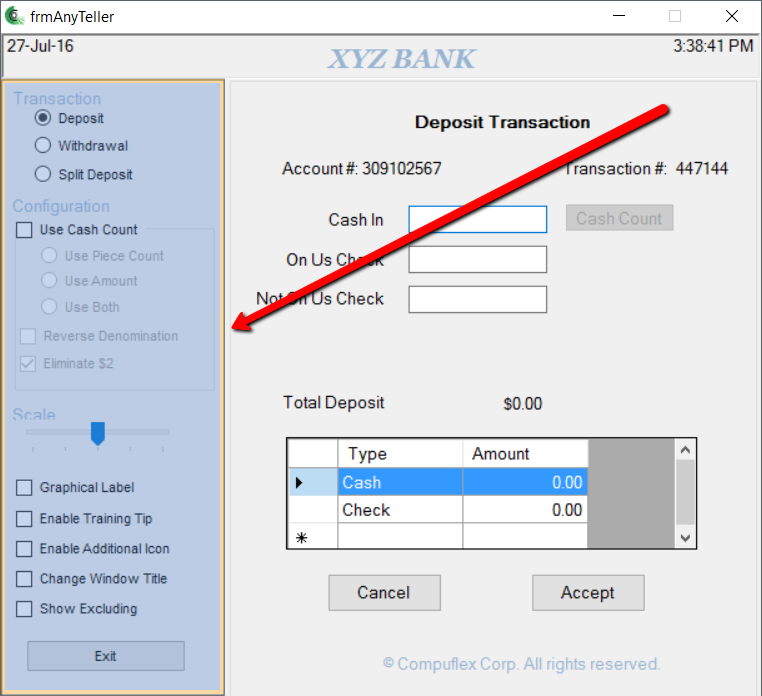
RECORDING

About Recording

Recording is an important part of UiPath Studio, that can help you save a lot of time when automating your business processes. This functionality enables you to easily capture a user’s actions on the screen and translates them into sequences.

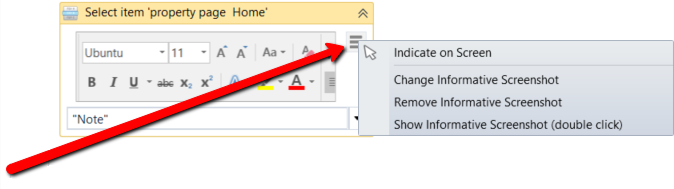
These workflows can be modified and parametrized so that you can easily replay and reuse them in as many other processes as you need.

All user interface elements are highlighted while you record, as you can see in the following screenshot, so that you can be sure the correct buttons, fields or menus are selected.



Interactions with UI elements yield informative screenshots in the workflow. These can be changed, hidden, removed or shown in full size by selecting the respective action from the Options menu.

All screenshots are automatically saved as .png files in the same location as your project, in a separate folder named “.screenshot.“ By default, the path is: C:\Users\your\_user\_name\Documents\UiPath\your\_project\_name\.screenshots.



Regardless of the type of recording selected, some actions are recordable and some are not.

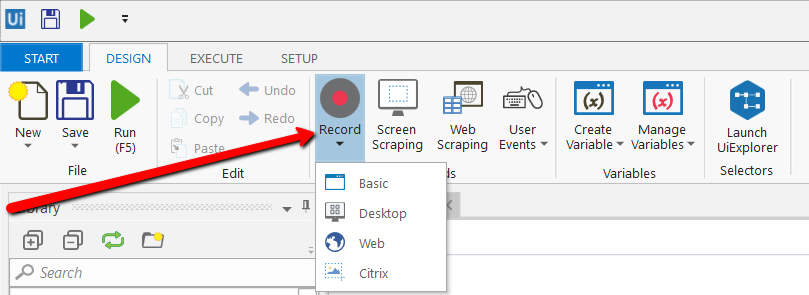
|  |  |
| --- | --- |
| **Recordable** | **Non-recordable** |
| * Left-click on buttons, check boxes, drop-down lists and other GUI elements * Text typing | * Keyboard shortcuts * Modifier keys * Right-click * Mouse hover |

About Recording Types

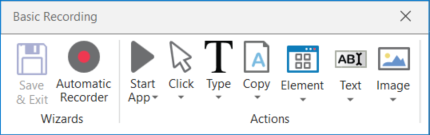
There are four types of recordings available in UiPath Studio:

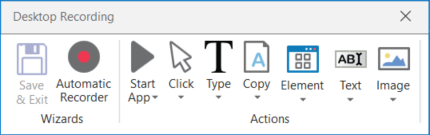
* **Basic** – generates a full selector for each activity and no container, the resulted workflow is slower than one that uses containers and is suitable for single activities.
* **Desktop** – suitable for all types of desktop apps and multiple actions; it is faster than the Basic recorder, and generates a container (with the selector of the top level window) in which activities are enclosed, and partial selectors for each activity.
* **Web** – designed for recording in web apps and browsers (supported: Internet Explorer, Google Chrome), generates containers and uses the **Simulate Type/Click**input method by default.
* **Citrix** – used to record virtualized environments (VNC, virtual machines, Citrix, etc.) or SAP, permits only image, text and keyboard automation, and requires explicit positioning.

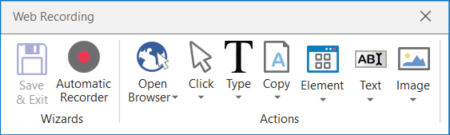
To see all the available types of recordings and select the one most suited for your workflow, click **Record**, in the **Wizards** group of the **Design** ribbon tab.



All recording types come with their own controllers (or toolbar) that give you access to actions particular to each environment, but also some common ones.

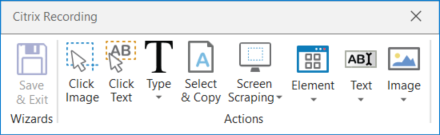






The **Desktop**, **Basic** and **Web Recording** toolbars are quite similar and enable you to:

* Automatically record multiple actions performed on the screen
* Manual recording (single actions):
  + Start or close an application or web browser
  + Click an interface element
  + Select an option from a drop-down list
  + Select a check box
  + Simulate keystrokes or keyboard shortcuts
  + Copy text from a UI element or perform screen scraping
  + Look for elements or wait for them to vanish
  + Find an image
  + Activate a window



The **Citrix Recording** toolbar enables you to:

* Click an image or text
* Simulate keystrokes or hotkeys
* Select and copy text from a window
* Scrape UI elements
* Look for elements or wait for them to vanish
* Find an image or wait for it to vanish
* Activate a window

**Note:** The Citrix Recording toolbar supports only manual recording (single actions).

To figure out if you should use automatic or manual recording in your workflow, you should better understand the differences between them and their capabilities.

|  |  |
| --- | --- |
| **Automatic Recorder** | **Manual Recorder** |
| * Left-clicks on windows, buttons, check boxes, drop-down lists etc. * Text typing | * Keyboard shortcuts * Modifier keys * Right-click * Mouse hover * Getting text * Find elements and images * Copy to Clipboard |

Keyboard shortcuts that you can use:

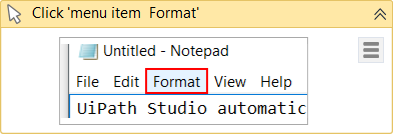
* F2 – pauses for 3 seconds. A countdown timer is displayed in the bottom left corner of the screen. Can be useful with menus that automatically hide.
* Esc – exits the automatic or manual recording. If you press the Escape key again, the recording is saved as a sequence, and you return to the main view.
* Right-click – exit the recording.

Automatic Recording

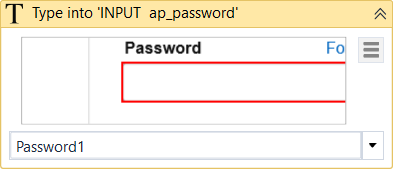
Automatic recording is very useful and time-saving as it can provide you with a skeleton for your business processes, and can be easily customized and parametrized.

As you can see in the examples below, for the actions that are recordable with the **Automatic Recording**, some activities are automatically generated:

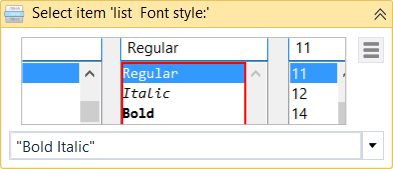
* **Click** – it is generated when you click a button (Basic and Desktop) or a link (Web). The options available in the **Properties** panel enable you to add a time delay before or after the action, change the click type and add key modifiers.



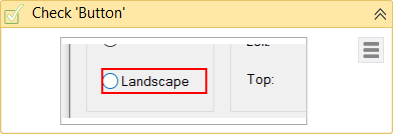
* **Type Into** – generated when typing into a text field or any editable UI element. The options available in the **Properties** panel enable you to add a time delay before or after the action or between key strokes, change the text at any point, and erase the entire field before writing to it (**EmptyField**).



* **Select Item** - generated when you select an item from a drop-down list or combo box. The options available in the **Properties** panel enable you to add a time delay before or after the action, and change the selected item.



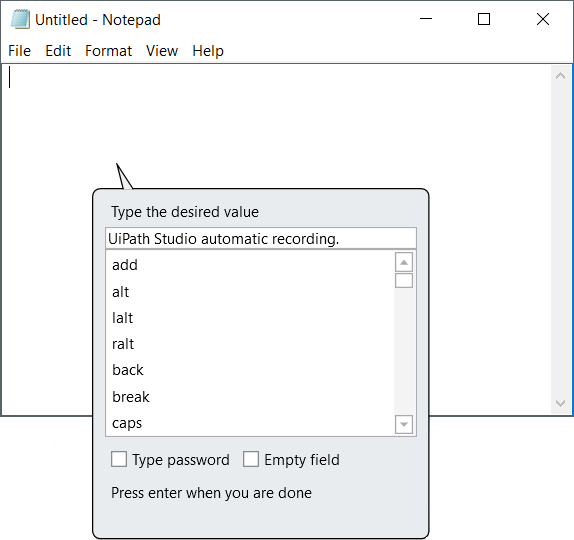
* **Check** - generated when a radio button or check box is clicked. The options available in the **Properties** panel enable you to add a time delay before or after the action, and select or unselect the check box.



Example of Automatic Recording with Basic and Desktop

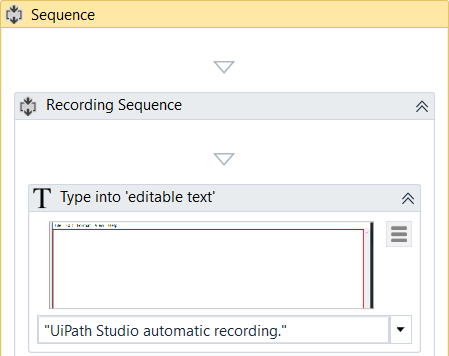
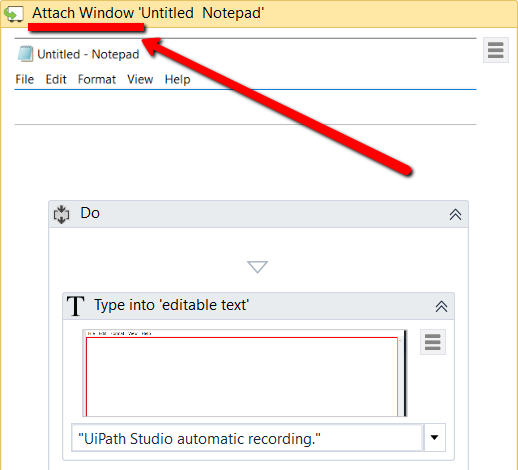
To exemplify how you can use the automatic recording and understand more about the differences between **Basic** and **Desktop**, let’s create the same workflow for the two.

1. Open Notepad.
2. In UiPath Studio, create a new sequence.
   1. In the **Design** ribbon tab, in the **Wizards** group, **select Record > Basic**. The **Basic Recording** toolbar is displayed and the main view is minimized.
   2. In the **Design** ribbon tab, in the **Wizards** group, **select Record > Desktop**. The **Desktop Recording** toolbar is displayed and the main view is minimized.
3. In the **Wizards** group, click **Automatic Recorder**. The automating recording process starts.
4. In Notepad, click in the main panel. A pop-up window is displayed.

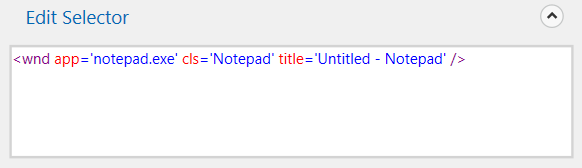


1. Type a custom text and press Enter. The string is displayed in Notepad.
2. **Note:** Select the **Empty field** check box to delete previously existing text. You can also select this option after the recording is finished, in the **Properties** panel of the **Type Into** activity.
3. From the **Format** menu, select **Font**. The **Font** window is displayed.
4. Select a different font style, such as Bold Italic, and click **OK**.
5. Press Esc two times. You exit the recording view and the saved workflow is displayed in the **Main** panel.
6. Press F5. The workflow is executed as expected.

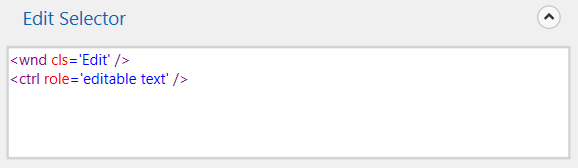
The two screenshots below display part of the resulted workflows for the **Basic** (on the left) and **Desktop** (on the right) automatic recordings. As you can see, the second one generates an **Attach Window** container, while the **Desktop** one does not.

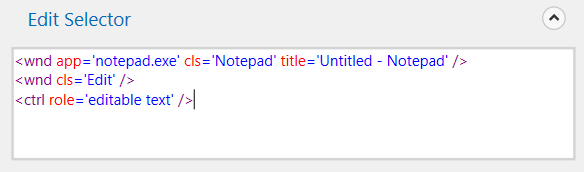
**Desktop** recorder - The top level window selector from the **Attach Window** container:



**Desktop** recorder - The partial selector for the **Type Into** activity:



**Basic** recorder – The full selector for the **Type Into** activity:



[Click here to download the Basic recording example.](https://www.uipath.com/hubfs/Documentation/WorkflowExamples/BasicRecording.xaml?t=1492088952431)

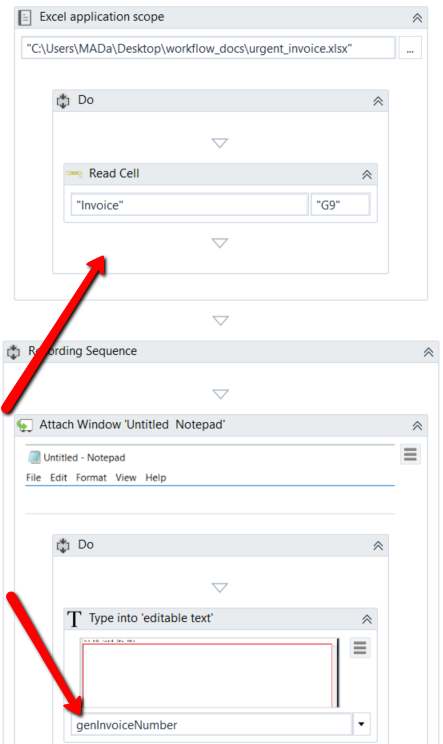
[Click here to download the Desktop recording example.](https://www.uipath.com/hubfs/Documentation/WorkflowExamples/DesktopRecording.xaml?t=1492088952431)

You can also customize and parametrize this workflow after it is recorded. To exemplify this, let’s take the **Desktop** recording example and build upon it.

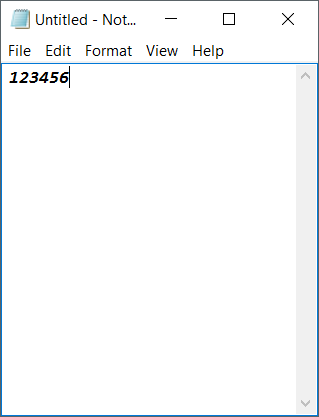
For example, let’s assume that we want to extract the number of an invoice from an Excel file, copy it to the Notepad window used in the previous example and continue the workflow as before.

1. Add an **Excel Application Scope** activity before the recording sequence.
2. In the **WorkbookPath** field, type the path of the Excel file you need to extract information from.
3. Add a **Read Cell** activity in the **Excel Application Scope**.
4. In the **Properties** panel, add the **Sheet Name** and **Cell** information from the Excel file used.
5. Right-click in the **Result** field, and click **Create Variable**. The **Set Name** field is displayed.
6. Fill in the name, such as genInvoiceNumber, and press Enter. The variable is created and displayed in the **Result** field and **Variables**
7. Change the scope of the variable to **Main**.
8. In the recording sequence, in the **Type Into** activity, in the **Text** field, add the **genInvoiceNumber** variable. This copies the value stored in the variable to Notepad.

What was added to the workflow should look as in the following screenshot.



