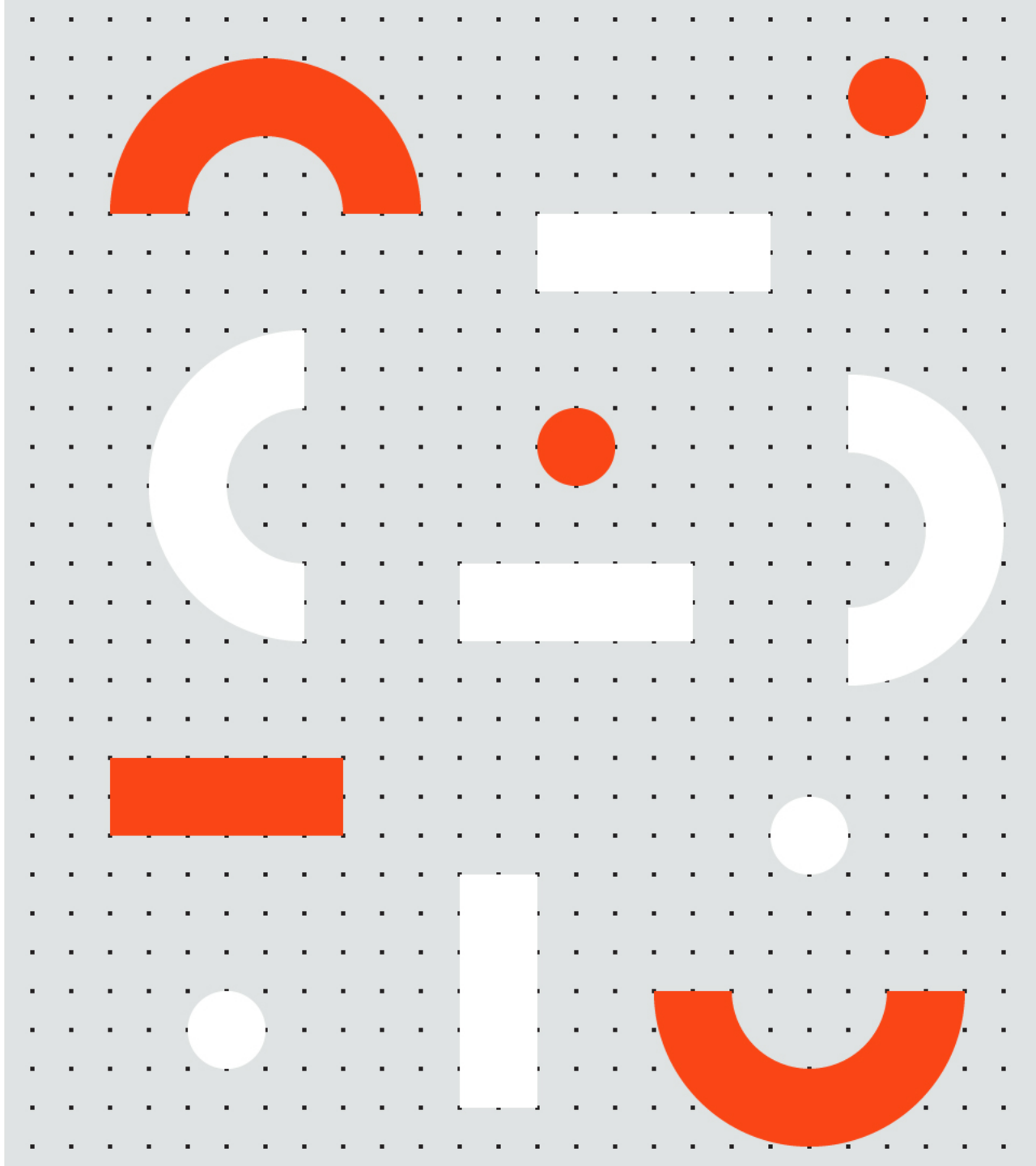
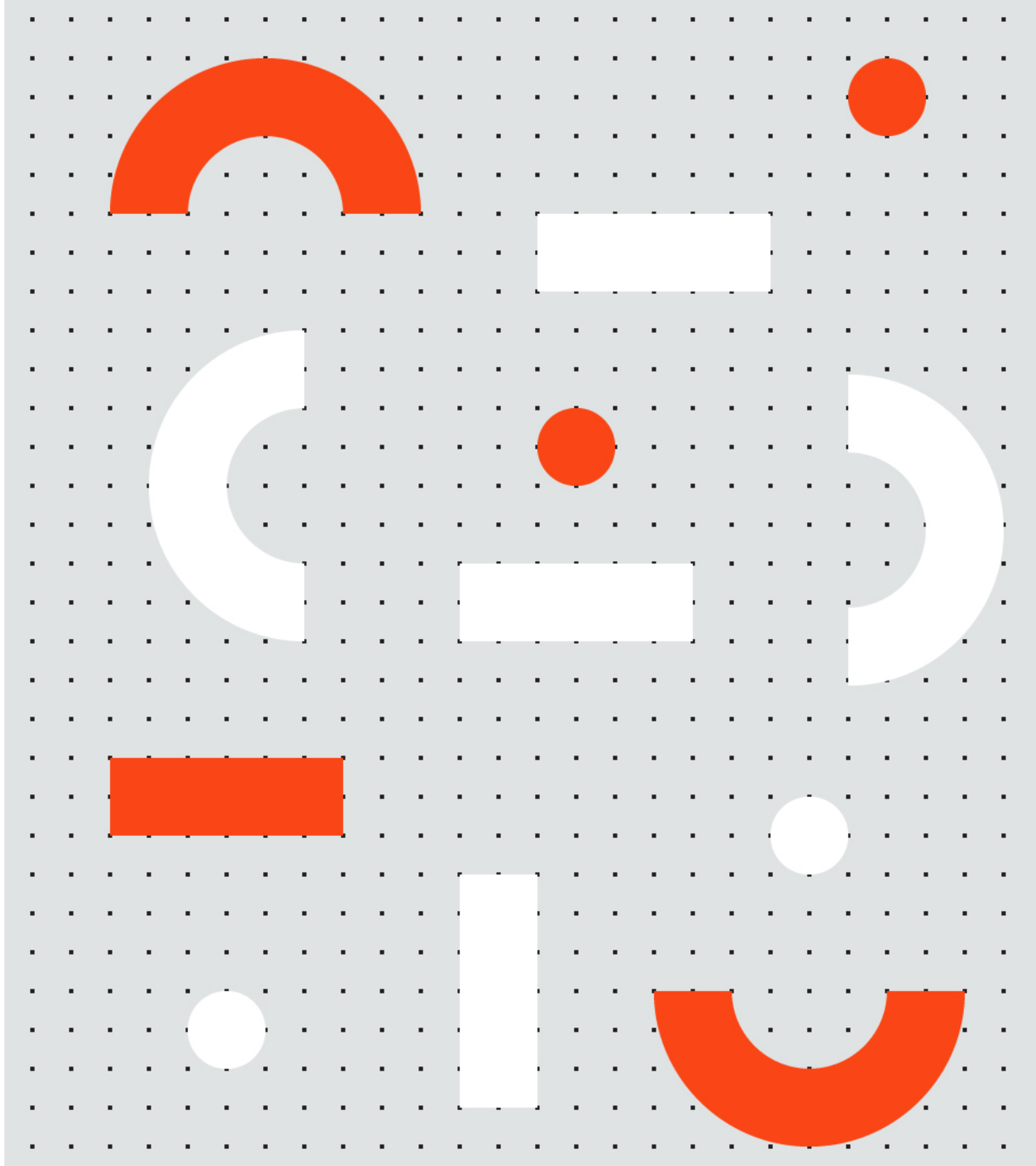
RPA Advanced Developer Training

Foundation Recap



Essential Topics



Essential Topics

01 Control Diagrams

02 Variables

03 Arrays

04 Data Types

05 String Methods

06 Arguments

07 Selectors

08 UI Automation

Essential Topics

Essential Topics

17 Debugging

18 Logging

19 HTTP Requests

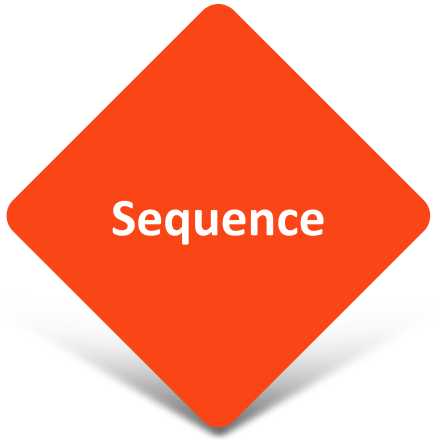
20 Invoke Code

21 Invoke Method

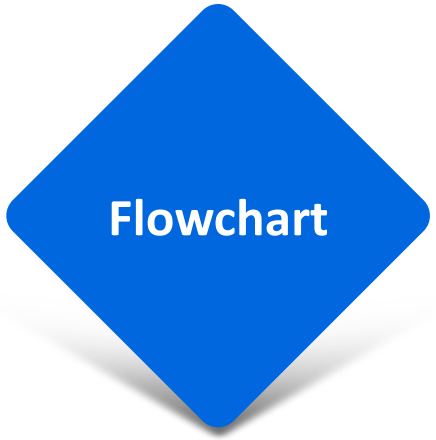
22 Source Control

Project Layouts

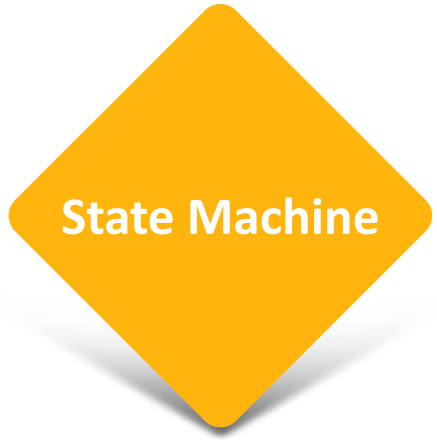
A **project layout** defines how the various components of a project are organized and connected. There are three types of layouts:



A container in which activities are placed one after another and executed linearly



A diagrammatical representation of various steps involved in completing activities connected in multiple ways



A type of automation that uses a finite number of states in its execution

Choosing a Project Layout

An appropriate project layout should be selected for a workflow.

When to use?

Sequence

- when there is a clear succession of steps, without too many conditions

Flowchart

- when there is a complex flow with several conditions
- when there is a flow that runs continuously or that terminates only in specific conditions

State Machine

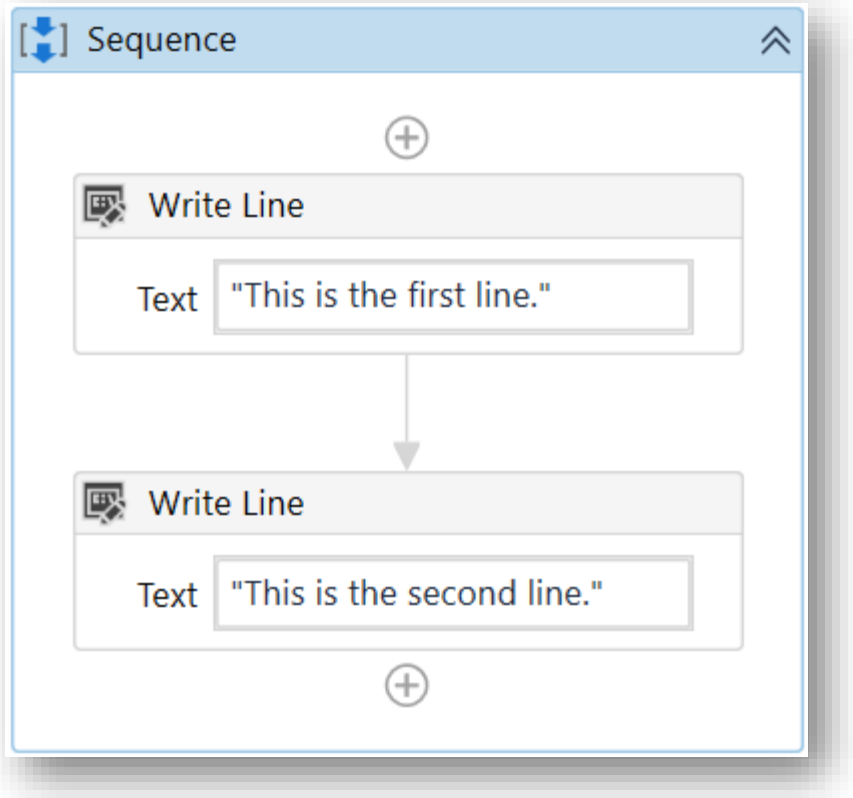
- when there are a finite number of clear and stable states to go through in the workflow
- when there are continuous workflows that are more complex

Control Diagrams - Sequence

A sequence is suitable for workflows when:

- There is a clear succession of steps, without too many conditions

- Using a sequence offers several advantages, such as:
  - Easy to understand and follow due to a top-to-bottom approach
  - Useful for simple logic, like searching for an item on the internet
  - However, it has a disadvantage as well:
  - Nesting too many conditions in the same sequence make the process hard to read



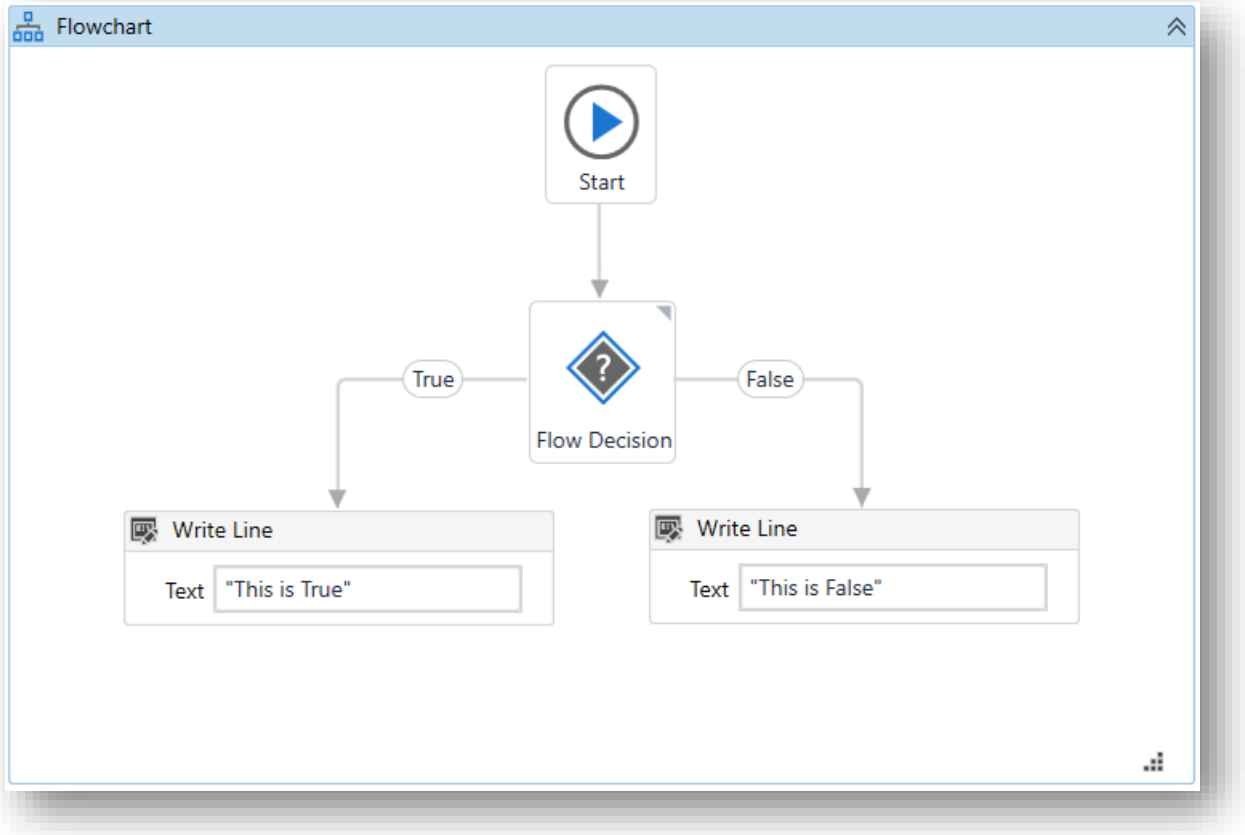
Control Diagrams - Flowchart

A flowchart is suitable for workflows when:

- There is a complex flow with several conditions
- There is a flow that runs continuously, or that terminates only in specific conditions

Using a flowchart offers several advantages, such as:

- Easy to understand
- Can be used for continuous workflows
- However, it has a disadvantage as well:
- Can be used only as a general workflow (with sequences nested inside), not for individual parts of projects (nested inside other workflows)



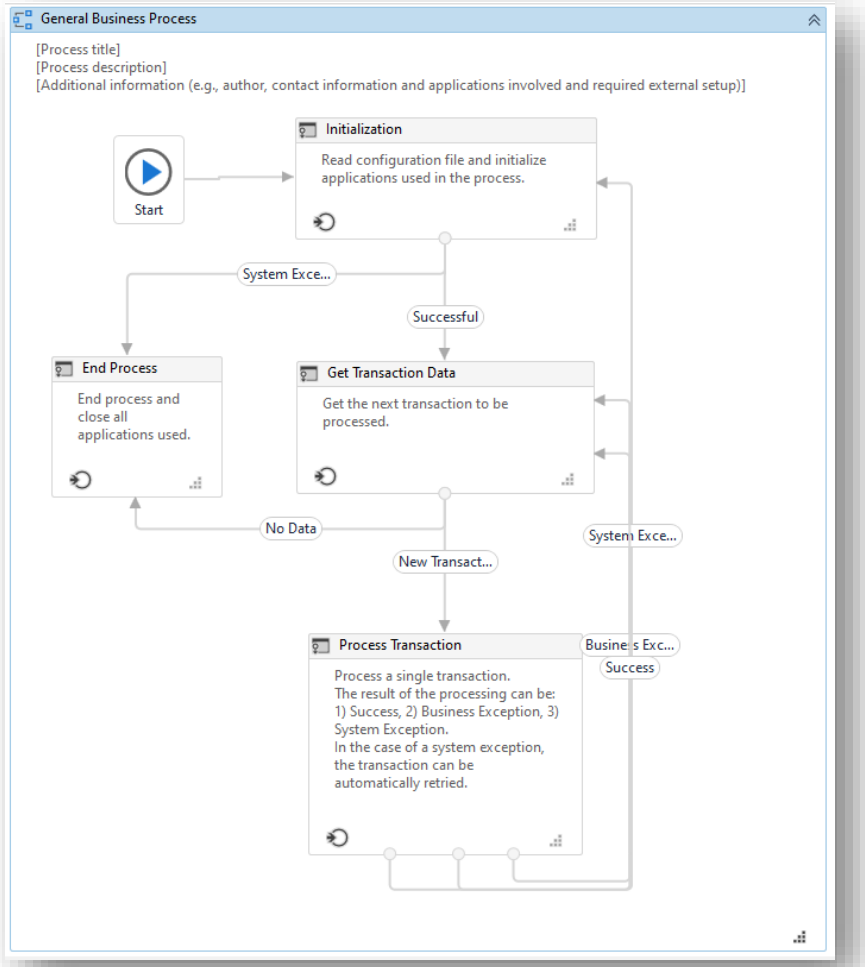
Control Diagrams – State Machine

A state machine is suitable for workflows when:

- There are a finite number of clear and stable states to go through in the workflow
- There are continuous workflows that are more complex

Using a state machine offers several advantages, such as:

- Transitions between states can be easily defined
- Can accommodate processes that are more complex and cannot be captured by simple loops and If statements
- Allows the user to cover all the possible cases/transitions
- However, it has disadvantages as well:
- More complex than sequence and flowchart
- Needs longer development time



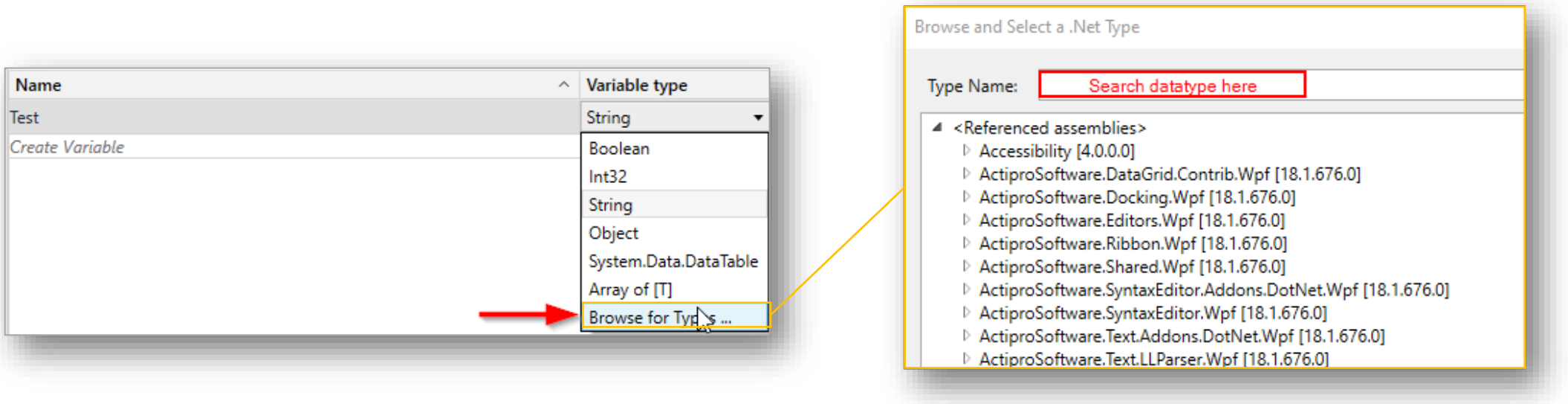
Variables

Variables are containers that can hold multiple data entries (values) of the same data type. Variables are differentiated through their properties:

Name	Type	Value	Scope
The name of the variable, as given by the Developer	What kind of data can be stored in the variable	The data that the variable holds at a given time.	The part of the workflow in which the variable can be used.

Variables

- As a best practice, a variable should have the scope only where it is used (not higher scope).
- Use clear and consistent naming conventions - one of the most common is **Camel Case**.
- You can use any datatype from the .NET platform, UiPath specific types or you can create your own type.



Arrays

The array variable is a type of variable that enables storing multiple values of the same data type, each identifiable through their index. The data type of the objects in the Array can be any .NET data type.

Examples:

- Array of Int32: IntArray = {34, 25, 1}
- Array of String: StrArray = {"hello", "goodbye", "morning", "night"}
- Array of Object: CombinedArray = {"Jenny", 25}

Array Methods

Examples:

- Array of Int32: IntArray = {34, 25, 1}
- Array of String: StrArray = {"hello", "goodbye", "morning", "night"}
- Array of Object: CombinedArray = {"Jenny", 25}

Index

Use parenthesis and index to get a single element. Index starts with 0.

- IntArray(0) -> 34
- IntArray(2) -> 1
- StrArray(3) -> "night"

Length

The length method returns the number of elements in an array.

- IntArray.Length -> 3
- StrArray.Length -> 4

Most Common Data Types Used in UiPath

01

Numeric

Def.: handles numbers on various representations
Sub-types: Int32, Double, Long
E.g.: 32, 23.6

02

Boolean

Def.: refers to a system of logical thought that is used to create true/false statements
Values: True or False

03

Date and Time

Def.: Storing time coordinates or tims durations and having specific manipulation methods
E.g.: DateTime.Now, DateTime.Now.AddDays(-3)

04

String

Def.: a sequence of characters, used for storing text. It has specific manipulation methods
E.g.: “abc”, “123”, “abc123”

05

Collection

Def.: acts as a container for objects of all types.
E.g.: Arrays, Lists, Dictionaries

06

GenericValue

Def.: UiPath proprietary type
Values: number, boolean, string or date

07

DataTable

Def.: Store data in the format of a generic table
E.g.: DT = new Datatable

08

MailMessage

Def.: Stores data of mail object like Subject, From, Body etc
E.g.: mailmessage = new MailMessage

Most Common String Methods

Example:

- MyText = “Test example “

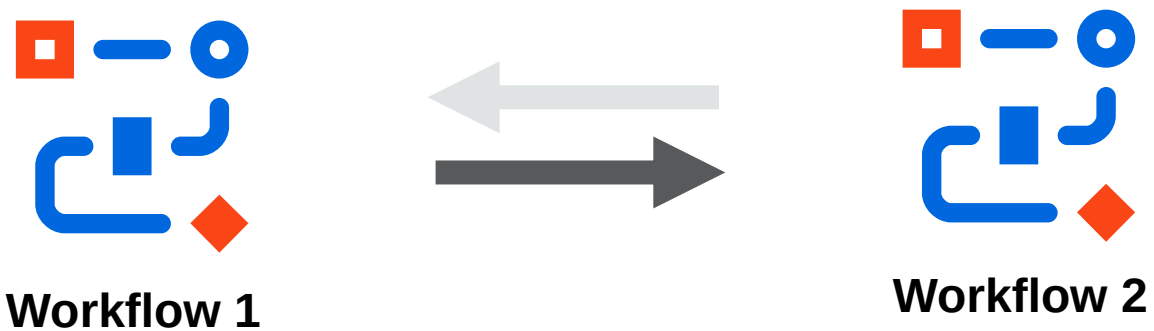
Method	Syntax	Output
Trim/TrimStart/TrimEnd	MyText.Trim	"Test example"
ToLower/ToUpper	MyText.Trim.ToUpper	"TEST EXAMPLE"
Split	MyText.Trim.Split(" "c)	{"Test", "example"}
Contains	MyText.Contains("example")	True
Length	MyText.Trim.Length	12
EndsWith/StartsWith	MyText.StartsWith("Test")	True
Substring	MyText.Substring(5)	"example "
Replace	MyText.Replace("Test", "Hello")	"Hello example "

String Methods - Conversions

Variable Value	Method	Syntax	Conversion type	Output
MyVar = 3	ToString	MyVar.ToString	Int32 to String	“3”
MyVar = “3”	Cint	Cint(MyVar)	String to Int32	3
MyVar = "33.4"	CDbl	CDbl(MyVar)	String to Double	33.4
MyVar = "6/10/2020 11:41:21"	CDate	CDate(MyVar)	String to Date	6/10/2020 11:41:21
MyVar = True	ToString	MyVar.ToString	Boolean to String	"True"
MyVar={"Hello","you"}	String.Join	String.Join(",", MyVar)	String[] to String	"Hello,you"
MyVar = new List(of String) from {"Hello", "there"}	String.Join	String.Join(" ", MyVar)	List<String> to String	"Hello there"

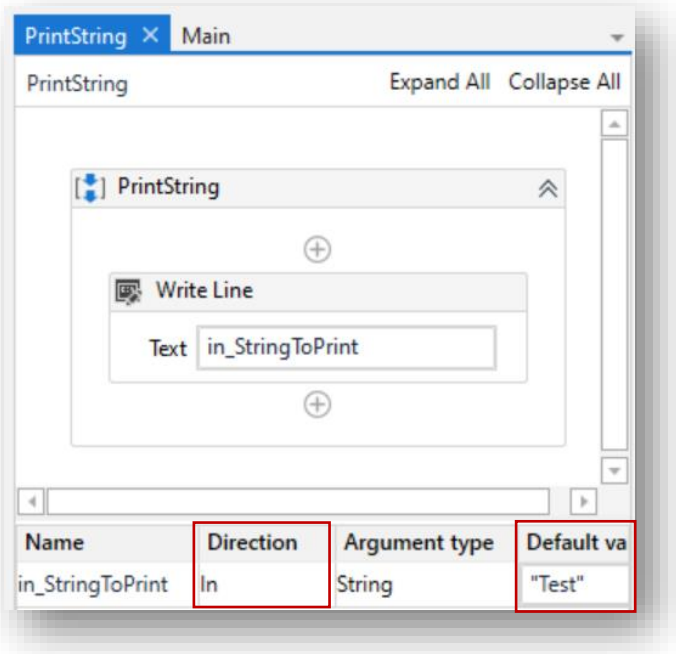
Arguments

- Arguments are a stored location for data.
- Arguments must have a type (like variables).
- Properties:
  - Used to pass data from one workflow to another
  - The value of an argument can be changed though external input, data manipulation or passing data from one activity to another
  - Have an additional property – the direction : in/out/io
- Best practice: the arguments must have the direction as low as possible.
- Note: the value of an argument from invoking the workflow will always overwrite the default value of the argument inside the workflow



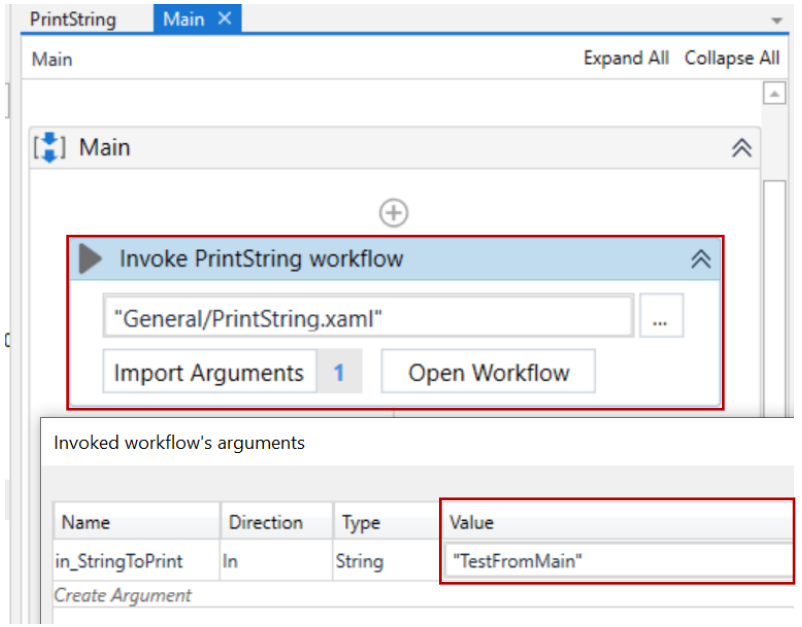
Using Arguments

Step 1



- create a new workflow sequence called "PrintString"
- add a new argument in_StringToPrint
- put a default value for that argument
- run "PrintToString"

Step 2



- from the Main workflow, invoke PrintString workflow
- Click "Import arguments" -> run the Main without changing the value of the argument
- Click "Import arguments" -> change value for argument in_StringToPrint to "TestFromMain" and run Main sequence

Introduction to Selectors

Selectors are used for identifying individual UI elements on the screen.

Selector Editor

- Enables the user to access the automatically generated selectors and edit their attributes
- **Properties:** Validate, Indicate Element, Repair, Highlight

Selector Structure: <node_1/><node_2/>...<node_N/>

Example:

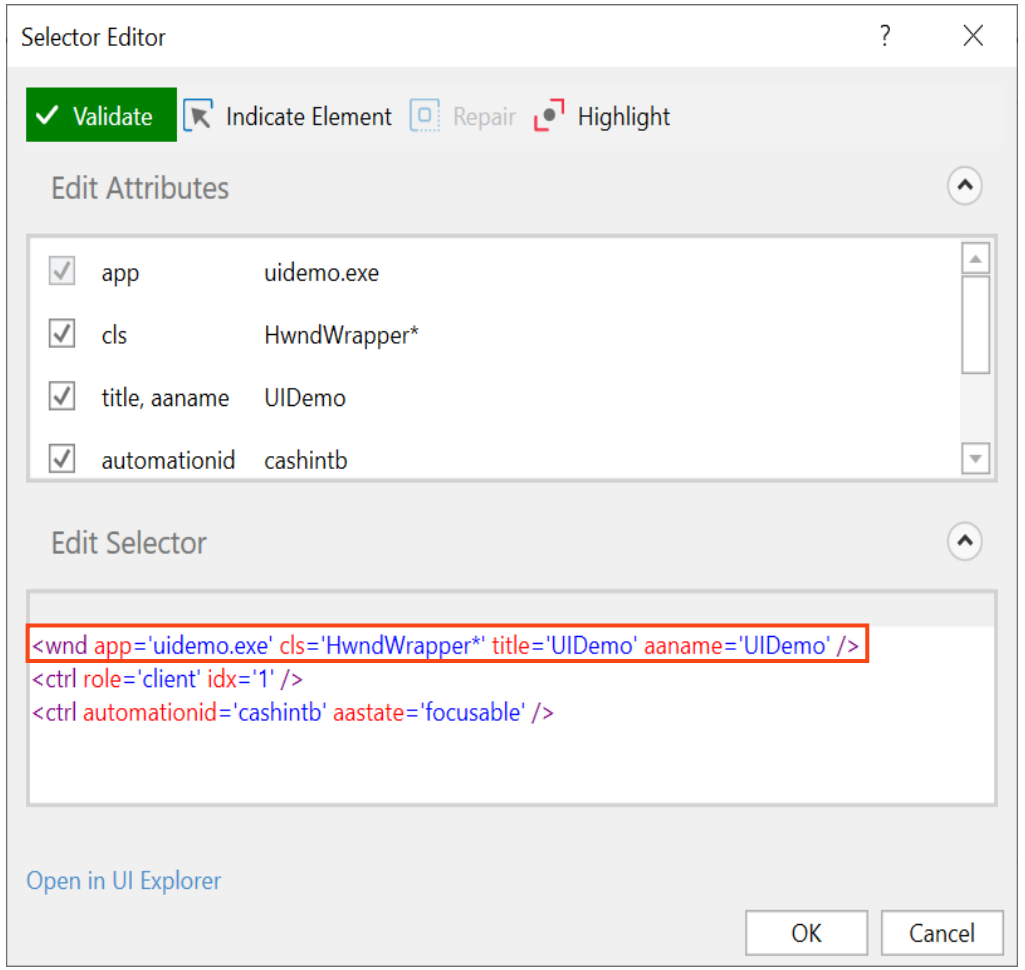
- <wnd app='uidemo.exe' cls='HwndWrapper*' title='UIDemo' aaname='UIDemo' />

Tags:

- Nodes in the selector XML fragment
- First node is the app window
- Last node is the element itself
- In the example, the tag is: wnd

Attributes:

- Help to correctly identify a specific level of the selected application
- Have a name and a value
- In the example, the attributes are:
  - app='uidemo.exe'
  - cls='HwndWrapper*'
  - title='UIDemo'
  - aaname='UIDemo'



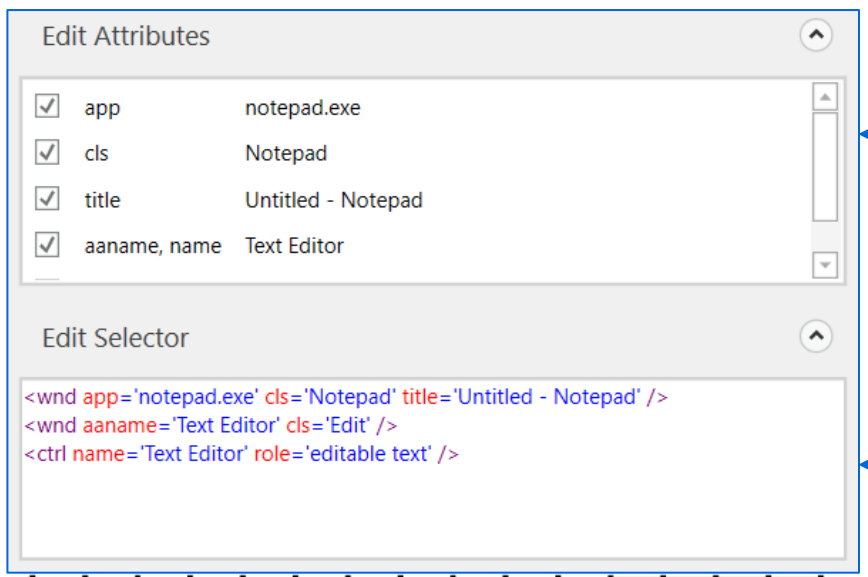
Full Selectors vs. Partial Selectors

Full Selectors

- Contain all the elements needed to identify a UI element, including the top-level window
- Generated by the Basic Recorder
- Best suited when the actions performed require switching between multiple windows

Partial Selectors

- Don’t contain the information of the top-level window; thus, the activities with partial selectors must be enclosed in containers
- Generated by the Desktop Recorder
- Best suited for performing multiple actions in the same window



Editor Section

Explorer Section

Dynamic Selectors

Used to find the address of a UI element dynamically and identify the attributes of the element across windows.

Description:

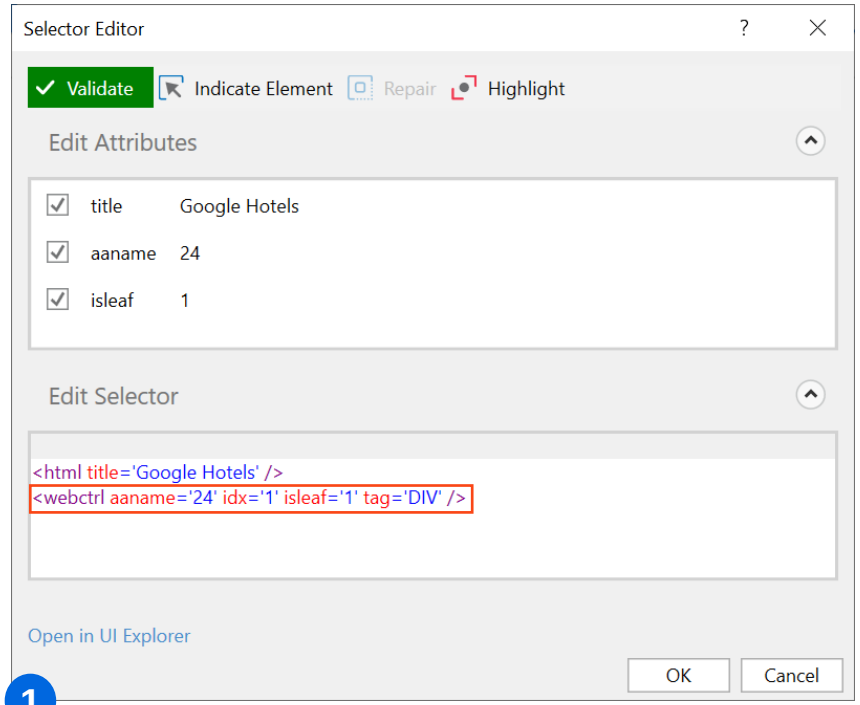
- Uses a variable or an argument as a property for the attribute of the target tag
- Allows easy identification of a target element based on the value of the variable or argument
- Best suited for situations in which the targeted element constantly changes its value

Format:

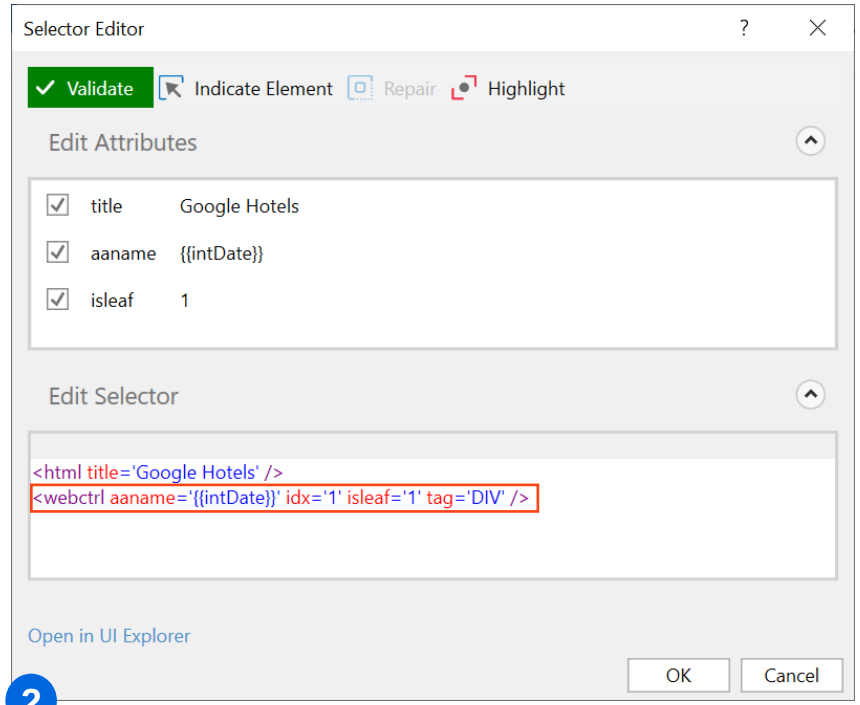
- <tag attribute=`{{Value}}` />

Wildcard:

- A special character that can replace the dynamic part of a selector
- Two types of wildcards: Asterisk and Question mark



1 Date attribute value is constant



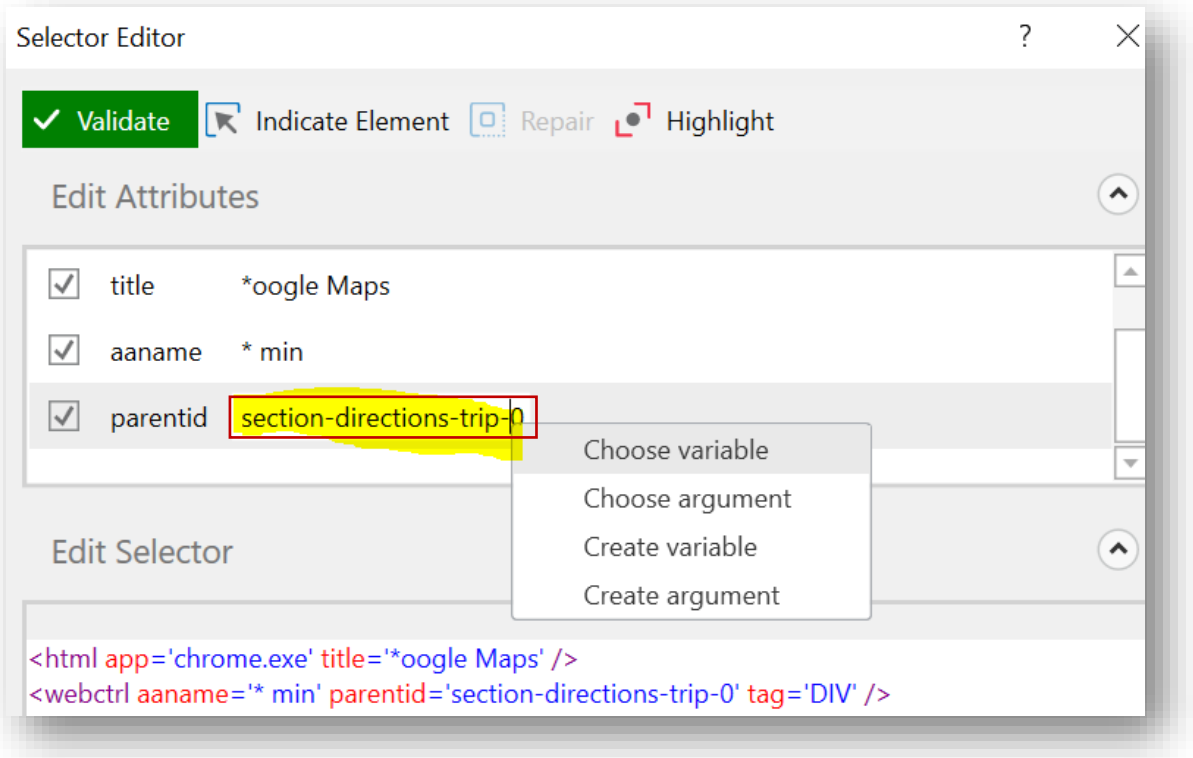
2 Date attribute value is dynamic

Dynamic Selectors

Variables

If the value of an attribute isn't always the same, but the robot knows what it should be when running the automation, we can add **variable** parts to the selector:
To Make an attribute dynamic we can use:

- By creating a variable or an argument(see image)
- By selecting an existing variable or argument(see image)
- By changing the string of the selector (directly in the properties, without opening selector editor)



Selectors – Dynamic Selectors

Wildcards

There are two types of wildcards:

- Asterisk (*) – replaces zero or more characters
- Question mark (?) – replaces a single character

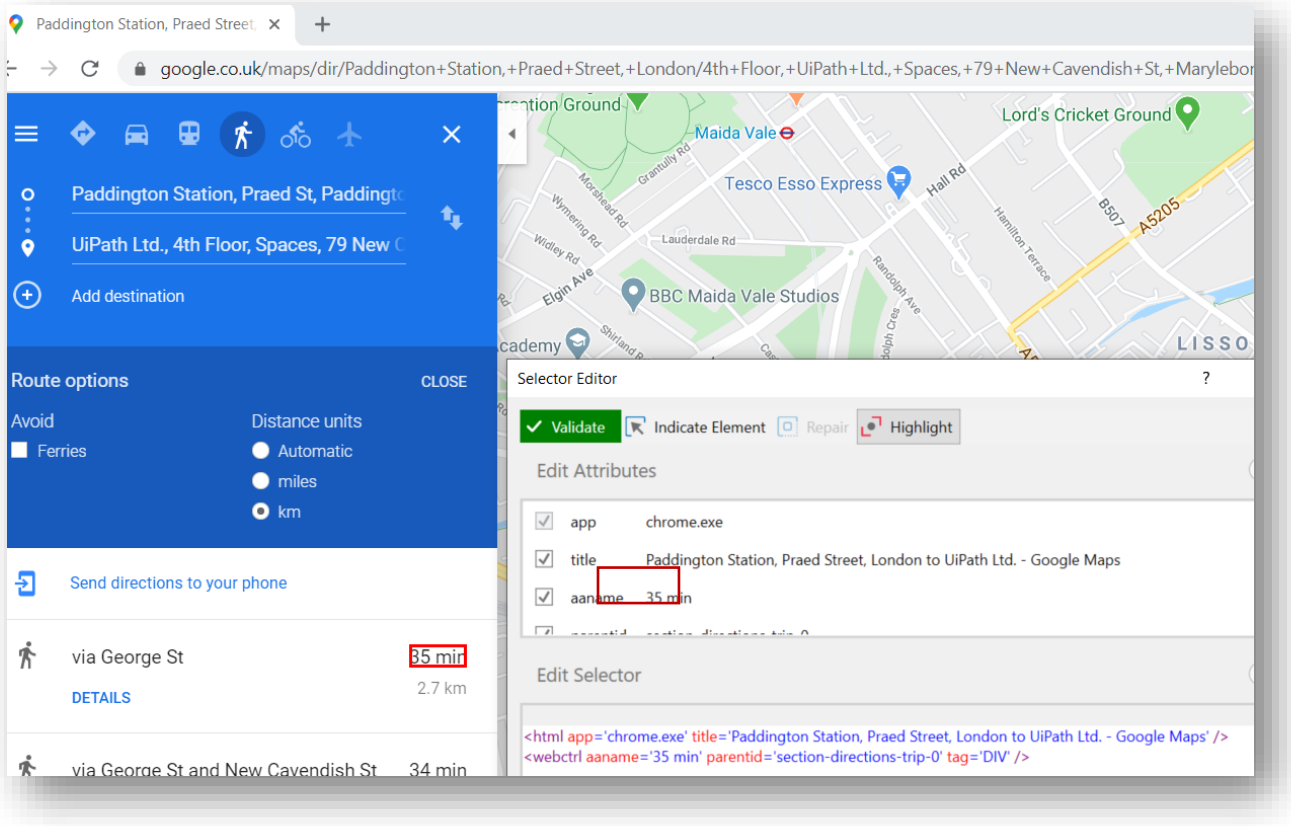
Example:

What would happen on the example if we changed the starting point?

