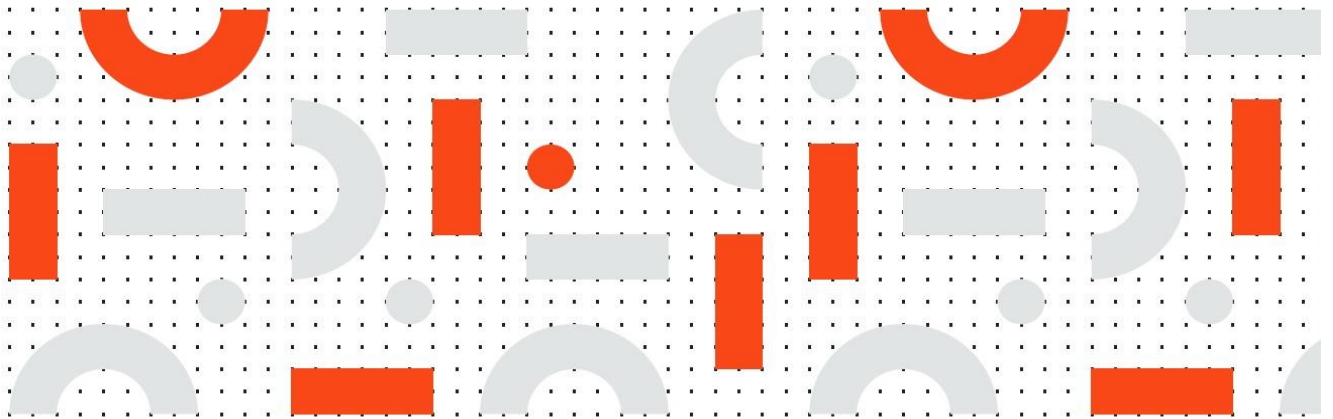


RPA Design & Development V2.0

Student Manual






Welcome to 'RPA Design and Development Course'.

Lesson 7: Automation Concepts and Techniques

The seventh lesson of this course is Automation Concepts and Techniques.

Agenda




01

Extraction and Its Techniques

02


Automation Techniques



The agenda of this lesson is:

- Extraction and Its Techniques
- Automation Techniques

Learning Objectives




01

Explain extraction and apply its techniques

02

Explain and use automation techniques

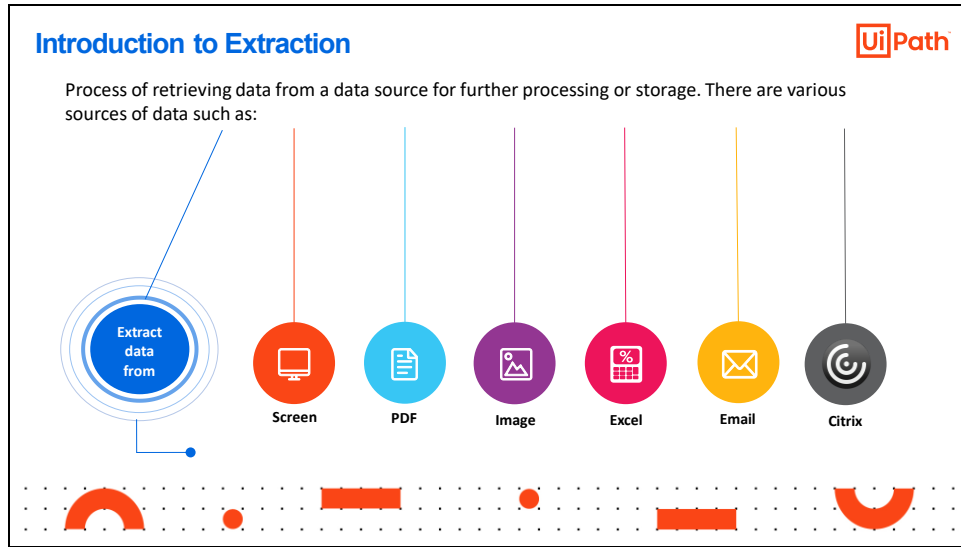


By the end of this lesson, you will be able to:

- Explain extraction and apply its techniques
- Explain and use automation techniques

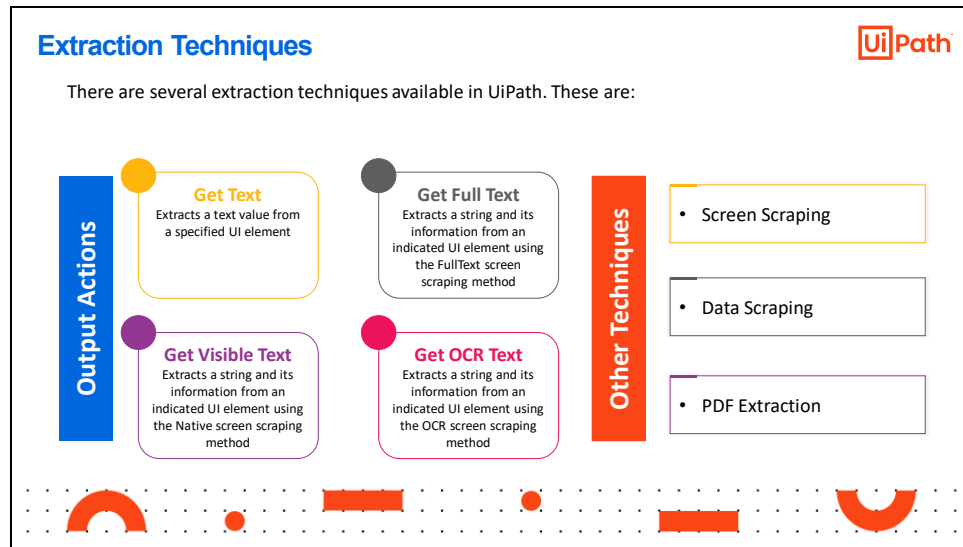


This section gives an introduction to extraction and its techniques.



Process of retrieving data from a data source for further processing or storage. There are various data sources from where data can be extracted such as:

- Screen
- PDF
- Image
- Excel
- Email
- Citrix, etc.



There are several extraction techniques available in UiPath. These include:

Output actions:

- **Get Text:** Extracts a text value from a specified UI element.
- **Get Full Text:** Automatically generated when performing Screen scraping with the FullText method, along with a container. It extracts a string and its information from an indicated UI element. The hidden text is also captured by default.
- **Get Visible Text:** Automatically generated when performing screen scraping with the Native method chosen, along with a container. It extracts a string and its information from an indicated UI element.
- **Get OCR Text:** Automatically generated when performing screen scraping, along with a container. By default, the Google OCR engine is used (but it can be changed to the other options available). It extracts a string and its information from an indicated UI element.

Techniques for extraction:

- Screen Scraping
- Data Scraping
- PDF Extraction


These are discussed in detail in the coming slides.

Apart from text, the users can also extract the attributes from a UI element. For this, **Extract Attributes** is the category of activities that can be used when the user doesn't want to extract

the text out of the UI element, but maybe the color, the position or an ancestor. There are 3 different UiPath activities to do this:

- **Get Ancestor:** UI elements are in parents-children structures (a text document has the Notepad app as a parent, which has the category of apps as a parent, and so on). Get Ancestor retrieves the ancestor (or the parent) of a UI element.
- **Get Attributes:** UI elements have plenty of attributes. For example, a button on a website – it has a color, a name, a state, and so on. Get Attributes allows the user to indicate an attribute, and then it retrieves the value of that specific attribute.
- **Get Position:** It retrieves the actual position of a specific element on the screen. This can be very useful when there are many similar elements on a screen and without knowing their actual position, it would become very difficult to identify each of them.

Screen Scraping

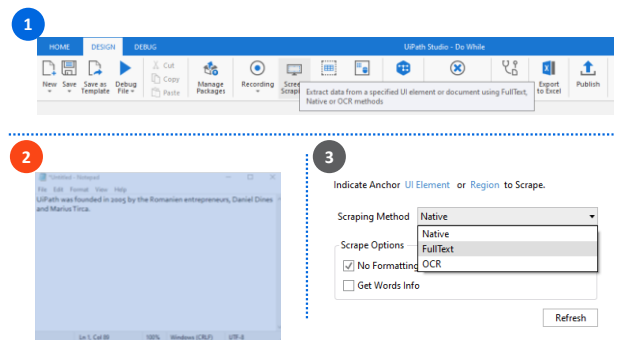


Process of extracting data from a specified UI element or document, such as a pdf file. The Screen Scraping Wizard enables the user to point at a UI element and extract text from it.

STEP 1 Start the Screen Scraping Wizard from the Design ribbon in UiPath Studio.

STEP 2 Select the UI element on screen.

STEP 3 Select the Screen Scraping Method from the Options panel.




Screen scraping is the process of extracting data from a specified UI element or document, such as a pdf file. Screen scraping methods are the core of all the activities that enable extracting data from a specified UI element or document.

In some cases, data is sent directly to the users or from one activity to another by using variables or arguments. But in other cases, other methods are needed to take data out from various screens or documents. This is where Screen scraping methods are used. They enable data extraction from different UI elements that the automation workflows interact with.

The Screen Scraping Wizard enables the user to point at a UI element and extract text from it. The steps are:


- Start the Screen Scraping Wizard from the Design ribbon in UiPath Studio.
- The screen goes into Computer Vision mode, highlighting the UI elements that it identifies with blue. Once the user selects the UI element, Studio automatically chooses a screen scraping method.
- After it finishes extracting the text, the wizard shows the outcome. The user can select one of the three methods and customize it using the properties. Clicking the 'Refresh' button displays the outcome according to the new settings.

Screen Scraping Methods




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


FullText



Native



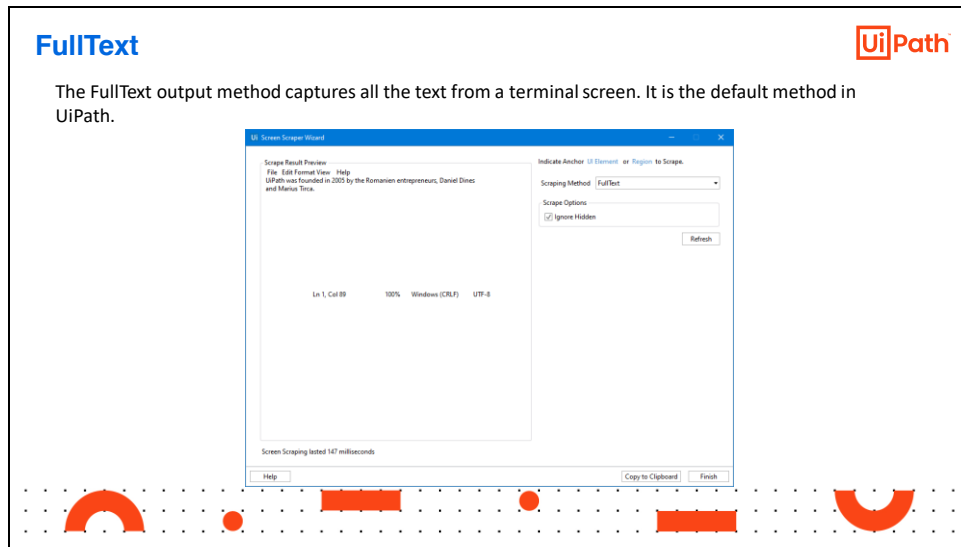
OCR



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

To extract text from a UI element, the user can use one of the three screen scraping (or output) methods that UiPath supports. These are:

- **The FullText method:** This is the default output method. It is the fastest method with 100% accuracy and can work in the background. It can extract hidden text for example, fetching options available in a drop-down list. However, it doesn't support Citrix and doesn't capture text position and formatting.
- **The Native method:** This is compatible with applications that use the Graphics Design Interface, and Microsoft API responsible for representing a graphical object. Its speed is lower than FullText, and it cannot work in the background. However, it can extract the text position and formatting with 100% accuracy. Just like FullText, it doesn't support Citrix.
- **The OCR (Optical Character Recognition) method:** It is the only method in UiPath which works with Citrix. This technology relies on recognizing each character just like we recognize faces in photography. It cannot work in the background, can't extract hidden text, and its speed is the lowest. Its accuracy varies from one text to another, and changing settings can improve the results. Just like the Native method, it also captures the text position.