1. Press F5. The workflow is executed as expected.

[Click here to download this example.](https://www.uipath.com/hubfs/Documentation/WorkflowExamples/DesktopRecordingWithExcel.xaml?t=1492088952431)

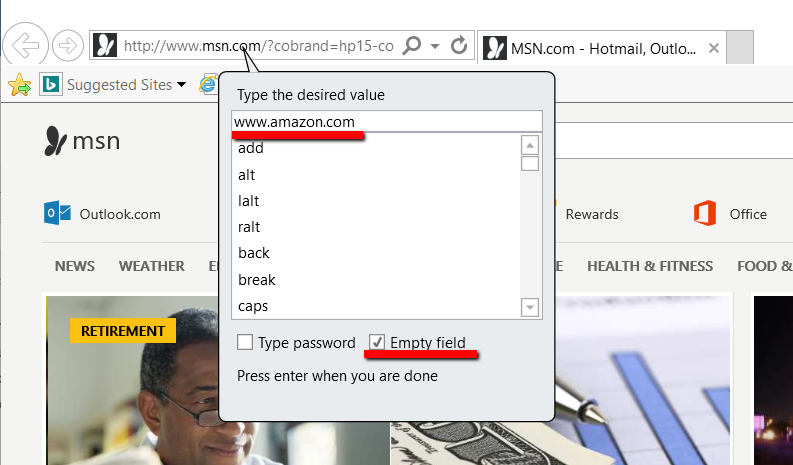
Top of Form

# 

# Example of Automatic Recording with Web

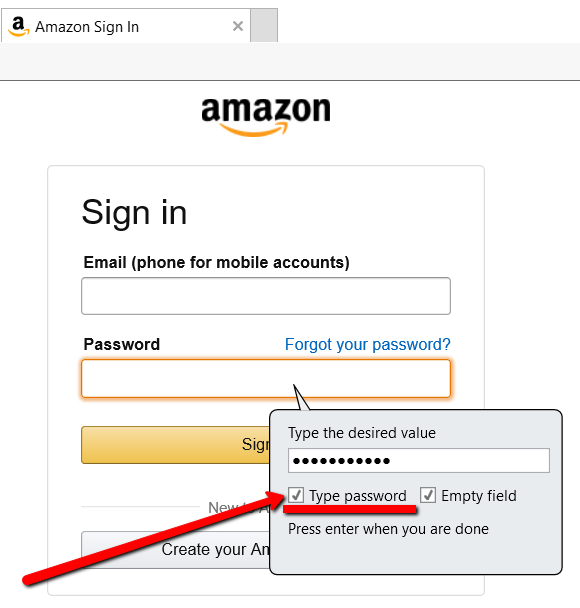
To exemplify how you can use the web recording and understand how it works, let’s create a workflow that enables you to go to Amazon and sign in to your account.

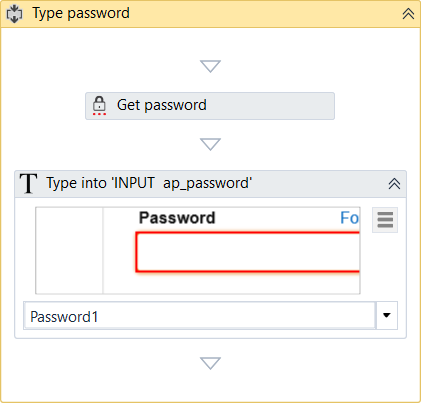
1. Open an Internet Explorer instance.
2. In UiPath Studio, create a new sequence.
3. In the **Design** tab, in the **Wizards** group, select **Record > Web**. The **Web Recording**toolbar is displayed and the main view is minimized.
4. Click **Automatic Recording**. The automating recording process starts.
5. Navigate to [www.amazon.com](http://www.amazon.com/).



1. Go to the Sign In page and input your e-mail and password.

**Note:** When the **Type Into** pop-up is displayed for your password, make sure that you select the **Type Password** check box. Besides the **Type Into** activity, another activity, **Get Password**, is created in the sequence. This activity hides the password behind asterisks (\*) and stores it in a string variable.





1. Click **Sign In** and press Esc two times. The recording is saved and displayed as a workflow in the **Main** panel.
2. Close and open Internet Explorer manually.
3. In Studio, press F5. The workflow is executed as expected.

As you can see, the workflow requires you to close and open Internet Explorer manually. If you want to automate this part too, you need to manually add the **Open Browser** and **Close Tab** activities at the begging and end of the workflow.

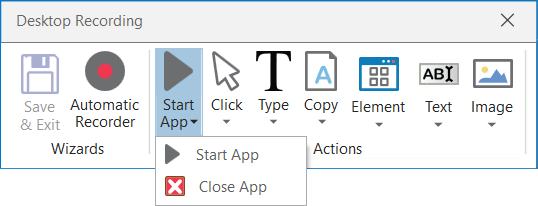
[Click here to download this example.](https://www.uipath.com/hubfs/Documentation/WorkflowExamples/WebRecording.xaml?t=1492088952431)

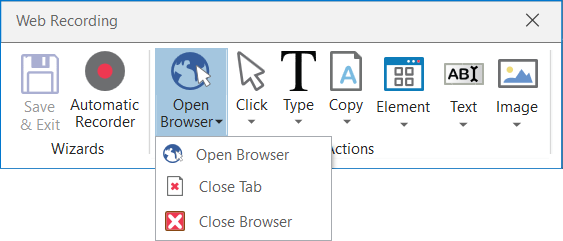
# Manual Recording

As explained in the [About Recording Types chapter](https://www.uipath.com/guides/about-recording-types), there are some actions that cannot be handled by the automatic recorder. We refer to these as single actions or manual recordings. You can use both manual and automatic recording in workflows to achieve full automation of a task. Single actions can be found in the **Actions** group of any recording toolbar.

## Types of Single Actions

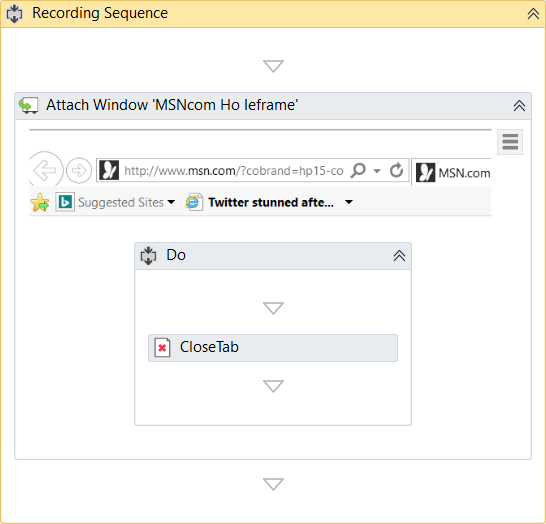
### 1. Start and Stop an App or Browser

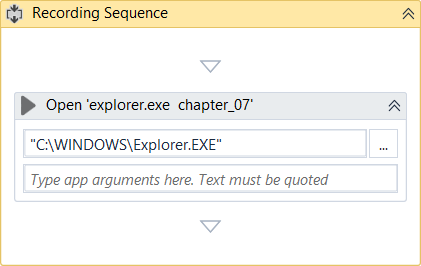




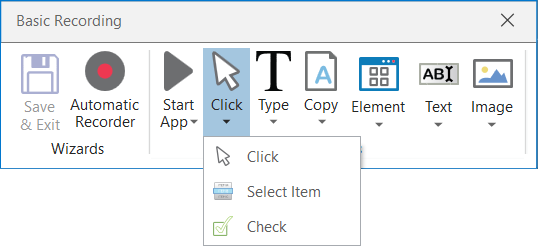
These single actions enable you to open an app or browser, as well as close them, by pointing and clicking them.

The activities generated using the desktop and web manual recorders contain partial selectors and containers (first screenshot), while the activities generated by the basic recorder contains a full selector and no container (second screenshot), just like with the automatic recording.



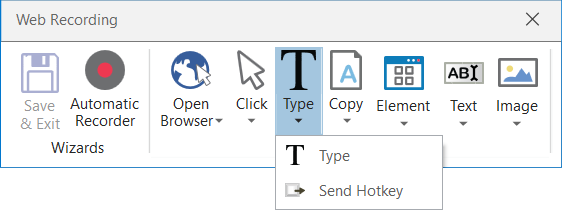


### 2. Click

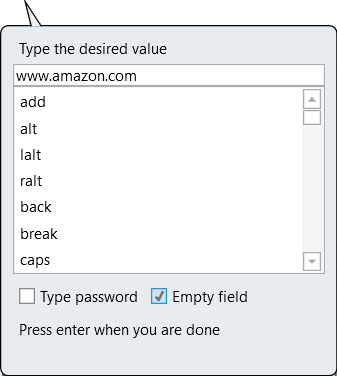
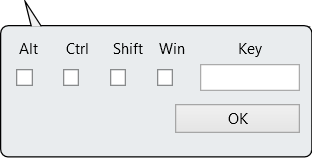


These types of actions enable you to record clicks on the desktop or a running application, select an option from a drop-down list or combo box, and select a check box or a radio button.

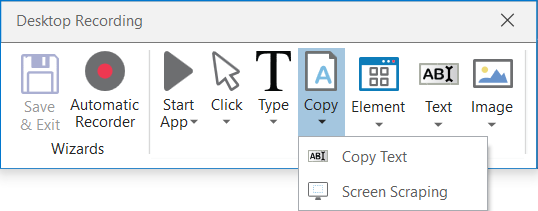
### 3. Type

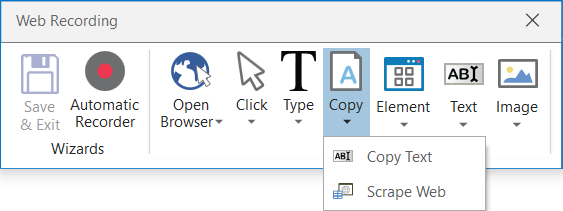


These single actions include those that require input from the keyboard, such as keyboard shortcuts and keypresses. To achieve this, two pop-up windows are used to retrieve your keyboard input.

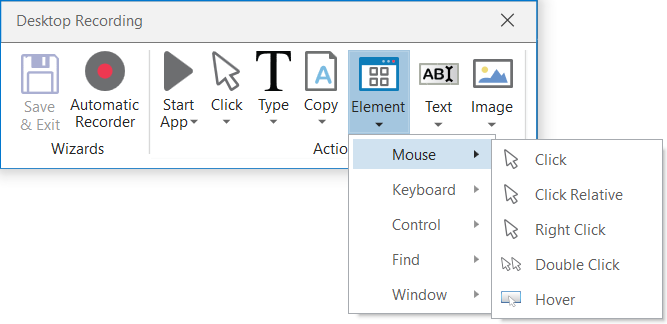
### 4. Copy





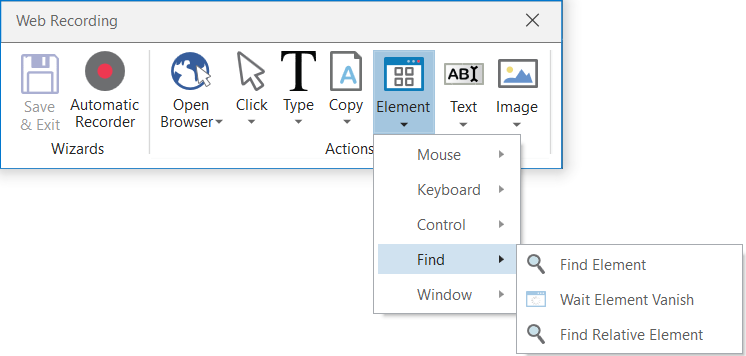
These actions enable you to copy a selected text from an opened application or web browser, so that you can use it later in the workflow. Screen scraping is also available under the **Copy** menu, as it enables you to extract images and text from an app or browser. For more information, see [Output or Screen Scraping Methods](https://www.uipath.com/guides/output-or-screen-scraping-methods).

### 5. Mouse Element



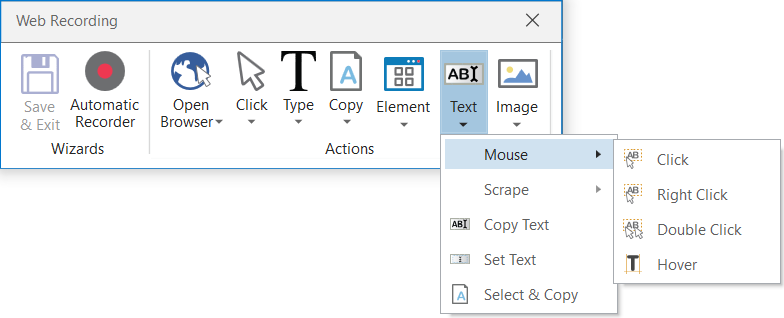
These types of actions enable you to simulate mouse movements that cannot be recorded but give you access to more functionalities, such as right-clicking, hovering or double-clicking.

### 6. Find Element



These types of single actions enable you to identify specific UI elements or pause the workflow until a particular window closes or an UI element is no longer displayed. The find relative element action is useful with apps that do not allow direct interaction with UI elements, such as Citrix.

### 7. Text



Text single actions enable you to select or hover over text to make tooltips visible for scraping, right-click to make the context menu visible, copy and paste text and many others.

### 8. Image

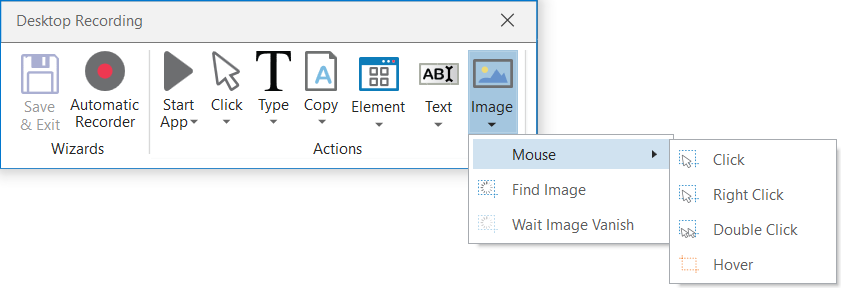


Image single actions enable you to wait for a specified image to disappear, to find a specific image in an app or website, right-click or hover over an image and others. This type of manual recording can be useful with UI elements that are saved as graphics, as they cannot be highlighted as normal fields or text.

UI ELEMENTS

About UI Elements

UI elements refer to all graphical user interface pieces that construct an application, be they windows, check boxes, text fields or drop-down lists, and so on. Knowing how to interact with them enables you to implement UI automation much faster and easier.

All interactions with the UI can be split into input and output. This categorization helps you better understand which actions to use in different scenarios, when to use them, and the technology behind them. These are also going to be useful when trying to deal with scraping. For more information, see [Input Methods](https://www.uipath.com/guides/input-methods) and [Output or Screen Scraping Methods](https://www.uipath.com/guides/output-or-screen-scraping-methods).

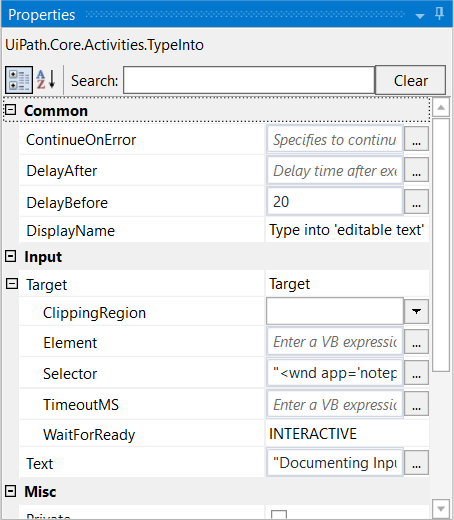
|  |  |
| --- | --- |
| Input Actions | Output Actions |
| * Clicks * Text Typing * Keyboard shortcuts * Right-clicks * Mouse hover * Clipboard actions * Etc. | * Getting text * Finding elements and images * Clipboard actions * Etc. |

UI Activities Properties

There are multiple activities that can be used to automate apps or web-apps and you can find them in the **Activities** panel, under the **UI Automation** category.

All of these activities have multiple properties in common:

* **ContinueOnError** – specifies if the workflow should continue, even if the activity throws an error. This field only supports boolean values (True, False).
* **DelayAfter** – adds a pause after the activity, in milliseconds.
* **DelayBefore** – adds a pause before the activity, in milliseconds.
* **TimeoutMS** – specifies the amount of time (in milliseconds) to wait for a specified element to be found before an error is thrown. The default value is 30000 milliseconds (30 seconds).
* **WaitForReady** – wait for the target to become ready, before performing the activity. There are three available options:
  + **None**– does not wait for the target to be ready.
  + **Interactive –**waits until only a part of the app is loaded.
  + **Complete –**waits for the entire app to be loaded.
* **Target** – identifies the UI element the activity works with.



The target is composed of multiple pieces, namely the container, selector and clipping region, to ensure that you correctly identify a UI element.

A container gives you a little more context for the button or field you want to use, so that you can tell windows apart or different areas of the same app. They are automatically generated, but you can make changes to them in the **Properties** panel.

The following are containers:

* **Attach Window**
* **Open Application**
* **Attach Browser**
* **Open Browser**
* **Get Active Window**

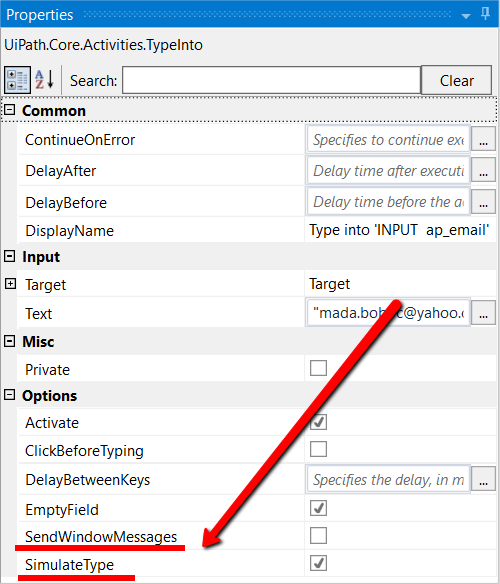
Input Methods

Input actions require you or the robot to directly interact with an opened application or web page. There are three types of input methods for click and type actions, that differ in terms of compatibility and capability.

