

MODULE-1

Robotic Process Automation: Overview:

| MODULE-1 |
|---|
| Robotic Process Automation: Overview: Return on Investment (ROI), Automation Types, UiPath StudioX. Common Concepts: Notebook, Activity Inputs, Activity Outputs, Common Properties, Common Activities, Write Line: Message Box, Input Dialog, Modify Text., Text to Left/Right, Delay, if, Switch., Repeat Number Of Times. , Skip Current, Exit Loop, Get Username/Password, Get Orchestrator Asset, Save For Later, Wait for Download, Group. Chapter 1 , Chapter 3 |

Overview:

1. What is Digital Transformation?

- It means using technology to improve the way companies work.
- Companies want to:
 - Reduce costs (especially human labor costs).
 - Increase speed and accuracy.
 - Make employees happier by reducing boring work.

2. What is RPA (Robotic Process Automation)?

- **RPA** uses **software robots** (not physical robots) to do **repetitive tasks** that humans usually do on a computer.
- These tasks include:
 - Copying data from one file to another.
 - Sending emails automatically.
 - Filling online forms.

3. What is UiPath StudioX?

- A **tool used to create RPA robots**.
- **No programming needed** – it's called **low-code/no-code**.
- Anyone (even non-programmers) can build automation.
- Example: A student can build a robot to organize files by date and subject automatically.

4. Is It Safe to Use RPA?

- Yes, robots:
 - Only do what they are told in the automation design.
 - Can be given limited access to systems.
 - Can ask the user for input if needed.
 - Can also use **AI** to make smarter decisions.

5. Why is RPA Useful?

- ✓ **Saves Time** – no need to do the same work again and again.
- ✓ **Reduces Errors** – robots don't make silly mistakes.
- ✓ **Makes Work Interesting** – employees can focus on creative work.

- ✓ **Increases Productivity** – more work done in less time.
- ✓ **Helps Employees Learn Automation** – even students can try!

6. RPA and Humans – Working Together

- RPA is not here to **replace people**.
- It helps people by **removing boring tasks** so they can focus on **important thinking tasks**.
- Example: A teacher can use RPA to automatically generate attendance reports and spend more time helping students.

7. Real-Life Example

- Imagine you check emails every day and copy some data into Excel.
- With RPA, a robot can:
 - Open your email.
 - Read the message.
 - Copy the data into Excel.
 - Send a reply.
- All automatically!

Return on Investment (ROI)

What is ROI?

Return on Investment (ROI) means the benefit or profit a company gains in return for the money it spends. When a business invests in a tool like RPA, it expects to save time, money, or increase efficiency as a result.

How RPA Gives Good ROI

1. **Reduces Repetitive Work:**
 - RPA automates tasks that are done again and again manually (like data entry).
 - This saves employees' time and increases productivity.
2. **Cost-Efficient:**
 - RPA software is **cheaper** than traditional business software.
 - So companies spend **less money** to automate their tasks.
3. **Uses Existing Systems:**
 - RPA tools work with current (existing) software like Excel, email, web apps, etc.
 - There's **no need to change or disturb live systems**.
 - This **reduces project costs** and technical risks.

Low Infrastructure Requirements

4. **Easy to Set Up:**
 - Robots need very **little infrastructure** to run.
 - They can be run on:
 - A **user's desktop** (called *attended automation*), or
 - A **virtual machine** in the background (called *unattended automation*).

Flexibility in Automation

5. **Choose Who Does What:**

- In a process, users can decide:
 - which steps are done by a **human**, and
 - which are done by a **robot**.
 - This gives flexibility in designing the workflow.
6. **Adaptable to Changes:**
- If the software UI (user interface) changes due to an update, RPA workflows can be **easily modified** to handle these changes.
 - This keeps automation running smoothly without big changes to the system.

Main Advantage

- RPA allows businesses to **create flexible workflows** that can **adapt to changes** in software and systems, with **very little impact** on the infrastructure.
- This makes RPA **efficient, adaptable, and cost-saving** — all leading to a **high ROI**.

Automation Types

There are **different types of robots** used in RPA to handle different kinds of tasks. The main types are:

1. **Attended Robots**
2. **Unattended Robots**
3. **Hybrid Robots**

These robots help in **automating business processes** and allow even **non-technical people (citizen developers)** to build automations using simple tools like **UiPath StudioX**.