- The attribute title would likely change
- The aaname could become “45 min”

To Make an attribute dynamic we can use:

- ‘*’ to replace any character: “* min”
- The aaname could become “45 min”



Fine Tuning of Selectors

The process of refining selectors in order to have the workflow correctly executed. The reliability of selectors can be improved by using:

Relative Selectors

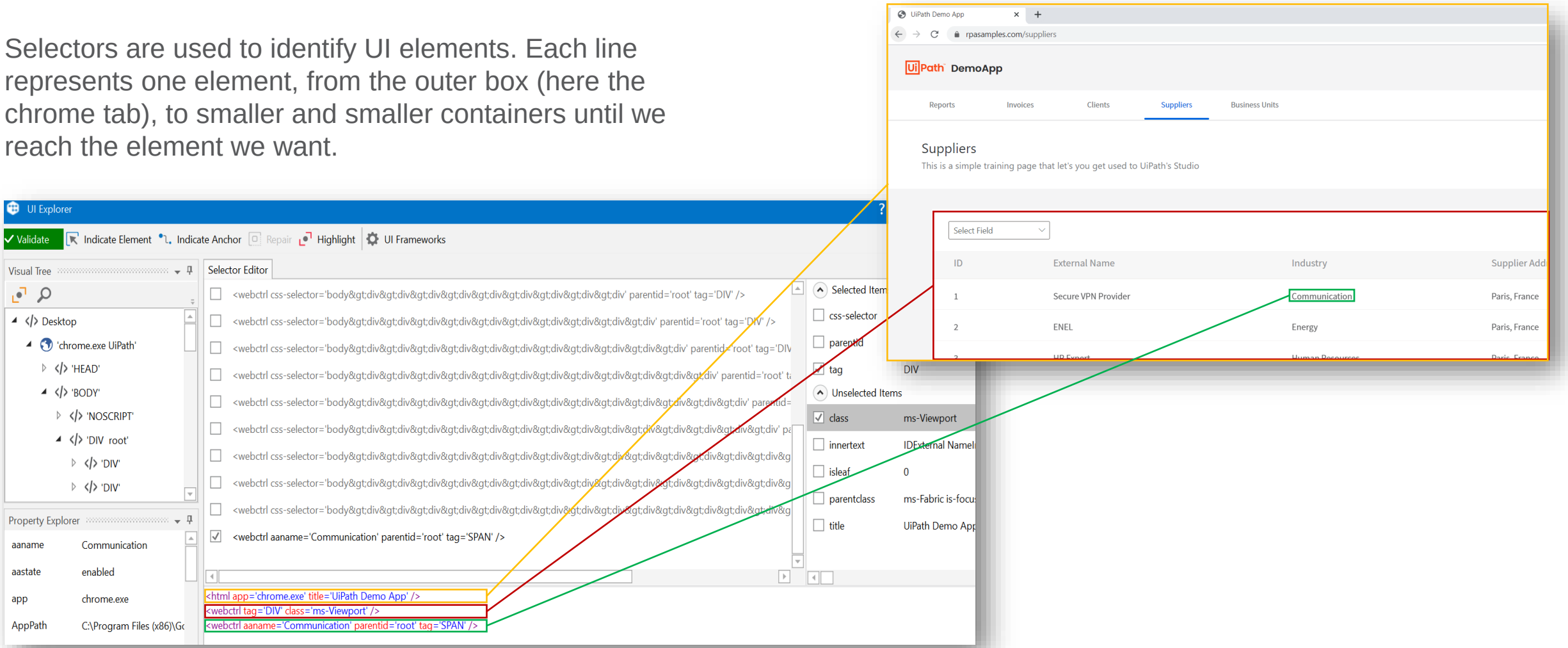
Selectors can be improved by using the ‘Indicate Anchor’ option in UI Explorer

Anchor Base

This is an activity with two parts: the first will locate the anchor, and the second will perform the desired action

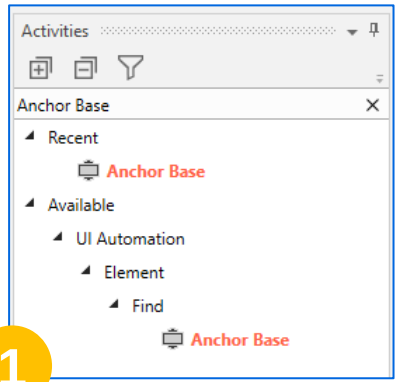
Selectors – UI Explorer

Selectors are used to identify UI elements. Each line represents one element, from the outer box (here the chrome tab), to smaller and smaller containers until we reach the element we want.



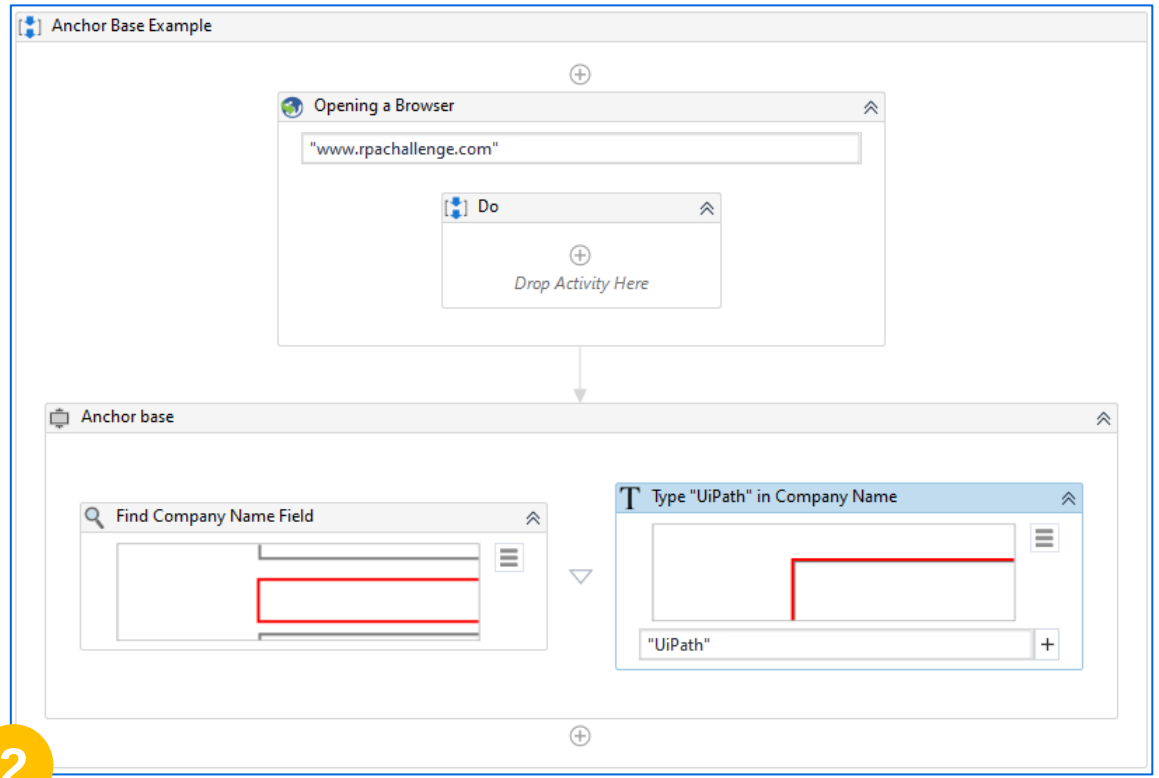
Selectors - Anchors

Anchors are labels used to interact with an element that has an unstable selector.



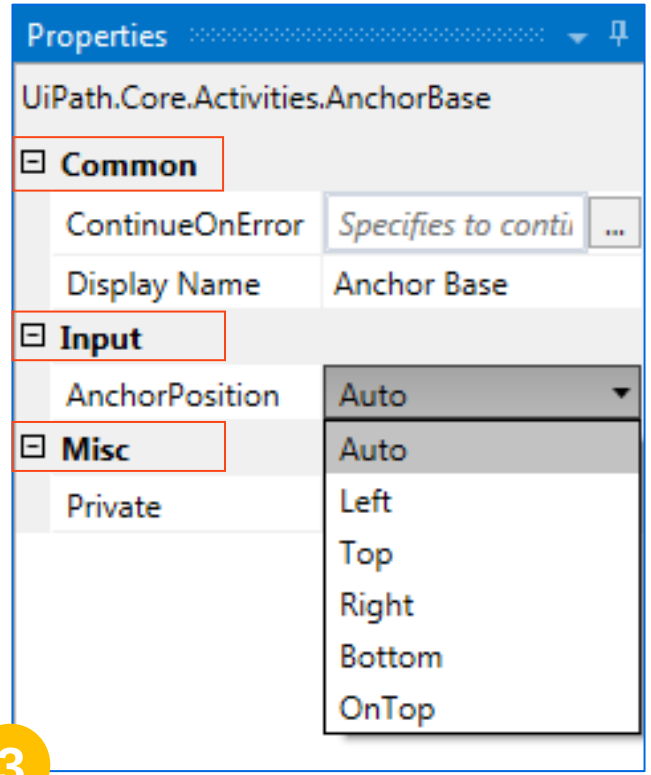
1

Anchor Base activity in Activities panel searches for a UI element by using other UI elements as anchors



2

View of Anchor Base activity in the Designer panel



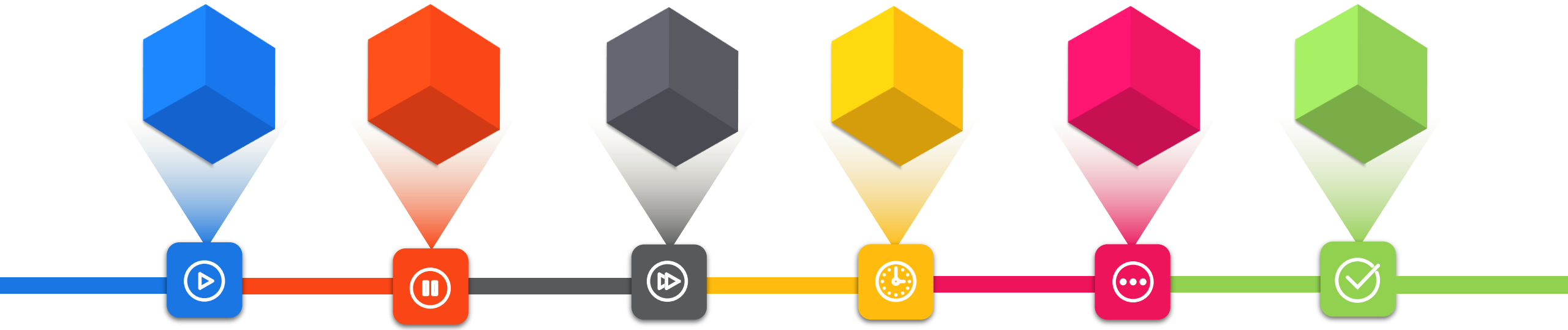
3

Three categories of properties:

- Common
- Input
- Miscellaneous

Common Properties of UI Activities

The **common properties** of UI activities are:



Continue on Error
Specifies whether the automation should continue if it shows an activity error

Delay After
Adds a pause after the activity, in milliseconds

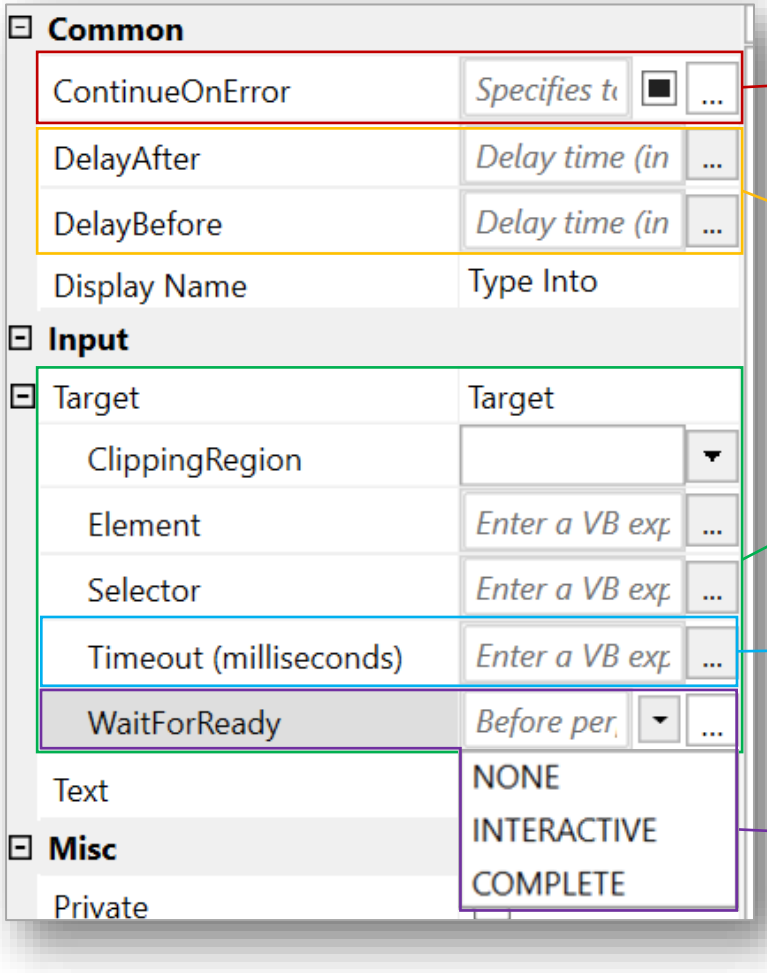
Delay Before
Adds a pause before the activity, in milliseconds

Timeout MS
Specifies the amount of time (in milliseconds) to wait for a specified element

Wait for Ready
Before performing the actions, waits for the target to become ready

Target
Identifies the UI element with which the activity works

UI Activities



ContinueOnError: continues even if it doesn't find the element

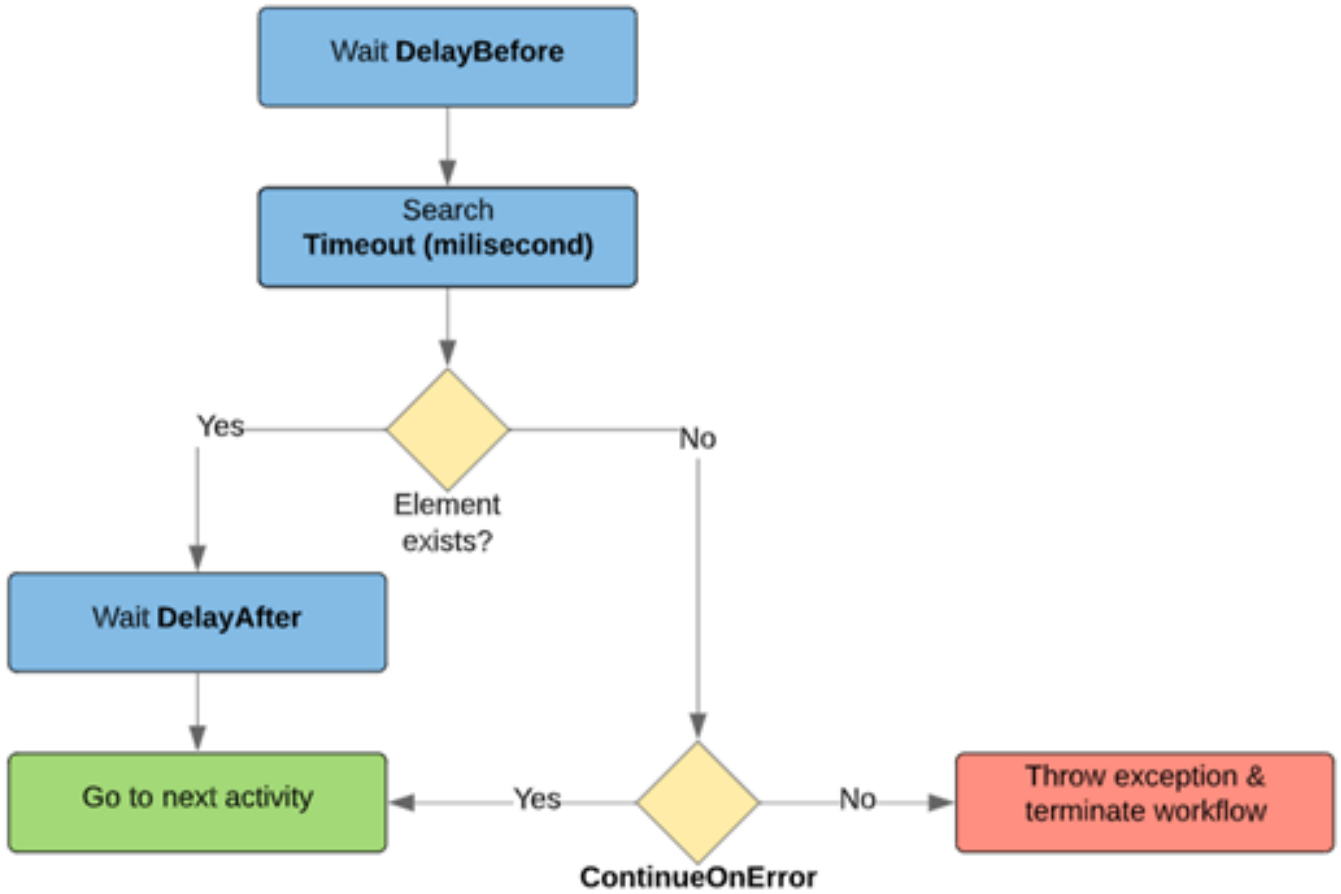
Delay After/Before: adds delay in milliseconds before or after

Target: defines the target element

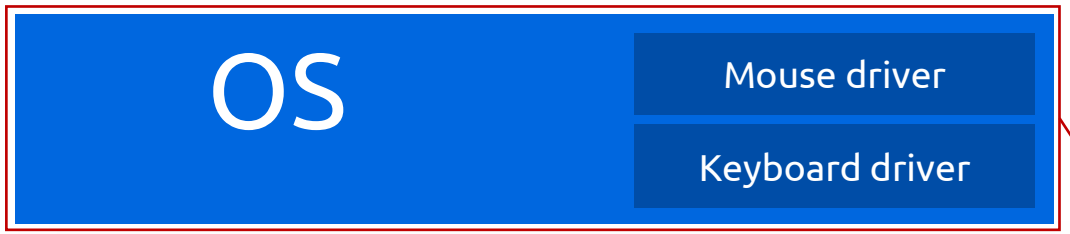
Timeout: how long the robot will keep looking for the target element

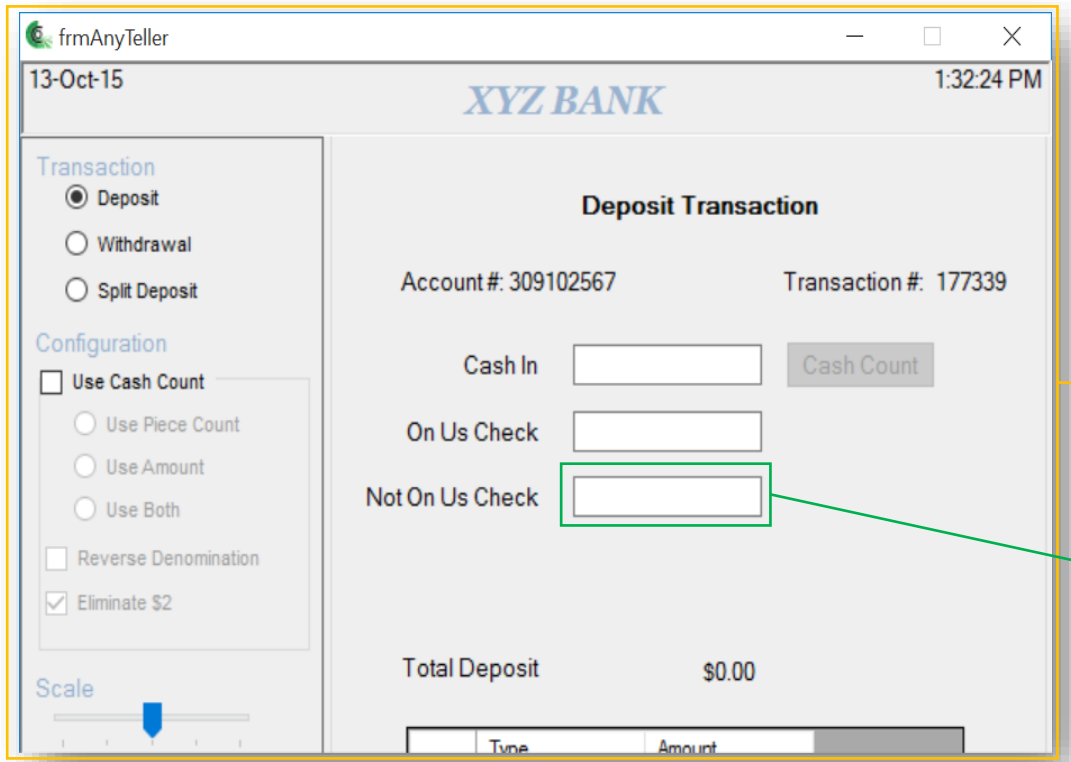
WaitForReady: NONE/INTERACTIVE/READY wait for the whole page to load or not

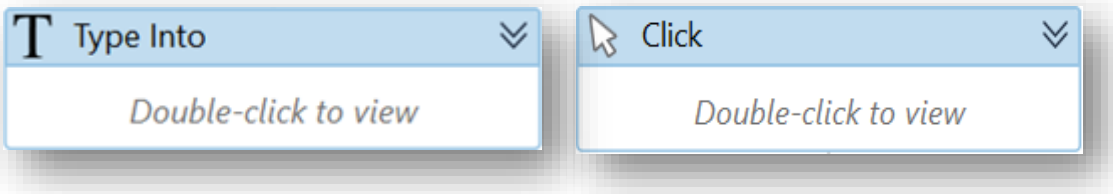
UI Activities Timeline

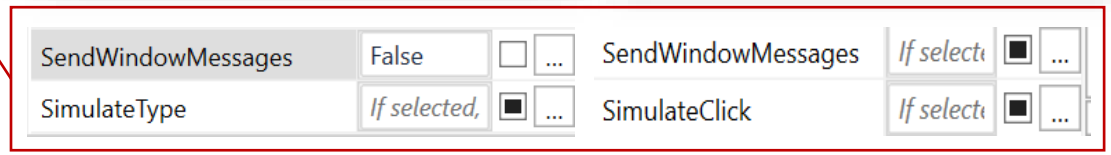


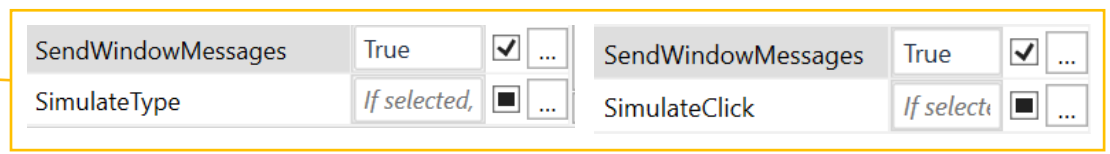
UI Activities

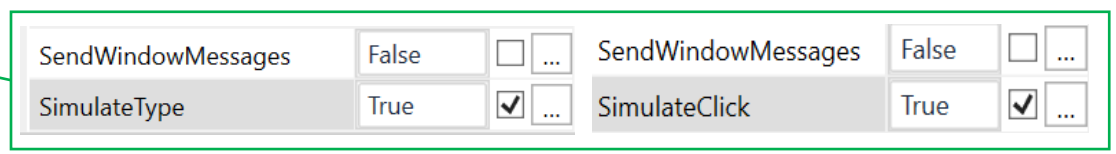












UI Automation – Input Methods

	Uses the driver	Uses OS Window Messages	Uses software's tech	Can add hotkeys	Works in the Background	Application needs to be in the foreground
Hardware Events	✔			✔		✔
Window Messages		✔		✔	✔	
Simulate Type / Click			✔		✔	

Comparison of Input Methods

	Default	SendWindowMessages	Simulate Type/Click
Strong Points	• Supports special keys like Enter, Tab, and other hotkeys	• Supports special keys like Enter, Tab, and other hotkeys • Users can work on other activities during the execution of the automated processes	• Can automatically erase previously written text • Users can work on other activities during the execution of the automated processes
Limitations	• Does not automatically erase previously written text • Does not work in the background	• Does not automatically erase previously written text • Works only with applications that respond to window messages	• Does not support special keys like Enter, Tab, and other hotkeys • Has lower compatibility than the other two methods

Extraction and its Techniques

Extraction is the process of retrieving data from a data source for further processing or storage. There are several extraction techniques available in UiPath. These are:

Output Actions

Get Text
Extracts a text value from a specified UI element

Get Full Text
Extracts a string and its information from an indicated UI element using the FullText screen scraping method

Get Visible Text
Extracts a string and its information from an indicated UI element using the Native screen scraping method

Get OCR Text
Extracts a string and its information from an indicated UI element using the OCR screen scraping method

Other Techniques

- Data Scraping
- Screen Scraping
- PDF Extraction

UI Automation - Output Methods

There are three screen scraping (or output) methods available in UiPath:

FullText

- is the default method;
- the fastest, it has 100% accuracy and can work in the background;
- it can extract hidden text;
- it doesn't support virtual environments and it doesn't capture text position and formatting.

Native

- is compatible with applications that use Graphics Design Interface (GDI);
- It can extract the text position and formatting;
- has 100% accuracy on the applications that support GDI;
- Its speed is somewhat lower than FullText, it doesn't extract hidden text and it cannot work in the background and it doesn't support virtual environments.

OCR

- is the only output method that works with virtual environments and with "reading" text from images;
- it cannot work in the background, it cannot extract hidden text, and its speed is by far the lowest;
- Its accuracy varies from one text to another and changing settings can also improve the results.

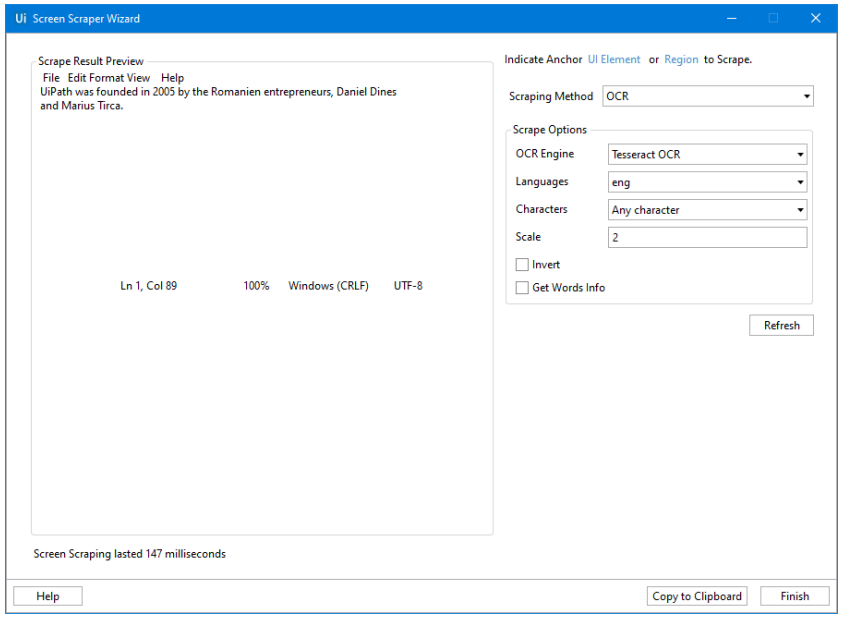
	Speed	Accuracy	Background	Extract Text Position	Extracts Hidden Text	Supports Citrix
FullText	10/10	100%	YES	NO	YES	NO
Native	8/10	100%	NO	YES	NO	NO
OCR	3/10	98%	NO	YES	NO	YES

Types of OCR

OCR output method has two default engines: Google Tesseract and Microsoft MODI.

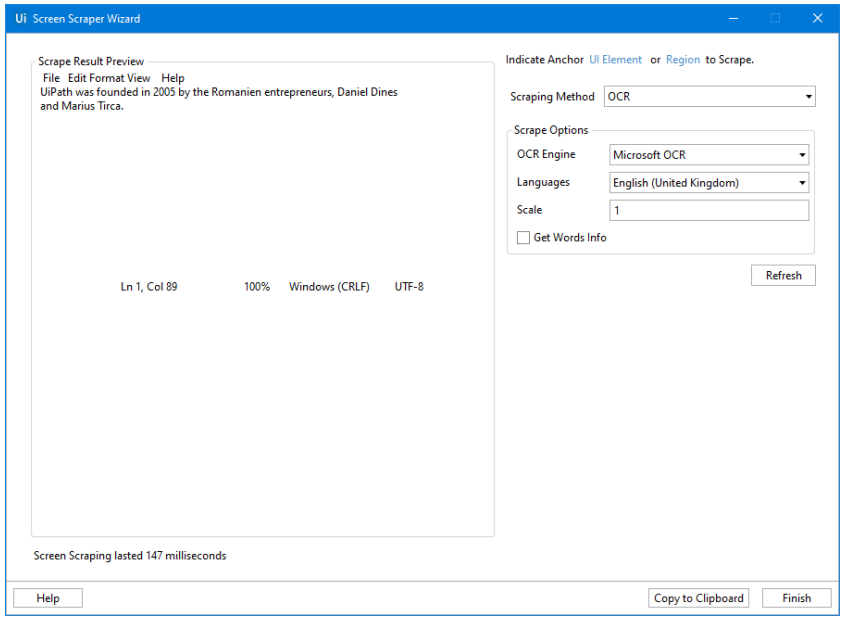
Google Tesseract OCR

- Used for character recognition on smaller size areas
- Supports color inversion



Microsoft OCR

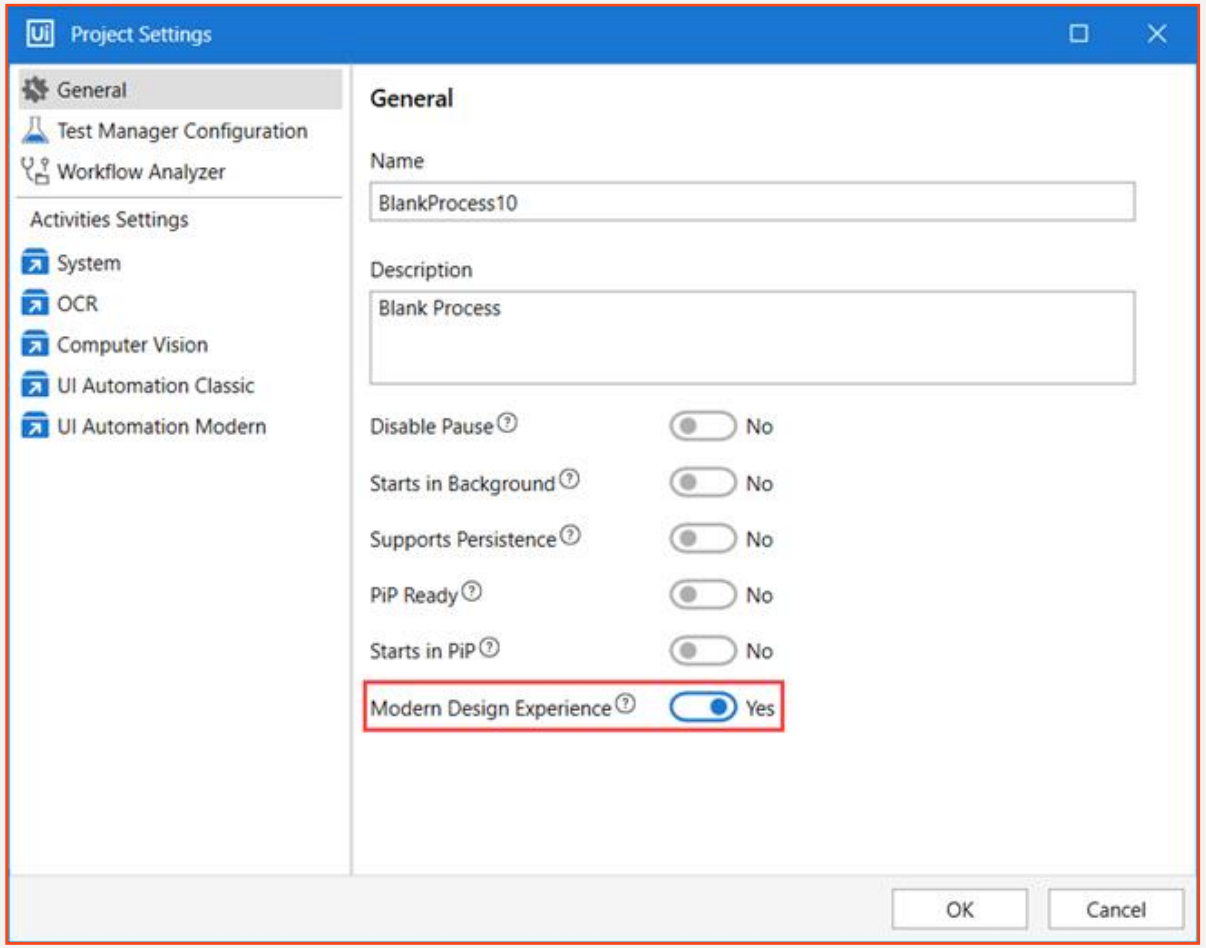
- Used to work with Microsoft fonts and on larger size areas
- Supports multiple languages



UI automation using Modern Experience

Modern Experience is a way in which you identify, configure, and verify target UI elements with new activities, an all-in-one recorder, and a new data scraping wizard. It leverages new unified method of targeting UI elements for automation called **Unified Target**.

The modern design experience is not enabled by default. You can enable the modern experience for each project and configure a global setting that makes it the default experience for all new projects. The project-level setting overrides the global setting.



UI Activities in Modern Experience and Classic Experience

Modern Activities	Classic Activities
Check App State	On Element Appear, On Element Vanish, On Image Appear, On Image Vanish, Wait Element Vanish, Wait Image Vanish, Find Image, Image Exists
Check/Uncheck	Check
Click	Click, Double Click Click Image, Double Click Image
Extract Table Data	Extract Structured Data, Get Full Text, Get Visible Text
Get Text	Get Text
Go to URL	Navigate To
Highlight	Highlight
Hover	Hover, Hover Image

Modern Activities	Classic Activities
Keyboard Shortcuts	Send Hotkey
Navigate Browser	Close Tab, Go Back, Go Forward, Go Home, Refresh Browser
Select Item	Select Item
Take Screenshot	Take Screenshot
Type Into	Type Into, Type Secure Text
Use Application/Browser	Open Application, Open Browser, Attach Window, Attach Browser, Element Scope, Close Window Start Process
	Anchor Base, Context Aware Anchor

DataTables

Is the type of variable that can store data as a simple spreadsheet with rows and columns, so that each piece of data can be identified based on their unique column and row coordinates.

