Introduction:

Since RPA aims to automate tasks performed by humans, you can expect UI elements and UI interactions to be extensively used. UI elements are elements that we identify in a process (like a completion icon or a status bar), while UI interactions cover everything that we do to interact with a system at UI interface level.

UI interactions can be split into input actions and output actions. Input actions are those actions through which we send data to an application (by clicking or typing, for example) to produce an outcome. On the other hand, output actions are those actions through which we take out data from an application.

UI automation offers a framework of automating human actions using the user interface. However, there are many applications and types of files (such as Excel) with which workflows in UiPath can be built by using activity packages that work at API level or by using data manipulation specific techniques. These are desirable vs generic UI automation, as they are easier built and more reliable.

1. Input Actions and Methods

Overview:

Every time we insert data into an application, or we send a command to a system to produce a change (or a continuation), we perform an input action. In UiPath, the main input actions are Click, Type into, Send Hotkey and Hover. These are also the main actions that a human user would perform to input data in an application.

Input Methods:

Input methods provide the input actions with the means to input data. Each input action lets the user switch between the 3 methods: 'Simulate Type/Click/Hover', 'SendWindowMessages' and default. 'Simulate Type/Click/Hover' and 'SendWindowMessages' can be chosen by checking the corresponding box from the Properties panel; if none of the two boxes is checked, the default input method is used.

I. Default

II. SendWindowMessages

III. Simulate Type/Click

I. Default:

How does it work?

- Clicks: the mouse cursor moves across the screen
- Typing: the keyboard driver is used to type individual characters

What are the implications?

- Attended User cannot touch the mouse or keyboard during the automation
- It has a lower speed and load times can impact accuracy

What are the strong points?

- Supports special keys like Enter, Tab, and other hotkeys
- 100% compatibility

What are the limitations?

- Does not automatically erase previously written text
- Does not work in the background

II. Send Window Messages:

How does it work?

- Replays the window messages that the target application receives when the mouse/keyboard is used
- Clicking and typing occur instantly

What are the implications?

- Works in the background
- Comparable to the Default method in terms of speed

What are the strong points?

- Supports special keys like Enter, Tab, and other hotkeys
- Users can work on other activities during the execution of the automated processes

What are the limitations?

- Does not automatically erase previously written text
- Works only with applications that respond to Window Messages

III. Simulate:

How does it work?

- Uses the technology of the target application (the API level) to send instructions
- Clicking and typing occur instantly

What are the implications?

- Works in the background

- Actions are a lot faster, but there are some compatibility limitations

What are the strong points?

- Can automatically erase previously written text
- Users can work on other activities during the execution of the automated processes

What are the limitations?

- Does not support special keys like Enter, Tab, and other hotkeys
- It has a lower compatibility than the other 2 methods

S.No.	Capability Method	Compatibility	Background Execution	Speed	Hotkey Support	Auto Empty Field
1	**Default**	100%	No	50%	Yes	No
2	**SendWindowMessages**	80%	Yes	50%	Yes	No
3	**Simulate Type/Click**	99% - Web Apps 66% - Desktop Apps	Yes	100%	No	Yes

Input Actions:

Click, **Type Into** and **Send Hotkey** are simple in terms of what they primarily do. All the input actions share several properties:

Delay: Can be used to set a delay before or after the click;

WaitForReady: Can be configured to wait for the target to become ready by verifying certain application tags.

What is maybe more important to see is the options that each of them offer:

Input Actions	Properties
Click	ClickType: Can be single or double (changing the default to double makes it very similar to a separate activity – Double Click); MouseButton: Can be configured to left, middle or right button; Timeout: specifies the duration in which the activity is retried until an error is thrown, KeyModifiers: Can press the Alt, Ctrl, Shift and/or the Win key while performing the action. *You can find out more about the Click activity and its properties here:* **https://docs.uipath.com/activities/docs/click**

Type Into	Activate: The field in which the text is typed is brought to the foreground and activated, before the typing; ClickBeforeTyping: The UI element in which the text will be typed into will be clicked on before typing; DelayBetweenKeys: The delay between each key is typed; EmptyField: The UI element will be emptied before typing. You can find out more about the Type into activity and its properties here: **https://docs.uipath.com/activities/docs/type-into** For typing into protected fields, such as 'Password', there is a more suitable activity – **Type Secure Text**. You can find out more about it and its properties here: **https://activities.uipath.com/docs/type-secure-tex**t
Send Hotkey	Activate: The field in which the text is typed is brought to the foreground and activated, before the typing; ClickBeforeTyping: The UI element in which the text will be typed into will be clicked on before typing; DelayBetweenKeys: the delay between each key is typed; EmptyField: the UI element will be emptied before typing. You can find out more about the Send Hotkey activity and its properties here: **https://activities.uipath.com/docs/send-hotkey**

Best Practices - Input Actions & Methods:

Press F2 to pause the UiExplorer for 3 seconds. This is very handy, for example, when you need to indicate on the screen an element that appears on hover;

Use Send Hotkey activities only inside containers to avoid sending them to unintended places.

2. Output Actions and Methods

Overview:

Output actions are used in UiPath either to extract data (in general, as text) from a UI element. Output methods are what enables output actions to extract data from UI elements. Let's discuss these first.