Robotic Process Automation (RPA) is categorized into **three main types** based on how the automation is triggered and whether human interaction is needed. These types are:

1. Attended Automation

- **Definition:** Attended bots work **alongside humans**, typically on the **user's desktop**.
- **Usage:** Used in **real-time customer-facing tasks** where a human triggers the bot.
- **Trigger:** **Manual** (button click, field input, etc.)
- **Example:** A customer support agent uses a bot to fetch customer details instantly during a call.

2. Unattended Automation

- **Definition:** These bots work **without any human intervention**, usually in the **background**.
- **Usage:** Used for **back-office operations** such as batch jobs or scheduled tasks.
- **Trigger:** **Automatic** (schedule, system event, or workflow).
- **Example:** A bot that processes salary slips for employees every month automatically at midnight.

3. Hybrid Automation

- **Definition:** A combination of both attended and unattended automation.
- **Usage:** Used in **complex workflows** where part of the task needs human involvement, and the rest is automated.
- **Example:** A bank clerk collects customer documents (attended) → bot validates the documents in background (unattended) → another bot sends email notification (unattended).

Table Format for Quick Comparison (Optional)

| Type | Human Involvement | Trigger Type | Example Use Case |
|------------|-------------------|--------------|--------------------------------------|
| Attended | Yes | Manual | Fetching customer data during a call |
| Unattended | No | Automatic | Processing payroll in background |
| Hybrid | Partial | Both | Loan application workflow |

Conclusion:

Each type of RPA is suited for different business needs:

- **Attended** – Human+Bot teamwork
- **Unattended** – Full automation
- **Hybrid** – Balanced mix for complex scenarios

UiPath StudioX

What is UiPath StudioX?

- **UiPath StudioX** is a **no-code Robotic Process Automation (RPA)** tool developed by **UiPath**, a company founded in **2005 in Romania** by **Daniel Dines**.
- It is specially designed for **business users (non-programmers)** to create automation without any technical background.
- It uses a **drag-and-drop interface** and provides **templates and built-in support** for common apps like **Microsoft Office**.

Features of StudioX:

1. **No Coding Required:**
 - StudioX provides a **user-friendly interface**.
 - Users can build automation **without writing code**.
2. **Pre-Built Templates and Integrations:**
 - Includes **ready-made automation templates**.
 - **Supports apps like Excel, Word, Outlook, etc.**
3. **Runs on Desktop:**
 - Robots created using StudioX can run **locally** on the user's machine.
 - **No need for IT deployment** or complex setup.
4. **Governance and Compliance:**
 - Built-in features to support **audit tracking** and **company rules**.
5. **Support from UiPath Assistant & Orchestrator:**
 - Used for **scheduling, sharing, and managing** bots easily.

Citizen Developer Model:

- StudioX supports a “**Citizen Developer**” model, meaning:
 - **Business users are trained** to build automation.
 - They get **access to tools** and can build bots **without help from IT teams**.
- This model increases **employee satisfaction**, boosts **productivity**, and reduces **attrition**.

Why Is It Useful?

- Helps in **automating simple, repetitive tasks** like sending emails, filling forms, or updating spreadsheets.
- Encourages **learning of new technical skills** by business employees.
- **Improves efficiency** and supports the **democratization of automation** (i.e., making automation accessible to everyone in the company)

UiPath StudioX is a no-code RPA tool developed by UiPath, enabling business users to automate routine tasks without programming knowledge. It uses a drag-and-drop interface and provides pre-built templates and integrations with popular apps like Excel and Outlook. StudioX supports local machine deployment, eliminating the need for IT support. It also includes governance features to ensure compliance. By supporting the citizen developer model, StudioX empowers non-technical users to learn and apply automation, boosting organizational productivity and technical skills across teams.

Common Concepts

Notebook

1. What is Notebook?

- The **Notebook** is an **Excel file** used in **UiPath StudioX** to handle data and logic **without writing code**.
- It acts as a **central place** for storing data, calculations, and formulas in automation.

2. Why is it Important?

- It enables **no-code automation** by allowing users to use **Excel formulas**.
- Most business users are familiar with Excel, so using Notebook is easy and effective.

3. Default Notebook

- Every new StudioX project automatically comes with a **default Notebook** (named `Project_Notebook.xlsx`).
- This default file includes **pre-built worksheets** like:
 - **Text** – for text functions (e.g., split, trim)
 - **Date** – for date/time functions
 - **Number** – for numeric calculations
 - **File** – for handling file paths

4. Using the Default Notebook

- You can:
 - **Write data** to specific cells using **Write Cell** activity.
 - Use Excel **formulas** to process that data.
 - **Read results** using **Read Cell** or use it in other activities.