How to create a datatable:

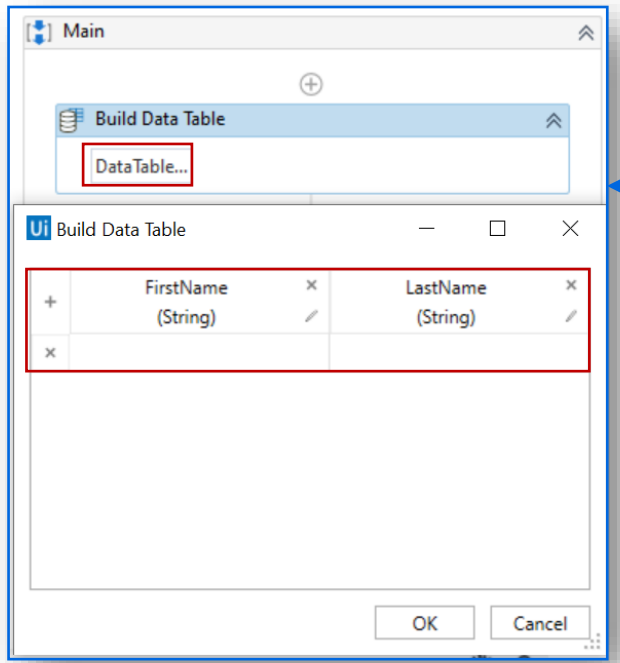
- **Build Data Table** activity
- **Read range** activity

(from Excel file)

- **Read CSV** activity
- **Data scraping** feature

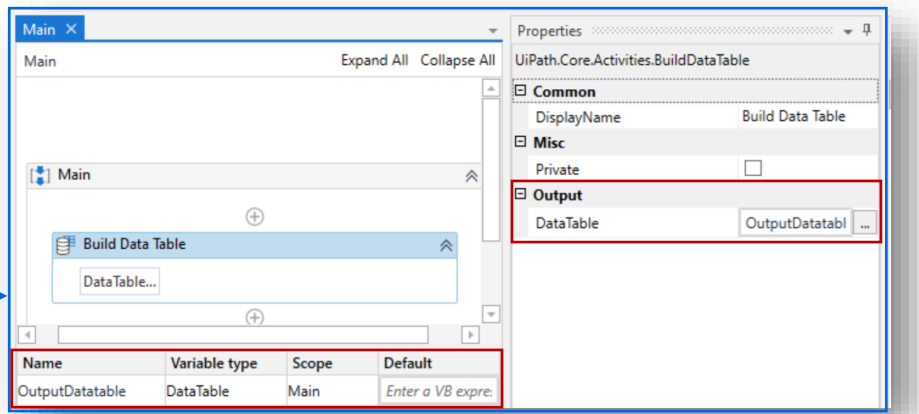
Activities (some examples):

- **Add Data Column**
- **Add Data Row**
- **Filter Data Table**
- **Join Data Table**
- **Output Data Table**



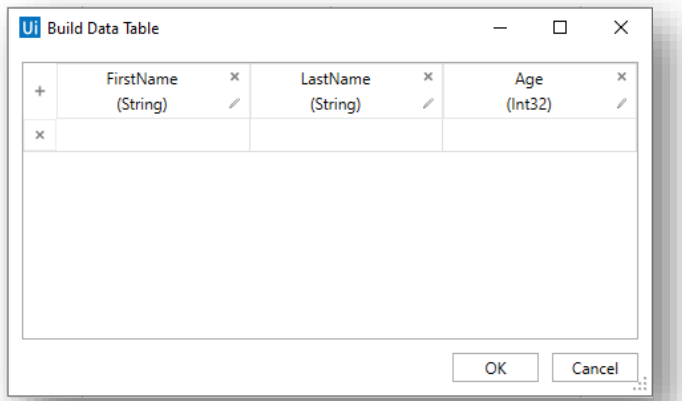
01

02

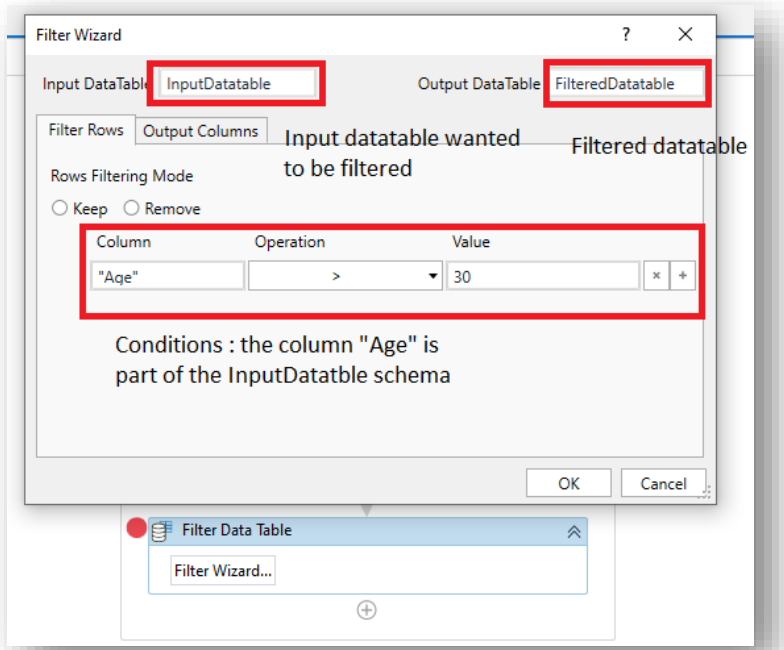


DataTable Filtering

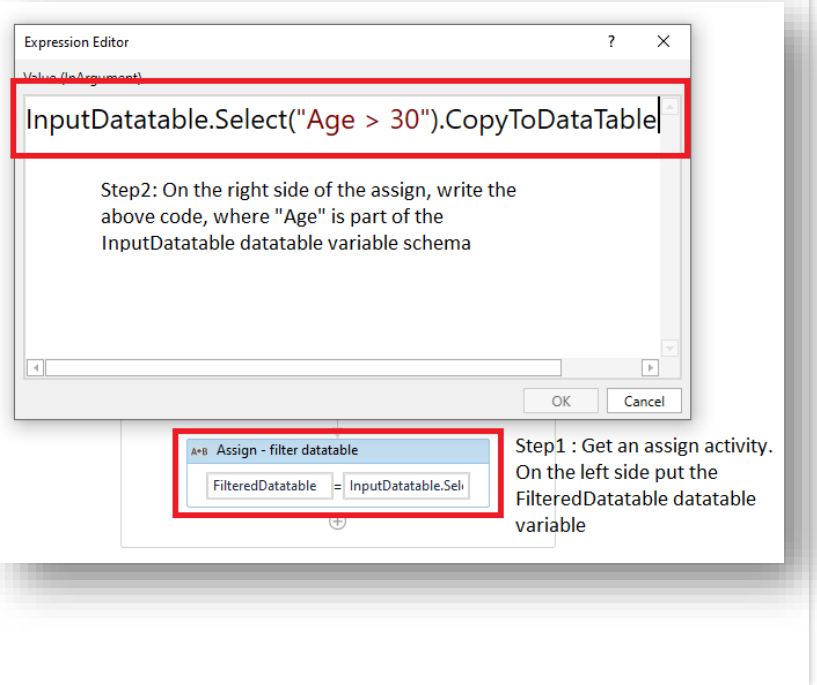
There are many ways of filtering a datatable. Example : for a table with the following schema:



The Filter Data Table Activity



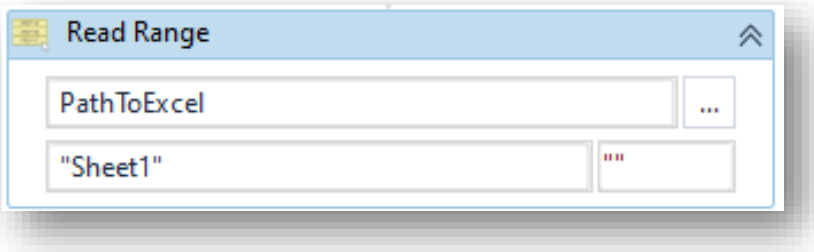
The Select Method

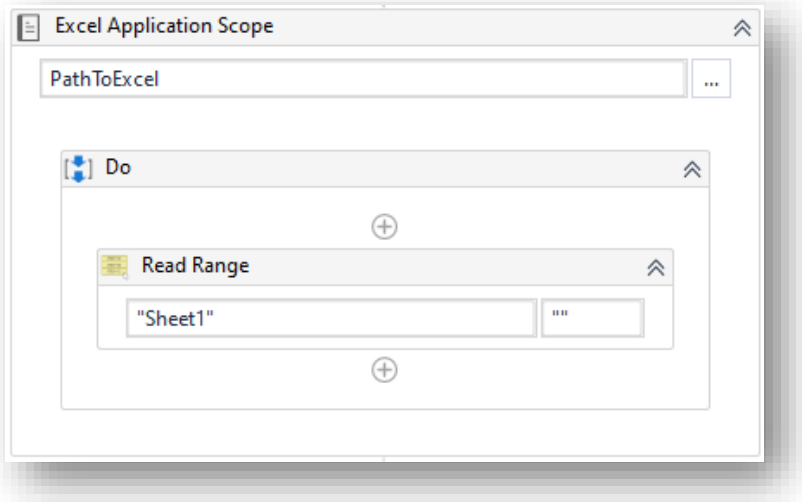


Play with the regex expression used into the Match or IsMatch Activities, you can use any online tool to understand the structure of a regex expression.

Excel Activities

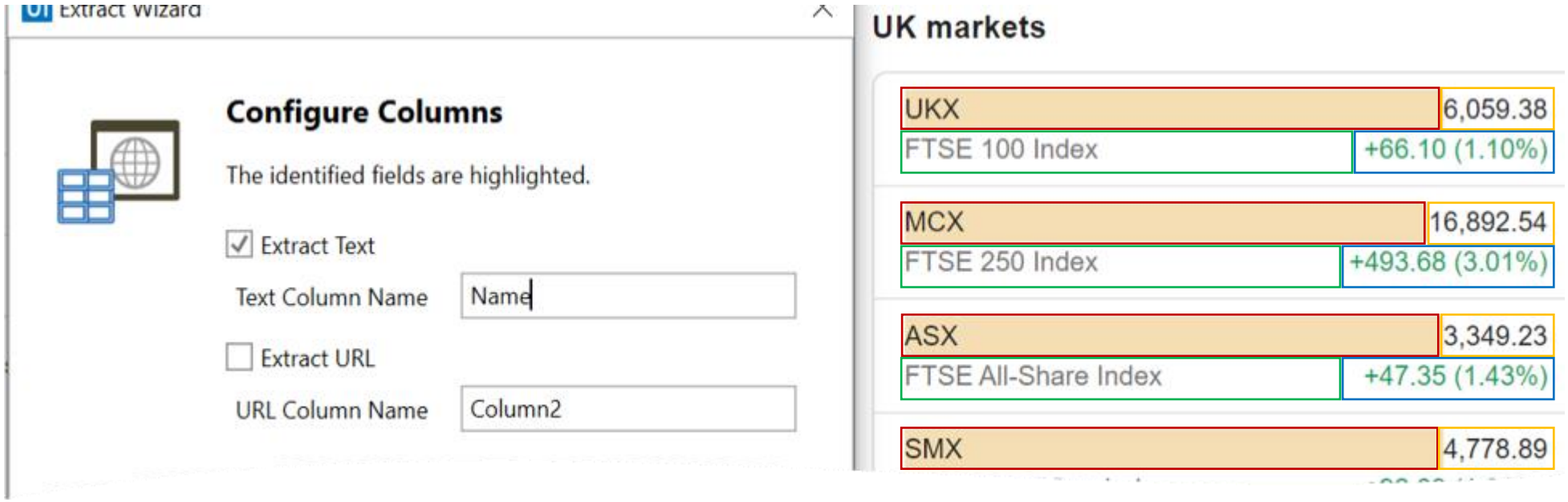
Workbooks	Excel Application Scope
All workbook activities will be executed in the background	All activities can be set to either be visible to the user or run in the background
Doesn't require Microsoft Excel to be installed	Microsoft Excel must be installed, even when ‘Visible’ box is unchecked
Can be faster and more reliable for some operation	If the file isn’t open, it will be opened, saved and closed for each activity
Works only with .xlsx files	Works with .xls and .xlsx, and it has some specific activities to work with .csv





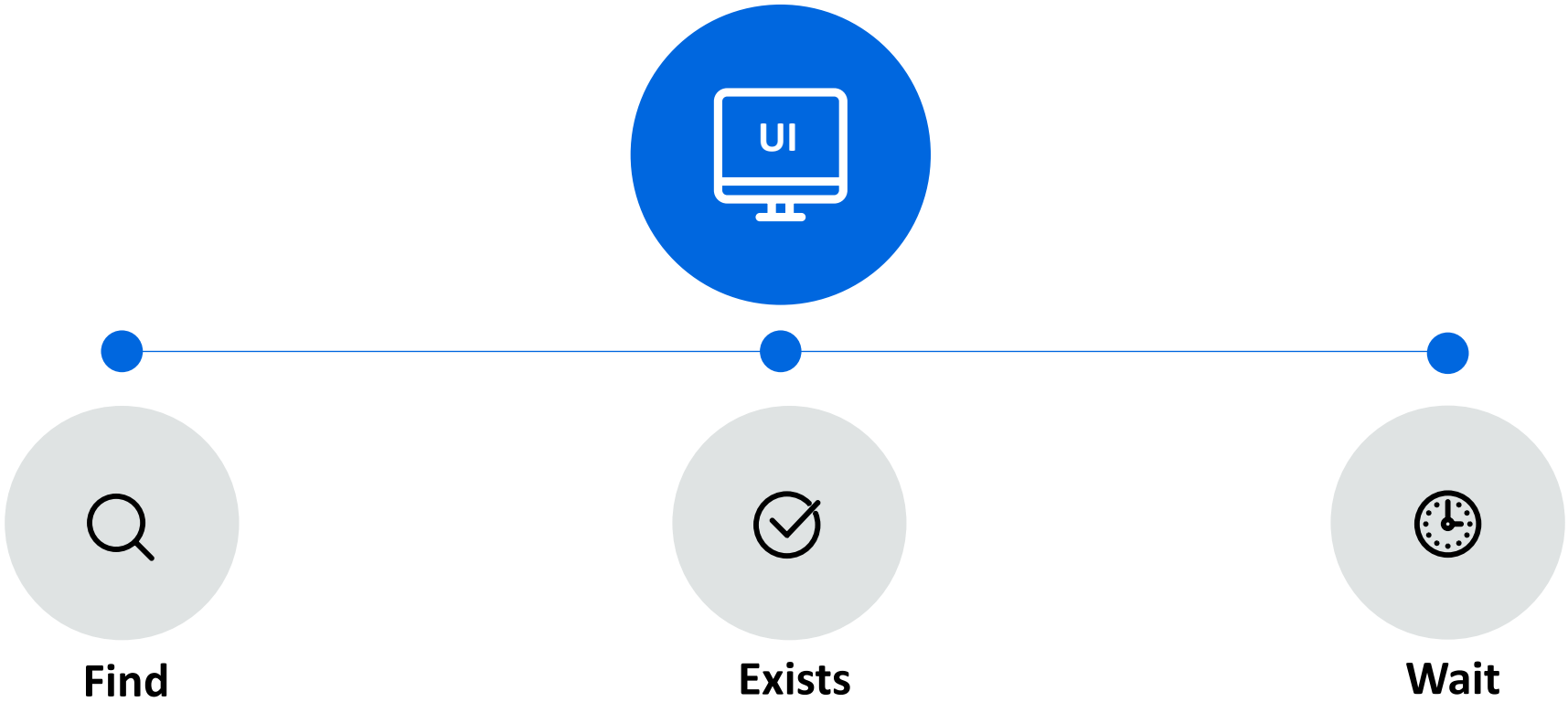
Data Scraping

- Extract structured data from the **web, PDF, java apps, SAP**, even paginated data
- Options to extract URL, extract multiple pages, extract correlated data
- Outputs a datatable with the extracted data



UI Synchronization Activities

UI synchronization is used to address the unexpected behavior of an application in the workflow by using different activities. These activities can be categorized into:

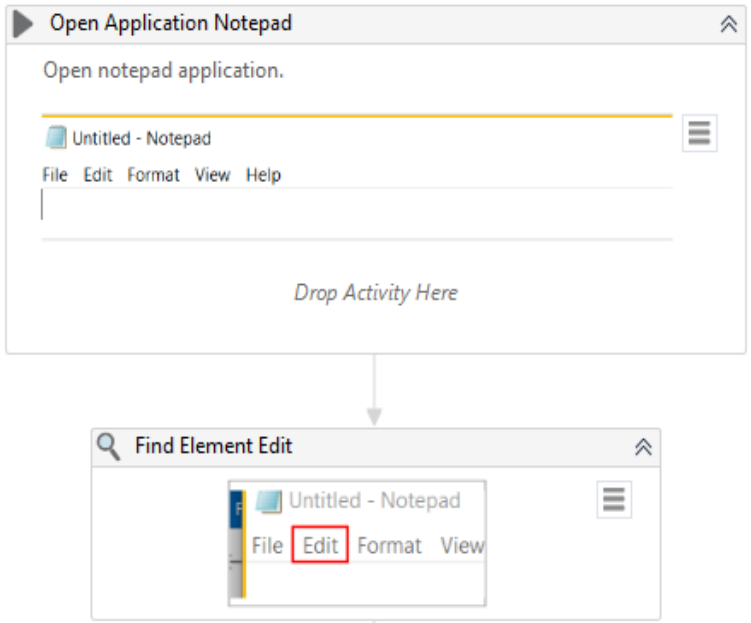


Find Element

The activities related to finding an element are:

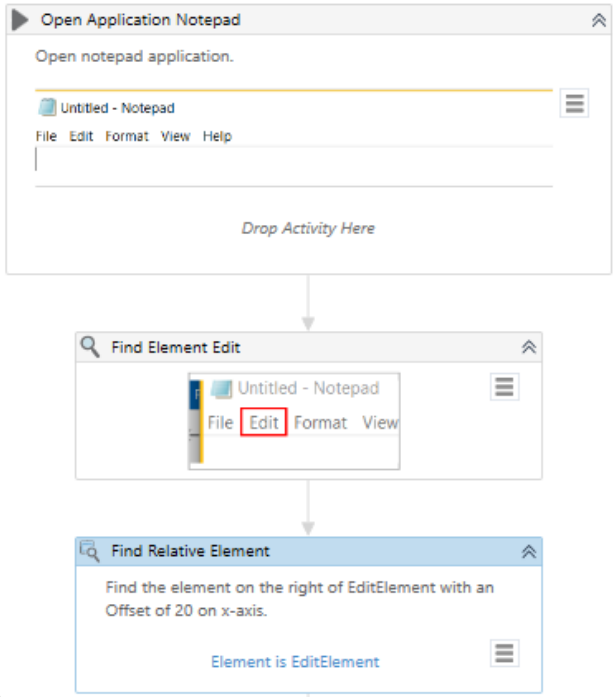
Find Element

- Waits for the specified UI element to appear on the screen
- Returns a UiElement variable
- Throws an error if the element is not found



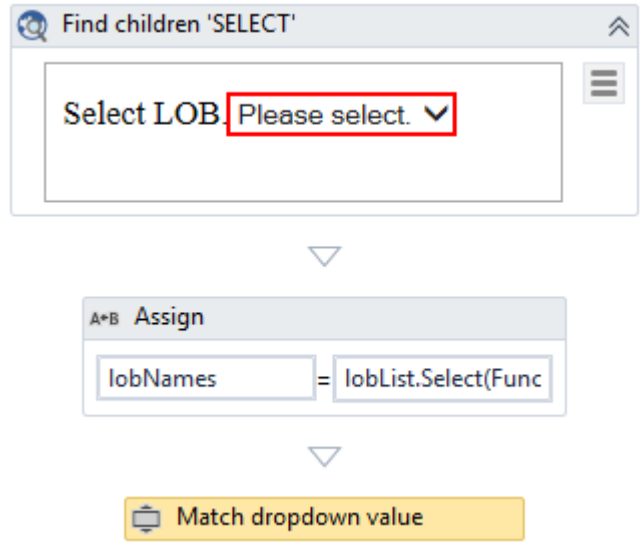
Find Relative Element

- Searches for a UI element by using a position relative to a fixed element
- Used when a reliable selector is not available



Find Children

- Retrieves a collection of children UI elements
- Output field supports only IEnumerable<UiElement> variables

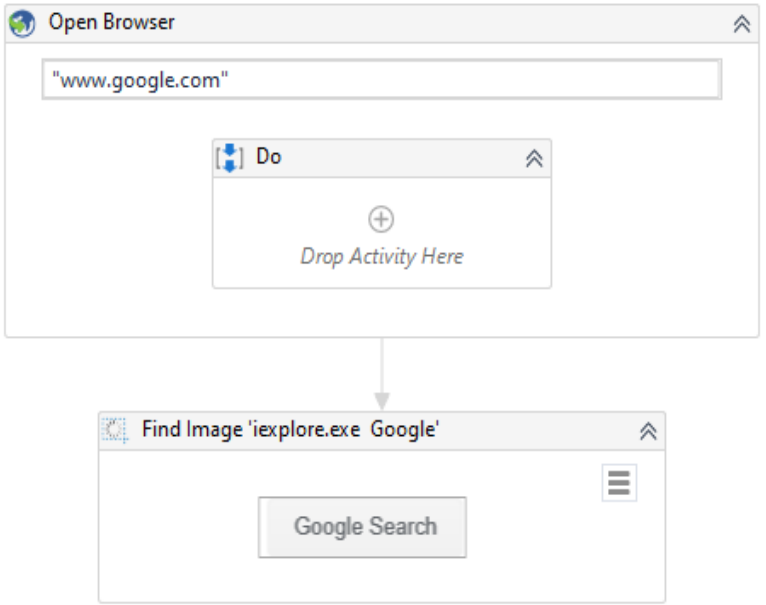


Find Image

The activities related to finding an image are:

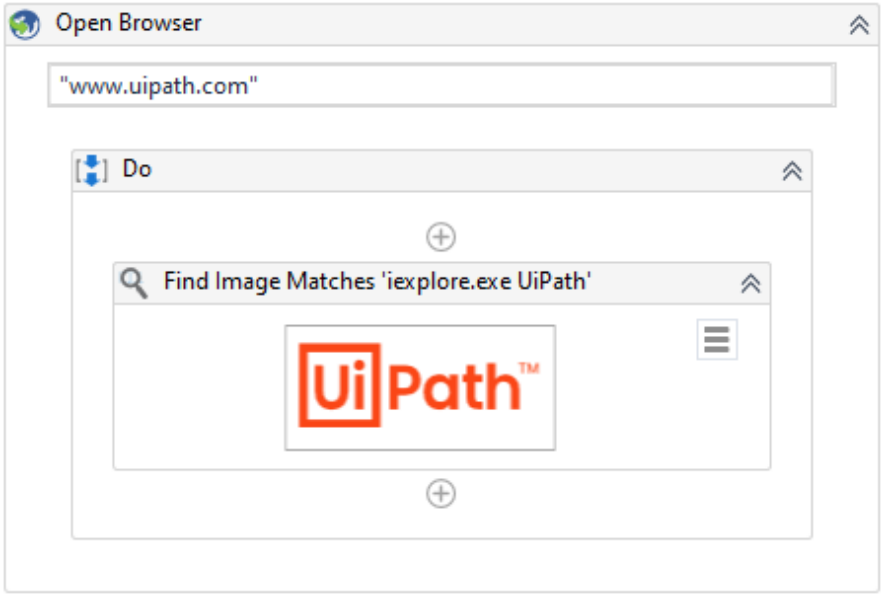
Find Image

- Waits for an image
- Returns a UI element



Find Image Matches

- Searches for matches for a specific image in a target UI element
- Returns a collection of UI elements

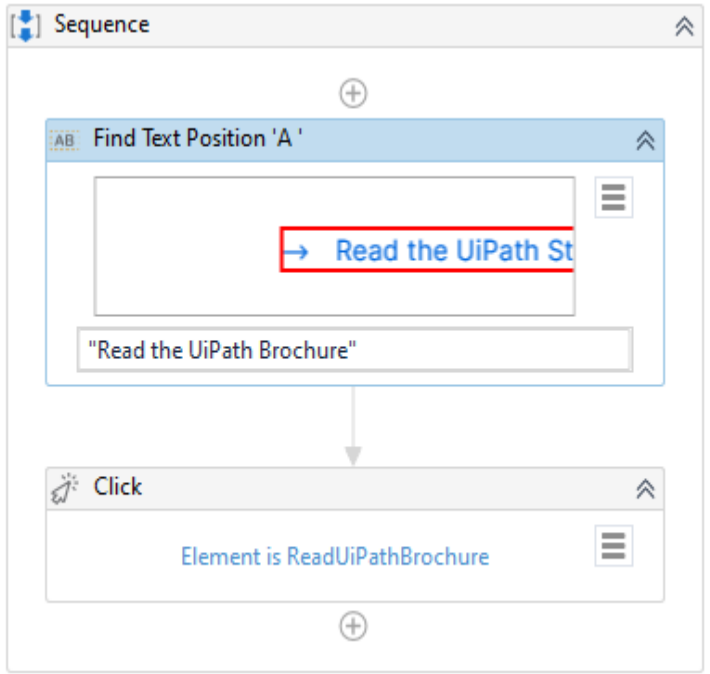


Find Text

The activities related to finding text are:

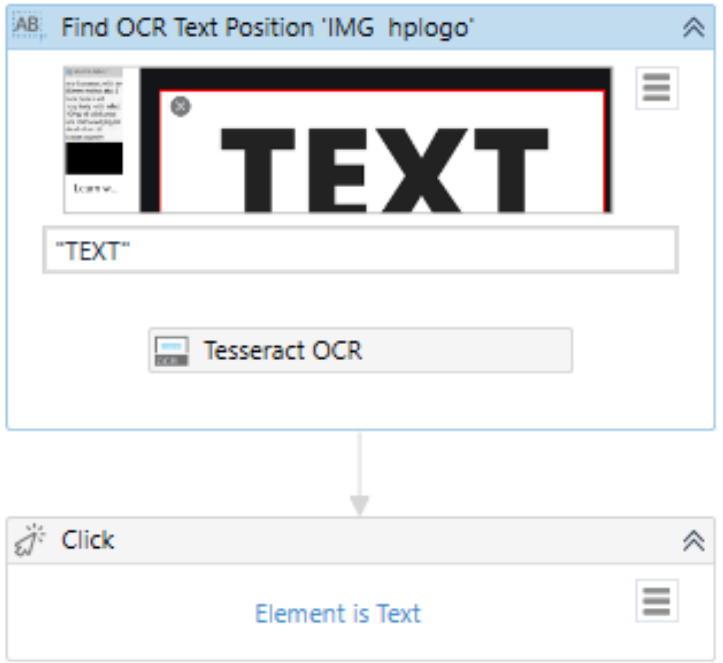
Find Text Position

- Searches for a given string in a UI element
- Returns a UI element



Find OCR Text Position

- Searches for a given string in a UI element or image
- Returns a UI element

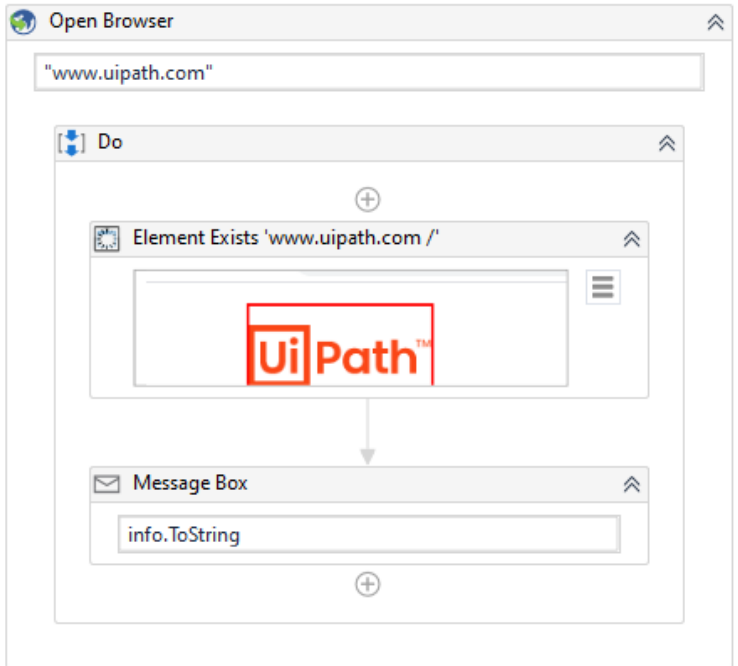


Element Exists

The activities related to an element under exists category are:

Element Exists

- Verifies if a UI element exists
- Returns True/False



CV Element Exists

- Searches for a specified UI element on the screen in the foreground using UiPath Computer Vision neural network
- Returns True/False

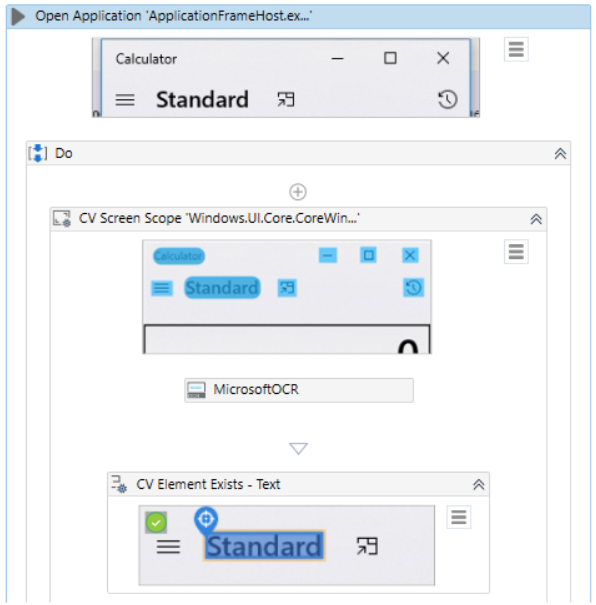
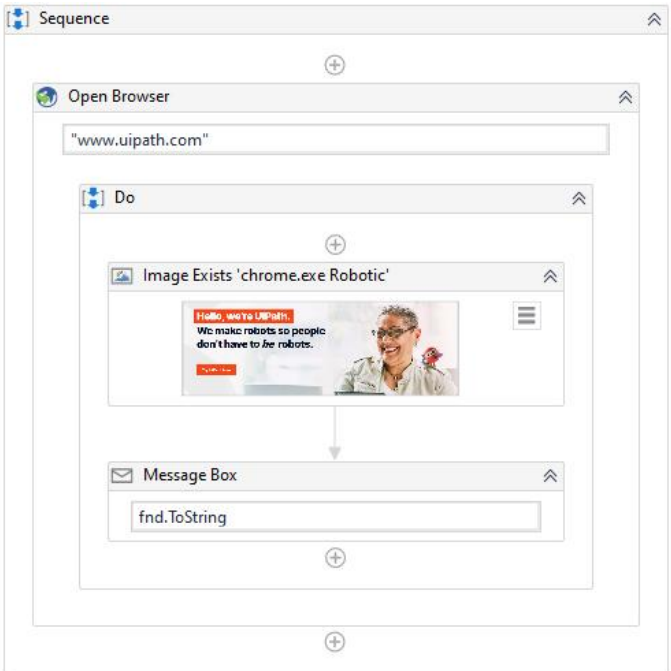


Image Exists

The activity related to an image under exists category is:

Image Exists

- Checks if an image is found within the specified UI element
- Returns True/False

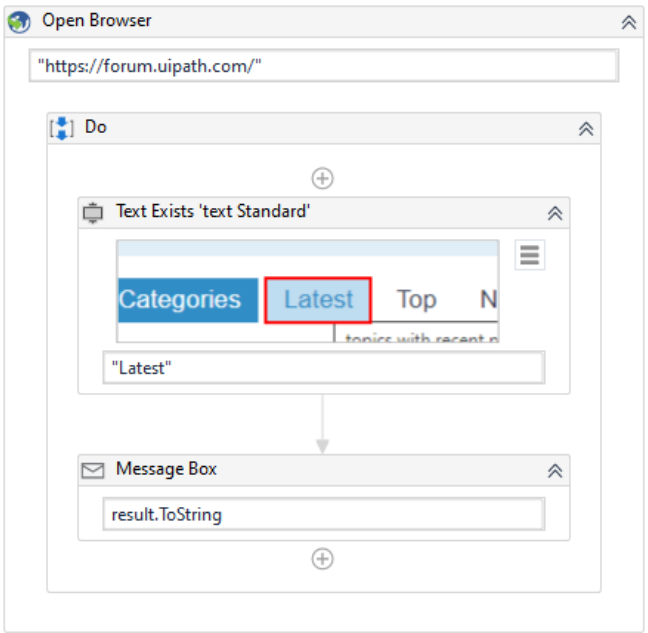


Text Exists

The activities related to text under exists category are:

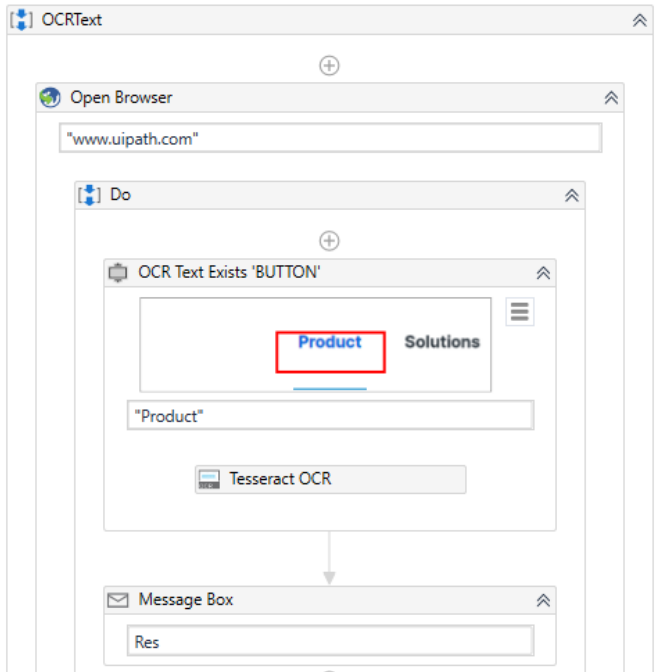
Text Exists

- Verifies if a text is found in a given UI element
- Returns True/False



OCR Text Exists

- Verifies if a text is found in a given UI element or image using OCR technology
- Returns True/False

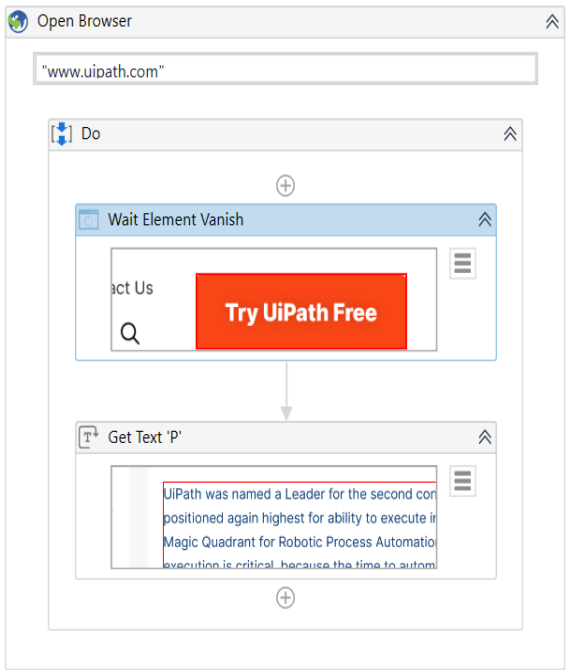


Wait synchronization activities

The activity related to an element under wait category is:

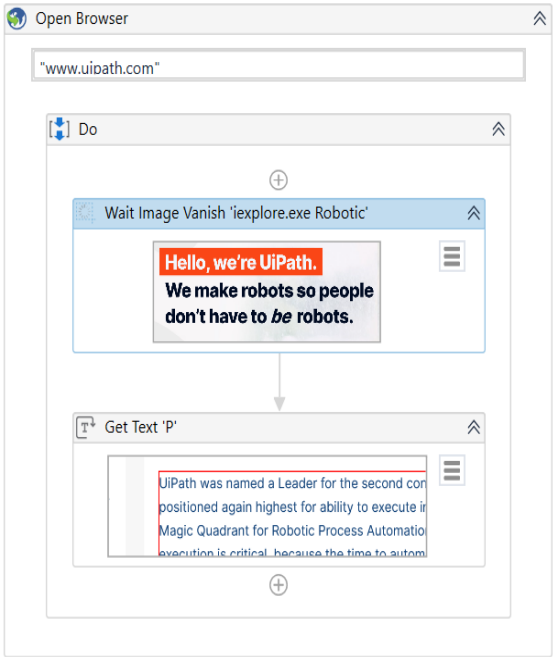
Wait Element Vanish

- Waits for the specified UI element to disappear from the screen



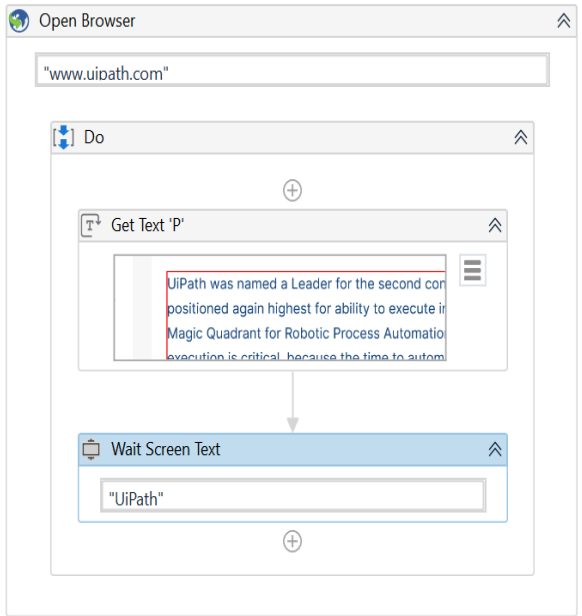
Wait Image Vanish

- Waits for an image to disappear from a UI element



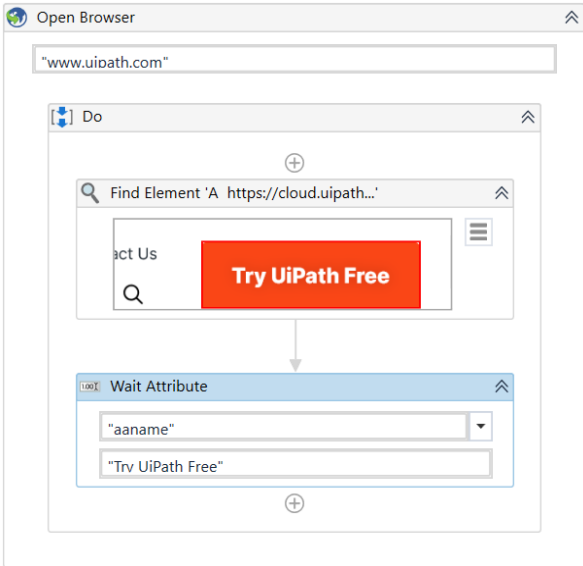
Wait Screen Text

- Waits for a specified amount of time for a string to appear



Wait Attribute

- Waits for the value of a specified UI element attribute to be equal to a string

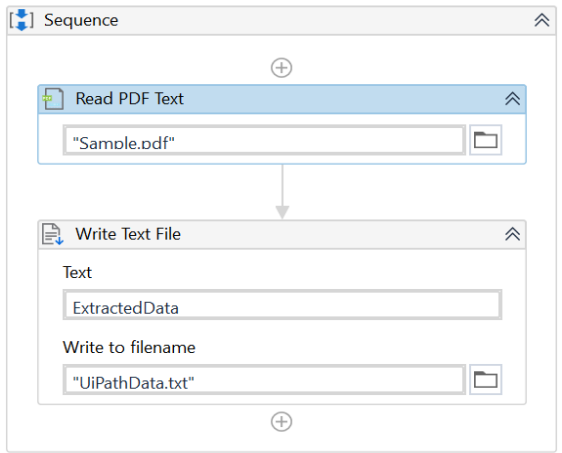


PDF Activities

PDF extraction is the process of extracting the raw data from PDF documents, which can contain text and images. There are two activities for extracting text from PDFs:

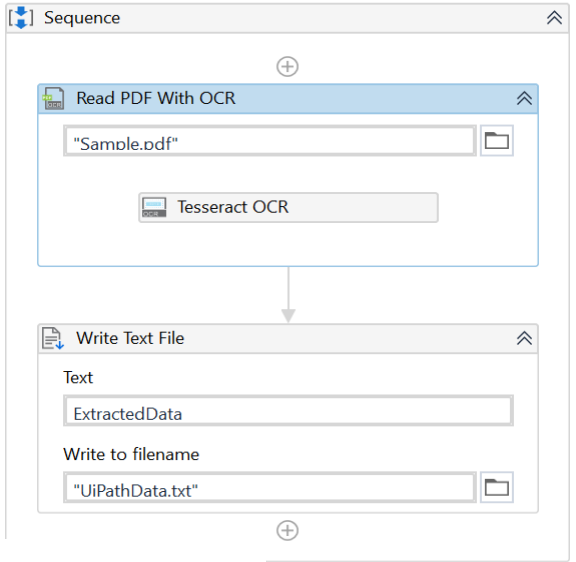
Read PDF Text

- Reads all characters from a specified PDF file and stores them in a string variable
- Extracts text from a Native PDF
- Runs in background



Read PDF with OCR

- Reads all characters from a specified PDF file and stores it in a string variable by using OCR technology
- Extracts text from a Scanned PDF
- Runs in background



- Note: requires the UiPath.PDF.Activities package

UiPath.PDF.Activities by UiPath
IMPORTANT: This package version requires Studio v2018.4.4 / v2019.2 or above. Installing it on versions lower than the recommended ones may cause unexpected changes in your workflows.
PDF/XPS Activities Pack.

