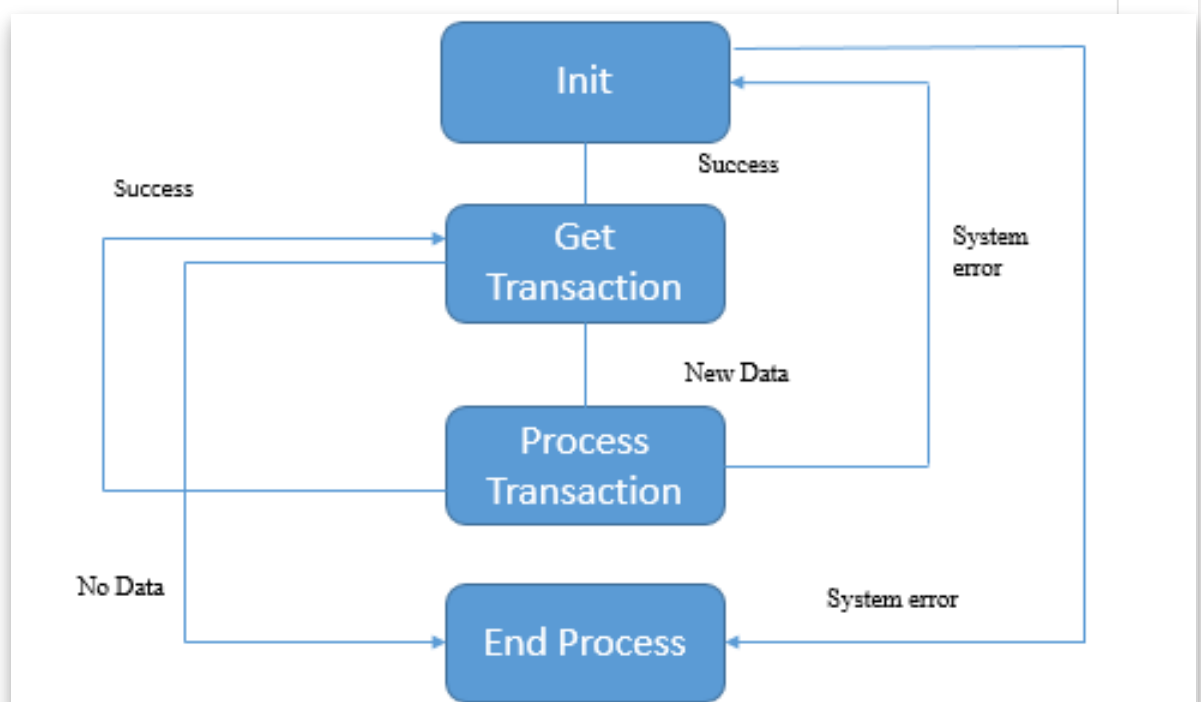
Re-framework architecture has four states :

Init : This is the Initial state, which is used to read and store the config data in a dictionary. Closes all the applications and Opens the required application. If Init passes successfully, then the robot moves to the next state.

Get Transaction Data: It is a data retrieval mechanism. Used to get the transaction from Queue, Data table, Database, Folder, etc..If it has a new transaction item, it goes to the next state.

Process Transaction : It is used to Process the transactions fetch from, get from the previous state. If there are no records to process, then the robot moves to the end process state.

End Process: It ends the robot successfully and then closes all the application and kill the apps.



RE-Framework is created to fit all of the best practices regarding logging, exception handling, application initialization, and others, being ready to tackle a complex business scenario.

The template contains several pre-made State containers for initializing applications, retrieving input data, processing it, and ending the transaction.

All of these States are connected through multiple Transitions which cover almost every need in a standard automation scenario.

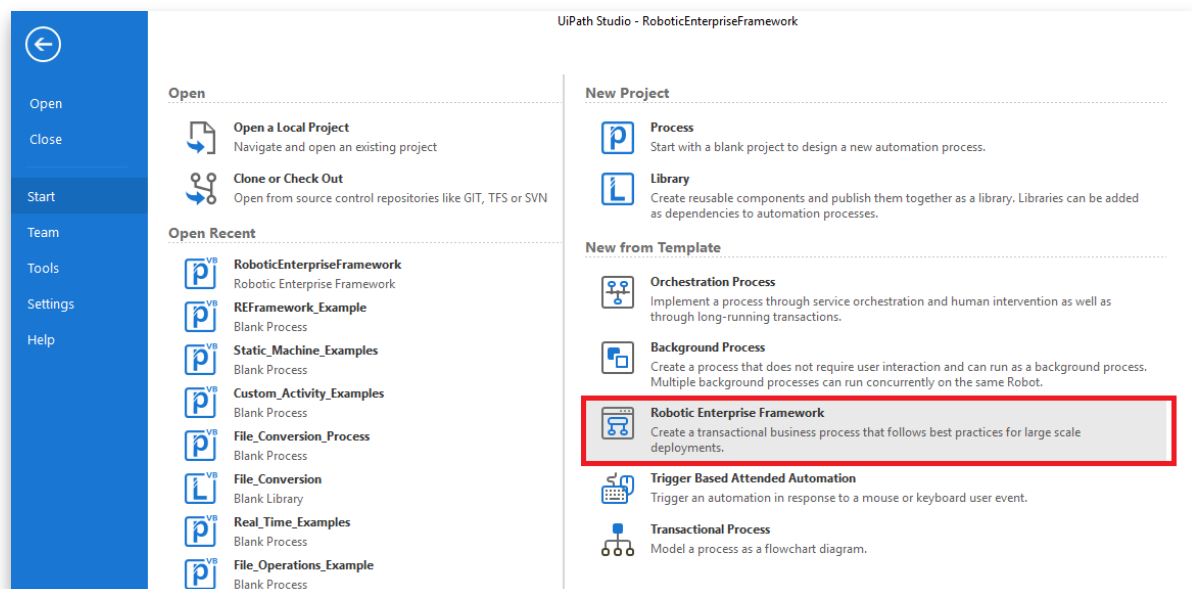
The default dependencies in a Robotic Enterprise Framework project are:

- UiPath.Credentials.Activities
- UiPath.Excel.Activities
- UiPath.Mail.Activities
- UiPath.System.Activities
- UiPath.UIAutomation.Activities.

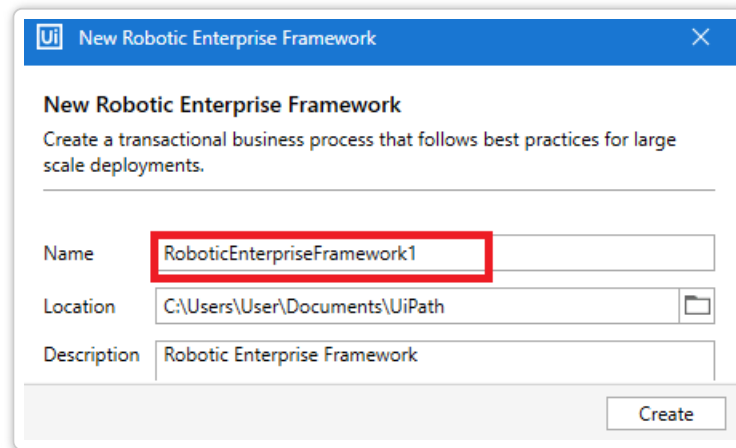
Real-time Example for the Re-Framework

This example demonstrates getting orders from Amazon. This example also explains how to implement Re-Framework for our business requirements.

Open the **UiPath Studio** home and then select **Robotic Enterprise Framework**



Create a New Robotic Enterprise Framework



New Robotic Enterprise Framework

Create a transactional business process that follows best practices for large scale deployments.

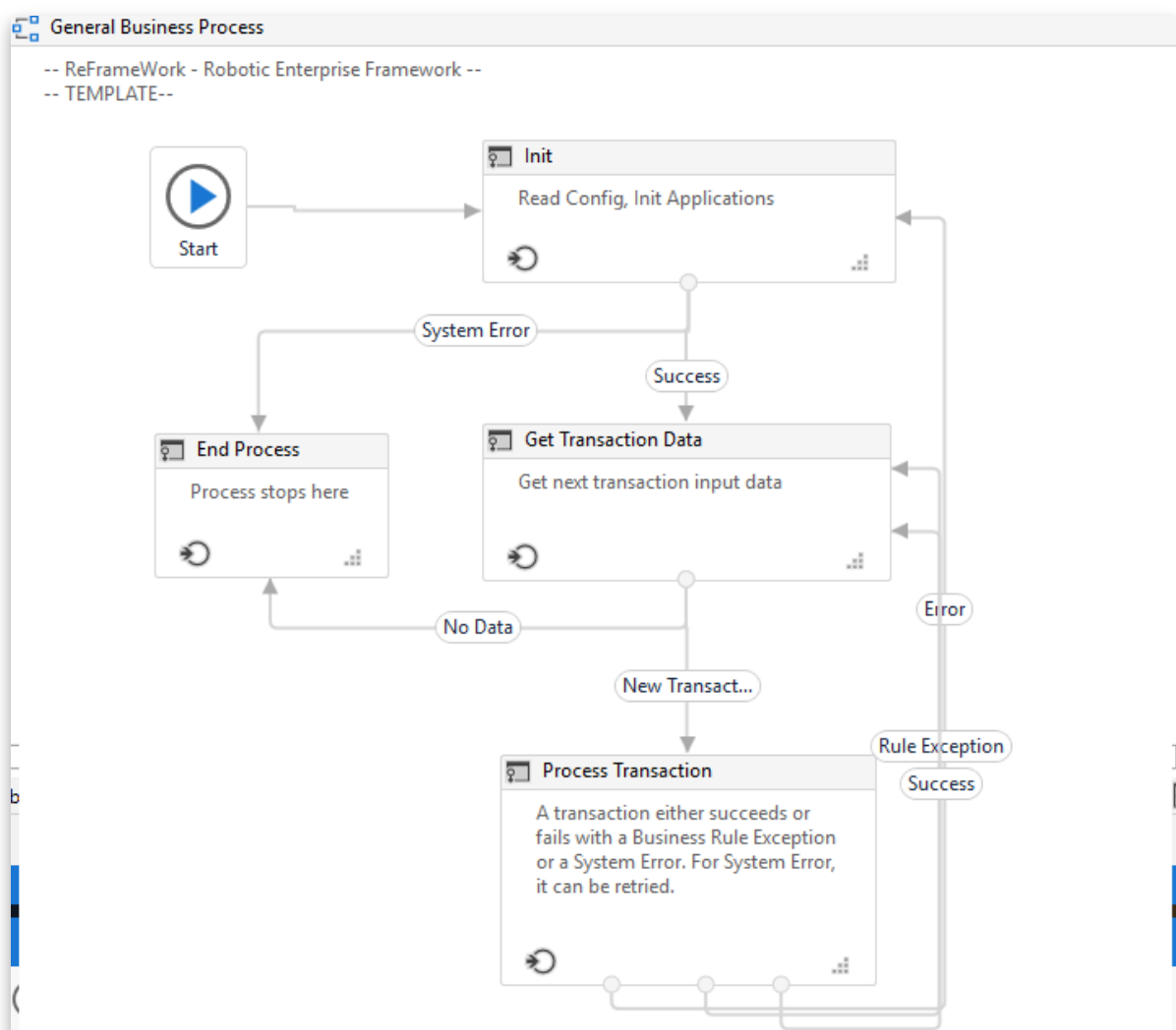
Name: **RoboticEnterpriseFramework1**

Location: C:\Users\User\Documents\UiPath

Description: Robotic Enterprise Framework

Create

Once it is opened in the UiPath Studio, it looks as shown below. The Re-framework is based on State Machine, and it contains four states as Init, Get Transaction Data, Process Data, and End Process.