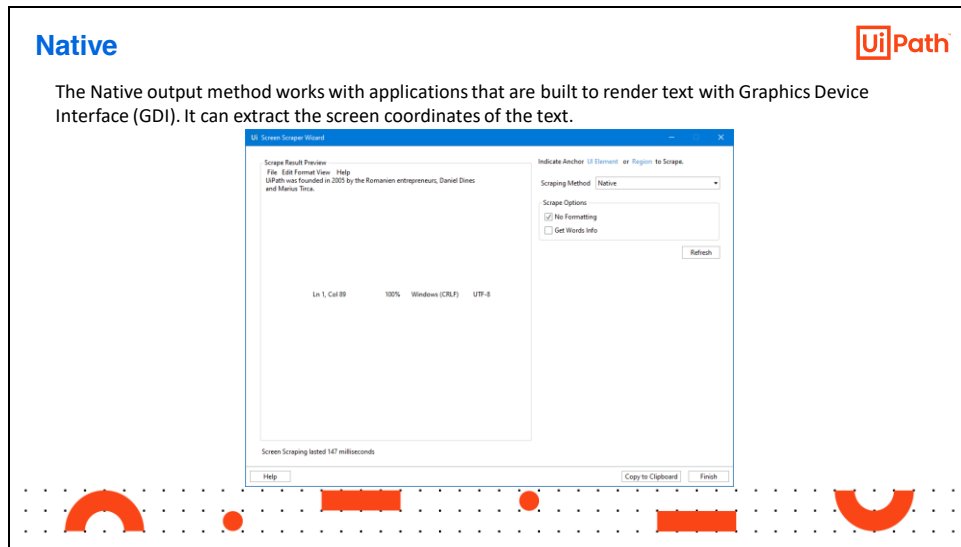


FullText Output Method:

- The FullText output method is the default output method in UiPath. It is fast, accurate and can work in the background. It captures all the text from the terminal screen, including the hidden text (for example, the options in a drop-down list).
- It doesn't work on Citrix and different virtual environments. It doesn't retain formatting and text position as well.

The method offers the following option:

- Ignore Hidden: When Ignore Hidden method is selected, it offers the option to ignore the hidden message and capture only the visible text.




Native Output Method:

- It can be used only with applications that are built to render text with Graphics Device Interface. It is useful when the coordinates of each word are needed, and when the font and color of the text are required to be retained accurately.
- However, it has its limitations as well. It only extracts the visible text, cannot work in the background, and it doesn't support Citrix.


The method offers the following options:

- **No Formatting:** When the font and color are not necessary to be captured, this box can be checked, and the wizard extracts only the text, just like the FullText method.
- **Get Words Info:** This option can extract the position of each word and can support several separators. By default, it can process all known characters as separators like comma or space but when specific separators are specified, it ignores the others. Additionally, the Custom Separators field is displayed, enabling the user to determine the characters used as separators. If the field is empty, all known text separators are used.

OCR


The OCR output method uses the Optical Character Recognition technology, to extract information from virtual environments. It has two default engines: Google Tesseract and Microsoft MODI.

	Multiple Languages Support	Preferred Area Size	Support for Color Inversion	Filter Allowed Characters	Best with Microsoft Fonts
Google Tesseract	Can be added	Small	YES	YES	NO
Microsoft MODI	Supported by default	Large	NO	NO	YES



OCR (or Optical Character Recognition) Output Method:

- Although both FullText and Native have excellent results in terms of accuracy and speed, there are specific cases in which both ~~of them~~ are unusable. For example, if we need to extract information from virtual environments or “read” text from images, the Optical Character Recognition(OCR) output method should be used.
- It is based on the OCR technology used in recognition of scanned documents. It attempts to recognize each letter or given an image in the target document.
- It is slow when compared to the other methods, has lower accuracy (it varies from one text to another), it cannot extract hidden text and it cannot work in the background.

The OCR method has two default engines. The use of these engines depends on the type of information being extracted.

- **Google Tesseract**
- **Microsoft MODI**

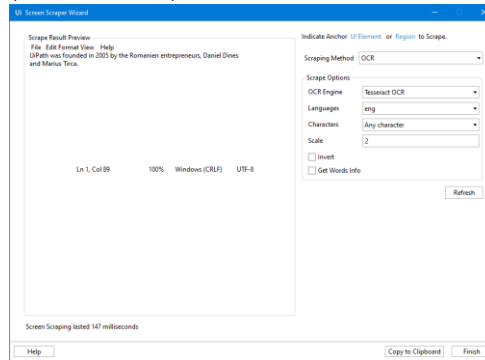
In general, it’s better to switch between the methods to see which of the engines brings better results for each particular situation.

If the user has downloaded and installed the packages of other OCR engines (such as Abbyy), they will also be available while choosing the OCR engine.

Google Tesseract OCR



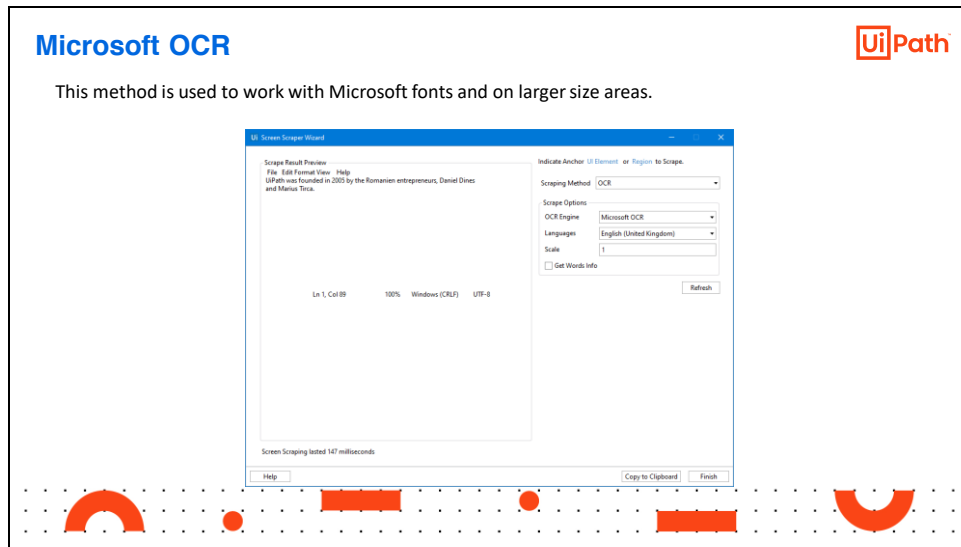
This method gets better results for character recognition on smaller size areas and supports color inversion. It offers multiple customization options.



This method gets better results for character recognition on smaller size areas and supports color inversion. It offers multiple customization options through filters that can be used to select only specific categories of characters.

The method offers the following options:


- Languages: This is English by default.
- Characters: Enables the user 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. When Custom is selected, two additional fields, Allowed and Denied, are displayed that enable the user to create custom rules on which types of characters to scrape and which to avoid.
- Invert: When this checkbox 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 enlarged is 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.




This method is used to work with Microsoft fonts and on larger size areas. It supports multiple languages.

The method offers the following options:


- **Languages:** Enables the user 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 enlarged is 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.

Classroom Exercise




Demonstrate how to scrape text from a UiPath blog post using **Screen Scraper Wizard** and store it in a Notepad File.


- Open a blog post on UiPath website
- Use Screen Scraper Wizard to scrape text
- Switch between Native, Full Text, and OCR Scraping Method to view result qualities in Scrape Result Preview window.
- Store scraped text in a Notepad file using only Full Text scraping method.
- Save and Close the file




Demonstrate how to scrape text from a UiPath blog post using **Screen Scraping** wizard and store it in a Notepad File.

- Open a blog post on UiPath Website.
- Go to UiPath Studio, and click Screen Scraping button from the Design ribbon. Indicate the text you want to copy from the UiPath blog page.
- In the Screen Scraper Wizard, switch between different scraping methods.
- Choose each one by one to see their results in the preview area. Click the Scraping Method drop down and select Native. Click the Refresh button and see the preview.
- Again click the Scraping Method drop down and select OCR and click Refresh button. The output of OCR is in the paragraph style as seen on the website.
- Click the Scraping Method drop down and select Full Text and click Refresh button and see the preview.
- Drag and drop Write Text File activity below Screen Scraping container. In the Text box, enter the variable in which the scraped content is stored. In the Write to filename: box, enter the filename "content.txt".
- Save and run the workflow.
- Open the saved Notepad to see if the scraped content is stored
- Outcome:
 - The content is stored in the file.


Practice Exercise





Build a workflow using **Screen Scraper Wizard** that scrapes text using **Full Text** scraping method and stores in a Notepad file.

- Search for “UiPath” in Google Search.
- Scrape information about UiPath shown on top right of the result page using **Full Text** scraping method.
- Store text in a Notepad file.





Build a workflow using **Screen Scraping** wizard that scrapes text using **Full Text** scraping method and stores in a MS Word file.

- Search for “UiPath” in Google Search.
- Scrape information about UiPath shown on top right of the result page using **Full Text** scraping method.
- Store text in a Notepad file.

Algorithm


- START
- Use Open Browser activity to open URL – “www.google.com”
- Use Type Into activity in Open Browser activity to indicate search bar of Google and search for “UiPath”
- Use Screen Scraping button in Design ribbon and select information about UiPath shown on the right of the Google search results page.
- Choose Full Text as Screen Scraping method.
- Use Write Text File activity to store scraped content in a Notepad.
- STOP

Practice Exercise




Build a workflow using **Screen Scraper Wizard** that scrapes text using **Tesseract OCR** scraping method from an image and stores in a Notepad.

- Search for "text images" in Google Images.
- Pick one image containing text from the search results.
- Scrape the text from the image using Tesseract OCR.
- Store text in a Notepad file.




Build a workflow using **Screen Scraping** wizard that scrapes text using **Tesseract OCR** scraping method from an image and stores in a Notepad file.

