CHAPTER 1

Robotic Process Automation: Overview

To remain competitive in today's hyper-automated world, digital transformation initiatives have become a primary focus across various industries. Traditionally, C-suite executives are increasingly interested in lowering operational expenditures, particularly costs associated with the human workforce. In addition, business leaders are simultaneously focused on driving increased efficiencies and employee satisfaction across the enterprise. As a result, companies are undergoing a higher level of scrutiny surrounding existing business processes to seek out opportunities for automation at a global scale. One area of process optimization is that which exists for desk-level procedures typically executed by business stakeholders. With such a high percentage of automation opportunities executed by the business stakeholders, a bottom-up approach is commonly seen in which the workforce chooses tasks to automate based on their individual needs. Not only can they help in identifying opportunities to automate but also create automations themselves so that they may focus on high-value tasks. With the advent of intelligent automation, specifically Robotic Process Automation (RPA), companies now have a proven way to automate business processes at the keystroke level.

A. Javed et al., *Robotic Process Automation using UiPath StudioX*,
https://doi.org/10.1007/978-1-4842-6794-3_1

UiPath provides a technology that enables the automation of business processes traditionally performed by business users, using configurable software referred to as "robots." UiPath's development platform, StudioX, is extremely flexible and user-friendly as it is a low-code/no-code solution. The software allows users the ability to interact with systems via a robot which leverages the users' own credentials or can be configured with distinct credentials and specific permissions. Robotic Process Automation tools interact with an application in the same way as end users do, through interactions with the user interface (UI), as well as through the back end of a system.

From a compliance and risk perspective, robots can only execute tasks that are specifically designed with the virtual robot worker in mind. The robot's access to both internal and external systems is limited to the design of the robot workflow, which demonstrates the rules-based nature of the software. The robot can also be designed to prompt the user for input or incorporate artificial intelligence (AI) to handle more cognitive tasks. As a result, Robotic Process Automation enables automation of the manual, repetitive tasks that are typically a fundamental component of a business user's daily job responsibilities.

A fundamental premise of Robotic Process Automation software is the belief that robotic software is designed to complement the human workforce, by empowering organizations with the ability to upskill employees to build simple automations or route more complex automations to a set of developers reporting into a core automation team. Adding Robotic Process Automation functionality into a business department can maximize the efficiency of employee outputs, minimize the risk of human error, and mitigate the number of tedious, manual processes employees are expected to execute, thus increasing the potential for a higher level of employee satisfaction.

Return on Investment (ROI)

In addition to enabling the automation of repetitive tasks, Robotic Process Automation software can provide substantial return on investment to both business process owners and the enterprise at large. Robotic Process Automation software allows firms to automate manual processes in a cost-efficient manner, due to the fact the price point of the RPA software is typically lower than that of traditional business applications. Robots are beneficial in minimizing the costs typically incurred in automation projects, as Robotic Process Automation tools can leverage existing infrastructure architecture without impacting live systems.

The infrastructure necessary to support robots is considerably minimal when compared to other tools, as robots can either run on an end user's desktop (attended automation) or a virtual machine (unattended automation). One of the many benefits of Robotic Process Automation is the ability users to dictate whether a human or a robot will be responsible for executing a particular step of the process within a given workflow. In addition, workflows can be customized to indicate when robots encounter changes in each system including routine software upgrades whereby elements of the user interface might deviate from previous versions. A significant benefit of Robotic Process Automation is the ability users to create workflows to support a dynamically changing environment with minimal impact to underlying infrastructure capabilities.

Automation Types

Moving forward, we will dive into the nuances of unattended robots and attended robots, to understand how the distinction between the two types of automation is driving a new approach to enabling business process automation through citizen development (business users with the ability to build automations). Robotic Process Automation can be

leveraged to automate a wide variety of processes including but not limited to payroll processing, customer service, advertising operations, report aggregation, and vendor onboarding. Robotic Process Automation also offers a wide variety of automation deployment models which can be used interchangeably to automate processes across the business including

- Attended robots that reside on the end user's computer or virtual machine for the purpose of automating simple manual processes that can be triggered by the actions of the user.
- Unattended robots that can be provisioned to reside on machines based on-premises (physical server based) or off-premises (virtual machines/cloud based) for the purpose of automating more complex back-office functions commonly scheduled to run based on a time or queue. Typically, unattended automation lends itself to more data-intensive tasks and processes with higher transaction volumes such as batch jobs.
- Hybrid robots that reside on a combination of end user and on-premises/off-premises solutions to enable a combination of attended and unattended style processing to enable the end-to-end automation of processes that require both human support and back-end functionality.

Each automation deployment model allows the end user the ability to determine the best way to interact with a robot based on the task at hand, alongside careful consideration of the existing variables in each environment. The various automation deployment models can be leveraged interchangeably as a part of a holistic enterprise-level automation platform and digital transformation strategy. As we move

forward, we'll focus on features and hands-on exercises specific to the Robotic Process Automation industry leader, UiPath, to discuss the unique value proposition the company offers citizen developers through the use of StudioX.

UiPath StudioX

UiPath is a global Robotic Process Automation software company based out of Romania. The company was founded in 2005 by Daniel Dines. The company originally offered automation libraries and software as an outsourced service, but quickly positioned itself to become an industry leader through a customer-centric model designed to democratize access to Robotic Process Automation capabilities. Through a robust product road map and unique approach to empower business users with the ability to automate simple business processes via StudioX, UiPath's enterprise platform demonstrates the seamless fusion that exists between business processes and automation capabilities.

StudioX is one product of UiPath's Robotic Process Automation platform designed to enable business users to build automation without the need for a traditional development background. The StudioX functionality includes a no-code interface with out-of-the-box drag-and-drop functionality to facilitate ease of use. In addition, StudioX contains predesigned templates and native integrations with common business applications such as the Microsoft Office suite to facilitate faster development of automation workflows. Business users can deploy a robot directly to a local machine, such as a desktop which removes the need for traditional IT deployment support. In addition, governance functionality is also built into the StudioX framework to allow auditing capabilities to ensure that existing company compliance protocols remain intact. Regarding the scheduling and sharing of automations, users can complete both tasks through the UiPath Assistant and Orchestrator components of UiPath.

One of the key elements that demonstrates the flexibility of StudioX is the fact the tool allows business users a user-friendly way to learn how to build automations that are beneficial to their job functions while simultaneously learning a new technical skill. In a world where technical prowess has become increasingly important, providing employees an opportunity to leverage Robotic Process Automation tools can help individuals to feel empowered and more satisfied, potentially leading to less attrition. The citizen developer model is the methodology by which business users are trained on the skills required to build automations while also being provisioned access to RPA tools to begin the development of robot workflows. As Robotic Process Automation continues to expand across a wide variety of industries, it will be important to continue to expand the knowledge of business users with tools such as StudioX to provide a wealth of benefits at an organizational level.

In the rest of the book, we will explore hands-on exercises with detailed reference guides for various activities and sample files to help you as you work to build your first RPA robots in StudioX. The goal of each chapter is to provide real-world business process scenarios for readers to reference as Robotic Process Automation learning tools. As you work through the exercises, make a note of any challenges you encounter to allow time to reflect on possible ways to solve any roadblocks you may have. This book is intended for both the business user looking to learn how to leverage StudioX for the first time and the experienced RPA developer looking to build upon existing knowledge to automate manual, repetitive tasks across your organization. As you step into the future of working with robots, remember that you have taken an important step in the journey to democratize automation and heighten your technical skill set. So, let us get started.

CHAPTER 3

Common Concepts

This chapter is going to introduce you to the concepts, properties, and activities commonly used for building automation in UiPath StudioX.

Learning Objectives

At the end of this chapter, you will learn how to

- Use project Notebook for using formulas and creating custom formulas
- Pass data to activities
- Store data returned from activities for use later
- Configure common activities in automation

Notebook

The Notebook is a great concept that makes UiPath StudioX indeed no code.

Most business users and citizen developers are familiar with Excel. By default, each UiPath StudioX automation has access to an Excel file, known as Notebook. You can use the default Notebook in your project or configure a custom Excel file as your project Notebook.

A. Javed et al., *Robotic Process Automation using UiPath StudioX*,
https://doi.org/10.1007/978-1-4842-6794-3_3

Default Notebook

The default Notebook has some pre-built formulas for manipulating Date, Text, Number, and File values. Figure 3-1 shows the Text worksheet of the Notebook that contains some pre-built formulas to manipulate text values.

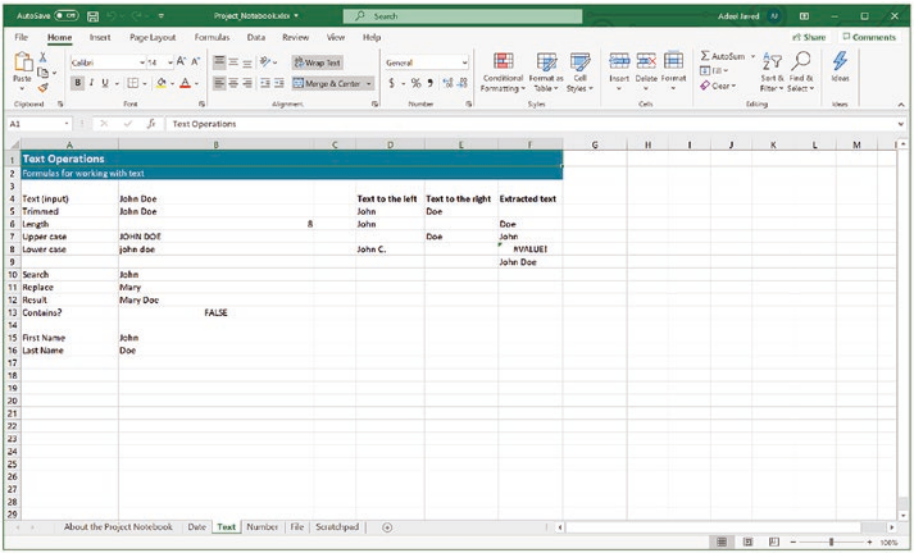

Figure 3-1. *Text worksheet of Project Notebook*

For additional custom fields and formulas, you can use the Scratchpad worksheet or add more worksheets to the Notebook and define your formulas.

To pass data to your custom formulas in Notebook, you can use the `Use Excel` and `Write Cell` activities to write data to a specific cell (see Chapter 7).

Any activity that requires input data can reference the Notebook to retrieve the data inputs.

Figure 3-2 shows an example of passing data to a custom formula in Notebook and then reading the result from Notebook. In this case, we are writing a name, John Doe, to a cell in the Text worksheet. Multiple formulas use this cell. In our case, we are interested in splitting the name into first name and last name and then printing it in the `Output` panel, so we output the cell that contains the First Name.

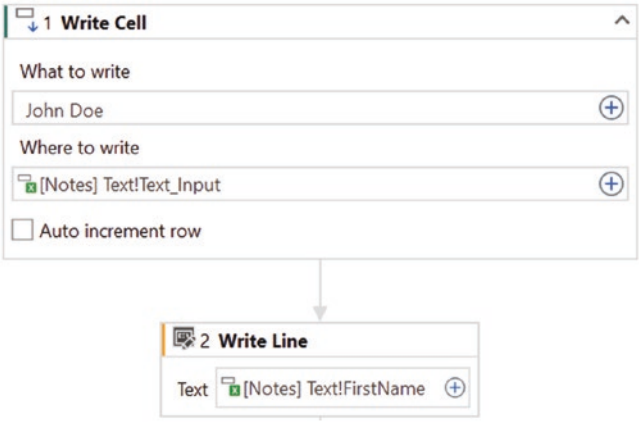

Figure 3-2. *Passing data to and reading data from Project Notebook*

Custom Notebook

You have the option to configure a custom Excel file as your project Notebook.

To configure a custom Excel as your project Notebook, click the `Configure Notebook` option from the Notebook menu item in the ribbon, and then select the custom Excel file in `Notebook file` field, shown in Figure 3-3.