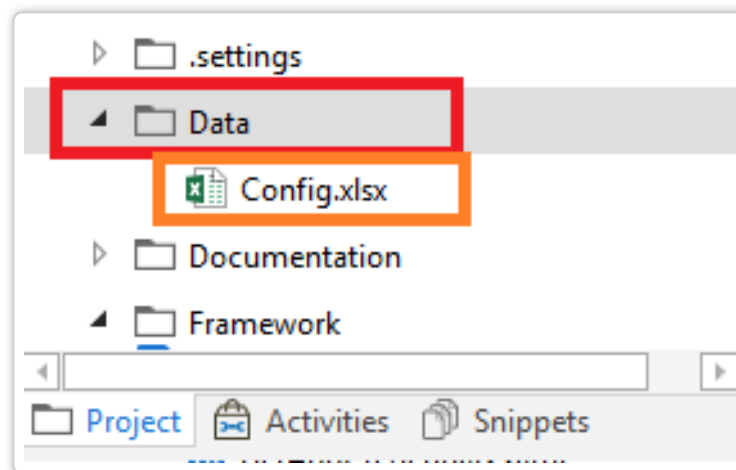


Next, click on any of the activities and then click on the **Variables** pane, you will

find some of the global variables which are created by the template itself.

Name	Variable type	Scope	Default
TransactionItem	QueueItem	General Business ...	Enter a VB expression
SystemError	Exception	General Business ...	Enter a VB expression
BusinessRuleException	BusinessRuleExcepti	General Business ...	Enter a VB expression
TransactionNumber	Int32	General Business ...	1
Config	Dictionary<String,O	General Business ...	Enter a VB expression
RetryNumber	Int32	General Business ...	0
TransactionField2	String	General Business ...	Enter a VB expression
TransactionField1	String	General Business ...	Enter a VB expression
TransactionID	String	General Business ...	Enter a VB expression
TransactionData	DataTable	General Business ...	Enter a VB expression

Where **Init state** which reads all the data from the **config file**. Go to the **Project Pane** and then click on the **Data** you will find the **Config file**.



Click on the **Config.xlsx** file, Once it is open, it looks as below

	A		C
1	Name	Value	Description
2	OrchestratorQueueName	KibanaDemoQueue	Orchestrator Queue Name. Be sure to match this name v
3			
4	logF_BusinessProcessName	Framework	This is a logging field which allows you to group the log
5			

Now I am going to add below details in the config file

	A	B	C
1	Name	Value	Description
2	OrchestratorQueueName	KibanaDemoQueue	Orchestrator Queue Name. Be sure to match this name v
3			
4	logF_BusinessProcessName	Framework	This is a logging field which allows you to group the log
5	loginurl	https://www.amazon.com	
6	credentials	amazonlogin	
7	output_file	output.xlsx	
8	year		2016
9			
10			

Where the **loginurl** is the basic URL instead of the variable, we have created

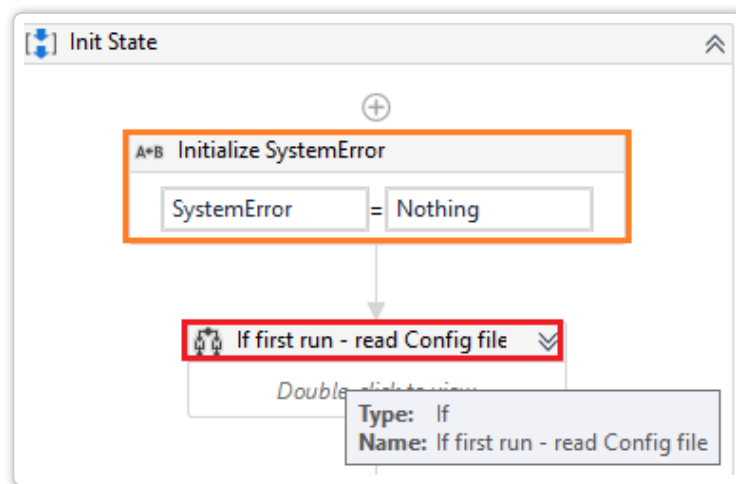
that in our config file, a this will be read from here. And **amazonlogin** is the name, however, and if you are connected with assets in orchestrator, then it looks for a name called **amazonlogin** ; if it is not connected, then it prompts for Userid and logs in.

Output_file is an output file after scraping the data it will take the data to the **output.xlsx** and next is the year it will ask the user to which year you want to scrape the data.

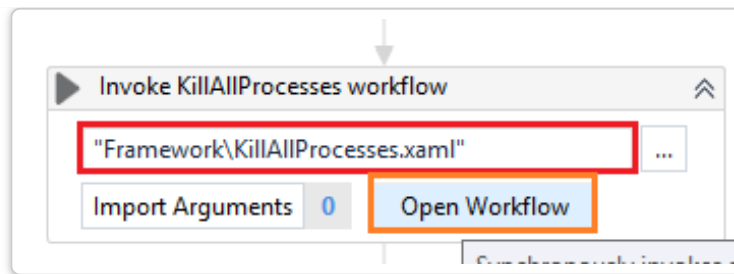
Like this, you can create your variables and values directly in the **config.xlsx** which can be used throughout your re-framework.

Init State:

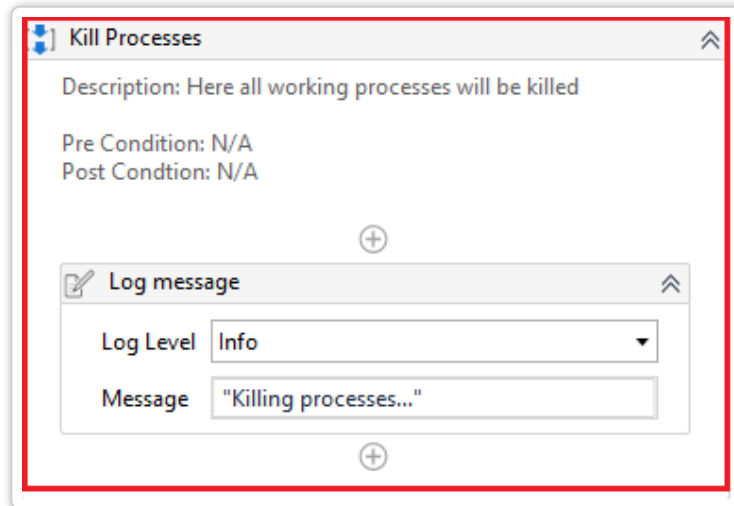
Click on the **Init State** and then Once the **Init state** as opened in the **UiPath Studio** then double click on the **if first run-read config file**.



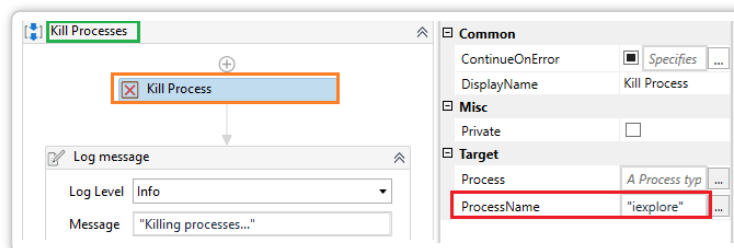
Once it is opened, scroll down to **Invoke KillAllProcesses.xml** and then click on the **Open Workflow**



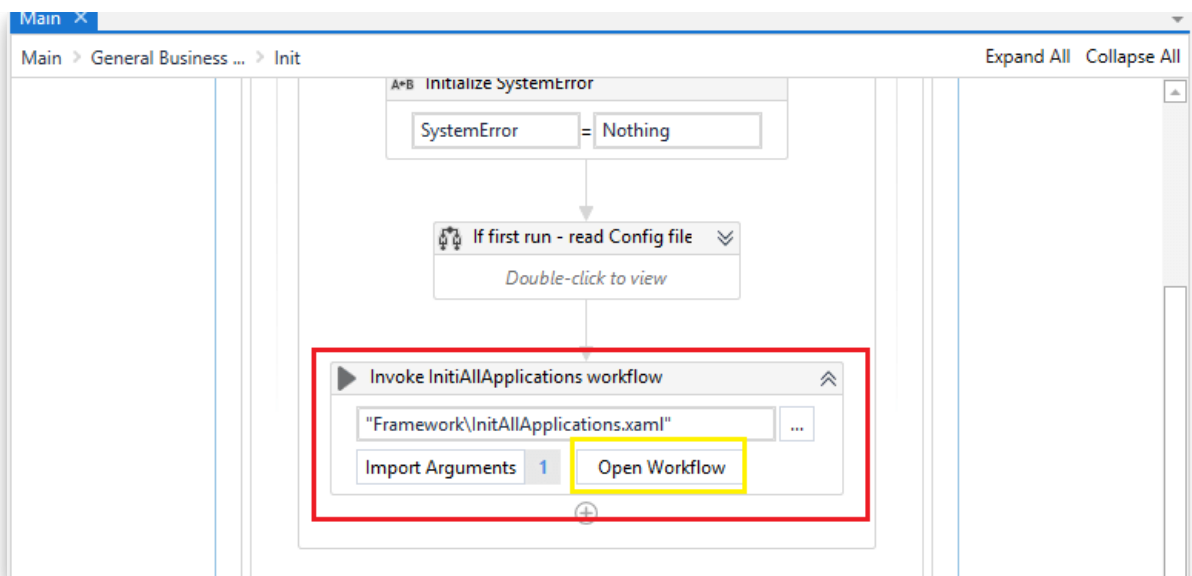
Next, the **Kill Process** activity will be opened as shown below



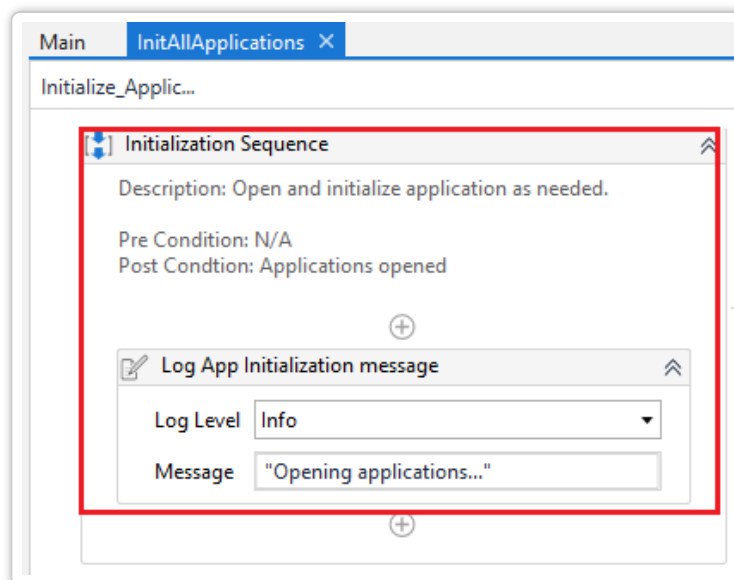
Add **Kill Process** activity inside the **Kill Process** and click on the **Kill Process** activity and add the process name as **iexplorer** in the **Properties Pane**, as shown below.