5. Example Use Case

- Write "John Doe" to B2 in Text sheet.
- Add formula in C2: `=LEFT(B2,SEARCH(" ",B2)-1)` → Extracts **First Name**.
- Use **Read Cell** activity to read C2 → Result = "John".

6. Scratchpad and Custom Formulas

- Use **Scratchpad worksheet** to test or write custom formulas.
- You can also **add new sheets** and define any number of formulas.

7. Referencing Notebook Data

- Activities (like Message Box or Email) can refer to cells in the Notebook to use dynamic data in automation.

8. Custom Notebook

- If you want to use your own Excel file instead of the default one:
 - Go to **Notebook > Configure Notebook** in the top ribbon.
 - Select your custom Excel file.
 - StudioX will now use that file as your main Notebook.

9. Real-World Benefits

- Makes automation:
 - **Easy** (for non-coders)
 - **Reusable** (just change the data)
 - **Clear** (logic is visible in Excel formulas)
- Great for handling dynamic business data like names, dates, amounts, etc.

Final Takeaway:

"Notebook is the bridge between StudioX and Excel. It lets you automate smartly using formulas, without writing code."

Activity Inputs

In UiPath StudioX, *activity inputs* refer to the **data provided to an activity to perform its operation**. Depending on the activity, this input can be text, values from Excel files, file properties, email content, or data from previous steps. StudioX provides multiple ways to supply input to activities, making automation flexible and efficient. Below are the main types of activity inputs:

1. Excel:

- Allows you to use data from Excel as input.
- You can specify a **cell address, range, table**, or even a **sheet**.
- You can also use values from a **specific row** of a table or workbook.

Example: Selecting `Sheet1 [A2]` as an input to enter that value into a form.

2. Mail:

- You can extract values from an **email** and use them as inputs.
- Inputs could be subject, body, sender, received date, etc.
- Useful when automating workflows involving Outlook.

Example: Use the **email subject** as the title of a document.

3. File:

- Lets you use **file attributes** as input to activities.
- Attributes include **file name, file path, last modified date**, etc.

Example: Move a file based on its **last modified date**.

4. Use Saved Value:

- This allows you to use the **output of a previous activity** as input to a new activity.
- Helps in creating **dependent steps** in automation.

Example: Save the name from one Excel cell, then use it in an email.

5. Text:

- Allows static or dynamic **text strings** as input.
- You can use the **Text Builder** to combine multiple values (from Excel, files, previous steps) with plain text.

Example: "Hello " + [Customer Name from Excel] + ", your order is confirmed."

Summary:

Activity Inputs in StudioX are essential for customizing automation tasks based on dynamic or static data sources. By combining different input types, we can build powerful and intelligent workflows.

Activity outputs

Introduction

In UiPath StudioX, **activities** are used to perform various tasks like reading a file, clicking a button, extracting data, etc. Once an activity completes its task, it may **return data** as output. This returned data is called the **Activity Output**.

For example:

- A **Read Text File** activity returns the **content of a text file**.
- A **Get Text** activity returns **text from a form field or screen**.

This output data is essential in building **data-driven and dynamic automation workflows**.

Purpose of Activity Outputs

- To **store, reuse, or pass** the result of one activity to another.
- Helps in creating **intelligent automation** by working with real-time data.
- Ensures a **smooth data flow** across multiple steps in the process.

Accessing Activity Output Options

- After adding an activity that returns data, a **plus (+) icon** appears next to it.
- Clicking this icon shows the **data output options menu**.
- This menu lets you choose how and where to store the returned data.

Types of Output Options

1. Excel

- Allows you to store the output directly into:
 - A specific **cell** (e.g., A1),
 - A **range** (e.g., A1:C3),
 - A **table** or **sheet**.
- It requires the activity to be inside a "Use Excel File" container.
- Useful for logging results or exporting data to Excel.

2. Save for Later Use

- Lets you create a **variable** to store the output.
- You can **name or rename** the variable for better understanding.
- These variables are stored in the **Data Manager panel**.
- Ideal for **reusing data** in later steps of the automation.

3. Copy to Clipboard

- Copies the output to the **system clipboard**.
- Can be **pasted later** using the "Paste from Clipboard" option.
- Useful for **temporary or one-time data use** between steps.