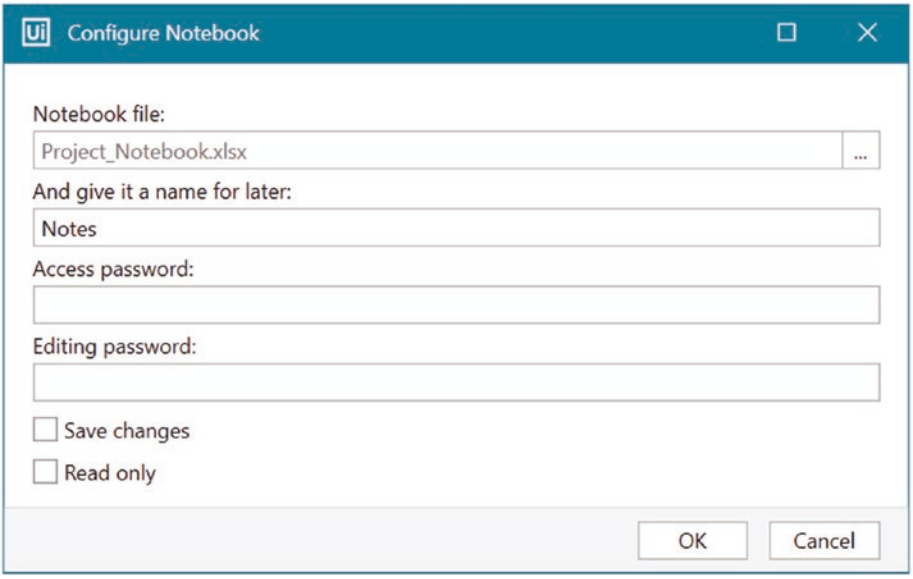

Figure 3-3. *Configure a custom Excel as Project Notebook*

Activity Inputs

Some of the activities require data inputs for processing. For example, you can provide text as an input to a `Write Line` activity. It will print the value in the Output panel, or you can provide text as an input for the `Type Into` activity, and it will enter the value in a form field. Each activity that requires input data gives you options to specify the input data. Figure 3-4 shows a sample menu for data input options. The data input options menu becomes available when you click the `Plus` icon in the field.

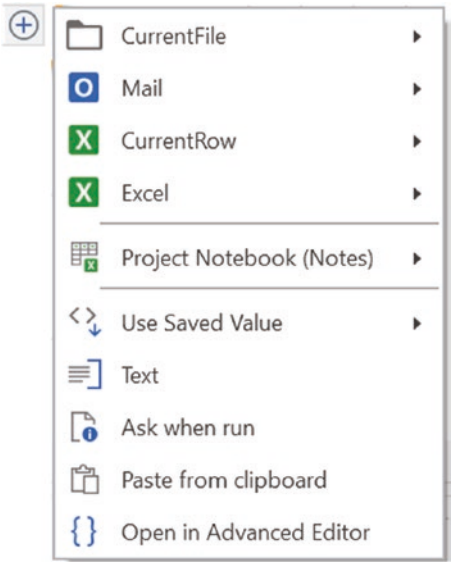

Figure 3-4. *Sample menu of data input options*

Excel: Allows you to specify a cell address, range, table, custom input, or even a sheet from the Notebook or any other Excel as an input to an activity. You can also select values from an individual Excel row as data inputs (see Chapter 7 for details).

Mail: Allows you to specify data from an email as an input to an activity (see Chapter 5 for details).

File: Allows you to specify attributes of a file as an input to an activity (see Chapter 9 for details), for example, the last modified date of a file or the full path of the file.

Use Saved Value: Allows you to specify data returned from a previous activity as an input to another activity.

Text: Allows you to specify a static or dynamic text string as an input to an activity. This option opens the `Text Builder` dialog. Using the `Text Builder`, you can combine multiple inputs into a single text string. For example, Figure 3-5 shows that two data inputs have been combined with some static text to return a single text input.

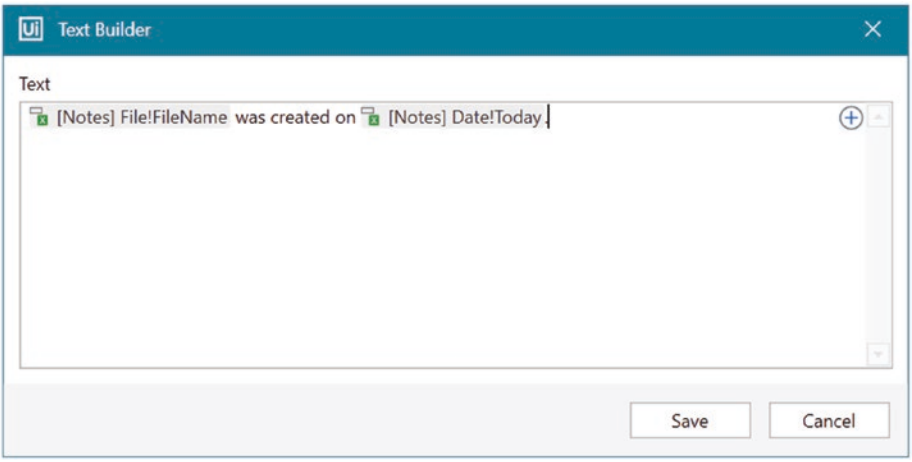

Figure 3-5. *Text input option*

Number: Allows you to specify a numeric calculation that can be used to input a number, basic calculations, or complex formulas from Excel to an activity.

Choose Date/Time: Allows you to specify a date or time as an input to an activity. Figure 3-6 shows the option to specify the date, while Figure 3-7 shows the option to specify the time.

Figure 3-6. *Date input option*

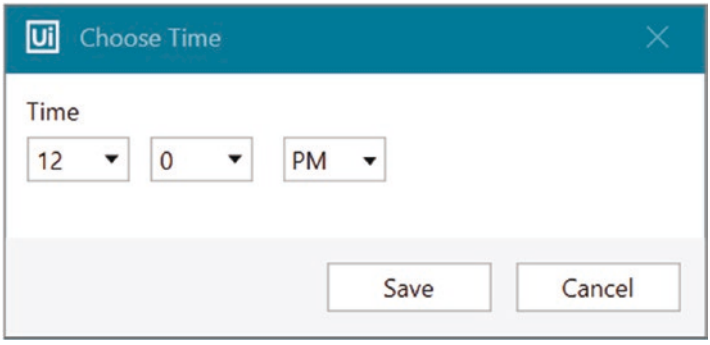

Figure 3-7. *Time input option*

Choose Duration: Allows you to specify a duration as an input to an activity. Figure 3-8 displays the option to specify the duration.

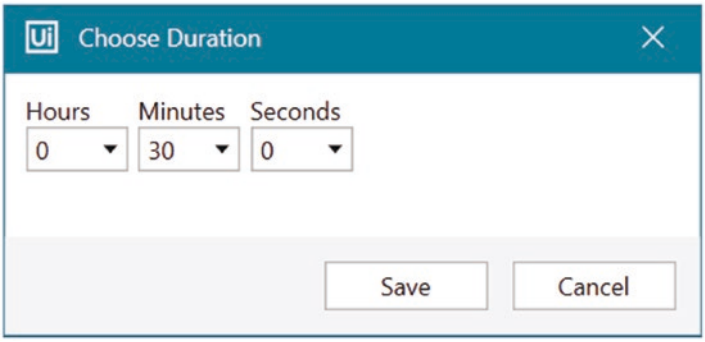

Figure 3-8. *Duration input option*

Ask when run: Allows you to specify a value as an input to activity at runtime. This option works well for attended robots.

Paste from clipboard: Allows you to paste clipboard data as an input to an activity.

Open in advanced editor: Allows you to enter a VB expression to specify the input value.

Activity Outputs

Some activities, once they have completed processing, return data. For example, a `Read Text File` activity will return contents of a text file, or `Get Text` activity will return contents of a form field. Each activity that returns data allows you to specify how you want to store that data. Figure 3-9 shows a sample menu of data output options. This data output options menu becomes available when you click the `plus` icon.

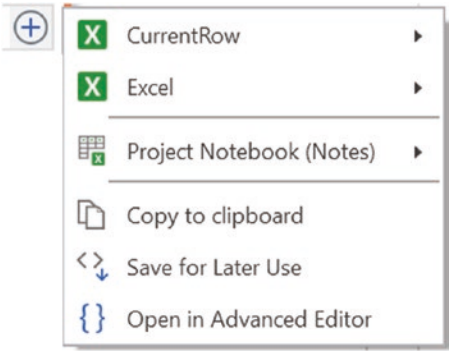

Figure 3-9. *Sample menu of data output options*

Excel: Allows you to specify a cell address, range, table, or sheet in the Project Notebook or Excel file from the parent `Use Excel File` activity where you want to store the data returned by an activity. Cells in individual Excel rows can also be specified (see Chapter 7 for details).

Save for Later Use: Allows you to specify a variable that will store data returned by an activity. Variables are accessible from the Data Manager panel. You can name or rename a variable to make it easy to understand. Figure 3-10 shows the option to store data in a variable.

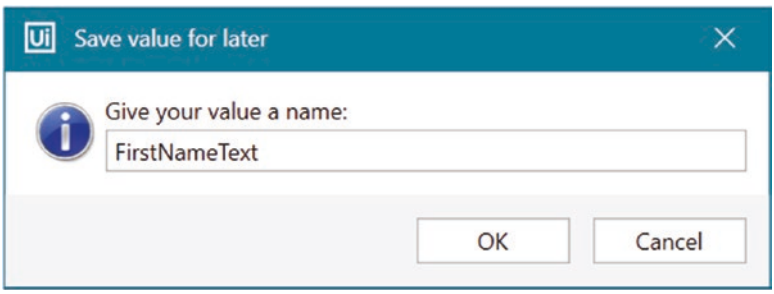

Figure 3-10. *Save for Later Use option*

Copy to clipboard: Allows you to copy data returned by an activity to your clipboard. This data can be accessed later using the `Paste from clipboard` option.

Common Properties

Common properties, as shown in Figure 3-11, can be accessed from the `Properties` panel under the `Common` section. Common properties can vary between activities.

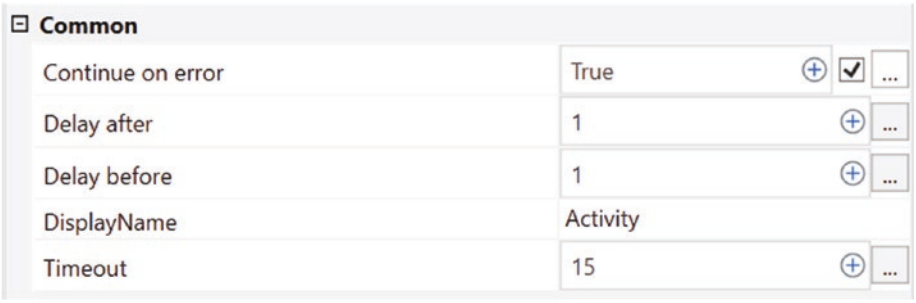

Figure 3-11. *Common properties for activities*

Table 3-1 provides a brief description of each property.

Table 3-1. *Common properties*

Property	Description
Continue on error	This property allows you to specify (true or false) if the automation should continue running, i.e., proceed to the next activity even if the selected activity encounters an error during execution.
Delay after	This property allows you to specify, in seconds, if the automation should wait after completing the selected activity and before starting the next activity. By default, the value is 0.3 seconds.
Delay before	This property allows you to specify, in seconds, if the automation should wait before starting the selected activity. By default, the value is 0.2 seconds.

(*continued*)

Table 2-2. (*continued*)

Property	Description
DisplayName	This property allows you to update the display name of the selected activity in the Designer panel.
Timeout	This property allows you to specify, in seconds, how long should the automation wait for the selected activity to complete before timing out. By default, the value is 30 seconds.

Common Activities

Activities that are commonly used across all automation types can be found under the **Common** tile, as shown in Figure 3-12. The following sections will provide instructions on how to configure and use each of these activities.

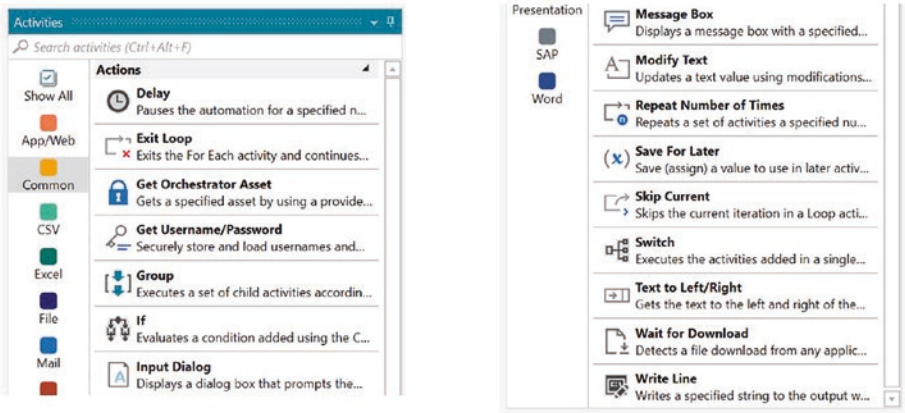

Figure 3-12. *Common activities for automation*

Write Line

The **Write Line** activity allows you to print a specified message to the Output panel.

Tip The Write Line activity is beneficial during and after the development phase of automation. Adding this activity can help you follow the flow of automation and debug any issues that might arise during development. Once you have developed the automation, this activity can help you create an audit log of your automation, which is essential in regulated industries.

Configuration

This section provides instructions on how to configure a **Write Line** activity, shown in Figure 3-13.