- Search for "text images" in Google Images
- Pick one image containing text from the search results
- Scrape the text from the image using Tesseract OCR
- Store text in a Notepad file




Algorithm

- START
- Use Open Browser activity and enter URL www.google.com/images
- Use Type Into activity in Open Browser activity and search "text images" in the URL
- Use Screen Scraping from Design ribbon to select an image containing text.
- Choose Tesseract OCR as Screen Scraping method. Change Scale to 5.
- Use Write Text File activity to store content in a Notepad file.
- STOP

Data Scraping



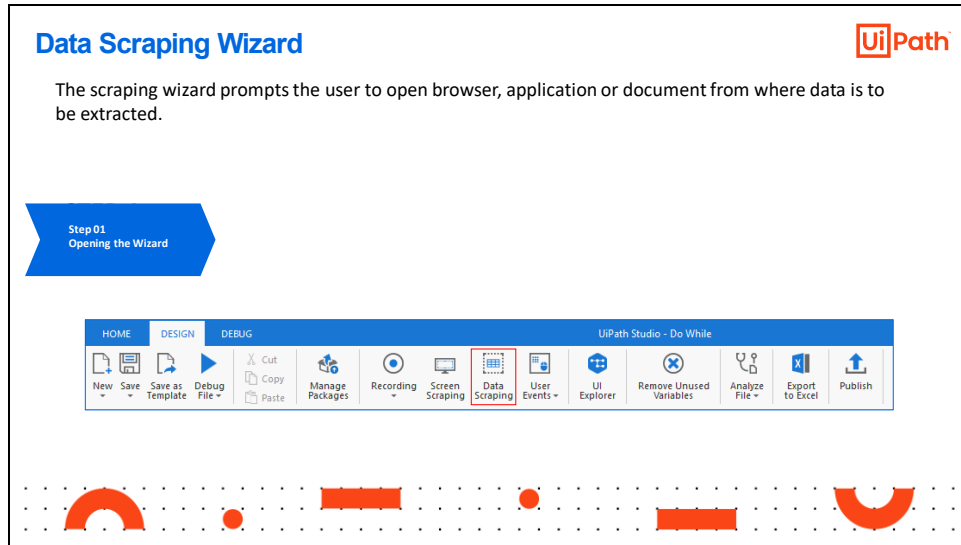
Process of extracting structured data from browser, application or document to a database, .csv file or Excel spreadsheet.

-  Extracts all the pattern-based data and stores it into the form of the data table automatically
-  Generates a container with a selector & Extract Structured Data activity with a partial selector to ensure correct identification of the app to be scraped
-  Stores the scraped information in a DataTable variable to populate a database or an Excel spreadsheet

Data scraping is the process of extracting structured data from browser, application or document to a database, .csv file or Excel spreadsheet. It is used to create a data table at run time. It can extract all the pattern-based data and store it into the form of the data table automatically.

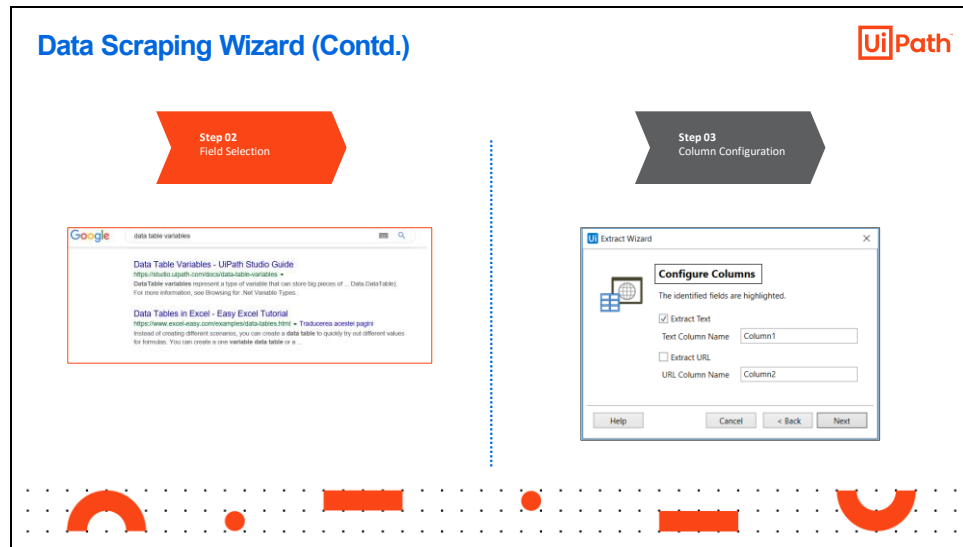
Data scraping always generates a container (Attach Browser or Attach Window) with a selector for the top-level window and an Extract Structured Data activity (which extracts data from an indicated web page) with a partial selector, thus ensuring a correct identification of the app to be scraped.

All the scraped information is stored in a DataTable variable, that can later be used to populate a database, a .csv file or an Excel spreadsheet.



For extracting data using the Data Scraping Wizard, the steps are:

1. Opening the Wizard: Start the Data Scraping Wizard from the Design ribbon in UiPath Studio.

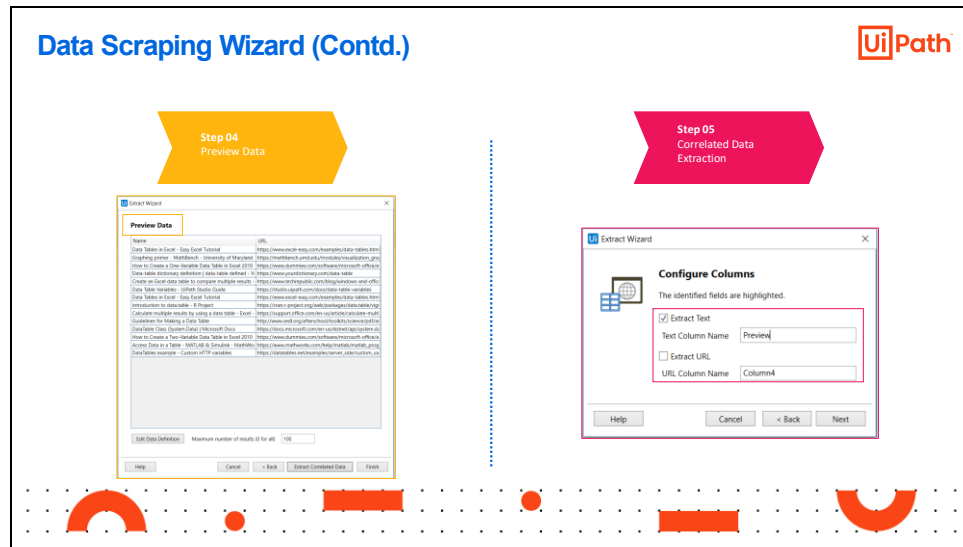


2. Field Selection: Select the First entry and Last entry of the field for extraction so that Studio can deduce the pattern of the information.

- The first entry refers to the first input UI element which the user wants to extract. Once the user selects the first UI element, this input will be taken as a "First entry" inside the Data scraping wizard.
- In order to make the wizard identify the structure of the content, the user needs to input another similar UI element. Once the user selects this other UI element, the Data scraping wizard will complete the structure, and this element will be referred to as the "Last entry".

3. Column Configuration: User can customize column header names and choose whether or not to extract URLs.

- After the first extraction is done, the wizard asks the user to name the column of the extracted data in the Configure Columns dialog box where the user needs to enter the column's name. The column name is added inside the "Text Column Name" field.
- To display the column's URL, check the "Extract URL" box and enter the URL Name and press Next to continue.

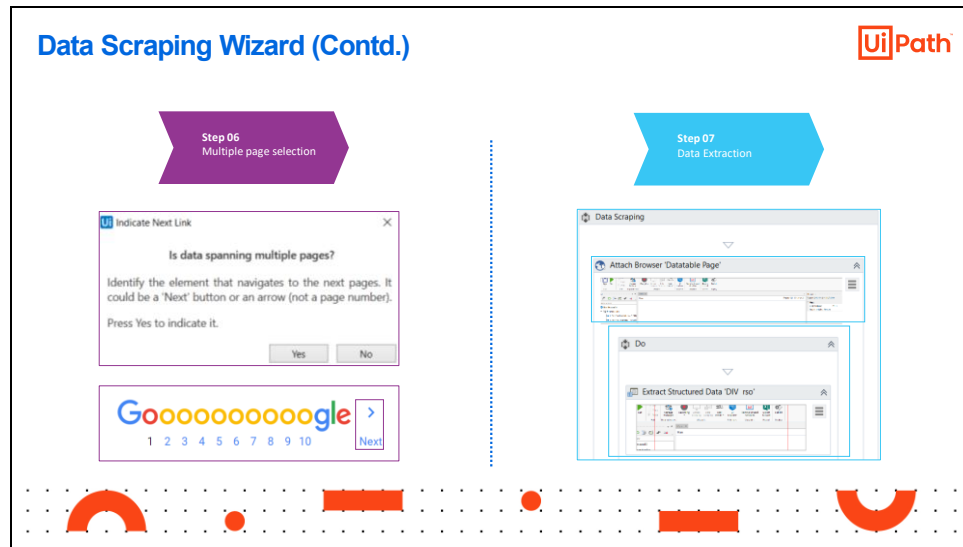


4. Preview Data: Once the 'Next' button is clicked, a preview of the data is shown

- The user may move the columns by dragging the mouse cursor as per requirement.
- The user can also select the amount of structured data to be extracted by entering it inside the "Maximum number of Results" field. If Zero (0) is entered, all the available data is extracted.

5. Correlated Data: The user also has the option 'Extract Correlated Data' in the pattern preview. This offers the opportunity to extract other fields of structured data, by the same process of indicating the first and the last entry.


- The process is particularly useful when the user wants to extract multiple fields. The correlated data can be extracted multiple times by adding the names of the column.




6. Multiple page selection: Indicates the Next button in the web page, application, or document.

- Once the user clicks 'Next', the wizard asks whether the data spans on multiple pages and if so, the user needs to select Yes. Now, the user can identify the element that navigates to the next pages.


7. Data Extraction: Once the user is done with multiple page selection, a sequence is generated in Studio.

Classroom Exercise




Demonstrate how to scrape first 100 results from Google search results using **Data Scraping** wizard.

- Search "Automation is future" in Google Search
- Extract all site titles and URLs by navigating through first ten pages in the search result
- Store scraped data in a CSV file





Demonstrate how to scrape first 100 results from Google search results using **Data Scraping** wizard.

- Insert **Open Browser** activity in the designer panel. Enter the URL "www.google.com".
- Run the workflow to see if it opens the URL
 - Google Search is opened successfully.
- Drag and drop a **Type Into** activity in the **Do** container. Click on the "Indicate element inside browser" link and indicate the search bar of Google.
- In the text area of Type Into activity, enter text "Automation is Future" followed by Enter as Special key.
- Run the workflow to see if the text is searched in the Google.
 - Text is successfully searched in Google.
- Go back to UiPath Studio. Click on the Data Scraping button in the Design ribbon.
- In Extract Wizard, click Next and select site title of the first search result. Now, again click Next and select site title of the second search result. This will help Wizard to identify pattern of results.
- In the Text Column Name box, change the default "Column 1" text to "Site Titles", and click Next. A preview of results is shown in the Preview section. Leave maximum number of results as 100. It will extract first 100 search results.
- Click Finish. A dialog box appears that asks if there are multiple pages. Click "Yes", and select the "Next" button at the bottom of the search result page.
- Insert a Write CSV file after the Data Scraping container. "Enter SearchResult.csv" in the first text box, and the name of the data table in the second text box.


- Save and run the workflow.
- Go to the folder where the CSV file is saved and open the file to see if the results are stored
- Outcome:
 - All results are stored successfully in the CSV file.

Practice Exercise




Build a workflow using **Data Scraping** wizard that scrapes blog post titles from UiPath Blog from multiple pages.

- Open UiPath Blog (<https://www.uipath.com/blog>).
- Extract all blog titles and URL by navigating through all the pages.
- Store scraped data in an Excel file.