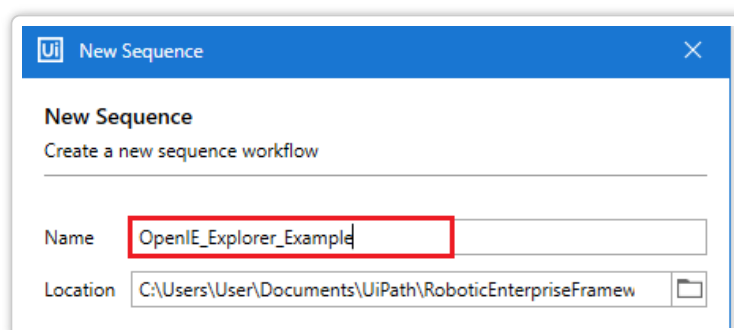
Close the **Kill Process** and then move to the **Invoke InitAllApplications workflow**



Click on the **Open Workflow** to Open the **Invoke InitAllApplication workflow**, Once it is opened, it looks as shown below.

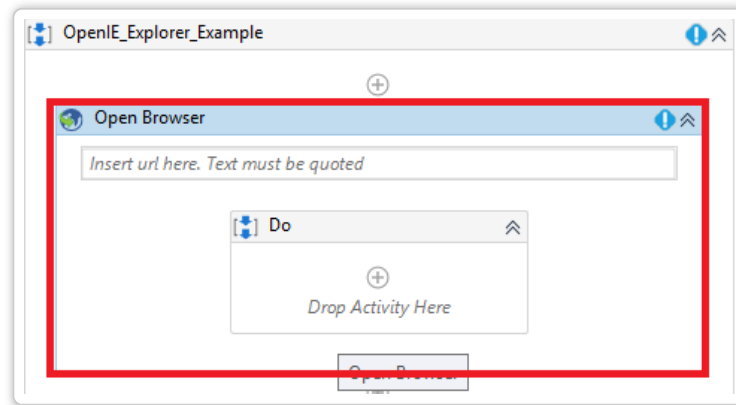


Create a **new sequence** to work on the browser, I am creating a new sequence called **OpenIE_explorer_Example**

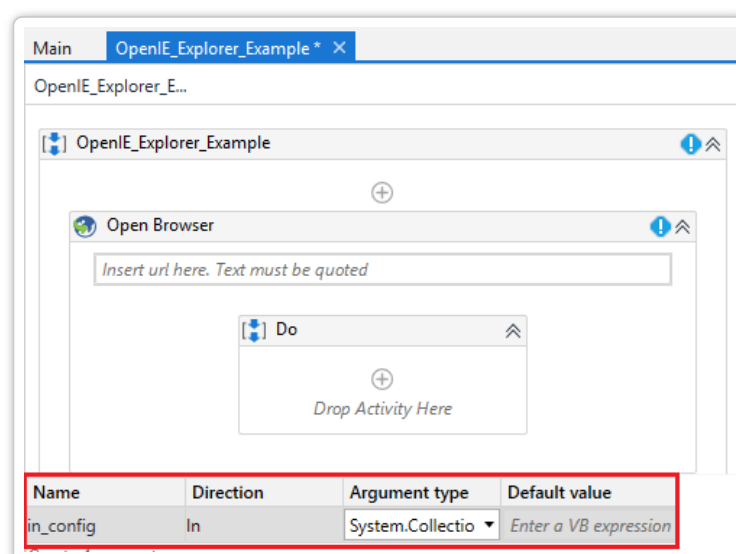
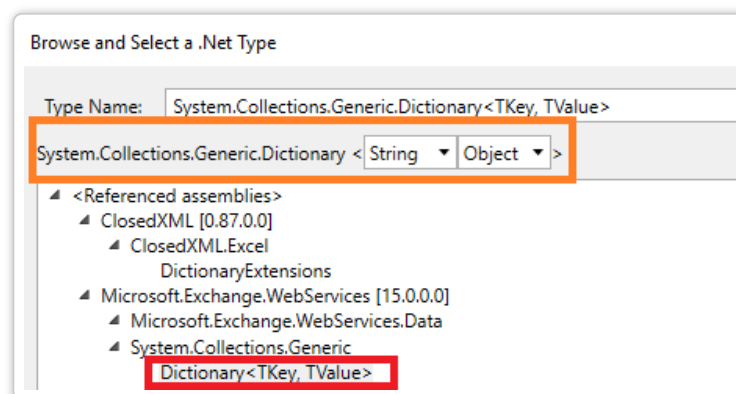


Once the sequence opened in the **UiPath studio**, then add **Open Browser**

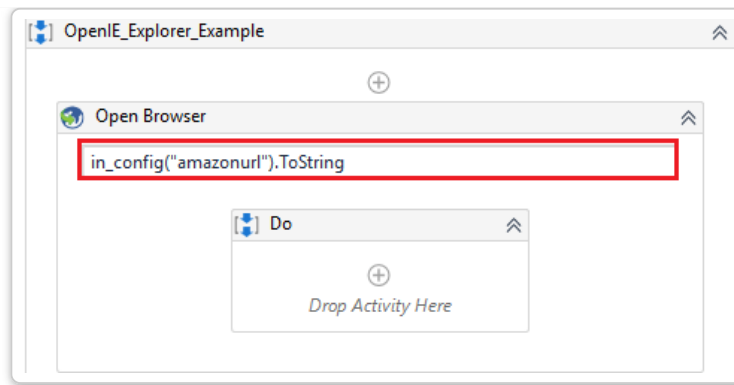
activity inside the sequence.



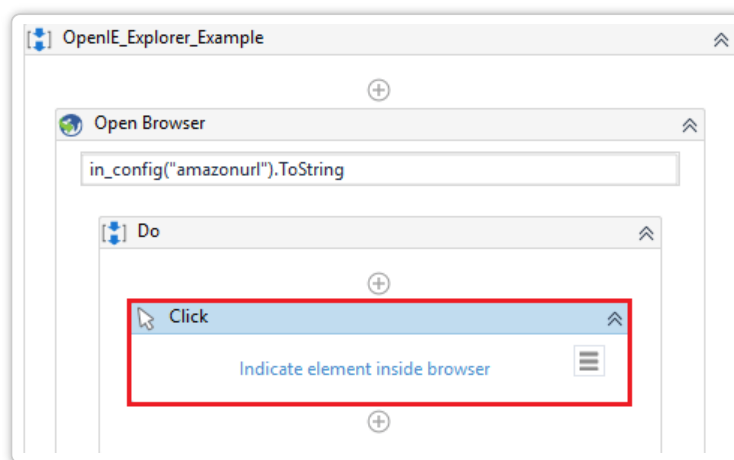
Click on the **Open browser** and then create a **new argument** in the **Argument pane**. And then enter the argument type as **System.Collections.Generic.Dictionary<TKey, TValue>** and then select the **Tkey** type as **String** and **TValue** type as **Object**



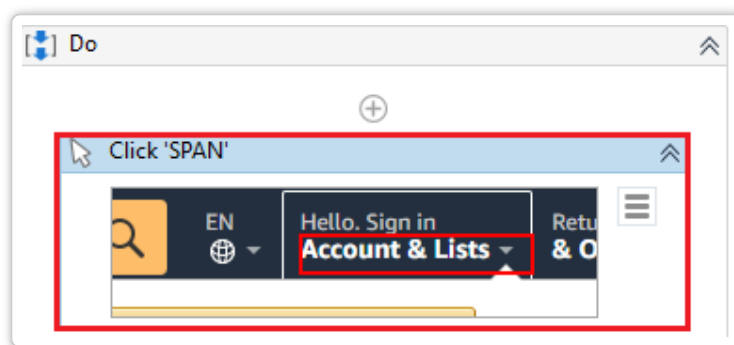
Next, enter the **URL** which we are going to get the **amazonurl** from excel via **in_config** file written as **in_config("amazonurl").ToString**



Next, Open the **Amazon login page** in the browser and then go to **UiPath Studio** and add **Click activity** inside the **Do Container** of the **Open browser**.



Next, Click on the **Indicate element inside browser** and click on the **Sign-in** button in the amazon login page in the browser.



Next, add one more **Type Into activity** inside the **Do Container** and then scrape the **Email or phone number page** as shown below. Click on the **Do Container** and create a new variable as **userid** and **password** in the **Variables pane**

Name	Variable type	Scope	Default
userid	String	Do	Enter a VB expression
password	String	Do	Enter a VB expression

Create Variable

Type Into 'INPUT ap_email'

Email or mobile phone number

userid

After entering the **email or phone number** you need to click on the **Continue button**, for that add **Click activity**, and then indicate the **Continue button** on the browser.

Click 'INPUT continue'

Contin

Next, add One more **Type Secure Text activity** and then scrape the **Password** in the browser and then enter the **password** variable under **Properties Pane** for **Secure Text Property** and also set the type of the **password** variable to **secure.security.securestring**

Browse and Select a .Net Type

Type Name: System.Security.SecureString

- <Referenced assemblies>
 - ActiproSoftware.Shared.Wpf [18.1.676.0]
 - ActiproSoftware.Windows
 - mscorlib [4.0.0.0]
 - System.Security
 - SecureString

Type Secure Text 'INPUT ap_password'

Password

Input

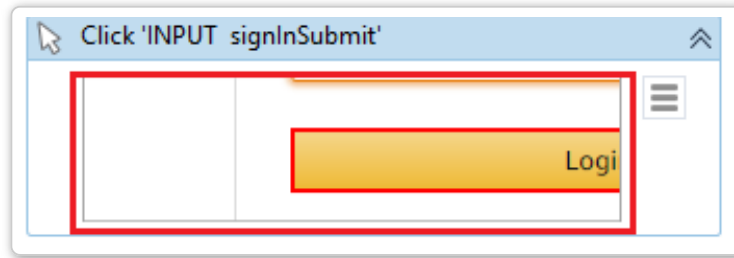
SecureText password

Target Target

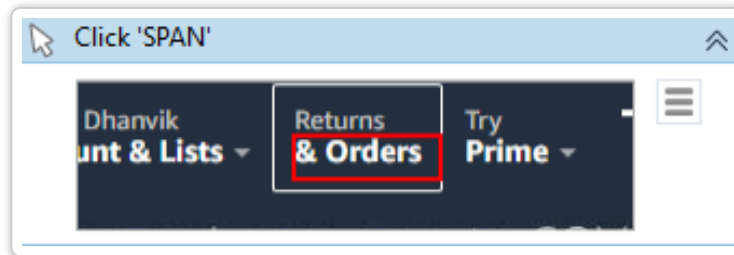
Misc

Private

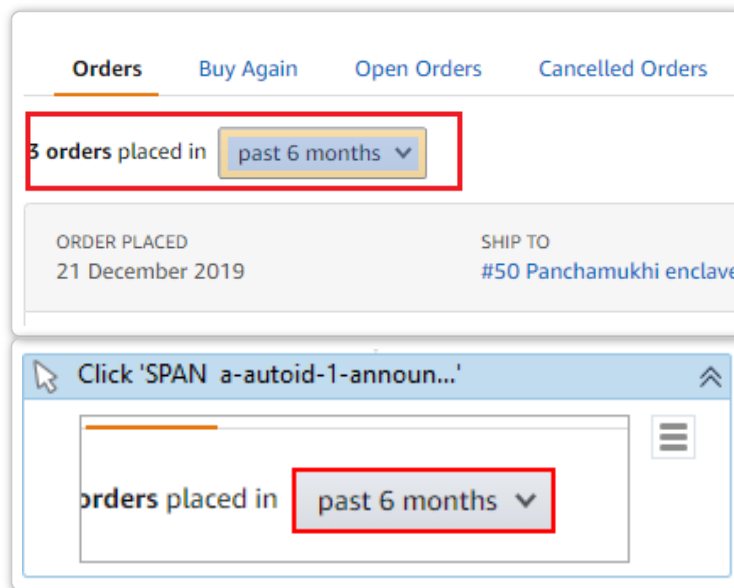
Next, add **Click activity** and then scrape the **Login button** on the browser



Once we login to the Amazon we need to order the items, so Add **Click activity** and then scrape the Order section as shown below.



