

LECTURE 07.

IMAGE AND TEXT AUTOMATION

Robotic Process Automation

[12 November 2019]

Elective Course, 2019-2020, Fall Semester

Camelia Chisăliță-Crețu, Lecturer PhD

Babeș-Bolyai University

Acknowledgements

This course is presented to our Faculty with the support of UiPath Romania.



Contents

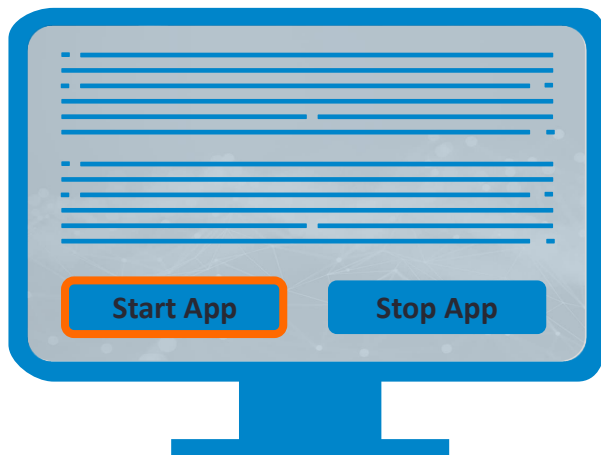
- **Image and Text Automation**
 - Motivation. Overview. Details
- **Image Automation**
 - Overview. Example
 - **Click Image Activity**
 - Details. Properties
 - **Click Text Activity**
 - Details. Properties
 - **Demo 1.** Click Image and Click Text
 - **Image Activities**
 - Details. Types
 - **Find Activity**
 - Overview
- **Keyboard Automation**
 - Details. Overview. Related Activities
 - **Demo 2.** Keyboard Automation
- **Information Retrieval**
 - Details. Functionalities
 - **Select&Copy Activity**
 - Details
- **Relative Scraping**
 - Details. Example
- **Demo 3.** Information Retrieval
- **Native Citrix Information**
 - Challenges. Details
- **When and Where to perform certain actions**
- **Best Practices**
- **References**

Image and Text Automation. Motivation

- **Image and Text automation** are used when
 - **selectors cannot be found by using regular methods;**
- E.g.: various applications like:
 - ones hosted on Citrix, Citrix Desktop,
 - scanned PDF, SAP;
- these are environments where selectors are not exposed by the application;
- **the scripting that allows selector generation is disabled.**

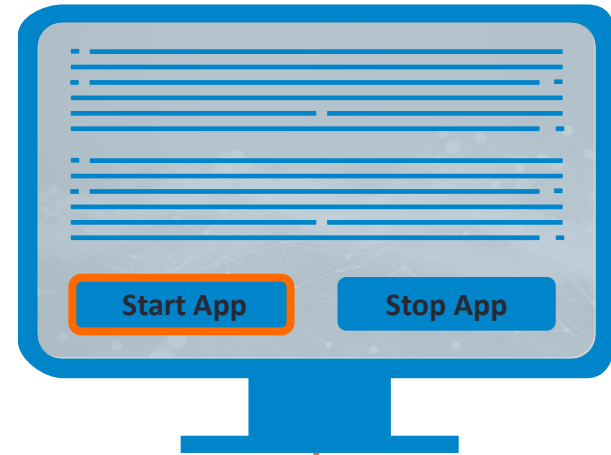
Image and Text Automation. Overview

Image based automation



- it relies on **user interface element recognition**.

Text based automation



- it relies on **identifying the text from certain user interface elements**.

Image and Text Automation. Details

- UiPath Studio provides activities for:
 - simulating **mouse input**: such as clicking, hovering or typing, text recognition and OCR (optical character recognition);
- these activities use:
 - **image recognition that work directly with images to identify UI elements**:
 - it can simulate human behavior by using images as a means of identifying UI elements;
 - **screen scraping to identify UI elements**:
 - it is able to extract text from running applications even when they are hidden or covered by another applications.

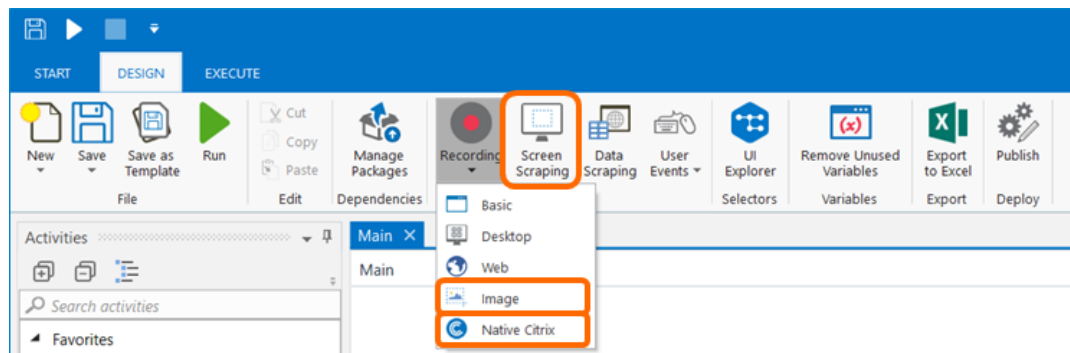


Image Automation. Overview

- **Image recognition** activities
 - can simulate human behavior by using images as a means of identifying UI elements;
 - are very useful in situations where **human behavior must be mimicked**.

