IBM Cloud Pak for Business Automation Demos and Labs 2024

Error Handling in IBM RPA

V 2.0

Pooja Luthra
pooja.luthra@ibm.com
Vinicius Dutra
v.dutra@ibm.com

NOTICES

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

      IBM Director of Licensing
      IBM Corporation
      North Castle Drive, MD-NC119
      Armonk, NY 10504-1785
      United States of America

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

TRADEMARKS

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

© Copyright International Business Machines Corporation 2020.

Table of Contents

1 Introduction

IBM RPA provides a comprehensive set of Robotic Process Automation (RPA) features:

- **Unattended bots**
  Use an RPA-driven digital workforce to automate repetitive tasks without human intervention.
- **Attended bots**
  Remote Desktop Automation (RDA) enables a human workforce to augment work using bots to perform repetitive tasks on demand.
- **Orchestrating Scripts**
  Combine message queues with the orchestrator technology in your IBM RPA Control Center environment to orchestrate scripts.
- **Workflows in IBM RPA**
  Combine BPMN files or create your own workflows in IBM RPA Studio and integrate them into scripts that implement the workflow process in IBM RPA.
- **Optical Character Recognition (OCR)**
  Process documents by extracting structured data from unstructured content.
- **Dashboards**
  Gain business insights into business operations.

With IBM RPA, IBM can provide customers with additional benefits:

- **Faster time to value**
  Speed and simplicity of purchasing and deploying through easier licensing.
- **A comprehensive platform to automate all types of use cases**
  Tighter integrations between RPA and the rest of IBM business automation platform.
- **Automate business and IT processes**
  Expand the IBM business automation mission to IT use cases.
- **Operationalize AI**
  Fulfill IBM's vision of operationalizing AI in every aspect of the business.

You can explore the [Documentation](#) to understand more details about IBM RPA.

2 Process Overview

The objective of this lab exercise is to demonstrate in practice how to use IBM RPA's Error Handling techniques.

The process automated by bots often deals with unexpected behaviour when running in a production environment which can lead to failure of not even the current execution, but future executions as well, disrupting normal flow of events which was automated. **Exception Handling** is configured in a way to cause the bot to raise **errors** as **exceptions** and handle them or try to keep going.
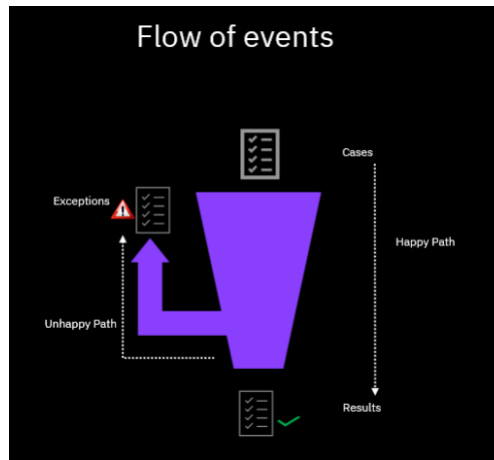
In this lab, we will walk you through the process of applying exception handling along with best practices. We will automate the *sales lead* and *claims submission* process using error handling. The same will be divided into micro scenarios which will be covering the business and system exceptions.

2.1 Importance of Error Handling in RPA

A process without added exception handling can result in a disrupted process and interruption in flow of operations.

1. Disrupted Processes – Errors can cause process to fail causing delay in flow of operations, resulting in delays in delivering products or services.
2. Increased Costs – Disrupted process could result in financial loss or operational costs to rectify them.
3. Inaccurate Data – Errors in data can lead to incorrect assessment of issue causing failure, impacting decision making ability and compliance.
4. Stakeholder reputation - Repeated errors can lead to lack of trust with customers, clients, and stakeholders, therefore damaging your organization's reputation.