Add **Click activity** and then scrape **past 6 months button** under **Orders**



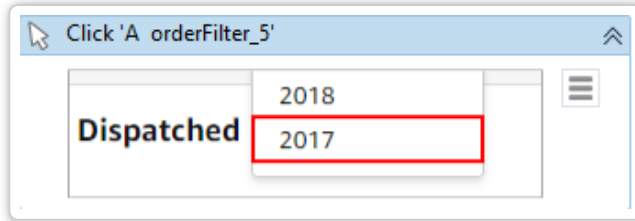
Now add **Click activity** into the **Do Container** and then,

Click on the **Indicate element inside browser** and then place the cursor on the **past 6 months**

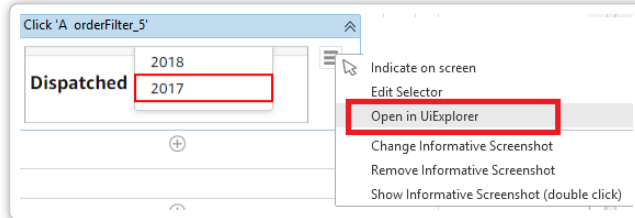
Press **Ctrl+F2** to select the year from the drop-down

Select the year from the drop-down and wait until the time zone ends

Then select the 2017 year and scrape it.

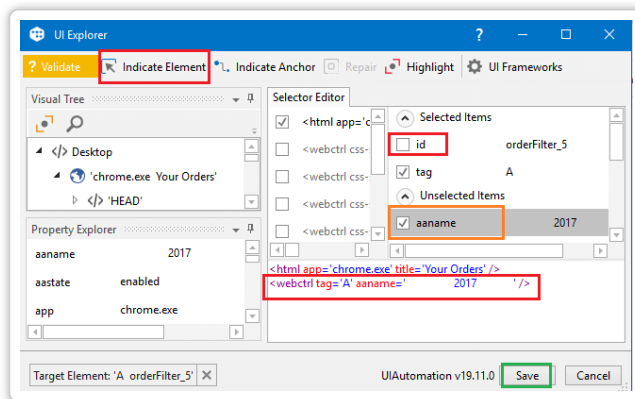


Click on the three horizontal lines(Hamburger) and then select **Open in UiExplorer**



Once it is opened in the **Ui explorer**, then click on the **Indicate element** and then select **2017** in the browser.

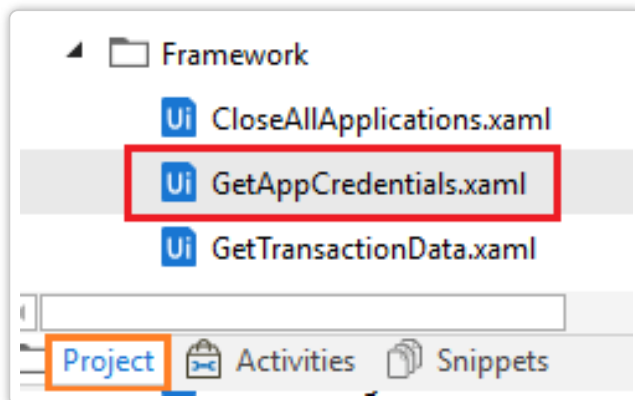
Next, unselect **id** and then select **aname** and then copy the below code and click on **save**.



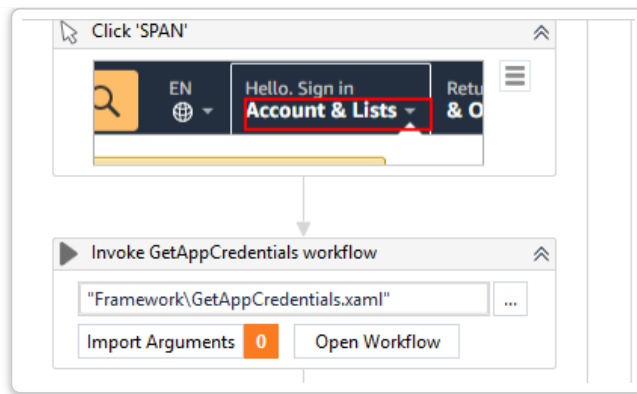
Click on the **Click Activity** which has scraped 2017 year and then go to the **Selectors** in the **Properties** pane, to manipulate the **aname** value as shown below.

```
<html app='chrome.exe' title='Your Orders' />
<webctrl tag='A' aname="+in_config("year").ToString+"/>
```

Go to the **Project** pane and then select **GetAppCredentials** under **Framework**



Next, add **Invoke GetAppCredentials workflow activity** inside the **Do container** and before the email scraping **click activity**.



So we are going to get the userid and password from the Orchestrator or from the User during the running the program.

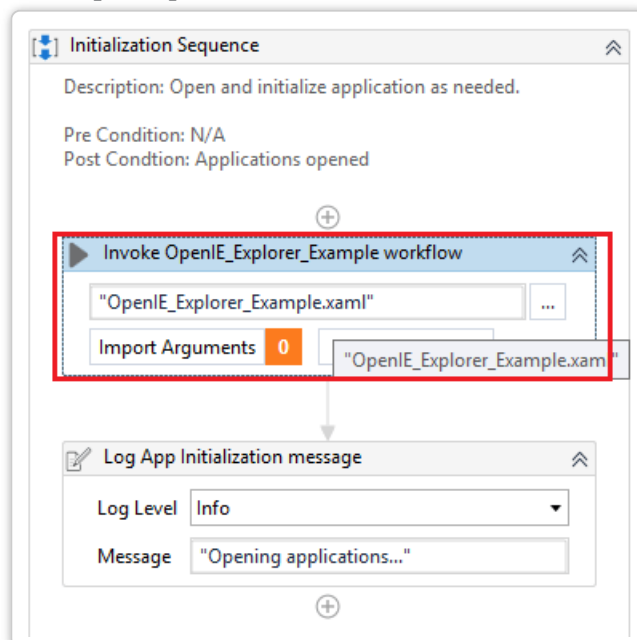
If Orchestrator is connected then we are going get the userid and password from the asset credentials with the name which we have defined in our excel.

In the Excel file, we defined as **Amazon credentials**

Now we have to pass the arguments, so click on the **Import Arguments** and then

Name	Direction	Type	Value
out_Username	Out	String	userid
in_Credential	In	String	"amazoncredentials"
out_Password	Out	SecureString	password

Now go to your **InitAllApplications workflow** and drag and drop the **OpenIE_Explorer_Example** sequence.



And then click on the **Import Arguments** and then pass the argument **In_config** which passes all the configuration related data to our **OpenIE_Explorer_Example**

sequence as shown below.

Name	Direction	Type	Value
in_config	In	Dictionary<String, Object>	in_Config

Create Argument

Now, we have done all the initializations.

*Now, let us go to the **Invoke KillAllProcess** workflow and the **open the workflow** and then add **Path Exist** activity inside the **Kill Process**.*

*And add **in_config** the argument in the argument pane and then enter the path to the file as shown below.*

Name	Direction	Argument type	Default value
in_config	In	Dictionary<String, Object>	Enter a VB expression

Create Argument

Path Exists

Path Type: File

Path: in_config("output_file").ToString

*Click on the **Path Exists** and then create a new variable called **existfile** in the **Properties Pane** for **Exists Property**.*

Properties

UiPath.Core.Activities.PathExists

Common

DisplayName: Path Exists

Input

Path: in_config("ou

PathType: File

Misc

Private: ☐

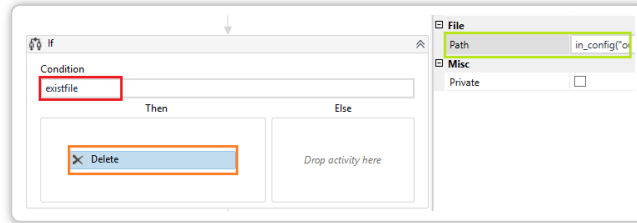
Output

Exists: existfile

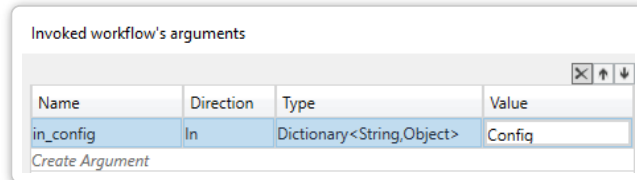
*Next, add **If Condition** after the **Path Exist** activity in the **Kill Process** and write the condition as shown below.*

*Add **existfile** variable inside the **if Condition** and then enter the path of the variable(`in_config("output_file").ToString`) in the properties pane for **path property***

And also add **Delete** activity inside the **Then** condition block



Next, go back to the **Invoke KillAllProcess** and then click on the **Import argument** and then pass the value



So We have completed our initialization part.

Transaction State:

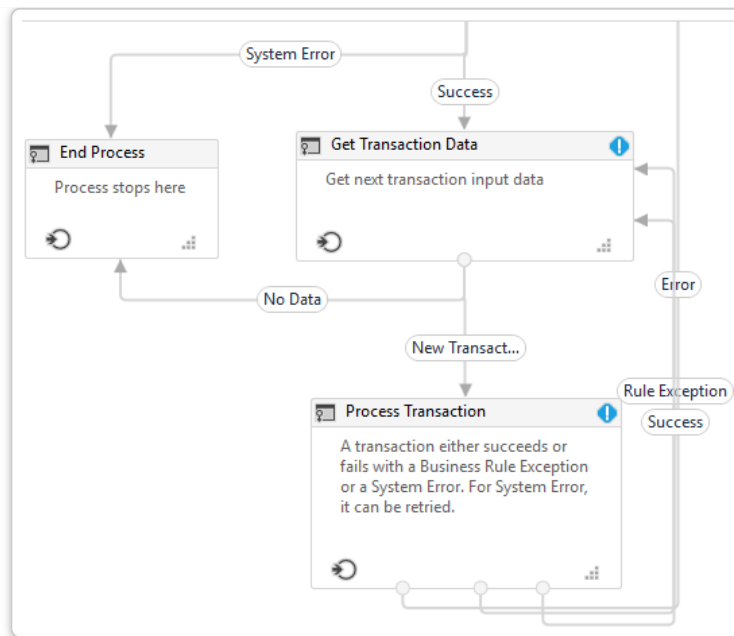
Now, let us go to the Get Transaction state. If you see the Transaction variable in the variable, the variable type is of QueueItem.