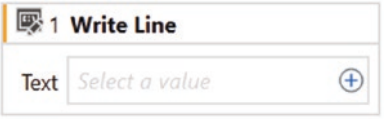

Figure 3-13. *Activity card for Write Line*

Text: This is an optional configuration available on the activity card. The text you specify in this configuration is going to be printed in the `Output` panel.

EXERCISE

Goal: Use the `Write Line` activity to print "Automation has started processing data for <Current Date>." message in the Output panel. The <Current Date> should be replaced with the actual date when the automation runs.

Source Code: Chapter_3-WriteLineExercise

Setup: Here are step-by-step implementation instructions:

1. In `StudioX`, add the `Write Line` activity to a blank process.
2. Next, click the `Plus` icon and select the `Text` option. In the `Text Builder`, enter `Automation has started processing data for`.
3. Next, from within the `Text Builder`, click the `Plus` icon and select `Project Notebook` ➤ `Date` ➤ `Today`. Your final text should be like `Automation has started processing data for [Notes] Date!Today`.

Once you have completed the exercise, the final configuration of the **Write Line** activity should resemble Figure 3-14. Once you run this automation, the automation will print the message in the Output panel, shown in Figure 3-15.

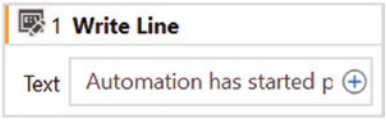

Figure 3-14. *Configuration of Write Line activity exercise*

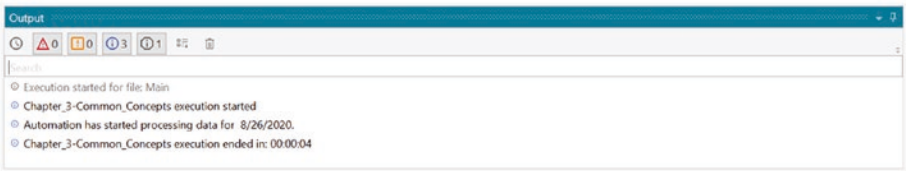

Figure 3-15. *The output of the Write Line activity exercise*

Message Box

The **Message Box** activity allows you to pause the automation and display text with specified buttons in a message box.

Tip The Message Box activity is particularly useful during the development phase of automation. Adding this activity can help you follow the flow of automation and debug any issues that might arise during development.

Configuration

This section provides instructions on how to configure a **Message Box** activity, shown in Figure 3-16.

Figure 3-16. *Activity card for Message Box*

Text: This is a required configuration available on the activity card. The text you specify here is going to be displayed in the message box.

Buttons: This is an optional configuration available on the `Properties` panel. This configuration allows you to specify buttons to display on the message box. Figure 3-17 shows all available options for buttons. By default, the selection is set to the `Ok` button.

Figure 3-17. *Button options for the Message Box activity*

Caption: This is an optional configuration available on the `Properties` panel. This configuration allows you to specify a title/label for the message box. The default caption is `Message Box`.

ChosenButton: This is an optional configuration available on the `Properties` panel. This configuration allows you to save the value of the button clicked (from the Buttons configuration). You can later use this value to make decisions for the automation workflow path.

TopMost: This is an optional configuration available on the `Properties` panel. This configuration allows you to specify if the message box should always be brought to the foreground.

EXERCISE

Goal: Use the Message Box activity to display a message box that asks the user to confirm this message "Word file processing is complete; Do you want to start Excel file processing?".

Source Code: Chapter_3-MessageBoxExercise

Setup: Here are step-by-step implementation instructions:

1. In StudioX, add the Message Box activity to a blank process.
2. Next, click the Plus icon, select Text option, and enter Word file processing is complete; Do you want to start Excel file processing?.
3. Next, from the Properties panel, select the YesNo option for the Buttons configuration.
4. Next, from the Properties panel, set Caption field to Confirmation.
5. Next, from the Properties panel, click the Plus icon in the ChosenButton field and select the Save for Later Use option. Name the variable as ChosenButton.
6. Next, add a Write Line activity, after the Message Box activity. Click the Plus icon and select Use Saved Value ➤ ChosenButton.

Once you have completed the exercise, the final configuration of the **Message Box** activity should resemble Figure 3-18.

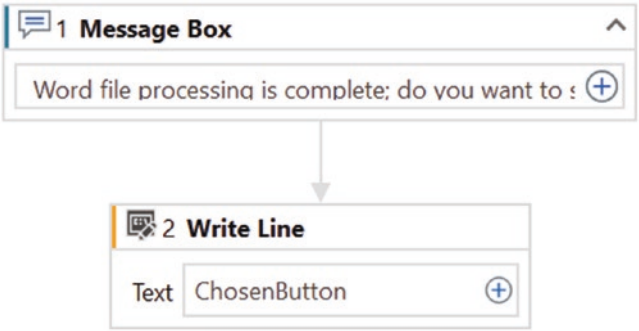

Figure 3-18. *Configuration of Message Box activity exercise*

Once you run the automation, it will display a message box with Yes/No confirmation buttons, as shown in Figure 3-19. Whichever button you click, the automation will print its value in the Output panel, as shown in Figure 3-20.

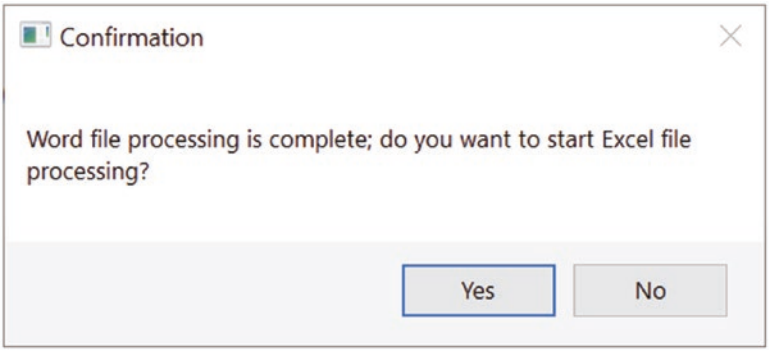

Figure 3-19. *Confirmation message box*

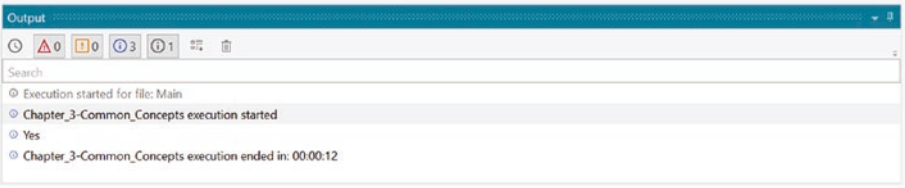

Figure 3-20. *The output of the automation*

Input Dialog

The **Input Dialog** activity allows you to prompt the user to provide input. This activity is particularly useful in attended automation for processes that require user input.

Configuration

This section provides instructions on how to configure an **Input Dialog** activity, shown in Figure 3-21.

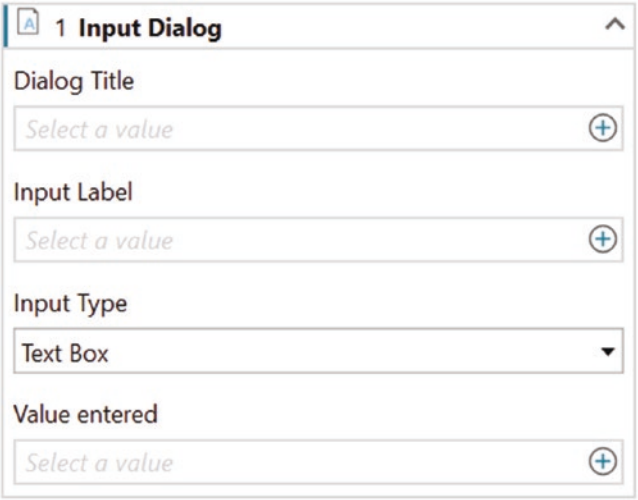

Figure 3-21. *Activity card for Input Dialog*

Dialog Title: This is an optional configuration available on the activity card. The text you specify here is going to be displayed in the title of the input dialog.

Input Label: This is an optional configuration available on the activity card. The text you specify here is going to be displayed as the label for input field.

Input Type: This is a required configuration available on the activity card. There are two options, `Text` and `Multiple Choice`. By default, the value is set to `Text`, which prompts the user with a freeform text field to provide input. The `Multiple Choice` option allows you to specify a set of choices, and the user is required to select from one of the options.

Input options (separate with ;): This is an optional configuration available on the activity card. This configuration is only available when you select `Multiple Choice` option in the `Input Type` field. This configuration allows you to specify a list of choices for the user to choose from. Multiple choice options are separated by ;.

Value entered: This is an optional configuration available on the activity card. This configuration allows you to save the provided input value in a variable. You can later use this value to make decisions for the automation workflow path.

EXERCISE

Goal: Use the `Input Dialog` activity to prompt the user to select the next action for an automation.

Source Code: Chapter_3-InputDialogExercise

Setup: Here are step-by-step implementation instructions:

1. In `StudioX`, add an `Input Dialog` activity to a blank process.
2. In the `Dialog Title` field, click the `Plus` icon, select `Text` option, and enter `Next Action`.
3. In the `Input Label` field, click the `Plus` icon, select `Text` option, and enter `Please select how you want to proceed with the automation`.
4. From the `Input Type` field, select `Multiple Choice`.
5. In the `Input options` field, click the `Plus` icon, select `Text` option, and enter `Process now;Process later;Exit`.
6. In the `Value entered` field, click the `Plus` icon, and select the `Save for Later Use` option. Name the variable as `InputDialogResult`.
7. Next, add a `Write Line` activity after the `Input Dialog` activity. Click the `Plus` icon, and hover over `Use Saved Value` to select `InputDialogResult`.

Once you have completed the exercise, the final configuration of the **Input Dialog** activity should resemble Figure 3-22.

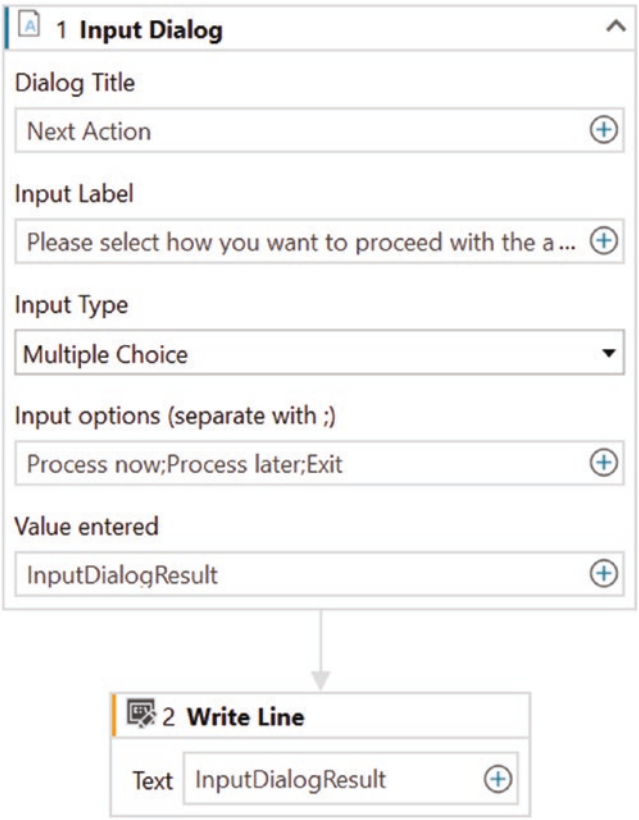

Figure 3-22. *Configuration of Input Dialog activity exercise*

Once you run the automation, it will display an input dialog box with three choices, as shown in Figure 3-23. Whichever option you choose, the automation will print its value in the Output panel.

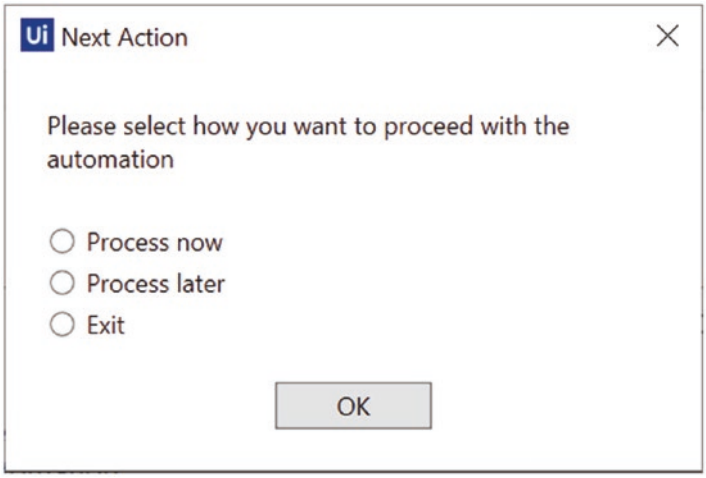

Figure 3-23. *Input dialog*

Modify Text

The **Modify Text** activity allows you to modify the input text value by performing different operations.