We generally recommend the **Simulate Type/Click** method as it is the fastest of the three and works in the background, but only if you do not need to send special keyboard shortcuts. If this does not work for you, try the **Windows Messages** method and then the **Default** one, as it is the slowest.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method\Capability** | **Compatibility** | **Background execution** | **Speed** | **Hotkey Support** | **Auto Empty Field** |
| **Default** | 100% | no | 50% | yes | no |
| **Window Messages** | 80% | yes | 50% | yes | no |
| **Simulate Type/Click** | 99% - web apps  60% - desktop apps | yes | 100% | no | yes |

The input method can be changed at any point from the **Properties** panel of the selected activity. If the **SimulateType** or **SendWindowMessages** check boxes are not selected, then the **Default** method is applied.

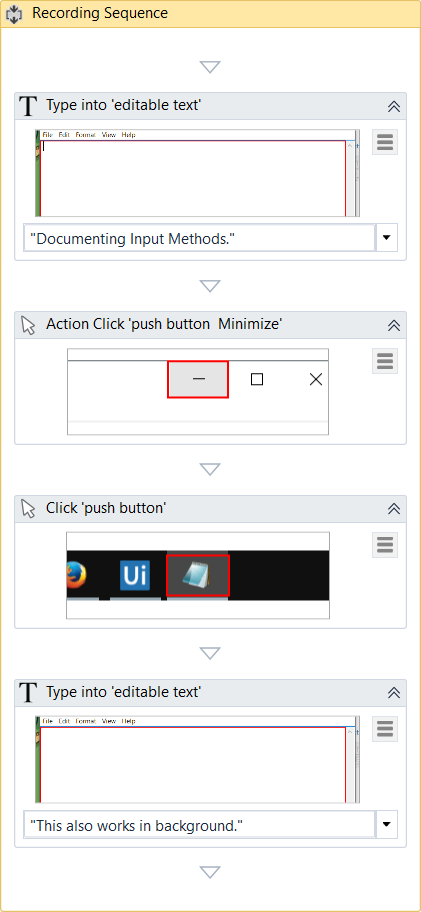


The **Default** application simulates a click or type with the help of the hardware driver, while the **Simulate Type/Click** method uses the technology of the target application. Lastly, the **SendWindowMessages** works by sending a specific message directly to the target application.

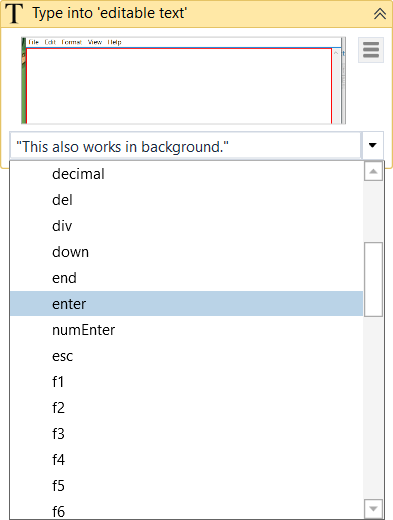
Example of Using Input Methods

To get a clearer picture of how the three input methods work, let’s create a simple workflow that writes something in a Notepad window and switch between the methods.

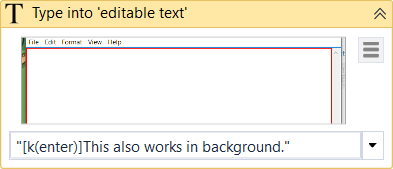
1. Open a Notepad window.
2. In Studio, from the **Basic Recording** toolbar, start the automatic recorder.
3. Type something into the Notepad window.
4. Minimize the window and restore it.
5. Type something else in the Notepad window.
6. Press Esc two times. The workflow is saved and displayed in the **Main** panel. It should look as in the following screenshot.



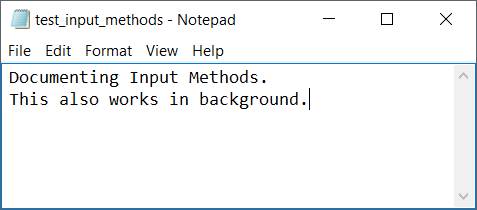
1. Move the activity that restores the Notepad window after the second **Type Into** one. We do this to check if the type of input method selected can also write to Notepad in background mode.
2. From the drop-down of the second **Type Into** activity, select **enter**. A special key string is displayed at the end of the previously-existing text.



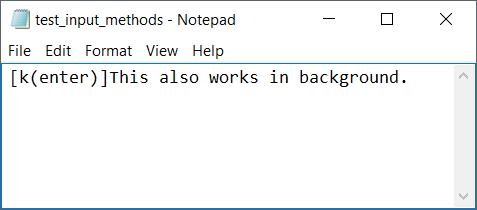
1. Copy the special key string at the beginning of the sentence. This enables you to test special keys, such as Enter that adds a new line in a text editor.



1. Run the workflow once with every input method. Note the differences:
   * The **Default** method – it does not automatically erase previously written text, supports special keys, but writing in the background does not work;
   * The **Window Messages**method – works in the background, supports special keys, but it does not erase pre-existing text (you have to manually select the **Empty Field** check box in the **Properties** panel);



* + The **Simulate Type/Click** method – works in the background, but it automatically erases pre-existing text, and does not support special keys.



Therefore, be careful to choose the method that best suits your needs. If special keys are a must, you might want to avoid the **Simulate Type/Click** method, or if speed is what matters most, then maybe **Simulate Type/Click** is the right one.

Output or Screen Scraping Methods

Output or screen scraping methods refer to those activities that enable you to extract data from a specified UI element or document, such as a .pdf file.

To understand which one is better for automating your business process, let’s see the differences between them.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Method\Capability** | **Speed** | **Accuracy** | **Background Execution** | **Extract Text Position** | **Extract Hidden Text** | **Support for 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 |

**FullText** is the default method, it is fast and accurate, yet unlike the **Native** method, it cannot extract the screen coordinates of the text.

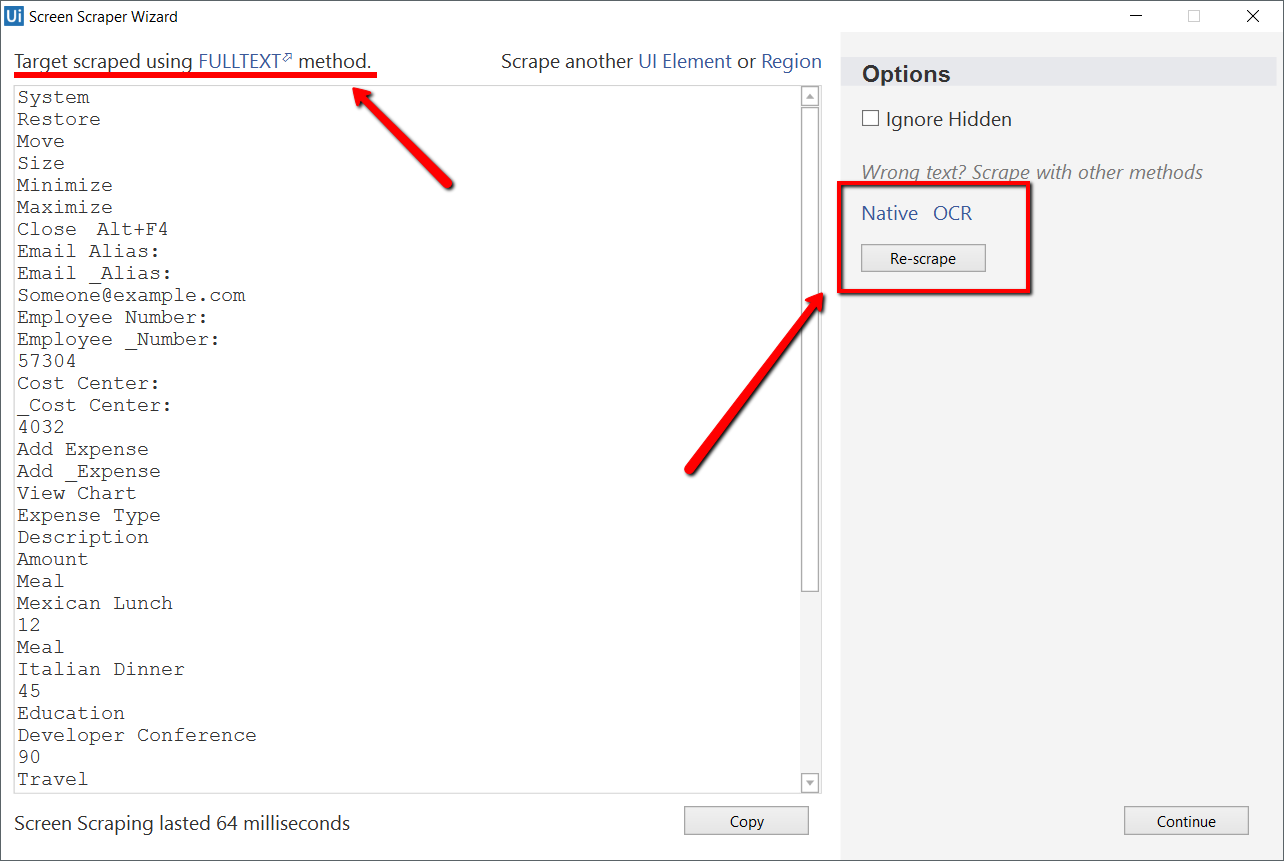
Both these methods work only with desktop applications, but the **Native** method cannot work with apps that are not built to render text with the Graphics Device Interface (GDI).

**OCR** is not 100% accurate, but can be useful to extract text that the other two methods could not, as it works with all applications including Citrix. Studio uses two OCR engines, by default: Google Tesseract and Microsoft Modi.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Method\Capability** | **Multiple languages support** | **Preffered area size** | **Support for color inversion** | **Set expected text format** | **Filter allowed characters** | **Best with Microsoft fonts** |
| **Google Tesseract** | Can be added | Small | yes | yes | yes | no |
| **Microsoft Modi** | Supported by default | Large | no | no | no | yes |

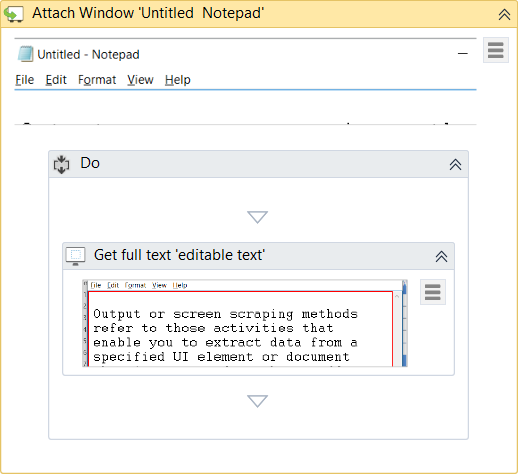
To start extracting text from various sources, click the **Screen Scraping** button, in the **Wizards** group, on the **Design** ribbon tab.

The screen scraping wizard enables you to point at a UI element and extract text from it, using one of the three output methods described above. Studio automatically choses a screen scraping method for you, and displays it at the top of the **Screen Scraper Wizard**window.



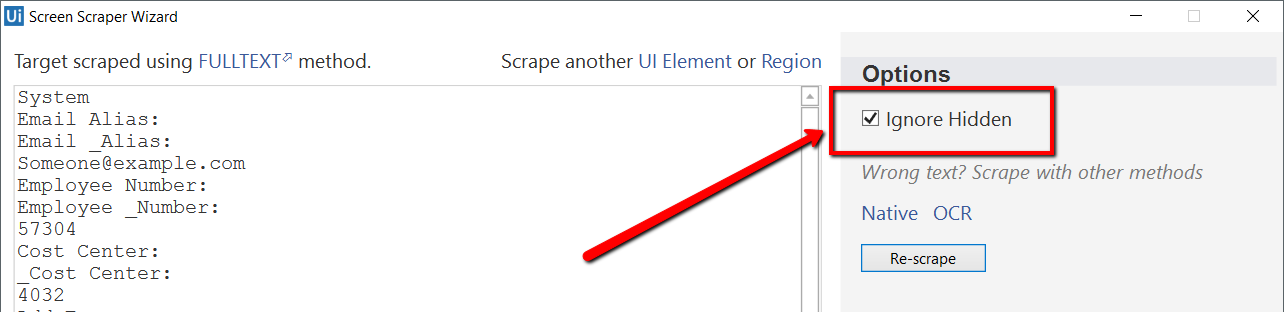
To change the method of screen scraping, select another one from the **Options** panel and then click **Re-scrape**.

When you are satisfied with the scraping results, click **Continue**or**Copy**. The latter option copies the extracted text to the Clipboard, while the first one saves your information to the **Main** panel and, just as with [desktop recording](https://www.uipath.com/guides/about-recording-types), generates a container (with the selector of the top level window) in which activities are enclosed, and partial selectors for each activity.



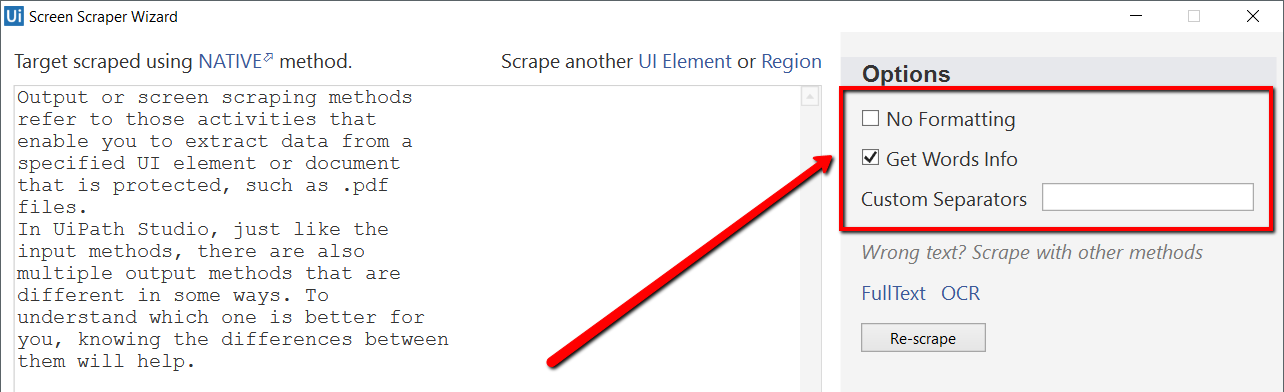
Each type of screen scraping comes with different features in the **Screen Scraper Wizard**, in the **Options** panel:

1. **FullText**

****

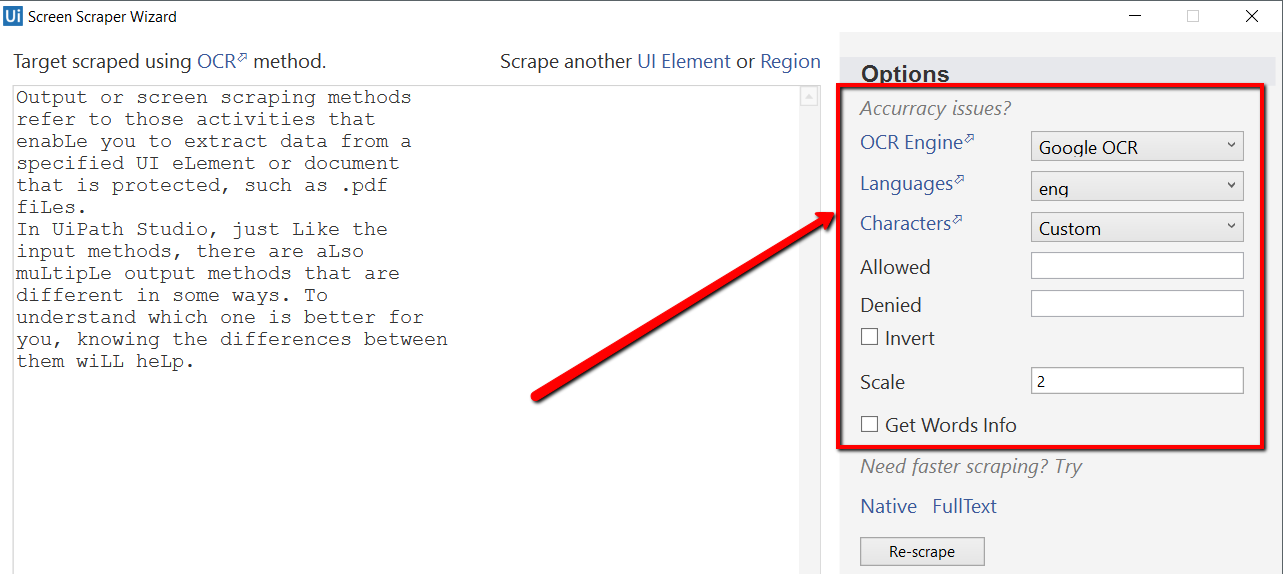
* + **Ignore Hidden** – when this check box is selected, the hidden text from the selected UI element is not copied.