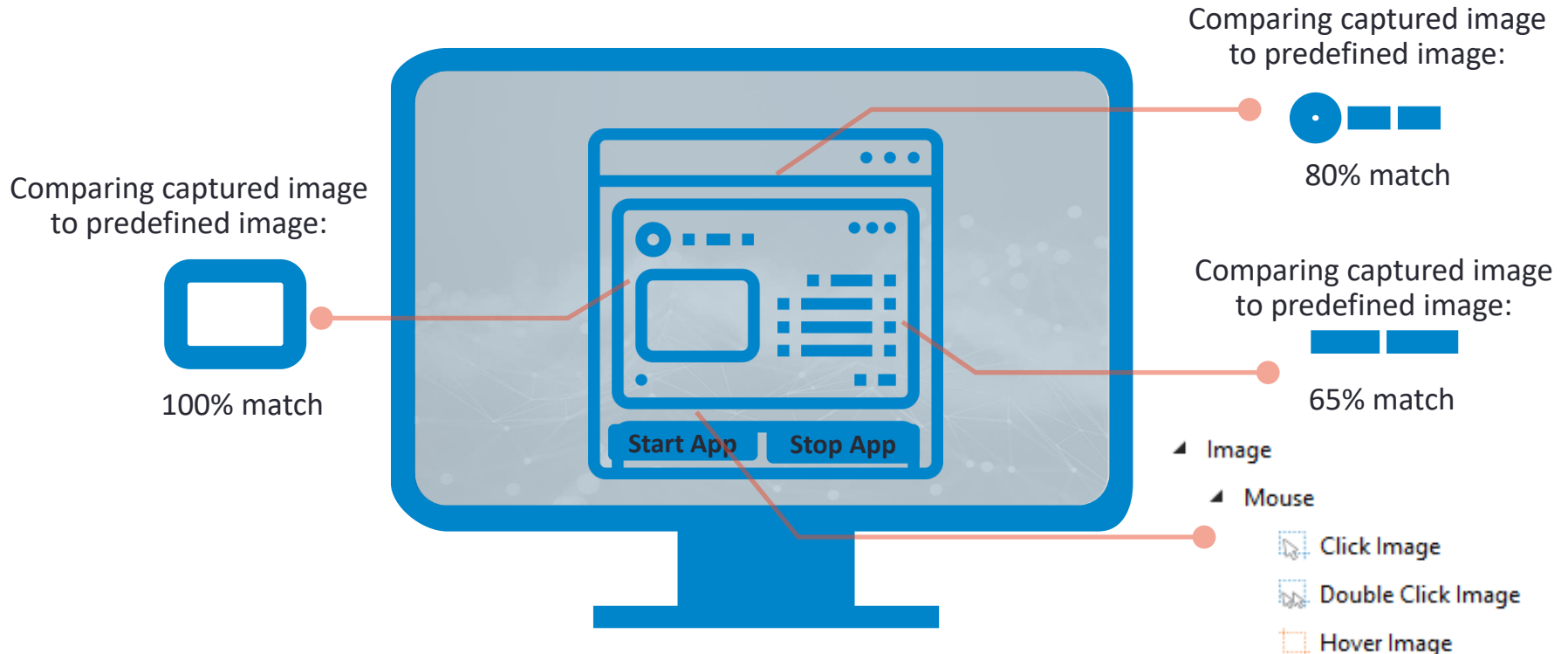
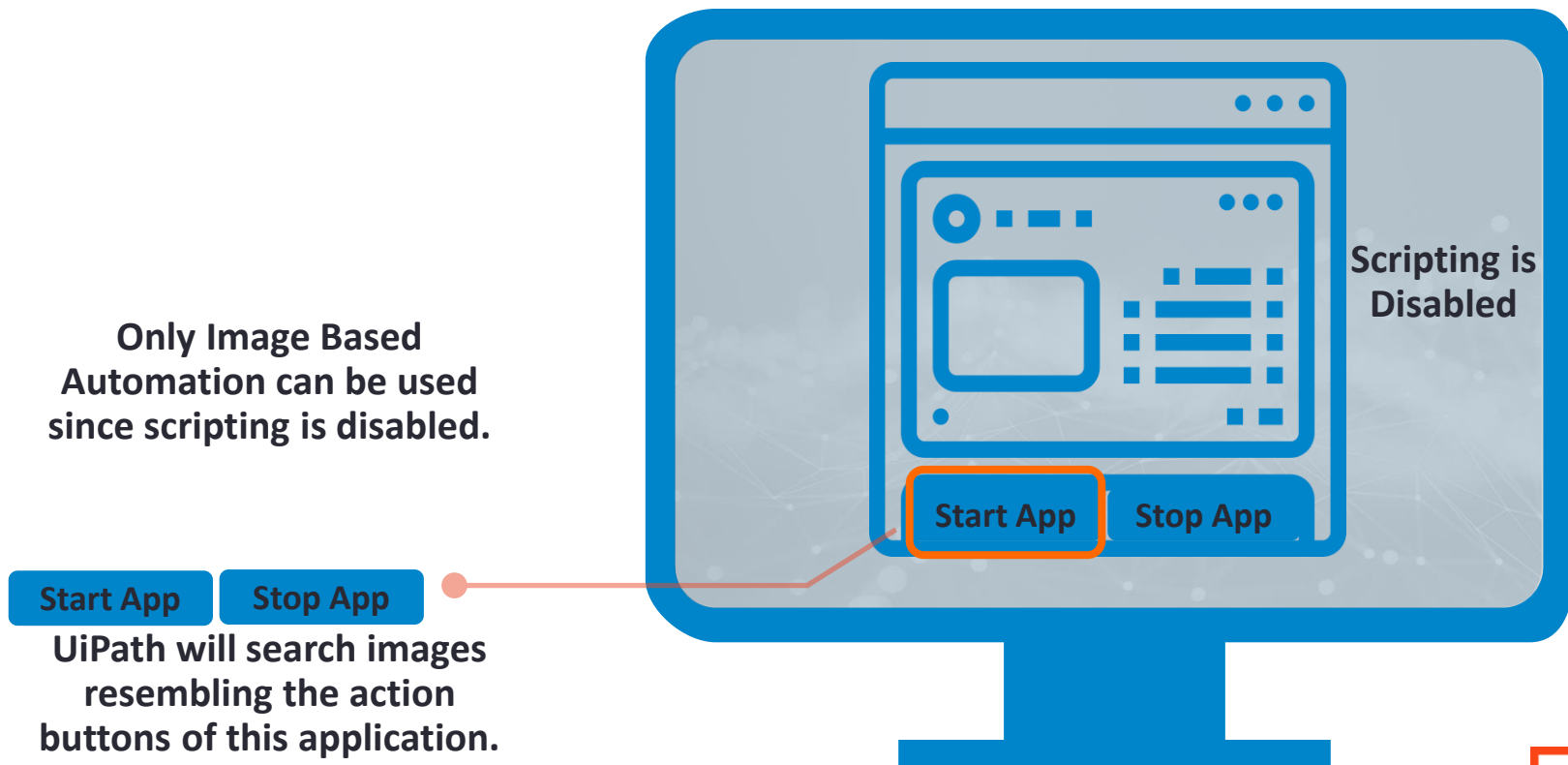


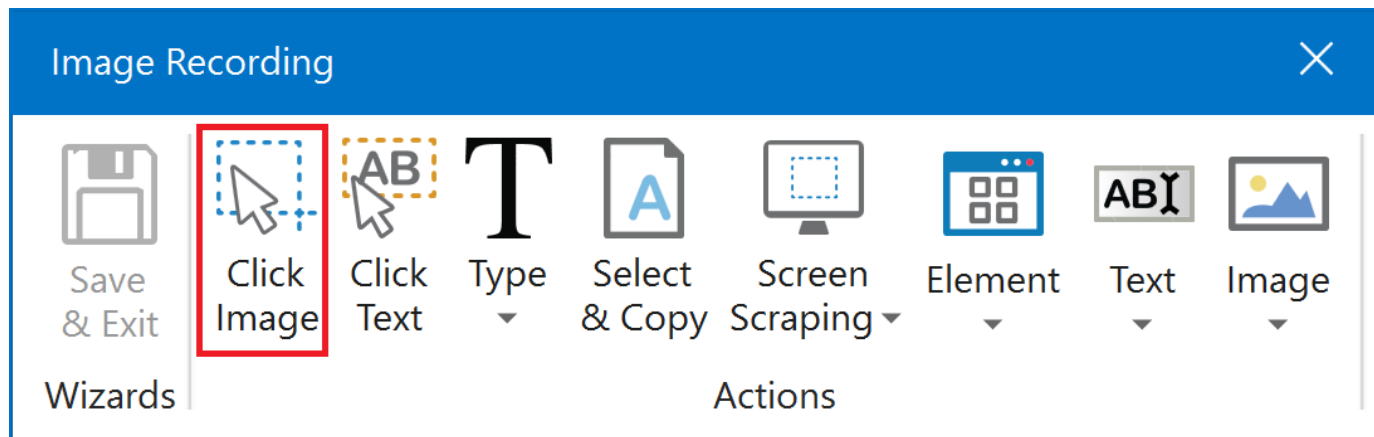
Image Automation. Example

- Image-based automation can be deployed in cases where scripting is not enabled.