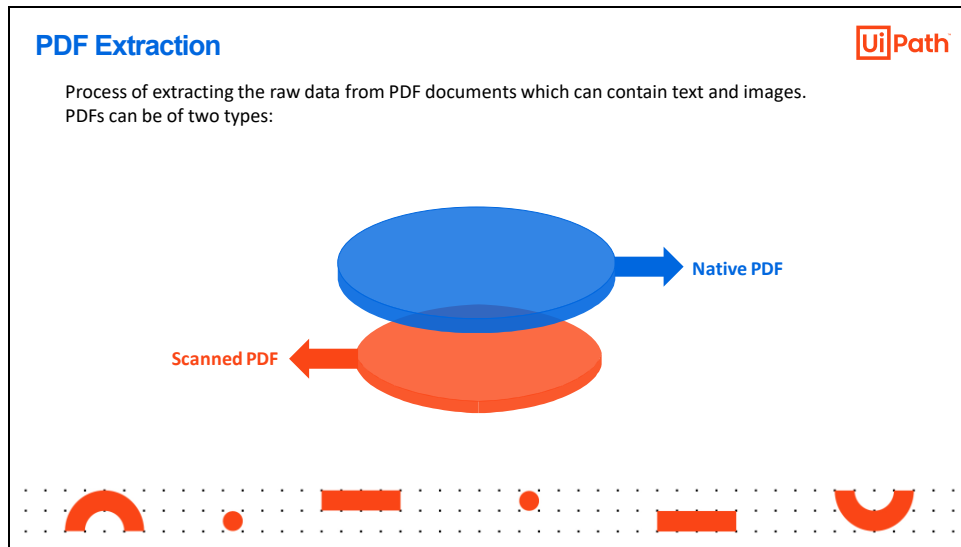


Build a workflow using **Data Scraping** wizard that scrapes blog post titles from UiPath Blog from multiple pages.

- Open UiPath Blog (<https://www.uipath.com/blog>).
- Extract all blog titles and URL by navigating through all the pages.
- Store scraped data in an Excel file.

Algorithm

- START
- Use Open Browser activity and enter URL – “www.uipath.com/blog”
- Use Data Scraping button in Design ribbon to indicate first two blog post titles
- Rename “Column1” to “Blog Titles”
- Use Indicate Next Link window of Data Scrapping tool to indicate Next link in the search results page
- Use Write CSV activity and save the data in a Notepad file
- STOP



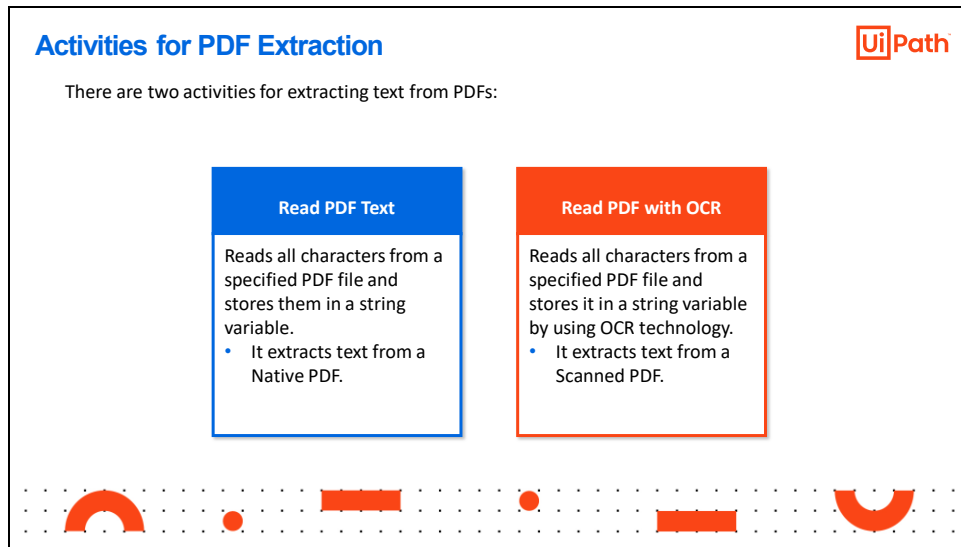
PDF (Portable Document Format) is a widely popular file format as it is one of the most reliable formats to store data.

PDF extraction is the process of extracting the raw data from PDF documents. PDF files can contain text, images, and sometimes text that are actually undercover images.

PDFs can be of two types:

- Native PDF: PDF File originally generated from a computer by Word, Excel, InDesign, Illustrator, or any other software that generates reports, spreadsheets, layouts. They are built of code that allows them to be viewed and read exactly as they were originally created.
- Scanned PDF: PDF made up of scanned images of a given document. With scanned PDFs, the user is generally unable to search across the text because the PDF is actually a collection of images.

UiPath offers several activities to extract data from PDFs. Before starting data extraction, the user must install the **UiPath.PDF.Activities** package on the system with the help of the Manage Package Section in UiPath Studio.




There are two activities for extracting text from PDFs:

- **Read PDF Text:** Reads all characters from a specified PDF file and stores them in a string variable.
 - It chooses the file to be read and outputs a text variable with the contents of the file. The result can be saved as a text file and displayed in a message box. Other string operations can be used to modify or extract information out of the generated text.
 - The activity extracts or converts only the text part of the document and any images in the document are ignored.
- **Read PDF with OCR:** Reads all characters from a specified PDF file and stores it in a string variable by using OCR technology.
 - It uses Optical Character Recognition to scan the images inside the PDF document and output all the text as a variable. It requires an OCR engine for the scanning procedure. Studio comes with OCR engines from Google, Microsoft and Abbyy.

An important property of these activities is **Range**. It specifies the range of pages that the user wants to read. The user can specify a single page (e.g. "7"), a range of pages (e.g. "7-12"), or a complex range, (e.g. "2-5, 7, 15-End" or "All"). The default value is "All". Only string variables and strings are supported.

Both these Read PDF activities are self-contained, i.e., they don't require other applications to be open, so they can run in the background.

Other PDF Activities



Some other activities related to PDFs in UiPath are:

Get PDF Page Count

- Provides the total number of pages in a PDF file.

Extract PDF Page Range

- Extracts text from a specified range of pages from a PDF document.

Export PDF Page As Image

- Creates an image from a page in a specified PDF file.

Join PDF Files

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

Extract Images From PDF


- Extracts images from a specified PDF file and saves them in a folder.


Manage PDF Password

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

Some other activities related to PDFs in UiPath are:


- Get PDF Page Count
 - Provides the total number of pages in a PDF file.
- Export PDF Page As Image
 - Creates an image from a page in a specified PDF file.
- Extract Images From PDF
 - Extracts images from a specified PDF file and saves them in a folder.
- Extract PDF Page Range
 - Extracts text from a specified range of pages from a PDF document.
- Join PDF Files
 - Joins multiple PDF files stored in an array of strings into a single PDF file.
- Manage PDF Password
 - Manages the password of a specified PDF file if current password details are known.

Classroom Exercise





Demonstrate how to extract text from a pdf file using **Read PDF Text** activity and store in a Notepad file.


- Install the dependency UiPath.PDF.Activities
- Go to <https://www.uipath.com/solutions/whitepapers> and download a PDF whitepaper
- Open the downloaded PDF file
- Scrape the text from the file using Read PDF Text activity
- Save the scraped text directly in a .txt file



Demonstrate how to extract text from a pdf file using **Read PDF Text** activity and store in a Notepad file.


- First download a whitepaper from UiPath Website.
Go to "<https://www.uipath.com/solutions/whitepapers>" and click on the first link. Fill the form with your details to download the PDF file. Enter name, email, company name, phone number, country, job level, and field of expertise. Now, hit Submit.
- Check your email and download the PDF file received.
- Go to UiPath Studio and click on Manage Packages button in the Design ribbon. In the pop up window, navigate to All Packages
- Search for UiPath.PDF.Activities and click the first result. In the right panel, click Install and Save. In the terms and conditions pop up window, select "I Accept". Wait for few seconds till the dependencies are installed.
- Drag and drop Read PDF Text activity in the designer panel. Click the Folder icon and select the downloaded PDF file. Go to the Properties panel of the Read PDF Text activity, and in the Output property, press Ctrl+K and insert a new variable called **content**.
- Insert Write Text File activity and insert below Read PDF Text activity. Enter the variable content in the first input box, and "scrapedText" in the second input box.
- Save and run the workflow.
- Go to the folder where the Notepad is saved and open it
- Outcome
 - All the texts from the PDF file is stored here.

Practice Exercise




Build a workflow using **Read PDF Text activity and extract only Email IDs and Phone Number from a PDF file and store in a MS Word file.**

- Download the practice excel file available on rpachallenge.com.
- Convert the file to PDF to use in this exercise.
- Read data from the PDF file using Read PDF Text activity.
- Extract only Name and email IDs from the PDF and store in a MS Word file.

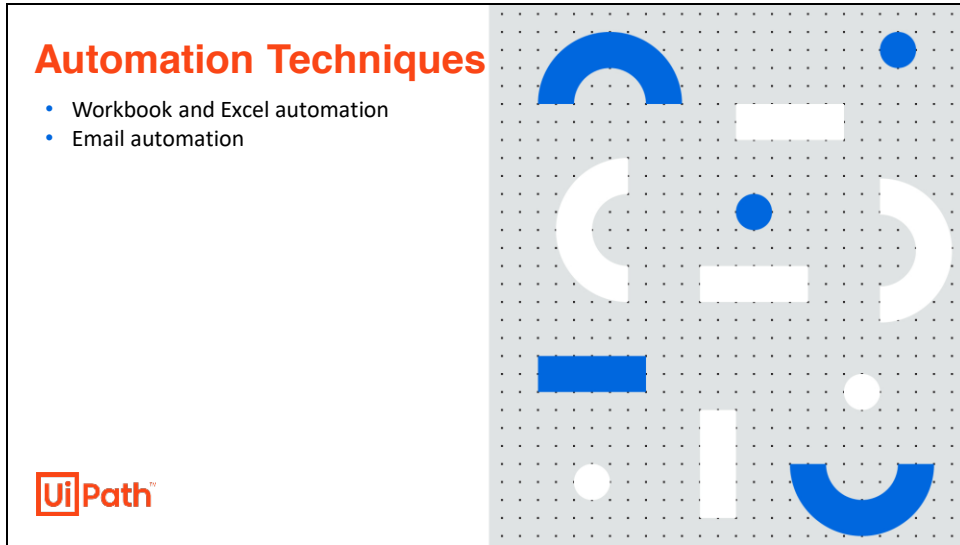


Build a workflow using Read PDF Text activity and extract only Email IDs and Phone Number from a PDF file and store in a MS Word file.

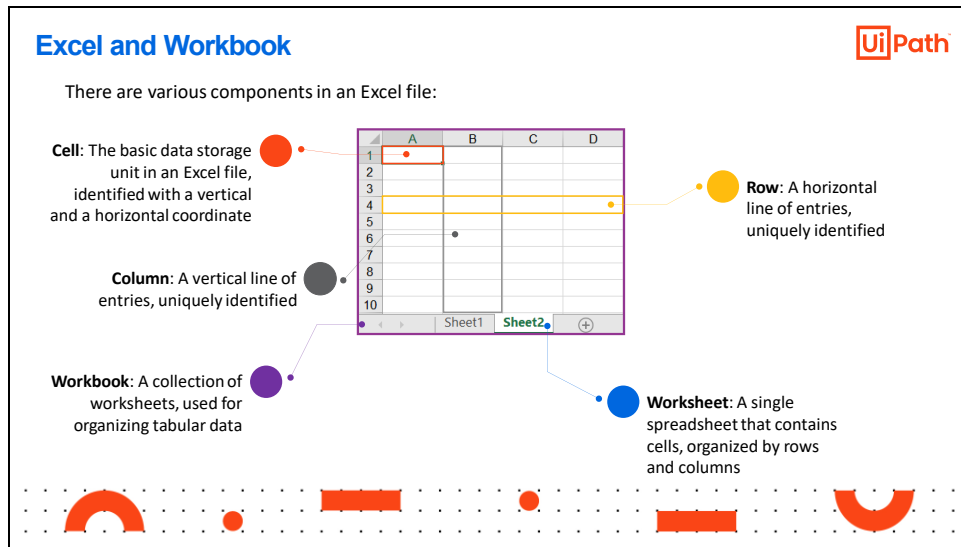
- Download the practice excel file available on rpachallenge.com.
- Convert the file to PDF to use in this exercise.
- Read data from the PDF file using Read PDF Text activity.
- Extract only Name and email IDs from the PDF and store in a MS Word file.