Configuration

This section provides instructions on how to configure a **Modify Text** activity, shown in Figure 3-24.

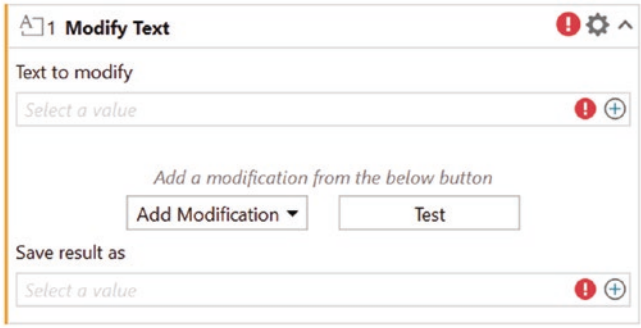

Figure 3-24. *Activity card for Modify Text*

Text to modify: This is a required configuration available on the activity card. This configuration allows you to specify the original text that will be modified by the activity.

Add Modification: This is an optional configuration available on the activity card. This configuration allows you to specify one or more modifications that you want to perform on the text. Figure 3-25 shows all the modifications available.

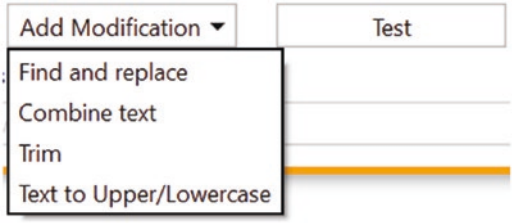

Figure 3-25. *Button options for Message Box activity*

`Find and replace` modification allows you to specify text to search for in the original text and text to replace it with. You can also specify if it should be an exact match.

`Combine text` modification allows you to add new text at the beginning or end of text.

`Trim` modification allows you to trim white spaces from left and right of text.

`Text to Upper/Lowercase` modification allows you to change the case of the text to either all uppercase or all lowercase.

You can also choose the `Test` button to check your modification logic while designing the automation.

Save results as: This is an optional configuration available on the activity card. This configuration allows you to save the value of the modified text for use later.

EXERCISE

Goal: Use the `Modify Text` to change replace % with _ in the input value, trim the text, convert it to uppercase, and then print the result.

Source Code: Chapter_3-TextManipulationExercise

Setup: Here are step-by-step implementation instructions:

1. In `StudioX`, add the `Modify Text` activity to a blank process.
2. In the `Text to modify` field, click the `Plus` icon, and select `Ask when run` option.
3. Next, from the `Add Modification` dropdown, select `Find and replace` option. This will add a child activity.
4. In the `Search for` field of `Find and replace text` child activity, click the `Plus` icon, select `Text` option, and type %. This specifies what text to look for.
5. In the `Replace with` field of `Find and replace text` child activity, click the `Plus` icon, select `Text` option, and type _. This specifies, that if found, what to replace % with.
6. Next, from the `Add Modification` dropdown, select `Trim` option. This will add a child activity. Leave the configurations as is.
7. Next, from the `Add Modification` dropdown, select `Text to Upper/Lowercase` option. This will add a child activity. Leave the default configurations of `Uppercase` as is.
8. Next, in the `Save result as` field, click the `Plus` icon and select the `Save for Later Use` option. Name the variable as `OutputText`.

9. Next, add a `Write Line` activity after the `Modify Text` activity. Click the `Plus` icon and select `Use Saved Value` ➤ `OutputText`.

Once you have completed the exercise, the final configuration of the **Modify Text** activity should resemble Figure 3-26.

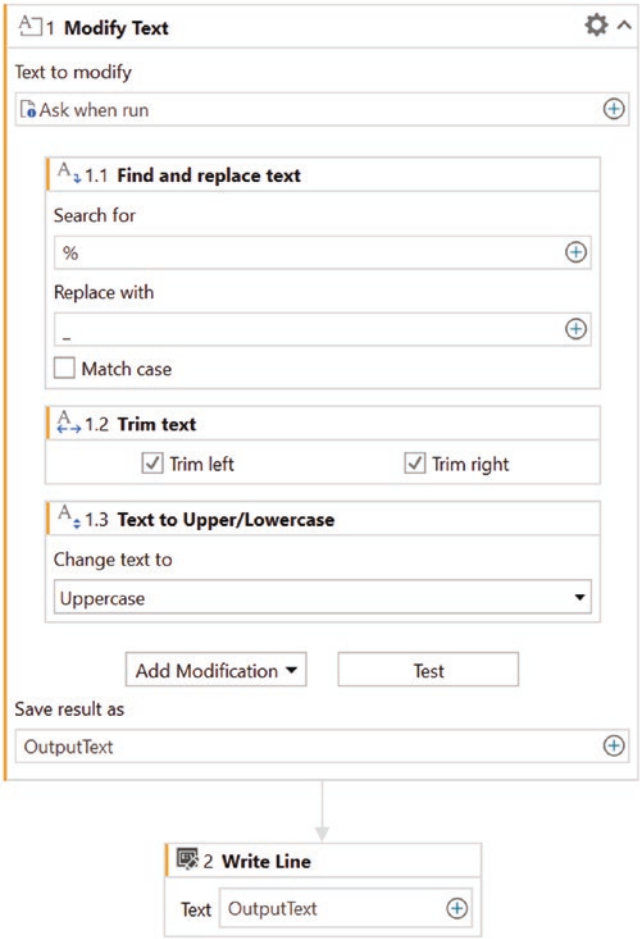

Figure 3-26. *Configuration of Modify Text activity exercise*

Once you run the automation, it will display an input dialog. Type `New%Hires%2020.xlsx` as the value, as shown in Figure 3-27, and click OK. Once all the text modifications have been performed, the result will be displayed in the Output panel, as shown in Figure 3-28.

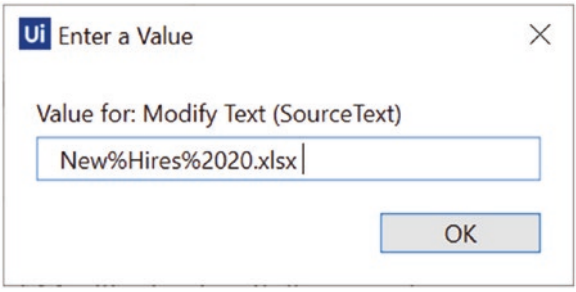

Figure 3-27. *Input value*

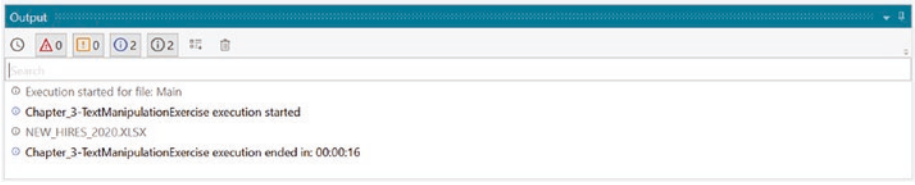

Figure 3-28. *The output of the automation*

Text to Left/Right

The **Text to Left/Right** activity allows you to split a text value into two text values using a separator.

Configuration

This section provides instructions on how to configure a **Text to Left/Right** activity, shown in Figure 3-29.

Figure 3-29. *Activity card for Text to Left/Right*

Full text: This is a required configuration available on the activity card. This configuration allows you to specify the text that will be split into two by the activity.

Separator: This is a required configuration available on the activity card. This configuration allows you to specify the text to use as a separator. The activity will use the first occurrence of the separator to split the original text.

Save text to left as: This is a required configuration available on the activity card. This configuration allows you to save the text to the left of the separator for later use.

Save text to right as: This is a required configuration available on the activity card. This configuration allows you to save the text to the right of the separator for later use.

EXERCISE

Goal: Use the `Text to Left/Right` to split the input text value using % as a separator and print the two new text values.

Source Code: Chapter_3-TextManipulationExercise

Setup: Here are step-by-step implementation instructions:

1. In StudioX, add the `Text to Left/Right` activity to a blank process.
2. In the `Full text` field, click the `Plus` icon, and select `Ask when run` option.
3. In the `Separator` field, click the `Plus` icon, select `Text` option, and type %.
4. In the `Save to left as` field, click the `Plus` icon, and select the `Save for Later Use` option. Rename the value as `TextToLeft`.
5. In the `Save to right as` field, click the `Plus` icon, and select the `Save for Later Use` option. Rename the value as `TextToRight`.
6. Next, add a `Write Line` activity after the `Text to Left/Right` activity. Click the `Plus` icon and select `Use Saved Value` ➤ `TextToLeft`.
7. Next, add a `Write Line` activity after the `Write Line` activity. Click the `Plus` icon and select `Use Saved Value` ➤ `TextToRight`.

Once you have completed the exercise, the final configuration of the **Text to Left/Right** activity should resemble Figure 3-30.

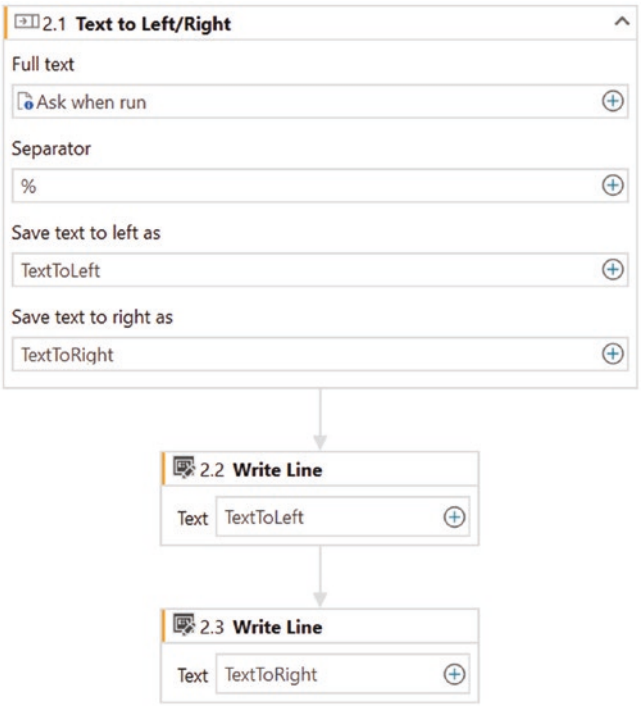

Figure 3-30. *Configuration of the Text to Left/Right activity exercise*

Once you run the automation, it will display an input dialog. Type `New%Hires%2020` as the value, as shown in Figure 3-31, and click OK. Once the activity has completed execution, the output will be displayed in the Output panel, as shown in Figure 3-32. As mentioned in the configuration, only the first occurrence is used as a separator; that is why you still see the second % sign in the right text value.

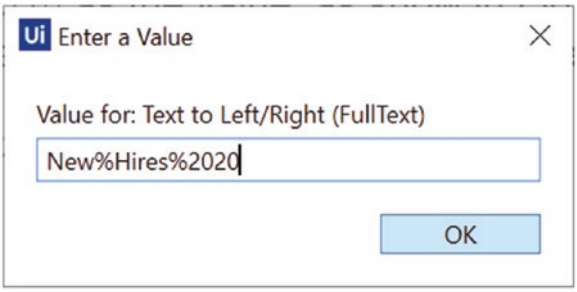

Figure 3-31. *Confirmation message box*

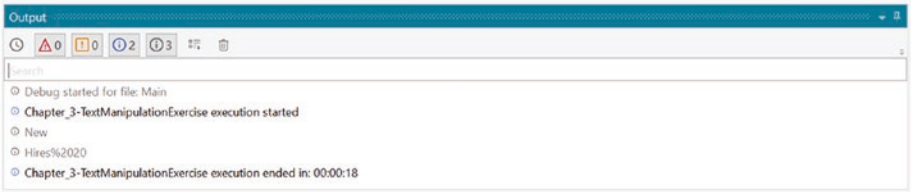

Figure 3-32. *The output of the automation*

Delay

The **Delay** activity allows you to pause your automation for a specified number of seconds.

Tip At times, issues such as slow system performance and latency in the network are unavoidable. The Delay activity can be useful in ensuring that such issues do not cause automation failures. This activity will help make your automation more robust and less prone to failure.

Configuration

This section provides instructions on how to configure a **Delay** activity, shown in Figure 3-33.

Figure 3-33. *Activity card for Delay*

Duration: This is a required configuration available on the activity card. This configuration, shown in Figure 3-34, allows you to specify the number of seconds that the automation should pause. By default, the value is 5 seconds.