1. **Native**

****

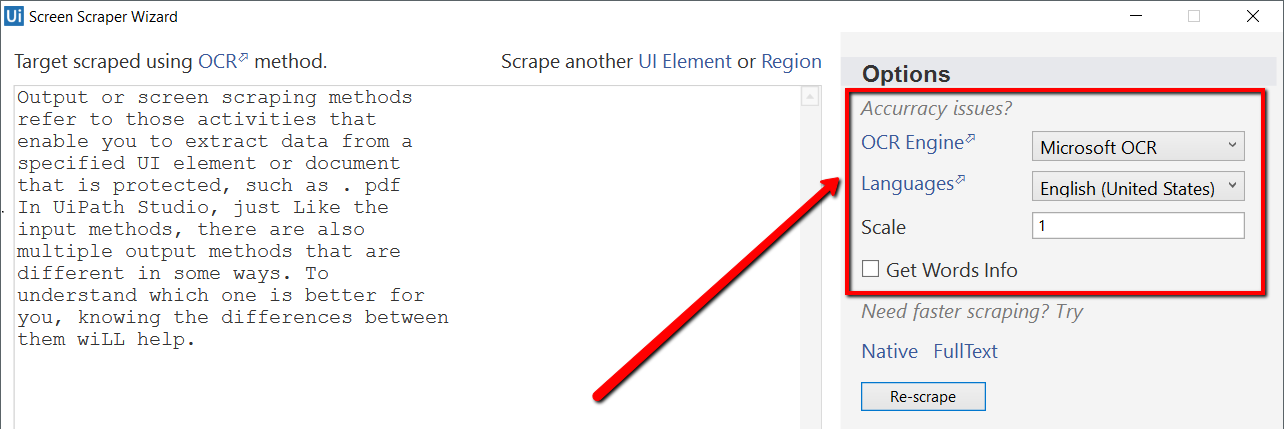
* + **No Formatting** – when this check box is selected, the copied text does not extract formatting information from the text. Otherwise, the extracted text’s relative position is retained.
  + **Get Words Info** – when this check box is selected, Studio also extracts the screen coordinates of each word. Additionally, the **Custom Separators** field is displayed, that enables you to specify the characters used as separators. If the field is empty, all known text separators are used.

1. **Google OCR**



* + **Languages** – only English is available by default.
  + **Characters** – enables you to select which types of characters to be extracted. The following options are available: **Any character**, **Numbers only**, **Letters**, **Uppercase**, **Lowercase**, **Phone numbers**, **Currency**, **Date** and **Custom**. If you select **Custom**, two additional fields, **Allowed** and **Denied**, are displayed that enable you to create custom rules on which types of characters to scrape and which to avoid.
  + **Invert** - when this check box is selected, the colors of the UI element are inverted before scraping. This is useful when the background is darker than the text color.
  + **Scale** – the scaling factor of the selected UI element or image. The higher the number is, the more you enlarge the image. This can provide a better OCR read and it is recommended with small images.
  + **Get Words Info** – gets the on-screen position of each scraped word.

1. **Microsoft OCR**



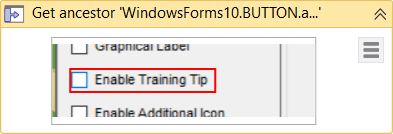
* + **Languages** - enables you to change the language of the scraped text. By default, English is selected.
  + **Scale** – the scaling factor of the selected UI element or image. The higher the number is, the more you enlarge the image. This can provide a better OCR read and it is recommended with small images.
  + **Get Words Info** - gets the on-screen position of each scraped word.

Besides getting text out of an indicated UI element, you can also extract the value of multiple types of attributes, its exact screen position and its ancestor.

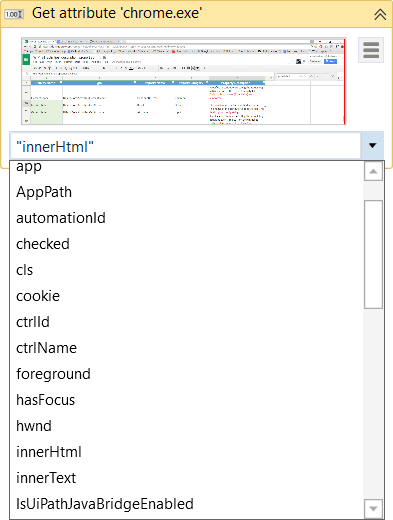
This type of information can be extracted through dedicated activities that are found in the **Activities** panel, under **UI Automation > Element > Find** and **UI Automation > Element > Attribute**.

These activities are:

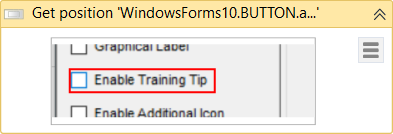
* **Get Ancestor** – enables you to retrieve an ancestor from a specified UI element. You can indicate at which level of the UI hierarchy to find the ancestor, and store the results in a UiElement variable.



* **Get Attribute** – retrieves the value of a specified UI element attribute. Once you indicate the UI element on screen, a drop-down list with all available [attributes](https://github.com/UiPath/SDK/wiki/UiNode-Attributes) is displayed.

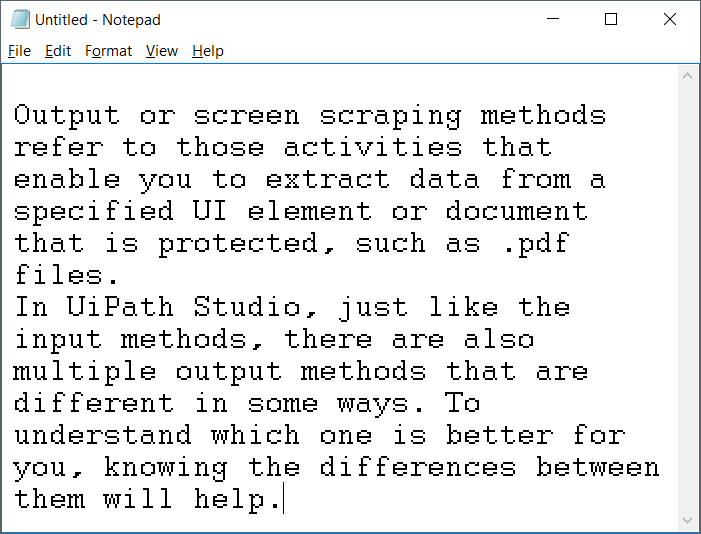


* **Get Position** – retrieves the bounding rectangle of the specified UiElement, and supports only Rectangle variables.

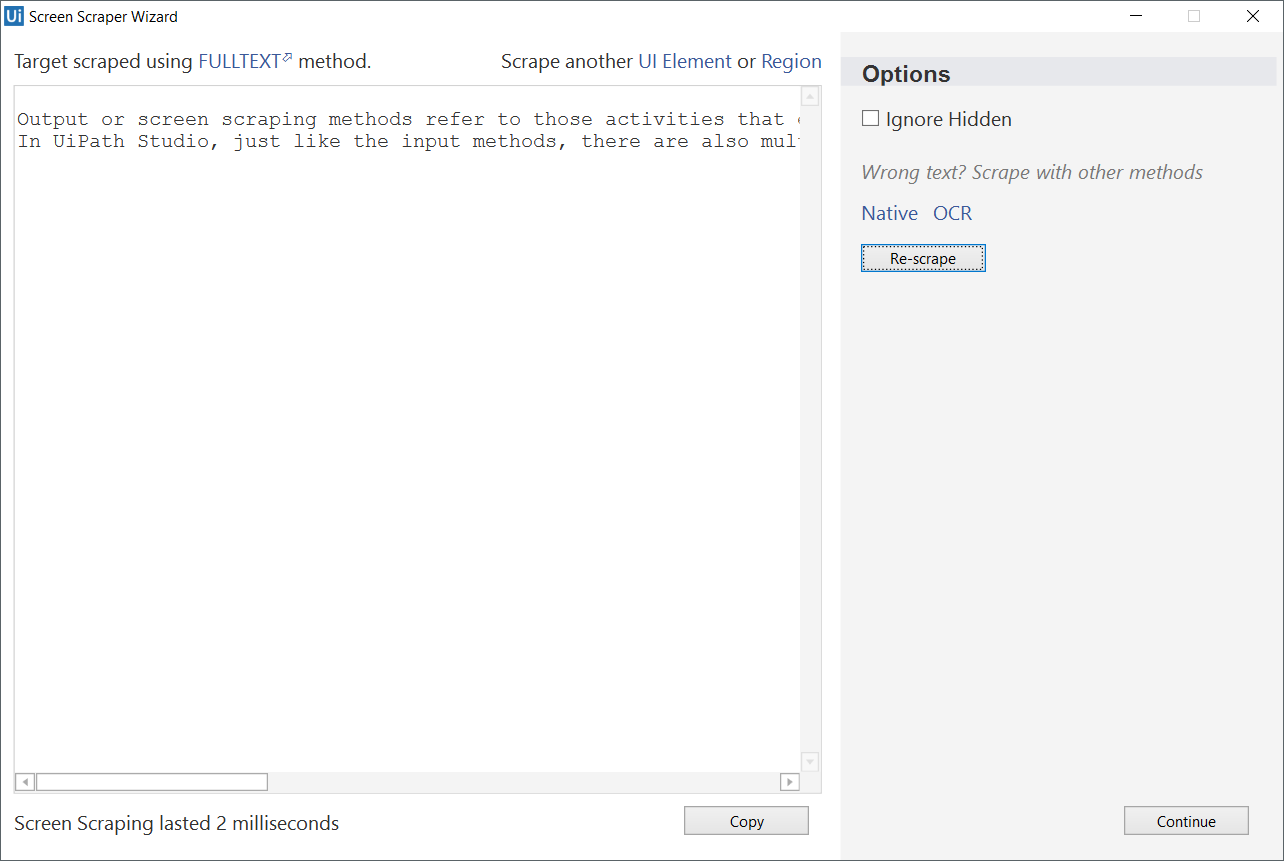


# Examples of Using Output or Screen Scraping Methods

To exemplify how to use the three screen scraping methods and the practical differences between them, let’s first scrape a Notepad window with some text and see what results we have. The following screenshot is what we used.

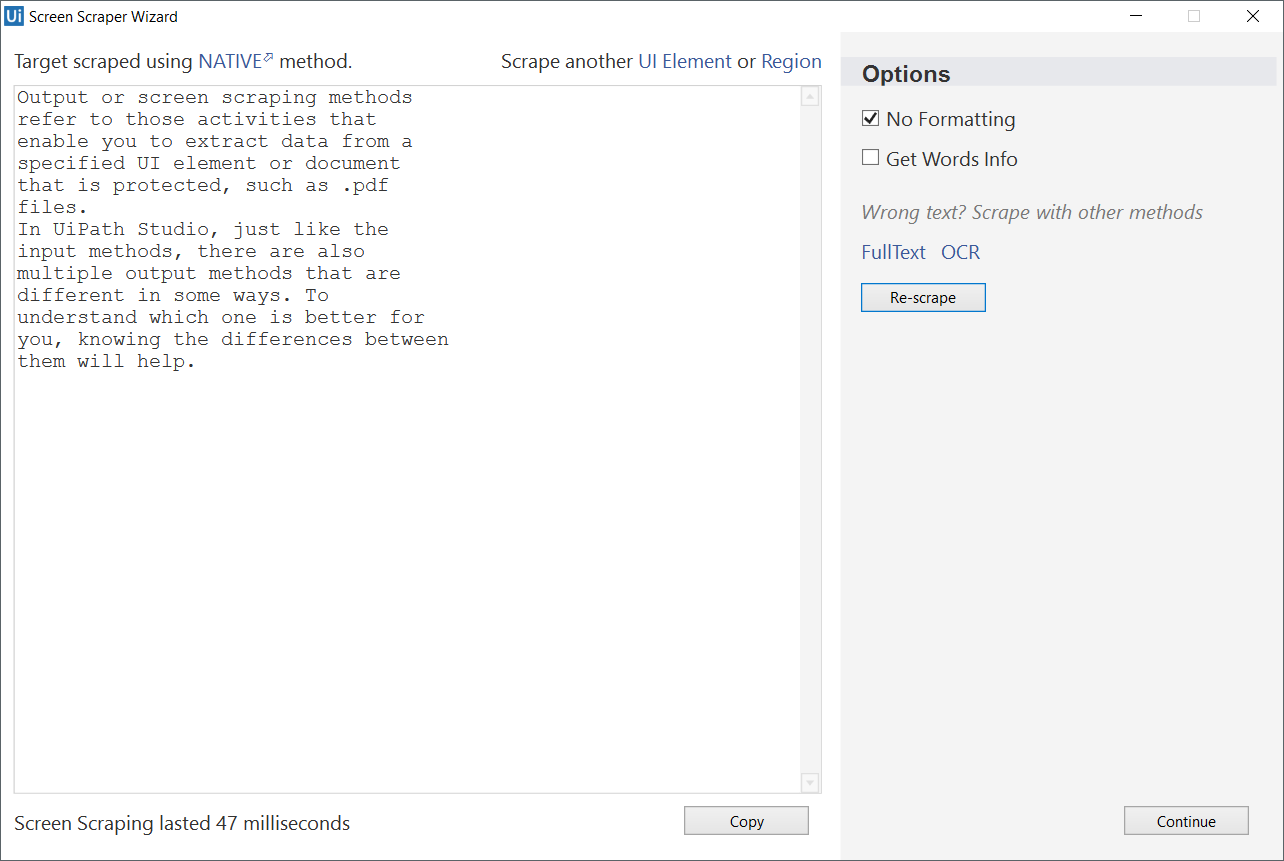


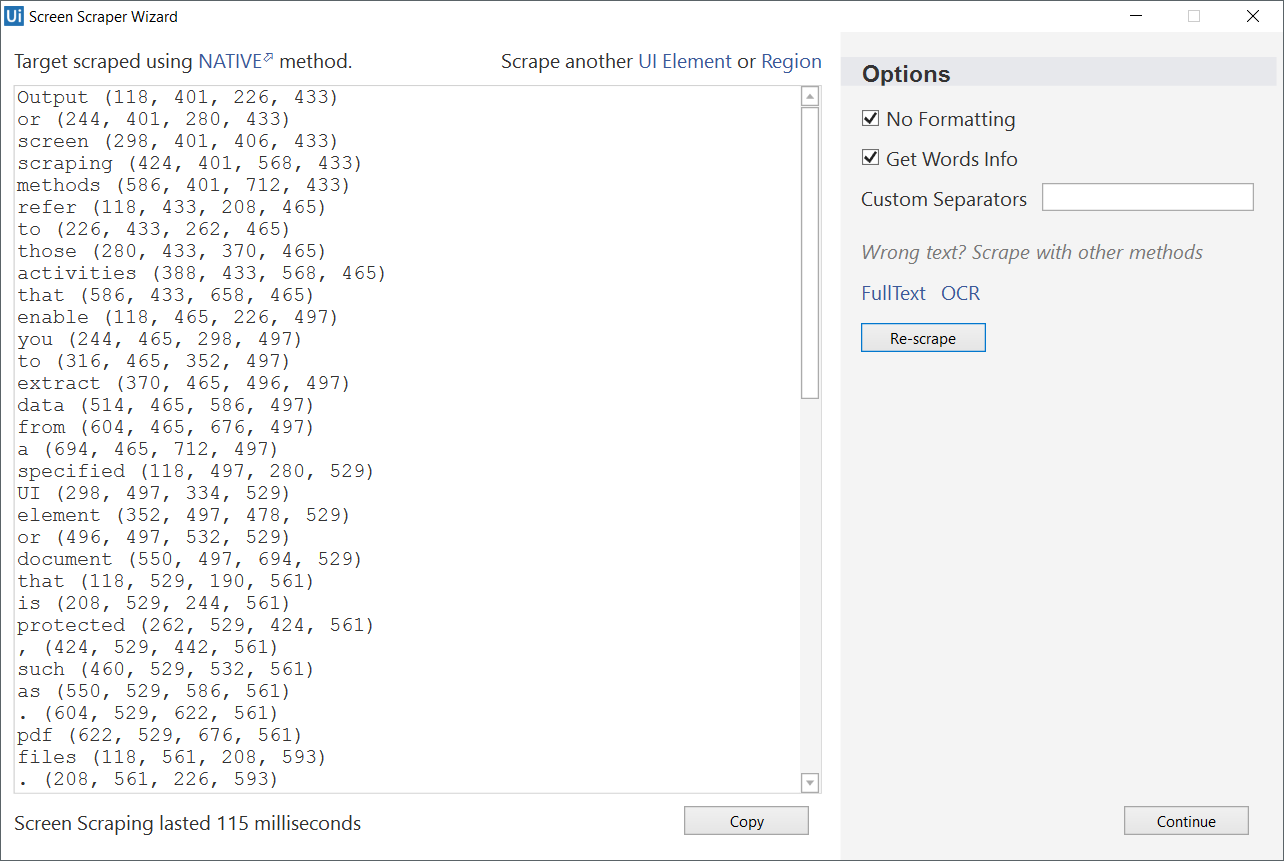
#### The FullText method

****

As you can see, no formatting is retained, but if you hide the Notepad window while scraping, the text is still retrieved. This is the fastest method.