Other PDF Activities

Some other activities related to PDFs in UiPath are:

Get PDF Page Count
- Provides the total number of pages in a PDF file

Extract PDF Page Range
- Extracts text from a specified range of pages from a PDF document

Export PDF Page As Image
- Creates an image from a page in a specified PDF file

Join PDF Files
- Joins multiple PDF files stored in an array of strings into a single PDF file

Extract Images From PDF
- Extracts images from a specified PDF file and saves them in a folder

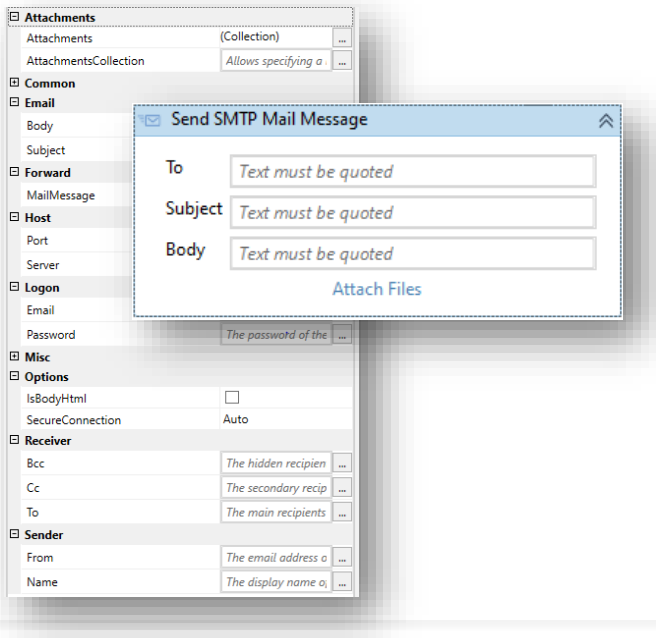
Manage PDF Password
- Manages the password of a specified PDF file if the current password is known

Email Activities – Most Common General Properties

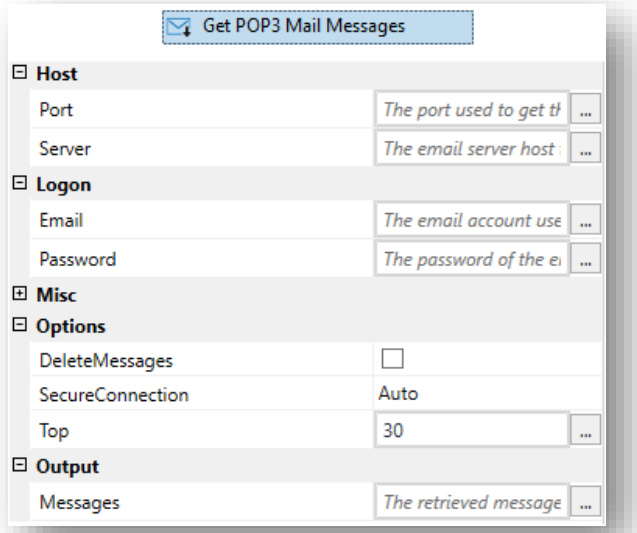
Email (POP3, IMAP, SMTP)
The email account used to get/send mail messages
Password (POP3, IMAP, SMTP)
The password for the email account used to get/send mail messages
Server (POP3, IMAP, SMTP)
The email server host that is to be used
Port (POP3, IMAP, SMTP)
The port used to get/send mail messages
Account (OUTLOOK)
The account used to access messages that are to be retrieved
MailFolder (IMAP, OUTLOOK)
The mail folder from which the messages are to be retrieved
Name (SMTP)
The display name of the sender
From (SMTP)
The email address of the sender
SecureConnection (SMTP, POP3, IMAP)
Specifies the SSL and/or TLS encryption to be used for the connection
To (SMTP, OUTLOOK)
The main recipients of the email message
Cc (SMTP, OUTLOOK)
The secondary recipients of the email message
Bcc (SMTP, OUTLOOK)
The hidden recipients of the email message
Subject (SMTP, OUTLOOK)
The subject of the email message
Body (SMTP, OUTLOOK)
The body of the email message
MailMessage (SMTP, OUTLOOK)
The message to be forwarded. This field only supports MailMessage objects
DeleteMessages (POP3, IMAP)
Specified if the read messages should be marked for delete
Messages (POP3, IMAP, OUTLOOK)
The retrieve messages as a collection of MailMessage objects
Top (POP3, IMAP, OUTLOOK)
The number of messages to be retrieved starting from the top of the list
MarkAsRead (POP3, IMAP, OUTLOOK)
Specified whether to mark retrieved messages as read
Filter (OUTLOOK)
A string used as a filter for the messages to be retrieved. Accepts JET queries and DASL queries

Email Activities

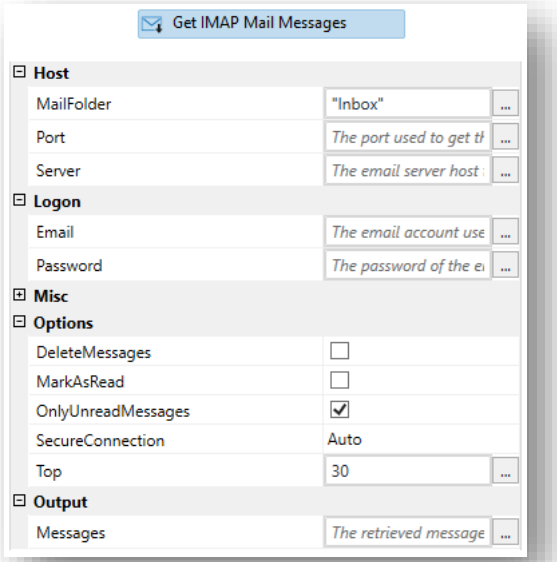
SMTP
used for sending out emails



POP3
used for retrieving emails



IMAP
used for retrieving emails

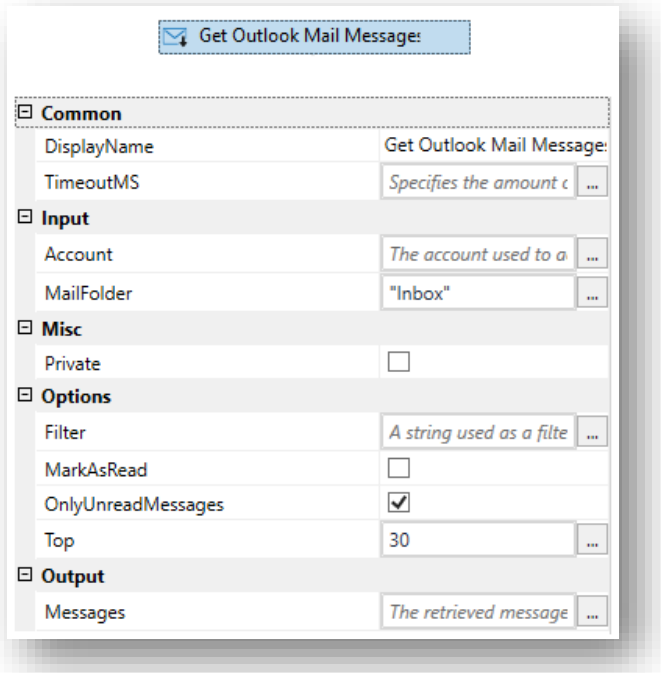


- Note: requires the UiPath.Mail.Activities package

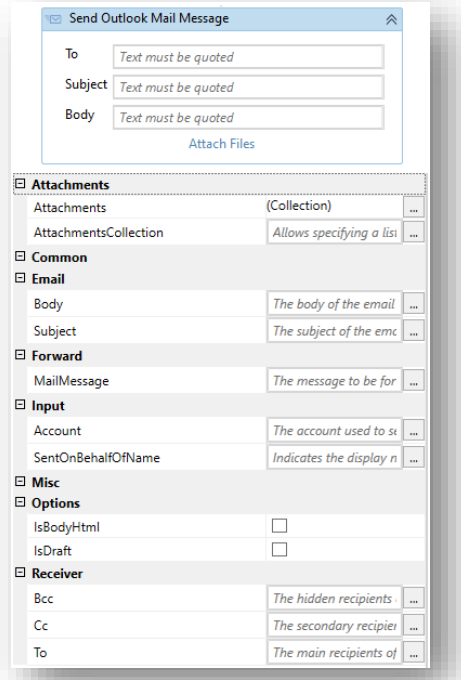
UiPath.Mail.Activities by UiPath
Retrieve and send mail using POP3, IMAP, SMTP and Exchange protocols

Email Activities - Outlook

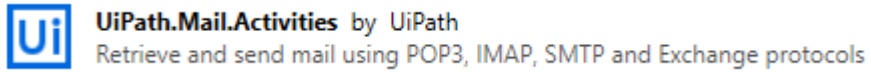
Get Outlook Mail Messages



Send Outlook Mail Message

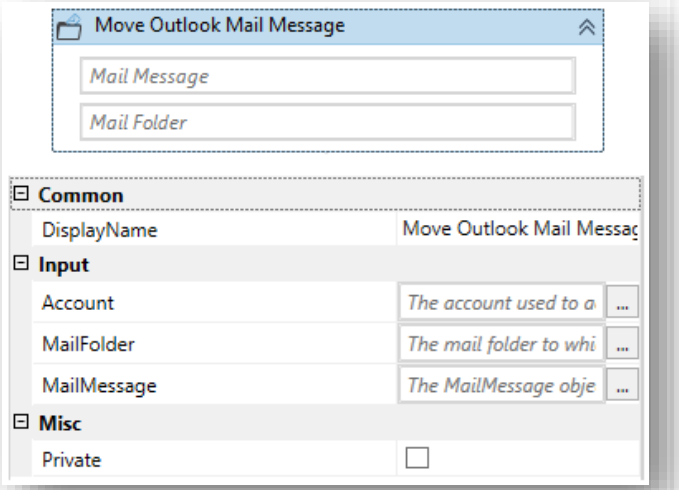


- Note: requires the UiPath.Mail.Activities package

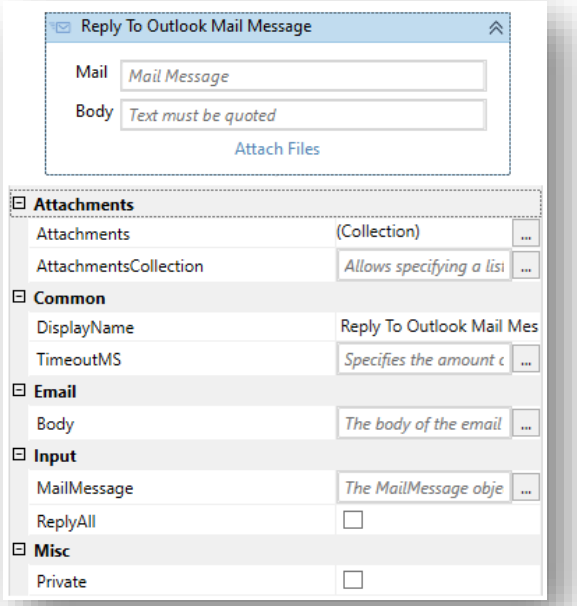


Email Activities - Outlook

Move Outlook Mail Message



Reply To Outlook Mail Message



- Note: requires the UiPath.Mail.Activities package

UiPath.Mail.Activities by UiPath
Retrieve and send mail using POP3, IMAP, SMTP and Exchange protocols

Errors and Exceptions

Errors are events that hamper the regular execution of the program. Based on their source, there are different types of errors:

Errors

Errors are events that hamper the regular execution of the program. Based on their source, there are different types of errors:

- Syntax errors
- User errors
- Programming errors (bugs)

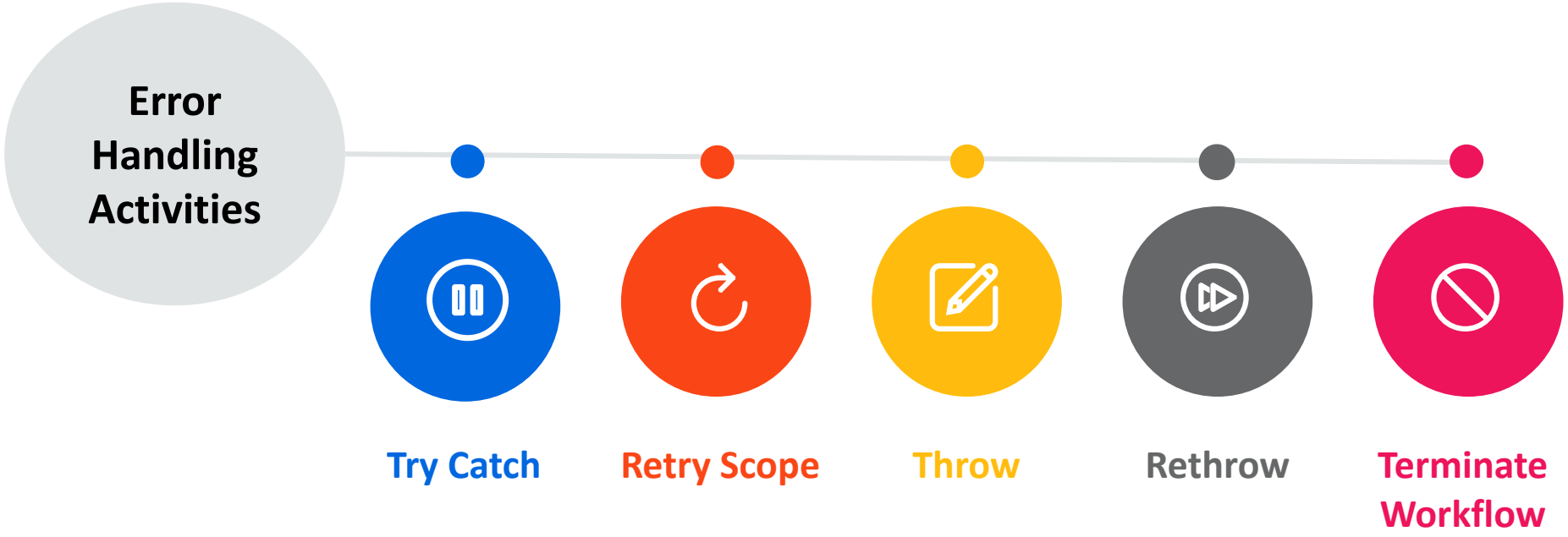
Exceptions

Exceptions are a subset of errors that are recognized (caught) by the program, categorized and handled. The two types of exceptions are:

- Application (System) Exception
- Business Exception

Error Handling Activities

Error handling is the mechanism for identifying and addressing the errors in a program. Some of the important error handling activities are:

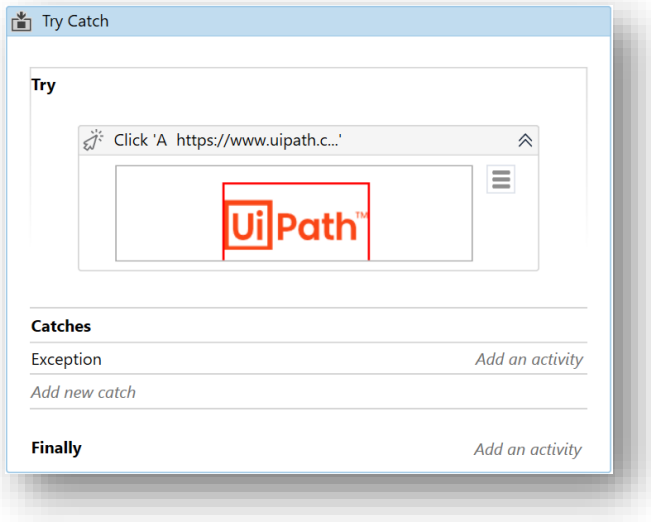


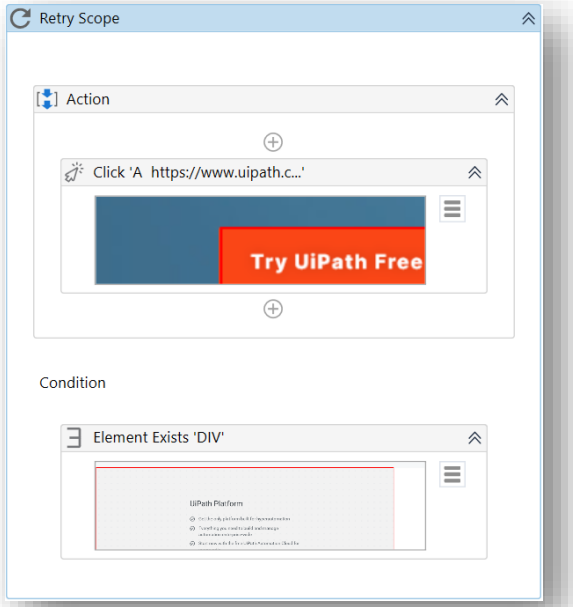
Error Handling Activities

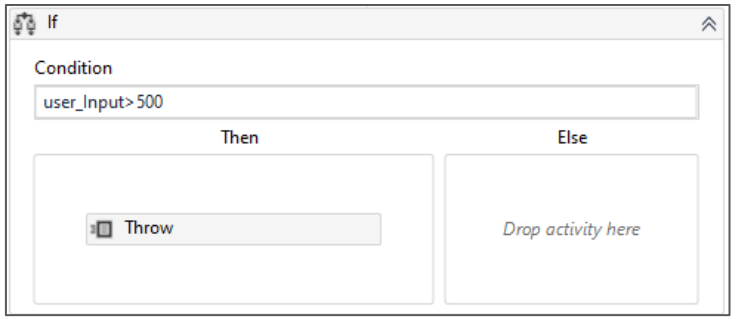
Try Catch
Catches a specified exception type in a sequence or activity, and either displays an error notification or dismisses it and continues the execution.

Retry Scope
Retries the contained activities as long as the condition is not met, or an error is thrown.

Throw
Throws a user-defined exception

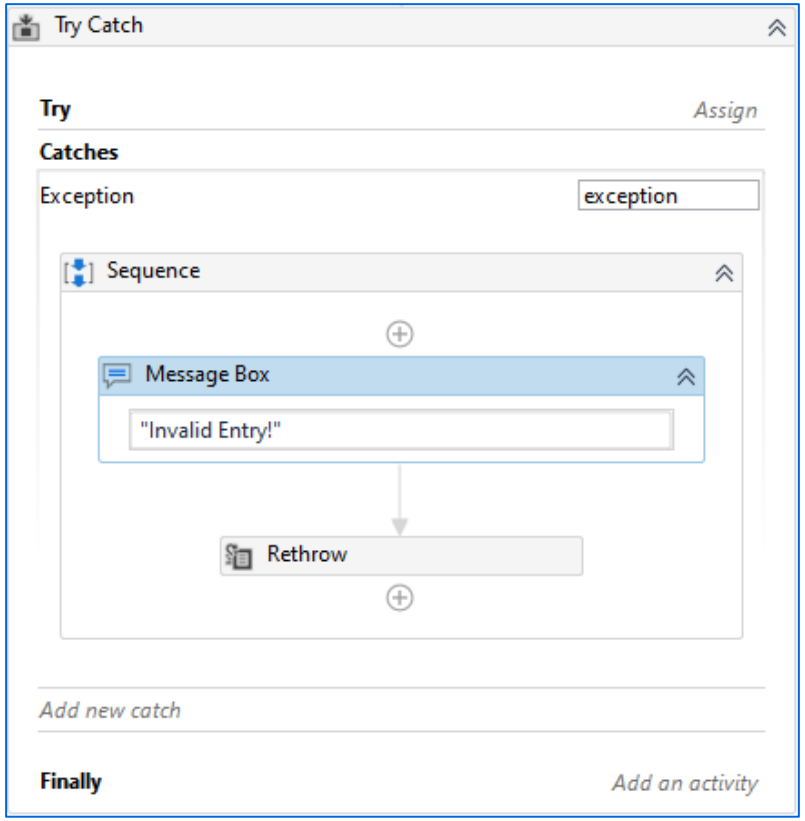




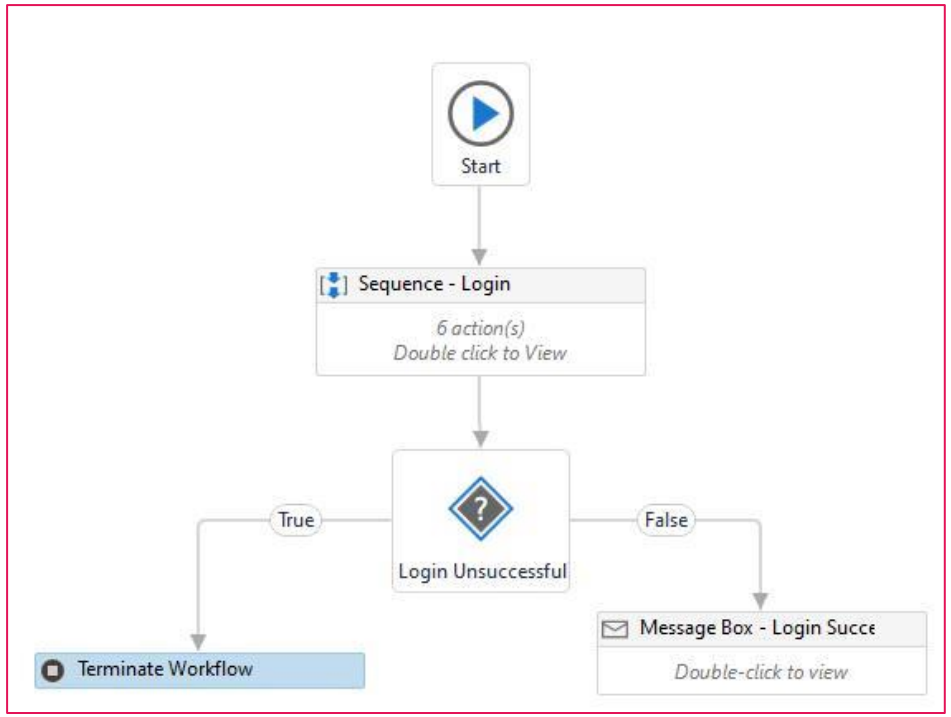


Error Handling Activities

Rethrow
Takes an existing exception that has been encountered and regenerates it at a higher level



Terminate Workflow
Terminates the workflow when the task encounters an error



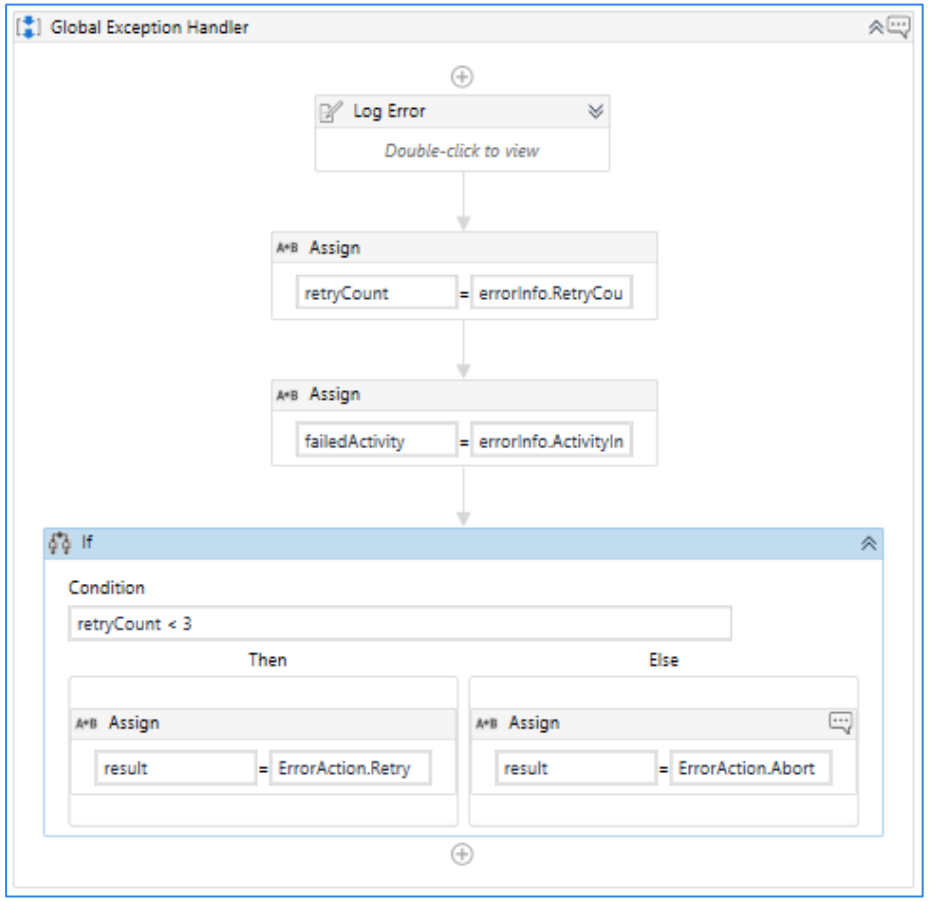
Global Exception Handler

The **Global Exception Handler** is a type of workflow designed to determine the behavior when encountering an execution error at the project level.

- Only one Global Exception Handler can be set per automation project.

- It is used in conjunction with Try Catch, and only uncaught exceptions will reach the Exception Handler.

- It identifies exceptions that are less likely to happen in a certain part of a project.



Common Exceptions

Object reference not set to an instance of an object.

Thrown when a variable has not been initialized (has no value). To avoid this, make sure a variable has a value before using its methods.

Index was outside the bounds of the array. Index out of range.

Thrown in an attempt to access array or collection elements with an index that is outside its capacity. Most common situation: getting an element of a list without checking if the list contains items

Cannot find the UI Element corresponding to this selector.

Occurs when the selector used to identify the control does not match any control on the screen. To debug, when this exception is thrown, you need to inspect the target application and observe if the control exists.

Image was not found in the provided timeout.

Occurs when the expected image is not identified on screen either because it is not visible in foreground or the image is slightly different due to environment settings (resolution, theme) or controls are in different state.

Text was not found.

Occurs in Text based automation activities. Text activities currently do not take into account the TimeoutMS property when searching for the text, so the text needs to be on the screen when the activity is reached.

Click generic error. Cannot use UI_CONTROL_API on this UI node. Please use UI_HARDWARE_EVENTS method.

Occurs when a click could not be executed. The most common issue is using SimulateClick or SendWindowMessages options in an application window that does not support them.

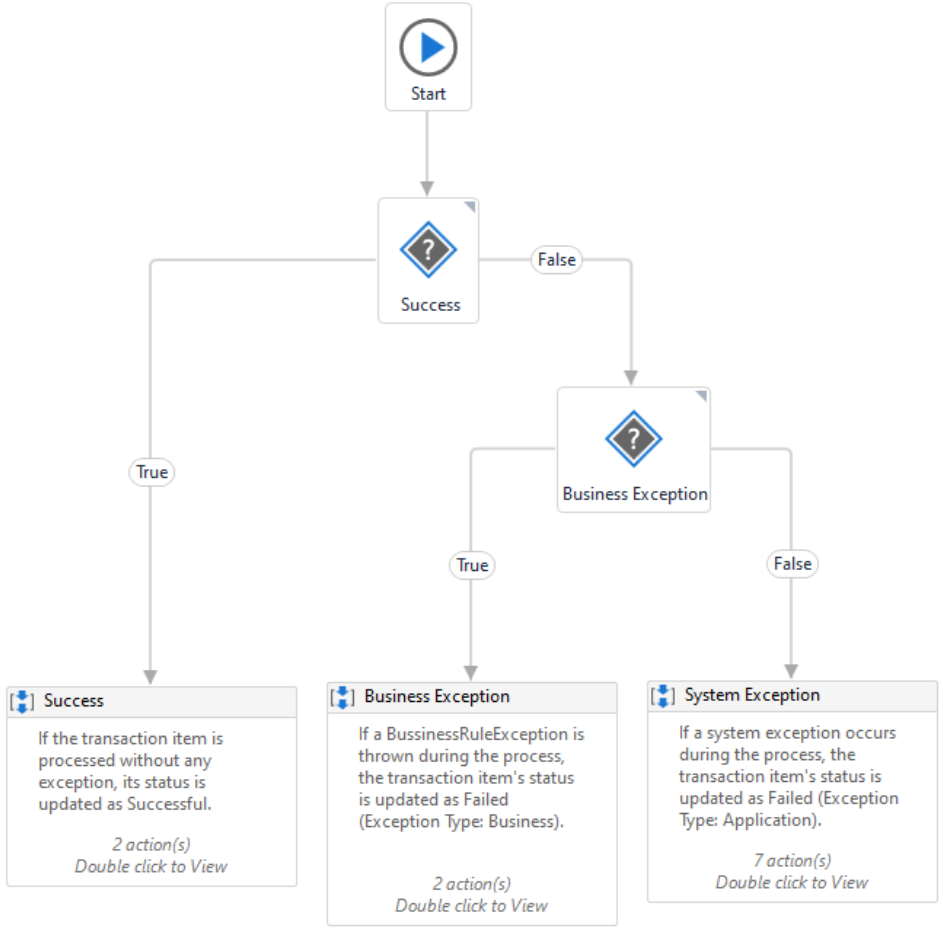
System Rule Exception vs Business Rule Exception

A **System Exception** (or Application Exception) describes an error rooted in a technical issue, such as an application that is not responding.

These kinds of issues have a chance of being solved simply by retrying the transaction, as the application can unfreeze.

A **Business Exception** describes an error rooted in the fact that certain data which the automation project depends on is incomplete or missing.

Retrying the transaction does not yield any chance of solving the issue, and there are other better courses of action, such as notifying the human user of this error.



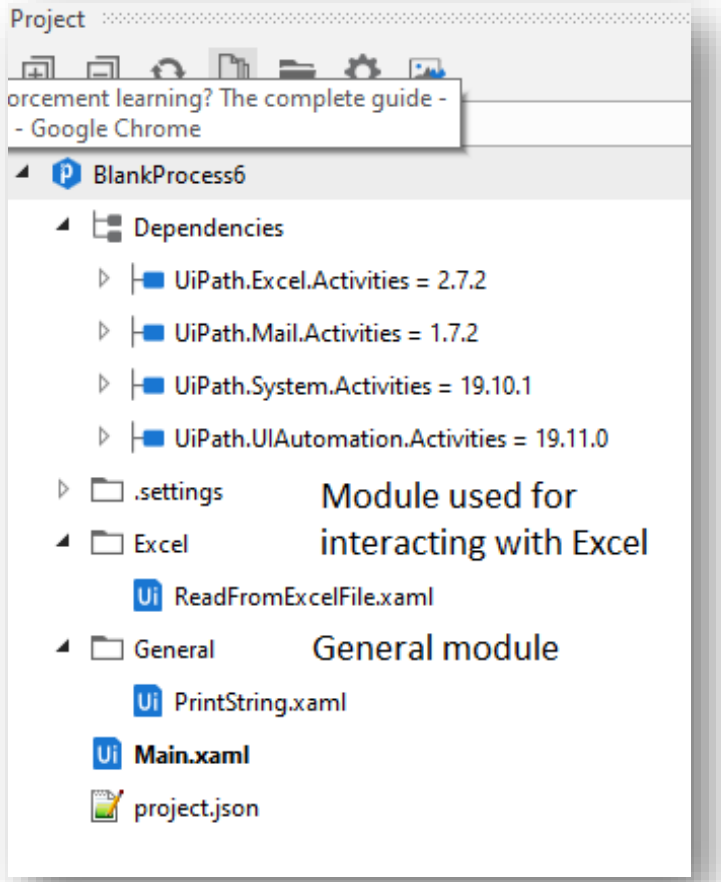
Example from the SetTransactionStatus.xaml from ReFramework

Project Organization

- The project organization is very important.
- A project must be clean, organized, modularized.
- The project must be structured into modules, and each module must be related to the application used, way of usage (UI interaction or logic). Some common folders (for reusable workflows) can be added.
- Adjust the Project settings helps you to set different project settings for the from the beginning.

Project organization

- Makes a project clean, structured, and easy to implement
- Structures a project into simple modules
- Helps in reusing workflows across projects



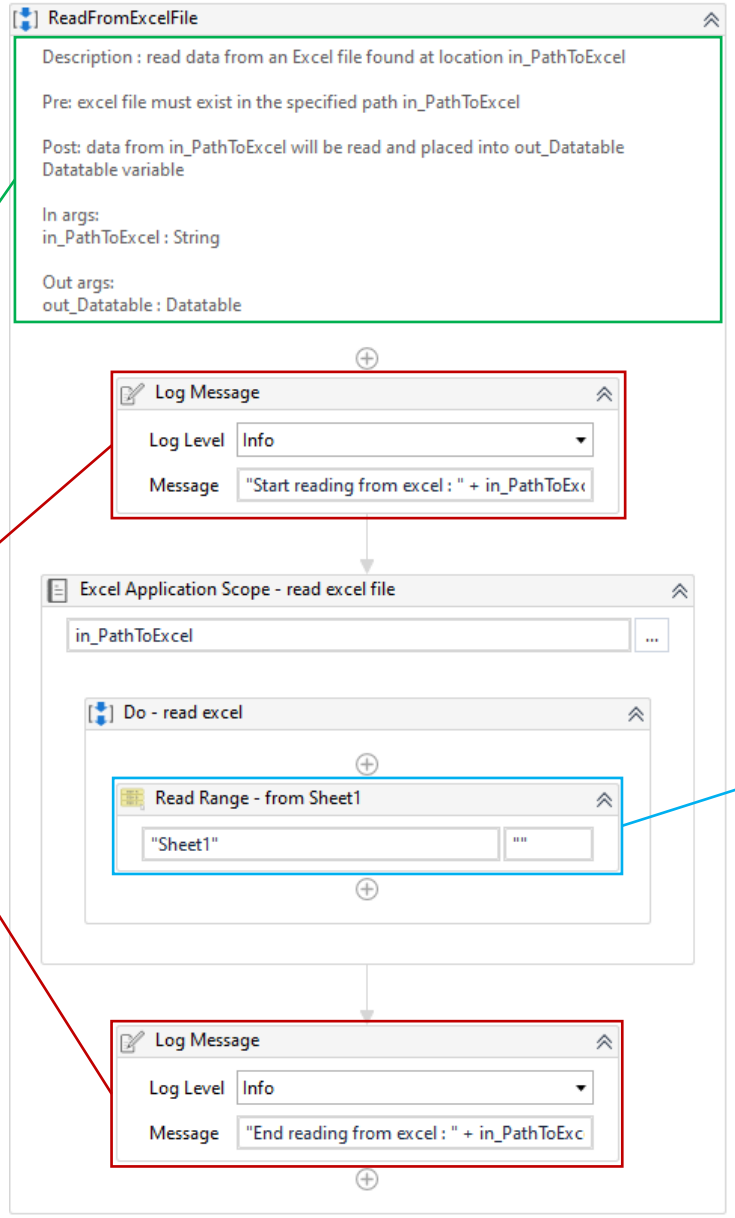
Project Organization – Workflow Organization

Add an annotation at the beginning of the workflow. Annotations should be added whenever a piece of code is complicated or needs further explaining

Add log messages at the beginning and at the end of each workflow, but also every time something changes

Name	Direction	Argument type	Default value
in_PathToExcel	In	String	*Enter a VB expression*
out_Datatable	Out	WorkbookApplicat	*Default value not supporte*

Arguments and variables must have a suggestive name, the direction must be appropriate to the way they are used



Rename with an appropriate name each activity used

Best Practices

A minimal set of best practices:

- Break the whole process in simple workflows.
- Pick an appropriate layout for each workflow (sequence/flowchart/state machine).
- Give meaningful names to all components (Workflows, activities, arguments and variables).
- Use the naming convention for arguments: use in_ / out_ / io_ prefix for identifying the argument type and the rest should be Camel Case.
- Use the Camel Case naming convention for variables and for workflows (e.g. *FirstName*, *GetTransactionData.xaml*).
- Keep it clean: close applications, windows, web pages.
- Keep environment settings in a config file.
- Collaborate working on separate files.
- Reuse workflows across projects.
- Develop and test pieces independently.
- Use REFramework or Enhanced REFramework for complex processes.

Best Practices

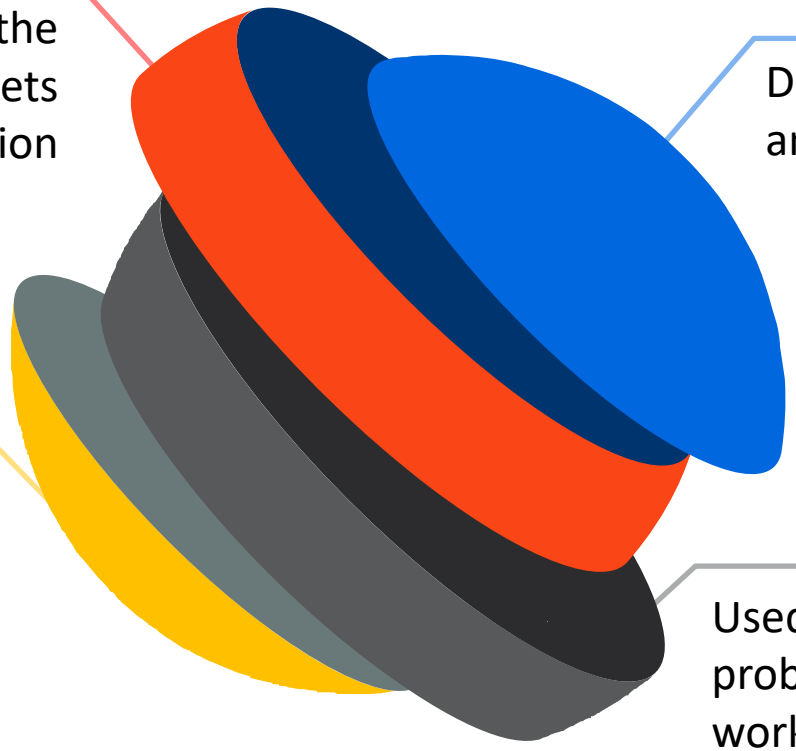
- Use a config file or similar for keeping the settings.
- Use annotations for every activity, explaining the purpose of it.
- Use an annotation in the start of a sequence to describe the process handled in the sequence + prerequisites.
- Queue items should not store sensitive data.
- Do not log sensitive data (e.g.: accounts, passwords, financial transactions etc.).
- No more than 2 nested IF activities, if the logic is more complex, try using a flowchart instead.
- Delete the activities and variables that were used for debug.
- Avoid using multiple nested try catches, otherwise it makes difficult seeing where the error comes from.
- Make sure the exceptions are used accordingly (e.g. BusinessRuleException vs ApplicationException).
- Use partial selectors when several activities are executed in the same window.
- Avoid using selectors with indexes - try using more attributes or nested selectors.
- Avoid using Delay Activities - Use On Element Appear/Element Exists or similar activities.

Automation Debugging

Debugging is the process of identifying and removing the errors which prevent the project from functioning correctly. In Studio, debugging is

Useful for verifying the data that each activity gets during execution

Done at activity, file, and project level

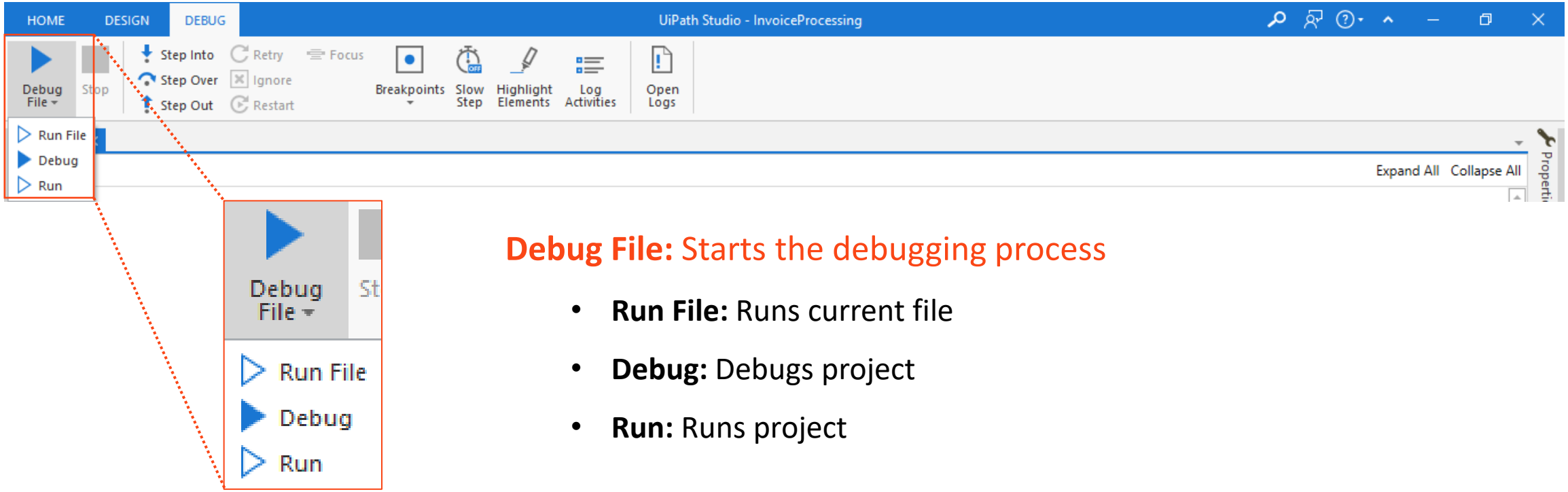


Done using options available in the Debug Ribbon

Used for finding and locating problems easily in complex workflows

Troubleshoot, Debug, and Modify processes

The **Debug tool** in UiPath Studio is a real-time engine that checks for errors while working with the workflow. Whenever an activity has errors, UiPath Studio Process Designer notifies and gives details about the issues encountered.



Debug File: Starts the debugging process

- **Run File:** Runs current file
- **Debug:** Debugs project
- **Run:** Runs project

Debugging Actions

The actions for debugging are:

Step Into
Debugs activities step-by-step

Ignore
Ignores an exception and executes from the next activity t

Slow Step
Debugging at a slower rate, takes a closer look at the activity

Step Over
Debugs the next activity without opening it

Restart
Restarts debugging from the first activity of the project

Highlight Elements
Highlights UI elements during debugging

Step Out
Pauses execution at current container

Break
Pauses the debugging process

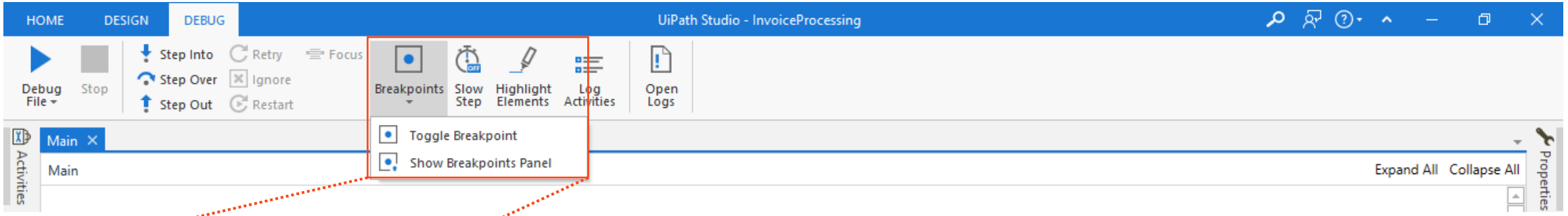
Log Activities
Displays debugged activities as Trace logs in the Output panel

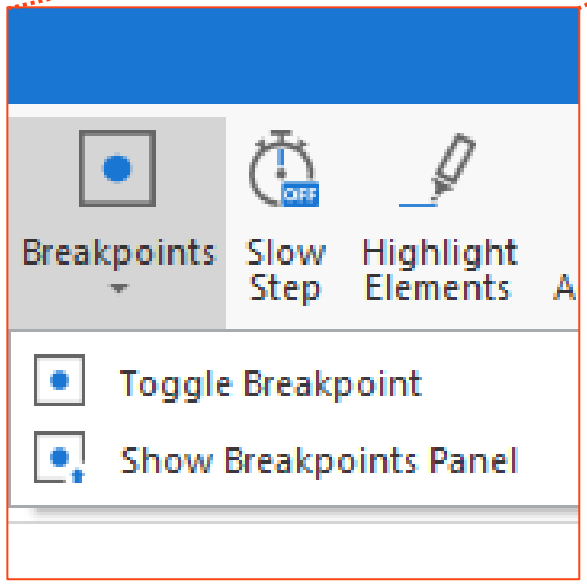
Retry
Re-executes previous activity

Focus
Returns to the activity that caused error and resumes debugging

Open Logs
Opens local folder where the logs are stored

Setting Breakpoints





Breakpoints

- Used when the user wants to pause the debugging process at a specific activity (that causes execution issue)
- The user identifies and corrects the error

Logging

Logging is the process of keeping logs of various events that occur during project execution. There are three types of logs:

Studio Logs

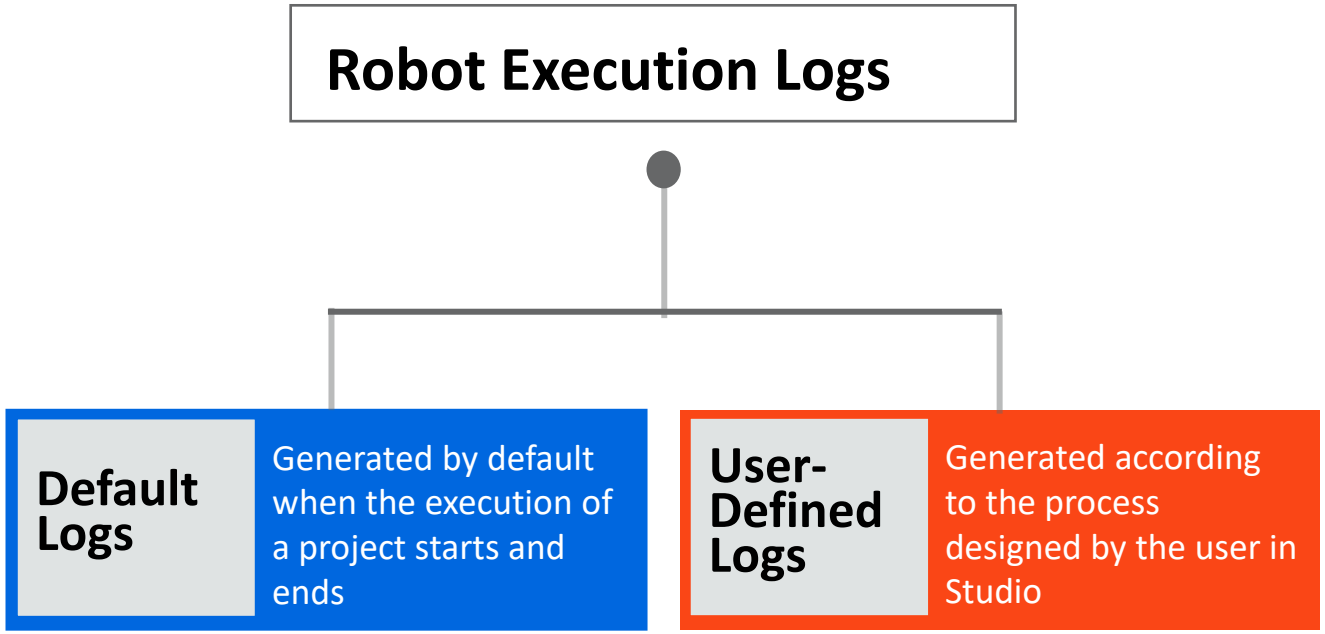
- Diagnostic log messages generated by Studio regarding its behavior

Robot Logs

- Diagnostic log messages providing information related to the Robot and its context

Orchestrator Logs

- Diagnostic log messages generated by Orchestrator regarding its behavior



Logging Levels

There are five levels of logging:

Fatal Level

Indicates critical issues such as a robot may not recover and stop working

Error Level

Indicates errors after which a robot retries for recovery and moves on

Warning Level

Indicates events that may have an adverse impact on a robot's performance

Information Level

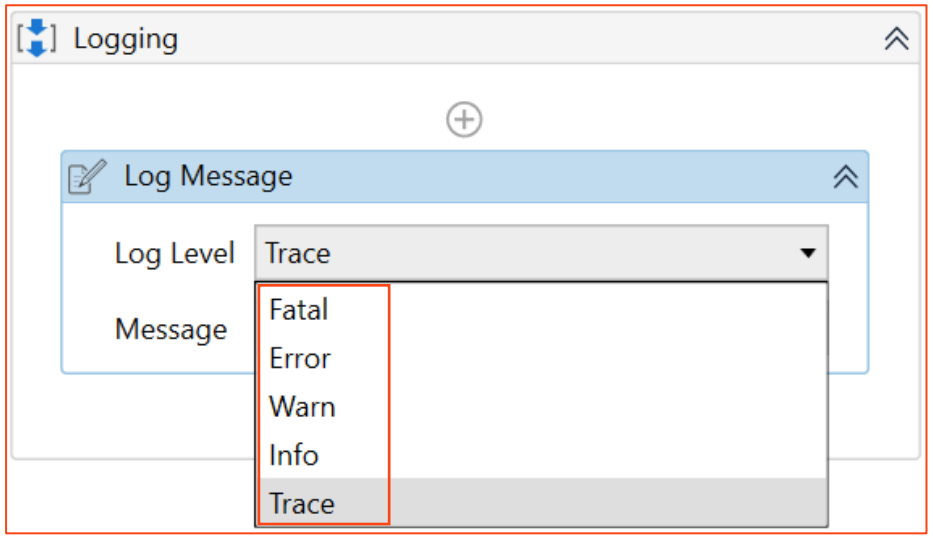
To know the progress of a robot at each stage of execution

Trace Level

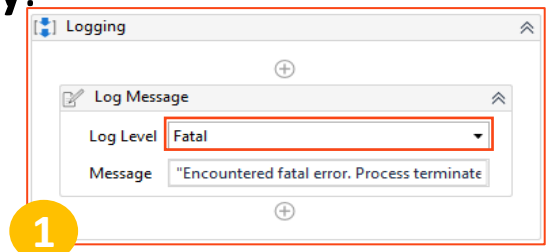
To collect information for developing or debugging

How to do logging?