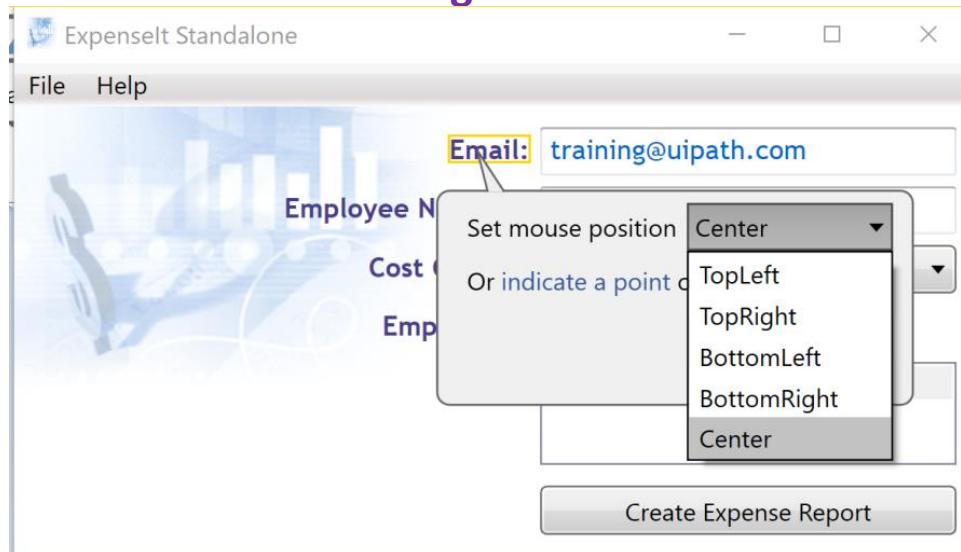
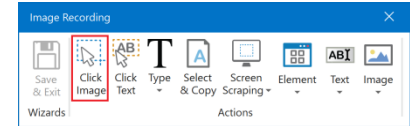
Click Image Activity. Details

- **Click Image** activity
 - allows to click on any selected area as long as this is unique on the screen/ virtual environment;
 - it scans the screen of the machine for UI elements which appear at random positions and return UI Element variables that have the clipping region set to the found element;
- **disadvantages:** such activities are fast and reliable, but sensitive to graphical variations, as they can fail if colors or background details change;
- different properties allow to customize the image recognition during execution.



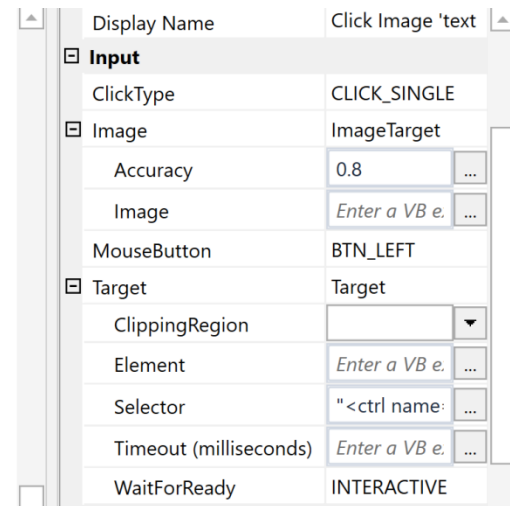
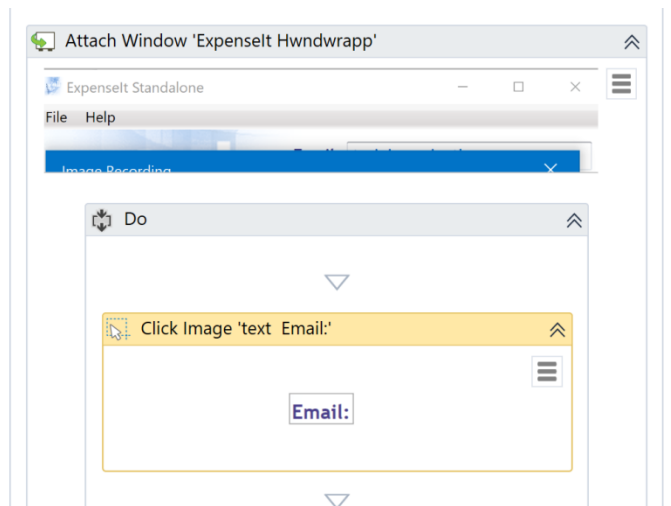
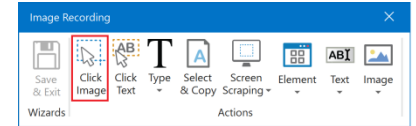
Click Image Activity. Properties (1)

- **Click Image** activity
 - allows to click on any selected area as long as this is unique on the screen/ virtual environment;
- properties permit to
 - **set the position of the mouse where to perform click:**
 - *inside* the selected area: **Center, Top Left, Top Right, etc.;**
 - *outside* the selected area: **indicating an area or a point** where to perform click;
 - **this is useful when there is a single element to match to on the image;**



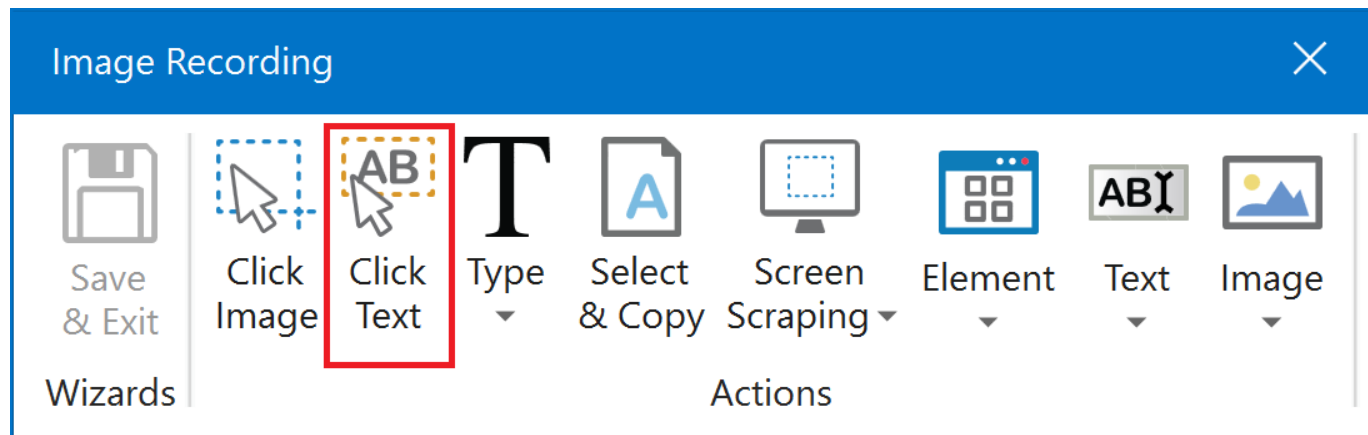
Click Image Activity. Properties (2)

- **Click Image** activity
 - allows to click on any selected area as long as this is unique on the screen/ virtual environment;
- properties permit to
 - **configure the accuracy of the matching action:**
 - **0.80** = good value, it allows a compromise between accuracy and reliability;
 - **1.00** = the image must match 100% to be registered as found;
 - **this is useful if graphical elements searched for may be slightly different;**