Name	Variable type	Scope	Default
TransactionItem	QueueItem	General Business ...	Enter a VB expressio
SystemError	Exception	UiPath.Core.QueueItem	Enter a VB expressio
BusinessRuleException	BusinessRule		Enter a VB expressio

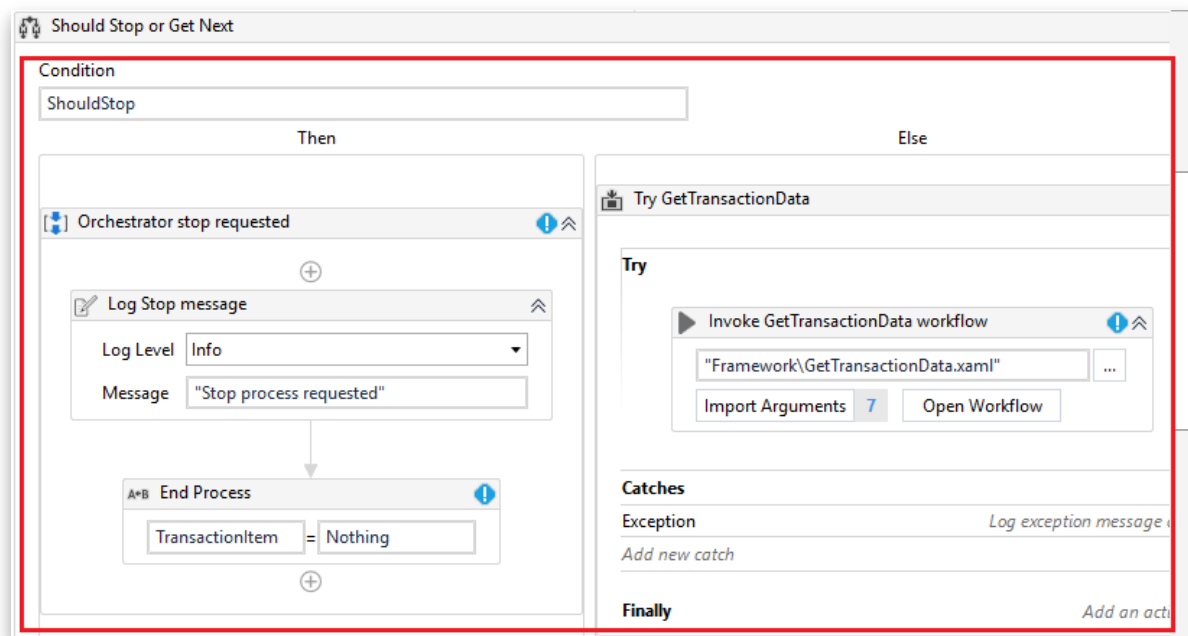
So here we are not using **Orchestrator** so we can change the variable type to **String**.

Name	Variable type	Scope	Default
TransactionItem	String	General Business ...	Enter a VB exp
SystemError	Exception	General Business ...	Enter a VB exp
BusinessRuleException	BusinessRuleExcepti	General Business ...	Enter a VB exp

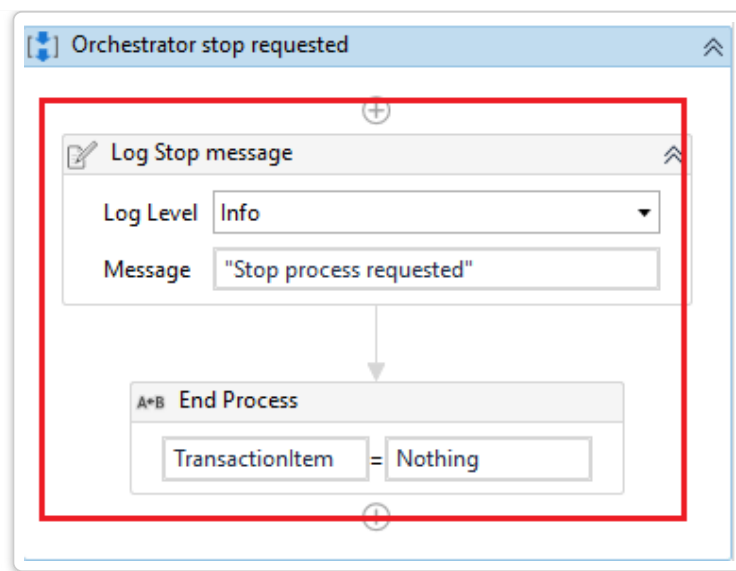
As soon as you set the **Transaction variable type to String** you will find many errors across the workflow because it has been passed as **QueueItem** in many locations.



Double click on the **Get Transaction Data**



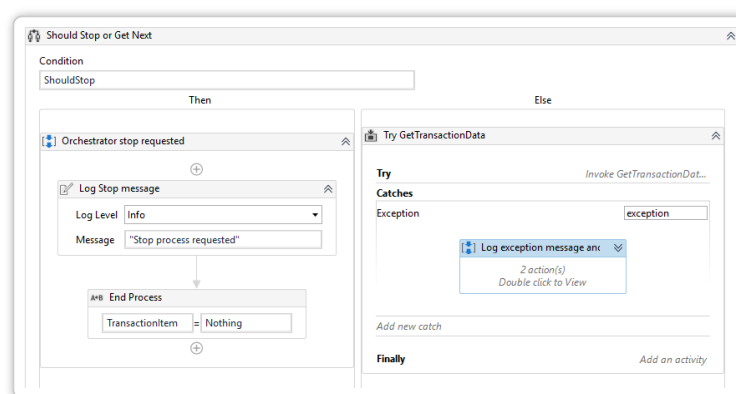
We are not going to assign anything to the string, so these are temporary errors,
Just click on the **TransactionItem** on the **End process** and cut it then paste it in
the same place so the error will go.



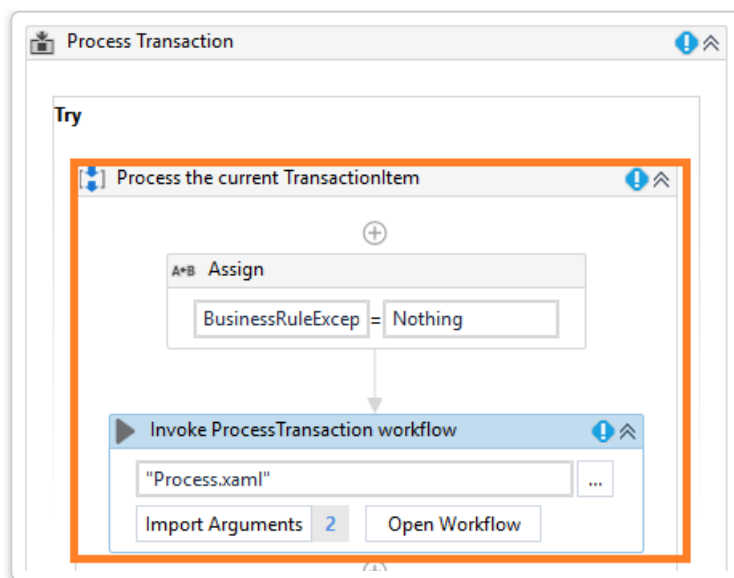
Next, click on the **Invoke GetTransactionData Workflow** and then click on the **Open Workflow** and then open the **Variable pane**, and wherever you will find the **Transaction variable** set the variable type to string.

When you click on the Import Argument on the **Invoke GetTransactionData Workflow** you need to freshly enter the argument because we have changed the variable type to string.

And even you need to traverse through **Try Catch exception** and then just cut and paste the Transaction variable after you do not see any errors.



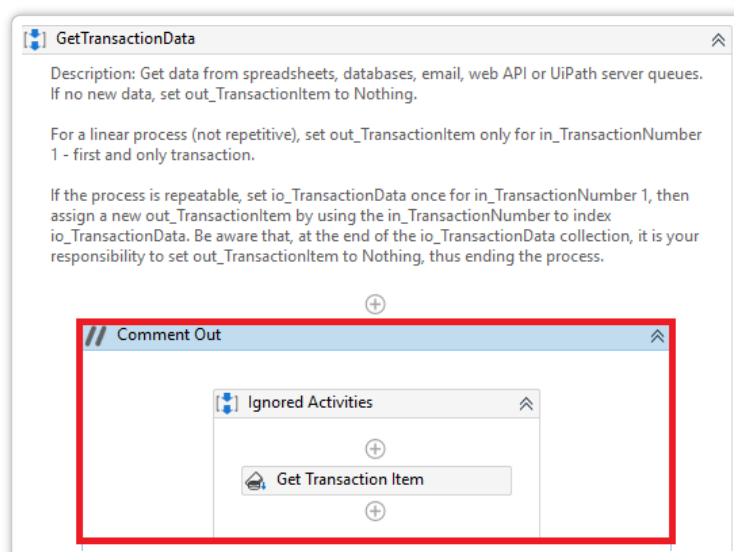
And we still have an error in the **Process Transaction**, and then click on the **Open Workflow** under **Invoke Process Transaction workflow**.



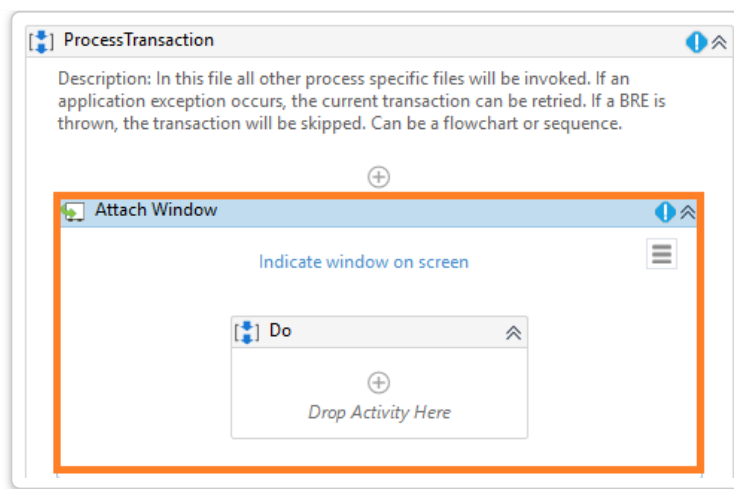
Once you open the workflow, you will find **Process Transaction** activity, click on the **Process Transaction** and then open the arguments in the **Arguments Pane** and then set the **In_Transaction** argument type to **String**.

In the same way, traverse through Try Catch and Finally block and set the Transaction variable type to string.

Now we have removed all the errors which occurred in the Workflow. Now go back to the **Invoke GetTransactionData Workflow** and then comment out the **Get TransactionItem** activity as shown below.