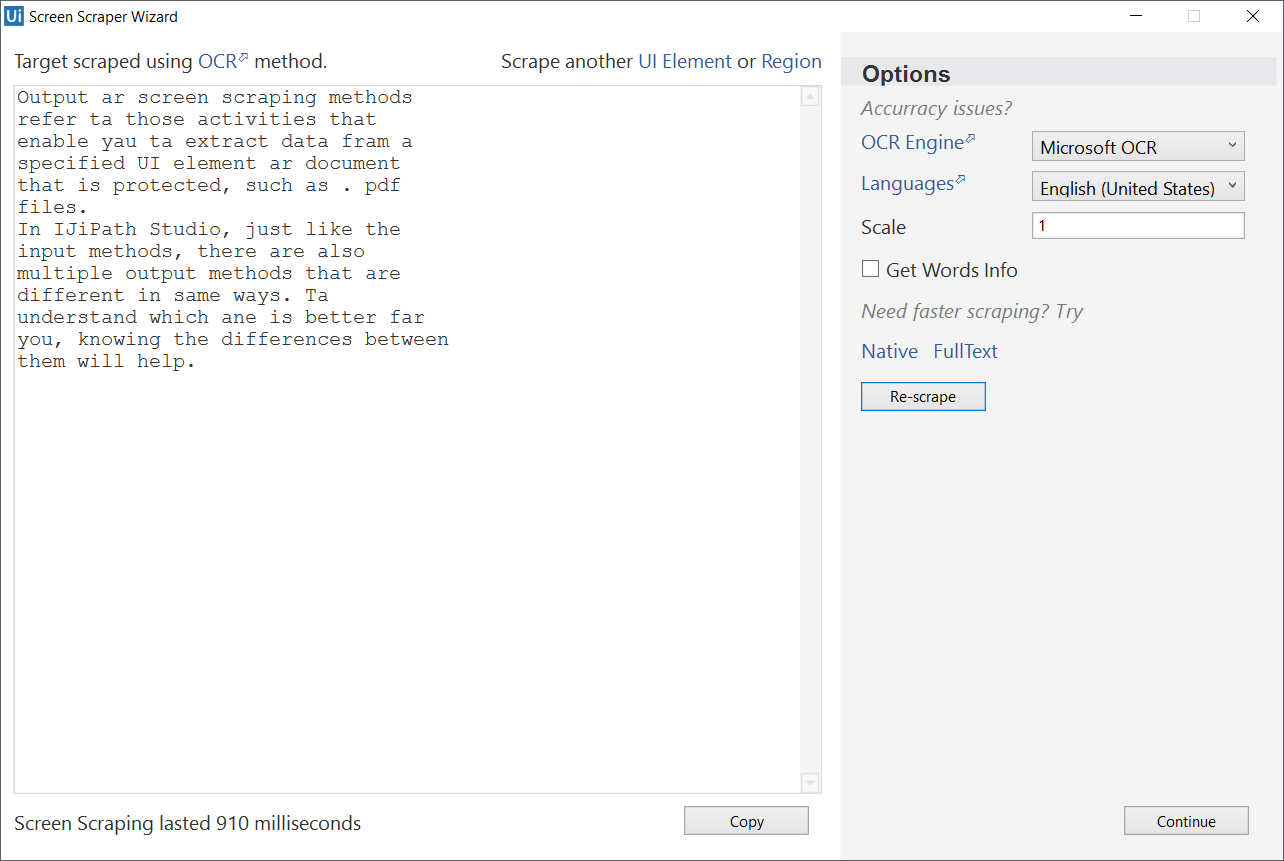
#### The Native method





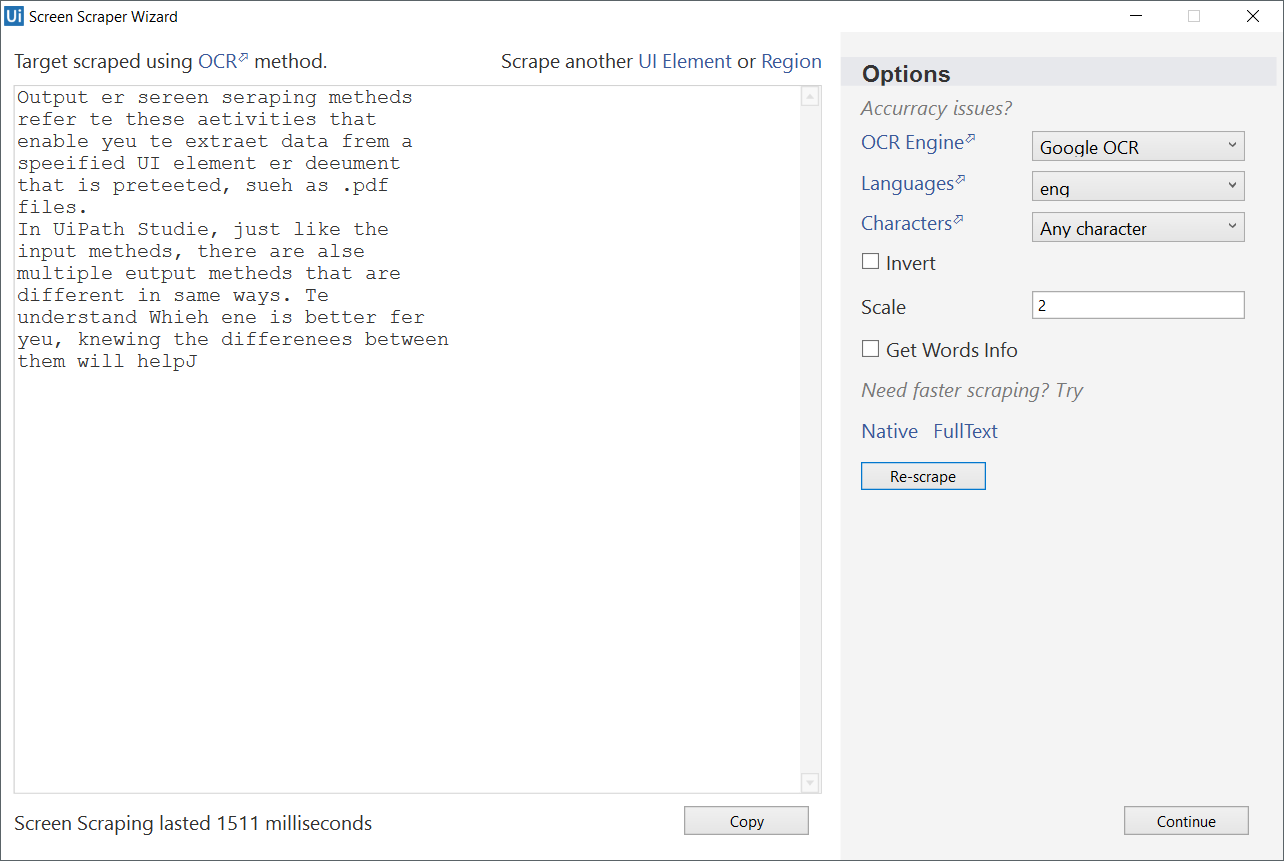
As you can see in the first screenshot, you can extract the text with its position on the screen, as well as retrieve the exact position of each word (second screenshot).

#### The Microsoft OCR method



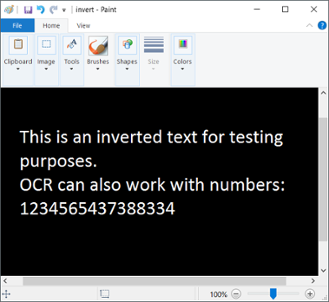
As you can see, the accuracy of this output method is not 100%, but it still manages to keep the position of the text. Getting the exact on-screen position, in pixels, is also available yet as you can see, it is not the fastest of the output methods.

#### The Google OCR method

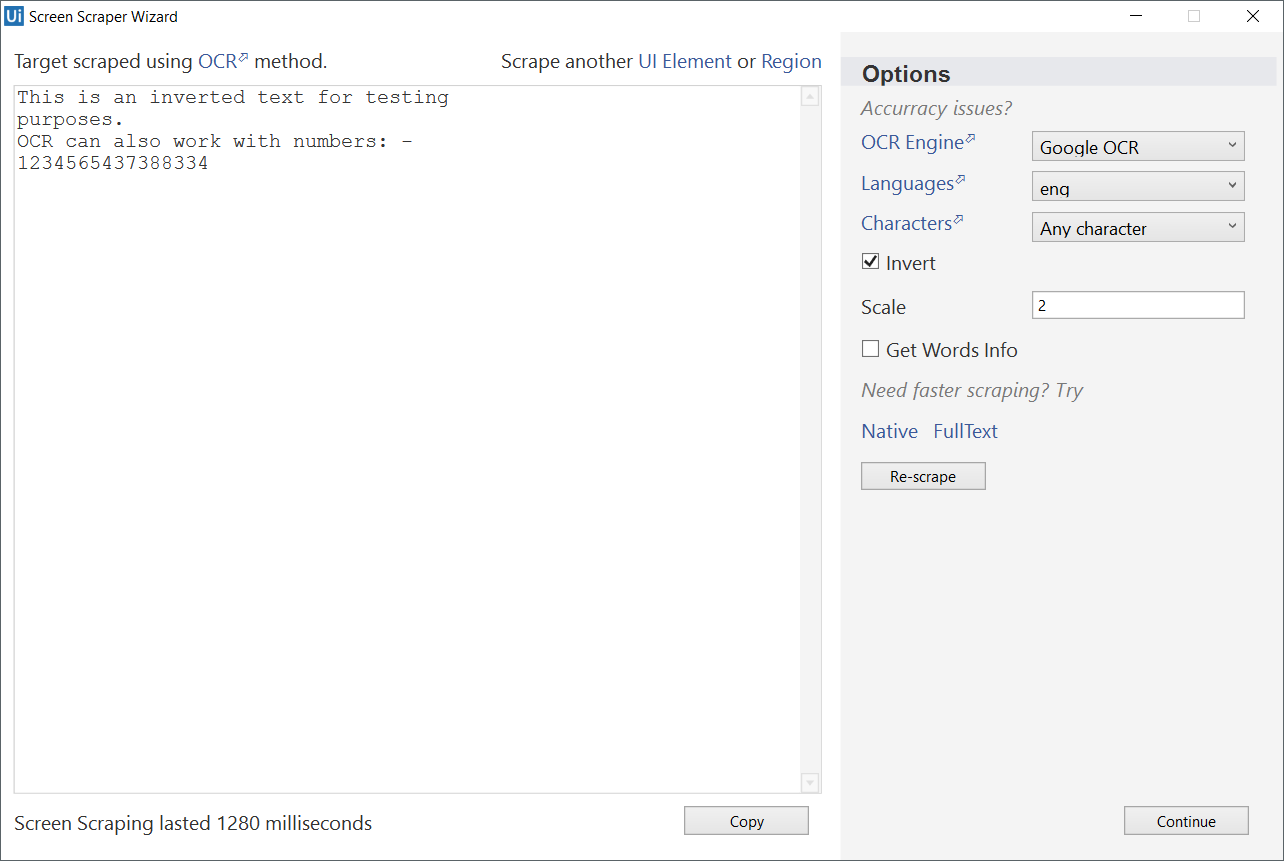


As with Microsoft’s Modi, the Google OCR method is not 100% accurate and takes longer when compared with the others. However, it retrieves the position within the window of the text.

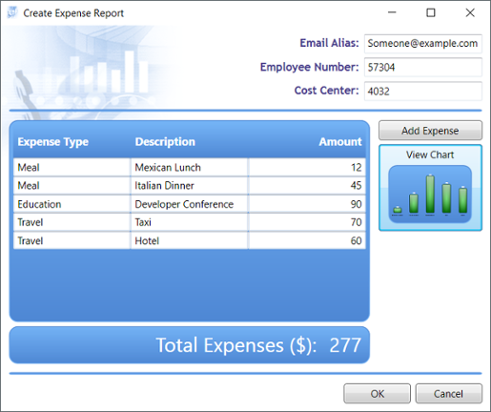
Now, add some white text over a black page in Paint, for example, and try to scrape it.



As you can see, only the OCR methods work in this scenario.

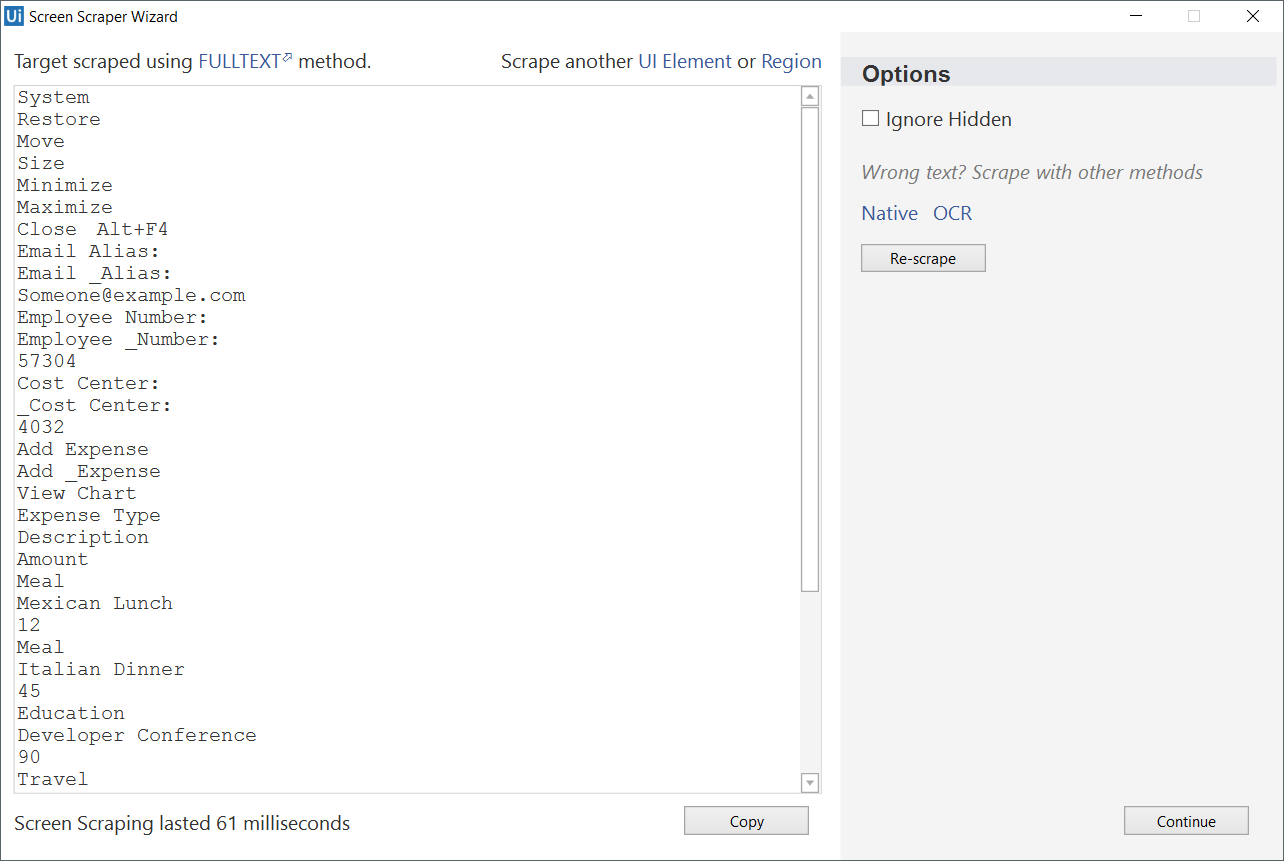


Now let’s try scraping an application and see the results. We use a dummy expense app, which you can [download here.](https://drive.google.com/open?id=0BxVAKI3l9Pc5N1ItTTNqbDFKUWM)