Algorithm

- START
- Use Read PDF Text activity to read content of the PDF file and store in a string variable.
- Use Matches activity below Read PDF Text activity
 - In RegEx column, select Email.
- Use For Each activity to iterate through each email item and store in a MS Word file using Type Into activity.
- Use another Matches activity below previous Matches activity.
 - In Value column, enter the expression: `(407)([0-9])`
- Use For Each activity to iterate through each phone number item and store in MS Word file using Type Into activity.
- STOP



This section gives an overview of automation techniques for workbook and Excel automation and Email automation.



There are various components in an Excel file:

- **Cell:** The basic data storage unit in an Excel file, identified with a vertical and a horizontal coordinate
- **Row:** A horizontal line of entries, uniquely identified
- **Column:** A vertical line of entries, uniquely identified
- **Worksheet:** A single spreadsheet that contains cells, organized by rows and columns
- **Workbook:** A collection of worksheets, used for organizing tabular data

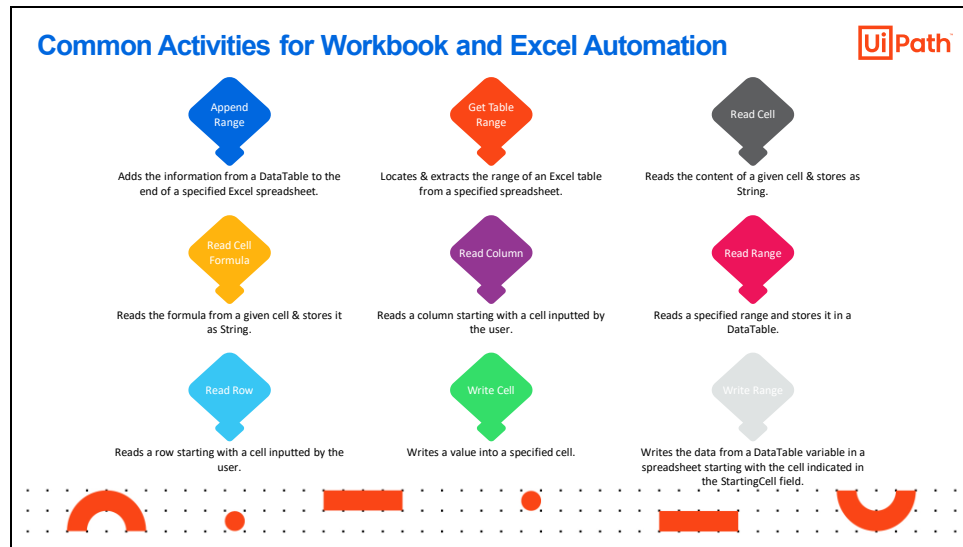


In many business scenarios, databases are stored in workbooks. From there, they can be inputted into DataTables and processed further.

UiPath offers two ways of accessing and manipulating workbooks:

- **File Access Level:** In this access method, all workbook activities are executed in the background.
 - It doesn't require Microsoft Excel to be installed.
 - This method is faster and more reliable for some operations when the user doesn't open the file.
 - It works only for .xlsx files.
- **Excel App Integration:** In this access method, UiPath opens Excel just like a human.
 - It requires Microsoft Excel to be installed. If the file isn't open, it will be opened, saved and closed for each activity.
 - All activities can be set to either be visible to the user or run in the background.
 - It works with .xls and .xlsm, and it has some specific activities for working with .csv.

Both these access levels share some common activities and Excel App Integration offers few more. These activities can be accessed from 'System > File > Workbook' and 'App Integration > Excel' for the respective method. The activities are discussed in the coming slides.



Common activities for Workbook and Excel automation:

- **Append Range:** Adds the information from a DataTable to the end of a specified Excel spreadsheet. If the sheet does not exist, it creates it.
- **Get Table Range:** Locates and extracts the range of an Excel table from a specified spreadsheet using the table name as input.

In addition to these, there are common activities for reading from and writing to workbook.


How to Read from a Workbook:


- **Read Cell:** Reads the content of a given cell and stores as String.
- **Read Cell Formula:** Reads the formula from a given cell and stores it as String.
- **Read Column:** Reads a column starting with a cell inputted by the user and stores it as an `IEnumerable<Object>` variable.
- **Read Range:** Reads a specified range and stores it in a DataTable. If 'Use filter' is checked in the Read Range activity under 'Excel Application Scope', it will read only the filtered data. This option does not exist for the Read Range activity under 'Workbook'.
- **Read Row:** Reads a row starting with a cell inputted by the user and stores it as an `IEnumerable<Object>` variable.

How to Write to a Workbook:

- **Write Cell:** Writes a value into a specified cell. If the cell contains data, the activity will overwrite it. If the sheet specified doesn't exist, it will be created.


- **Write Range:** Writes the data from a DataTable variable in a spreadsheet starting with the cell indicated in the StartingCell field. If the starting cell isn't specified, the data is written starting from the A1 cell. If the sheet does not exist, a new one is created with the SheetName value. All cells within the specified range are overwritten.

Classroom Exercise





Demonstrate how to print data from a workbook in the output panel using **Read Range** activity.


- Create an excel file containing age of ten students
- Read the data using Read Range activity
- Loop through each data and subtract it with current year to get year of birth.
- Display the result in the Output panel



Demonstrate how to print data from a workbook in the output panel using **Read Range** activity.


- First create an initial excel file for this activity. Open a new Excel file and create a called called Student Age. Enter age of ten students. Save and close the file.
- Go to UiPath Studio, and search Read Range activity under Workbook category in the Activities panel and insert in the designer panel. Click the ellipsis icon and select the excel file just created.
- Go the Properties panel of Read Range activity and in the DataTable property, hit Ctrl+K and enter a new variable called **studentAge**.
- Insert For Each Row activity below the Read Range activity. In the *VB expression* text box enter studentAge.
- Insert Assign activity in the Body section of For Each Row activity. In the first text box enter **row(1)** and in the adjacent text box enter the expression: **Date.Today.Year() - CInt(row(o))**
- Insert Output Data Table activity after For Each activity. Go to its Properties panel and enter studentAge in the DataTable property. In the Output property, hit Ctrl+K and enter a new variable called **ageOutput**.
- Insert Write Line activity below Output Data Table activity. Enter **ageOutput** in the text box.
- Save and run the workflow.
- Go to the Output panel to see the output
- Outcome:
 - Year of birth of all students are listed adjacent to their respective age.

Practice Exercise




Build a workflow using **Read Range** and **Append Range** activity to read data from a workbook and append data to another workbook.

- Create an excel file containing names of any five cities in small letters.
- Read the data from the file using Read Range activity.
- Convert all city names in capital letters.
- Add the updated names in a new spreadsheet using Append Range activity.

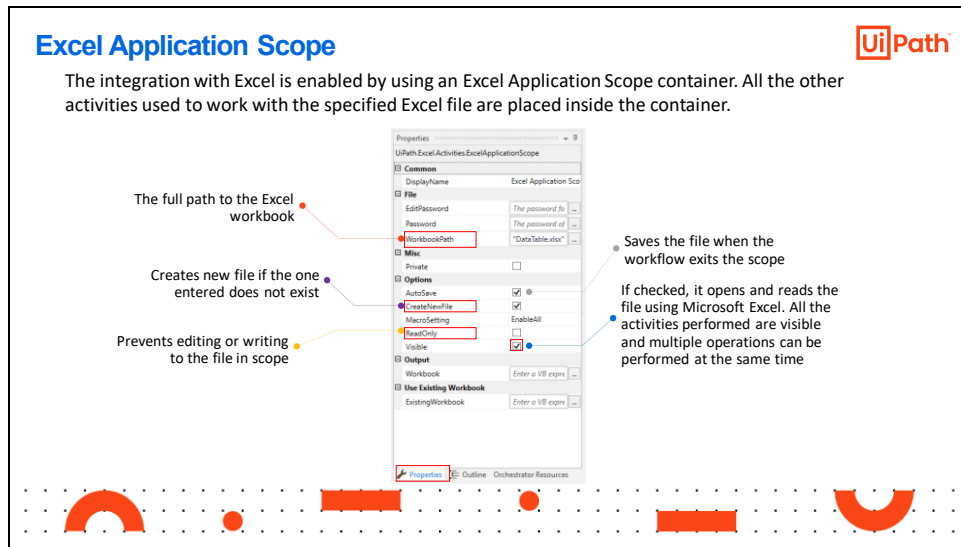


Build a workflow using **Read Range** and **Append Range** activity to read data from a workbook and append data to another workbook.

- Create an excel file containing names of any five cities in small letters.
- Read the data from the file using Read Range activity.
- Convert all city names in capital letters.
- Add the updated names in a new spreadsheet using Append Range activity.

Algorithm

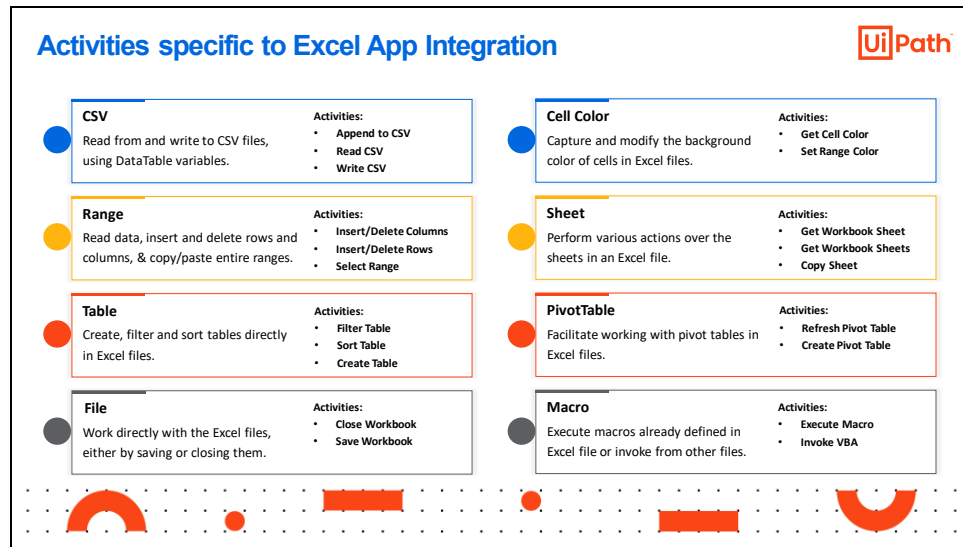
- START
- Use Read Range activity from Workbook category and to read data from the excel file path and store in a data table variable.
- Use Add Data Column activity add a new column in the data table called "City in Capitals".
- Use For Each Row activity to iterate through each row items in the data table.
- Use Assign activity to convert each row item to uppercase
- Use Append Range activity below For Each activity to store updated data tables with city names in capitals in a new excel file.
- STOP



The integration with Excel is enabled by using an Excel Application Scope container. All the other activities used to work with the specified Excel file are placed inside the container. The purpose of the container is to open the Excel workbook and provide a scope for Excel activities. When the execution ends, the specified workbook and the Excel application are closed. It can be configured to write the output of the activities in the container in a different file.