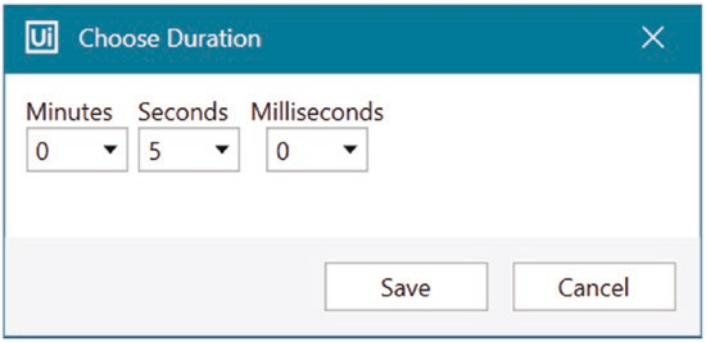

Figure 3-34. *Duration configuration for the Delay activity*

If

The **If** activity allows you to check for a condition and, based on the result, chooses which path the automation should follow.

The automation can either follow the **Then** path or the **Else** path. The automation will follow the **Then** path in cases where the condition is true. Otherwise, the automation will follow the **Else** path.

Note The If activity is also known as a decision activity, condition activity, or if-then-else activity.

Configuration

This section provides instructions on how to configure an **If** activity, shown in Figure 3-35.

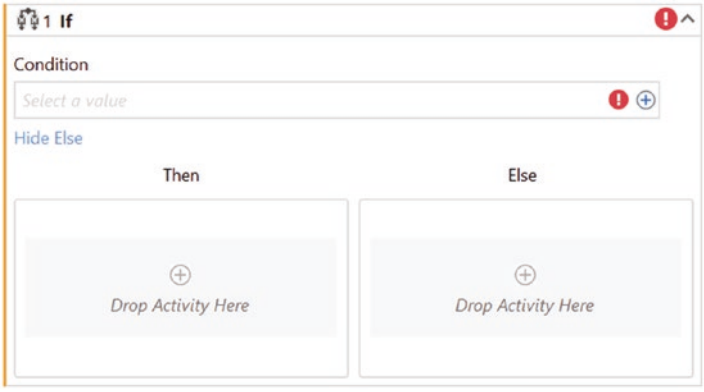

Figure 3-35. *Activity card for If activity*

Condition: This is a required configuration available on the activity card. You can use the `Condition Builder` shown in Figure 3-36 to build a condition that the automation will evaluate.

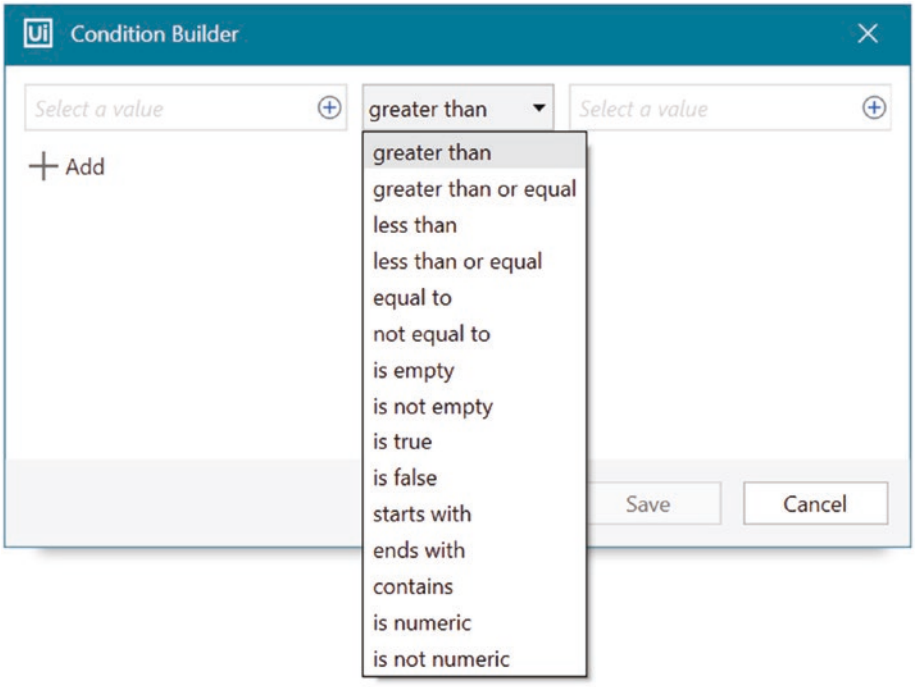

Figure 3-36. *Condition Builder*

You can click the `Add` button to specify more than one condition in the `Condition Builder`. In the case of multiple conditions, using the dropdown on the top left, shown in Figure 3-37, you will also have the option to specify how the automation should evaluate them.

The AND option means that all conditions must be true for the overall condition to be considered true.

The OR option means that any one of the conditions can be true for the overall condition to be considered true.

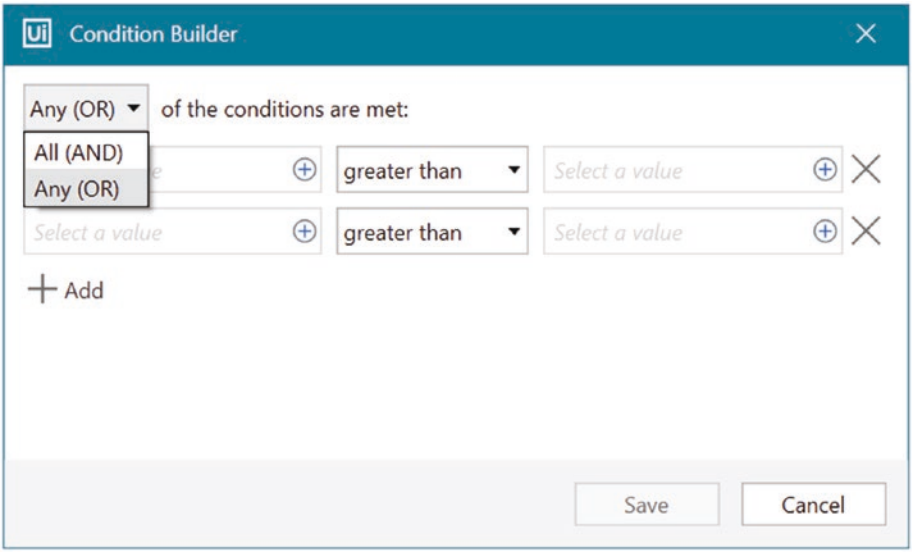

Figure 3-37. *Configuration for multiple conditions*

EXERCISE

Goal: Use the `If` activity to check if the address in project Notebook is a street address or a building address.

Source Code: Chapter_3-IfConditionExercise

Setup: Here are step-by-step implementation instructions:

1. In StudioX, click Notebook ➤ Open Notebook and open the `Scratchpad` worksheet.
2. Next, in cell `A1`, enter `Address`; this is just a label.

3. Next, in cell `B1`, enter a test address, `123 Main Street, Suite # 100, City, State`. Figure 3-38 shows the `Notebook`. Click `Save` and close the `Notebook`.

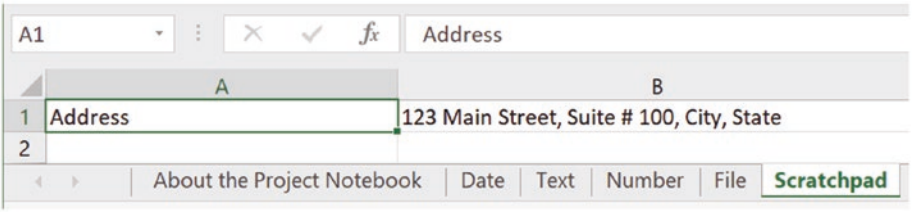

Figure 3-38. *Notebook containing input data*

4. Next, add an `If` activity to the `Designer` panel.
5. Open `Condition Builder` by clicking the `Plus` icon.
6. On the left side of the condition, select `Notebook` ➤ `Indicate in Excel` option, as shown in Figure 3-39.

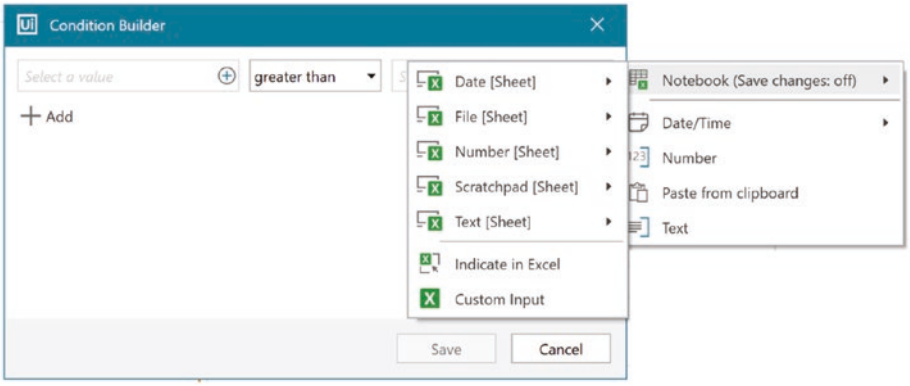

Figure 3-39. *Select input data from the Notebook*

7. Select cell `B1` and click `Confirm` from the ribbon in Excel.
8. From the condition dropdown, select `contains`.

9. On the right side of the condition, click the `Text` option, and enter `Apt #`. This condition will check if the provided address contains Apt #.
10. Click `+ Add` to add another condition.
11. On the left side of the condition, select `Project Notebook ➤ Indicate in Excel` option. Select cell `B1` and click `Confirm` from the ribbon in Excel.
12. From the condition dropdown, select `contains`.
13. On the right side of the condition, click the `Text` option, and enter `Suite #`. This condition will check if the provided address contains Suite #.
14. Select `Any (OR)` from the top-left dropdown. At this point, the `Condition Builder` should look like Figure 3-40. Click `Save`.

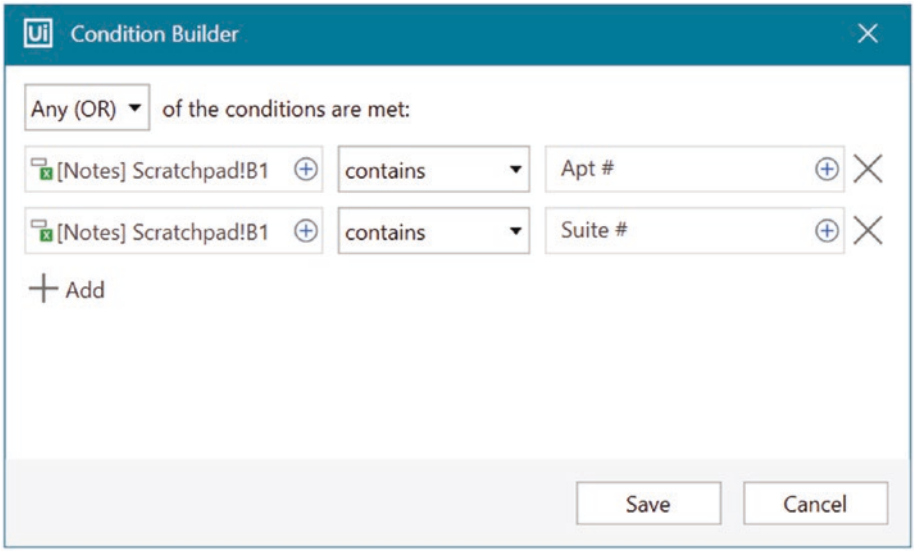

Figure 3-40. *Condition configuration*

15. Next, add a `Write Line` activity in the Then body of `If` activity.
16. Click the `Plus` icon, select the `Text` option, and enter `This is a building address`.
17. Next, add a `Write Line` activity in the `Else` body of `If` activity.
18. Click the `Plus` icon, select the `Text` option, and enter `This is a street address`.

Once you have completed the exercise, the final configuration of the **If** activity should resemble Figure 3-41. Figure 3-42 shows the output of the run.