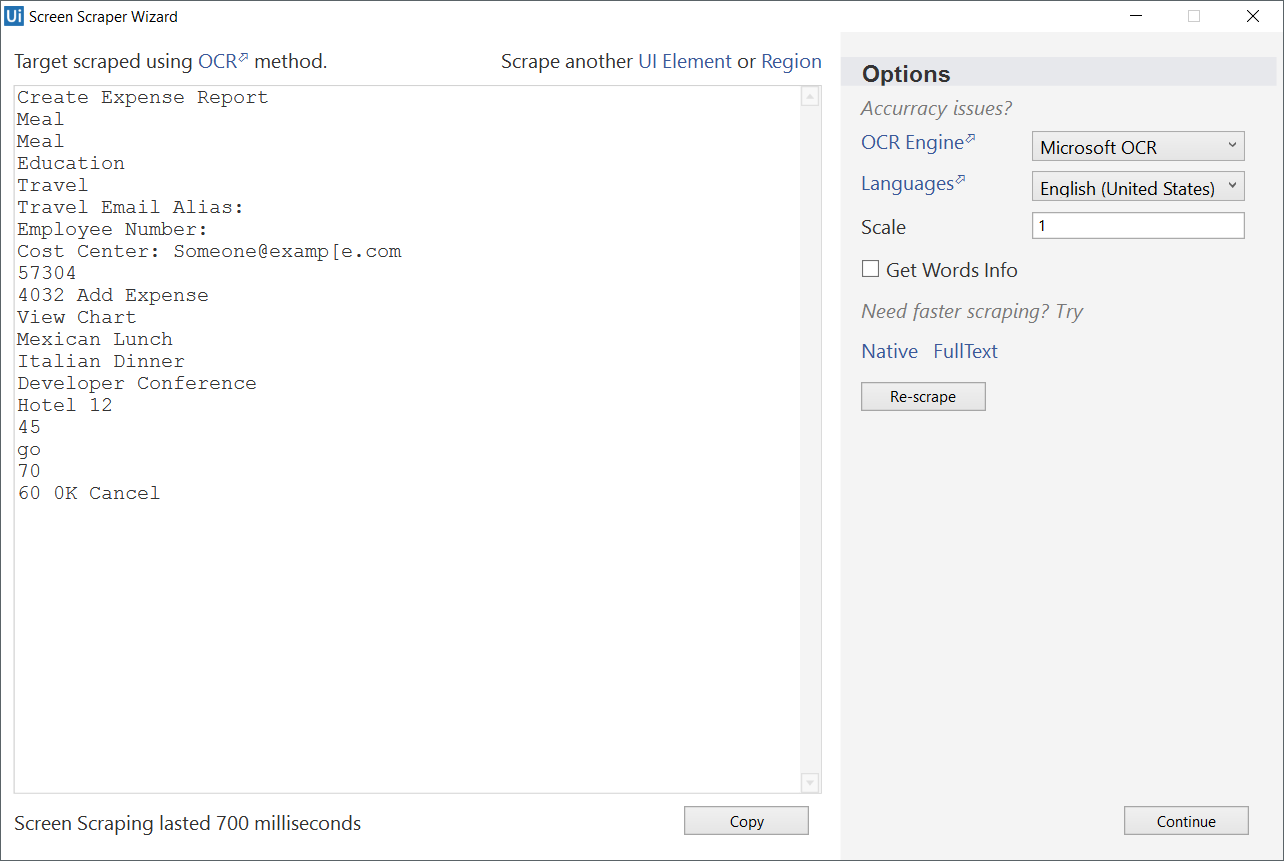


If we scrape this entire window, we receive the following results:

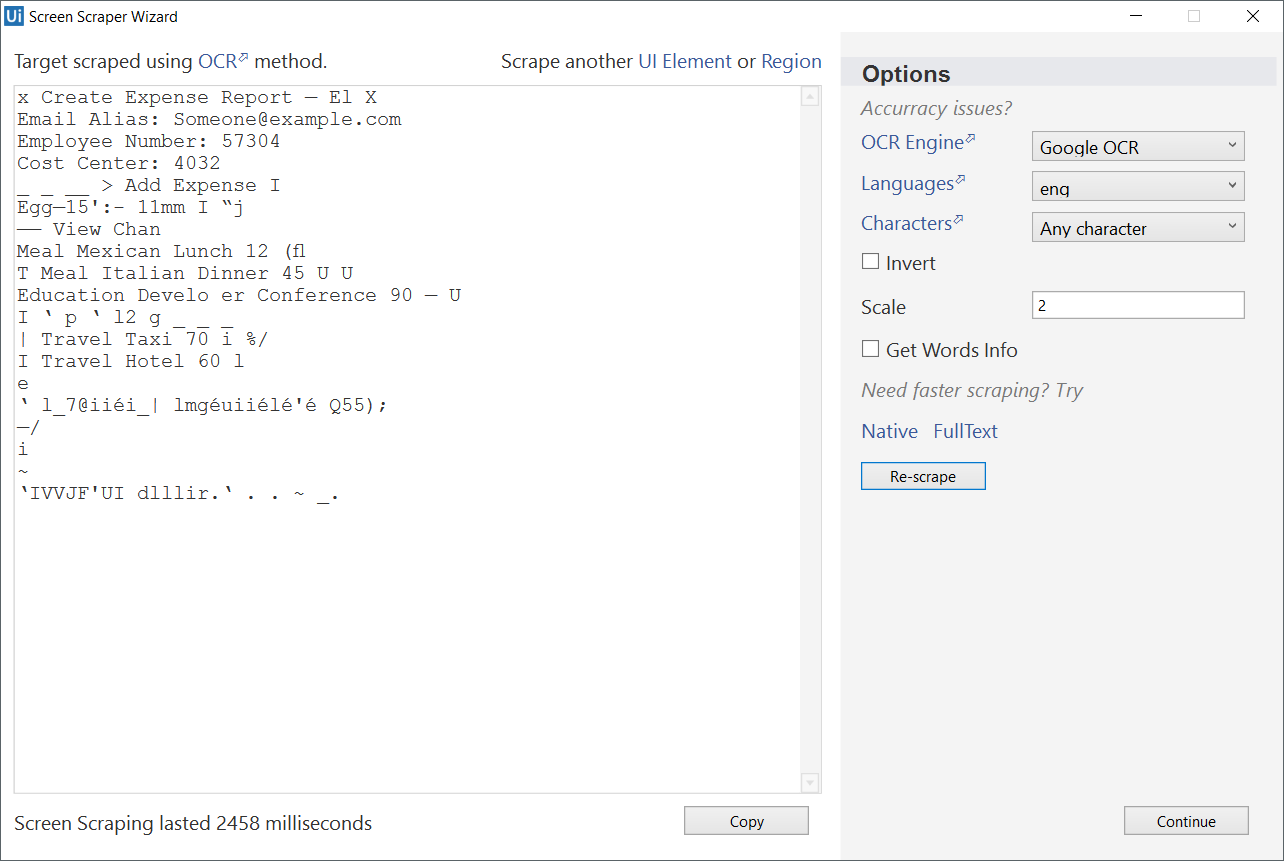
1. **FullText** with hidden text works really well, being able to read even the minimize and restore buttons.



1. **Native** does not work on this UI as it does not make use of GDI to render text.
2. **Microsoft OCR** works pretty well, although accuracy is still not 100%.



1. **Google OCR** does not handle this UI very well, as the scraped area is quite large.



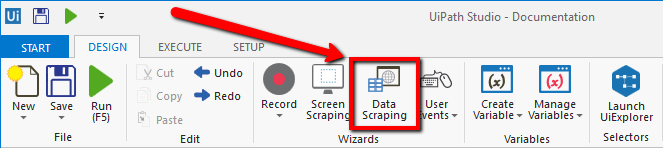
DATA SCRAPING

About Data Scraping

Data scraping enables you to extract structured data from your browser to a database, .csv file or even Excel spreadsheet.

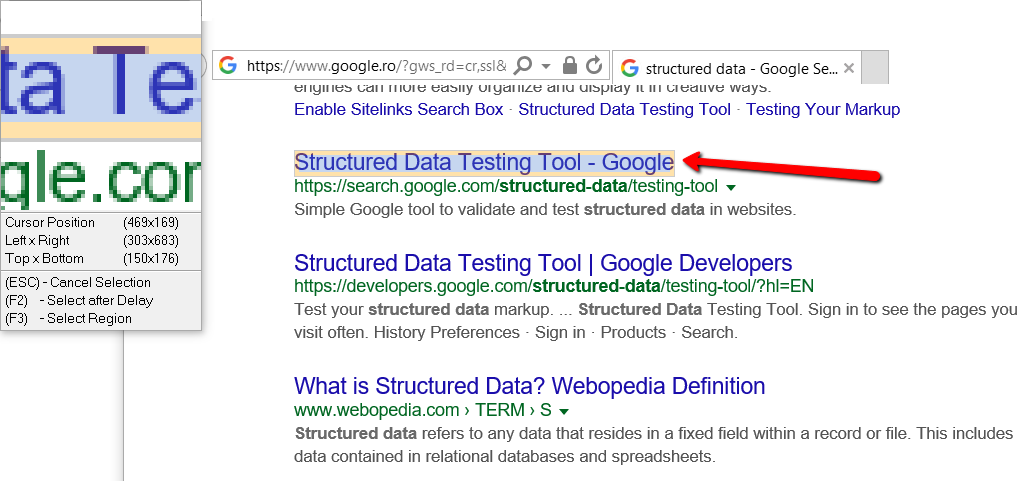
Structured data is a specific kind of information that is highly organized and is presented in a predictable pattern. For example, all Google search results have the same structure (a link at the top, a string of the URL and a description of the web page), which enables Studio to easily extract the information, as it always knows where to find it.

The scraping wizard can be opened from the **Design** tab, by clicking the **Data Scraping**button.

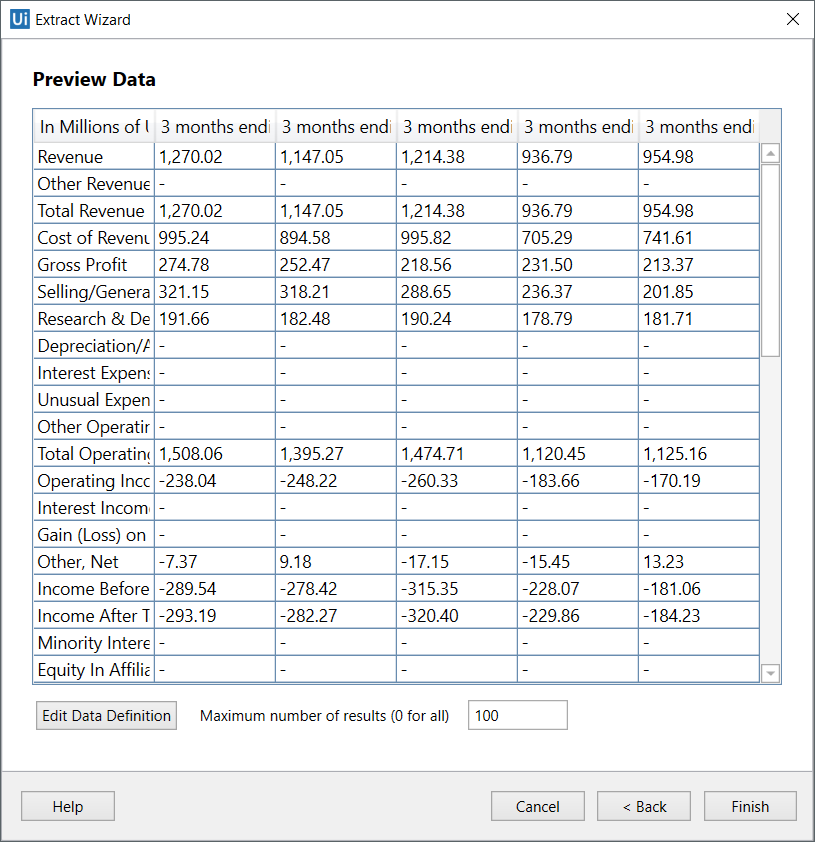


The main steps of the data scraping wizard are:

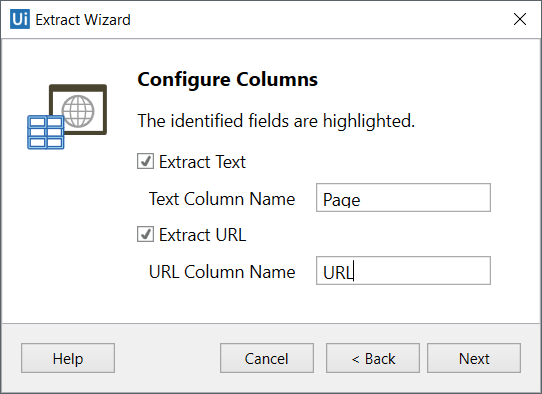
1. Select the first and last fields in the web page that you want to extract data from, so that Studio can deduce the pattern of the information.



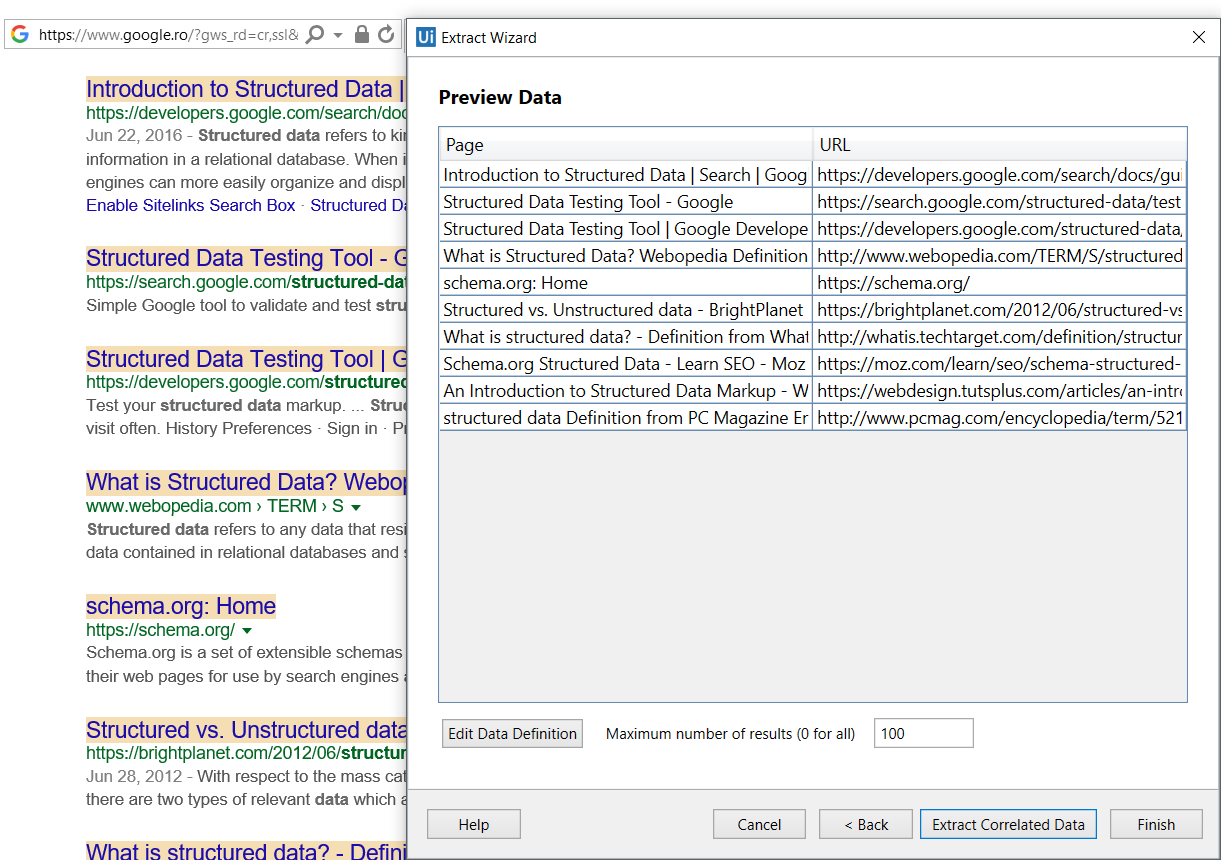
**Note:** Studio automatically detects if you indicated a table cell, and asks you if you want to extract the entire table. If you click **Yes**, the **Extract Wizard** displays a preview of the selected table data.



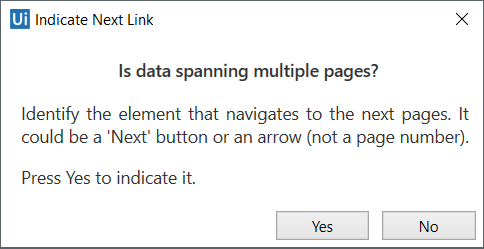
1. Customize column headers and choose whether or not to extract URLs.



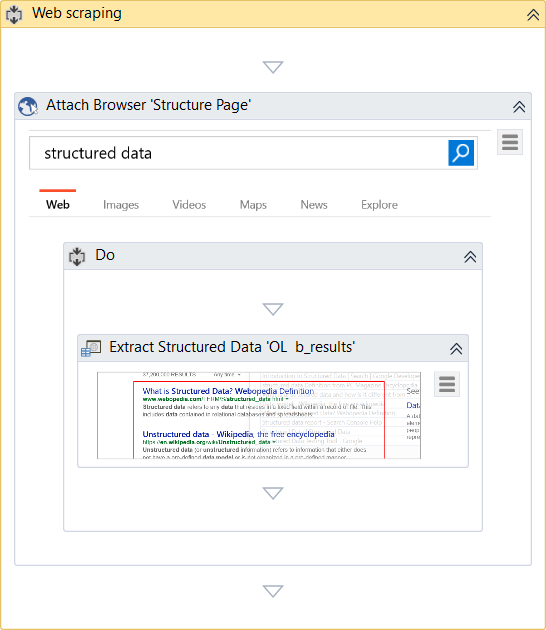
1. Preview the data, edit the number of maximum results to be extracted and change the order of the columns.



1. **Optionally**click **Extract Correlated Data**. This enables you to go through the **Extract Wizard** again, to extract additional info and add it as a new column in the same table.
2. Indicate the **Next** button in the web page (if the information you want to extract spans multiple pages).



After you are finished with the wizard, a workflow is generated in Studio.



Data scraping always generates a container (**Attach Browser** or**Attach Window**) with a selector for the top-level window and an **Extract Structured Data** activity with a partial selector, thus ensuring a correct identification of the page to be scraped.