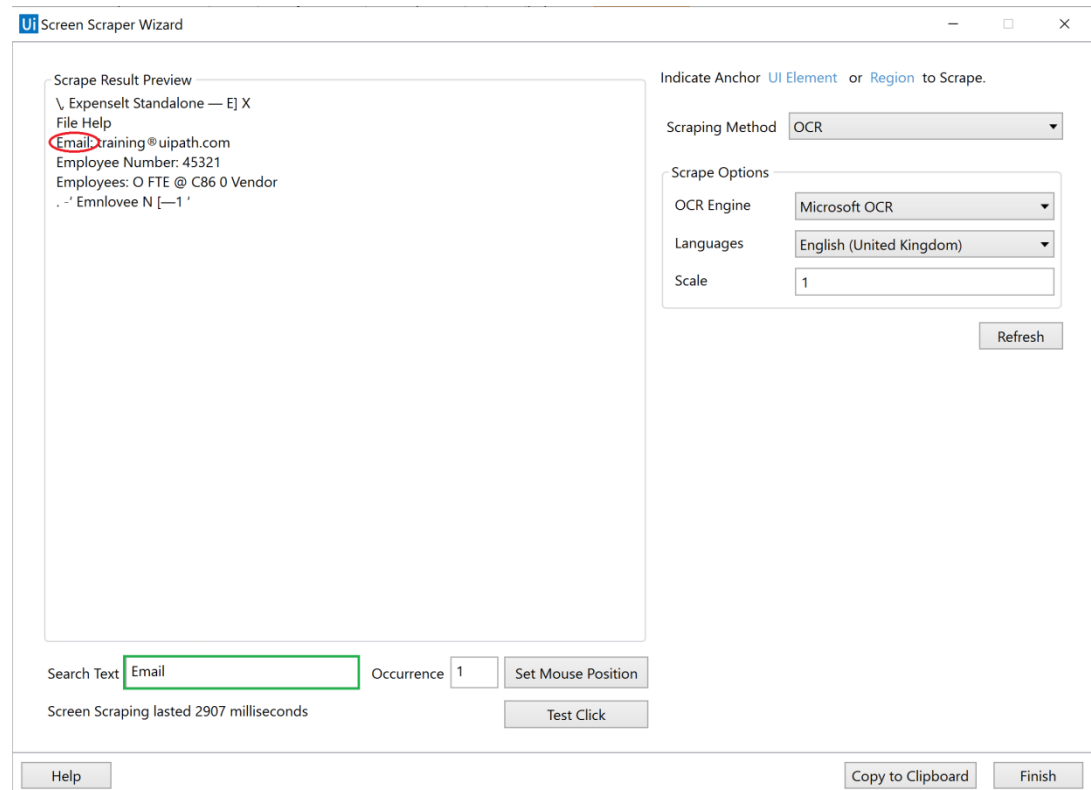
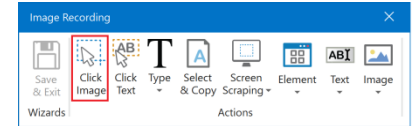
Click Text Activity. Details

- **Click Text** activity
 - allows to use an OCR engine to scan for a specific text available on the screen/virtual environment;
 - the window is similar to the one of Screen Scraping with some particularities for the mouse actions;
 - useful when window theme is different or the text is different, but is the same;
- **disadvantages: speed, OCR accuracy and its settings;**
- the properties refer mainly to the OCR engine used to recognize the text during execution.



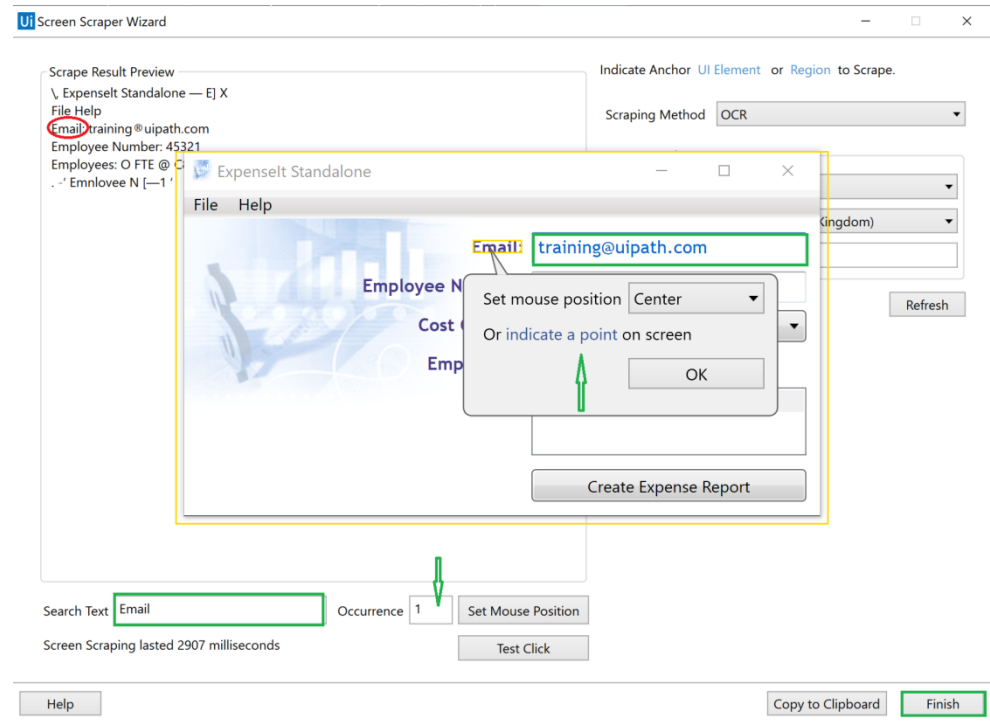
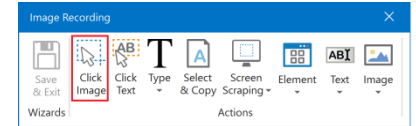
Click Text Activity. Properties (1)

- **Click Text** activity
 - allows to use an OCR engine to scan for a specific text available on the screen/virtual environment;
- properties permit to
 - **set the text searched for:**
 - the user specifies the text to perform click on;



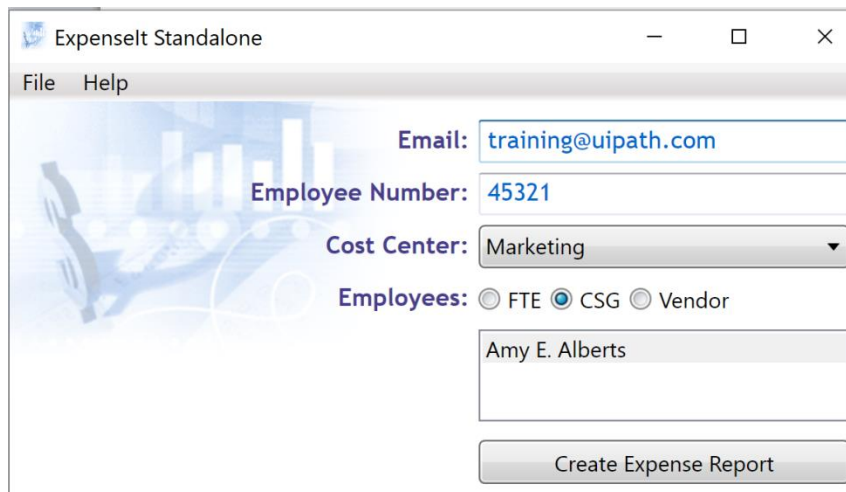
Click Text Activity. Properties (2)

- **Click Text** activity
 - allows to use an OCR engine to scan for a specific text available on the screen/virtual environment;
- properties permit to
 - **configure the occurrences of the text searched for:**
 - the user specifies the number of occurrences and the position of the mouse (*inside, outside the found text*);



Demo 1. Click Image and Click Text

- Use the **Click Image** and **Click Text** activities to fill in the form required to generate a report;
 - application **Expenselt**;
 - Email: myEmail@company.com
 - Employee Number: 12345;
 - Cost Center: Marketing;
 - Employees: CSG;
 - select the employee Amy E. Alberts;



The screenshot shows a window titled "Expenselt Standalone" with a menu bar containing "File" and "Help". The main area features a light blue background with a faint bar chart and a hand holding a pen. The form contains the following fields and controls:

- Email:** A text box containing "training@uipath.com".
- Employee Number:** A text box containing "45321".
- Cost Center:** A dropdown menu with "Marketing" selected.
- Employees:** Radio buttons for "FTE", "CSG" (which is selected), and "Vendor".
- Employee Selection:** A list box containing the name "Amy E. Alberts".
- Create Expense Report:** A button at the bottom of the form.