2.2 Exceptions

Exceptions are subset of errors that are caught by the bot and handled. Some of the exceptions are linked to systems used while others are linked to logic of business process. It can be classified into following types of exceptions:
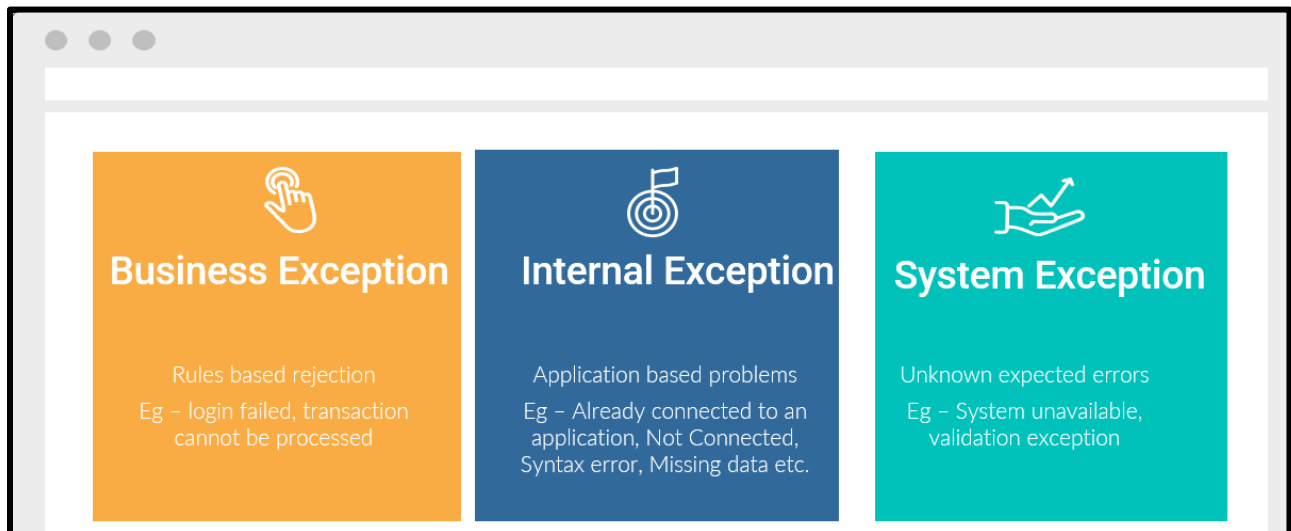
3 Handling Exceptions

RPA developers can create automation that can gracefully recover from an error and save the error details for analysis or to help predict when errors might occur.
Important parameters to consider:
1. Error identification and consideration of type of error.
2. Applying preventive user validation.
3. Exception Handling subroutine to address specific error scenarios.
4. Applying Retrying mechanism.
5. Resolution procedures to handle errors.
6. Error notification
7. Implement Fallback Mechanism

> **Best Practice**:
> - Use appropriate type of log message – Information, Warning or Error to give correct representation of situation.
> - Capture the screen where error is not known to ease evaluation of error.
> - **Email ids** of business SME to be mentioned in **Parameters** so can be managed by users with ease.
> - Let the bot fail rather than provide false information.

4 Pre-requisites

For this lab, you need to reserve an IBM Robotic Process Automation environment from IBM Technology Zone (see chapter 5). All the pre-requisites have been pre-installed/configured in the lab template. The information below is just for information purposes.

IBM Products:

- IBM Robotic Process Automation v23.0.x.

Custom Solutions/Code:

- The important files to run this lab are in C:\CP4AutoDemo\Lab 4 – Exception Handling in IBM RPA

    o ClaimsInput.csv – File containing input data for claims which are raised to be submitted.

    o SalesLeads.csv – File containing input data for sales lead requests to be submitted.

    o HandlingExpectedIgnore.wal – This script covers scenario for one of the type of error handling.

    o Sales-lead-automation-NoExceptionHandling.wal – This script contains scenarios of creating sales lead and submitting claim requests. This will be used as a base script as it does not have any exception handling added.

    o Sales-lead-automation-completed.wal – This script contains all exception handling which was added in each scenario. This can be referred to see completed scenarios with exception handling.

    o For each exercise, a new script will be created. The same are provided at path C:\CP4AutoDemo\Lab 4 – Exception Handling\Scripts\Output Scripts. After trying each exercise, scripts can be verified from ones in output folder.

- A web application containing portal for creating sales lead and claim.

4.1 References

1. [IBM Robotic Process Automation Documentation](#)
2. [IBM Robotic Process Automation Command Documentation](#)
3. [IBM RPA Exception Handling Documentation](#)

5 Accessing the Environment

If you have already reserved a lab environment from IBM Technology Zone, please go to Chapter 5.2 directly.