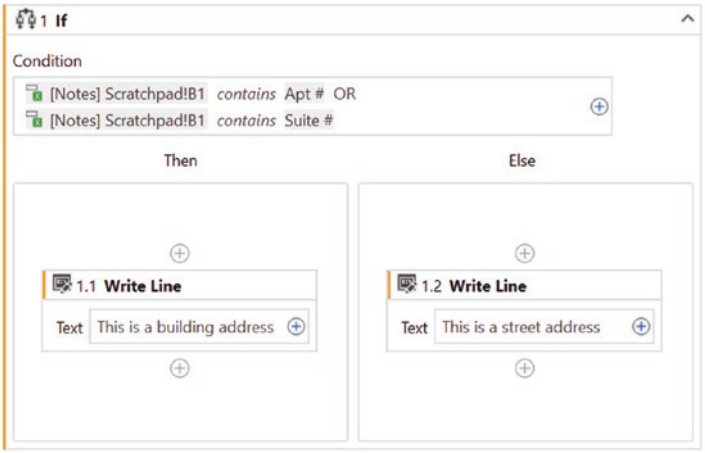

Figure 3-41. *Configuration of If activity exercise*

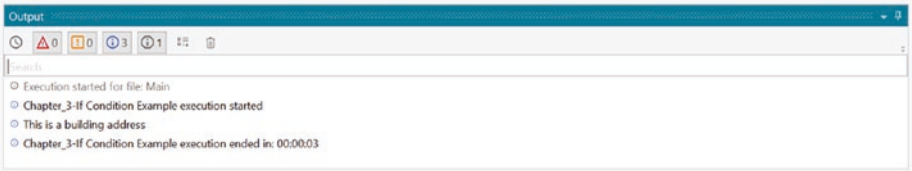

Figure 3-42. *The output of the automation run*

Switch

The **Switch** activity allows you to define multiple cases, and then based on an expression, it executes a single case.

The If activity and Switch activity are conditional activities, that is, based on the expression results, they follow one of the defined paths. You can use an If activity when there are two choices, while a Switch activity is better suited for multiple choices.

Configuration

This section provides instructions on how to configure a **Switch** activity, shown in Figure 3-43.

Figure 3-43. *Activity card for Switch*

TypeArgument: This is an optional configuration available on the `Properties` panel. From the dropdown shown in Figure 3-44, you need to select the type of value that your expression will return.

If your expression result is true or false, then you will select `Boolean`. If your expression result is a number, that is, 1, 2, 3, and so on, then you will select `Int32`. If your expression result is text, that is, January, February, March, and so on, then you will select `String`.

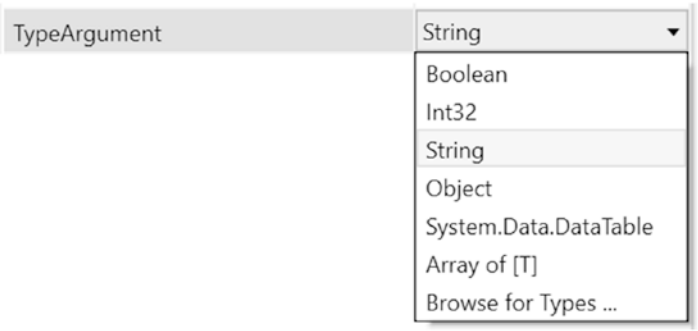

Figure 3-44. *Argument type options*

Expression: This is a required configuration available on the activity card. This expression will return one of the values defined as a case. For example, this could point to a column in Excel that has status values, and based on the status value, you want to perform specific actions.

Add new case: This is a required configuration available on the activity card, that is, a Default case must be provided. This configuration allows you to add a new Case value to the Switch activity, as shown in Figure 3-45.

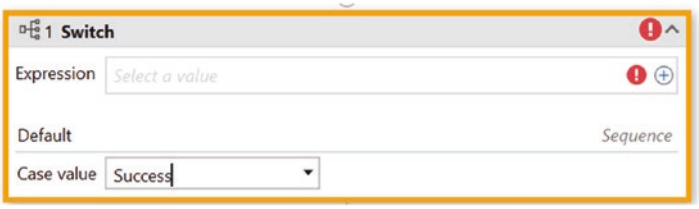

Figure 3-45. *Add new case in Switch activity*

Once you have added a new `Case` value, you can add activities to the body of this `Case` value, as shown in Figure 3-46. During execution, if the value returned by the expression matches this case, then the activities in the body of this case will be executed.

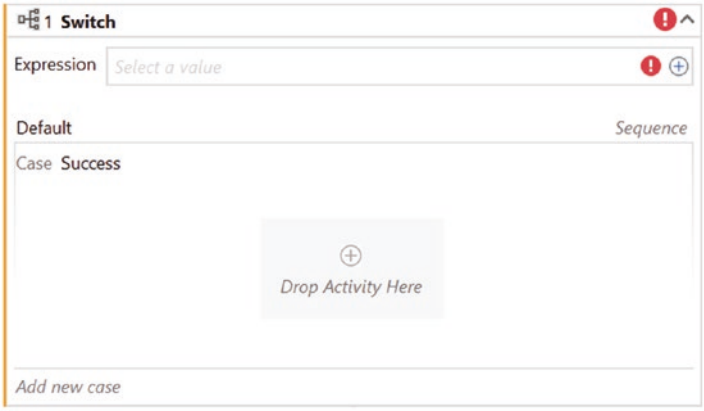

Figure 3-46. *Switch activity with a case*

EXERCISE

Goal: Use the `Switch` activity to execute a different set of activities depending on the status of data.

Source Code: Chapter_3_SwitchRepeatActivitiesExercise

Setup: Here are step-by-step implementation instructions:

1. In `StudioX`, add the `Switch` activity to a blank process.
2. From the `Properties` panel of `Switch` activity, under `Misc` section, update `TypeArgument` to `String`.
3. Select `Ask when run` from `Expression`.

4. Next, click `Add new case` link and enter the value of the case as `In Progress`.
5. Add a `Write Line` activity to the body of this new case and set `Text` to `Executing In Progress case`.
6. Next, click `Add new case` link and enter the value of the case as `Success`.
7. Add a `Write Line` activity to the body of this new case and set `Text` to `Executing Success case`.
8. Next, click `Add new case` link and enter the value of the case as `Failure`.
9. Add a `Write Line` activity to the body of this new case and set `Text` to `Executing Failure case`.
10. Next, add a `Write Line` activity to the body of `Default` case and set `Text` to `Executing Default case`.

Once you have completed the exercise, the final configuration of the **Switch** activity should resemble Figure 3-47. Once you execute the activities shown in Figure 3-48, the automation will prompt you to either select or enter a case value. Based on your selection, the automation will print the value to the Output panel, as shown in Figure 3-49.

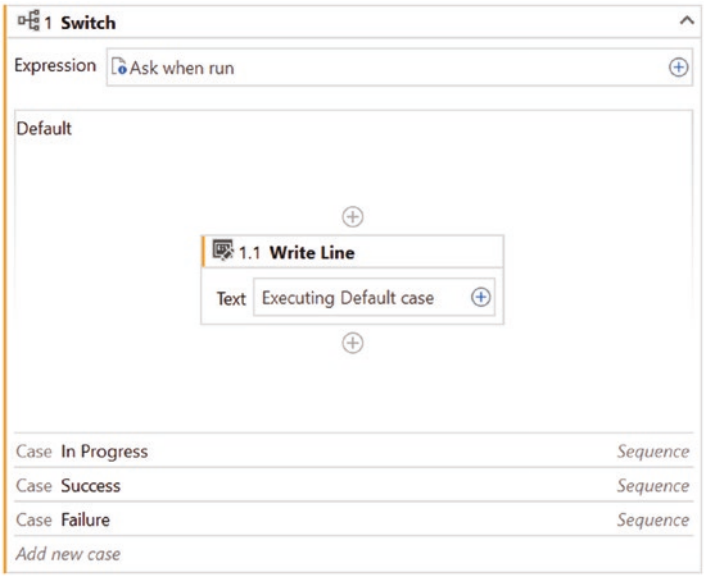

Figure 3-47. *Configuration of Switch activity exercise*

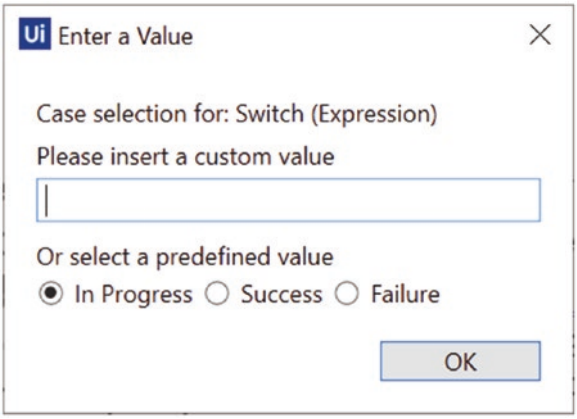

Figure 3-48. *Runtime prompt for the value*

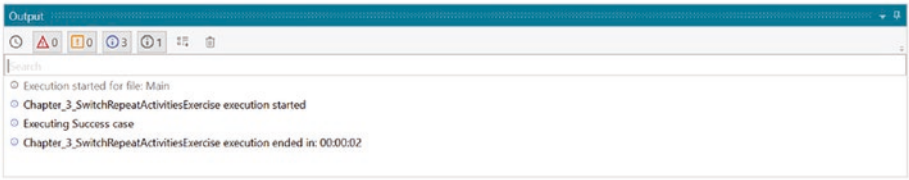

Figure 3-49. *The output of the automation run*

Repeat Number Of Times

The **Repeat Number Of Times** activity allows you to repeat one or more activities a specified number of times.

Tip Typically, this activity will be used in conjunction with the If, Skip Current, and Exit Loop activities.

Configuration

This section provides instructions on how to configure a **Repeat Number Of Times** activity, shown in Figure 3-50.

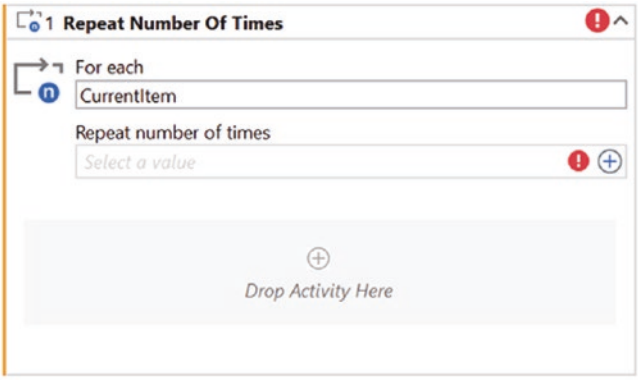

Figure 3-50. *Activity card for Repeat Number Of Times*

Repeat number of times: This is a required configuration available on the activity card. This configuration allows you to specify the number of times this activity should repeat. Commonly, this configuration is specified through the Number builder option.

For each: This is a required configuration available on the activity card. As the activity repeats, the index will keep increasing; for example, when the activity is repeating the second time, the index will be 2. This configuration allows you to name the index so that you can reference it in the body of the activity. You can use this index to decide if you need to skip a specific iteration or simply stop the iterations before reaching the `Repeat Number Of Times` value. By default, the text is set to `CurrentItem` and can be updated to suit your automation; for example, you can rename `CurrentItem` to Vendor or Employee.

EXERCISE

Goal: Use the `Repeat Number Of Times` activity to repeat the Switch activity created in the previous exercise three times. This exercise builds upon the previous exercise for `Switch` activity.

Source Code: Chapter_3_SwitchRepeatActivitiesExercise

Setup: Here are step-by-step implementation instructions:

1. In `StudioX`, add the `Repeat Number Of Times` activity before the `Switch` activity in the `Designer` panel.
2. In the `Repeat number of times` field, click the `Plus` icon, select the `Number` option, and type `3`.
3. Move the `Switch` activity (from the previous exercise) to the body of `Repeat Number Of Times` activity.

Once you have completed the exercise, the final configuration of the **Repeat Number Of Times** activity should resemble Figure 3-51.

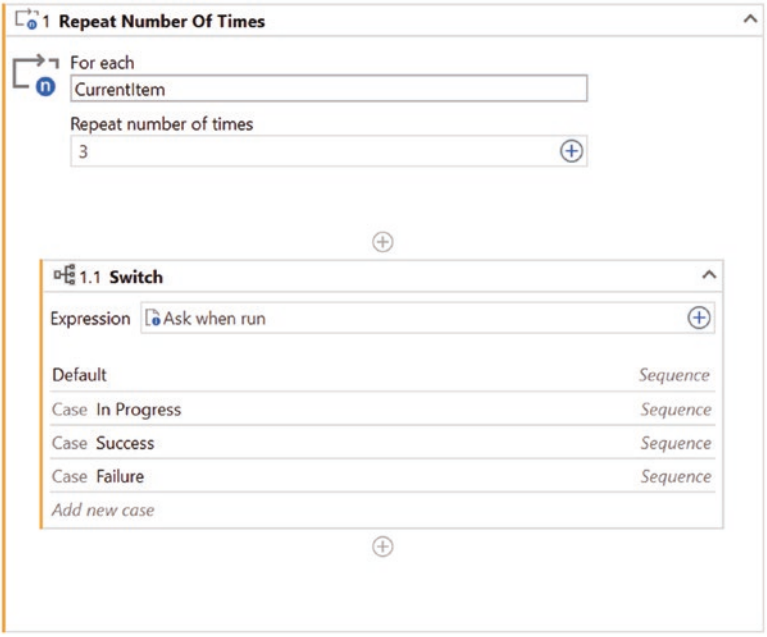

Figure 3-51. *Configuration of Repeat Number Of Times activity exercise*

Once you execute the activities shown in Figure 3-51, the automation will prompt you three times to either select or enter a case value, shown in Figure 3-48. Based on your selection, the automation will print the value to the Output panel, as shown in Figure 3-52.