Image Activities. Details

- Find Image activities
 - waits for specific UI components to appear;
- steps:
 - 1. it presents the model of UI element image by the user to be searched;
 - 2. once the UI component appears, the image activity with the UI Element variable and clipping image set region;
- the image activity is a useful tool that identifies the UI element, variable and component in the virtual machine;
- it provides the best decision choosing option that image is displayed or not apart from this can it managed manifest actions;
- it enables to take and make decisions in image activities;
- it uses **Retry Scope** activity to handle the condition of image activity.

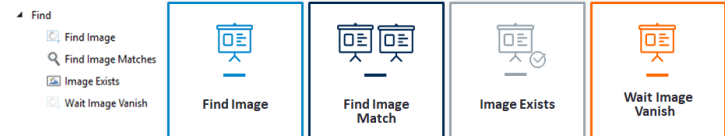
Find

- Find Image
- Find Image Matches
- Image Exists
- Wait Image Vanish







Image Activities. Types

- Find Image activity
 - is a major part of Image Activities that waits for the images to appear in UI elements. When this process is completed, it changes the UI element image with the clipping region set. After that, it gives an element along with UI elements;
 - source code: `UiPath.Core.Activities.WaitImageAppear;`
- Find Image Match activity
 - it searches and matches a particular image in a UI element and delivers the collection of UI element;
 - in the collection, it keeps the clipping region set of matching screens.
 - source Code: `UiPath.Core.Activities.FindImageMatches;`
- Image Exists activity
 - it specifies any images found with individual UI element;
 - source Code: `UiPath.Core.Activities.ImageFound;`
- Wait Image Vanish activity
 - it waits for the image when it disappears from UI element;
 - source Code: `UiPath.Core.Activities.WaitImageVanish.`



Find Activity. Overview

Find

-  Find Image
-  Find Image Matches
-  Image Exists
-  Wait Image Vanish



Find Image



Find Image Match





Image Exists



Wait Image Vanish



Event

-  On Image Appear
-  On Image Vanish



Monitoring Events

File

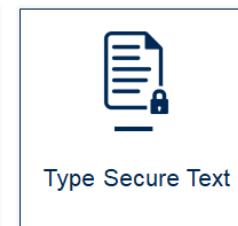
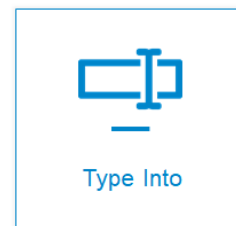
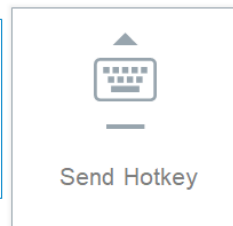
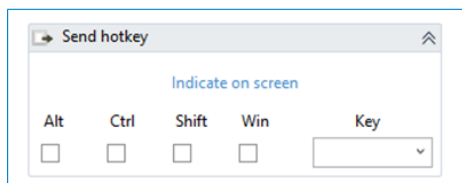
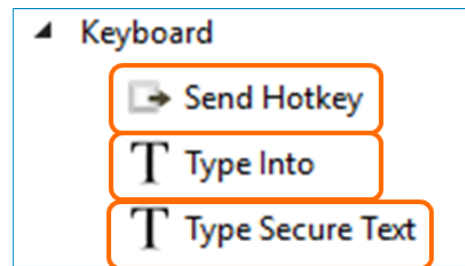
-  Load Image
-  Save Image



File Activities

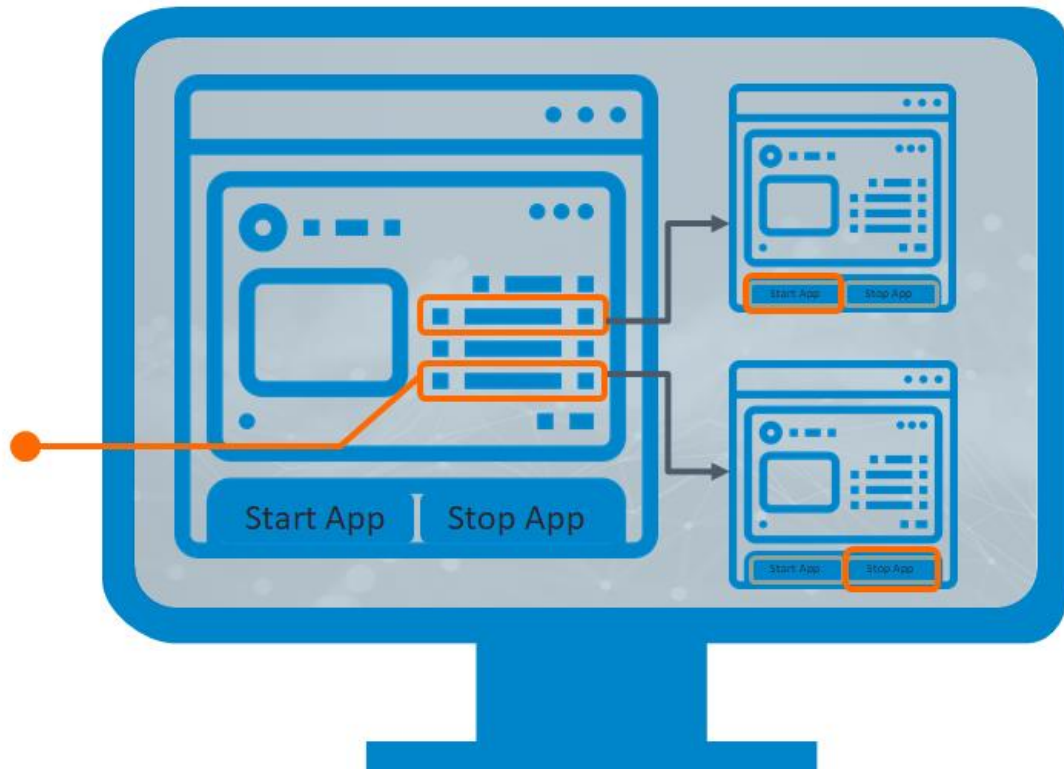
Keyboard Automation. Details

- UiPath Studio provides activities for:
 - simulate any type of **keyboard** input that a regular human user would use.
- the activity passes the UI element variable by input text;
- it identifies the action that is automated; during this process the target can automatically create screen functionality that identifies the UI elements in a proper region and produces selectors for them;



Keyboard Automation. Overview

With the use of hotkeys and shortcuts, specific actions can be set to reduce the number of steps performed.