Example Scenario

Suppose you are using the **Get Text** activity to extract a user's name from a form:

- You can **store it in a variable** (Save for Later Use) for use in a personalized email.
- Or, you can **save it to Excel** if you're creating a report.
- If needed immediately in the next field, you can **copy it to clipboard** and paste it.

Benefits of Using Activity Outputs

- Makes automation **dynamic and intelligent**.
- Promotes **modular design** using variables and reusable data.
- Reduces manual effort and increases **automation accuracy**.

Activity Outputs are a core concept in UiPath StudioX that allow developers to **capture and utilize** data generated during automation. By using options like **Excel**, **Save for Later Use**, and **Clipboard**, users can design flexible, efficient, and professional automation workflows. Mastering activity outputs is essential for building real-world RPA solutions.

Common Properties

Introduction

In UiPath StudioX, **Common Properties** are settings that are available for most activities and can be used to **control the behavior** of an activity during execution. These properties are found in the **Properties Panel** under the "**Common**" section.

These settings help in managing **errors, delays, timeouts**, and customizing how activities appear and behave in the workflow.

Purpose of Common Properties

- To **control the flow** of automation before and after an activity.
- To handle **errors** gracefully without stopping the automation.
- To improve the **readability and maintainability** of workflows.
- To allow **fine-tuning** of activity execution time and behavior.

List of Common Properties and Their Description

| Property | Description |
|--------------------------|--|
| Continue on Error | Allows automation to continue execution even if the activity throws an error. Useful for non-critical steps. <i>(Value: True/False)</i> |
| Delay Before | Sets a pause before the activity starts. Helps in waiting for UI elements to load. <i>(Default: 0.2 seconds)</i> |
| Delay After | Sets a pause after the activity completes. Helps prevent overlap with the next activity. <i>(Default: 0.3 seconds)</i> |
| DisplayName | Allows you to change the label of the activity as it appears in the Designer panel. Improves clarity of the workflow. |
| Timeout | Specifies how long (in seconds) to wait for the activity to complete before it throws a timeout error. <i>(Default: 30 seconds)</i> |

Explanation of Use Cases

1. ☒ **Continue on Error**
 - Useful when the activity is optional or non-critical.
 - Example: If reading a file fails, continue with default data.
2. ☐ **Delay Before/After**
 - Helps to wait for **UI elements to load** or **avoid rapid execution**.
 - Useful in web automation, form filling, etc.
3. ☒ **DisplayName**
 - Helps rename the activity for better understanding.
 - Example: Changing "Click" to "Click Login Button".
4. ☐ **Timeout**
 - Helps handle activities that **may take longer to respond** (e.g., loading a web page).
 - Prevents automation from hanging indefinitely.

Benefits of Using Common Properties

- Ensures **smooth and controlled execution** of workflows.
- Reduces chances of **unexpected failures or crashes**.
- Improves the **readability and debugging** process.
- Makes automation more **reliable and flexible**.

Conclusion

Common properties in UiPath StudioX play a key role in **managing execution flow, handling delays, errors, and timeouts**. By customizing these properties, developers can create **more stable, readable, and efficient automation workflows**. Understanding and using these properties wisely is essential for every RPA developer.

1. Continue on Error

Example:

You are extracting a user's profile picture from a website. Sometimes the profile picture may not be available.

Use Case:

Set `ContinueOnError = True` so that even if the image is missing, the automation continues with the remaining data like name and email.

☛ *“Even if an error occurs, don’t stop the automation.”*

2. Delay Before

Example:

Before clicking a **Login** button on a web page, you want to ensure that the button is visible and the page is fully loaded.

Use Case:

Set `DelayBefore = 3000` (i.e., 3 seconds) to wait before clicking the button.

☛ *“Wait for a few seconds before executing the next step.”*

3. Delay After

Example:

After typing a username in a textbox, wait for the system to auto-fill suggestions or validate it.

Use Case:

Set `DelayAfter = 2000` (i.e., 2 seconds) to pause before moving to the next field.

“Pause briefly after completing this activity.”

4. DisplayName

Example:

Instead of using the default name "Click", rename the activity to "**Click Submit Button**".

Use Case:

Helps you and your team understand the workflow more clearly while debugging or reviewing.

“Make the activity name meaningful in the workflow.”

5. Timeout

Example:

Waiting for a slow-loading confirmation message after submitting a form.