Additionally, the **Extract Structured Data**activity also comes with an automatically generated XML string (in the **ExtractMetadata** property) that indicates the data to be extracted.

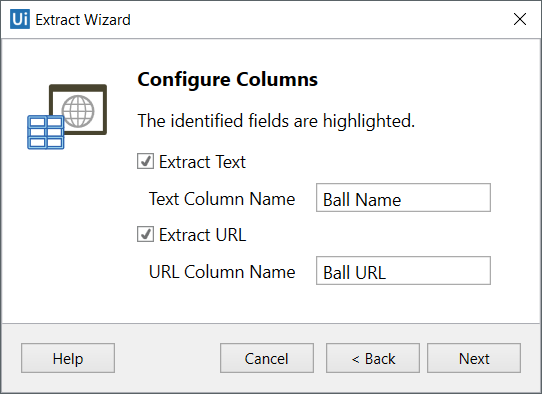
Lastly, all the scraped information is stored in a Data Table variable, that you can later use to populate a database, a .csv file or an Excel spreadsheet.

Example of Using Data Scraping

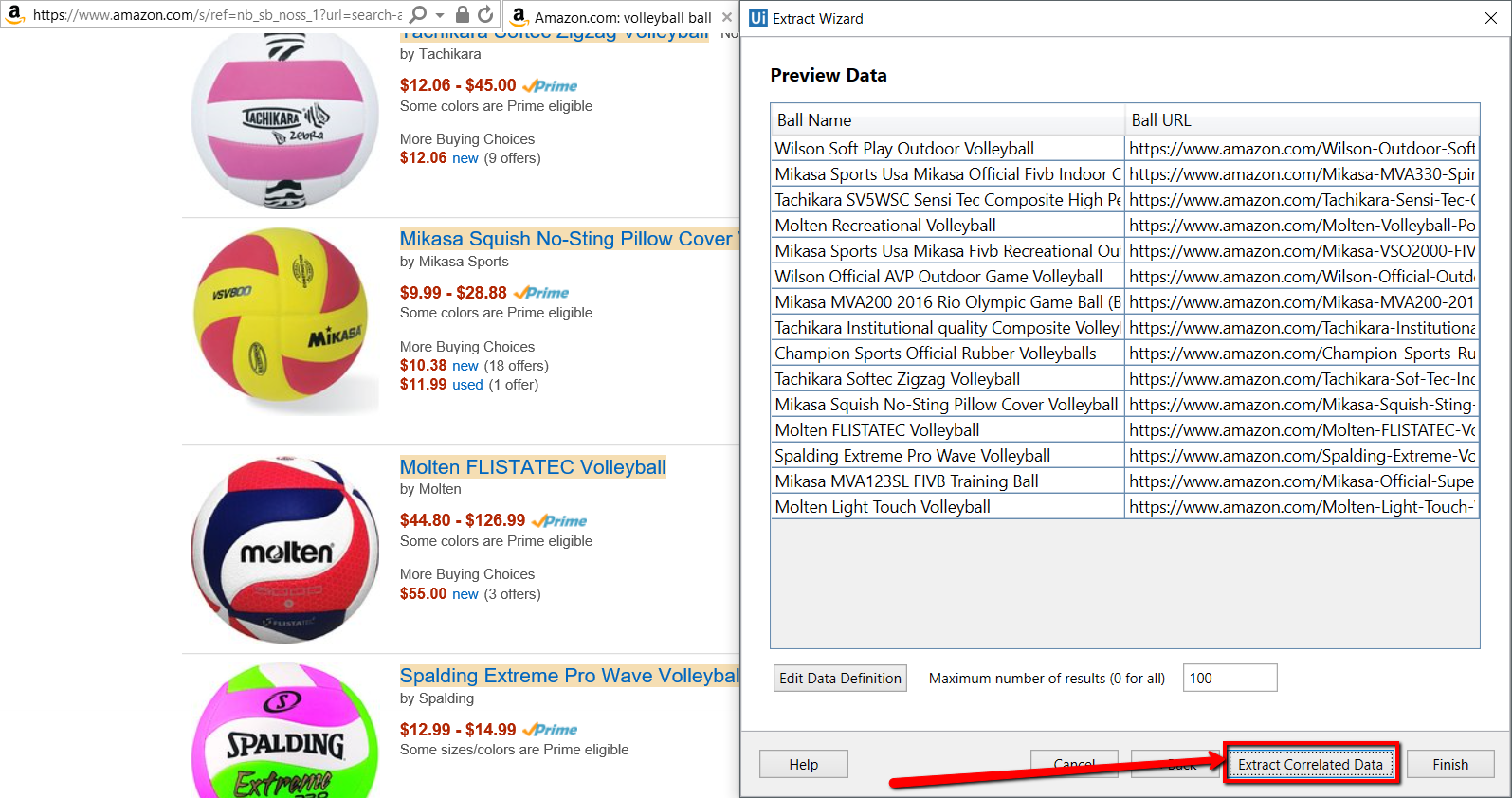
To better understand how you can take advantage of the data scraping functionality, let’s create a workflow that extracts some specific information from Amazon.

Let’s say you are a sports gear vendor and are interested in finding out the latest prices for volleyball balls on Amazon. You can do the following:

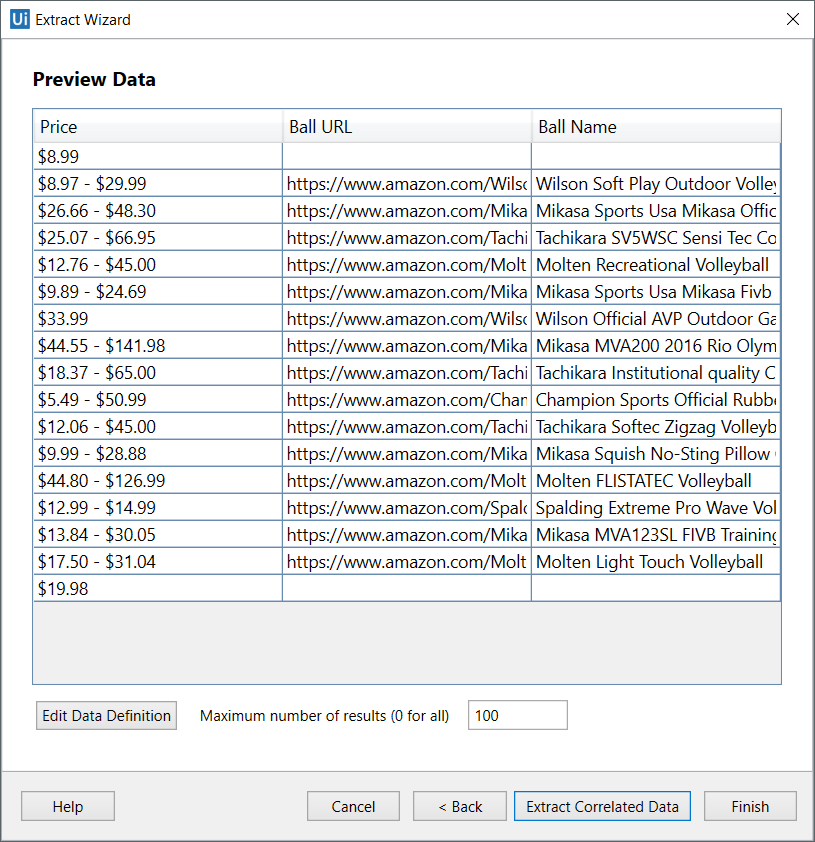
1. Open Internet Explorer and navigate to [www.amazon.com](http://www.amazon.com/).
2. In the search box type “volleyball ball” and press Enter. Results are displayed in the web page.
3. In Studio, on the **Design** tab, in the **Wizards** group, click **Data Scraping**. The **Extract Wizard** is displayed.
4. Following the wizard, select the first and last items in the web page. The **Configure Columns** wizard step is displayed.
5. Select the **Extract URL** check box.
6. Change the name of the column headers.



1. Click **Next**. A preview of the data is displayed and the fields you selected are highlighted in the web browser.



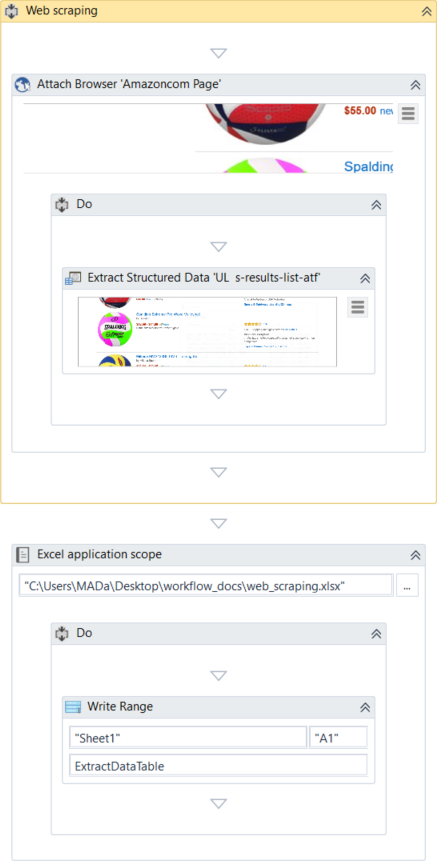
1. Click the **Extract Correlated Data**.The **Extract Wizard** starts again.
2. Following the wizard again indicate the prices of the items. You get to the **Configure Columns**step.
3. Change the name of the new column, and click **Next**. The data preview is displayed.



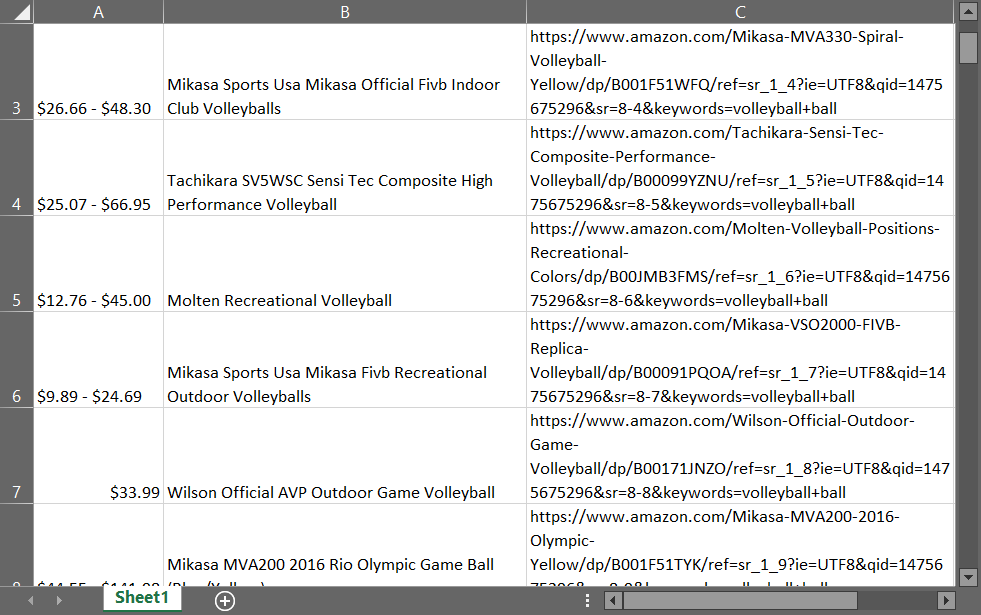
1. (Optionally) Change the order of the columns by dragging them in place.
2. Click **Finish**. The **Indicate Next Link** window is displayed prompting you to indicate the **Next** button if the spans more than one page.
3. Click **Yes** and select the **Next Page** button in Amazon. The workflow is saved and displayed in the **Main** panel. Note that a data table variable, **ExtractDataTable**, has been automatically generated.
4. Drag an **Excel Application Scope** activity under the **Data Scraping**container.

**Note:** Install the Excel activities package using the [**Manage Packager**](https://www.uipath.com/guides/managing-packages)to have access to these activities.

1. In the **Properties** panel, in the **WorkbookPath** field, type the file path of an existing Excel file to which you want to write the data.
2. In the **Variables** panel, change the scope of the automatically generated data table variable to **Main**.
3. In the **Excel Application Scope**, drag a **Write Range** activitiy.
4. In the **Properties** panel, in the **DataTable** field, add the **ExtractDataTable** variable. The final workflow should look as in the following screenshot.



1. Press F5. The workflow is executed.
2. Open the Excel file you used at step 15. Note that all columns are populated correctly.



[Click here to download this example.](https://www.uipath.com/hubfs/Documentation/WorkflowExamples/DataScraping.xaml?t=1492088952431)

SELECTORS

# About Selectors

To automate specific actions in the user interface, you are required to interact with various windows, buttons, drop-down lists and many others. Most applications do this by relying on the screen position of UI elements, a method that is not at all dependable.

To overcome this problem, UiPath uses what we call selectors. These store the attributes of a graphical user interface element and its parents, in the shape of an XML fragment.   
Most of the times, selectors are automatically generated by Studio and do not require further input from you, especially if the apps you are trying to automate have a static user interface.

However, some software programs have changing layouts and attribute nodes with volatile values, such as some web-apps. UiPath cannot predict these changes and therefore, you might have to manually generate some selectors.

A selector has the following structure:

<node\_1><node\_2>...<node\_N>

The last node represents the GUI element that interests you, and all the previous ones represent the parents of that element. <node\_1> is usually referred to as a root node.   
Each node has one or more attributes that help you correctly identify a specific level of the selected application.

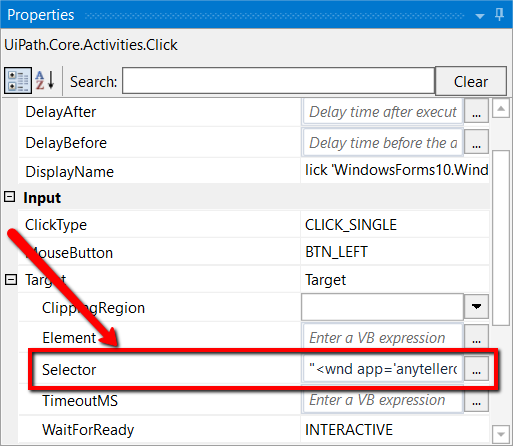
Each node has the following format:

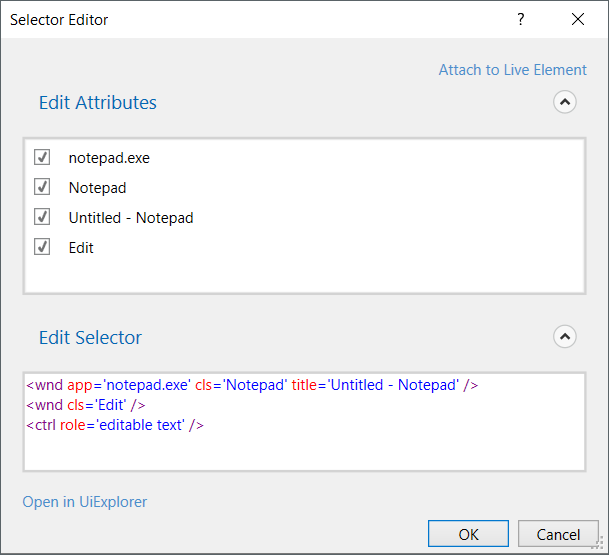
<ui\_system attr\_name\_1='attr\_value\_1' ... attr\_name\_N='attr\_value\_N'/>

Every attribute has an assigned value which represents a unique identifier.

You have to pick attributes with a constant value. If the value of an attribute changes, then the selector will not be able to correctly identify the element.

Selectors are stored in the **Properties** panel of activities, under**Input > Target > Selector**. All activities related to graphical elements have this property.

The **Selector Editor** window enables you to see the automatically-generated selector and edit it and its attributes. To open this window, click the Ellipsis Ellipsis button button next to the **Selector** field, in the **Properties** panel.

The**Edit Attributes** section contains all the application components needed to identify the target application (a window, a button etc.). The **Edit Selector** section holds the actual selector. Both of these sections are editable.

# Selectors with Wildcards

Wildcards are symbols that enable you to replace one or multiple characters in a string. These can be quite useful when dealing with dynamically-changing attributes in a selector.

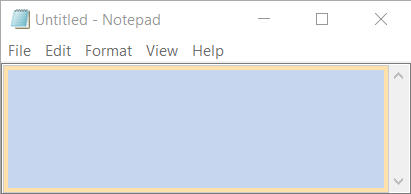
* Asterisk (\*) – replaces one or more characters
* Question mark (?) – replaces a single character

The Selector Editor enables you to automatically generate a selector with wildcards, by using the **Attach to Live Element Attach to Live Element button** button.