Keyboard Automation. Related Activities

- activities that make the keyboard-based automation easier:
 - Send HotKey** activity
 - it is used to access the UI element application shortcut and to simplify the automation project;
 - E.g.: it can replace multiple activities that perform UI automation with the help of keyboard shortcuts;
 - these shortcuts contain a longer path to proceed with the automation process.
 - Type Into** activity
 - it sends the keystrokes to an UI element;
 - it supports the select and drop-down list of variables and components.
 - Type Secure Text** activity
 - it sends a secure string to a UI element which is useful for secure automation;
 - it uses passwords that are stored in SecureString.



Demo 2. Keyboard Automation

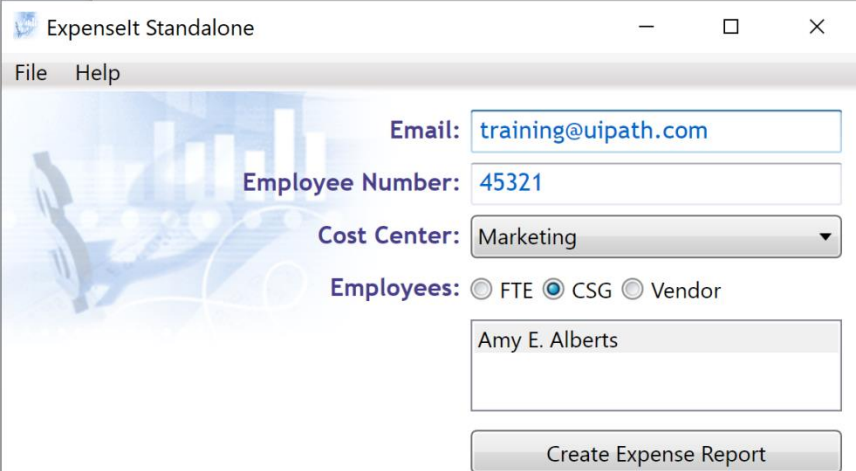
- Use the **keyboard automation** activities to fill in the form required to generate a report;

- application **Expenselt**;
 - Email: myEmail@company.com
 - Employee Number: 12345;
 - Cost Center: Marketing;
 - Employees: CSG;
 - select the employee Amy E. Alberts;

- possible sequence of keys:

- "[k(end)d(shift)d(home)u(shift)u(home)]myEmail@comany.com[k(tab)]"
+"[k(end)d(shift)d(home)u(shift)u(home)]12345[k(tab)][k(down)][k(tab)]
[k(right)][k(tab)][k(down)][k(tab)][k(enter)]"

- "[k(end)d(shift)d(home)u(shift)u(home)]" allows to select the entire content of the text field.

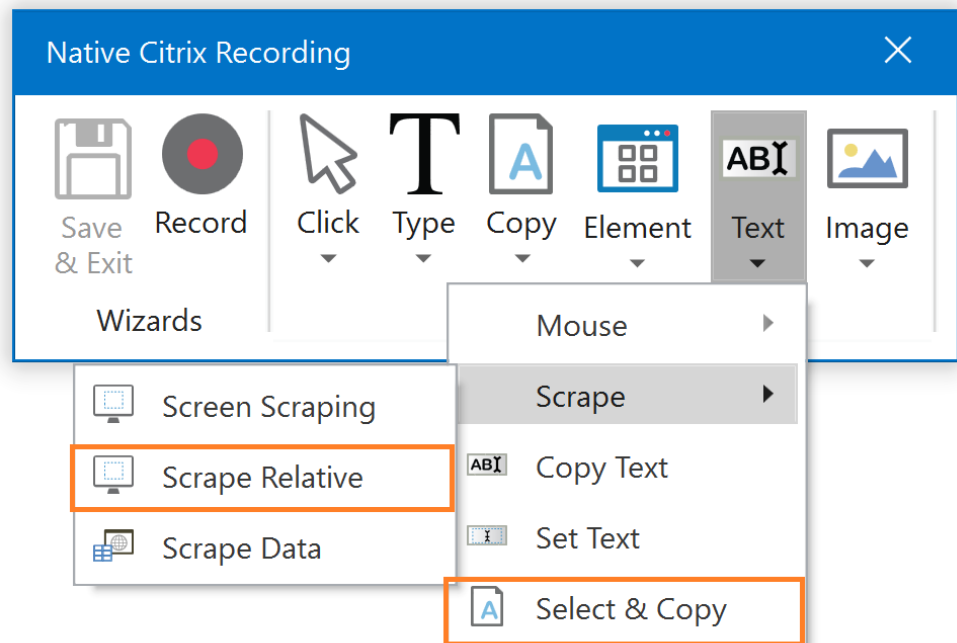


The screenshot shows a window titled "Expenselt Standalone" with a menu bar containing "File" and "Help". The main area contains a form with the following fields and controls:

- Email:** A text box containing "training@uipath.com".
- Employee Number:** A text box containing "45321".
- Cost Center:** A dropdown menu currently showing "Marketing".
- Employees:** Three radio buttons labeled "FTE", "CSG" (which is selected), and "Vendor".
- Employee Name:** A text box containing "Amy E. Alberts".
- Create Expense Report:** A button at the bottom right of the form.

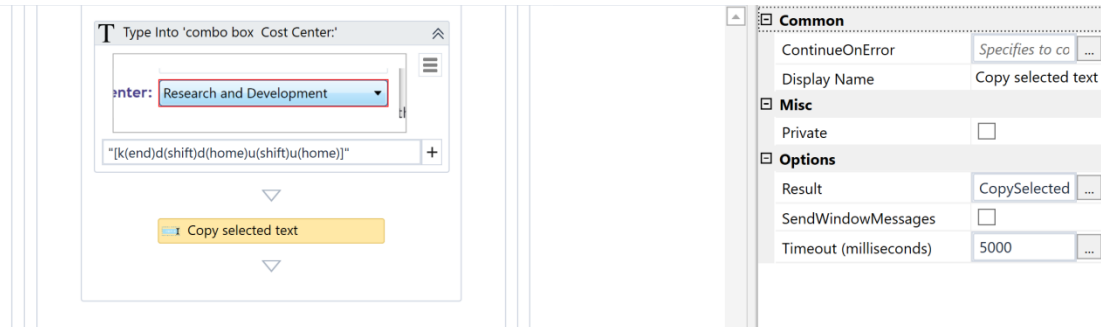
Information Retrieval. Details

- **Information retrieval**
 - out of an application accessed by a Virtual Machine (VM) can be performed by:
 - **copy the data from an editable text** or
 - use Optical Character Recognition (**OCR**) to **extract the relevant information**;
 - actions involved:
 - **Select&Copy;**
 - **Relative Scraping;**
- related actions:
 - **Copy Text;**
 - **Screen Scraping.**



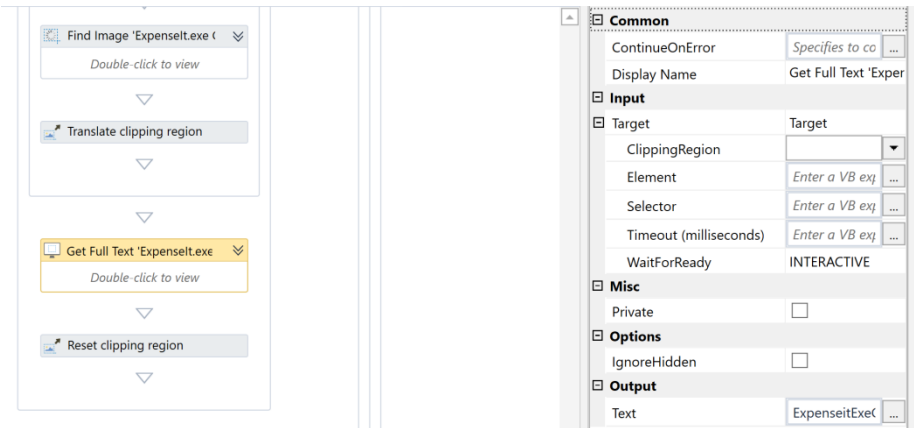
Select&Copy Activity. Details

- **Select&Copy** wizard
 - selects editable texts and copies it to the UiPath environment through the clipboard;
 - **all action is performed on the active text field;**
 - two activities are generated by the wizard:
 - **Type Into** activity:
 - it this is the command used to select the text:
"[k(end)d(shift)d(home)u(shift)u(home)]";
 - **Copy Selected Text** activity:
 - it involves copying the text activity from the target UI to the UiPath environment;
 - the copied text appear as **GenericValue** variable;
- this can be combined with any of the three input methods (default, send messages, simulate type/click).