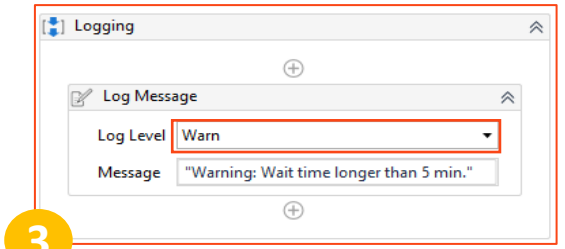
Logs are applied using the **Log Message activity**.



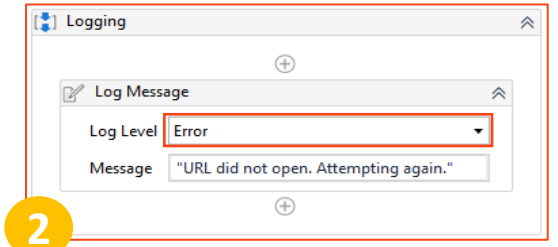
Log Message Activity



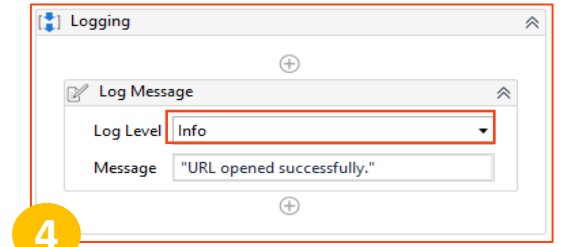
1 **Fatal Log Message**



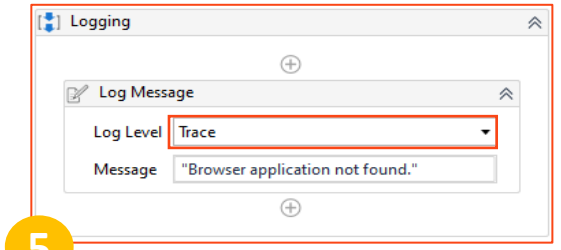
3 **Warn Log Message**



2 **Error Log Message**



4 **Info Log Message**



5 **Trace Log Message**

Logging

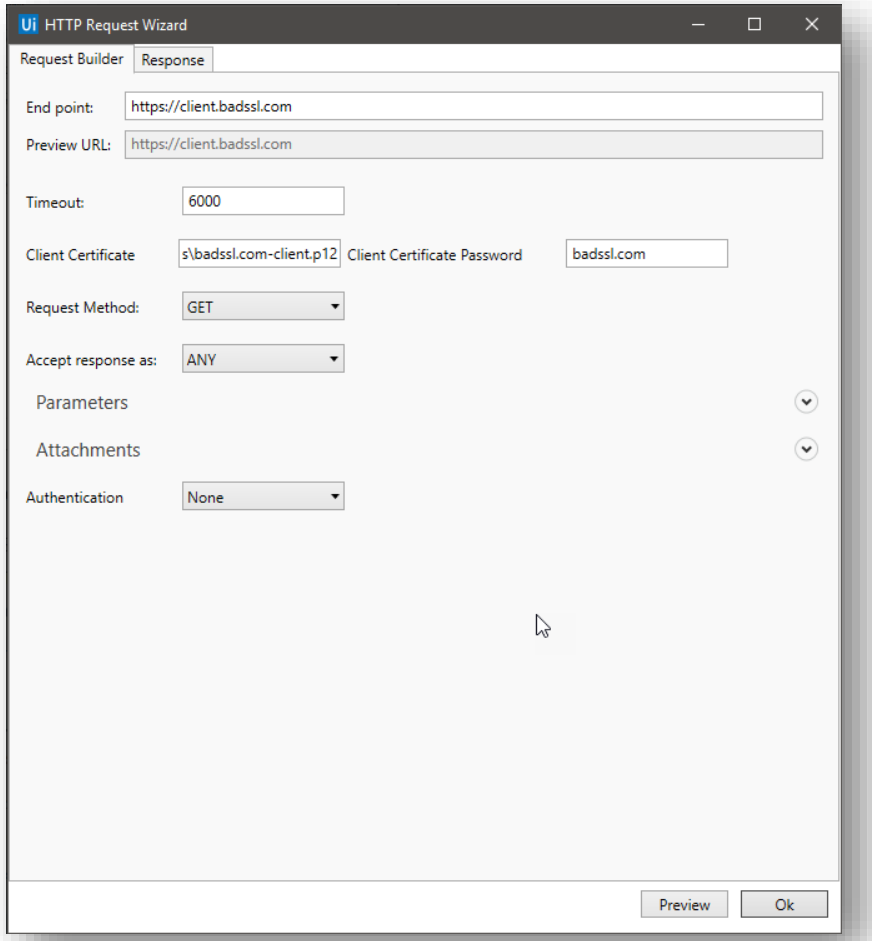
Log message activities should ideally be used:

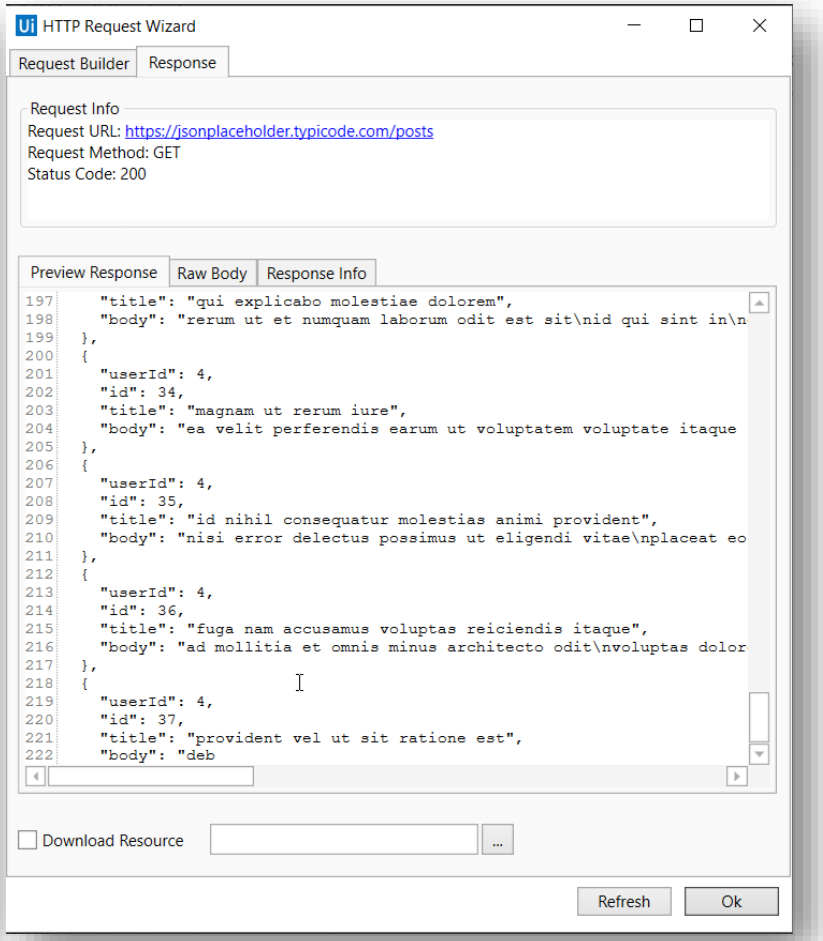
- at the beginning and the end of every workflow (Log level = Information)
- each time an exception is caught in a Catch block (Log level = Error)
- each time a Business Rule Exception is thrown (Log Level = Error)
- when data is read from external sources (for example, log a message at Information level when an Excel file is read)
- in Parallel or Pick activities, log messages on every branch, in order to trace the branch taken (Log Level = Information)
- in If/Flowchart Decision/Switch/Flow Switch activities (however, since processes might have a lot of these activities, we can decrease the Log Level from Information to Trace, in order not to have a lot of these logs in the database)

What are other use cases you can think of?

HTTP Requests

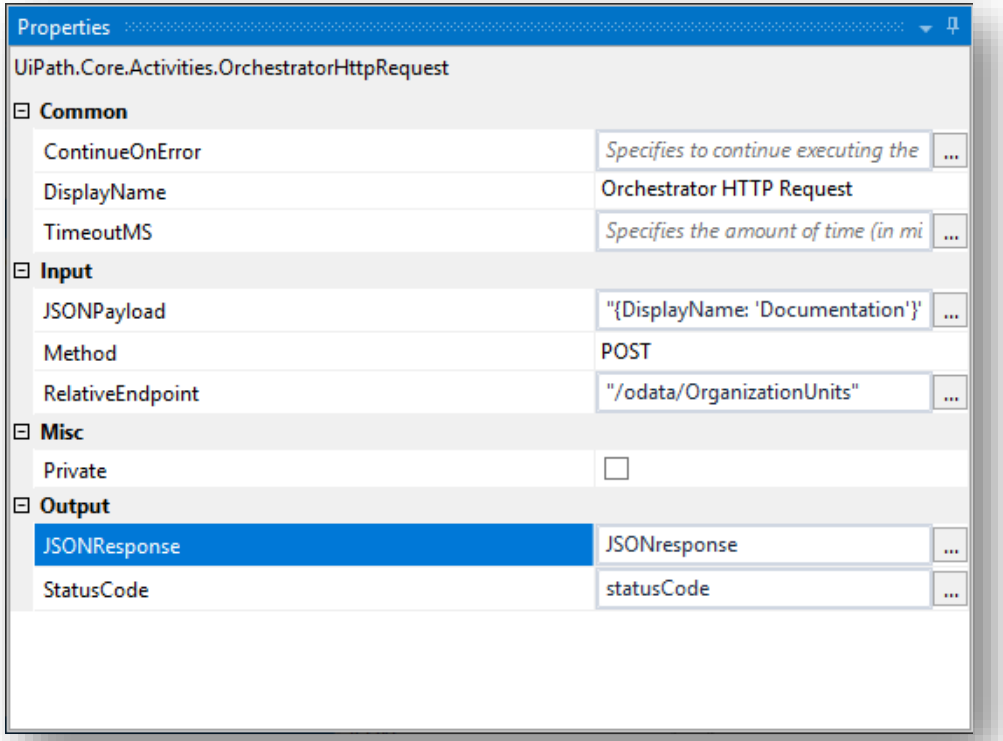
- The **HTTP request activity** enables you to perform HTTP requests to web APIs.
- All the fields present in the Request Builder window are also available in the **Properties** panel. As a result, you can still edit your preferences after closing the wizard.
- The Response tab enables you to preview the request that you want to include in your workflow. It can be accessed by clicking the **Preview** button in the HTTP Request wizard once you fill in all the fields and configure the request.





HTTP Requests

- The **Orchestrator HTTP Request** activity performs HTTP requests to the Orchestrator API by authenticating under the Robot it is executed on.
- You can use the GET, POST, PUT, PATCH and DELETE verbs to extract data or manipulate it, and send specific information through JSON.
- The Orchestrator API Swagger definition can be accessed as follows, depending on your deployment type:
  - **on-premise** - add the following suffix: /swagger/ui/index#/ to your Orchestrator URL. For example, https://myOrchestrator.com/swagger/ui/index#/.
  - **Automation Cloud** - add the account and tenant name, as well as the /swagger/ui/index#/ suffix to the URL. For example, https://cloud.uipath.com/[AccountLogicalName]/[TenantLogicalName]/swagger/ui/index#/.
    Find your Account Logical Name and Tenant Logical Name in the **API Access** page of your Automation Cloud account.

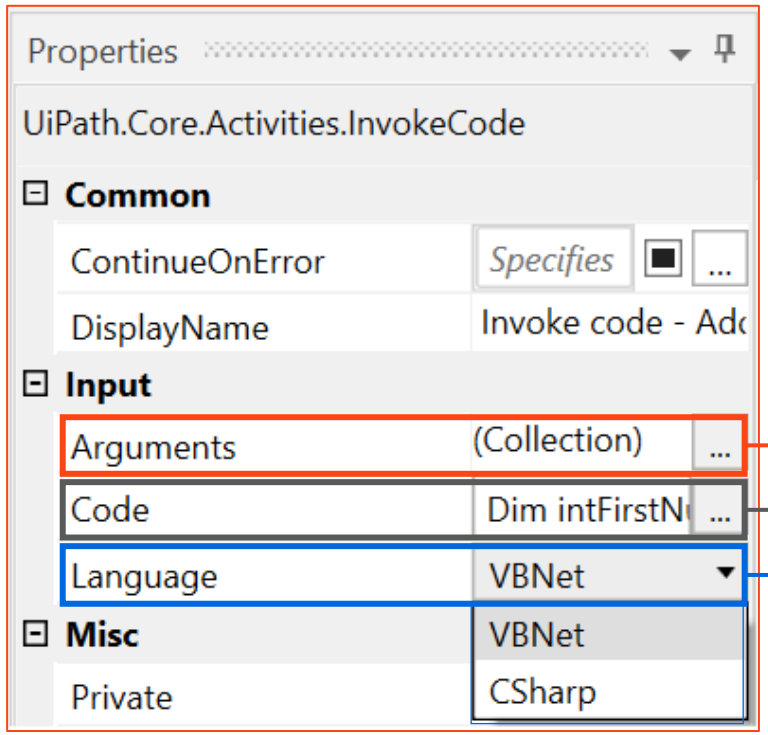


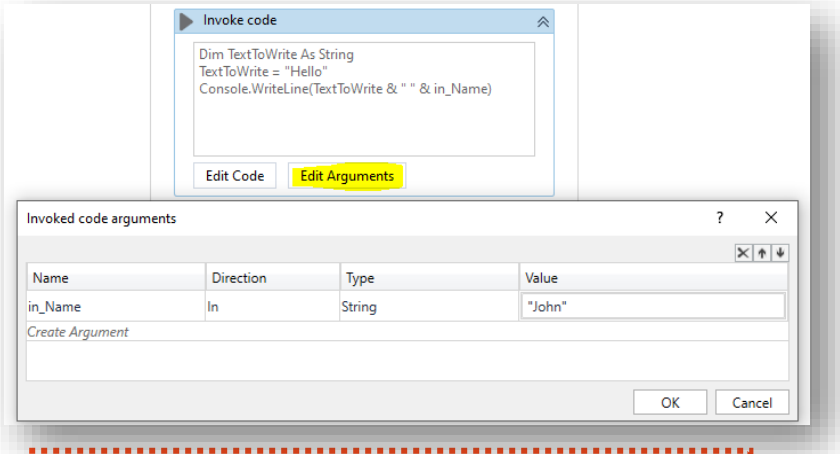
Introduction to Invoke Code Activity

Invoke Code activity can be used to invoke VB.NET or CSharp (C#) codes and write custom codes for some steps in the process.

It helps the users to:

- Simplify data manipulation procedures
- Reduce the number of Assign and Invoke Method activities, by replacing several with a single Invoke Code activity

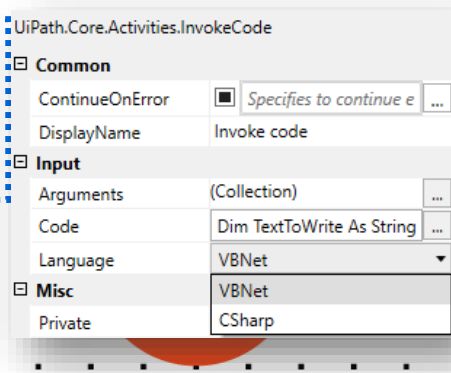




Arguments
Pass the parameters to the invoked code

Code
Pass the codes to be invoked

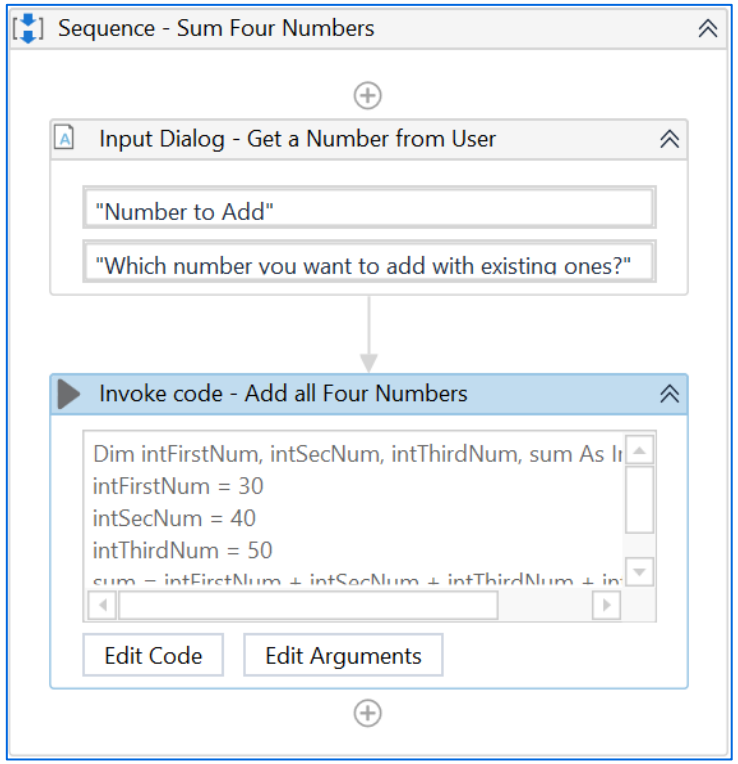
Language
Select the programming language (VB.NET or C#)



Invoke Code Activity with VB.NET and C#

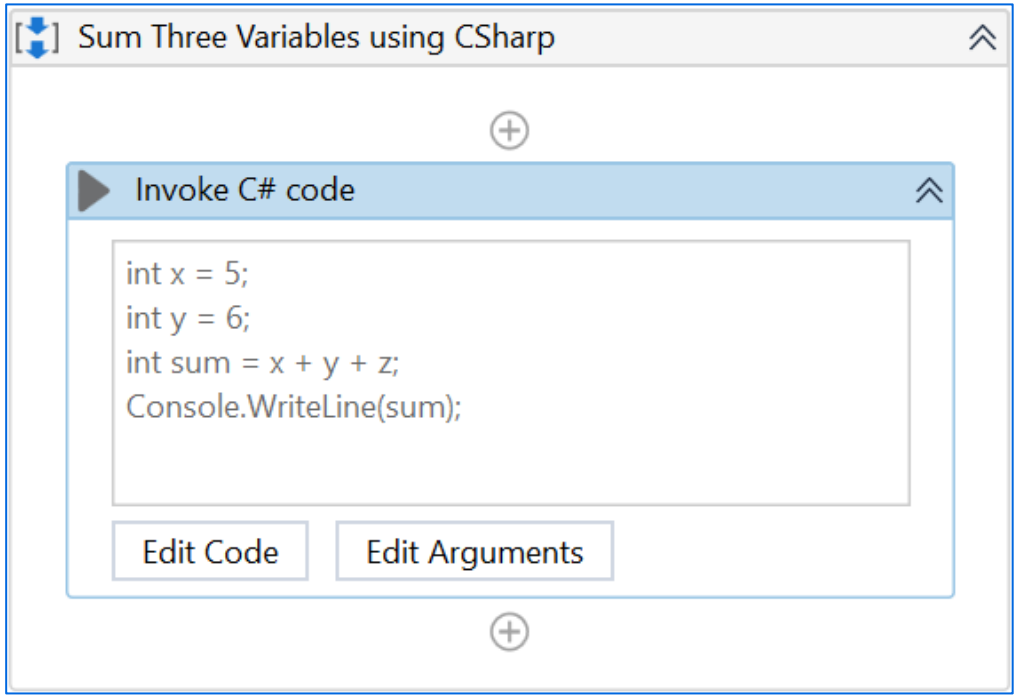
Invoke Code Activity with VB.NET

- VB.NET code is used inside the Invoke Code activity
- Example: A VB.NET code for adding numbers is invoked using the activity



Invoke Code Activity with C#

- C# code is used inside the Invoke Code activity
- Example: A C# code for adding two numbers is invoked using the activity



Introduction to Invoke Method Activity

Invoke Method activity is used to call a method that is outside the standard built-in activities of Studio, such as in a DLL file.

Invoke Method Activity

- Calls a specific method in a class
- Calls a public method of a specified type or object
- Acts on either Target Type or Target Object

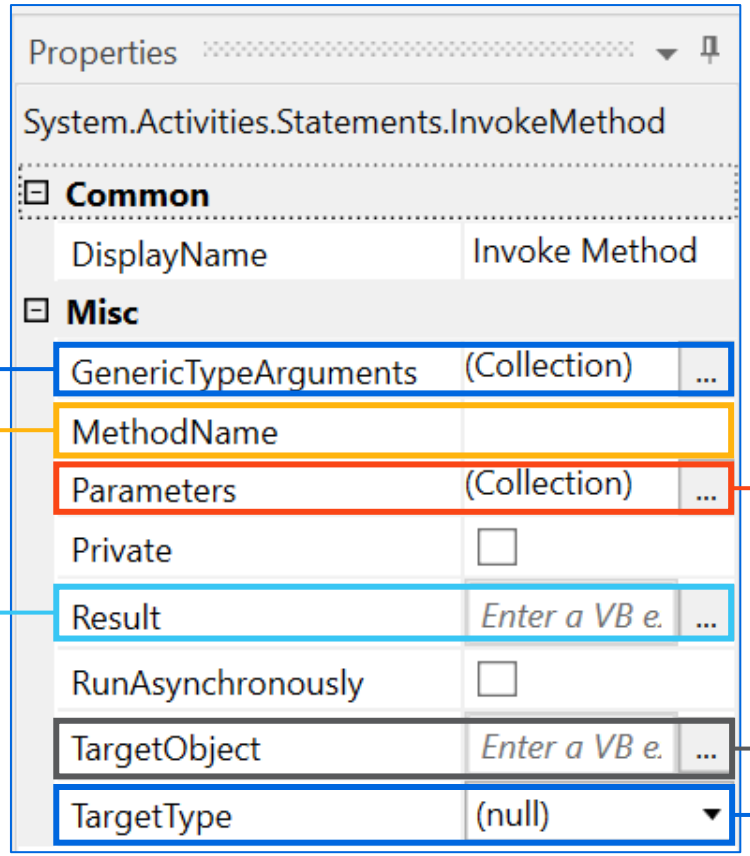
Properties of Invoke Method Activity

The properties of Invoke Method activity are:

GenericTypeArguments
Sets the generic type arguments

MethodName
Inserts the method name to be invoked

Result
Enters a variable to store output



Parameters
Enters the list of parameters to be passed to the method

TargetObject
Passes the object whose method is invoked

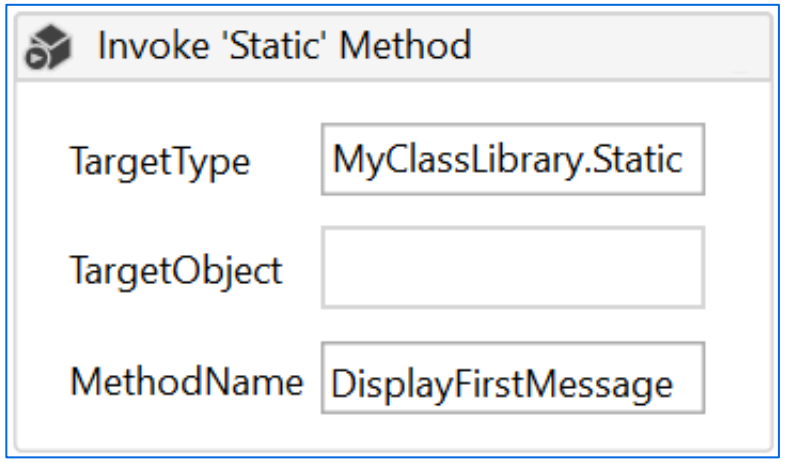
TargetType
Enters the target whose method is invoked

Types of Methods in Invoke Method Activity

Invoke Method activity can be used with the following methods:

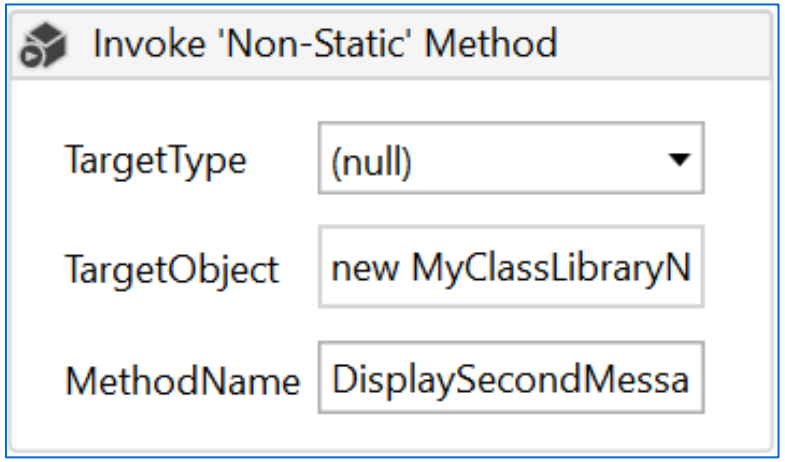
Static Method

- Used for a class defined as static
- No need to define the object
- TargetType is used



Non-Static Method

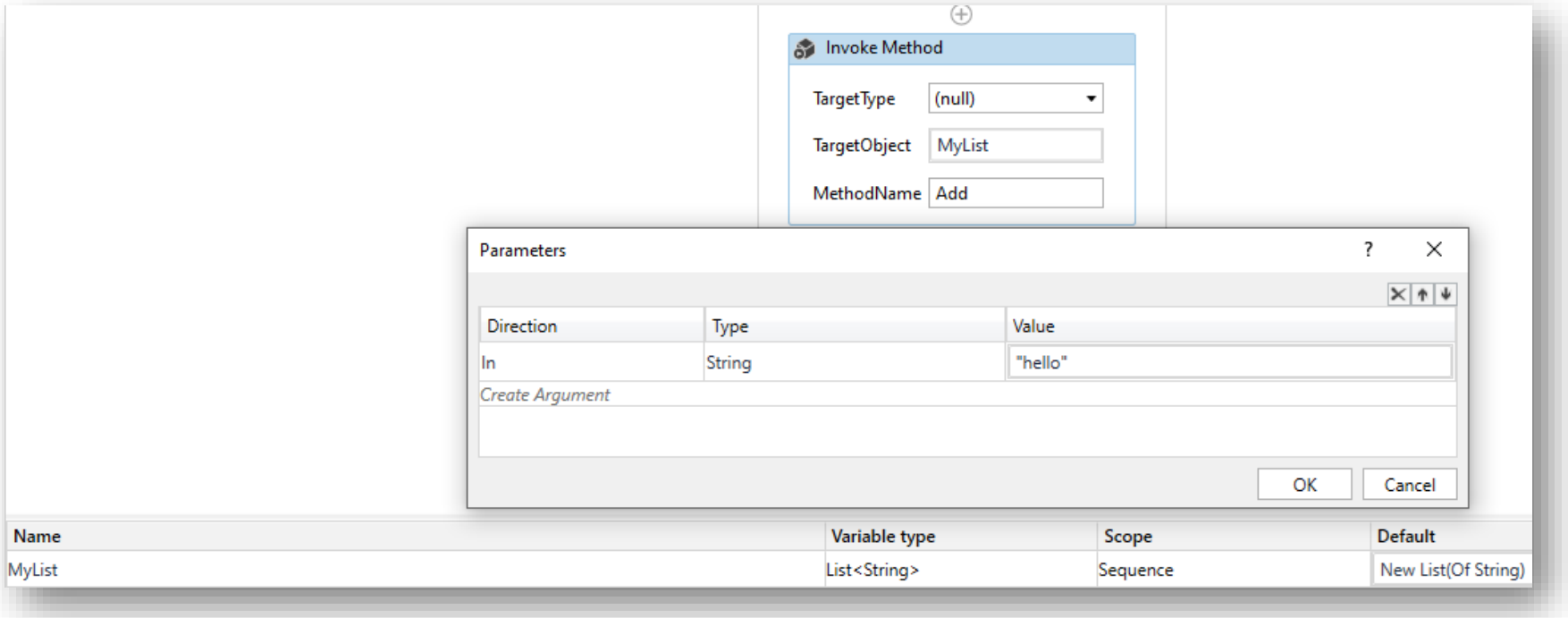
- The object (from which the method is derived) needs to be defined
- TargetObject is used



Invoke Method

Invoke method activity is useful when we need to call a public method of a class.

It is particularly used to invoke void methods.



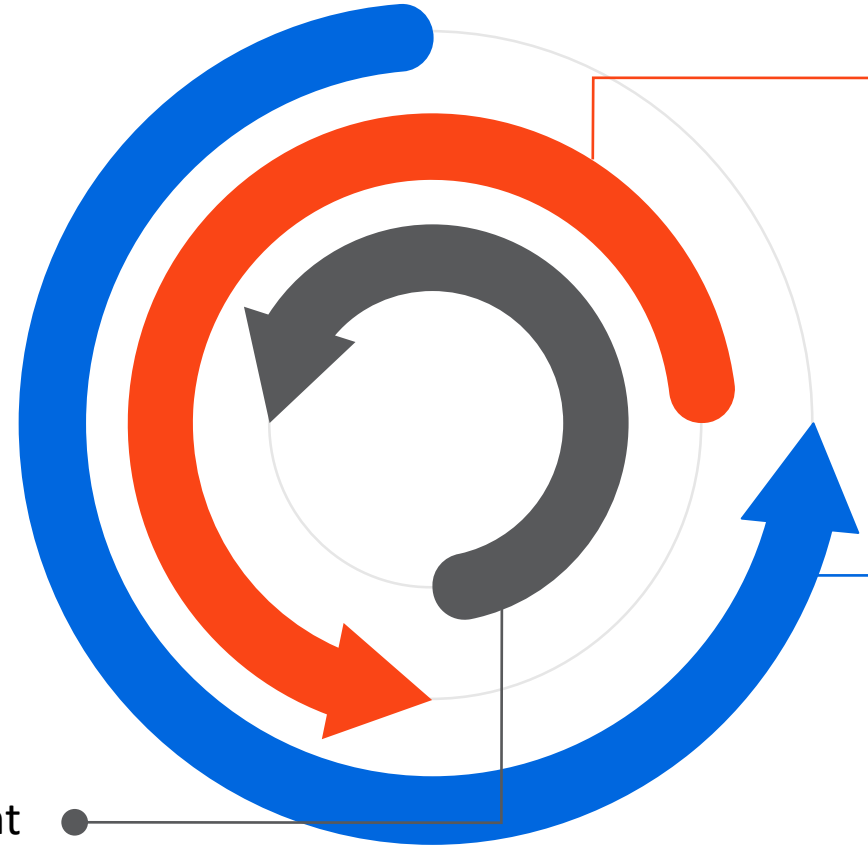
This activity adds the string "hello" to the list of strings variable named MyList.

Configure Invoke Method Activity for DLL Files

The steps to call a method from a DLL file using Invoke Method Activity are:

Source Control

Source control is used to track and manage any changes in the project. Source control systems are useful when developing larger projects that require smooth collaboration between multiple users.



Allows a user to check revision history of the project (offers recoverability)

Offers smooth collaboration between multiple users

Allows users in different teams and locations to access the same resources

Version Control in Studio

Automation projects are connected to version control systems through the Team tab in Studio. The version control systems are:

GIT

GIT is a distributed version-control system for tracking changes in source code during software development.

- **Clone Repository:** Clone a remote GIT repository.
- **Copy to GIT:** Copy the current project to an existing GIT repository.
- **Add to GIT:** Add the current project to a new local GIT repository.
- **Disconnect:** Disconnect the current project from source control.

TFS

TFS is the source code management established by Microsoft, used for the project and release management.

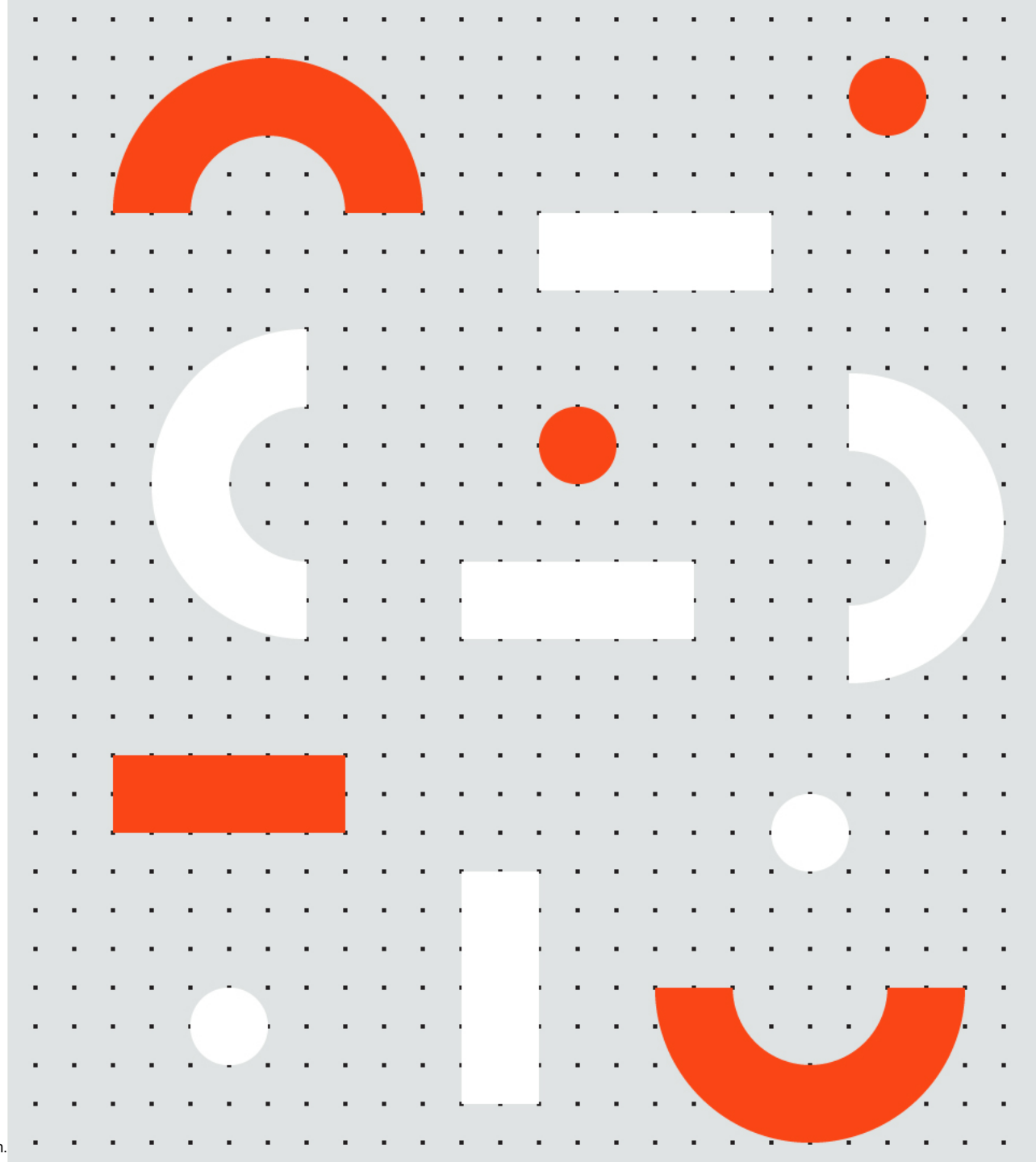
- **Open from TFS:** Open a project from a TFS repository.
- **Add to TFS:** Add the current project to TFS source control.
- **Manage TFS Online:** Go to the web management interface.
- **Disconnect:** Disconnect the current project from source control.

SVN

SVN is a software versioning and revision control system distributed as open source.

- **Open from SVN:** Open a project from an SVN repository.
- **Add to SVN:** Add the current project to SVN source control.
- **Disconnect:** Disconnect the current project from source control.

Optional Topics



Optional Topics

01 Databases

02 Monitor Events Activities

03 Citrix and RDP Automation

04 Image and Text Automation

05 Recording

06 Word Activities

07 AI Computer Vision

08 Test Suite

Optional Topics

09 Insights

10 Elasticsearch

11 Kibana

12 Custom Activities

13 Document Understanding

14 Action Center, Human in the Loop and Orchestration Process

15 Terminals

16 Attended Automation

Optional Topics

17 The UiAutomationNext Package

18 WorkFlow Analyzer

19 PowerShell Activity

20 VBA Activity

21 Python Activities Package

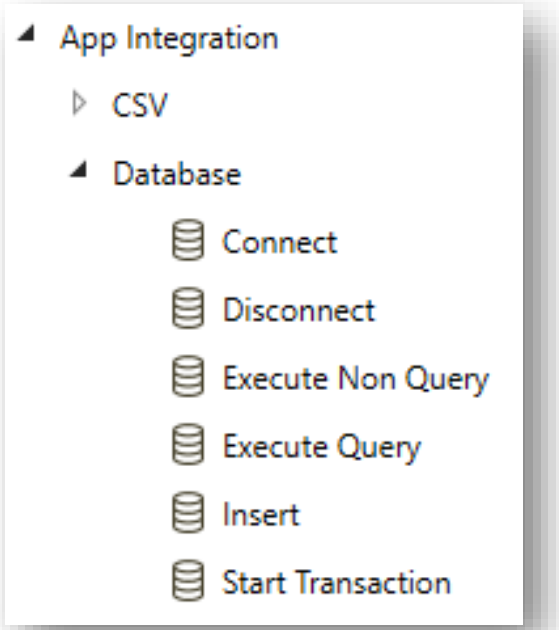
22 Java Activities Package

23 JavaScript Activity

24 Activity Project Settings

Databases

- **The Database Activities Pack** enables the user to connect to a database and executing transactions or queries and non-queries.
- The **Start Transaction** activity is a container designed to enable the user to connect to a database, perform multiple actions and then disconnect.
- The **Connect** and **Disconnect** activities can also be used to connect to a database and consequently disconnect, performing any desired actions with individual connections.
- The connection process works based on the usage of a DatabaseConnection variable.



- The **Database** activity pack is open-source! Feel free to collaborate on UiPath's Github repositories.

Monitor Events Activities

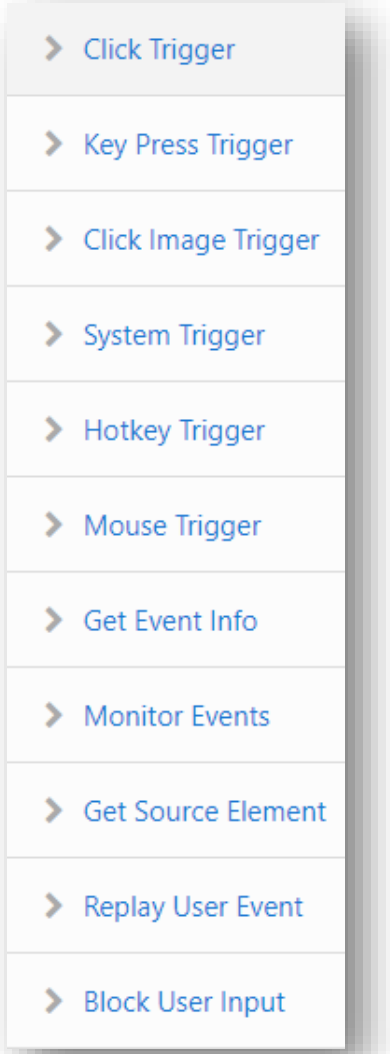
- Capture keyboard usage or mouse clicks anywhere on the screen, regardless of the application you are interacting with.
- Monitor changes in the file system, so you can trigger events based on folders and files changes.
- Capture keyboard usage or mouse clicks on a specific UI element that we want to monitor.
- Monitor mouse clicks on a specific image.

Monitoring properties

- **RepeatForever** - the response is triggered only one time - first time or every time the event happens.

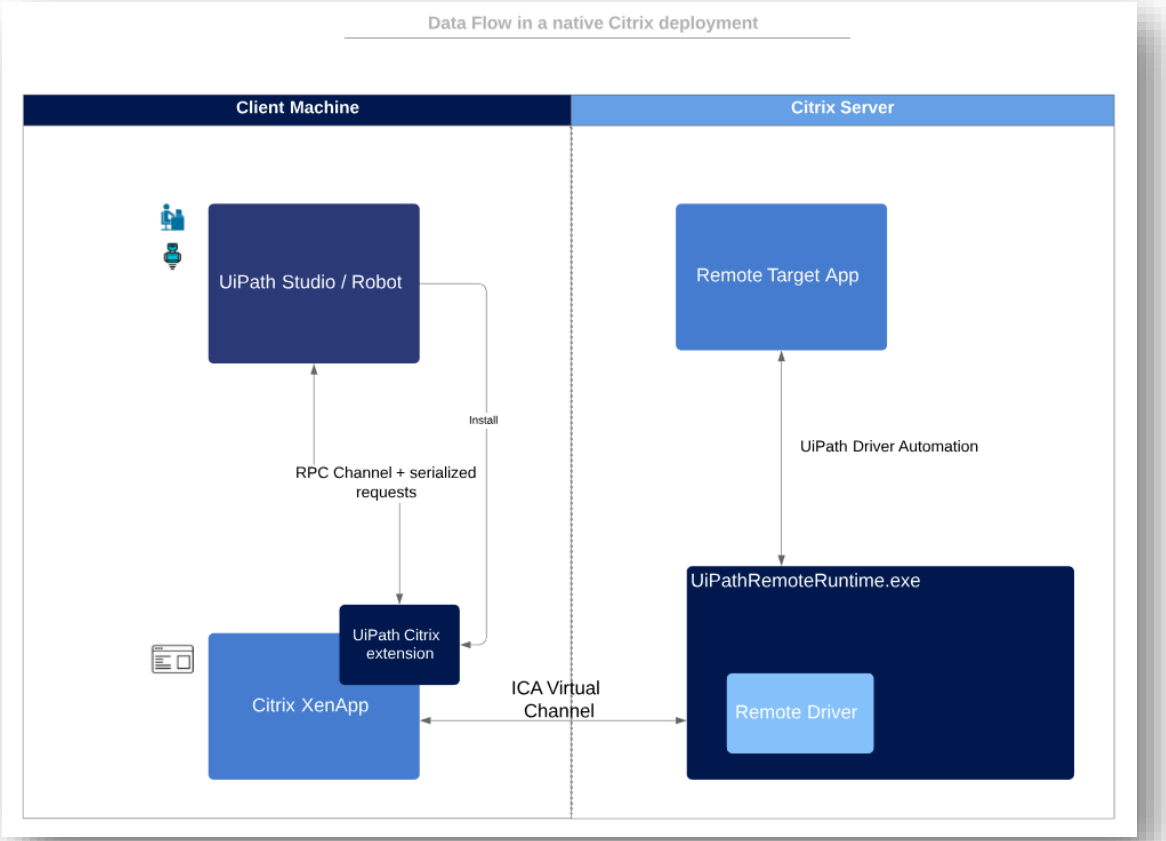
Trigger properties

- **Synchronous and Asynchronous Events Capturing** - in a synchronous case, workflow response happens before or instead of the app's original response; in an asynchronous case, workflow response does not interfere with the original response and happens in parallel.
- **Block event** - stops the original response from taking place; makes sense only in Synchronous scenario.