Figure 3-52. *The output from the automation run*

Skip Current

The **Skip Current** activity allows you to skip the current iteration in a repeating activity.

This activity is useful when you are iterating through a list of items using a repeating activity. Based on certain conditions, you may only want to process some items while ignoring the rest; then, you can use the `Skip Current` activity to skip the current iteration.

Configuration

The **Skip Current** activity, shown in Figure 3-53, does not have any configurations. The only requirement for this activity is that it must be used inside a repeating activity such as the **Repeat Number Of Times** activity or the **For Each** activity.

Figure 3-53. *Activity card for Skip Current*

EXERCISE

Goal: Use the `Skip Current` activity to skip the current iteration if the status is In Progress. This exercise builds upon the previous exercises for `Switch` and `Repeat Number Of Times` activities.

Source Code: Chapter_3_SwitchRepeatActivitiesExercise

Setup: Here are step-by-step implementation instructions:

1. In StudioX, add the `Skip Current` activity to the body `In Progress` case before the `Write Line` activity. This `Write Line` activity will not execute.

Once you have completed the exercise, the final configuration of the **Skip Current** activity should resemble Figure 3-54.

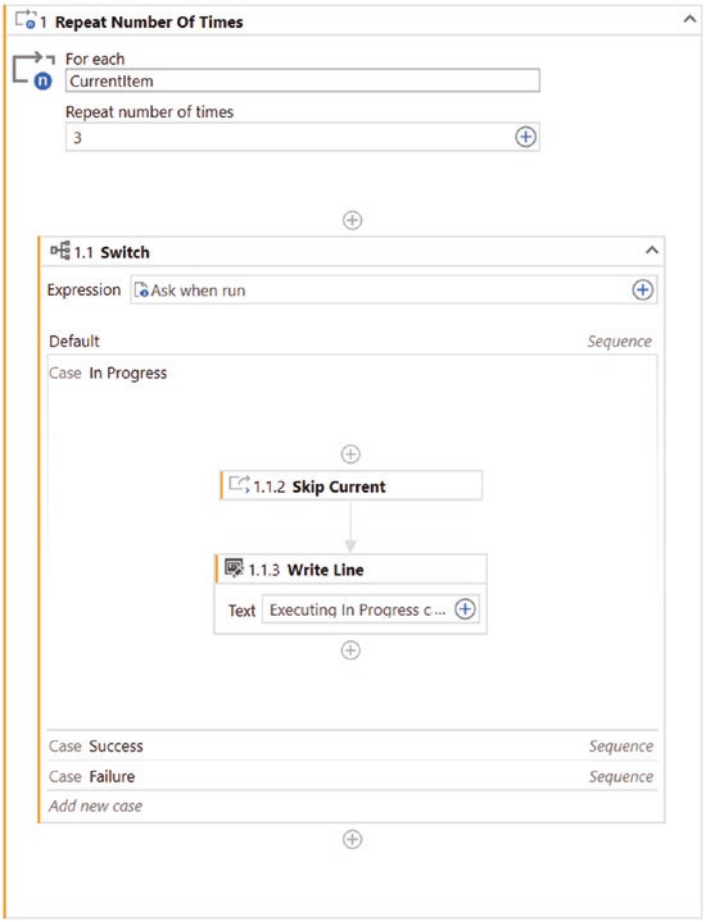

Figure 3-54. *Configuration of Skip Current activity exercise*

Once you execute the activities shown in Figure 3-54, the automation will prompt you three times to either select or enter a case value, shown in Figure 3-48. Based on your selection, automation will print the values to

the Output panel, as shown in Figure 3-55. If you select `Success`, then `In Progress`, and then `Failure`, you will see that `In Progress` was skipped and hence was not printed in the Output panel.

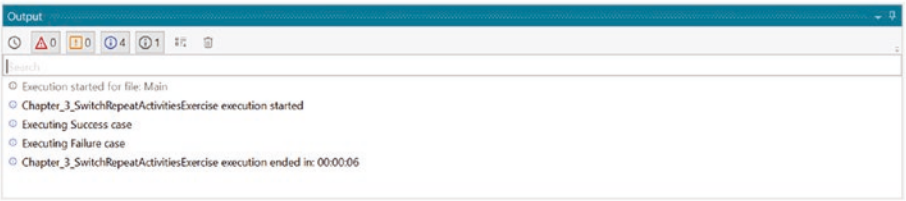

Figure 3-55. *The output of the automation run*

Exit Loop

The **Exit Loop** activity allows you to terminate the repeating activity.

This activity is useful when you are iterating through a list of items using a repeating activity. Based on certain conditions, if you want to terminate the repeating activity, then you can use `Exit Loop` activity.

Configuration

The **Exit Loop** activity, shown in Figure 3-56, does not have any configurations. The only requirement for this activity is that it must be used inside a repeating activity such as the **Repeat Number Of Times** activity or the **For Each** activity.

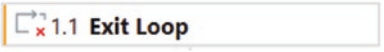

Figure 3-56. *Activity card for Exit Loop*

EXERCISE

Goal: Use the `Exit Loop` activity to terminate the loop as soon as the automation finds an item with a Failure status. This exercise builds upon the previous exercises for the `Switch`, `Repeat Number Of Times`, and `Skip Current` activities.

Source Code: Chapter_3_SwitchRepeatActivitiesExercise

Setup: Here are step-by-step implementation instructions:

1. In `StudioX`, add the `Exit Loop` activity to the body `Failure` case before the `Write Line` activity. This `Write Line` activity will not execute.

Once you have completed the exercise, the final configuration of the **Exit Loop** activity should resemble Figure 3-57.

Once you execute the activities shown in Figure 3-57, the automation will prompt you three times to either select or enter a case value, shown in Figure 3-48. Based on your selection, automation will print the values to the `Output` panel, as shown in Figure 3-58. If you select `Success` and then `Failure`, you will notice that the automation only printed one line to the `Output` panel, and you were never prompted for the third run.

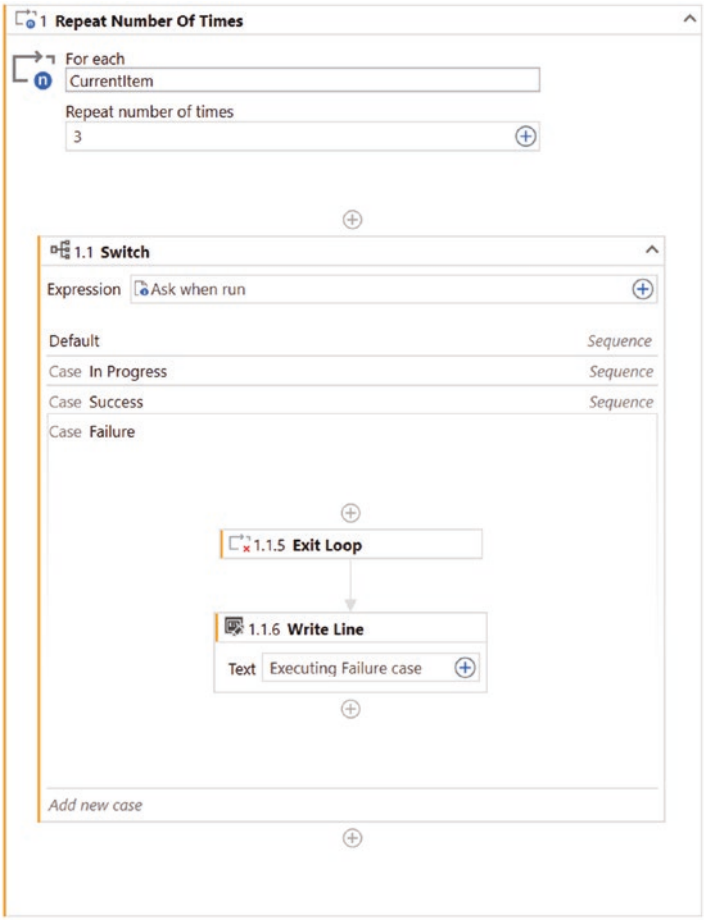

Figure 3-57. *Configuration of Exit Loop activity exercise*

Figure 3-58. *The output of the automation run*

Get Username/Password

The **Get Username/Password** activity allows you to retrieve credentials for a specific application from the Windows Credentials Manager.

Note To log in to an application, utilize the Get Username/Password activity in conjunction with the Type Into activity to type in the Username and Password saved from this activity execution.

Configuration

This section provides instructions on how to configure a **Get Username/Password** activity, shown in Figure 3-59.

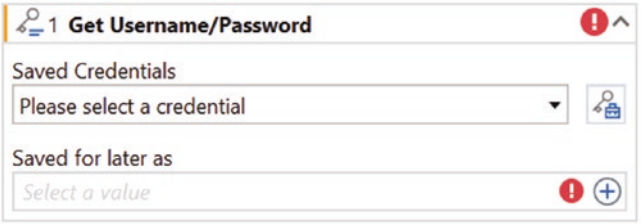

Figure 3-59. *Activity card for Get Username/Password*

Saved Credentials: This is a required configuration available on the activity card. This configuration allows you to select existing credentials or, if required, add new credentials.

To add credentials, click the Key icon, and it will open the `UiPath Credential Manager`, shown in Figure 3-60.

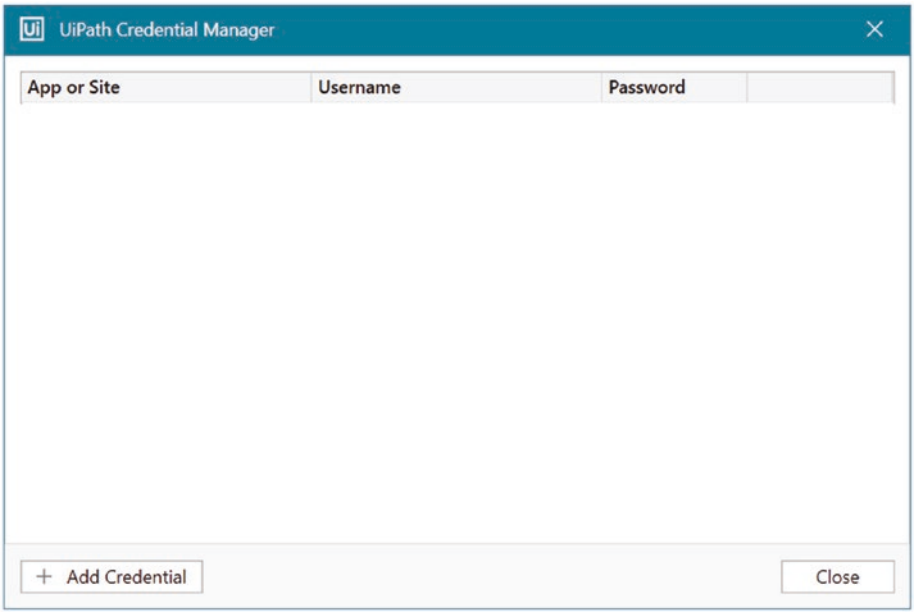

Figure 3-60. *UiPath Credential Manager*

Next, click `+ Add Credential`, and enter credentials as shown in Figure 3-61.

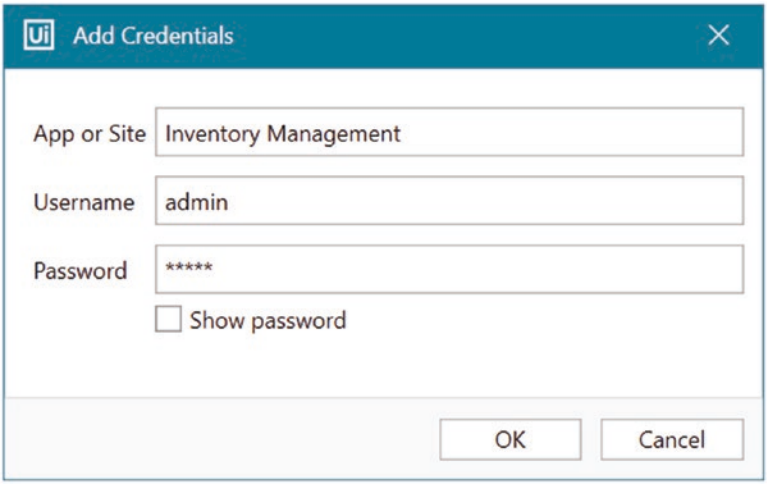

Figure 3-61. *Add Credentials dialog*

Once you have added new credentials in `UiPath Credential Manager`, they also get added to the `Windows Credentials Manager`, as shown in Figure 3-62.