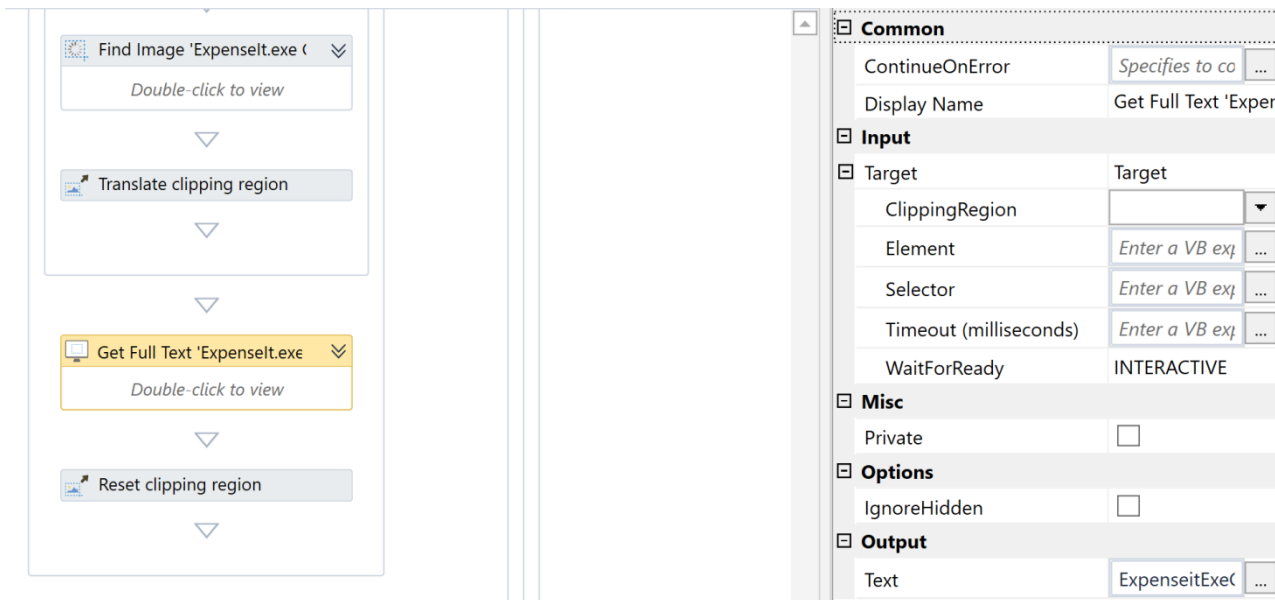
Relative Scraping. Details

- **Relative Scraping** wizard
 - scrapes just a portion of an image relative to an anchor;
 - it locates a fixed element on the screen and then using OCR extracts the information;
 - four activities are generated by the wizard:
 - **Find Image** activity:
 - used to find the anchor image;
 - **Translate Clipping Region** activity:
 - used to find the clipping region, offset to the anchor image;
 - **Get OCR Text** activity:
 - used to extract the data from the clipping region;
 - other **Get Text**-based activities may be used according to the output methods used, **FullText** or **Native**;
 - **Reset Clipping Region** activity:
 - used to avoid interference with other operations.



Relative Scraping. Example

- E.g.:
 - using OCR, it can be deployed information retrieval for customers that handle a large flow of documents such as Purchase Orders (PO);
 - or, someone can manually scan them and upload them on SAP;
 - each PO can be scanned out of millions of documents and details can be extracted like **Invoice Number, Date, Tax, Total**, and so on;
 - the information can be organized in specific fields for further storage and handling.



Demo 3. Information Retrieval

- Use the **information retrieval** wizards to get information out of the created expense report;
 - application **Expenselt**;

Create Expense Report

Email Alias: Someone@example.com

Employee Number: 57304

Cost Center: 4032

Expense Type	Description	Amount
Meal	Mexican Lunch	12
Meal	Italian Dinner	45
Education	Developer Conference	90
Travel	Taxi	70
Travel	Hotel	60