Use Case:

Set `Timeout = 60000` (i.e., 60 seconds). If the message doesn't appear within this time, throw an error.

“Wait up to X seconds for this to finish, or stop if it takes too long.”

Summary Table

| Property | Real-Time Example | Purpose |
|--------------------------|--|---------------------------------|
| Continue on Error | Missing image in a form – continue with other fields | Avoid stopping for minor issues |
| Delay Before | Wait before clicking Login | Ensure UI is ready |
| Delay After | Wait after typing into a field | Allow page processing |
| DisplayName | Rename activity to "Click Submit Button" | Improve readability |
| Timeout | Wait up to 60 sec for a slow response | Prevent infinite waiting |

✓Introduction

UiPath StudioX provides a set of **commonly used activities** that are applicable across all types of automation (web, Excel, file, etc.). These activities are grouped under the **"Common" tile** in the Activities panel (as shown in Figure 3-12).

They help manage control flow, interact with users, perform loops, display messages, and handle variables.

List of Common Activities with Descriptions and Real-Time Use Cases

◆ 1. Write Line

- **Purpose:** Prints a message to the **Output panel**.
- **Use Case:** Used to **debug** workflows and create an **audit log**.
- **Configuration:**
 - **Text:** The message or variable to print (e.g., "Process Started").
- **Example:**
 - Write Line: "Login Successful" – logs a confirmation message.

◆ 2. Message Box

- **Purpose:** Displays a **popup box** with a message to the user.
- **Use Case:** Useful for alerts or confirmations.
- **Example:**
 - "Data processed successfully!"

◆ 3. Delay

- **Purpose:** Pauses the workflow for a **specific duration**.
- **Use Case:** Wait for a web page or app to load.
- **Example:**
 - Delay: 00:00:03 (3 seconds).

◆ 4. Input Dialog

- **Purpose:** Displays a **dialog box** to collect input from the user.
- **Use Case:** Get user name, email, or any runtime input.
- **Example:**
 - Ask the user: "Please enter your Employee ID".

◆ 5. Exit Loop

- **Purpose:** Exits from a loop prematurely.
- **Use Case:** Stop looping when a specific condition is met.
- **Example:**
 - Exit loop if user clicks cancel.

◆ 6. Repeat Number of Times

- **Purpose:** Repeats a set of activities a fixed number of times.
- **Use Case:** Retry login 3 times if it fails.
- **Example:**
 - Repeat 5 times → Send notification email.

◆ 7. If

- **Purpose:** Executes actions based on a **condition (True/False)**.
- **Use Case:** Decision making in workflows.
- **Example:**
 - If amount > 10000 → Send for approval.

◆ 8. Get Username/Password

- **Purpose:** Securely stores and retrieves user credentials.
- **Use Case:** Login to websites or applications.

◆ 9. Group

- **Purpose:** Groups multiple activities logically.
- **Use Case:** Organize related steps like "Login Process".

◆ 10. Save for Later

- **Purpose:** Assigns a value to be reused later in the workflow.
- **Use Case:** Store a user's name for sending emails later.

◆ 11. Skip Current

- **Purpose:** Skips the current loop iteration.
- **Use Case:** Skip processing if data is empty or invalid.

◆ 12. Switch

- **Purpose:** Executes one of multiple branches based on a value.
- **Use Case:** Choose actions based on department name or type.

◆ 13. Text to Left/Right

- **Purpose:** Extracts a portion of text to the **left or right** of a character.
- **Use Case:** Extract domain name from email.

◆ 14. Wait for Download

- **Purpose:** Waits for a file download to complete from any application.
- **Use Case:** Monitor downloads before processing.

◆ 15. Get Orchestrator Asset

- **Purpose:** Retrieves assets (like API keys, credentials) from Orchestrator.

These **common activities in UiPath StudioX** are essential building blocks used across all automation workflows. From logging and input handling to conditional logic and secure data management, they help in creating **efficient, robust, and user-friendly** automations. Understanding and using them effectively is key to mastering RPA development.

16. Modify Text Activity – Explanation

The **Modify Text** activity in UiPath allows you to change an existing piece of text by applying one or more modifications.

It's useful when you want to clean, reformat, or update text values during automation.

1. Configuration Options

a) Text to modify (Required)

- This is where you enter the **original text** you want to change.
- Example: " Hello World " (notice the extra spaces).

b) Add Modification (Optional)

This option lets you apply **different types of text changes**.