5.1 Reserve Environment

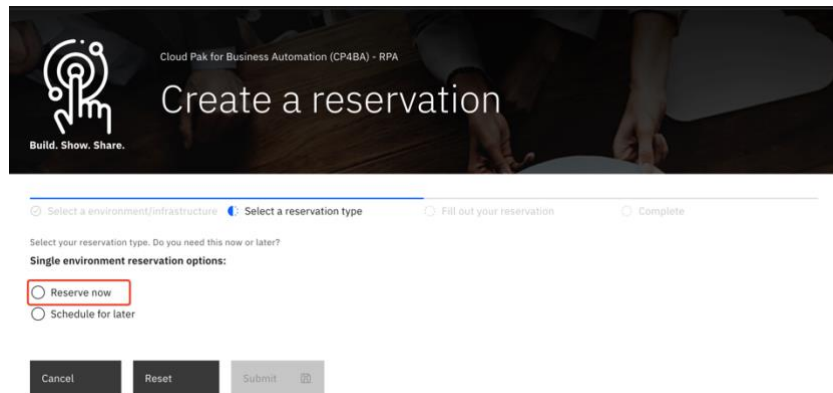To get started with this lab, please follow the below steps to reserve an environment:

1. Click here to open IBM Technology Zone Reservation portal. You need to use your IBMID to login to the portal.
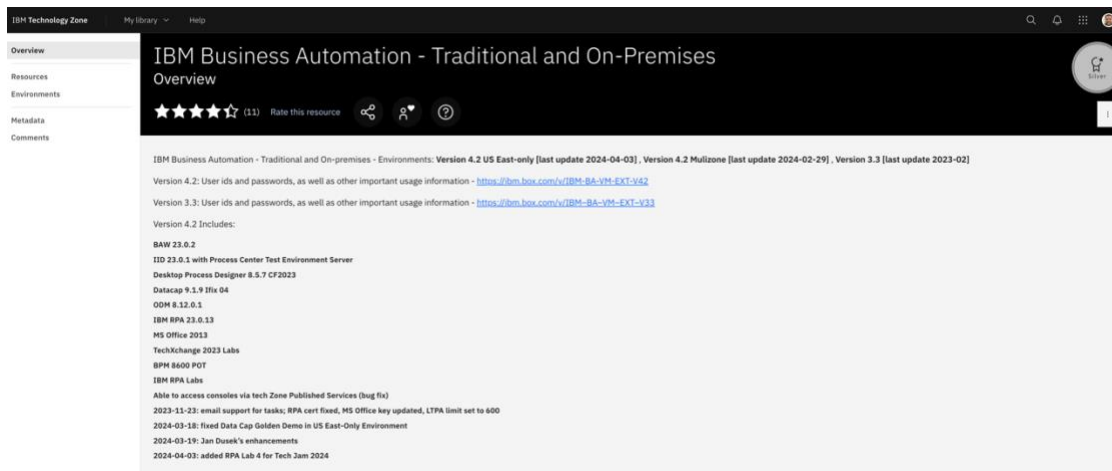
2. Click **Environments** on the left panel, and then reserve the last environment on click the blue button.
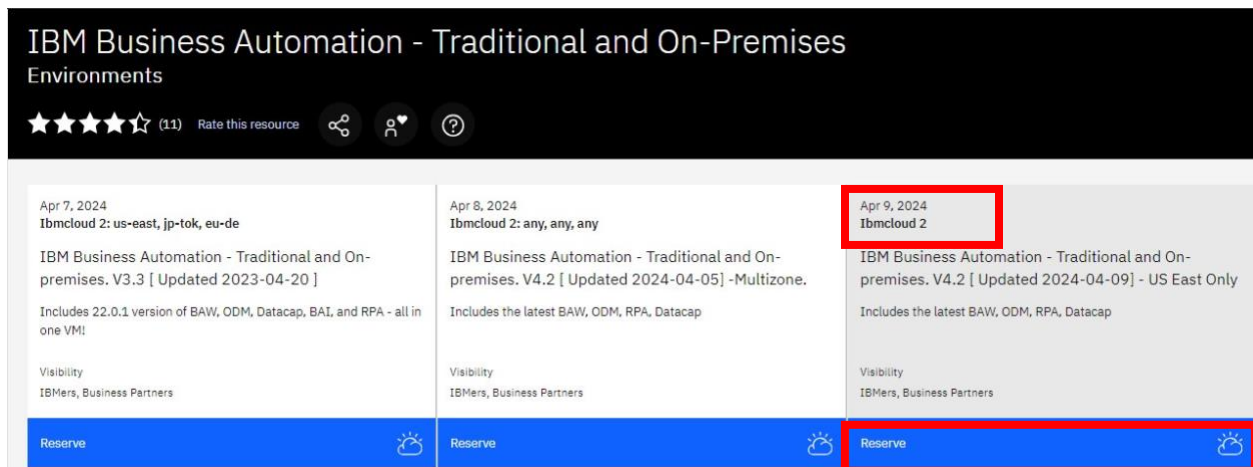
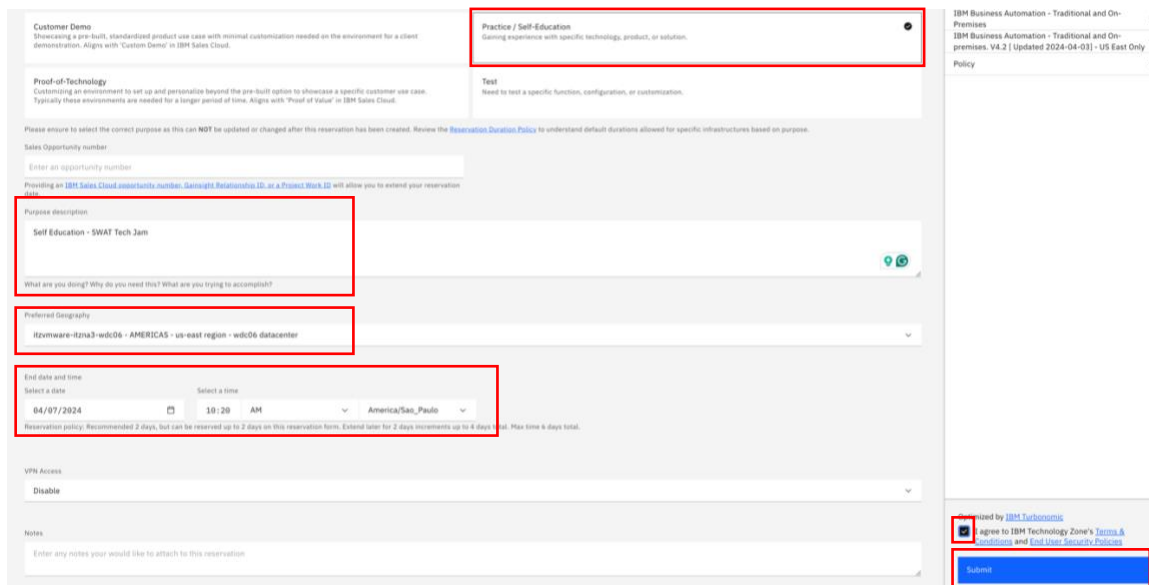3. Select **Reserve for now**, then click **Submit**.