Citrix and RDP Automation

- **UiPath Remote Runtime** is a component which facilitates the communication between a remote application or desktop, such as Citrix Virtual Apps, and the dedicated UiPath extension - the **UiPath Extension for Citrix**, the **UiPath Extension for Windows Remote Desktop**, or the **UiPath Extension for VMware Horizon**.
- It gathers info about targeted UI elements of remote applications and sends them to the corresponding extension so that selectors are natively generated in UIExplorer.
- With the UiPath Extension for Citrix and UiPath Remote Runtime component installed, the following actions are enabled:
  - Generating selectors for UI elements in Citrix Apps and Desktops.
  - Using the activities from the UiPath.UIAutomation.Activities package (such as Click, Type Into, and more).
  - Using Data and Screen Scraping Wizards.
  - Automating browsers opened as Citrix Apps.



This is what the generated selector for Calculator opened as a Citrix App looks like:

```
<wnd app='win32calc.exe' cls='CalcFrame' isremoteapp='1' title='Calculator' />
<wnd ctrlid='137' />
```

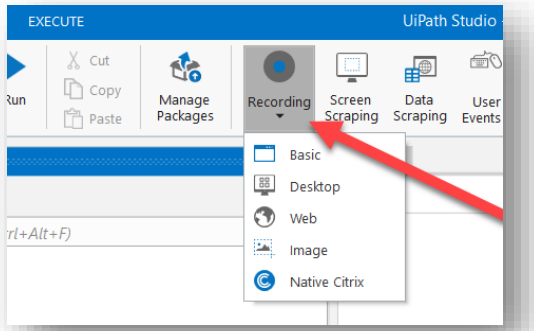
Image and Text Automation

- UiPath Studio features activities that simulate **keyboard and mouse** input, such as clicking, hovering or typing, **text recognition** and **OCR** activities that use screen scraping to identify UI elements, and **image recognition** activities that work directly with images to identify UI elements.
- These activities are useful in legacy applications were few or no selectors are available, or when any other automation method didn't work.
- **Mouse and keyboard activities:** Click, Hover, Type Into, Type Secure Text, Send Hotkey, Set Focus
- **Text activities:** Click Text, Hover Text, Find Text Position, Get Full text, Get Visible Text, Extract Structured Text, Text Exists
- **OCR Activities:** Click OCR text, Hover OCR Text, Get OCR Text, Find OCR Text Position, OCR Text Exists
- **Image activities:** Click Image, Hover Image, Find Image, Image Exists, On Image Appear, On Image Vanish

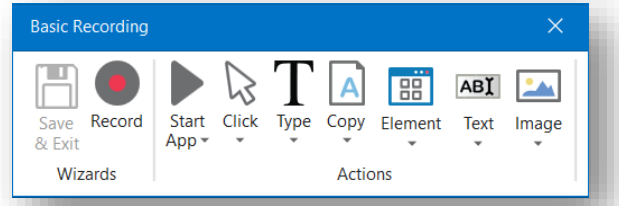
- When using image activities, pay attention to resolution changes! Images taken with a certain resolution will fail to be recognized on a screen with another resolution.

Recording

There are five types of recordings available in UiPath.



Web – Designed for recording in web apps and browsers, generates containers and uses the **Simulate Type/Click** input method by default.



Basic – Generates a full selector for each activity and no container. The resulted automation is slower than one that uses containers and is suitable for single activities.

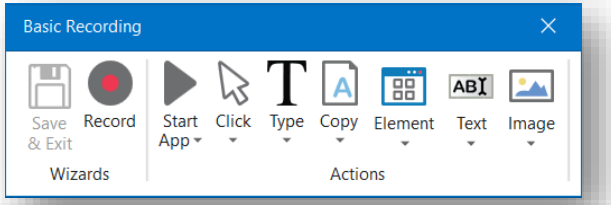
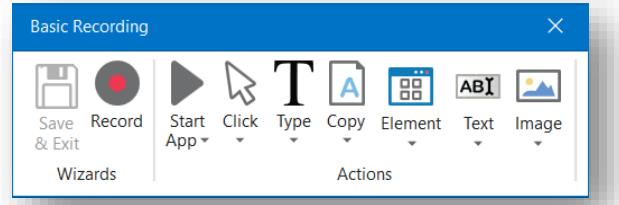
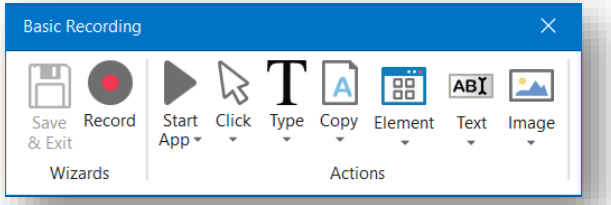


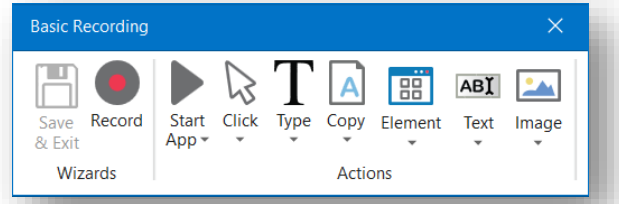
Image – Used to record virtualized environments (such as VNC, virtual machines, Citrix, and more) or SAP. It permits only image, text and keyboard automation, and requires explicit positioning.



Desktop – Suitable for all types of desktop apps and multiple actions. It is faster than the Basic recorder and generates a container (with the selector of the top-level window) in which activities are enclosed, and partial selectors for each activity.



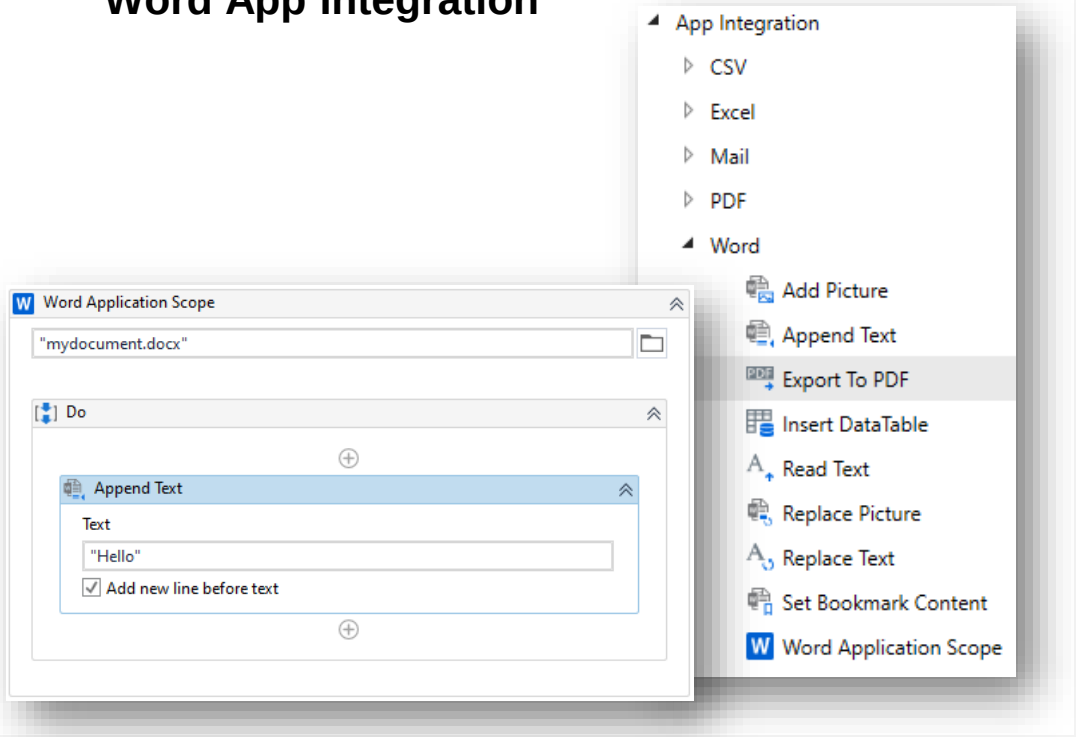
Native Citrix - Is the equivalent of the **Desktop** recorder, but for Citrix environments. Only use this in your Native Citrix automation projects.



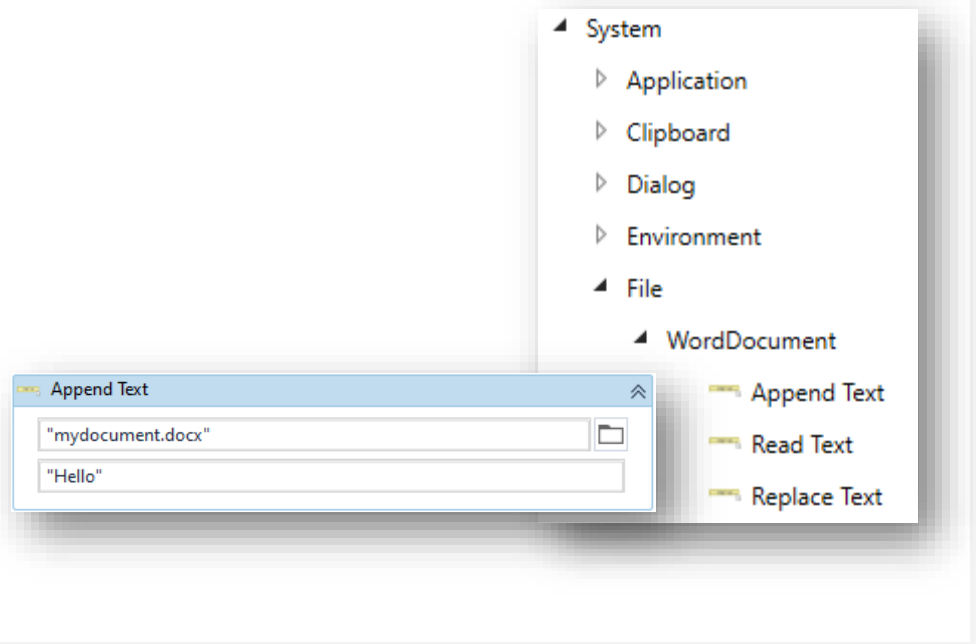
Word Activities

- The Word package contains multiple activities that enable you to manipulate .docx files.
- It is important to note that all the activities under the **App Integration** category can only be used within Word Application Scope, while the others in the **System** category can be used without it.

Word App Integration

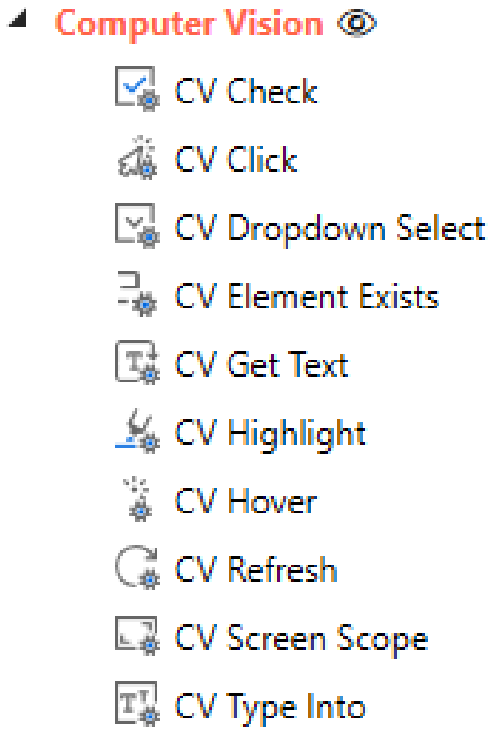


System



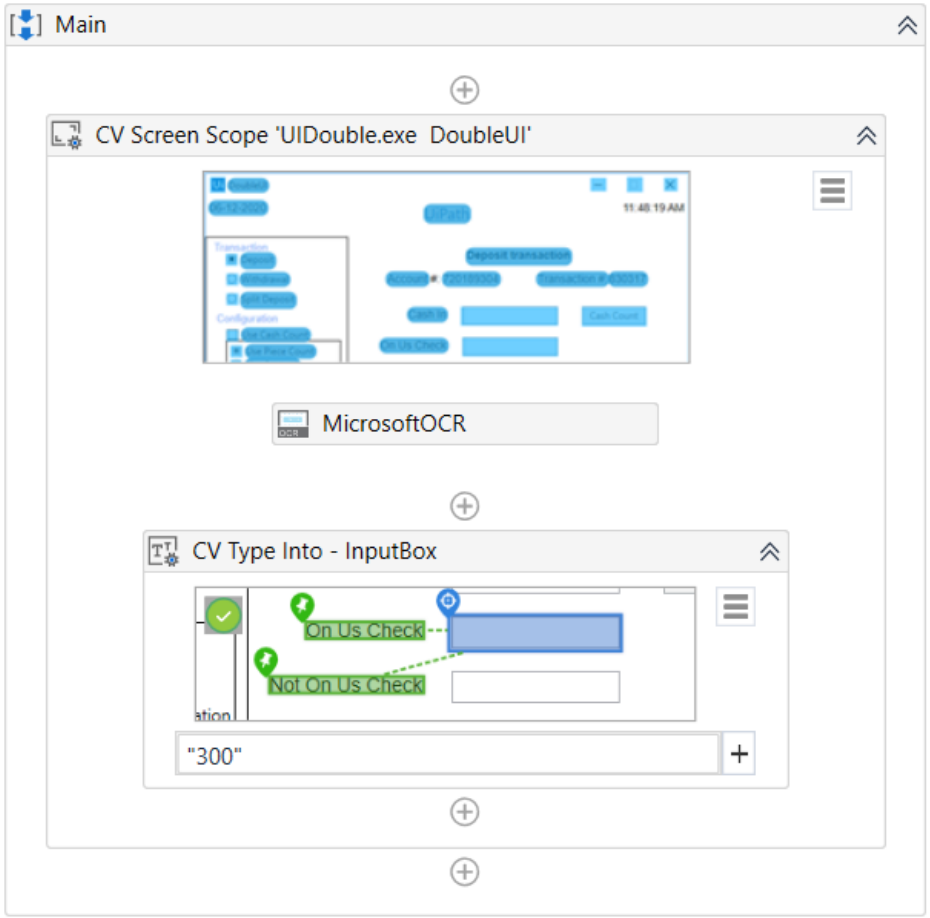
AI Computer Vision

- **Computer Vision** is a Machine Learning based method that **allows robots to "see"** the screen and identify all the elements, rather than relying on selectors or Image Automation.
- *When to use it:* to automate Virtual Desktop Environments (e.g. Citrix, VMware, Windows Remote Desktop), to automate legacy applications that do not provide selectors, etc.
- *Deployment options:* UiPath CV server (https://cv.uipath.com) or on-prem instance.



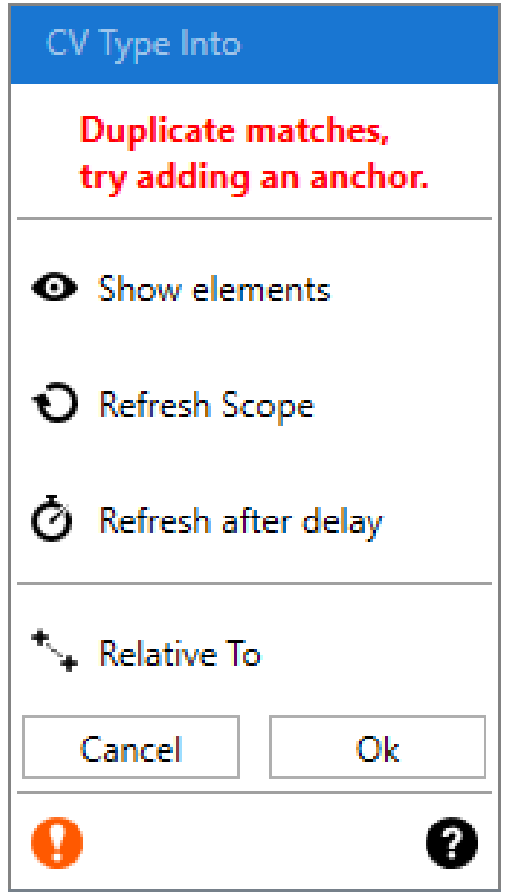
AI Computer Vision

- *How to use:* Make sure that the **UIAutomation** package is installed (version **19.10+**).
- Make sure you have at least one **OCR engine**
- Get an **API key** from Cloud Platform.
- Start your CV automations with a **CV Screen Scope**, as all the other CV activities need to be contained in one.
- The target element of a CV activity is identified using **one or more anchors**.



AI Computer Vision

- The **CV Helper** is presented when configuring a CV activity.
- The top part contains indications for the developer.
- **Show elements** will show all detected elements within the scope.
- **Refresh Scope/Refresh after delay** will refresh the analysis done in the CV Screen Scope, in case it needs to be updated.
- Relative to can be used to interact with elements which we not recognized, by using another element as a reference for the action.



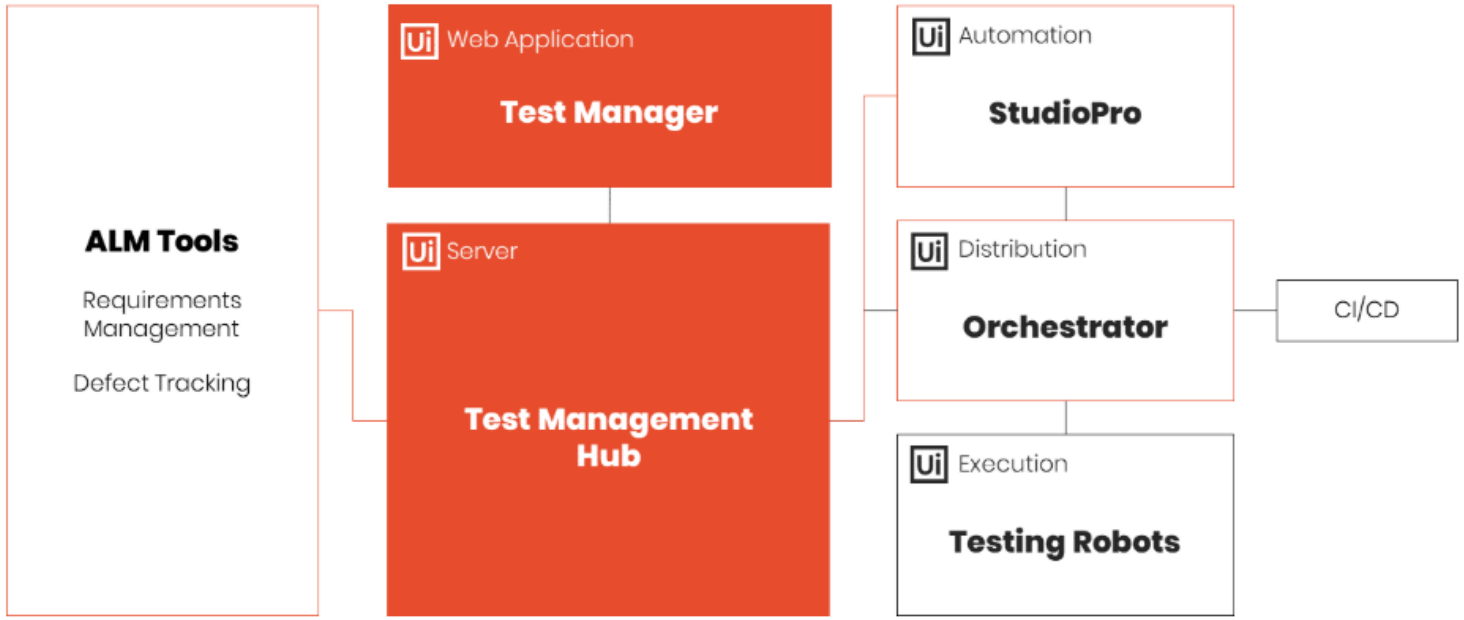
Test Suite

- Introduced in 20.4; only for Enterprise licenses.
- Offers CI/CD pipelines, with Jira, Jenkins and Git.
- Features:
  - Application testing - specialized test cases for conducting automated application testing, used for verifying data, and included in CI/CD pipeline scenarios.
  - RPA Testing - designed for testing workflows and viewing the activity coverage during execution. Such testing processes ensure that execution is performed, and all corner cases are covered.
  - API testing - integration with Postman.

Test Suite

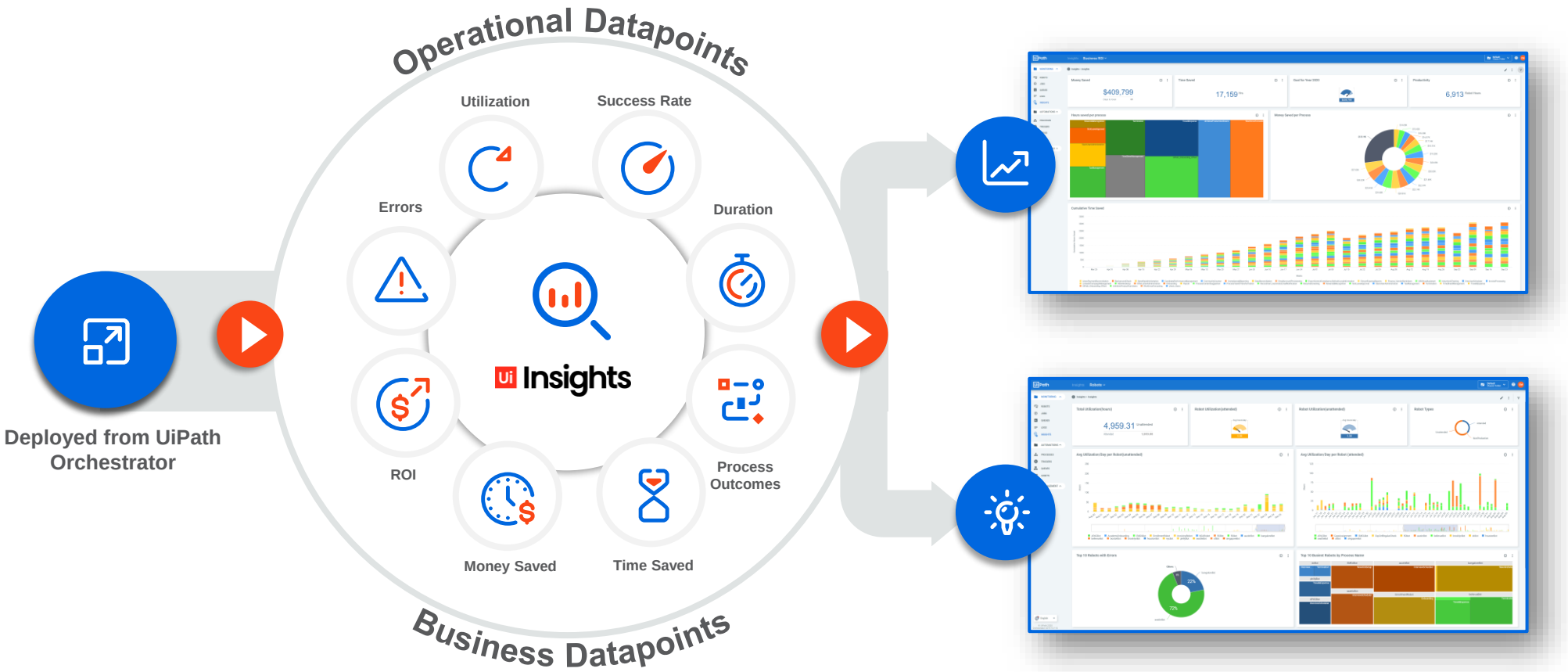
UiPath Test Manager is the component of the **UiPath Test Suite** used for managing test projects and test requirements and for managing test results.

UiPath Test Manager consists of the web application and the hub/server. The server integrates with **UiPath StudioPro** and **UiPath Orchestrator** and can be integrated with third party Application Lifecycle Management (ALM) tools (e.g. Jira).

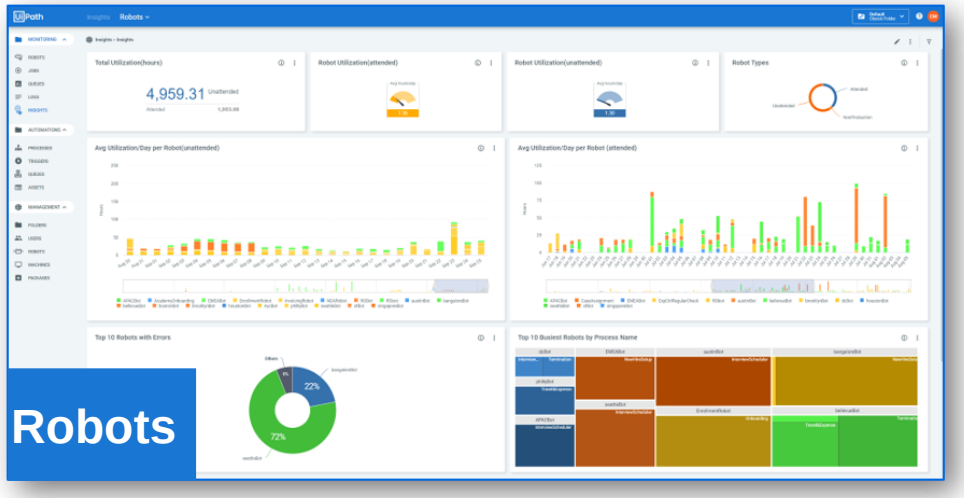


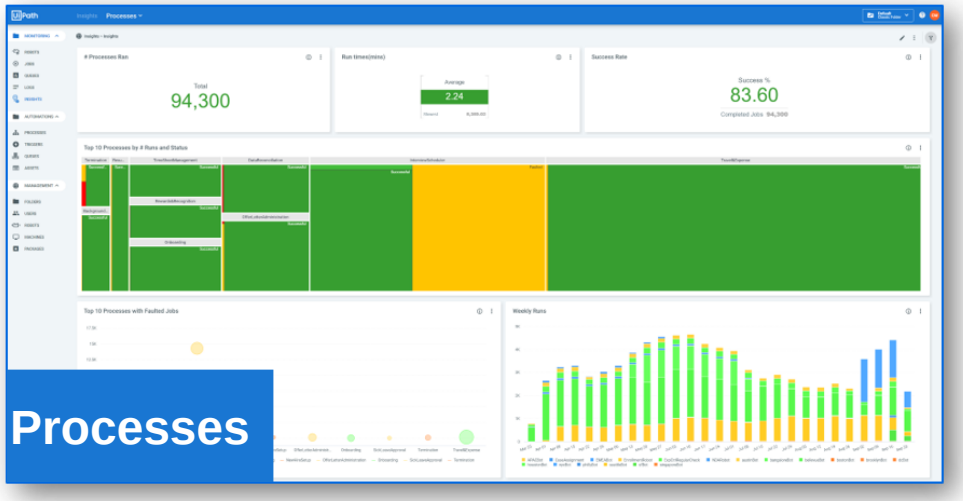
Insights – Powerful, Embedded Analytics

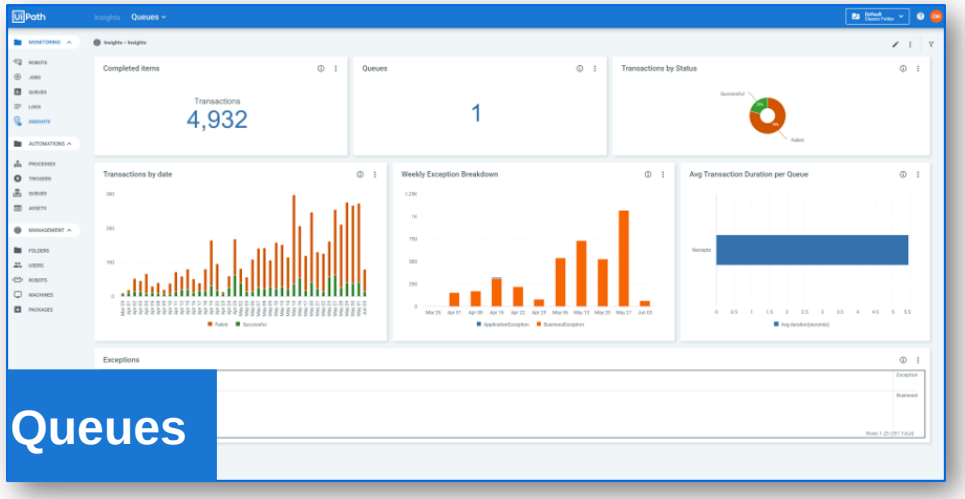
1. Easily launched from Orchestrator
2. Track and measure RPA and process performance
3. Easily align and report the business impact of your automations



Insights



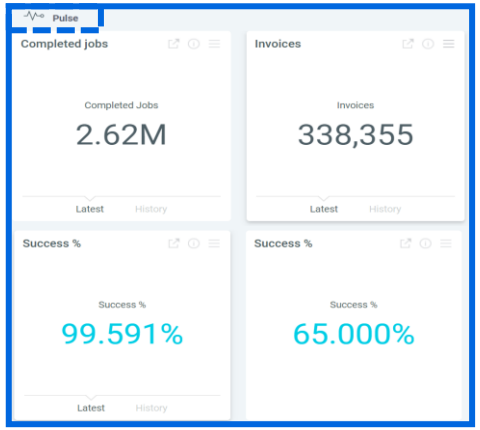




Insights

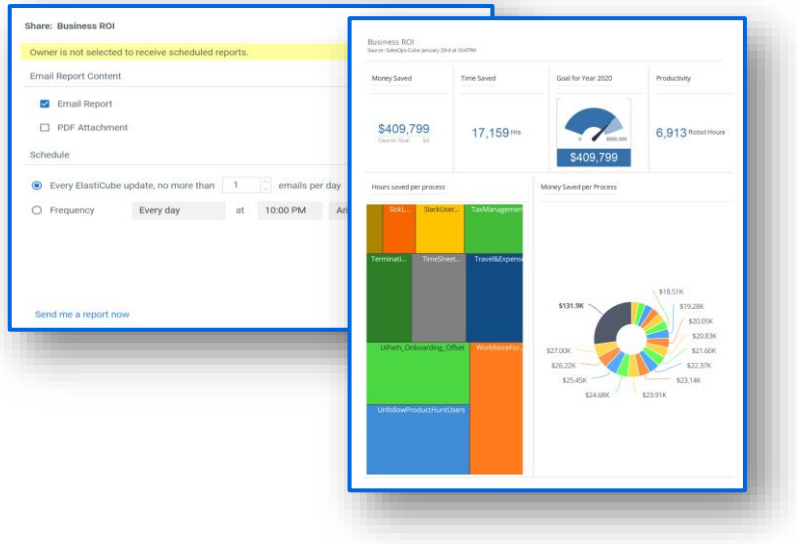
Smart alerts

Instant notification of anomalies, critical events and major milestones



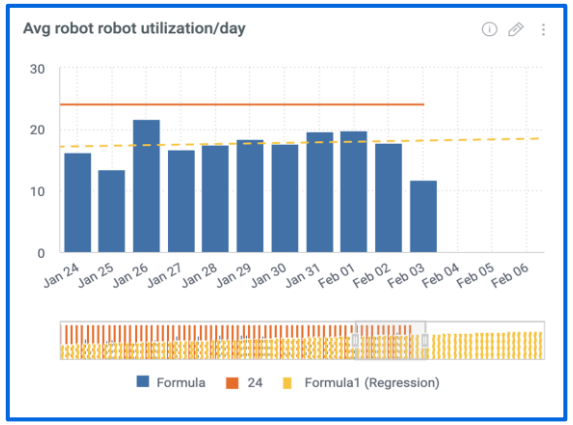
Shareable reports

Schedule to instantly share dashboards and reports



Forecasting based on historical trends

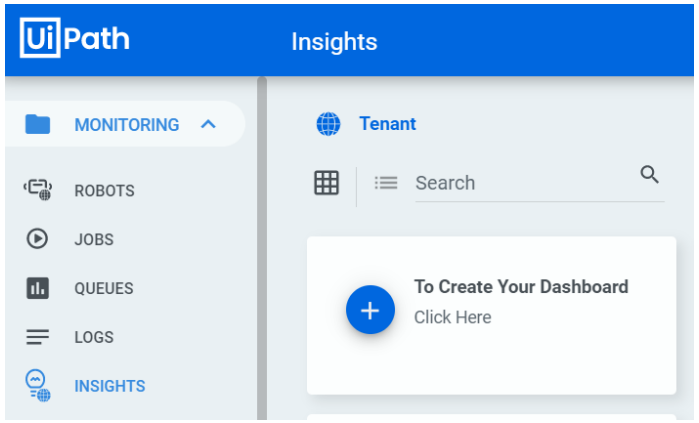
Forecast critical KPIs, including robot utilization and process bottlenecks



Insights

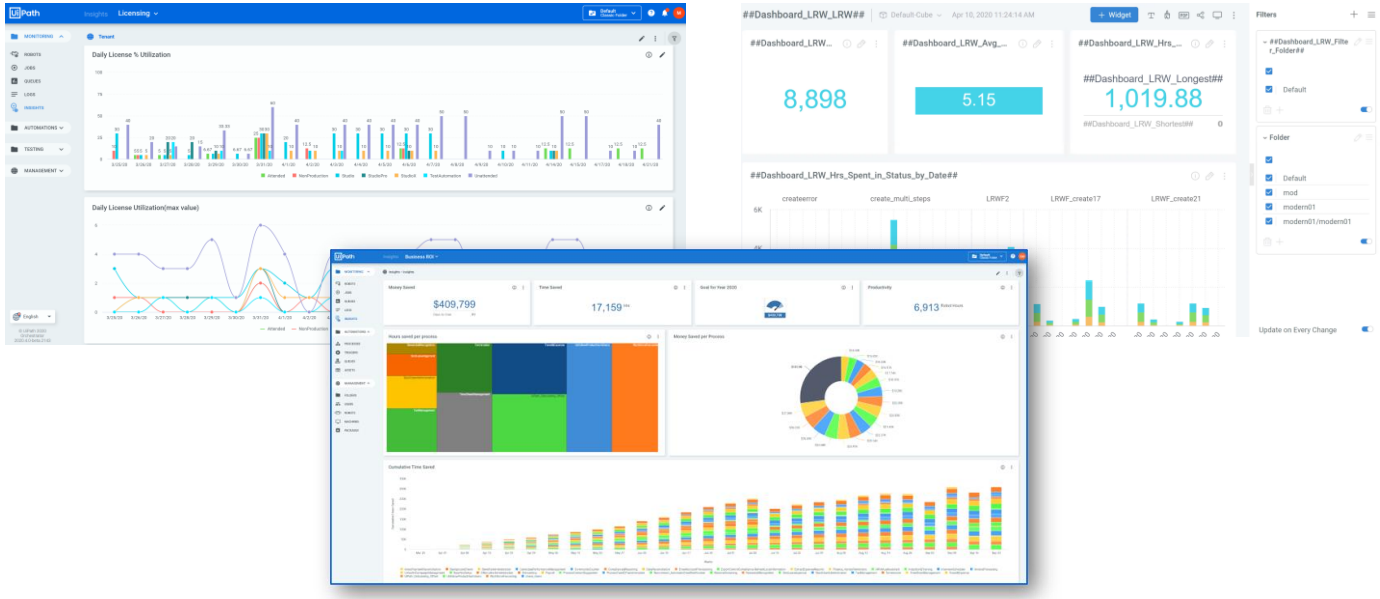
Simple setup

*Deploy, launch and use Insights into the **existing Orchestrator interface***



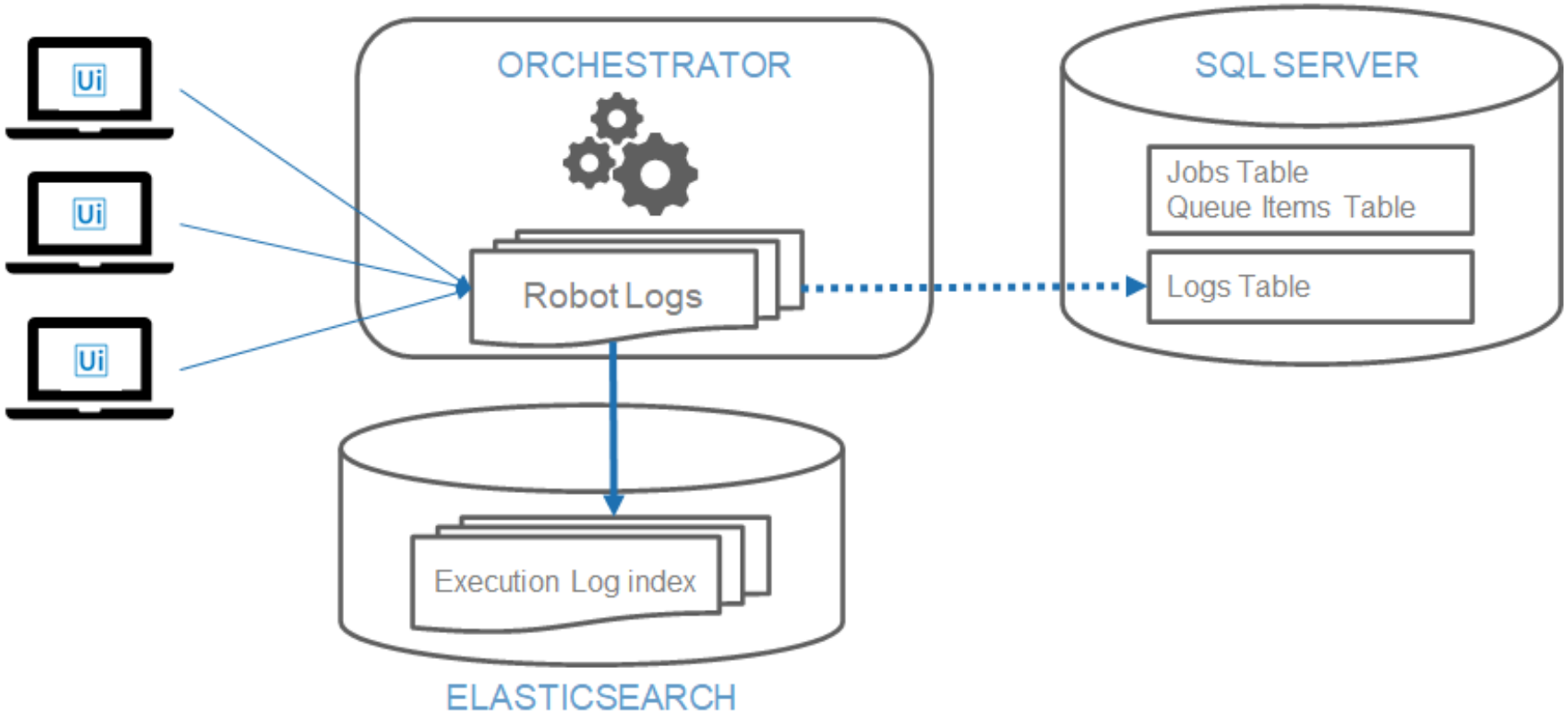
Dynamic dashboards

*Drag-and-drop library of **out-of-the-box dashboards***

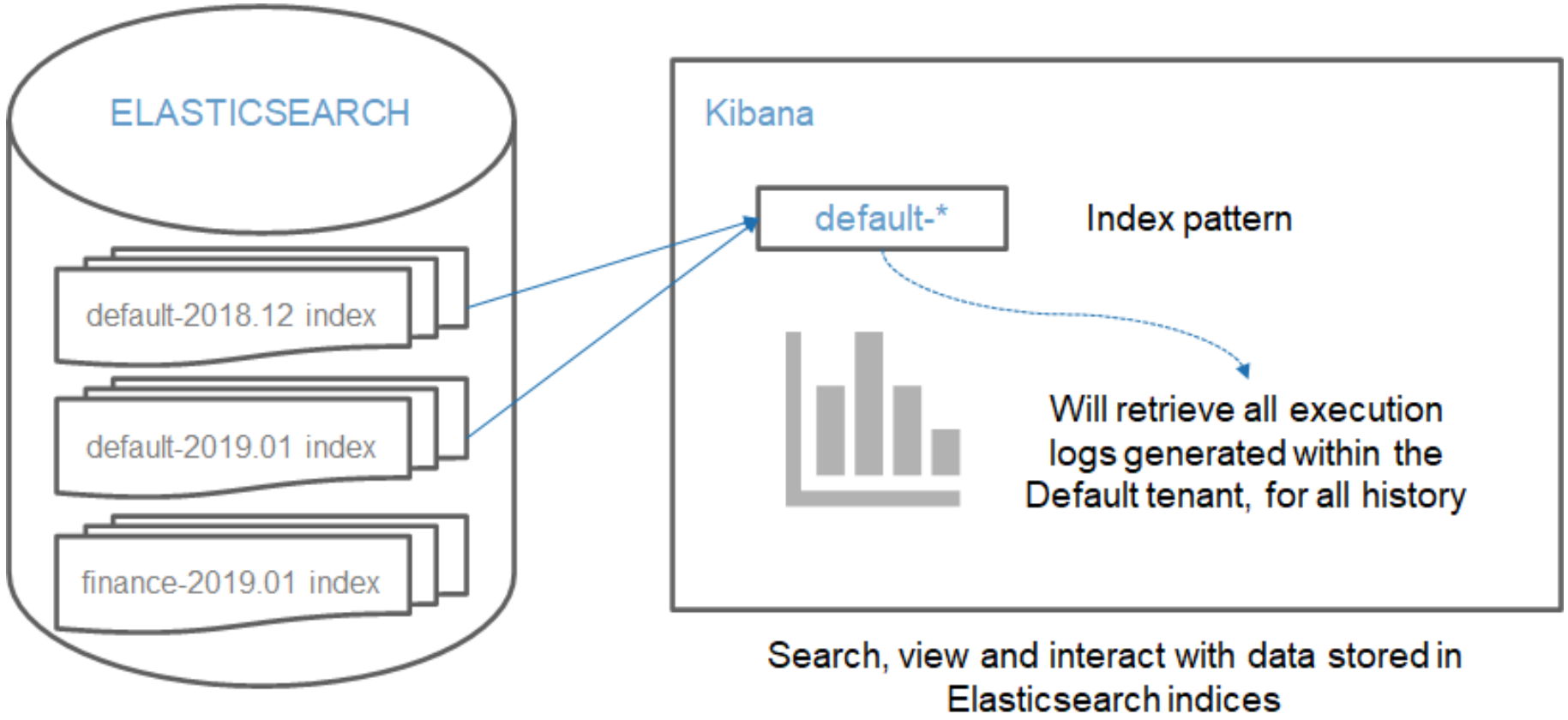


Elasticsearch

ES is a NoSQL, distributed full text database and search engine.