You can choose one or more from the following:

1. Find and Replace

- Search for a specific word or phrase in the original text and replace it with something else.
- Option for **exact match**.
- Example: Replace "World" with "UiPath" → "Hello UiPath".

2. Combine Text

- Add new text **before** or **after** the original text.
- Example: Add "Welcome " at the beginning → "Welcome Hello World".

3. Trim

- Remove extra spaces from the **left** and **right** ends of the text.
- Example: " Hello World " → "Hello World".

4. Text to Upper/Lowercase

- Change all text to **uppercase** or **lowercase**.
- Example: "Hello" → "HELLO" or "hello".

💡 **Test Button** – You can check how your modifications will look before running the automation.

c) Save results as (Optional)

- Stores the **modified text** in a variable for later use in the workflow.

- Example: Save the updated "HELLO UIPATH" into a variable called UpdatedText.

Quick Example:

Original text: " hello world "

Modifications:

1. Trim spaces → "hello world"
2. Replace "world" with "UiPath" → "hello UiPath"
3. Convert to uppercase → "HELLO UIPATH"

If you want, I can also make a **diagram showing how “Modify Text” works step-by-step** so it’s easier for students to understand.

Text to Left/Right Activity – Explanation

The **Text to Left/Right** activity is used to **split a text value into two parts** based on a **separator**.

It’s helpful when you have a string containing two different pieces of information separated by a symbol, word, or character.

1. Configuration Options

a) Full text (Required)

- The **original text** you want to split.
- Example: "Name:John".

b) Separator (Required)

- The text, symbol, or character that divides the original text into **two parts**.
- The activity will find the **first occurrence** of this separator and split the text at that point.
- Example: Separator = ":" will split "Name:John" into "Name" and "John".

c) Save text to left as (Required)

- The variable where the **text before the separator** will be stored.
- Example: "Name" → saved into variable LeftText.

d) Save text to right as (Required)

- The variable where the **text after the separator** will be stored.
- Example: "John" → saved into variable RightText.

✓ Quick Example:

```
vbnet
CopyEdit
Full text: "Email=abc@example.com"
Separator: "="
Left text: "Email"
Right text: "abc@example.com"
```

💡 Common Uses in Automation:

- Splitting names into first and last name.
- Separating keys and values in configuration data.
- Extracting domain names from email addresses.
- Breaking down product codes into category and ID.

Text to Left/Right Activity – Explanation

The **Text to Left/Right** activity is used to **split a text value into two parts** based on a **separator**. It's helpful when you have a string containing two different pieces of information separated by a symbol, word, or character.

1. Configuration Options

a) Full text (Required)

- The **original text** you want to split.
- Example: "Name:John".

b) Separator (Required)

- The text, symbol, or character that divides the original text into **two parts**.
- The activity will find the **first occurrence** of this separator and split the text at that point.
- Example: Separator = ":" will split "Name:John" into "Name" and "John".

c) Save text to left as (Required)

- The variable where the **text before the separator** will be stored.
- Example: "Name" → saved into variable `LeftText`.

d) Save text to right as (Required)