4. On the reservation page, make the appropriate selections as below. Once done, click **Submit**.

    Purpose: Select Practice/Self-Education.

    Purpose description: Enter something like **Self Education**.

    End date and time: Select the end date and time that the environment will be deleted.
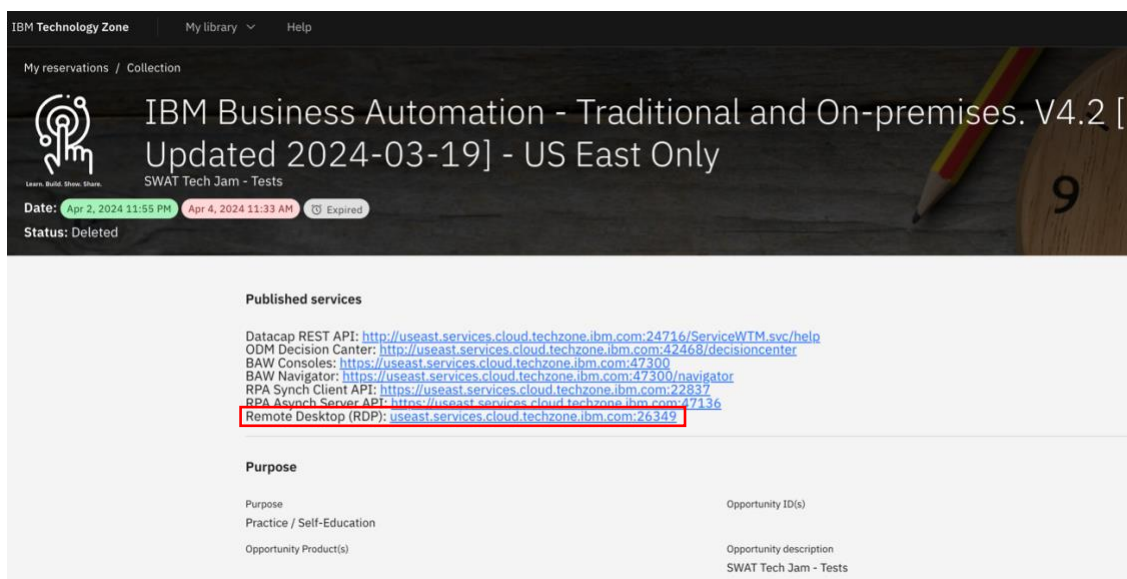
    Preferred Geography: Select the geography where your environment will be created. To get a better network connection, select the same geography as where you are located in.