Kibana

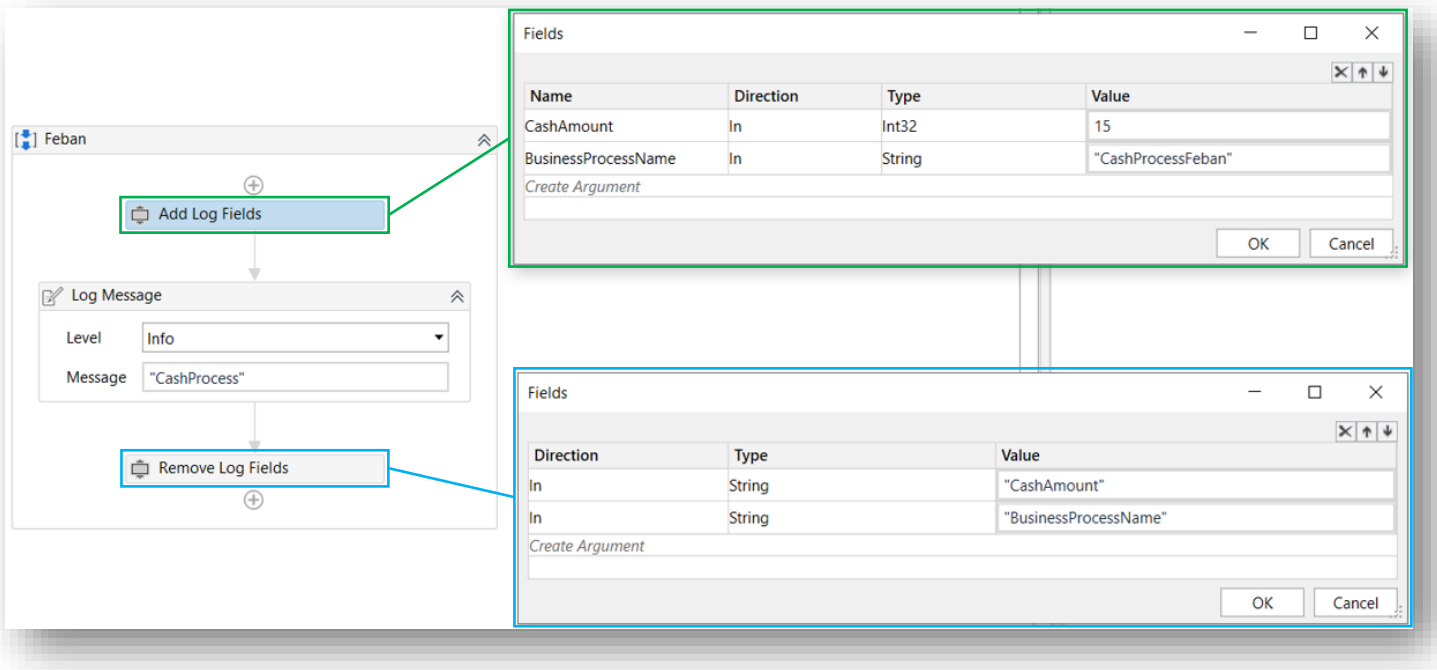


Elasticsearch and Kibana Best Practices

- Only log what is relevant for the business case
- Use the "message" field to store human-readable results
- Use atomic log fields
- Standardize the logs, create and maintain a unique data model
- Logically break down the process in work blocks that need to be individually monitored
- Each log should contain all the information required for that work block – joining logs is difficult in NoSQL systems.
- Each work block should have no more than one execution log – avoid double counting
- Do not use execution logs for primary storage of business data

Adding Logs for Kibana Query Purposes

- **CashProcess** - a set of processes used by a bank for cash payments
- **CashProcessFeban** – a subprocess of CashProcess
- **Example 1**: Kibana visualization "Total Amount Cash per Day" – create a line chart of total amount processed per day in all Cash Processes by summing all the amounts of the field **CashAmount** where the log message contains "**CashProcess**"
- **Example 2**: Kibana visualization "Total Amount Cash processed in Feban per Day" – same as above, with the added condition that **BusinessProcessName** is **"CashProcessFeban"**



Don't forget to remove the specially added log fields after the log message!

Forgetting to do so in this case would add the amount to the total for each subsequent log message.

Custom Activities

- Prerequisites:
  - Microsoft Visual Studio with the .NET desktop development and Workflow foundation workloads installed.
  - NuGet Package Explorer.
  - Microsoft Nuget Package source: https://nuget.org/api/v2/
- There are two major steps required to create a custom activity:
  - Writing the custom activity code.
  - Adding the external assembly (.dll) in UiPath Studio.
- Custom activities are created by either assembling system-provided activities into composite activities or by creating new types that derive from CodeActivity, AsyncCodeActivity or NativeActivity.
- Steps:
  - 1.Implement custom activity.
  - 2.Test the custom activity in a unit test project, in a console application project and (lastly) in UiPath Studio.
  - 3.Create a Nuget package from the main project. Don't manually add dependent dlls, use Nuget Package Manager to add them.
  - 4.Increment the package version.
  - 5.Import it in UiPath Studio and test the custom activity.
  - 6.(Optionally) Publish it on https://connect.uipath.com/marketplace/.

Custom Activities

Arguments are used to define the way data flows into and out of an activity. Types: In, Out, In/Out.

```
namespace UiPath.Workshop.Activities
{
    0 references | 0 changes | 0 authors, 0 changes
    public class SumActivity : CodeActivity
    {
        [RequiredArgument]
        [Category("Input")]
        [DisplayName("First Number")]
        [Description("The first operand of the sum")]
        1 reference | 0 changes | 0 authors, 0 changes
        public InArgument<int> FirstNumber { get; set; }

        [RequiredArgument]
        [Category("Input")]
        [DisplayName("Second Number")]
        [Description("The second operand of the sum")]
        1 reference | 0 changes | 0 authors, 0 changes
        public InArgument<int> SecondNumber { get; set; }

        [RequiredArgument]
        [Category("Output")]
        [DisplayName("Result")]
        [Description("The result of the sum")]
        1 reference | 0 changes | 0 authors, 0 changes
        public OutArgument<int> Result { get; set; }

        1 reference | 0 changes | 0 authors, 0 changes
        protected override void Execute(CodeActivityContext context)
        {
            int first = FirstNumber.Get(context);
            int second = SecondNumber.Get(context);
            int result = first + second;
            Result.Set(context, result);
        }
    }
}
```

Custom activities can also have a designer (although not mandatory).

1) Tight coupling

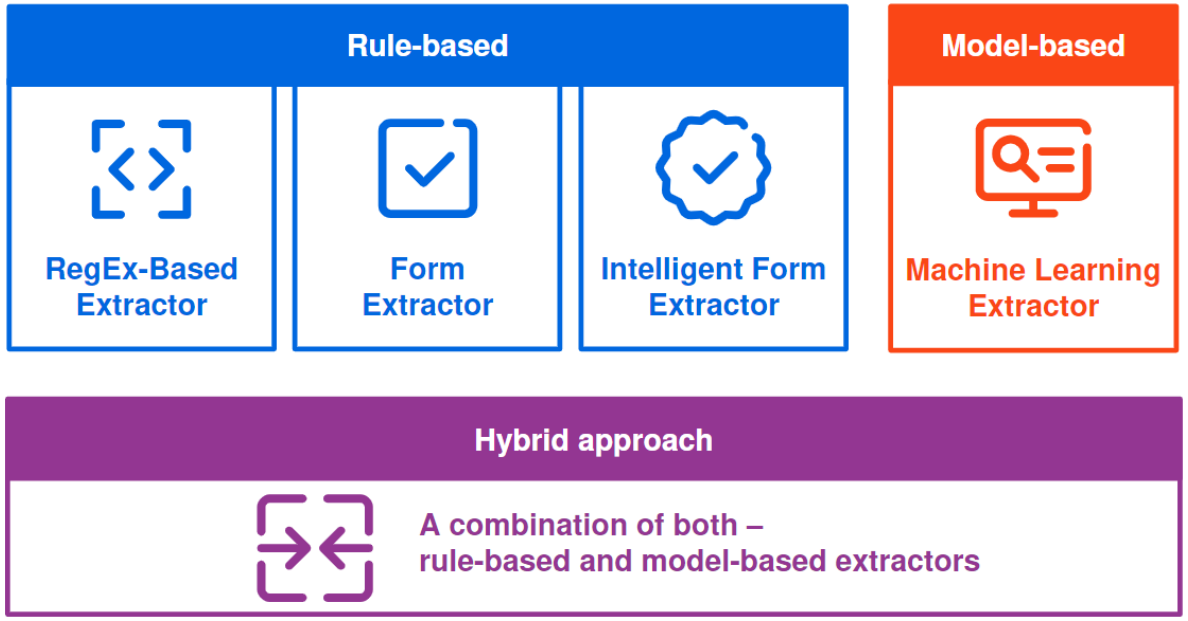
```
[Designer(typeof(MyActivityDesigner))]
public class MyActivity : CodeActivity { … }
```

2) Loose coupling

```
public class DesignerMetadata : IRegisterMetadata
    {
        public void Register()
        {
            var builder = new AttributeTableBuilder();
            builder.AddCustomAttributes(
                typeof(MyActivity),
                new DesignerAttribute(typeof(MyActivityDesigner)));
            MetadataStore.AddAttributeTable(builder.CreateTable());
        }
    }
```

Document Understanding

- Extract data from structured, semi-structured and unstructured documents.



Rule-based
[template-based approach]

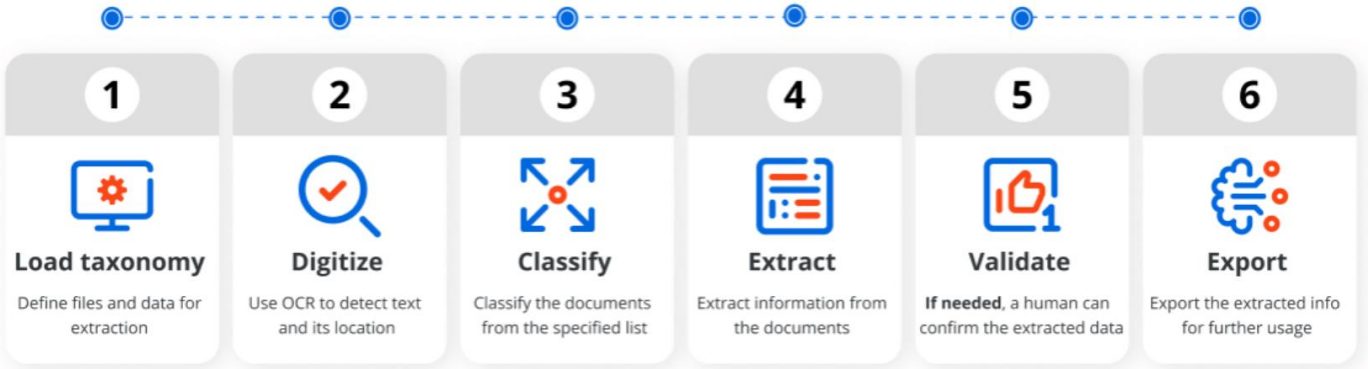
- ✓ Relies on rules and templates
- ✓ Processes fixed in format structured data like forms or licenses
- ✓ High accuracy for already known documents
- ! Requires extra costs for addition of templates/rules and ongoing maintenance
- ! Doesn't work with unknown documents

Model-based
[template-less approach]

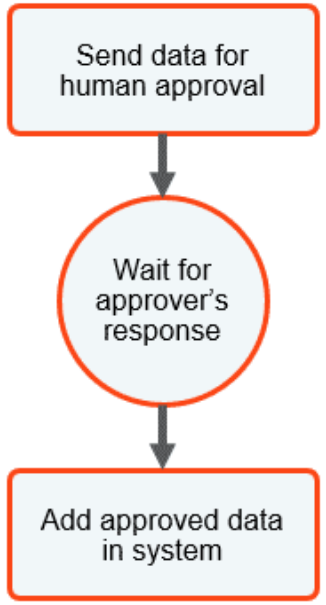
- ✓ Relies on ML models
- ✓ Processes less structured varying layouts like invoices or receipts
- ✓ Understands even unknown complex documents
- ! Requires pre-trained ML models with further retraining capabilities

Document Understanding

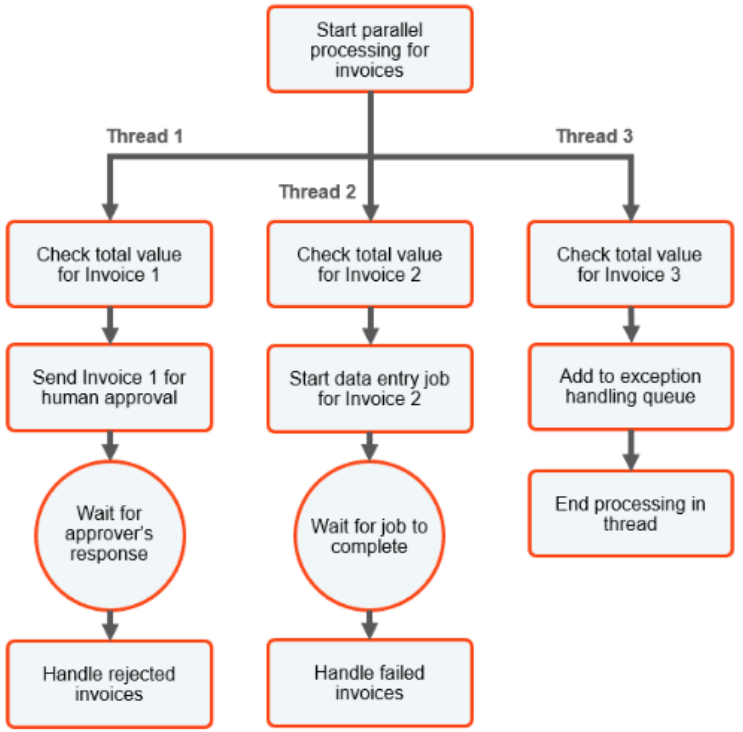
- Steps of a DU project:
  - Load taxonomy
  - Document digitization (using an OCR engine of choice)
  - Document classification
  - Data extraction
  - Human validation (attended Validation Station or Validation action – in Orchestrator)
  - Data export



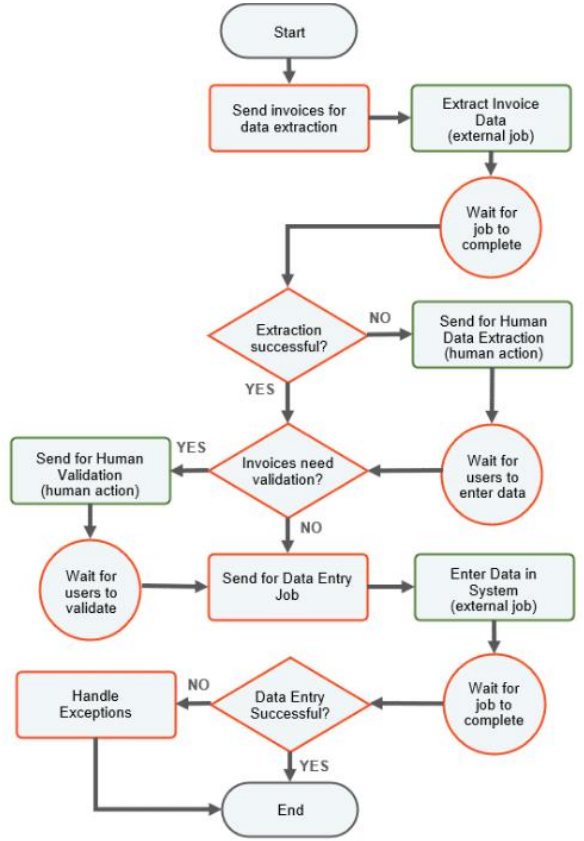
Action Center, Human in the Loop and Orchestration Process Key Concepts



Long-running workflow - a workflow which needs to wait for an external service to complete or a Human user to provide input before it can continue. In the "waiting" state, the job is marked as **Suspended**.

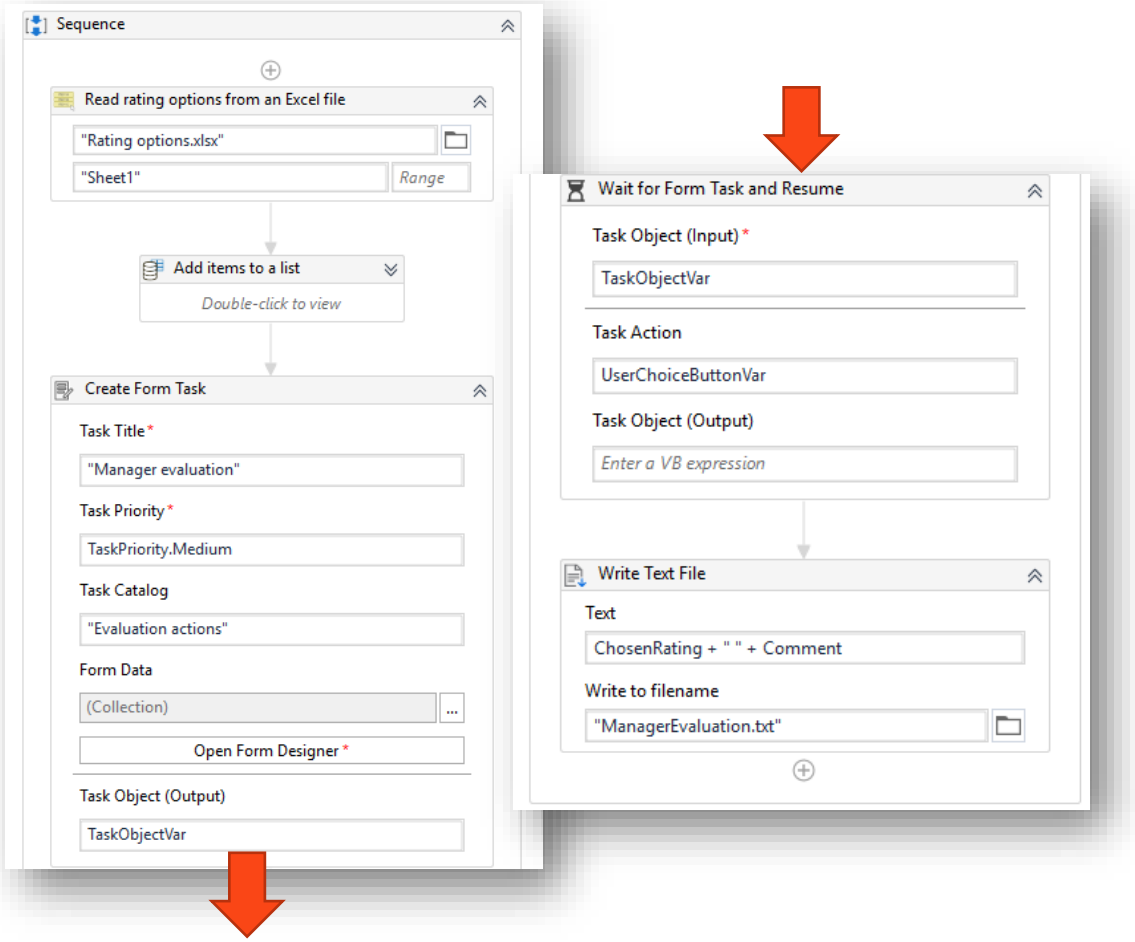


Asynchronous Processing - Parallel for Each activity - multiple transactions can be processed at the same time on parallel threads.



The Orchestration Process - new type of UiPath process in Studio to allow long-running workflows and async processing.

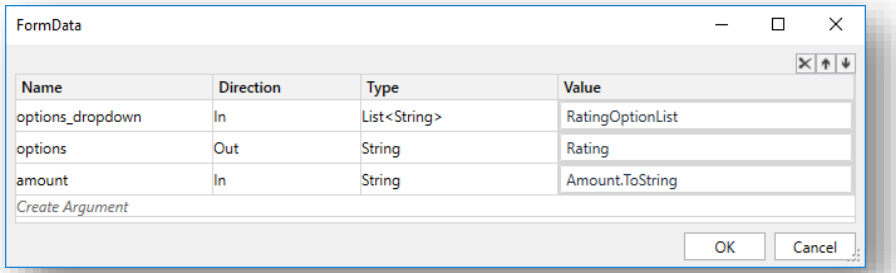
Action Center, Human in the Loop and Orchestration Process



Workflow

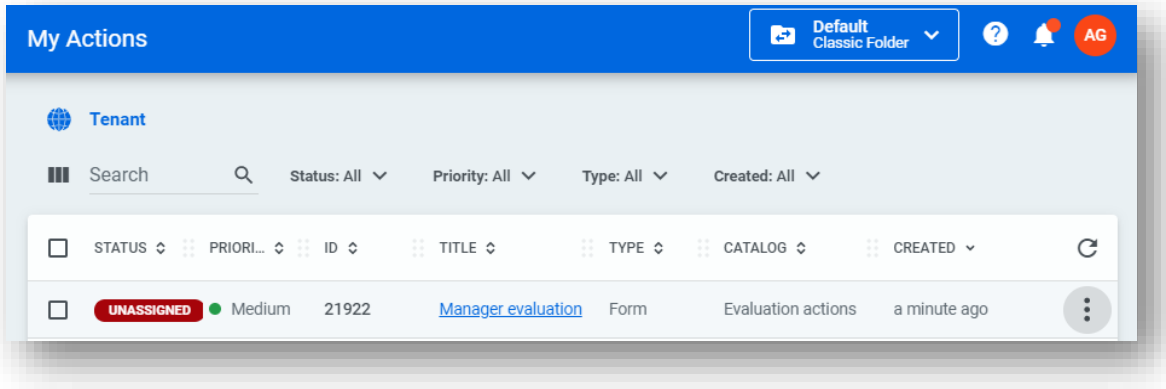
Workflow

- There are long-running activities which can be used in projects to build persistence points (e.g. for this case, Create Form Task & Wait for Form Task and Resume)
- **Field mapping**
- Each action task has a GUI containing data fields & buttons; this GUI is easily built in Studio using drag and drop actions;
- The form fields need to be mapped to project variables in order to pass data to the form and retrieve data from the user.



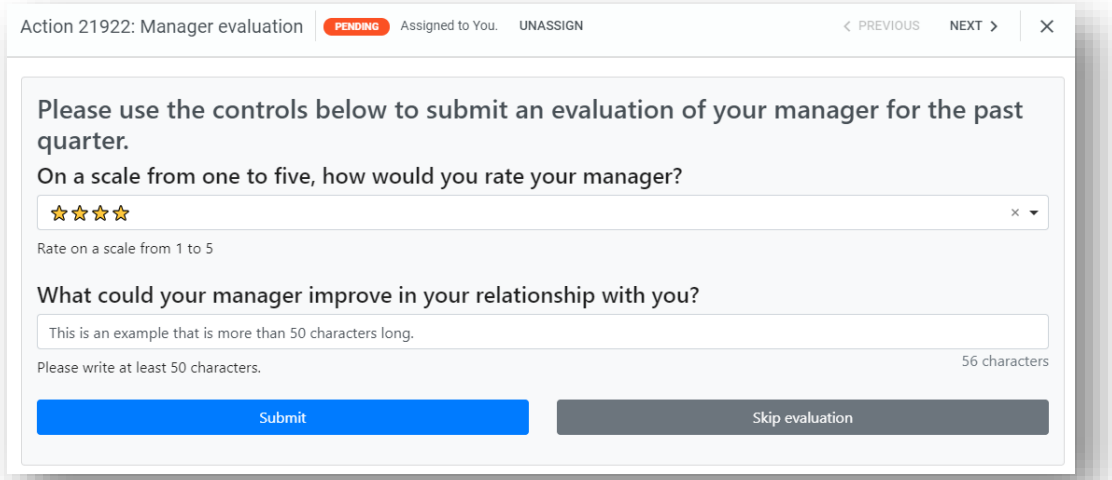
Field mapping

Action Center, Human in the Loop and Orchestration Process



Actions in Orchestrator

- The actions can be seen in a list in Orchestrator, under "My Actions".



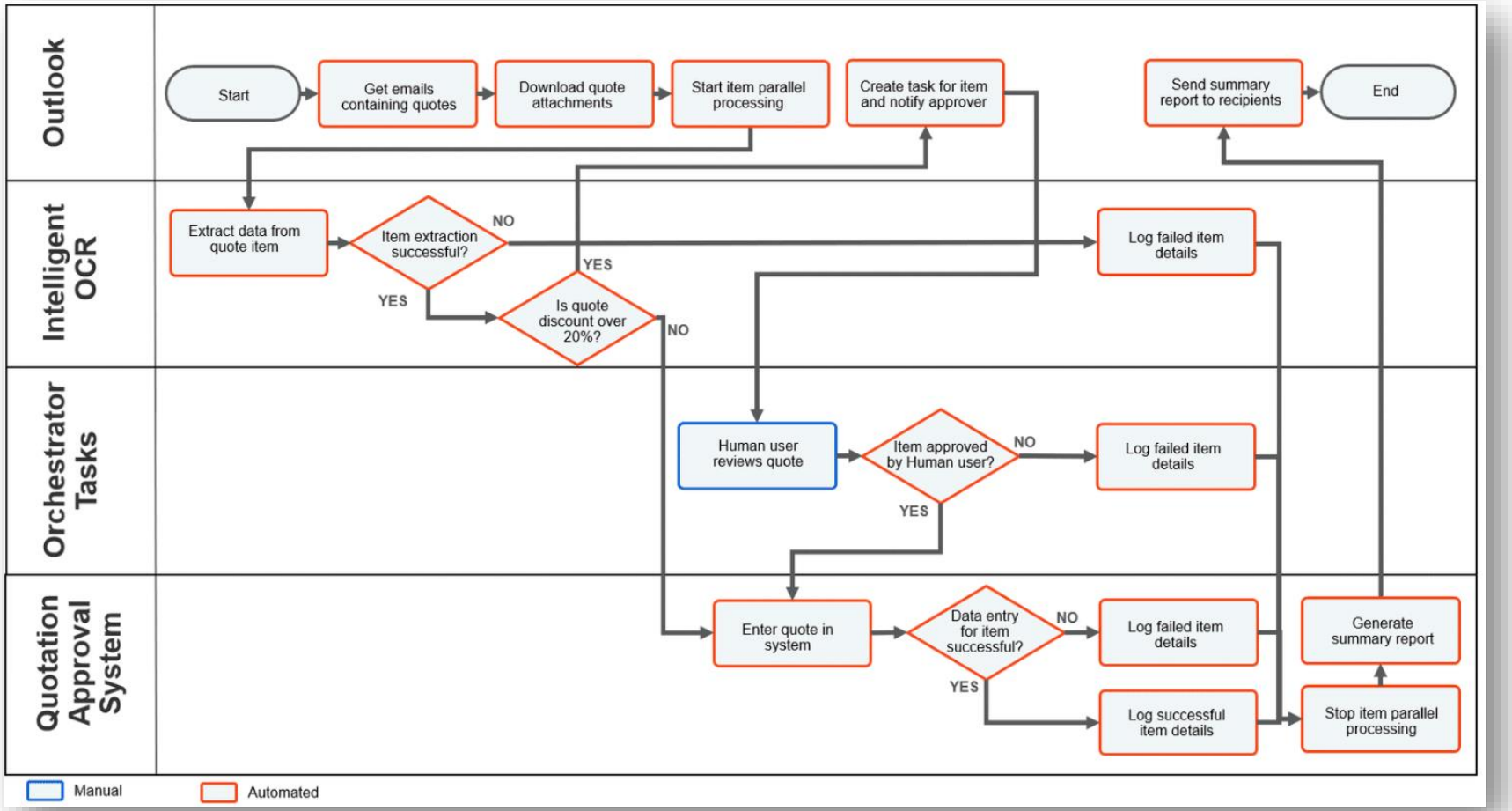
Action form open in Orchestrator

- Once an action is opened, the GUI configured in the project will pop-up and information will be submitted through the form.
- The button clicked by the user can be recorded.

Action Center, Human in the Loop and Orchestration Process

Process Example

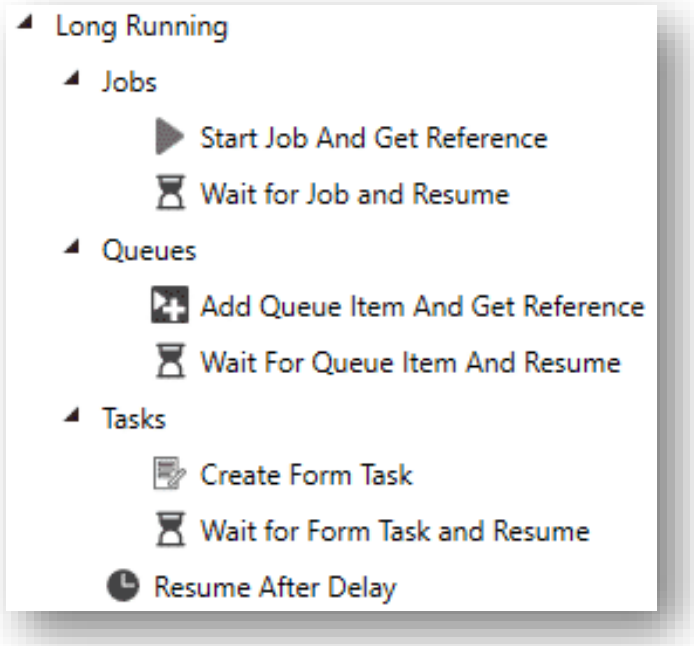
1. Check an Outlook mailbox and get emails containing quotes.
2. Download the quotes to a shared drive and save their paths.
3. Send the paths to an Intelligent OCR machine to extract the data.
4. If the discount value on a quote exceeds the 20% threshold, send the quote for human approval.
5. Send all reviewer approved or below threshold quotes to be entered into the approval system.
6. When the processing is done, generate a report with the results for each quote and email it to the stakeholders.



Action Center, Human in the Loop and Orchestration Process

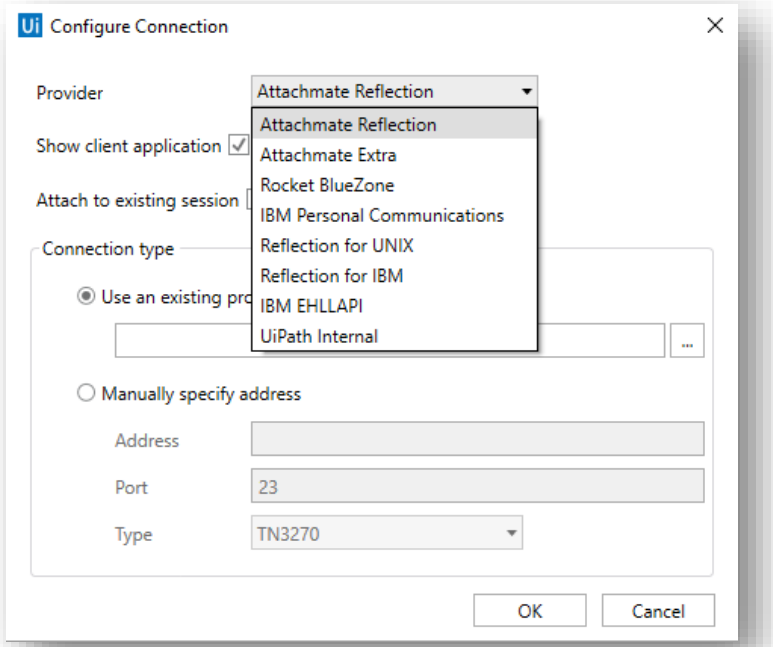
How to use:

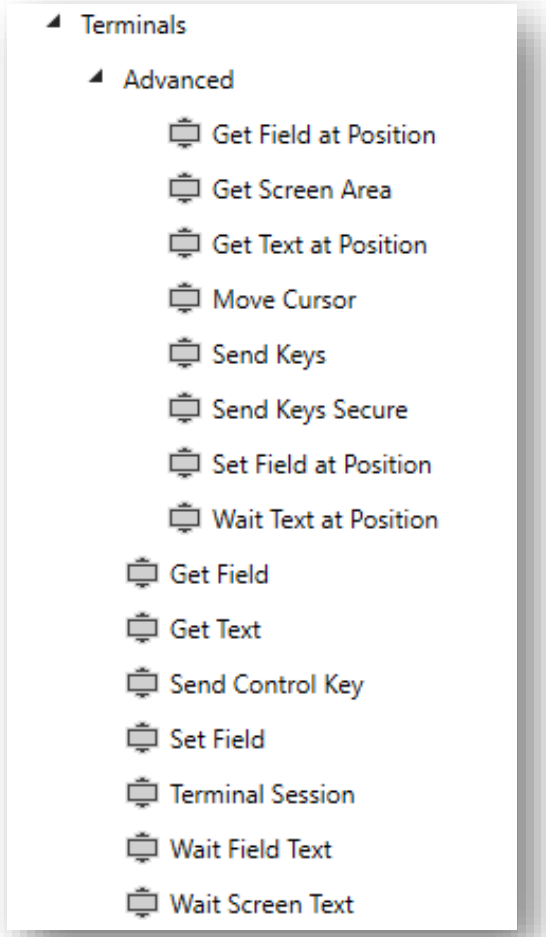
- Open **Orchestration Process** template in Studio; it contains the default UiPath.Persistence.Activities and UiPath.System.Activities dependencies.
- The activities can only be used in an Orchestration Process entry point, the file marked as Main.xaml.
- All variables used in the scope of a long-running activity must be **serializable**. Non-serializable data types can be used in separate sequence scopes in the same workflow or by invoking other files
- Persistence points (Wait and Resume activities) must not be used within the body of a **For Each** activity as they suspend the workflow after executing the first iteration and this might not be the desired behavior. Alternatively, try using loops with persistence activities, for example, the Parallel For Each activity.
- The **Delay** activity must not be used.



Terminals

- The UiPath.Terminal.Activities package allow the robot to automate terminals reliably.
- You can retrieve text, fields or screen positions, send keys, text, or wait for certain text or fields to appear as triggers.
- The first step is to use Terminal Session activity and choose a supported provider:



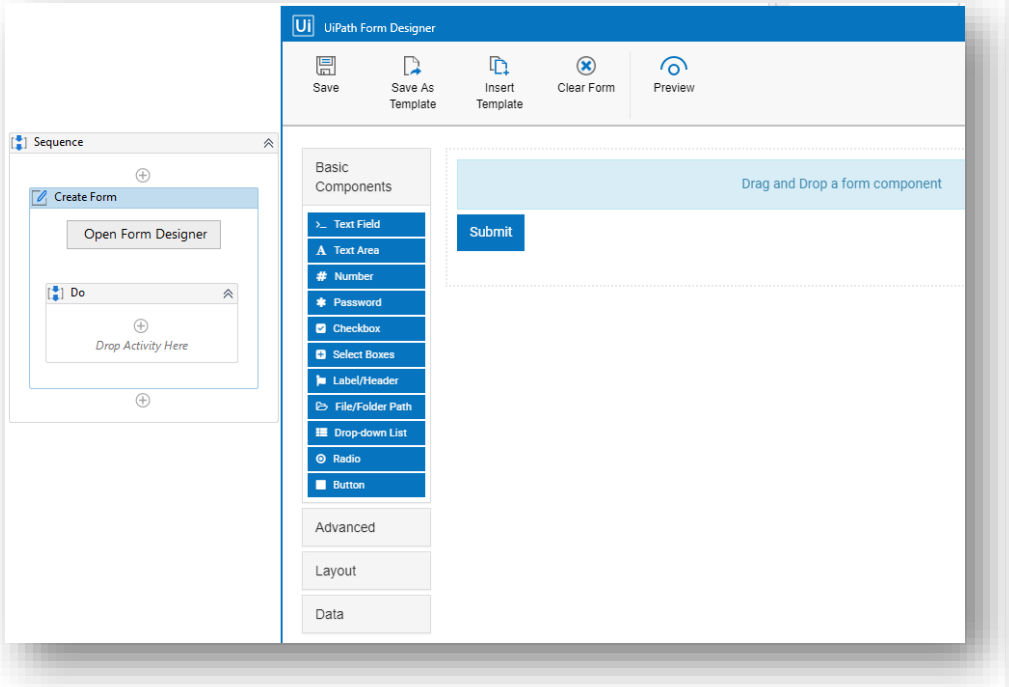


Attended Automation

- These are automations that run under human supervision, on attended robots.
- Agent Desktop replaced the old Robot Tray, for a better user experience.
- Useful activities for attended automations:
  - Forms
  - Callouts/tooltips
    - -> available with UiAutomation v.19.10 and above

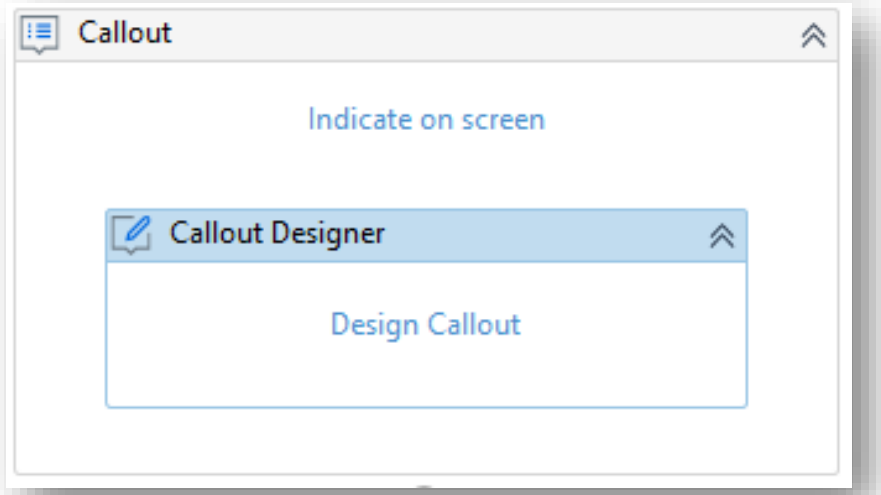
Attended Automation Activities

Forms



- UiPath.Form.Activities;
- Create customizable forms to capture user input.

Callout and Callout Designer



- The Callout Designer activity requires UiPath.Form.Activities
- The Callout activity is found in UiPath.UiAutomation.Activities

The UiAutomationNext Package

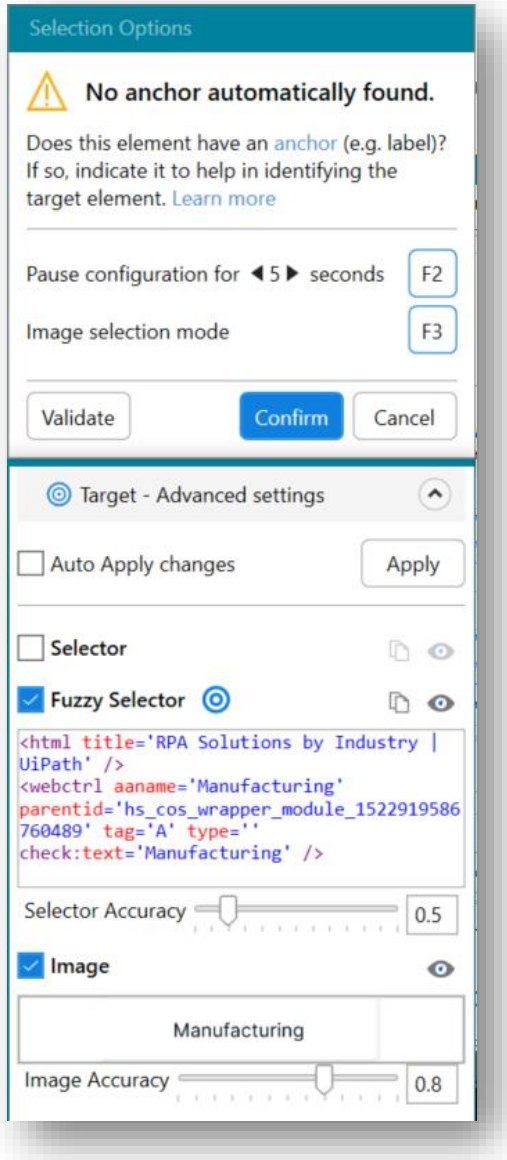
- UI Automation Next uses a combination of technologies for identifying and locating UI elements: **full selector**, **fuzzy selector**, and **image**, all wrapped in an **anchor-based algorithm**.
- After selecting the element to be identified, you can validate the selection. The performance of each method is displayed in the Advanced Settings section of the Selection Options window using one of the following icons:

The method was the first to successfully identify the element.

The method successfully identified the element.

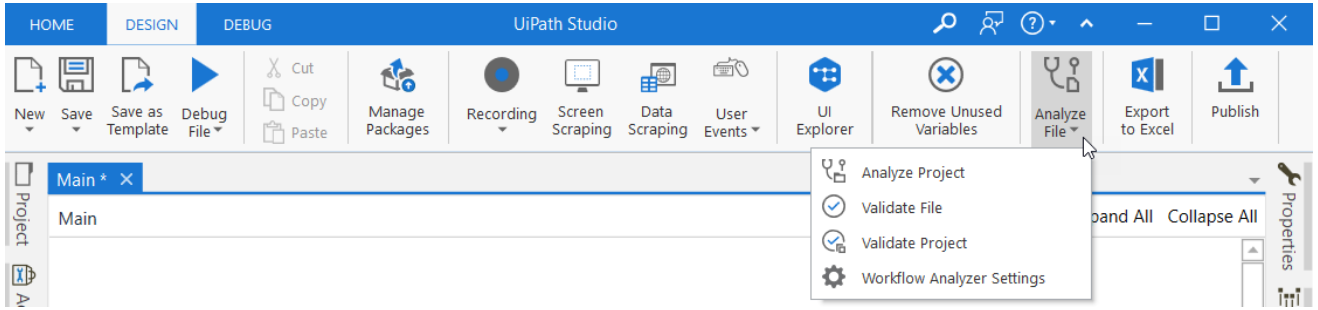
The method failed to identify the element because too many duplicates were found.

The method failed to identify the element.

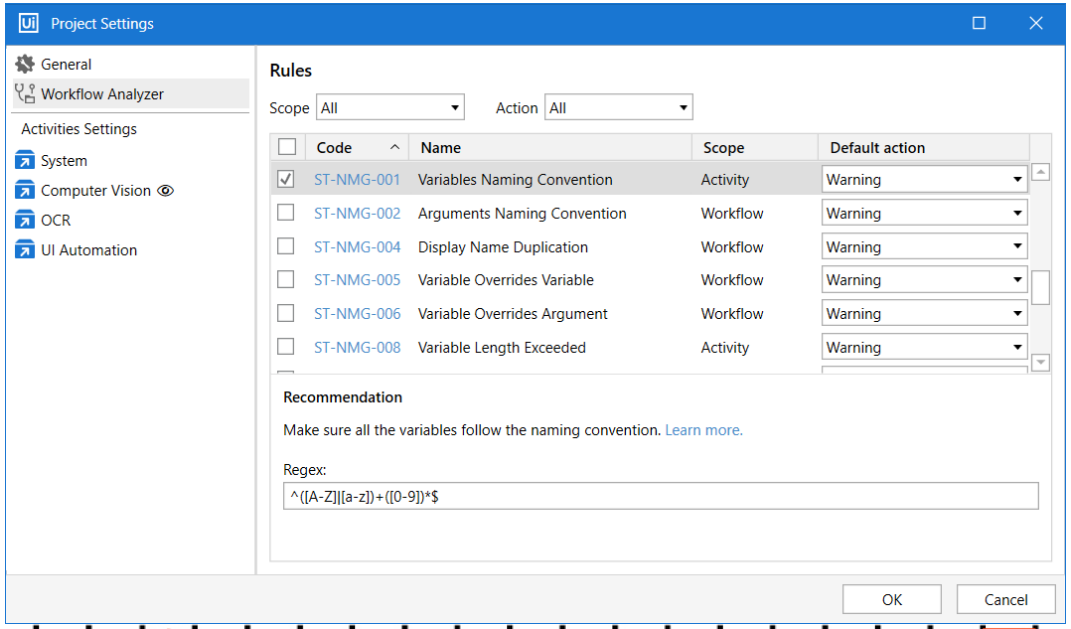


Workflow Analyzer

- Workflow Analyzer is a static code analyzer that ensures your project meets high quality and reliability standards. A static code analyzer checks for inconsistencies without actually executing the project, as opposed to dynamic analyzers which step in during execution.