Output Methods:

Output Method	Description
FullText	The FullText method is the default method and good enough in most cases. It is the fastest, it has 100% accuracy and can work in the background. Moreover, it can extract hidden text (for example, the options in a drop-down list). On the other hand, it doesn't support virtual environments and it doesn't capture text position and formatting. This method offers the option to ignore the hidden message and capture only the visible text.
Native	The Native method is compatible with applications that use Graphics Design Interface (GDI), the Microsoft API used for representing graphical objects. It can extract the text position and formatting (including text color) and 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 just like FullText, it doesn't support virtual environments. By default, it can process all known characters as separators (comma, space, and so on), but when only certain separators are specified, it can ignore all the others.
OCR	OCR (or Optical Character Recognition) is the only output method that works with virtual environments and with "reading" text from images. Its technology relies on recognizing each character and its position. On the other hand, 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. Just like the Native method, it also captures the text position. The OCR method has two default engines that can be used alternatively - Google Tesseract and Microsoft MODI. There are additional OCR engines that can be installed free of charge (such as Omnipage and Abbyy Embedded) or paid (IntelligentOCR offered by Abbyy).

S.No.	Capability Method	Speed	Accuracy	Background Execution	Extract Text Position	Extract HiddenText	Support For Citrix

1	FullText	10-Jan	100%	Yes	No	Yes	No
2	Native	08-Jan	100%	No	Yes	No	No
3	OCR	03-Jan	98%	No	Yes	No	Yes

You can switch between all these methods by accessing the **Screen Scraping** wizard, once you indicate an area to extract the text from and the Preview window is launched.

More about Output Methods:

You can see some examples and find out more details in the UiPath Documentation Guide.

https://docs.uipath.com/studio/docs/examples-of-using-output-or-screen-scraping-methods

Output Actions:

Below are the most important output actions. As a difference from the Input Actions, where all the Input Methods are available for each Input Action, the Output Actions are somewhat matched with the Output Methods, as you will see.

Output Action	Description
Get Text	Extracts a text value from a specified UI element. Find out more about Get Text here.
Get Full Text	Extracts a string and its information from an indicated UI element using the FullText screen scraping method. Thus, the hidden text is also captured by default (although it provides the option to ignore hidden text). This activity is automatically generated when performing screen scraping with the FullText method, along with a container. Find out more about Get Full Text here.
Get Visible Text	Extracts a string and its information from an indicated UI element using the Native screen scraping method. This activity is automatically generated when performing screen scraping with the Native method chosen, along with a container. Find out more about Get Visible Text here.

Get OCR Text	Extracts a string and its information from an indicated UI element using the OCR screen scraping method. This activity can also be automatically generated when performing screen scraping, along with a container. By default, the Tesseract OCR engine is used. Find out more about Get OCR Text here.
Data Scrapping Wizard	Data scraping is a functionality of UiPath Studio that allows the extraction of structured information from an application, browser or document to a DataTable variable. The functionality can be accessed directly from the Design ribbon of UiPath Studio, the 'Data Scraping' button. The first element chosen is used to populate the first column, and the option to extract the URL (where these exist) is also presented. The user may change the order of the columns and specify the maximum number of entries to be extracted (the default is 100 and leaving 0 means extracting all the results). In the Preview stage, the 'Extract Correlated Data' option can be used to extract other fields of data, by indicating the first and the second entry, just like for the main field. Find out more about Data Scraping here.
Extract Attributes	This is actually a category of activities that can be used when you don't want to extract the text out of the UI element, but maybe the color, the position or an ancestor. There are 3 different UiPath activities to do this: • **Get Ancestor**: UI elements are in parents-children structures (a text document has the Notepad app as a parent, who has the category of apps as a parent, and so on). Get Ancestor retrieves the ancestor (or the parent) of an UI element. • **Get Attribute**: UI elements have plenty of attributes. Think of a button on a website – it definitely has a color, a name, a state, and so on. Get Attribute allows the user to indicate an attribute, and the activity retrieves the value of that specific attribute. • **Get Position**: this activity retrieves the actual position on the screen of a specific element. This can be very useful when there are many similar elements on a screen; without having their actual position, it would become very difficult to identify each of them. Find out more about Get Ancestor here. From there you can navigate from the menu on the left to read about Get Attributes and Get Position.

3. Working With UI Elements

Input and Output Actions can be viewed as a succession of 2 micro-steps: first, recognizing UI elements, and then inputting (or extracting) data. In many cases, such as most of the examples that we covered in the previous chapters, the first step is easily sorted out either by including it in the activity, or by using the delay option. This is why, in many cases, it's not at all necessary to have a separate activity for locating an UI element (like a Find Element activity) before an actual Click or Type activity.

In some business scenarios however, building reliable workflows means accommodating situations in which UI elements move or become visible at only at various times. Moreover, some solutions that seem handy, like setting up a big Delay before executing a Click activity, may trigger a significant increase in the overall duration of an automation (if, for example, there are 50 Click activities).

There are several actions that are meant to locate UI elements and it's important to know when each of them is useful:

Action	Description
Find Element	Waits for the specified UI element to appear on the screen (to be in the foreground) and returns it as a UiElement variable. This is useful, for example, when a certain action needs to be performed on the UI Element found. Learn more about Find Element here.
Element Exists	Enables you to verify if a UI element exists, even if it is not visible. It returns a Boolean variable, which makes it very useful in 'If statement' activities, for example. Learn more about Element Exists here.
Wait Element Vanish	Waits for the specified UI element to disappear from the screen. It's an alternative to Find Element, for example when the disappearance of an element (a loading sign) is more reliable then the appearance of another element. Learn more about Wait Element Vanish here.
On Element Appear	A container that waits for a UI element to appear and enables you to perform multiple actions within it. Learn more about On Element Appear here.

On Element Vanish	A container that enables you to perform one or multiple actions after a specified UI element vanishes. Learn more about On Element Vanish here.
Text Exists	Checks if a text is found in a given UI element. There's an alternative version of this that uses OCR technology to check for a given UI element. This is useful when UI elements are not accessible otherwise than as images. Learn more about Text Exists here.

Most of the activities presented above that involve UI elements have versions that involve finding and matching images uploaded by the user. They even present a matching index.