5. Once you have reserved an environment, you will receive an email with a link to access the environment's management console, click on Reservation ID.

6. You can access the environment using Remote Desktop (RDP) or Remote Console (Web). Our recommendation is to use Remote Console (Web) for practicality. If you prefer to use the RPD, use the Remote Desktop (RDP) link, or keep rolling the page to access the Remote Console (Web).

7. After waiting for the VM to load, open the folder BA TOOLS on the Desktop to access the IBM RPA Studio.
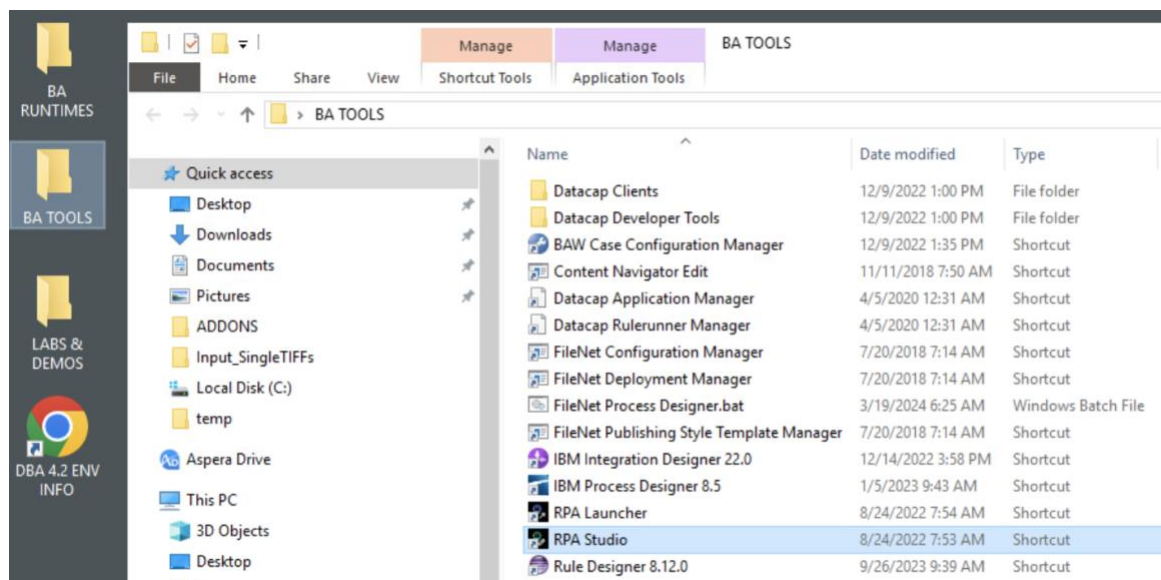
5.2 Accessing Environment