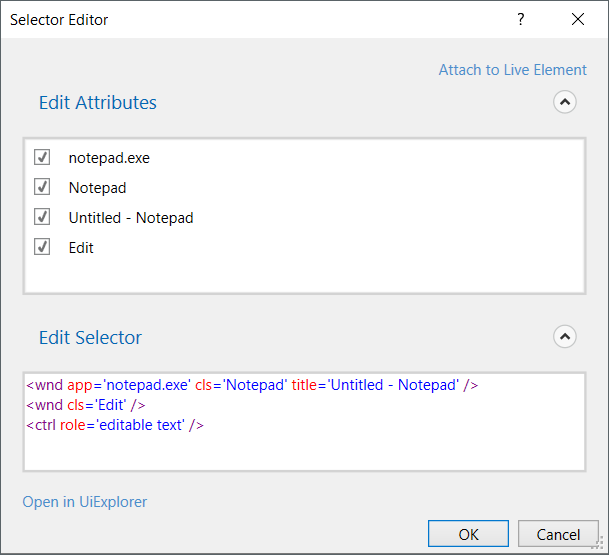
### Example of Generating a Selector with Wildcards in the Selector Editor Window

Part of the name of a Notepad window changes according to the .txt file you open with it. This is where a well-placed wildcard can really help. Do the following to generate it:

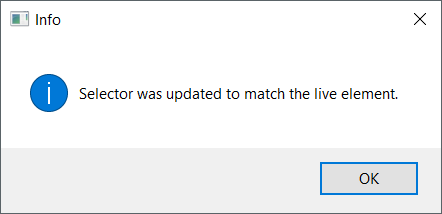
1. Open an empty Notepad window. Note that the window title is Untitled – Notepad.
2. In Studio, create a new sequence.
3. Drag a **Type Into** activity to the **Main** panel.
4. Click **Indicate on Screen** and indicate the editable text field in Notepad. A selector is automatically generated and stored in the **Selector** field.



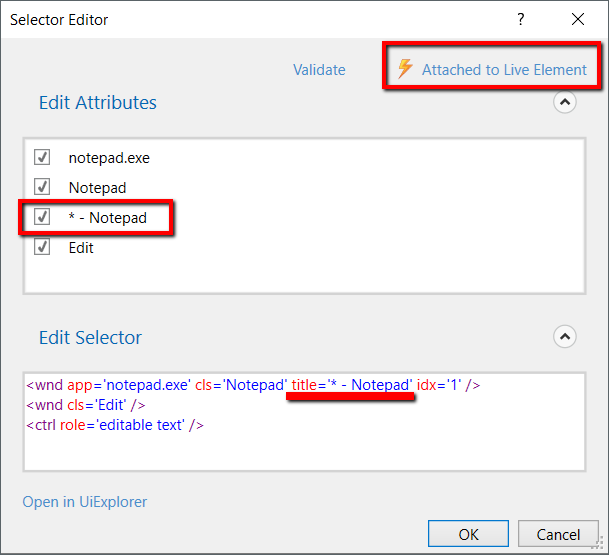
1. In the **Properties** panel, click the **Ellipsis Ellipsis button** button next to the **Selector** field. The **Selector Editor** window is displayed.



1. Open any .txt file with Notepad. Note that the window title is partially different than the one at step 1.
2. In Studio, in the **Selector Editor** window, click**Attach to Live Element** and indicate the **Notepad** window opened at step 6. A dialog box indicating that the selector was updated is displayed.



1. Click **OK**. The **Selector Editor** window and the selector are updated with a wildcard.



Full Versus Partial Selectors

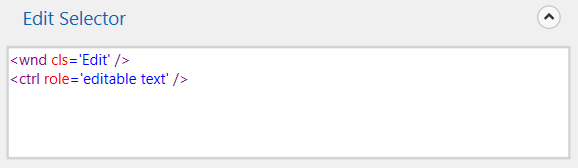
Full selectors:

* Contain all the elements needed to identify a UI element, including the top-level window
* Generated by the [Basic recorder](https://www.uipath.com/guides/about-recording-types)
* Recommended when switching between multiple windows

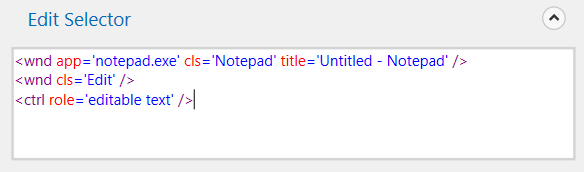
Partial selectors:

* Generated by the [Desktop recorder](https://www.uipath.com/guides/about-recording-types)
* Do not contain information about the top-level window
* Activities containing partial selectors are enclosed in a container (**Attach Browser** or **Attach Window**) that contains a full selector of the top-level window
* Recommended when performing multiple actions in the same window

Example of a partial selector for the editable panel in Notepad:



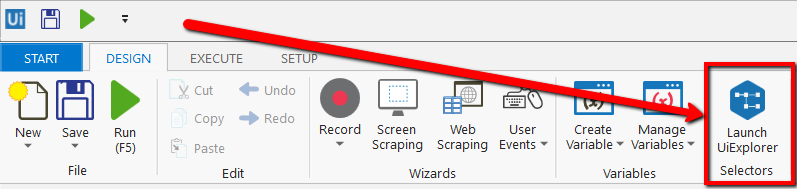
Example of a full selector for the editable panel in Notepad:



# UiPath Explorer

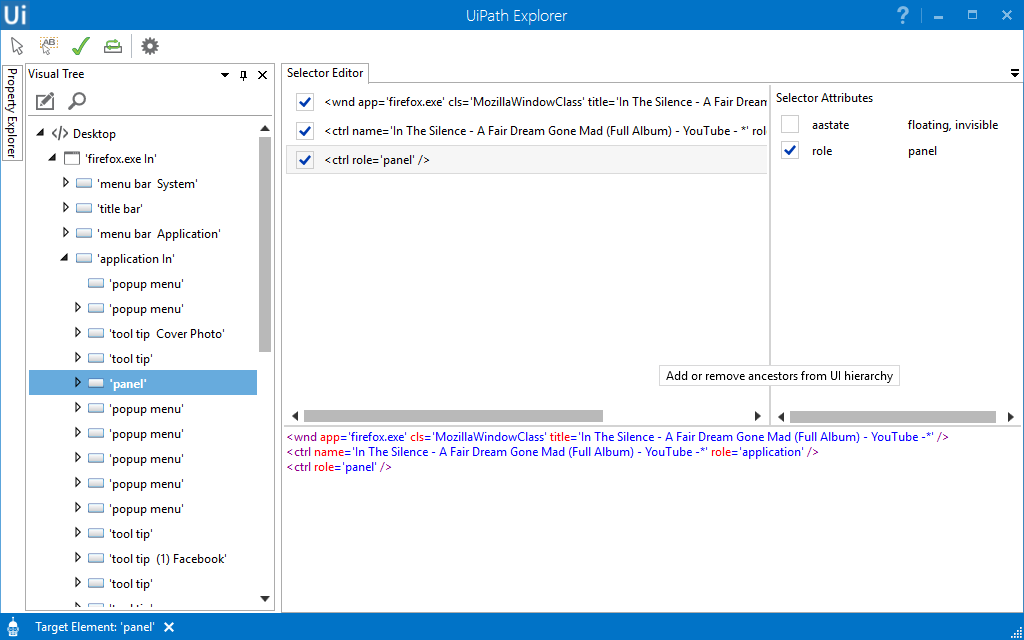
**UiPath Explorer** is an advanced tool that enables you to create a custom selector for a specific UI element.

To open the **UiPath Explorer** window, click **Launch UiExplorer** in the **Design** tab.



To be sure that you choose the best selector, remember to:

* Add or remove attributes
* Add parent or children tags
* Use wildcards to replace common values



### Field Descriptions for the UiPath Explorer Window

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Select Target Elementselect_target_element_button.png** | Automatically generate a selector by indicating a UI element on your screen. |
| **Select Relative Elementselect_anchor_element_button.png** | Enables you to find an anchor in a UI element, so that you can find a better selector. |
| **Validatevalidate_selector_button.png** | Verifies if your selector is valid or not. A dialog box is displayed letting you know about the result. |
| **Resetreset_button.png** | Resets all the panels from the UiPath Explorer window to their default states. |
| **UiFrameworksuiframeworks_button.png** | Changes the technology used to determine UI elements and their selectors. The following options are available:   * **Default** – UiPath proprietary method. Usually works fine with all types of user interfaces. * **Active Accessibility** – an eariler solution from Microsoft for making apps accessible. It is recommended that you use this option with legacy software, when the **Default** one does not work. * **UI Automation** – the improved accessibility model from Microsoft. It is recommended that you use this option with newer apps, when the **Default** one does not work. |

#### ****The Visual Tree Panel****

Displays a tree of the UI hierarchy and enables you to navigate through it, by clicking the arrows in front of each node.

By default, the first time when you open **UiPath Explorer** or after you click the **Reset**button, this panel displays all opened applications, in alphabetical order.

Double-clicking a UI element (or right-clicking and selecting **Set as Target Element**) from the tree, populates the **Selector Editor**, **Selector Attributes** and **Property Explorer** panels.

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Highlight highlight_button.png** | Highlights the target element and/or anchor. |
| **Show Search Optionssearch_button.png** | Displays the search box and search filter options. |
| **Search box** | Enables you to look for a specific string. If an exact match is not found, nodes containing the nearest match are displayed.  Wildcards (\*,?) are supported.  Depending on the attribute selected from the **Search by** drop-down list, the search can be case sensitive.  **Note:** The search only looks for matches in the tree structure under the selected UI object. |
| **Search by** | Filters your search to a selected attribute or a selector. The contents of this drop-down list change according to the selected UI element.  **Note:** If **Search by** is set to **Selector**, you can only input one node in the <attribute name1='value1' ... /> format. |
| **Children Only** | Limit your search to the first level children of the selected node. By default, this check box is not selected. |

#### ****The Selector Editor Panel****

Displays the selector for the specified UI object and enables you to customize it.

The top part of the panel displays the actual XML fragment that you have to use in a workflow. Once you find the selector you want, you can copy it from here and paste it in the **Properties** panel of an activity, in the **Selector** field.

The bottom part of this panel enables you to view all the nodes in a selector and eliminate the ones that are not necessary by clearing the check box in front of them.

Selecting a node here, displays its attributes in the **Selector Attributes** and **Property Explorer** panels.

#### ****The Selector Editor Panel****

Displays all the available attributes of a selected node (from the Selector Editor panel).

You can add or eliminate some of the node attributes by selecting or clearing the check box in front of each attribute.

Additionally, you can change the value of each attribute yet this modification is retained only if the new selector points at the originally selected UI object.

#### ****The Property Explorer Panel****

Displays all the attributes that a specified UI object can have, including the ones that do not appear in the selector. They cannot be changed.

SAP Automation

# How to Automate SAP Applications -without GUI Scripting

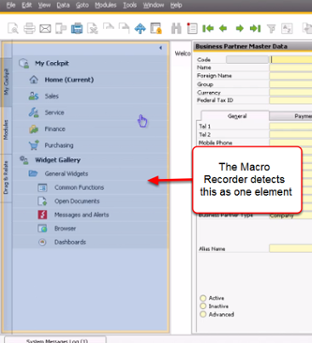
Quick Answers

[UI Automation](https://www.uipath.com/kb-articles/topic/ui-automation)

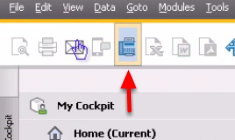
SAP (Systems, Applications and Products in Data Processing) is a well known enterprise software company that makes applications that manage business operations and customer relations.

## How does UiPath work with SAP?

There are elements on the **SAP GUI**, just like **Citrix**, that are detected by the recorder as a single block, so it's impossible to use a simple recording or automation tool with them. UiPath is well adapted and very capable of creating an automation, even with these types of interfaces. UiPath has a powerful screen scraping engine that helps you in extracting data from the GUI in less than a second! This method is 100% accurate.



Most elements of the **SAP GUI** can be detected by the **Macro Recorder** individually, like the Standard Buttons, Menus and Submenus.



### The Basic Workflow

Let's say you want to add sales or member data into the application database. Manually, you have to pull up the right form and then start inserting data manually.

In order to achieve this in UiPath, 2 GUI automation techniques will be used:

* automation through keyboard entries like **tab**, **arrow keys** and the **return** key
* automation by scraping text on the screen and clicking a text event

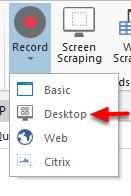
1. Automation through Keyboard entries makes it easier and faster to move between different text fields

2. Scraping the text allows us you to build **Activities** from individual elements in a block element.

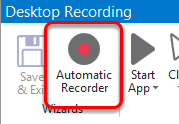
### The Process

Let's assume that the UiPath is installed on the same machine as the SAP application.

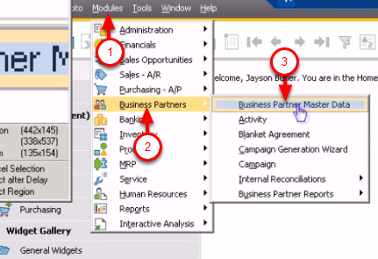
1. With the **SAP Business One** already pulled up, start building the **Workflow** by pulling up the **Macro recorder**. Select **Desktop**.



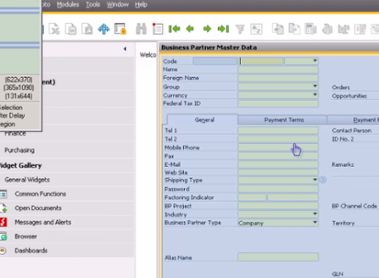
2. Select the **Automatic Recorder** from the Wizard.



3. To pull-up the data form, we can simply use a left mouse click, using the **Recorder**.

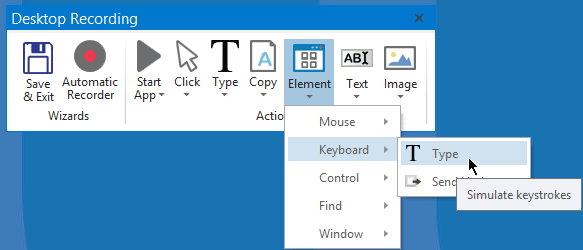


4. Now the form is ready. The individual elements in this form cannot be identified by the **Recorder**. You will use a different method to fill in the data. Press **Escape**. The previous activities are temporarily saved in the queue.

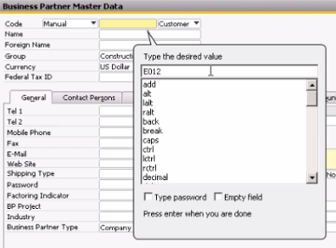


5. The next step is to start filling in the data. The cursor is now positioned in the place where you can start putting in the information ("Code").

Click on **Element->Keyboard->Type*.***

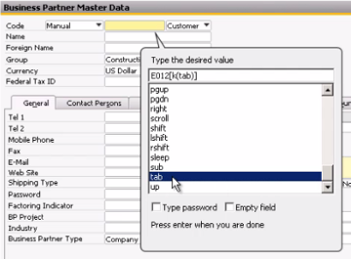


6. Just click anywhere in the form and it will initiate the input dialog box in the field where the cursor is in.



7. You can start by typing in initial data. Remember that you can use variables later to manipulate the data, so you can insert your own data sources.

The text dialog box also allows you to add keyboard commands such as **Return** and **Tab**. Let's add a **Tab** after the "Code" section. This will take you to the next field, which is a dropdown.



8. You can notice that it creates a code right next to the text you entered. It's the same thing you can apply when adding more keyboard commands. This is how you can manipulate the movement of the cursor:

Arrow Down: **[k(down)]**

Enter: **[k(enter)]**

Tab:**[k(tab)]**

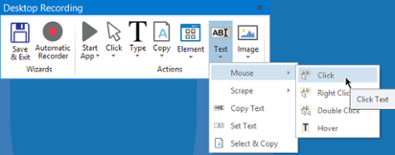
You can follow the same steps from step 5 to fill in any other input fields.

### Automating SAP via Click on Text and Screen Scraping

The keyboard commands are very helpful when moving to and manipulating input fields. If you want to jump to a different field from within the form it could be hard to do so using keyboard activities. That's where the UiPath built-in screen scraping comes in.

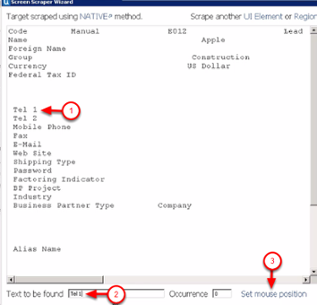
You have to run an OCR which reads the text in the form and captures the targeted text.

1. Click on **Text->Mouse->Click**.

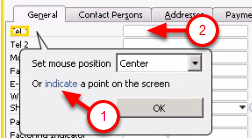


2. This will pull up the **Screen Scraping** wizard. The target field is next to "Tel 1". The cursor position should be set next to it.

Copy "Tel 1" and put copy it in the **Text to be found**field.



3. Click on **Set mouse position**. "Tel1" will be detected. Click on the **indicate**link, then click on the input field next to it.



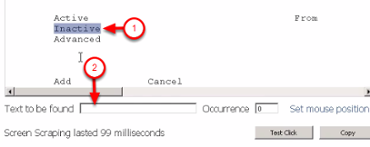
4. Click on the ***Continue***button . This will position the cursor next to "Tel1". You can use the same technique from step 5 (presented in the previous section) to continue filling out other data.

### Radio Buttons

https://www.uipath.com/hs-fs/hubfs/QuickAnswers/SAP15.png?t=1492088952431&width=73&name=SAP15.png

For the radio buttons, the same screen scraping technique can be used.

1. Click on **Text->Mouse->Click**and then copy paste "Inactive" into the **Text to be found field**. There is no need to set the mouse position for this one. The mouse will click in the middle of the element.



2. Use the same technique to click other GUI elements such as buttons

You are now ready to build your SAP automation out of these techniques.