Moreover, you have the option to add credentials directly in the `Windows Credentials Manager` and access them in UiPath StudioX; just make sure to prefix the name with `UiPath`.

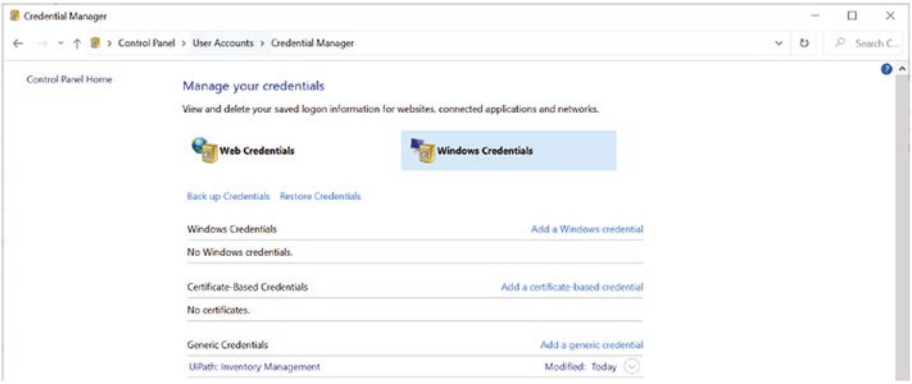

Figure 3-62. *Windows Credentials Manager*

Saved for later as: This is a required configuration available on the activity card. This configuration allows you to store credentials in a value for later use in the automation. You can later access both the `Username` and `Password` from the saved value, as demonstrated in the inventory management example in Figure 3-63.

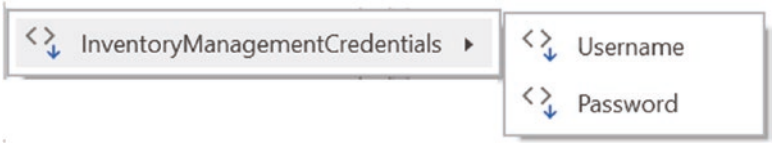

Figure 3-63. *Accessing username and password example*

Get Orchestrator Asset

The **Get Orchestrator Asset** activity allows you to retrieve credentials and other assets from Orchestrator for use in automation.

Tip For consistency and reusability, values that are shared across multiple automations should be stored in Orchestrator.

Configuration

This section provides instructions on how to configure a **Get Orchestrator Asset** activity, shown in Figure 3-64.

Figure 3-64. *Activity card for Get Orchestrator Asset*

Asset name: This is a required configuration available on the activity card. This configuration allows you to specify the asset that you want to retrieve from Orchestrator.

Orchestrator folder path: This is a required configuration available in the `Properties` panel. This configuration allows you to specify the folder path of the asset in Orchestrator.

Save for later as: This is an optional configuration available on the activity card. This configuration allows you to save the asset returned from Orchestrator for later use in automation. You can later reference the asset from the Use Saved Value menu.

EXERCISE

Goal: Use the `Get Orchestrator Asset` activity to retrieve a `DocumentPath` asset from Orchestrator, shown in Figure 3-65.

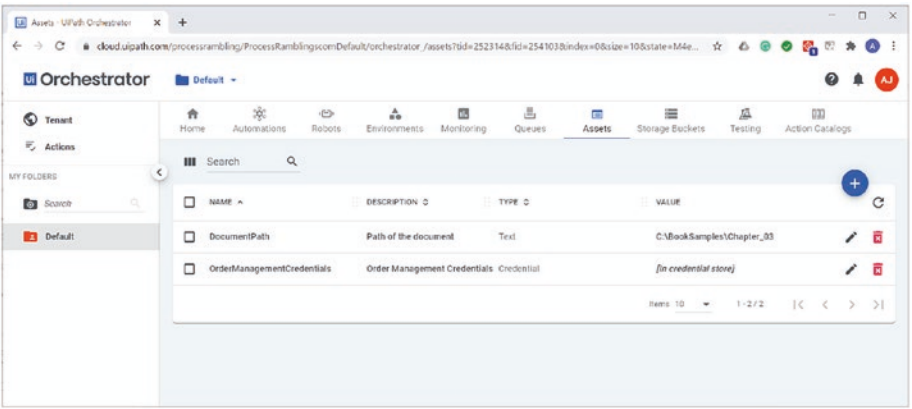

Figure 3-65. *DocumentPath asset in Orchestrator*

Source Code: Chapter_3-GetOrchestratorAssetExercise

Setup: Here are step-by-step implementation instructions:

1. In `StudioX`, add the `Get Orchestrator Asset` activity to a blank process.
2. In the `Asset name` field, click the `Plus` icon, select the `Text` option, and type `DocumentPath`.
3. In the `Orchestrator folder path` field in `Properties` panel, click the `Plus` icon, select the `Text` option, and type `/`. This specifies that we are looking for the asset in the parent folder.
4. Click the `Save for later as` field, click the `Plus` icon, and select `Save for Later Use`. Enter `OrchestratorAssetValue` as the value name.

5. Next, add the `Write Line` activity after the `Get Orchestrator Asset` activity.

6. In the `Text` field, click the `Plus` icon, and hover over `Use Saved Value` to select `OrchestratorAssetValue`.

Once you have completed the exercise, the final configuration of the **Get Orchestrator Asset** activity should resemble Figure 3-66.

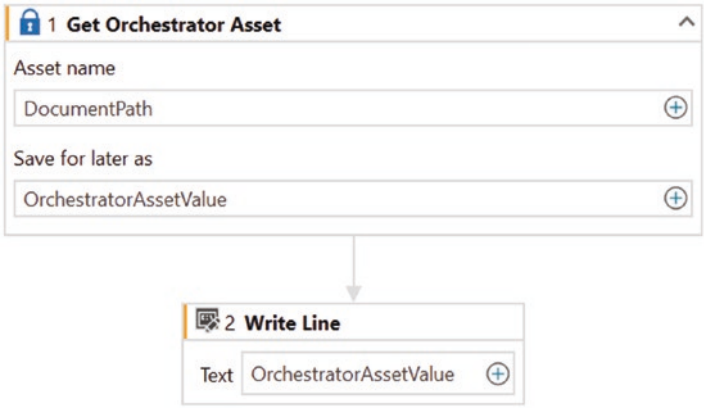

Figure 3-66. *Configuration of Get Orchestrator Asset activity exercise*

Once you execute the activities shown in Figure 3-66, the automation will retrieve the asset from Orchestrator and print the value to the Output panel, as shown in Figure 3-67.

Figure 3-67. *The output of the automation run*

Save For Later

The **Save For Later** activity allows you to save any value for use later.

Configuration

This section provides instructions on how to configure a **Save For Later** activity, shown in Figure 3-68.

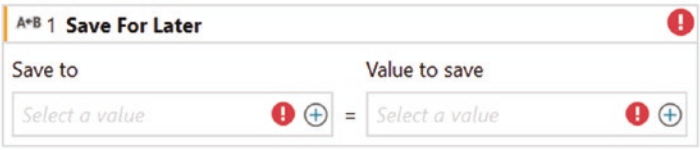

Figure 3-68. *Activity card for Get Username/Password*

Save to: This is a required configuration available on the activity card. This configuration allows you to either select an existing value name or create a new value name, as shown in Figure 3-69. This value name will start showing up in the `Data Manager` panel. To use this value later in an activity, click the `Plus` icon and select it from the `Use Saved Value` menu.

Figure 3-69. *Save to options menu*

Value to save: This is a required configuration available on the activity card. This configuration allows you to specify what value you want to save. Figure 3-70 shows different options available to specify the value.

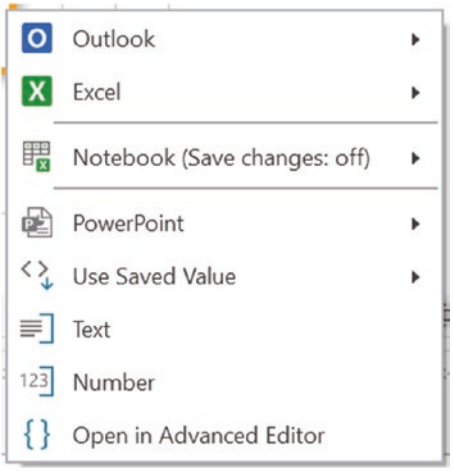

Figure 3-70. *Options to specify value to save*

Wait for Download

The **Wait for Download** activity allows you to monitor and wait for a file download to complete before proceeding.

Configuration

This section provides instructions on how to configure a **Wait for Download** activity, shown in Figure 3-71.

Any activity that is going to interact with the application to start file download must be added to the body of **Wait for Download** activity.

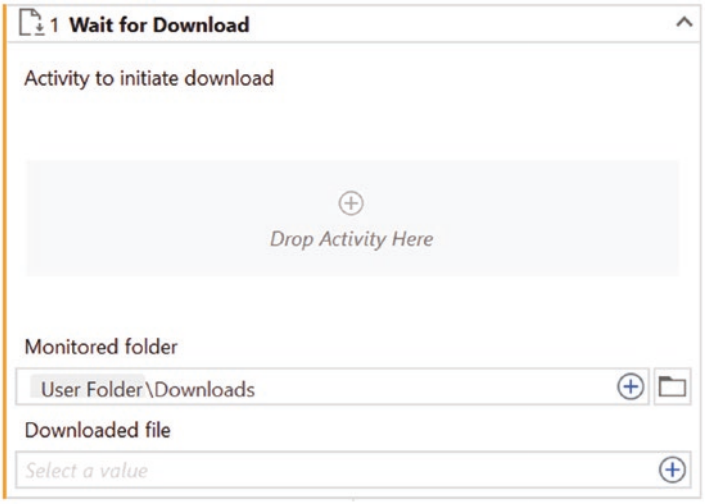

Figure 3-71. *Activity card for Wait for Download*

Monitored folder: This is a required configuration available on the activity card. This configuration allows you to specify a folder that the automation will monitor for the file download. By default, the value is `User Folder\Downloads`, that is, the automation monitors the default `Downloads` folder of your Windows profile or browser.

Downloaded file: This is an optional configuration available on the activity card. This configuration allows you to store downloaded file details for later use. For example, you could use the file path to copy it to another folder for processing.

Timeout: This is an optional configuration available on the `Properties` panel. This configuration allows you to specify how long (in seconds) should the activity wait for file download to complete before proceeding. By default, the timeout value is set to 300. This should be increased for files with larger size.

There is no exercise for this activity in this chapter. We will use the `Wait for Download` activity in the exercise for the `Hover` activity in Chapter 4. Figure 3-72 shows how this activity looks like when configured.

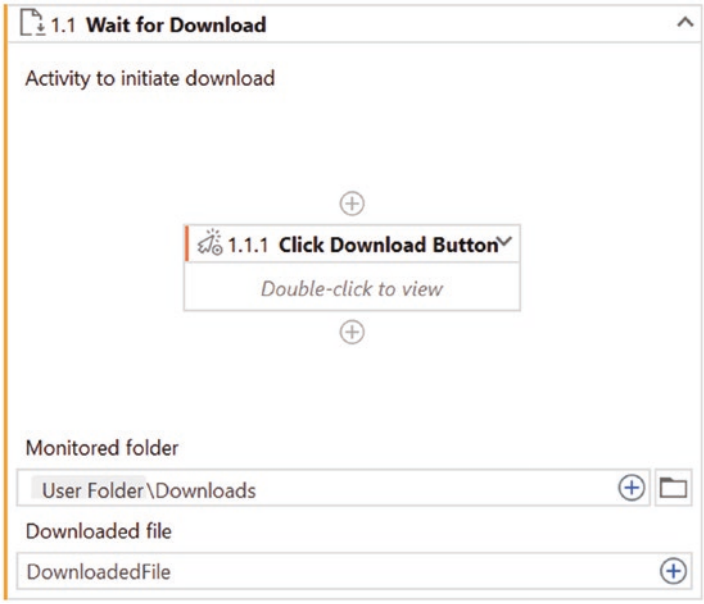

Figure 3-72. *Configuration for Wait for Download activity*

Group

The **Group** activity allows you to logically group a set of activities. All the activities inside the body of the Group are executed in the same sequence. This is just a great way to logically split a large automation project.

Configuration

The **Group** activity, shown in Figure 3-73, does not have any configurations.

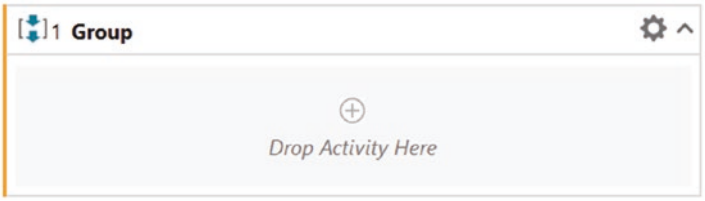

Figure 3-73. *Activity card for Group*