Some important properties of Excel Application Scope are:

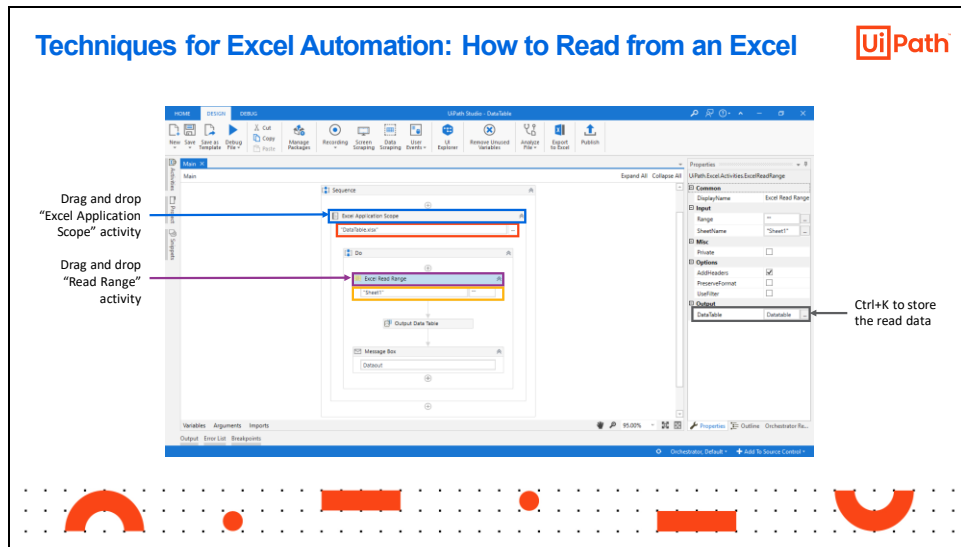
- **WorkbookPath:** The full path to the Excel workbook.
- **AutoSave:** Saves the file when the workflow exits the scope.
- **CreateNewFile:** Creates new file if the one entered does not exist.
- **ReadOnly:** Prevents editing or writing to the file in scope.
- **Visible:** If checked, it opens and reads the file using Microsoft Excel. All the activities performed are visible and multiple operations can be performed at the same time.



Activities specific to Excel Automation:

- **CSV:** These activities can read from and write to CSV files, using DataTable variables. Although found under Excel App Integration, they work even if they are not placed inside an Excel Application Scope container. The activities under this category are:
 - **Append to CSV:** Adds the information from a DataTable to a CSV file, creating it if it doesn't exist. The activity does not overwrite existing data
 - **Read CSV:** Reads all entries from a CSV file and store them in a DataTable
 - **Write CSV:** Overwrites a CSV with the information from a DataTable
- **Range:** These activities can read data, insert and delete rows and columns, and even copy/paste entire ranges. They are similar to the corresponding activities under DataTable, but they work directly in the Excel file. The activities under this category are:
 - **Delete Column:** Removes a column from an Excel file based on the name.
 - **Insert Column:** Inserts a blank column in an Excel file, at a certain position.
 - **Insert/Delete Columns:** Either adds blank columns or removes existing columns, based on the specified change type.
 - **Insert/Delete Rows:** Either adds blank rows or removes existing rows, based on the specified change type.
 - **Select Range:** Selects a specific range in an Excel file. In general, it is paired with another activity that performs a certain manipulation over the selected data.
 - **Get Selected Range:** Outputs a given range as String.
 - **Delete Range:** Removes a specified range from an Excel file.
 - **Auto Fill Range:** Applies a given formula over a given range in an Excel file.

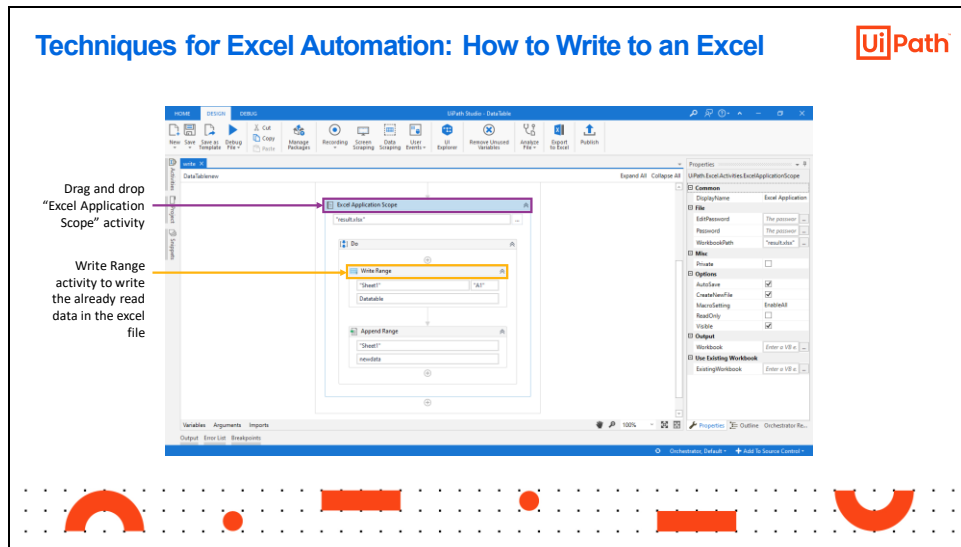
- Copy Paste Range: Copies and pastes an entire range (values, formulas and formatting) from a source sheet to a destination sheet.
- Lookup Range: Searches for a value in all the cells in a given range.
- Remove Duplicate Range: Deletes all duplicate rows from a given range.
- Table: These activities create, filter and sort tables directly in Excel files. The activities under this category are:
 - Filter Table: Applies a filter on all the values from a column in a table inside an Excel file. Once the file is saved, only the rows that meet the filter will be displayed. This activity does not remove the rows that do not meet the criteria, but only hides them.
 - Sort Table: Sorts a table in an Excel file based on the values in a given column.
 - Create Table: Creates a table (with name) in a range specified in the Properties panel.
- File: These activities work directly with the Excel files, either by saving or closing them. The activities under this category are:
 - Close Workbook: Closes an opened Excel workbook.
 - Save Workbook: Saves changes to the workbook specified in the WorkbookPath property of the Excel Application Scope. It can only be used in the Excel Application Scope activity.
- Cell Color: These activities are able to capture and modify the background color of cells in Excel files. The activities under this category are:
 - Get Cell Color: Reads the background color of a given cell in an Excel file and stores it as color variable output.
 - Set Range Color: Changes the background color of all the cells in a given range. The input is a color variable.
- Sheet: These activities can perform various actions over the sheets in an Excel file. The activities under this category are:
 - Get Workbook Sheet: Reads the name of a sheet by its index.
 - Get Workbook Sheets: Extracts the sheet names and stores them ordered by index.
 - Copy Sheet: Copies a sheet in an Excel file and pastes either in the same Excel file or in a different one specified.
- PivotTable: These activities facilitate working with pivot tables in Excel files. The activities under this category are:
 - Refresh Pivot Table: Refreshes a pivot table in an Excel file. This is useful when pivot table source data changes, as the refresh is not automatic.
 - Create Pivot Table: Creates a pivot table using a specified sheet and given parameters.
- Macro: These activities can execute macros that were already defined in the Excel file, or can invoke macros from other files. These activities work with .xslm files. The activities under this category are:
 - Execute Macro: Executes the macro within a workbook. The Workbook file needs to be a Macro-Enabled Workbook
 - Invoke VBA: Invokes a macro from an external file containing VBA code and runs it against an Excel file



How to Read from an Excel:


- Open UiPath Studio. Search for "Excel Application Scope" activity and drag-drop it in the Designer panel.
- Search for Read Range activity and drag-drop this activity within the Excel Application Scope activity.
- Specify the location to read the excel file. Click on the button with three dots on the top right side of the Read Range activity to choose the file and its location on the computer.
- Provide the name of the sheet within the Excel file.
- Go to properties panel and click on output box and type "Ctrl + K". It prompts the user to give a name to the variable. It creates a data table to store the read data.


Techniques for Excel Automation: How to Write to an Excel



How to Write to an Excel:


- Open UiPath Studio. Search for "Excel Application Scope" activity and drag-drop it in the Designer panel.
- Search for Write Range activity and drag-drop this activity within the Excel Application Scope activity.
- Specify the path of the new excel file to write data.
- Provide the name of the memory space with the previously read data. On execution, data is written to a new Excel.

Classroom Exercise




Demonstrate how to read and write in Excel by comparing first two columns and inserting result in third column using Excel activities.

- Create an excel file containing ten random numbers between 1 to 100 in two columns
- Read file and transport into data table, and add a third column
 - If value in first column is greater than second column, enter "Greater" in third column as result.
 - If value in first column is less than second column, enter "Lesser" in third column as result
 - If value in first column is equal to second column, enter "Equal" in third column as result
- Write back the updated data table in the same excel file in a new sheet





Demonstrate how to read and write in Excel by comparing first two columns and inserting result in third column using Excel activities.

- First create an initial excel file for this activity. Open a new Excel file and create two columns A and B. Enter ten random numbers between 1 and 100 in each column. Save and close the file.
- Go to UiPath Studio, and drag and drop Excel Application Scope activity in the designer panel. Click the ellipsis icon and select the excel file that we just created.
- Insert Read Range activity within the Do container of Excel Application Scope activity.
- Go to the Properties panel of Read Range activity, and in the Output property, hit Ctrl + K and enter a new variable called **dummyData**.
- Add Data Column activity below Read Range activity. Go to the Properties panel. In the ColumnName property, enter "Comparison", and in the DataTable property enter **dummyData**.
- Insert For Each Row activity below Add Data Column activity. In the *Vb expression* text box enter **dummyData**.
- Insert If activity in the **Body** section of **For Each Row** activity. Enter condition **CInt(row(0))>CInt(row(1))**. In the **Then** section, insert an Assign activity. In the first text box enter row(2) and in the adjacent text box enter the string "Greater".
- Insert another If activity in the **Else** section of first If activity. Enter condition **CInt(row(0))<CInt(row(1))**. In the **Then** section, insert an **Assign** activity. In the first text box enter **row(2)** and in the adjacent box enter the string "Lesser". In the **Else** section enter

another **Assign** activity. In the first text box enter **row(2)** and in the adjacent box enter the string "Equal".

- Insert Append Range activity below For Each Row activity. In the first text box enter "Sheet2", and in the second text box enter dummyData. It will insert the result in a new sheet in the same excel sheet.
- Save and run the workflow.
- Open the excel file to check the outcome
- Outcome:
- The comparison result is stored in third column in a new sheet.

Practice Exercise




Build a workflow that calculates total monthly deposit of a bank from an Excel file and store output in a new sheet.

- Download the Excel file link given for practice.
- File contains three deposit categories – Cash In, On-Us Check, and Not On-Us Check.
- Calculate total amount received in all three categories for the month of June.
- Store calculated values in a new sheet in the same excel file.