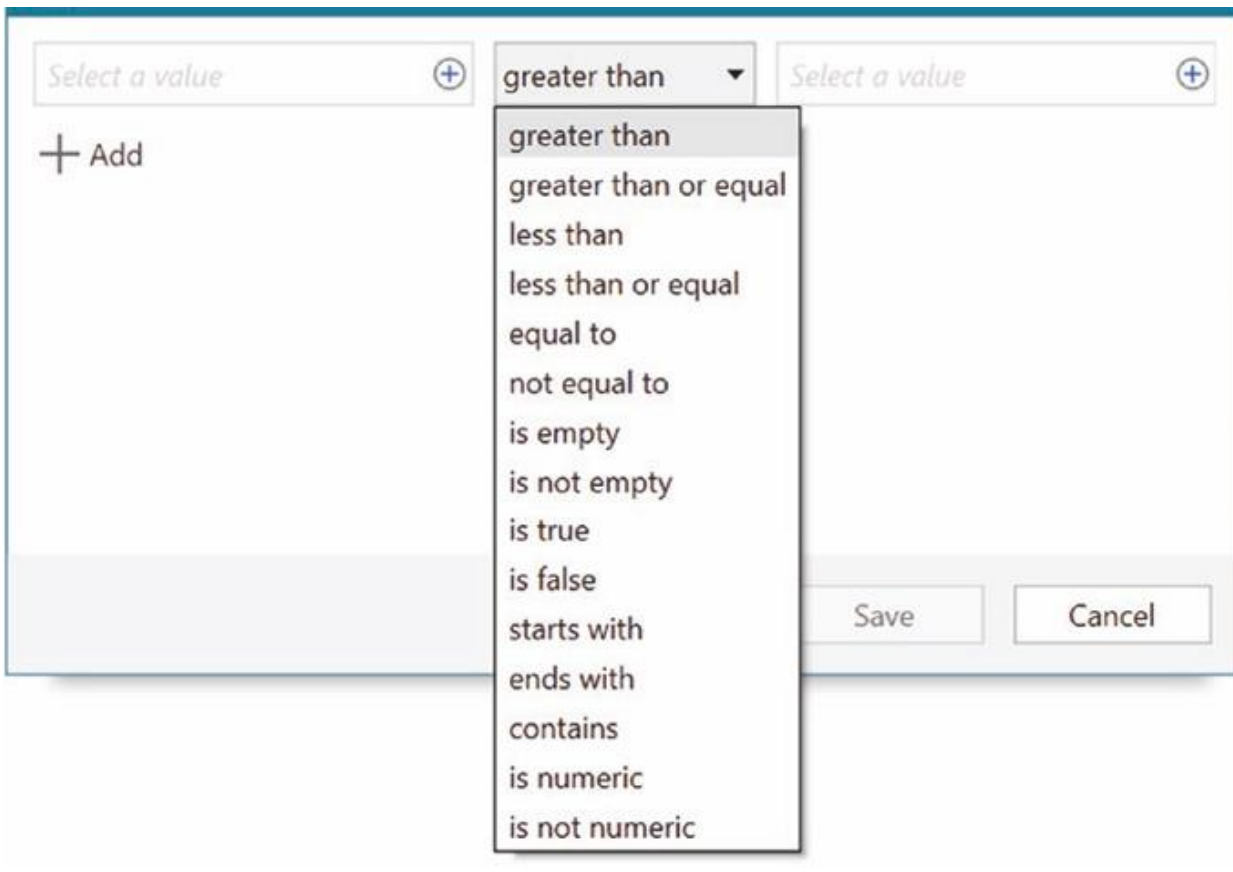
- The variable where the **text after the separator** will be stored.
- Example: "John" → saved into variable `RightText`.

✓ Quick Example:

```
vbnet
CopyEdit
Full text: "Email=abc@example.com"
Separator: "="
Left text: "Email"
Right text: "abc@example.com"
```

💡 Common Uses in Automation:

- Splitting names into first and last name.
- Separating keys and values in configuration data.
- Extracting domain names from email addresses.
- Breaking down product codes into category and ID.



Switch Activity - Explanation

The **Switch** activity allows you to define **multiple possible paths (cases)** and, based on the result of an expression, execute **only one matching case**.

It's like a **multi-choice decision** in automation.

1. When to Use Switch

- **If activity:** Best when you have **two** choices (True/False).
- **Switch activity:** Best when you have **three or more** possible outcomes.

Example:

If a column in Excel contains "Pending", "Approved", "Rejected", you can create a Switch to handle each status differently.

2. Configuration Options

a) TypeArgument (Optional - in Properties panel)

- Specifies **what type of value** the expression will return.
- Common types:
 - **Boolean** → For True/False results.
 - **Int32** → For numbers like 1, 2, 3.
 - **String** → For text values like "January", "Approved", etc.
- You must select the correct type so UiPath can match cases correctly.

b) Expression (Required – in Activity card)

- This is the condition that **returns a value** to be compared against the case values.
- Example:
 - Expression: `row("Status").ToString`
 - If the value is "Approved", it will go to the "Approved" case.

c) Add New Case (Required – Default case must exist)

- **Case Value:** The specific value you want to match with the expression result.
- **Default Case:** Executes when **no case matches**.
- You can add **multiple cases** and place different activities inside each one.

3. How It Works – Example

Scenario: Based on a student's grade, perform different actions.

- **TypeArgument:** String
- **Expression:** `grade`
- **Cases:**
 - "A" → Show "Excellent".
 - "B" → Show "Good".
 - "C" → Show "Needs Improvement".
 - **Default** → Show "Invalid Grade".

💡 Benefits of Using Switch in UiPath:

- Cleaner than multiple nested If statements.
- Easier to read and maintain.
- Supports **scalable decision-making** when many possible outcomes exist.