- Workflow Analyzer uses a set of rules to check for various inconsistencies unrelated to project execution. The rules are based on Automation Best Practices and take into consideration variable and argument naming, empty sequences or workflows, package restrictions, and so on.
- The analyzer does not identify errors in execution or compilation.



Workflow Analyzer - building Custom Rules

Add custom Rules to Studio

- Custom Workflow Analyzer rules can be integrated in Studio in two ways:
  - at global level by adding the external assembly (.dll) in the Studio install location
  - at project level by installing the custom activities pack.

At Global Level

To make custom rules available for all projects created in your instance of Studio, add the external assembly (.dll) package to the install location.

Follow the steps detailed in the **Creating a Custom Activity presented before** to export the code as a .dll file.

The exported file must now be added to Studio install location:

•.msi C:\Program Files (x86)\UiPath\Studio\Rules

•.exe %localappdata%\UiPath\app-%version%\Rules

At Project Level

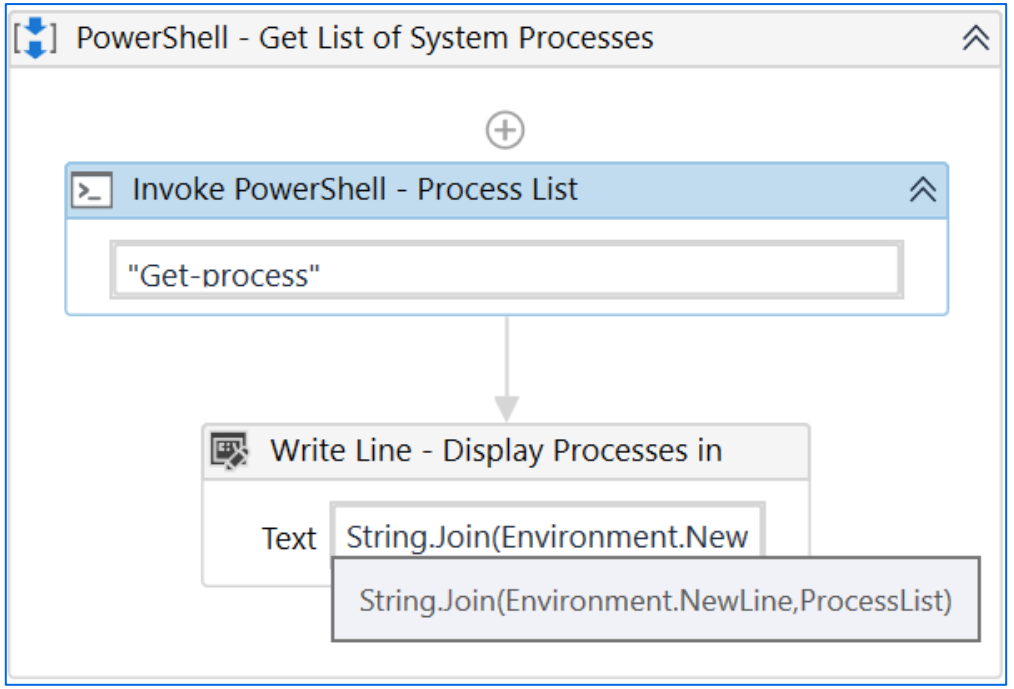
To make custom rules available only for a certain project, create a NuGet package (.nupkg) and install it in a Studio project as a dependency.

Introduction to Invoke PowerShell Activity

Windows PowerShell is a task automation and configuration management framework by Microsoft.

Invoke PowerShell activity:

- Executes PowerShell command with specified input parameters
- Can run one-liner command from within the activity or multiple lines of commands using a script file



Generate a list of current system processes and display the output

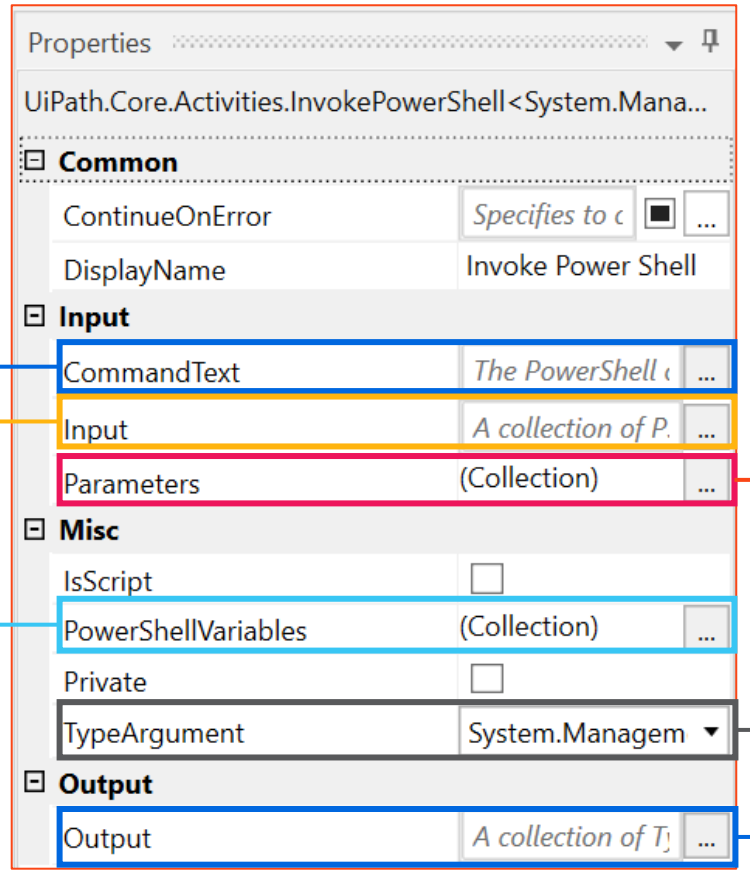
Properties of Invoke PowerShell activity

The properties of Invoke PowerShell activity are:

CommandText
Passes the PowerShell commands to be executed

Input
Passes the collection of PSObjects to the command

PowerShellVariables
Passes variables to be used in the current session of the command



Parameters
Enters a dictionary of PowerShell command parameters

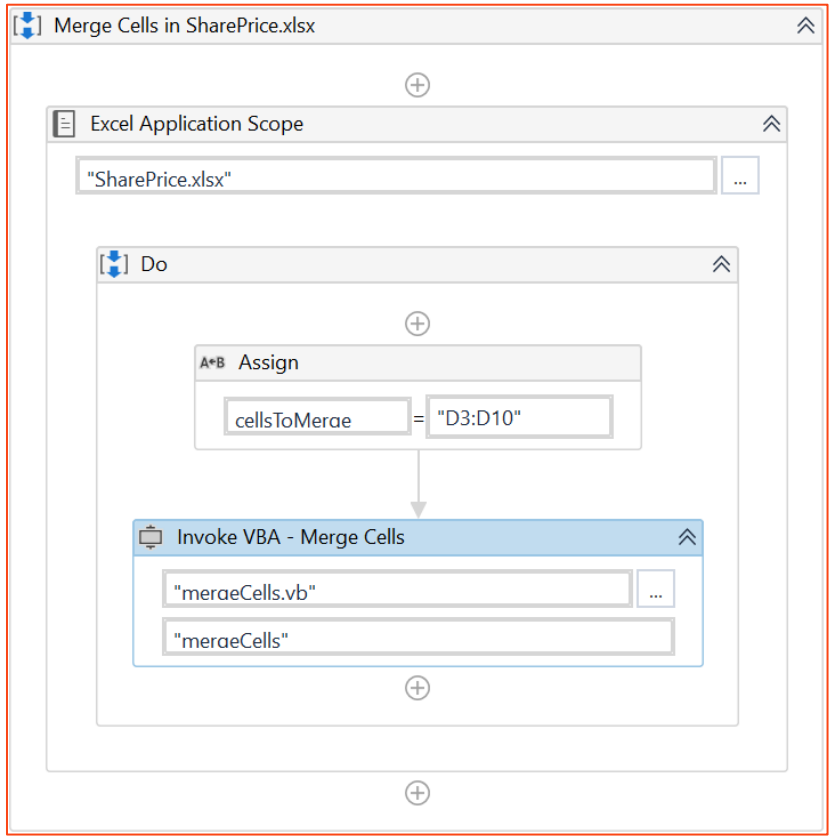
TypeArgument
Sets the type of the output variable

Output
Stores the TypeArgument object returned after command execution

Introduction to Invoke VBA Activity

Invoke VBA activity is used to invoke macro from an external file containing VBA codes and run against an Excel file.

- Used for Excel files with .xls or .xlsx extensions
- Used only within an Excel Application Scope activity

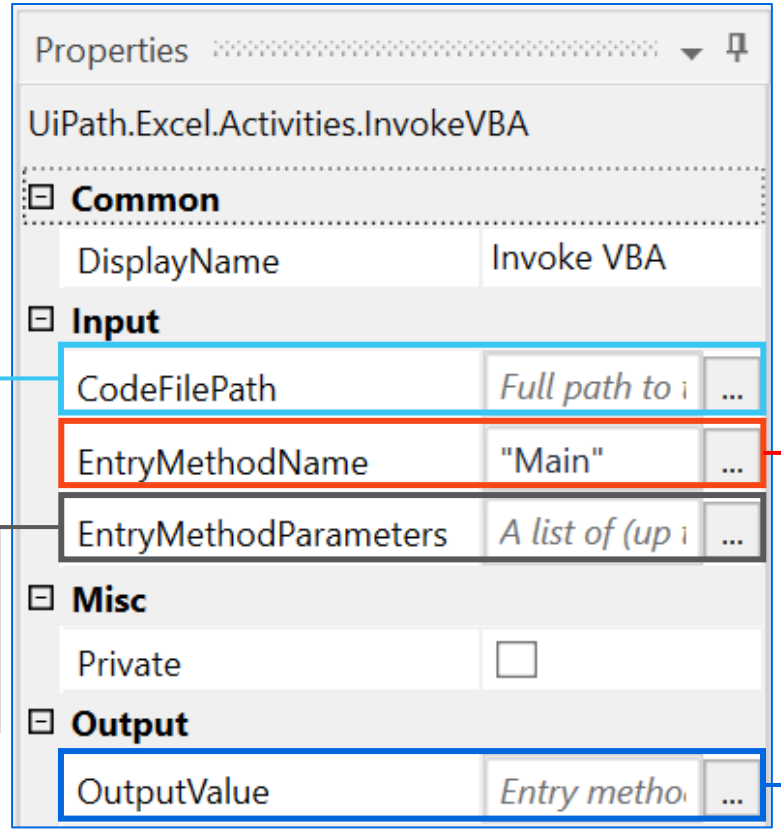


Merge the cells D3 to D10 of SharePrice.xlsx using Invoke VBA activity

Properties of Invoke VBA Activity

CodeFilePath
Enters the path to the macro file containing VBA Sub/Function definitions

EntryMethodParameters
Enters the list of up to 30 parameters that can be passed to the entry method



EntryMethodName
Enters the Sub/Function name that is to be invoked

OutputValue
Stores the value returned by the execution of the invoked code

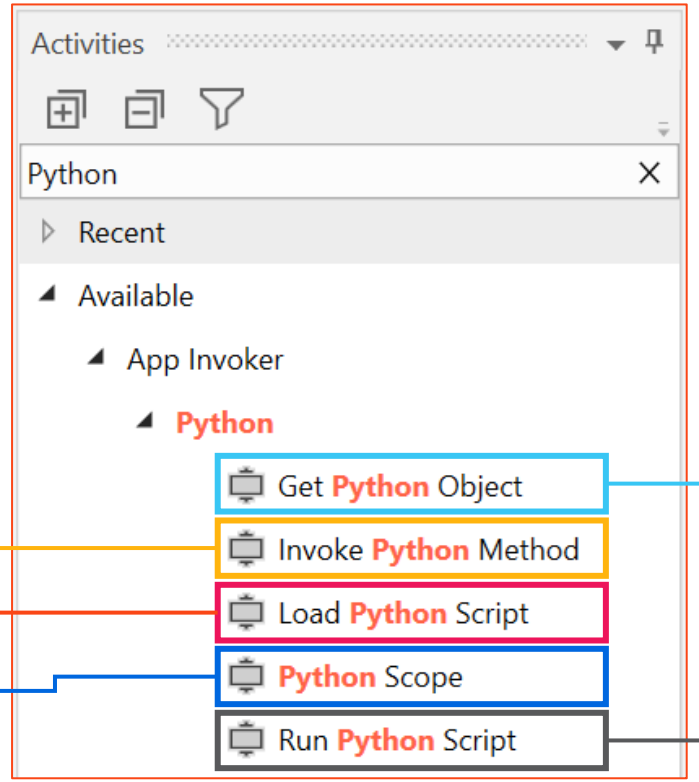
Introduction to Python Activities Package

Python activities package must be installed in Studio to run the Python codes. The activities available in the package are:

Invoke Python Method
Runs a specified method from a Python script

Load Python Script
Stores the Python script handlers in a Python object

Python Scope
Provides the scope for Python activities and initializes the specified Python environment



Get Python Object
Converts PythonObject variable to a .NET datatype

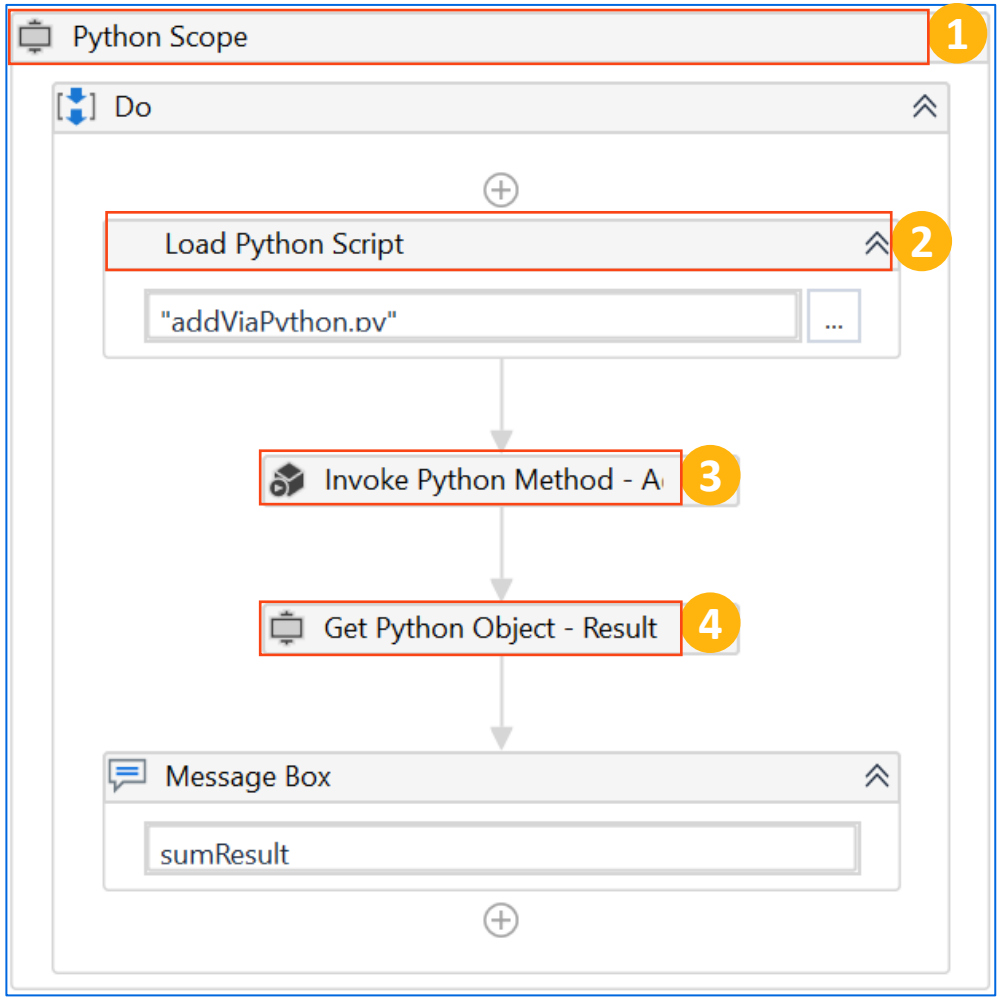
Run Python Script
Executes the Python code directly or via a file

Python Activities in a Workflow

Steps:

1. **Python Scope** activity initializes the Python environment
2. **Load Python Script** activity loads the addViaPython.py file containing Python codes
3. **Invoke Python Method** activity invokes AddTwoNumbers method from the script
4. **Get Python Object** activity converts the output from Invoke Python Method into a string variable and stores in sumResult variable

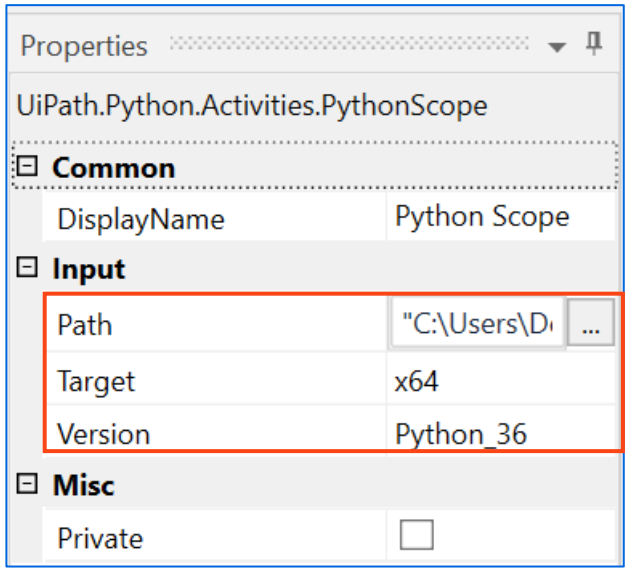
The value stored in sumResult variable is displayed in a message box.



Properties of Python Activities

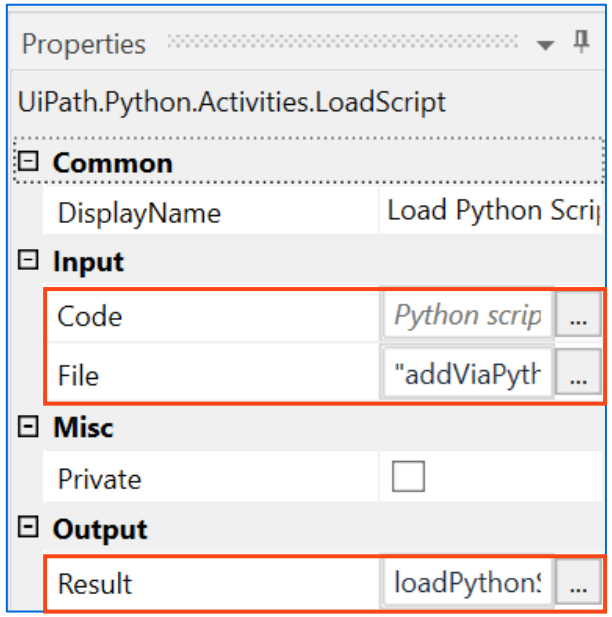
Python Scope Activity

- **Path:** Inserts the path of the Python.exe application
- **Target:** Chooses Python 32- or 64-bit versions
- **Version:** Chooses the Python versions



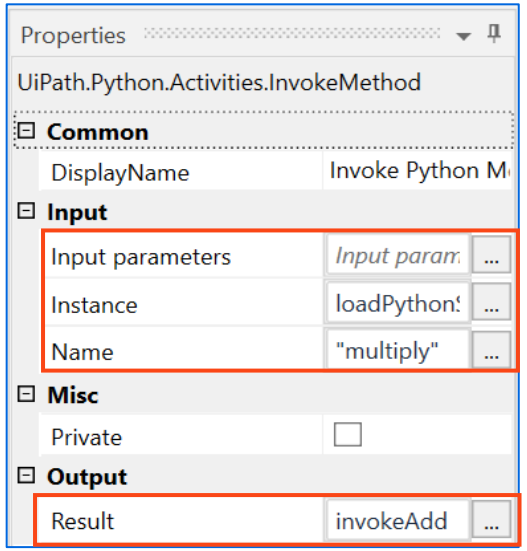
Load Python Script Activity

- **Code:** Inserts Python codes to be executed
- **File:** Choose the Python file to be executed
- **Result:** Store the output in a variable



Invoke Python Method Activity

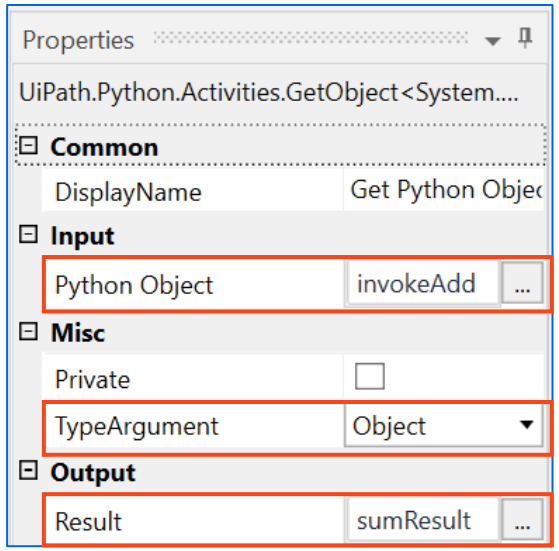
- **Input parameters:** Insert parameters to the script
- **Instance:** Insert the output of Load Python Script activity
- **Name:** Insert method names from the script
- **Result:** Store the output in a variable



Properties of Python Activities (Contd.)

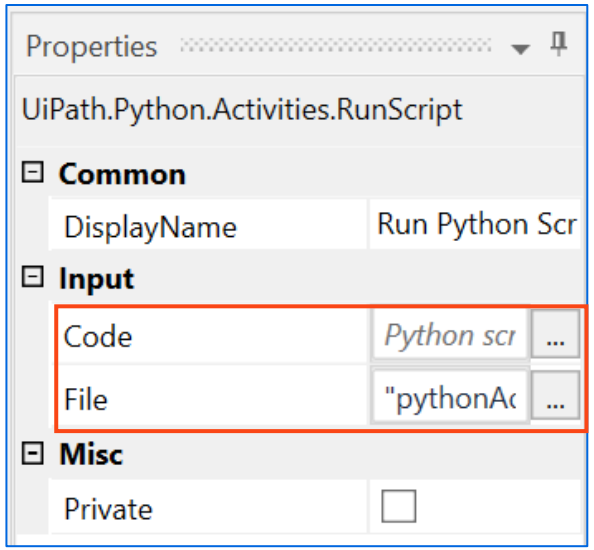
Get Python Object Activity

- **Python Object:** Insert the result of Invoke Python Method activity
- **TypeArgument:** Choose .NET datatype to which Python object is to be converted
- **Result**: Store the output of the activity in a variable



Run Python Script Activity

- **Code:** Run the Python code
- **File:** Input the file path of the Python code to run



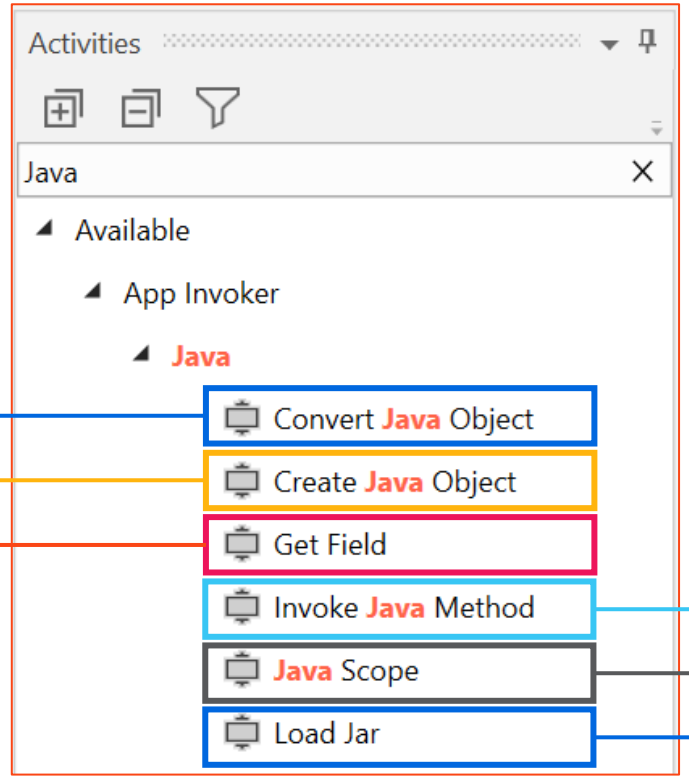
Java Activities Package

Java activities package must be installed in Studio to run Java codes. Adding the package will show the available Java activities in the Activities panel.

Convert Java Object
Converts JavaObject into .NET datatype

Create Java Object
Creates a JavaObject variable from a class

Get Field
Returns a public field from a specified JavaObject variable



Invoke Java Method
Invokes a specified Java method

Java Scope
Initializes a specified Java library and provide a scope for Java Activities

Load Jar
Loads a Java Library Package in a Java Scope activity

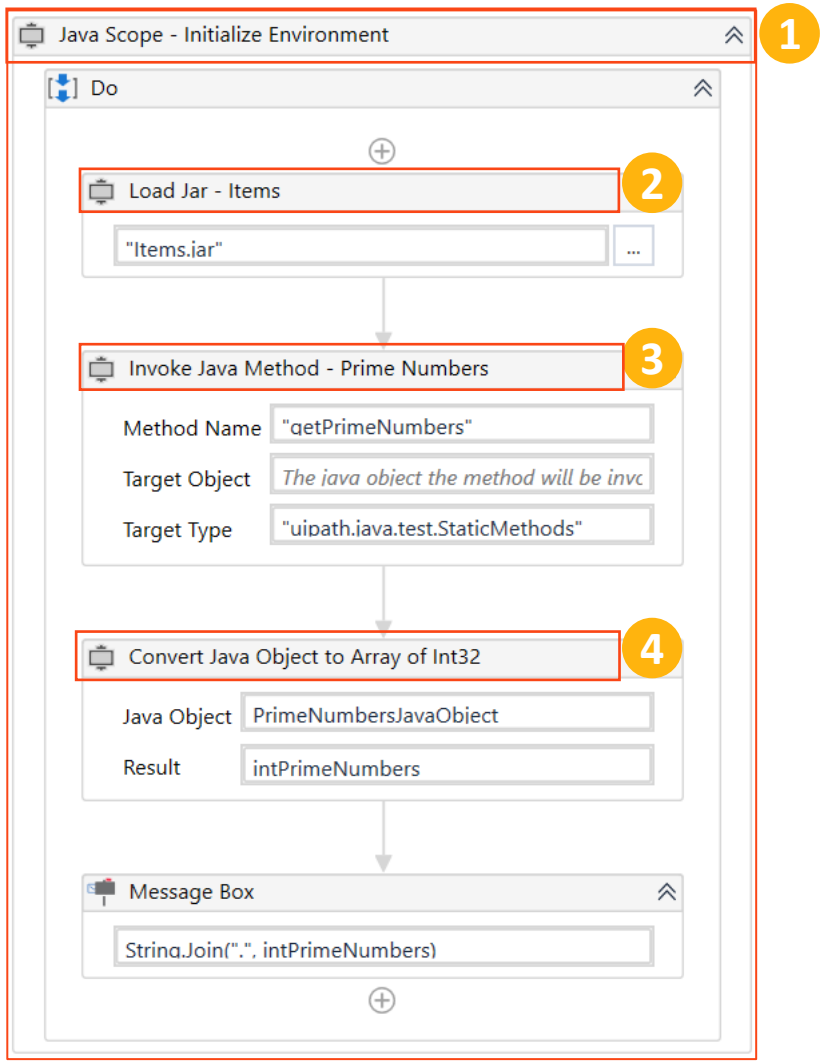
Java Activities in a Workflow

The use of Java activities within a workflow:

Steps:

1. **Java Scope** activity initializes a Java Library
2. **Load Jar** activity loads items.jar file containing the Java Code to generate prime numbers
3. **Invoke Java Method** activity runs a static method, getPrimeNumbers, and store the output in PrimeNumbersJavaObject
4. **Convert Java Object** activity converts the output from JavaObject to a .NET array of integers named intPrimeNumbers

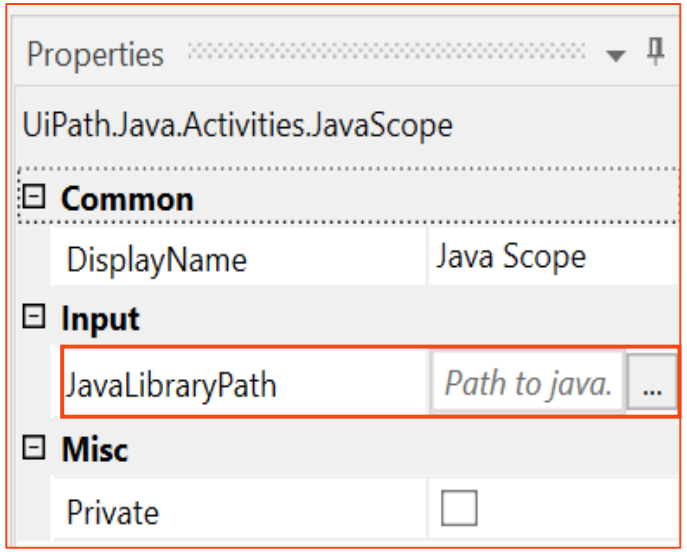
The array, intPrimeNumbers, is displayed as a string.



Properties of Java Activities

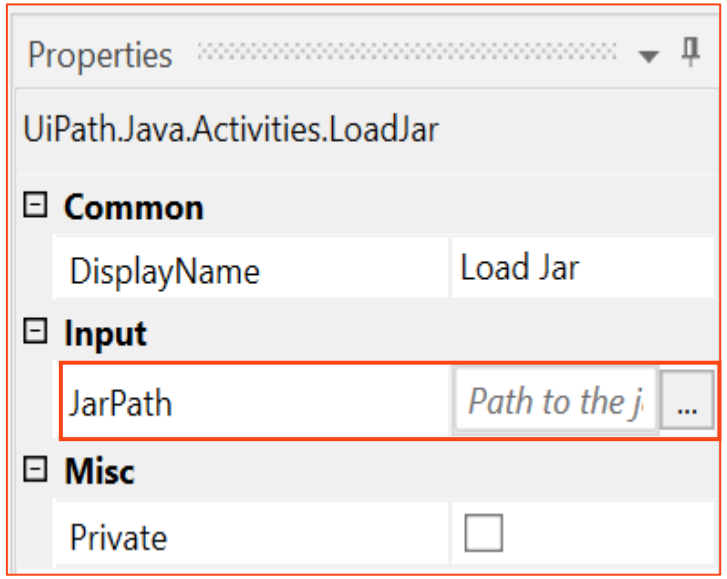
Java Scope Activity

- **JavaLibraryPath:** Enter the path to the Java executable java.exe file



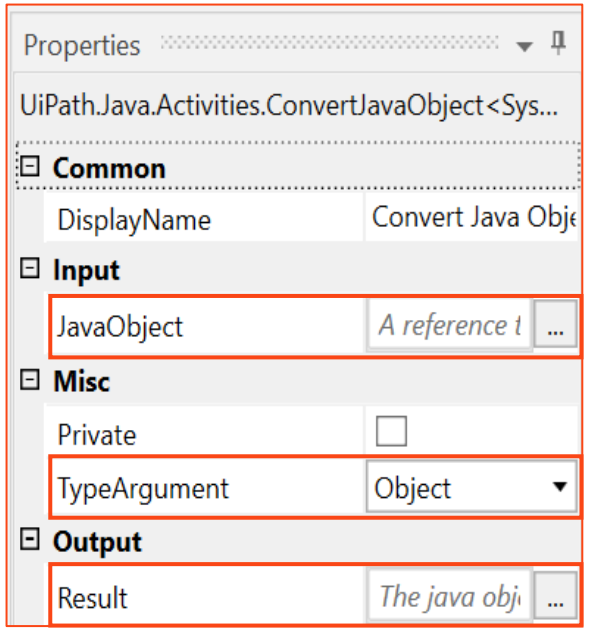
Load Jar Activity

- **JarPath:** Enter the path to the .jar file containing the Java Library Package



Convert Java Object Activity

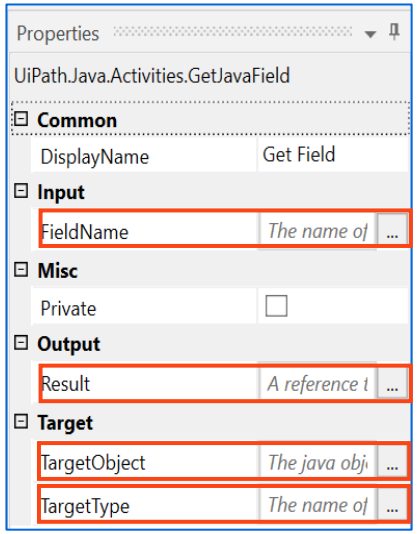
- **JavaObject:** Enter the JavaObject variable to be converted to .NET data type
- **TypeArgument:** Select .NET data type
- **Result:** Store the converted JavaObject stored in a .NET variable



Properties of Java Activities (Contd.)

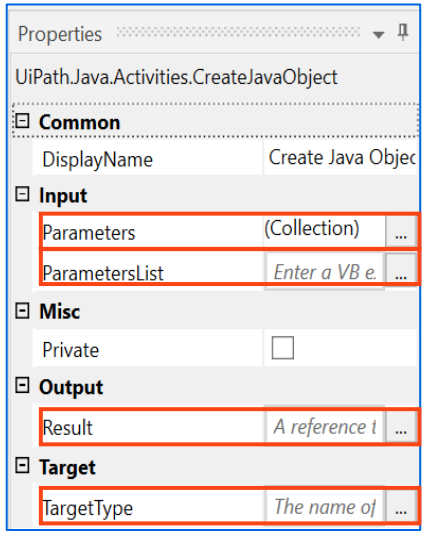
Get Field Activity

- **FieldName:** Return name of the field as JavaObject variable
- **Result:** Store Java field in a JavaObject variable
- **TargetObject:** Retrieve JavaObject containing the field
- **TargetType:** Use the Java class name to create JavaObject



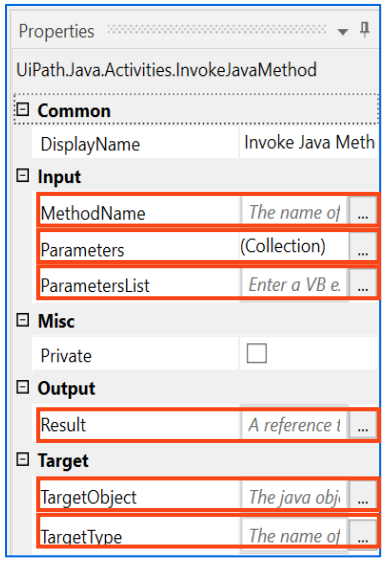
Create Java Object Activity

- **Parameters:** Pass arguments to Java method
- **ParametersList:** List of parameters to pass to a Java method
- **Result:** Store output in JavaObject variable
- **TargetType:** Use the Java class name to create JavaObject



Invoke Java Method Activity

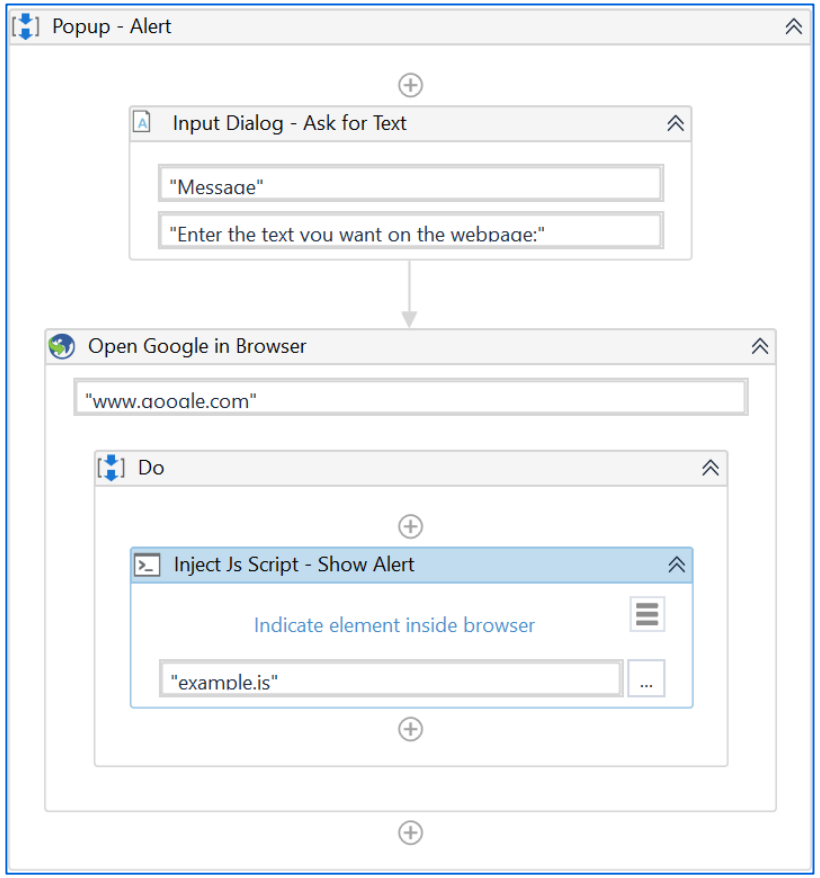
- **MethodName:** Pass name of the method to be invoked
- **Parameters:** Pass arguments to a Java method
- **TargetObject:** Enter name of Java Package on which to execute Java method
- **TargetType:** Enter Java package name class



JavaScript Activity

JavaScript programming language is a client-side scripting language which is commonly used to make dynamic webpages.

Inject JS Script
Activity used to execute the JavaScript codes



Run the code in example.js file to display a popup in a browser

Properties of Inject JS Script Activity

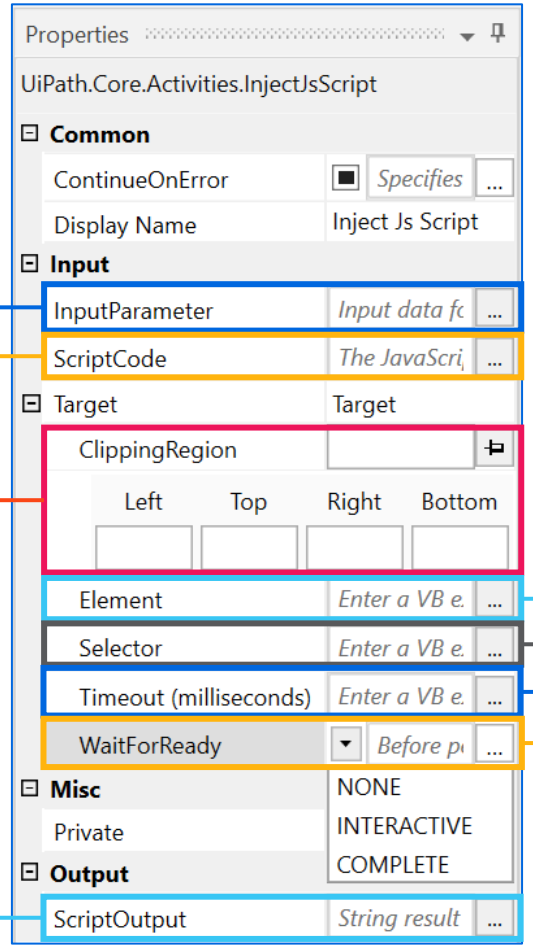
The properties of Inject JS Script activity are:

InputParameter
Inputs the data for JavaScript code

ScriptCode
Runs the JavaScript code as a string or full path to a .js file

ClippingRegion
Defines the clipping rectangle in pixels

ScriptOutput
Returns the string result from JavaScript code



Element
Returns the UiElement variable by another activity

Selector
Uses text property to find a UI element

Timeout (milliseconds)
Specifies the amount of time to wait for the activity to run

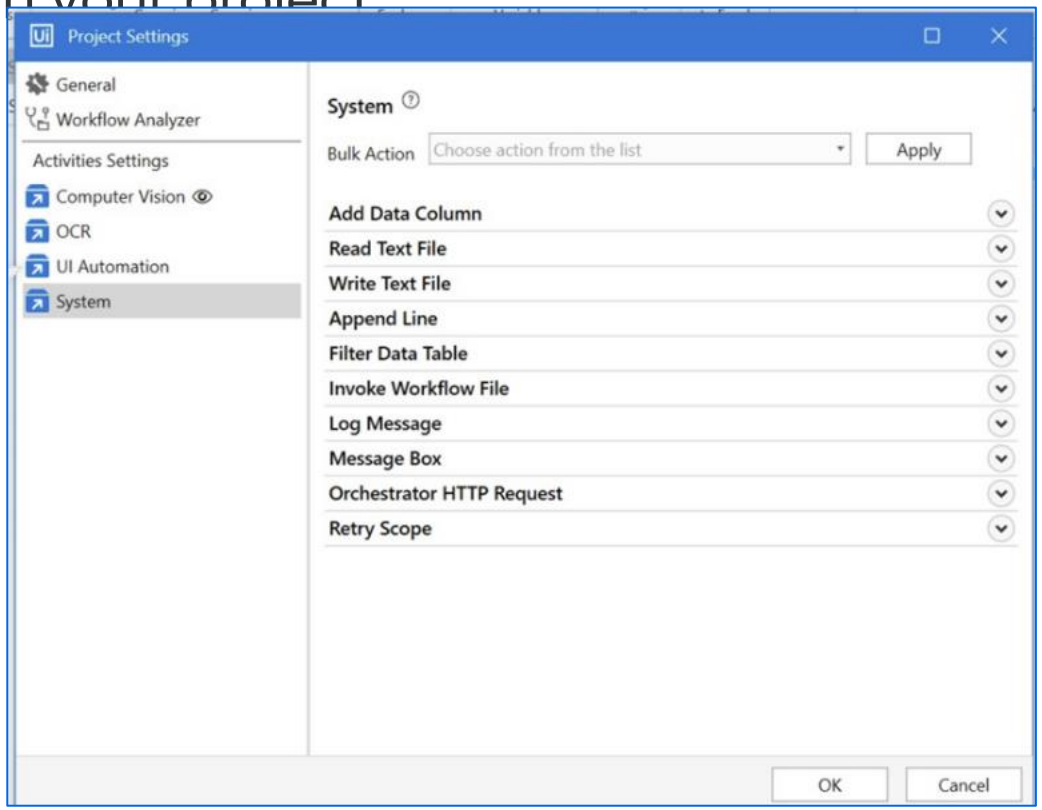
WaitForReady
Waits for the target to become ready

Activity Project Settings

Activity Project Setting are property changes that can be defined at project level and applied to all activities part of project dependencies. For example, you can change the value for the DelayBefore property in Project Settings for all UiAutomation activities that have this property in your project.

Settings can be configured for:

- Computer Vision
- OCR activities
- UI Automation activities
- System activities



Thank you

Trainer Name