Next, Go back to the **Process Transaction** state. Double click to open it. Add **Attach Window** activity inside the **Process Transaction**



Next, click on the **Indicate window on-screen** and then indicate the order history page on the browser.

State Machine in UiPath

A state machine is a concept used in designing computer programs or digital logic.

There are two types of state machines: Finite and infinite-state machines

The former is comprised of a finite number of states, transitions, and actions that can be modeled with flow graphs, where the path of logic can be detected when conditions are met.

A state machine is any device storing the status of something at a given time.

The status changes based on inputs, providing the resulting output for the implemented changes.

A finite state machine **has limited internal memory**.

Input symbols are read in a sequence producing an output feature in the form of a user interface.

State machines are represented using state diagrams.

The output of a state machine is a function of the input and the current state.

State machines play a significant role in areas such as electrical engineering, linguistics, computer science, philosophy, biology, mathematics, and logic.

They are **best used in the modeling of application behavior, software engineering, design of digital hardware systems, network protocols, compilers, and the study of computation and languages.**

The **operation of a state machine begins from a start state.**

On a successful transition, it ends up in an accept state.

The **transition takes place based on the inputs provided.**

The **current state depends on the past state of the system.**

The number of states formed depends on the available states of memory.

A transition is enabled based on certain conditions and indicates a state change.

An action describes an activity performed at the given moment.

The different types of actions are transition action, input action, entry action, and exit action.

Deterministic automata have exactly one transition in every state for each possible input.

In non-deterministic automata, a state input leads to one, many, or no transitions.

A state machine with only one state is called a combinatorial state machine and uses only input actions.

The **two different groups of state machines are acceptors and transducers.**

Acceptors produce a binary output based on whether the input is accepted or rejected by the machine.

While processing the input, if the current state is accepting, the input is accepted. Otherwise, it is rejected.

Languages accepted by state machines are called regular languages.

Start states are represented by an arrow pointing on it from anywhere, while accepted states are represented using double circles.

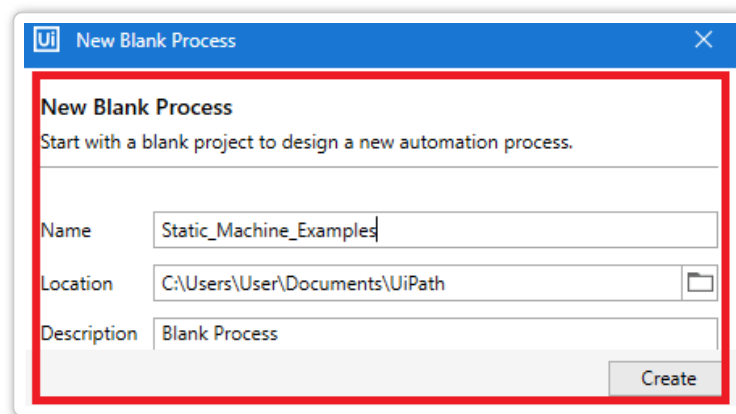
Transducers cater output based on a given input, using actions. Moore and Mealy machines are examples of sensors.

Unmodified modeling language state machines are also widely used as they have both the Moore and Mealy machine characteristics within them.

They include additional concepts such as orthogonal regions and hierarchically-nested states.

Real-time example of State Machine in UiPath

Create a new process called **Static_Machine_Examples**



In this example, we are considering Lift as a real-time example for Static Machine.

If a lift is on the 1st floor, then it will go to either the ground floor or the 2nd floor.

If Lift is on the ground floor, then it will go to the 1st floor.

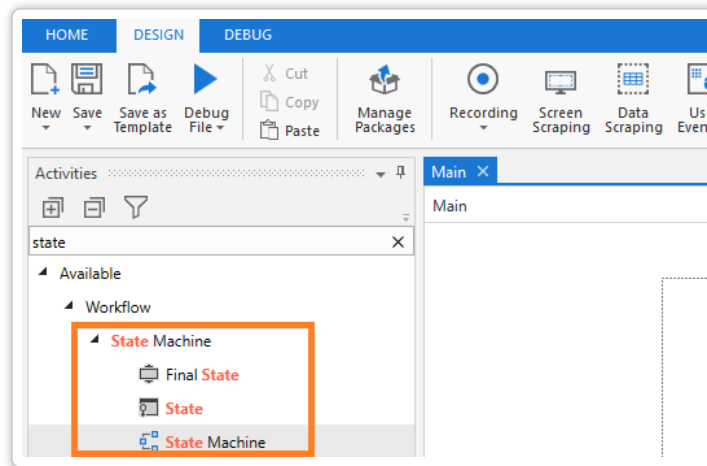
Let us take it as the Initial state is the **0th floor**, which is nothing but an Initialization.

The second state is getting people inside the Lift, which is nothing but a Get Transaction.

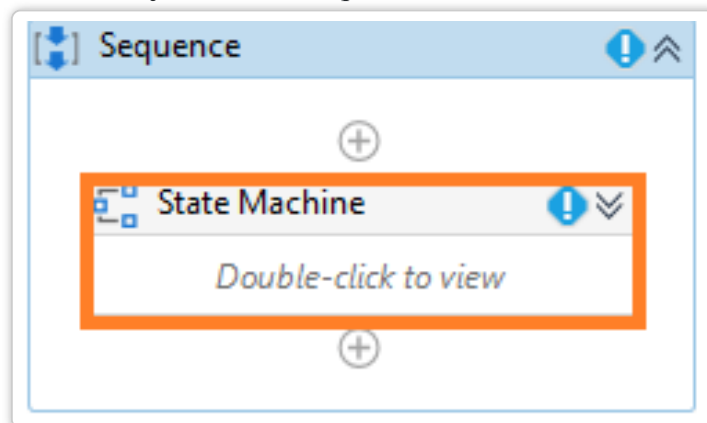
Once the people are in, then the Lift moves from one floor to another floor, which is nothing but a Processing state.

Once the Lift moves to the 11th floor, then it is the final floor and which is nothing but a Final state.

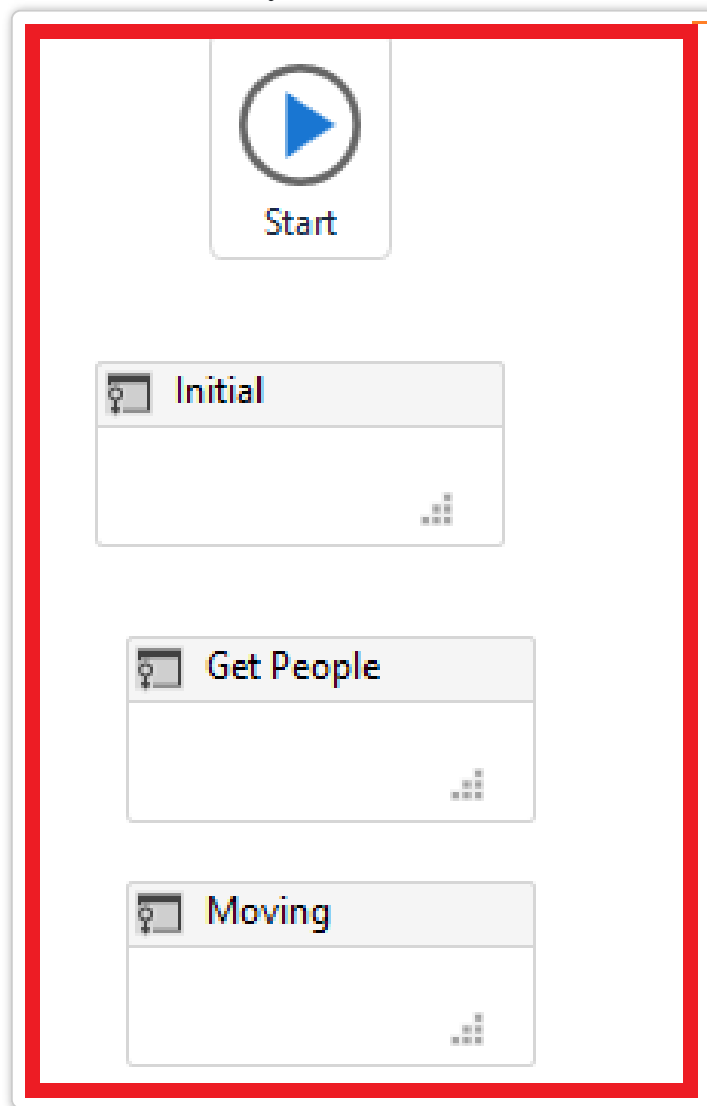
Now, go to **UiPath Studio**, search for **State Machine** in the **activity panel**.



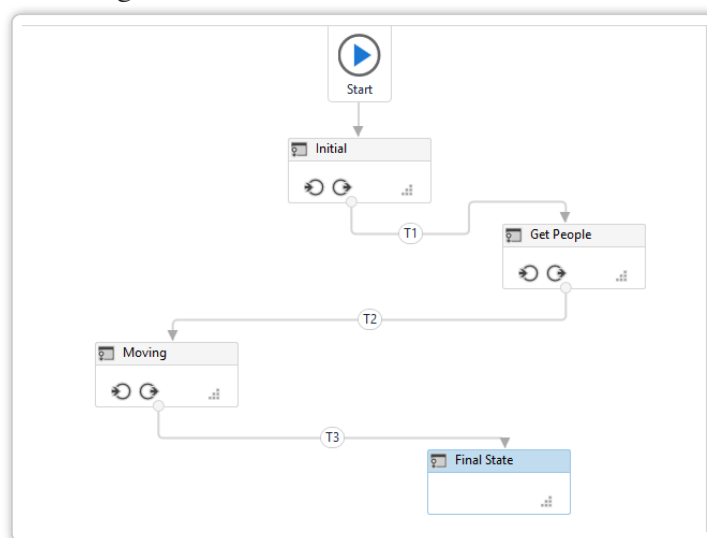
Next, add **State Machine** activity inside the sequence.



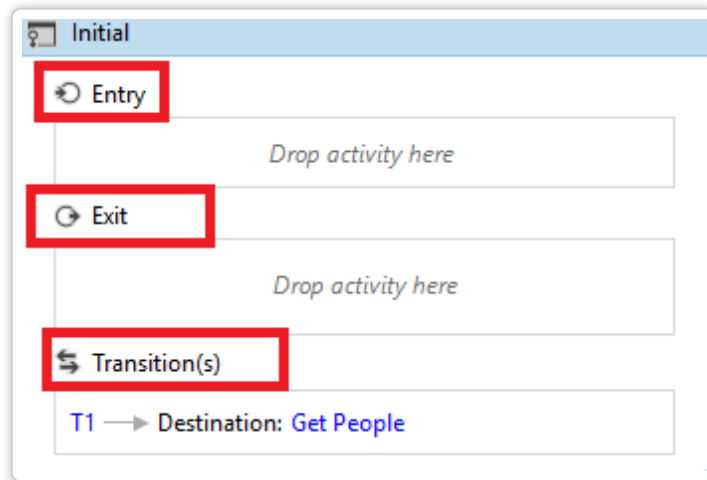
By double-clicking on the **State Machin**, you will find the **Start button**.