Note: Download initial Excel data for this practice from - https://www.uipath.com/hubfs/Documentation/WorkflowExamples/QueueItem_Example_Reports.xlsx

Build a workflow that calculates total monthly deposit of a bank from an Excel file and store output in a new sheet.

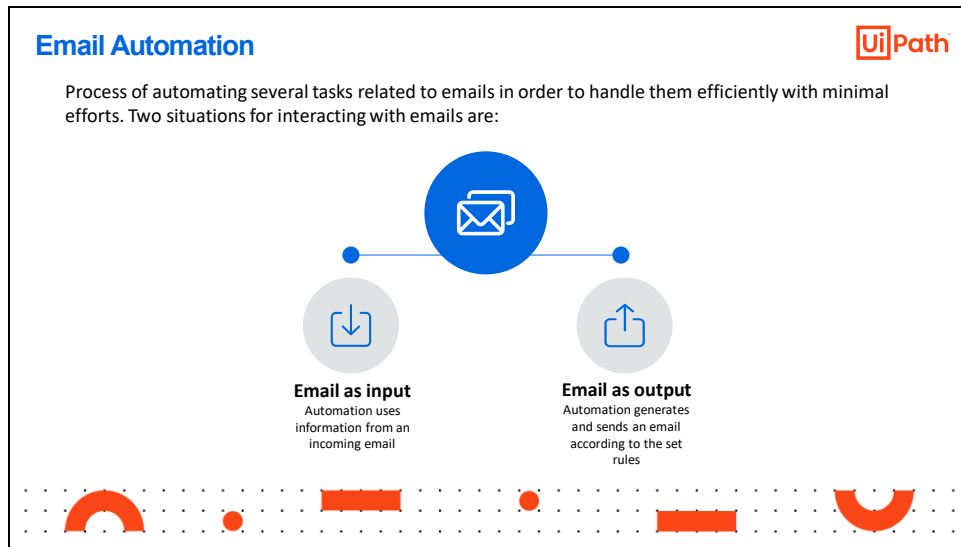
- Download the Excel file link given for practice.
- File contains three deposit categories – Cash In,
- On-Us Check, and Not On-Us Check.
- Calculate total amount received in all three categories for the month of June.
- Store calculated values in a new sheet in the same excel file.

Note: Download initial Excel data for this practice from - https://www.uipath.com/hubfs/Documentation/WorkflowExamples/QueueItem_Example_Reports.xlsx

Algorithm

- START
- Use Build Data Table activity and in its Properties panel, enter a new variable **sumTable**.
- Create a data table with three integer columns with names "Cash In Total", "On-Us Check Total", and "Not On-Us Total".
- Use Excel Application Scope activity and select the downloaded excel file
- Use Read Range activity in the Do container to read excel data from "June Report" sheet.
- Use an Assign activity within a For Each Row activity below the Read Range activity
 - In the first text box, enter new integer variable **sum1**.

- In the second text box, enter expression **CInt(row (1)) + sum1**.
- Use second Assign activity in the Body section
 - In the first text box, enter new integer variable **sum2**.
 - In the second text box, enter expression **CInt(row (2)) + sum2**.
- Use third Assign activity in the Body section.
 - In the first text box, enter new integer variable **sum3**.
 - In the second text box, enter expression **CInt(row (3)) + sum3**.
- Use Add Data Row activity below For Each Row activity. Go to its Properties panel, and in the ArrayRow property enter the expression: {sum1, sum2, sum3}. In the Data Table property enter **sumTable**.
- Use Write Range activity below Add Data Row activity,
 - Insert "June Total" in the first text box,
 - **sumTable** in the second text box.
 - Check the box of Add Headers property in the Properties panel
- STOP



Email is the oldest form of digital communication between people and to this day, many business processes are still triggered by a simple email. The users keep close check on their inbox throughout the day and thus, getting an email report about the status of an automated task might be essential sometimes to keeping it on track.

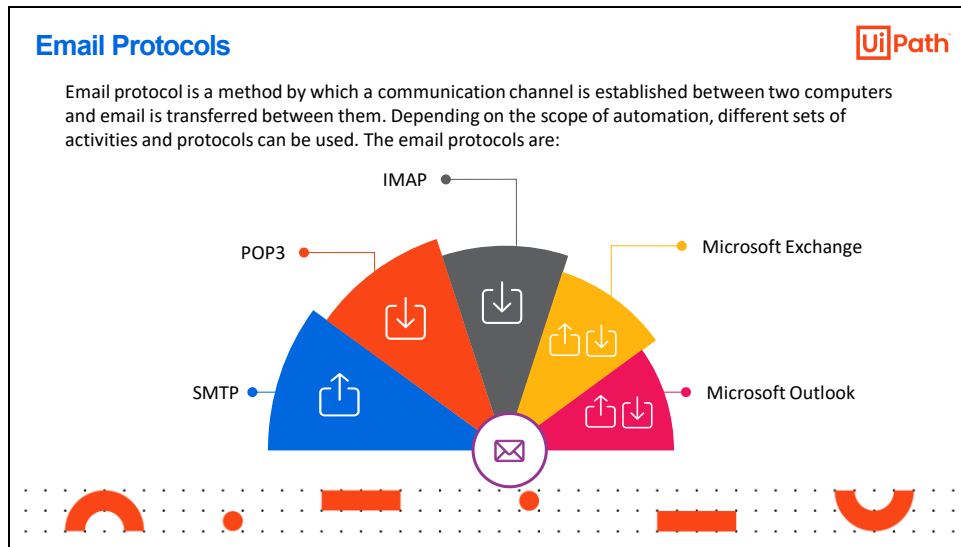
There are many tasks related to emails that can be automated. Email automation is a time-saving activity that helps to manage bulk emails with minimal human efforts. It is one of the best techniques to ease out the difficulty of efficiently handling massive data over the emails.

From an RPA perspective, two situations for interacting with emails are:

- **Email as the input for business process:** Here, the automation uses information from an incoming email. Data is received in textual form, as part of the subject or body of the email, or as an attachment (for example an Excel file, a PDF or other file types). Example:
 - Names or ID numbers from the subject or body
 - Input files coming as attachments (.xls. .pdf)
- **Email as the output of business process:** Here, the automation generates and sends an email according to the set rules. Email is used for sending status updates to users regarding different automation projects or when business or application exceptions occur. Example:
 - Progress reports
 - Exception alerts

Email automation involves a number of configuration steps and different type of applications and the process requires the usage of a number of if-else control statements and standard, structured or unstructured input and output types.

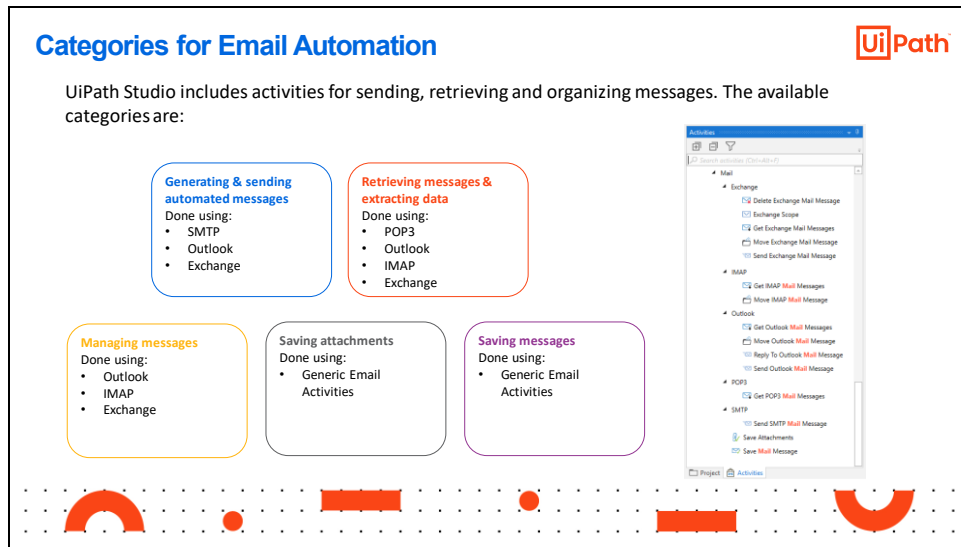
The **UiPath.Mail.Activities** package includes all the activities related to emails in UiPath.



Email protocol is a method by which a communication channel is established between two computers and email is transferred between them. Depending on the scope of automation, different sets of activities and protocols can be used. There are five email protocols:

- SMTP (Simple Mail Transfer Protocol) - Output
- POP₃ (Post Office Protocol) - Input
- IMAP (Internet Message Access Protocol) - Input
- Microsoft Exchange - Input/Output
- Microsoft Outlook - Input/Output

These are discussed in detail in the coming slides.



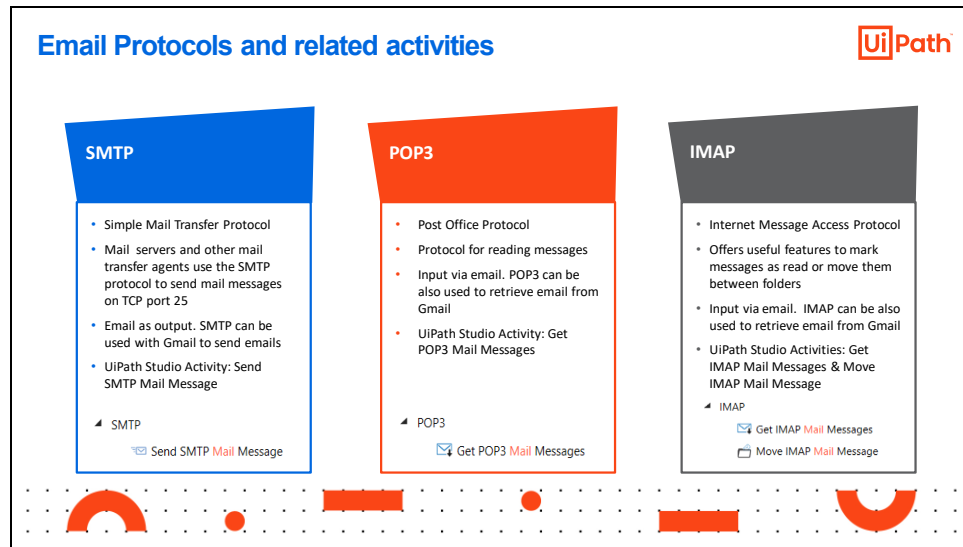
With the help of email automation in UiPath, the users can generate and send automated messages, retrieve messages and extract data. Email automation also helps to manage messages, save email attachments and essential information in well-defined and structured form.

UiPath Studio includes activities for sending, retrieving and organizing messages. The available categories are:

- **Generating & sending automated messages.** This is done using following protocols:
 - SMTP
 - Outlook
 - Exchange
- **Retrieving messages & extracting data.** This is done using following protocols:
 - POP3
 - Outlook
 - IMAP
 - Exchange
- **Managing messages.** This is done using following protocols:
 - Outlook
 - IMAP
 - Exchange
- **Saving attachments.** This is done using following:
 - Generic Email Activities
- **Saving messages.** This is done using following:

- Generic Email Activities

The activities are discussed in detail in the coming slides.