Repeat Number Of Times Activity – Explanation

The **Repeat Number Of Times** activity is used to **repeat one or more activities** a set number of times. It's useful when you know **exactly how many times** you need to perform a task.

1. Configuration Options

a) Repeat number of times (Required)

- Specifies **how many times** the loop should run.
- Usually set using the **Number Builder** option.
- Example: Set to 5 → the activities inside will run 5 times.

b) For each (Required)

- Represents the **loop index** — which keeps track of the current iteration.
- By default, it is called **CurrentItem**, but you can rename it for clarity (e.g., Counter, Vendor, Employee).

- Example:
 - 1st run → `CurrentItem = 1`
 - 2nd run → `CurrentItem = 2`
 - ... and so on.

2. Using the Index

You can use the index to:

- **Skip** a specific iteration → with **Skip Current**.
- **Stop** the loop early → with **Exit Loop**.
- Make decisions inside the loop based on the current count.

✓ Example – Sending an Email 3 Times

- **Repeat number of times:** 3
- **For each:** Counter
- Inside the loop: Send email → Subject: "Reminder " + Counter
- Output:
 1. "Reminder 1"
 2. "Reminder 2"
 3. "Reminder 3"

💡 Typical Use Cases:

- Performing a task a fixed number of times (e.g., retry attempts).
- Processing a limited batch of items.
- Testing automation logic by repeating actions.

Skip Current Activity – Explanation

The **Skip Current** activity is used to **skip the current iteration** of a loop and move to the **next iteration** without executing the remaining activities in the current cycle.

It's like saying:

“Don't do anything else for this item — just move on to the next one.”

1. When to Use

- You are looping through a list of items or repeating tasks.
- A **condition** tells you that the current item **should be ignored**.
- Instead of exiting the whole loop, you just **skip this one**.

Example:

If looping through numbers 1 to 10 and you only want **even numbers**, you can **Skip Current** when the number is odd.

2. Configuration

- **No configuration options** — you simply place it in your loop.
- **Requirement:** Must be inside a repeating activity like:

- **Repeat Number Of Times**
- **For Each**
- **While**
- **Do While**

✓ **Example – Processing Employees**

- **For Each** employee in a list:
 - If the employee's department is "HR" → **Skip Current**.
 - Otherwise → process the employee data.

💡 **Key Difference:**

- **Skip Current** → Skips just one iteration, loop continues.
- **Exit Loop** → Stops the loop entirely.

1. Get Username/Password Activity

Purpose

- Retrieves stored credentials (**username and password**) from **Windows Credential Manager** for use in automation.
- Often used with **Type Into** activity to log into applications without hardcoding passwords.

Configuration Steps

a) Saved Credentials (Required)

- Select **existing credentials** or **add new credentials**.
- To add:
 1. Click the **Key icon** → opens **UiPath Credential Manager**.
 2. Click + **Add Credential** → enter **username** and **password**.
 3. Credentials are stored both in **UiPath Credential Manager** and **Windows Credential Manager**.

💡 If you add credentials directly to Windows Credential Manager, **prefix the name with UiPath** so UiPath can find it.

b) Saved for later as (Required)

- Stores the retrieved credentials in a variable for later use.
- From that variable, you can access:
 - **Username**
 - **Password**

✓ **Example Usage:**

1. Use **Get Username/Password** to fetch credentials for `UiPath_GmailLogin`.
2. Use **Type Into** activity to type `username` and `password` into the login form.

2. Get Orchestrator Asset Activity

Purpose

- Retrieves **assets** (credentials, text, numbers, etc.) from **UiPath Orchestrator** for use in automation.
- Best for storing **shared values** across multiple robots/automations.

Configuration Steps

a) Asset Name (Required)

- Name of the asset in Orchestrator.
- Example: "ERP_Admin_Credentials".

b) Orchestrator Folder Path (Required – in Properties panel)

- Path to the folder in Orchestrator where the asset is stored.

c) Save for later as (Optional)

- Stores the retrieved asset in a variable for reuse in the workflow.

✓ Example Usage:

1. Create an asset in Orchestrator named `InvoiceAPIKey`.
2. In UiPath, use **Get Orchestrator Asset** to retrieve it.
3. Store it in a variable → Use in API calls.

💡 Key Difference:

- **Get Username/Password** → Fetches from local **Windows Credential Manager**.
- **Get Orchestrator Asset** → Fetches from **UiPath Orchestrator** (cloud/on-prem).