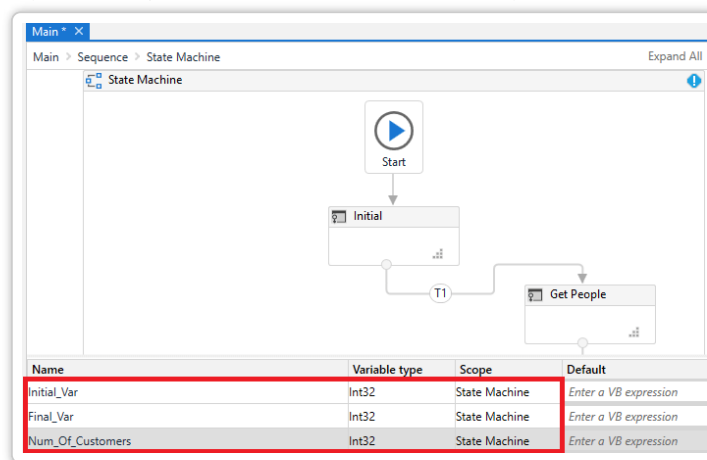
Next, add **Three-State activities** inside the **State Machine activities**, as shown below. If you need, you can also change the name of the state, as shown below.



Next, add **Final State Activity** inside the **State Machine activity**.



Connect each of them by drawing a line between each activity.

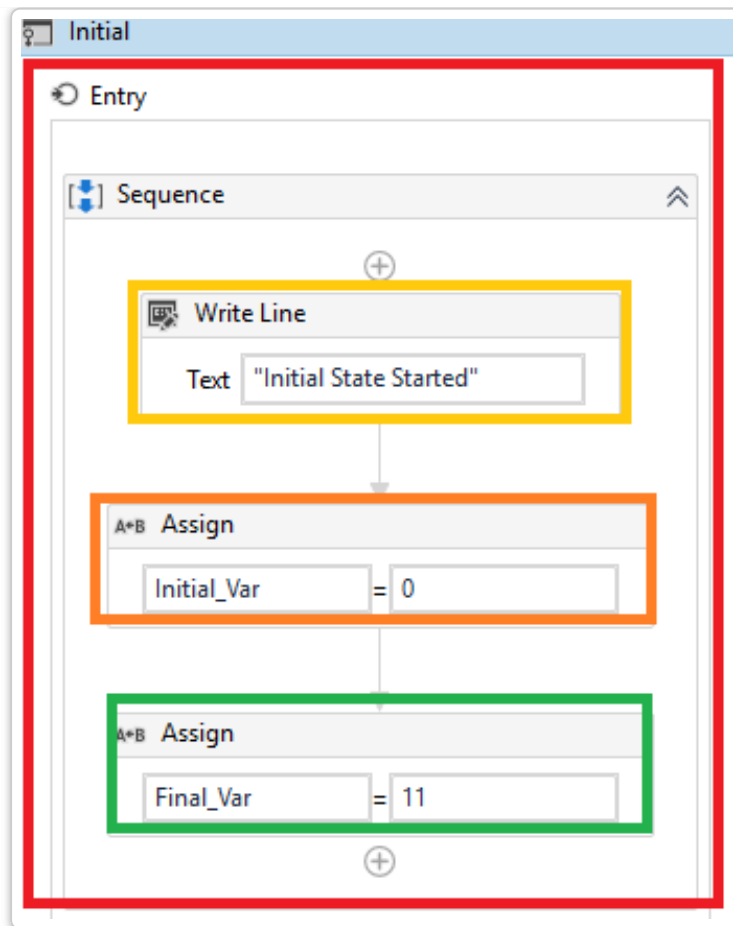


In the below image, you can see the **Transition numbers T1, T2, and T3**. Transitions happen when we are moving from one state to another state. In Transitions, there are conditions.

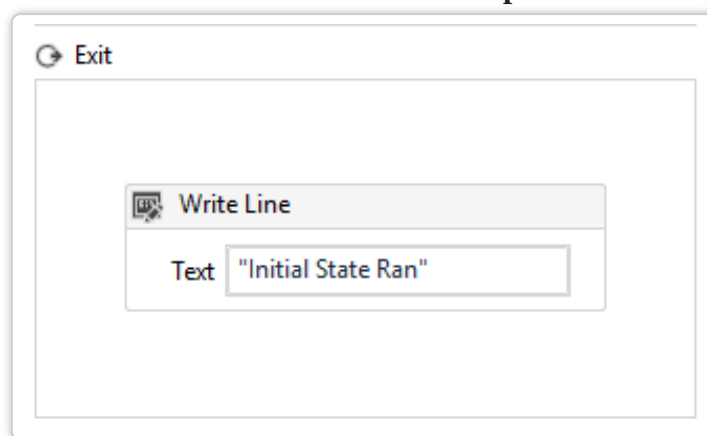
If we want to move from getting people to moving state, then this transition happens only when the people less than 10.

If there are more than ten people, then the transition won't happen.

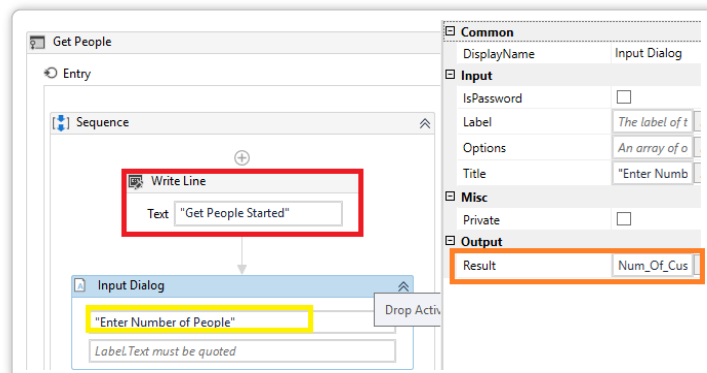
That means moving from one state to another state is based on the transition condition.



If you click on any transition, you can see the condition, **where the transition is happening from source Initial to the Destination Get People.**

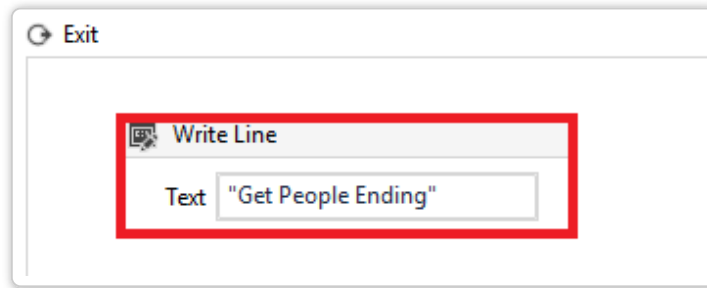


Now let us go to each state and have a closer look, you can see that each state is having Entry, Exit, and Transition state.



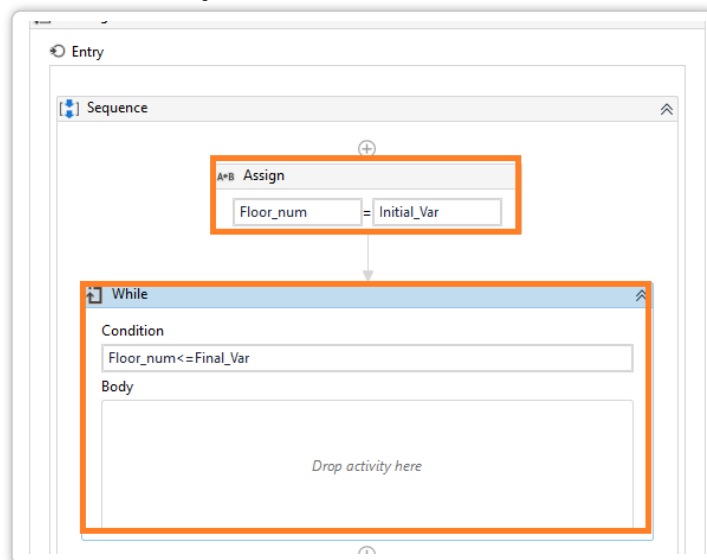
When you run the transition, the Entry state of each activity will run first, and then the Exit state of each state will be run next.

Next, click on the **State Machine activity** and then create a variable shown below.

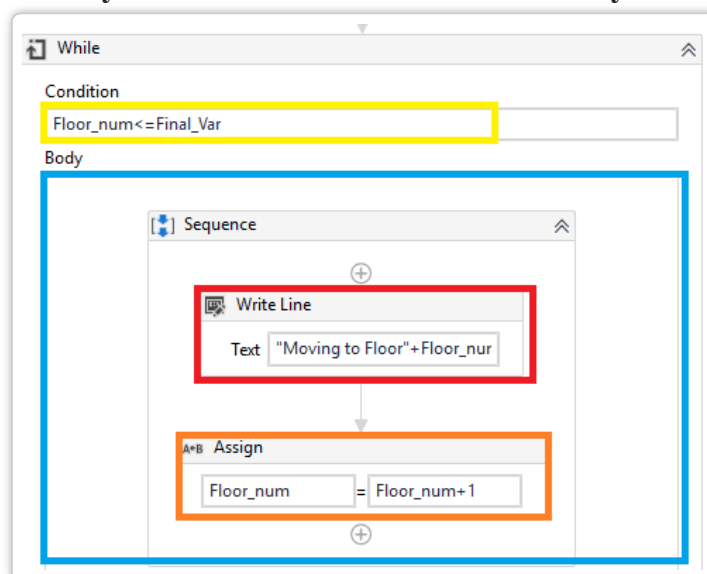


Click on the **Initial State activity** and add **Write Line activity** inside the Entry State in the Initial activity.

Add **Assign activity** inside the Entry State and enter the condition, as shown below.



Next, add **Write Line activity** inside the **Exit state of Initial activity**.

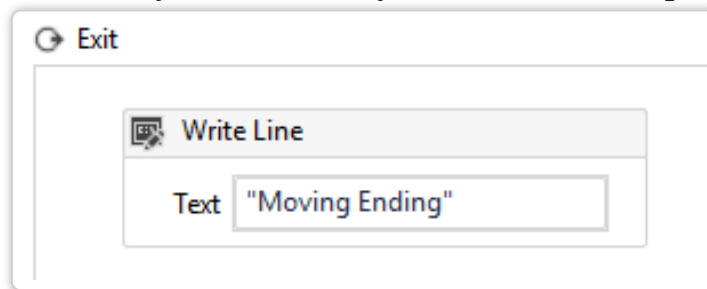


Next, click on the **Get People activity** and add **Input Dialog activity** inside the **Entry**

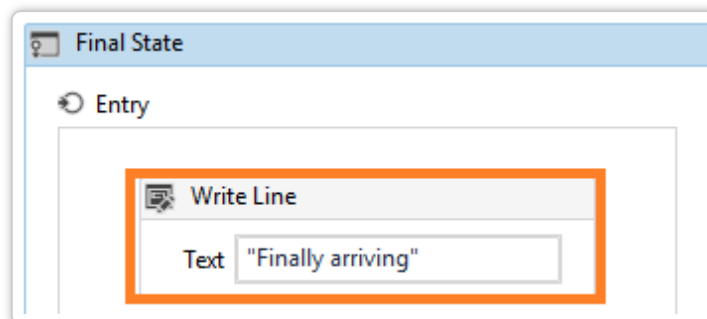
state of the **Get People activity** and write the condition as shown below.