SMTP:

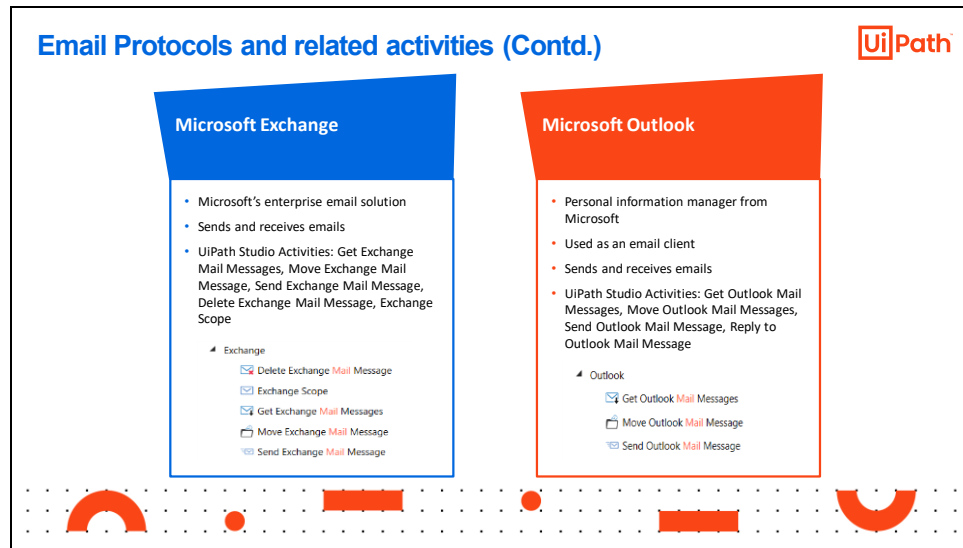
- Simple Mail Transfer Protocol
- Mail servers and other mail transfer agents use the SMTP protocol to send mail messages on TCP port 25
- Email as output. SMTP can be used with Gmail to send emails
- UiPath Studio Activity:
 - Send SMTP Mail Message: Sends an email message by using the SMTP protocol.

POP3:

- Post Office Protocol
- Protocol for reading messages
- Input via email. POP3 can be also used to retrieve email from Gmail
- UiPath Studio Activity:
 - Get POP3 Mail Messages: Retrieves a POP3 email message from a specified server.

IMAP:

- Internet Message Access Protocol
- Offers useful features to mark messages as read or move them between folders
- Input via email. IMAP can be also used to retrieve email from Gmail
- UiPath Studio Activities:
 - Get IMAP Mail Messages: Retrieves an IMAP email message from a specified server.
 - Move IMAP Mail Message: Moves an IMAP email message to a specified folder.



Microsoft Exchange


- Microsoft's enterprise email solution
- Sends and receives emails
- Uses a proprietary protocol called Messaging Application Programming Interface or MAPI to talk to email clients.
- UiPath Studio Activities:
 - Get Exchange Mail Messages: Retrieves an email message from Exchange.
 - Move Exchange Mail Message: Moves an email message from Exchange to another folder.
 - Send Exchange Mail Message: Sends an email message from Exchange.
 - Delete Exchange Mail Message: Deletes an Exchange email message.
 - Exchange Scope: Connects to Exchange and provides a scope for other Exchange activities.

Microsoft Outlook

- Personal information manager from Microsoft (information-sharing platform)
- Used as an email client
- Sends and receives emails
- Also comes with calendar, task manager, scheduling meeting request and many other information management facilities
- UiPath Studio Activities:
 - Get Outlook Mail Messages: Retrieves email messages from Outlook.
 - Move Outlook Mail Messages: Moves an Outlook email message to a specified folder.
 - Send Outlook Mail Message: Sends an email message from Outlook.

- Reply to Outlook Mail Message: Replies to an email message using Outlook.

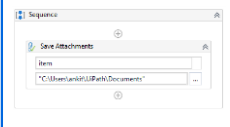
Generic Email activities



There are two generic email activities available for saving email attachments and email messages on the local drive.

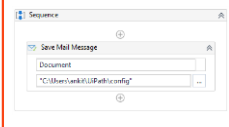
Save Attachments

- Saves the mail message attachments to the specified folder.



Save Mail Message

- Saves the email message to the specified folder.





There are two generic email activities available for saving email attachments and email messages on the local drive.

- **Save attachments:**
- Saves the mail message attachments to the specified folder.
- If the folder doesn't exist, it is created. If no folder is specified, the downloads are saved in the project folder.
- Files in the specified folder with the same name as the attachments are overwritten.
- Important properties are:
 - **FolderPath:** The full path of the folder where the attachments are to be saved.
 - **Message:** The MailMessage object whose attachments are to be saved.
 - **Attachments:** The retrieved attachments.

2. Save Mail Message:


- Saves the email message to the specified folder.
- If the folder doesn't exist, it is created. If no folder is specified, the downloads are saved in the project folder.
- Files in the specified folder with the same name as the messages are overwritten.
- Important properties are:
 - **MailMessage:** The MailMessage object to be saved.
 - **FilePath:** The full path where the MailMessage object is to be saved.

Classroom Exercise




Demonstrate how to extract From, Subject, and Body of emails using Get IMAP Mail Messages activity and store in a CSV file


- Set up IMAP configuration to access the email
- Loop through each email and extract From, Subject, and Body
- Store extracted data in an excel file in three different columns with header names as From, Subject, and Body




Demonstrate how to extract From, Subject, and Body of emails using Get IMAP Mail Messages activity and store in a CSV file.

- Insert a Build Data Table in the designer panel. Click on the Data Table button within the activity. In the pop up window, create three columns with headers as **Subject** with Data Type as String, **From** with Data Type as String, and **Body** with Data Type as String. Go to the Properties panel of this activity and in DataTable property, hit **Ctrl+K** and enter a new variable called **getMails**.
- Insert **Get IMAP Mail Messages** activity in the designer panel. Go to its Properties panel. In the MailFolder property, enter "Inbox". In the Port property, enter 993. In Server property enter "imap.gmail.com". In the Email property, enter your email address in double quotes and in the Password property, enter your password in double quotes. Ensure that the OnlyUnreadMessages property is unchecked. It will fetch all emails regardless of their *Read* status. In the Top property, enter **5**. In the Messages property, hit Ctrl+K and enter a variable **newMessages**.
- If using Gmail username and password, then allow Google Account for less secure apps and devices. Do this by visiting Security tab under your Google Account page.
- Insert a **For Each** activity below the Get IMAP Mail Messages activity. Replace the text "item" from the first text box with "mail", and in the adjacent text box enter the variable **newMessages**.
- Go to the Properties panel of the For Each activity and click the drop down of TypeArgument property and select "Browse for Types". In the pop up window, browse for MailMessage type. You can notice that there are two results shown. First is under System.Net.Mail and the second is under System.Web.Mail. Pick the first one.


- Insert Assign activity in the Body section of the For Each activity. In the first text box, hit **Ctrl+K** and enter a new variable called subject. In the adjacent box enter the expression: **mail.Subject**.
- Insert second Assign activity below the previous Assign activity. In the first text box, hit **Ctrl+K** and enter a new variable called from. In the adjacent box enter the expression: **mail.from.ToString**.
- Insert third Assign activity below the second Assign activity. In the first text box, hit **Ctrl+K** and enter a new variable called body. In the adjacent box enter the expression: **mail.Body**
- Insert Add Data Row activity below the third Assign activity. Go its properties panel, and in ArrayRow property, enter the expression: **{subject,from,body}**. Also, in the DataTable property, enter the variable **getMails**.
- Insert a Write CSV activity after For Each activity. In the first text box enter a new CSV file name called **freshMails.csv** and in the second text box enter the data table variable **getMails**.
- Save and run the workflow.
- Open the CSV file freshMails.csv
- Outcome:
- All the emails are stored in separate columns under Subject, From, and Body.

Classroom Exercise




Demonstrate how to extract and store From, Subject, and Body of emails in an excel file using **Get Outlook Mail Messages** activity.


- Use Get Outlook Mail Messages activity to identify Outlook mail folder
- Loop through each email and extract From, Subject, and Body
- Store extracted data in an excel file in three different columns with header names as From, Subject, and Body




Demonstrate how to extract and store From, Subject, and Body of emails in an excel file using **Get Outlook Mail Messages** activity.

- Before starting this activity, ensure that you have configured Microsoft Outlook on your computer and it is in working state.
- Now, insert a Build Data Table in the designer panel. Click on the Data Table button within the activity. In the pop up window, create three columns with headers as **Subject** with Data Type as String, **From** with Data Type as String, and **Body** with Data Type as String. Go to the Properties panel of this activity and in DataTable property, hit **Ctrl+K** and enter a new variable called **getMails**.
- Insert **Get Outlook Mail Messages** activity in the designer panel. Go to its Properties panel. In the Account property, specify the email account in double quotes from where you want to fetch the mails. In the MailFolder property, enter the folder name in double quotes from where you want to fetch mails. Ensure that the OnlyUnreadMessages property is unchecked. It will fetch all emails regardless of their *Read* status. In the Top property, enter **5**. In the Messages property, hit Ctrl+K and enter a variable **newMessages**.
- Insert a **For Each** activity below the Get Outlook Mail Messages activity. Replace the text "item" from the first text box with "mail", and in the adjacent text box enter the variable **newMessages**.
- Go to the Properties panel of the For Each activity and click the drop down of TypeArgument property and select "Browse for Types". In the pop up window, browse for MailMessage type. You can notice that there are two results shown. First is under System.Net.Mail and the second is under System.Web.Mail. Pick the first one.

- Insert an Assign activity in the Body section of the For Each activity. In the first text box, hit **Ctrl+K** and enter a new variable called subject. In the adjacent box enter the expression: **mail.Subject**.
- Insert second Assign activity below the previous Assign activity. In the first text box, hit **Ctrl+K** and enter a new variable called from. In the adjacent box enter the expression: **mail.from.ToString**.
- Insert third Assign activity below the second Assign activity. In the first text box, hit **Ctrl+K** and enter a new variable called body. In the adjacent box enter the expression: **mail.Body**
- Insert **Add Data Row** activity below the third Assign activity. Go its properties panel, and in ArrayRow property, enter the expression: **{subject,from,body}**. Also, in the DataTable property, enter the variable **getMails**.
- Insert an Excel Application Scope activity below the For Each activity. In the Workbook path text box, enter the text "freshMails.xlsx". Insert a Write Range activity within the Do container. Make sure that you pick the Write Range activity that is under Excel category from the Activities panel. In the Data table text box of the Write Range activity, enter **getMails**.
- Save and run the workflow.
- Open the excel file freshMails.xlsx...
- Outcome:
 - All the emails are stored in separate columns under Subject, From, and Body.

Practice Exercise




Build a workflow that extracts attachments from the emails containing the word “Resume” in its subject.

- Set up IMAP configuration to access the email.
- Loop through each email to identify subjects containing the word “Resume”.
- Download the attachments from the identified emails in a folder.

Note: Send 4-5 dummy emails to the email ID that you will use for practice. Emails must contain any file attachments and the word “Resume” in their subject.

Build a workflow that extracts attachments from the emails containing the word “Resume” in its subject.

- Set up IMAP configuration to access the email.
- Loop through each email to identify subjects containing the word “Resume”.
- Download the attachments from the identified emails in a folder.

Note: Send 4-5 dummy emails to the email ID that you will use for practice. Emails must contain any file attachments and the word “Resume” in their subject.

Algorithm

- START
- Use Get IMAP Mail Messages activity to retrieve email data.
- Use For Each activity to iterate through retrieved each email.
- Use If activity within the For Each activity and check for word “Resume” in their Subject
- Use Save Attachment activity within the If activity to store attachments that contains word “Resumes” in the Subject.
- STOP



To summarize, this lesson explained:

- Extraction and Its Techniques
- Automation Techniques