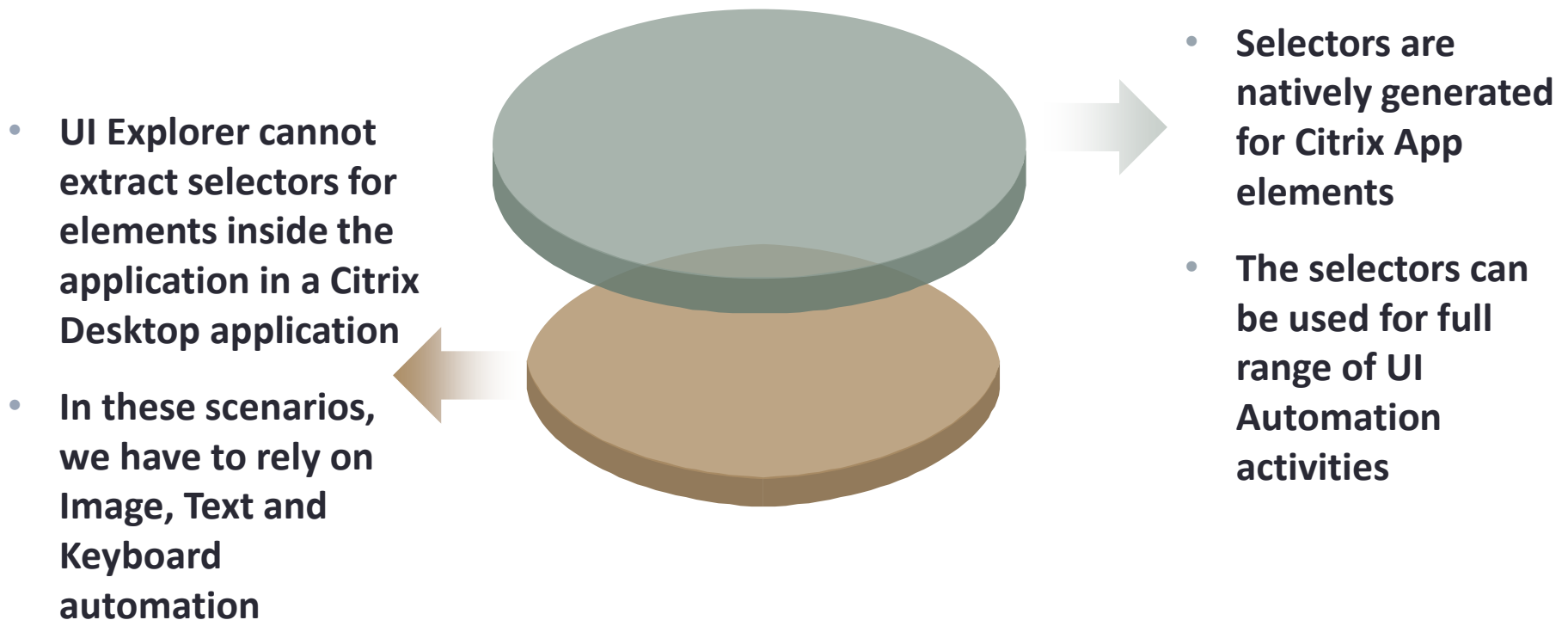
Add Expense

View Chart

Total Expenses (\$): 277

OK Cancel

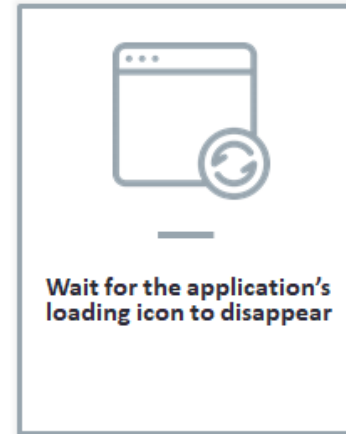
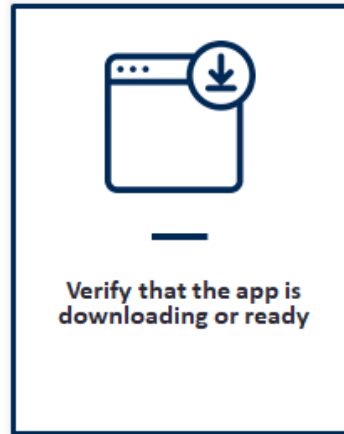
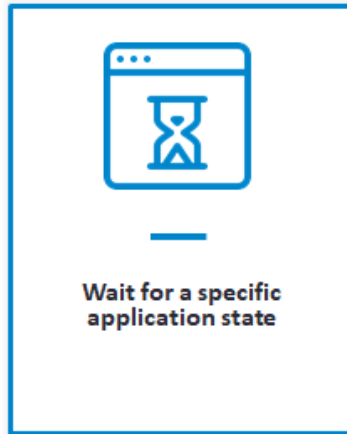
Native Citrix Automation Challenges



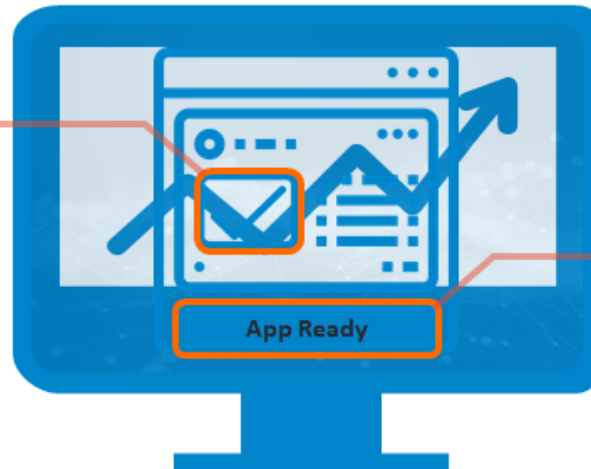
Native Citrix Automation. Details

- from UiPath v2018.4.1 onwards has native support for **Citrix Virtual applications** (formerly known as XenApp);
- by installing the **Citrix Extension** on the client machine and the UiPath Remote Runtime component (UiPathRemoteRuntime.msi) on the Citrix Virtual Apps application server, we can automate Citrix applications as if they were local applications;
- selectors are natively generated for a Citrix application elements, and we can use the full range of UI Automation activities, such as **Click**, **Type Into**, **Get Text**, **Extract Data**, etc.
- however, this is not the case for **Citrix Virtual Desktop** (XenDesktop), UI Explorer is unable to extract selectors for elements inside the applications in a Citrix Desktop application;
- the same case is with **Citrix Virtual applications** where **Citrix Extensions** are not enabled;
- in these scenarios, we have to rely on **Image**, **Text** and **Keyboard** automation.

When and Where to Perform Certain Actions (1)



Visual Anchor Example



Visual Anchor Example

When and Where to Perform Certain Actions (2)

- Automation can be fully used in cases where the errors in the automating project are minimal;
- E.g.:
 - when waiting for a particular state of an application, it is essential to create optimal automation that waits for the application or a web page to load or for a specific process to end; at this point, it is important to look and wait for the **visual element** to appear by keeping the **On Image Appear** activity on it.
 - a better and more general solution is to wait for the application's loading icon to disappear; an **On Image Vanish** activity can be used for this purpose, allowing the automation to continue **only when the loading icon vanishes**.

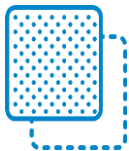
When and Where to Perform Certain Actions (3)

- Automation can be fully used in cases where the errors in the automating project are minimal;
- E.g.:
 - certain activities expect control to be already present in the normal field;
 - sending data to the application or extracting data from the application are some of such scenarios where we can get data by using click activities as:
 - **Click Image** activity and **Click Text** activity;
 - make use of clicking on an **anchor image relative to the text**;
 - make use of **keyboard** activities like sending **tab** keys.

When and Where to Perform Certain Actions (4)

- Automation can be fully used in cases where the errors in the automating project are minimal;
- E.g.: Identifying Elements
 - virtual environments do not provide any way to identify UI elements through standard means;
 - therefore, **visual anchors are only the options**;
 - **image** or **text-based** activities to extract text to compare with the expected result;
 - UiPath Studio provides activities that use OCR or Image Recognition technologies in such situations:
 - *OCR engines*: **Google Tesseract** (for scraping smaller areas), **Microsoft MODI** (more suitable for larger areas), and **Abbyy**.

Best Practices



Be sure the application is in the foreground



Use shortcut keys to launch applications



Be in the correct state of the application



Use shortcut keys to navigate



Use find image/text to look for elements



Exclude icon background

Best Practices (2)

- applications are usually opened by **clicking their shortcut** or **executable file**;
- the location of these files can normally be identified by several means, such as **screen coordinates** or **selectors**;
 - in virtualized environments the locations of shortcuts or executables are unavailable; **therefore, Image and OCR activities must be used to identify the location of the shortcut or executable file.**

Best Practices (3)

- these activities are based on image and text recognition and, slight graphical differences, such as **changes in resolution** or **highlighting the icon**, can cause the identification of the shortcut to fail;
 - a solution for this issue is to select an area of the icon that does not include any portion of the *background image*, such as **the center area of the icon**;
 - the best way to open an application in the virtual environment is to create a shortcut for the respective applications;
 - this can be further simplified by assigning the hotkey;
 - it should be made by using a complex key for the combination; using the **Command prompt**, the robot can start the applications in virtual environments;
 - the user can send the path of the application to the **Command Prompt** terminal with the **Send Hotkey** and **Type Into** activities; this method also enables to input arguments for the application to be opened.

References

- UiPath Academy - <https://academy.uipath.com>
 - Level 1 – Foundation Training, Lesson 7 an Lesson 8;
- UiPath Docs - <https://docs.uipath.com/studio>
 - Image and Text Automation - <https://docs.uipath.com/studio/v2018.3/docs/about-image-and-text-automation>