Click on the **Get People activity** and enter the variable name **Num_Of_Customers** in the **Result Property** under **Properties Pane**.

And also, add **Write Line activity** inside the **Entry state** of the **Get People activity**.

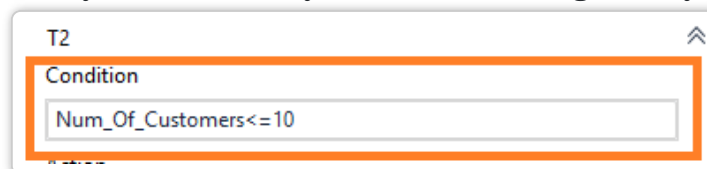


Add **Write Line activity** inside the **Exit state** of the **Get People activity**, as shown below.

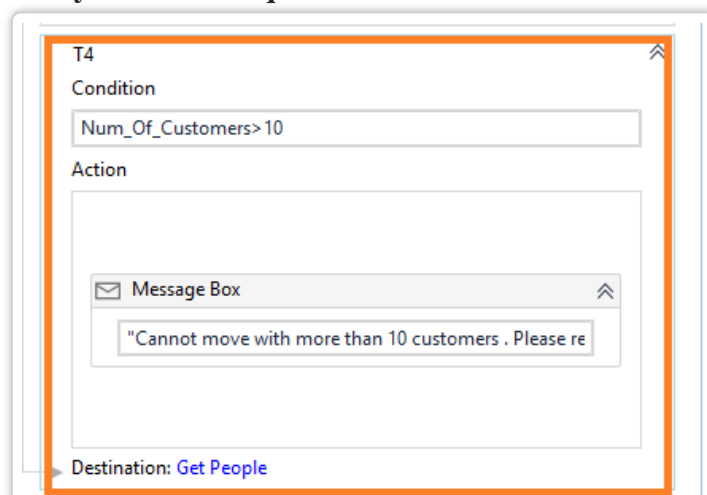


Next, click on the **Moving activity** and add **Assign activity** inside the **Entry State** and then create a new variable inside the Assign activity.

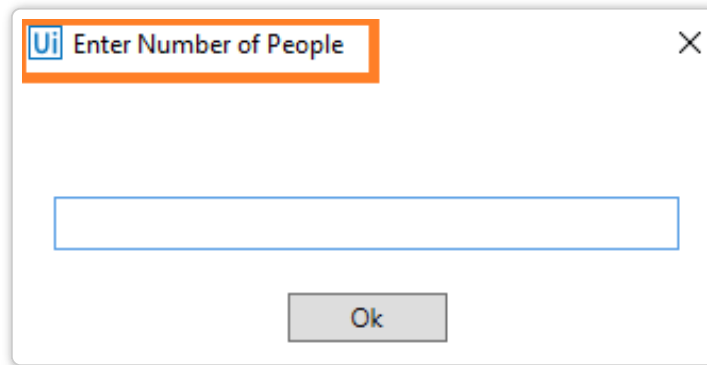
And then add **While activity** inside the **Entry state** of the **Moving activity**.



Next, add a **sequence** inside the **body of the while activity** and then add **Write Line activity** and **Assign activity** inside the **sequence** and write the conditions shown below.

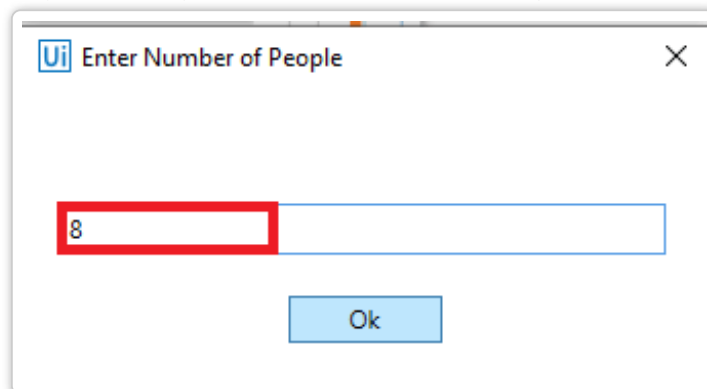


Click on the **Exit state** and add **Write Line activity** and enter the text as follow.



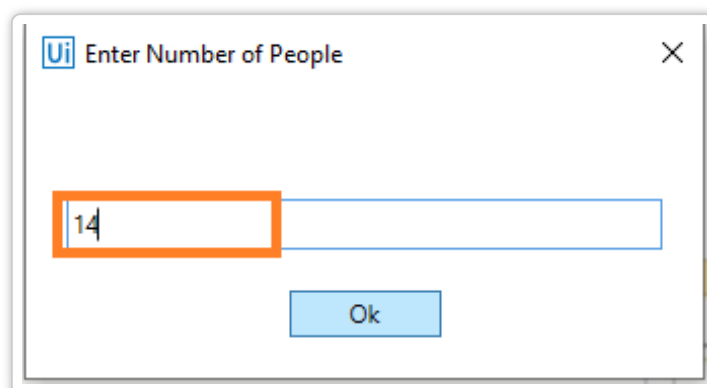
A screenshot of a dialog box titled "Enter Number of People". It features a text input field and an "Ok" button at the bottom.

Next, click on the **Final State activity** and add **Write Line activity** inside the **Entry State**, as you can see only the **Entry State** in the **Final Activity**.



A screenshot of a dialog box titled "Enter Number of People". The text input field contains the number "8". A blue "Ok" button is visible at the bottom.

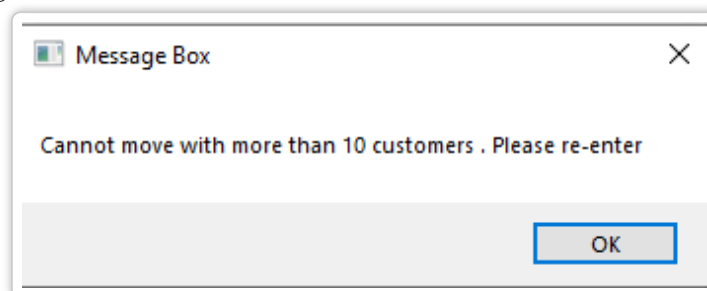
Click on the **T2 Transition** and write the condition for the same as shown below.



A screenshot of a dialog box titled "Enter Number of People". The text input field contains the number "14". A blue "Ok" button is visible at the bottom.

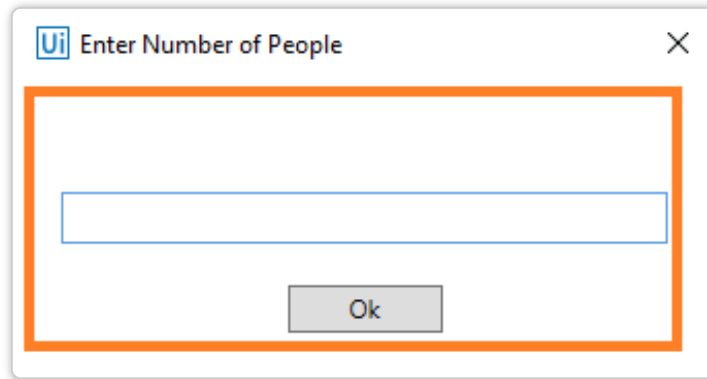
And if the customer number is greater than 10, then we cannot be able to move from one floor to another floor, so create another transition called **T4** in the **Get People activity** and write the condition for the same.

Next, add the **Message box** inside the **Action** and write the text, as shown below.



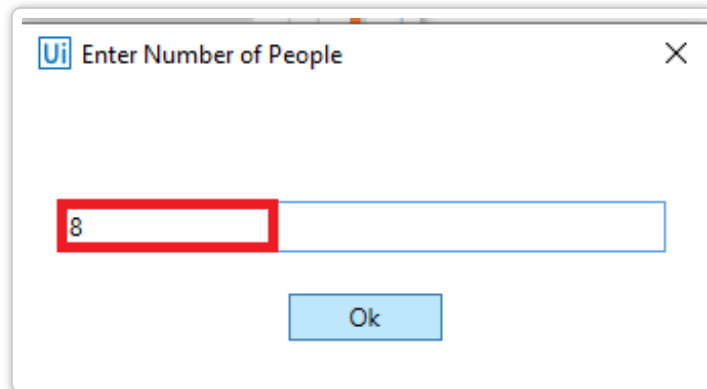
A screenshot of a "Message Box" dialog. It displays the message "Cannot move with more than 10 customers . Please re-enter" and has an "OK" button at the bottom right.

Save and Run the sequence, After the execution, **Enter the Number of People** pop-up message will occur, enter the number of your choice and then **click on ok**.



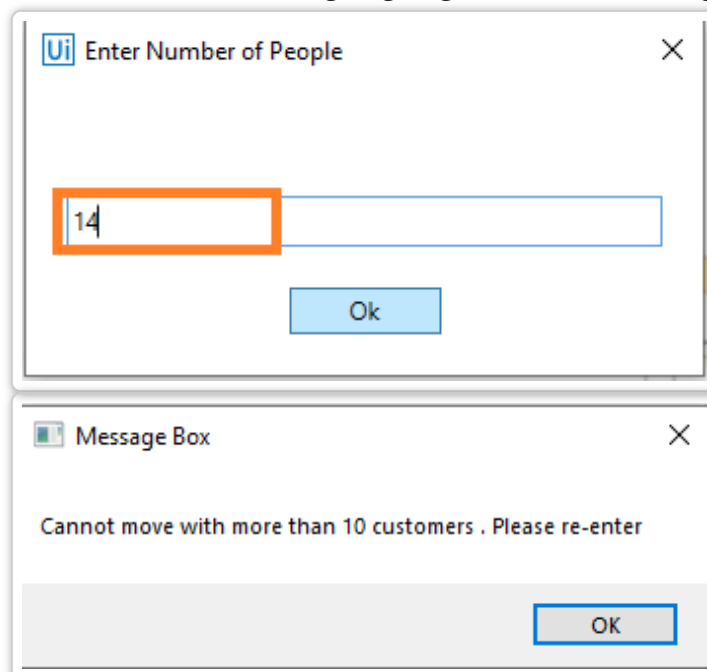
A screenshot of a dialog box titled "Enter Number of People" with a close button (X) in the top right corner. The dialog box contains a text input field and an "Ok" button. The entire content area of the dialog box is highlighted with a thick orange border.

Now **I am entering eight** and click on **ok**, and then it will execute the workflow.



A screenshot of the "Enter Number of People" dialog box. The text input field now contains the number "8", which is highlighted with a red border. The "Ok" button is highlighted in blue.

If Enter **number Greater than ten**, then I am going to get the below message.



Two screenshots are shown. The top screenshot shows the "Enter Number of People" dialog box with the number "14" entered in the text field, which is highlighted with an orange border. The "Ok" button is highlighted in blue. The bottom screenshot shows a "Message Box" dialog box with the text "Cannot move with more than 10 customers . Please re-enter" and an "OK" button.

If you **click on the Ok button**, then it will again ask you to enter the number.

Ui

Enter Number of People

×

Ok