5.2.1 Double click on RPA studio.

5.2.2 Enter the username
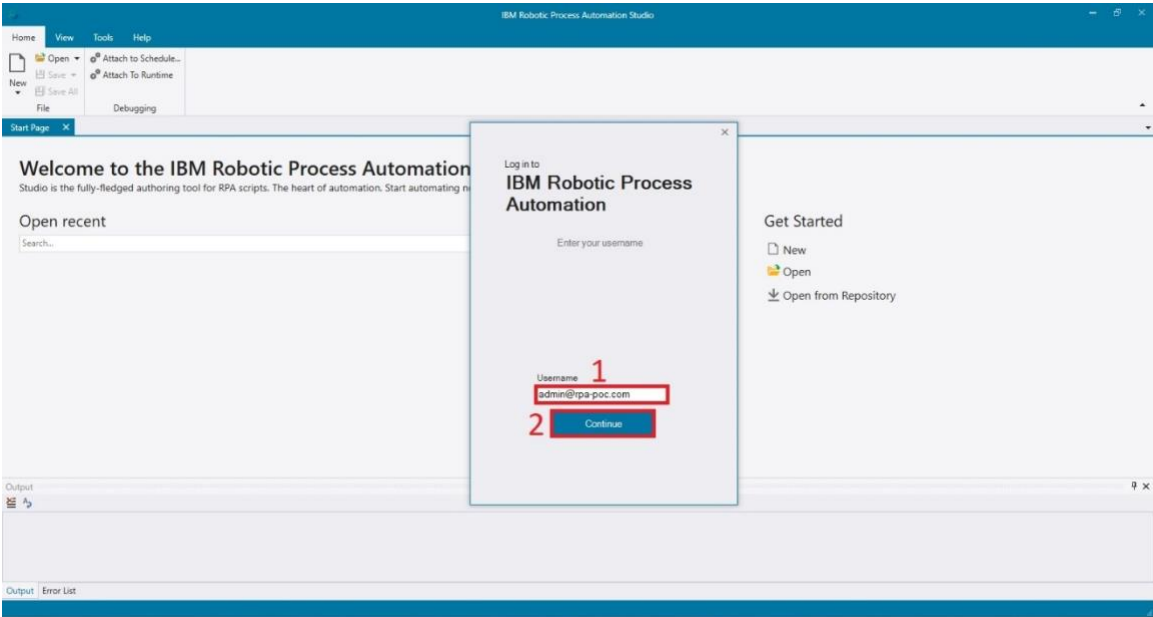
#	Description
1	Write "*admin@rpa-poc.com*" in the **Username** field and press **[enter]**
2	Click the **Continue** button

5.2.3 Select Tenant and enter Password.
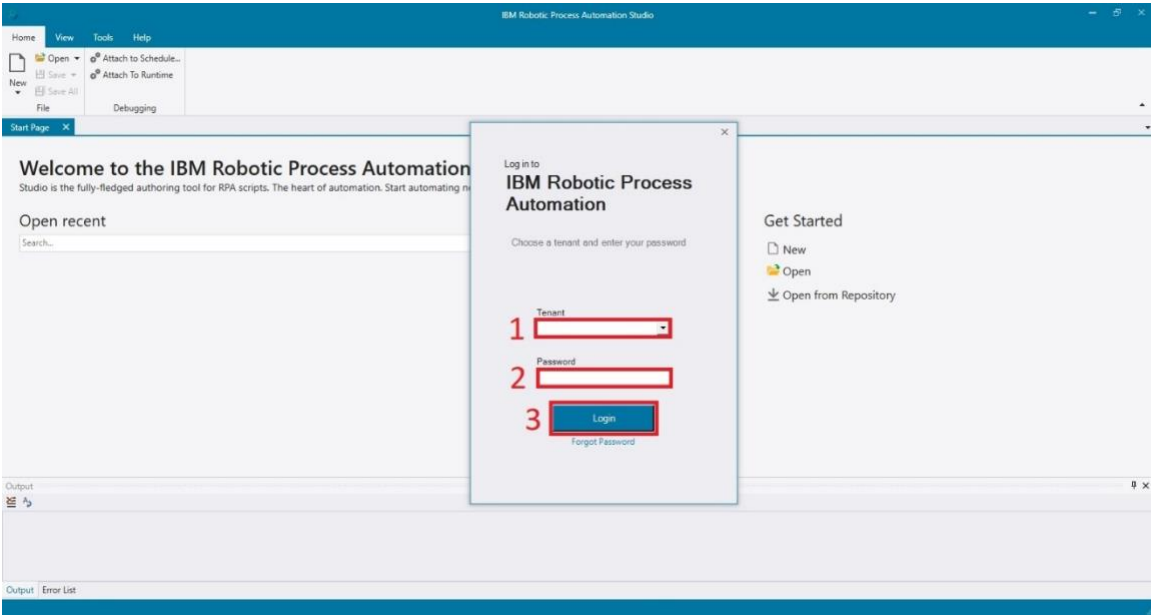
#	Description
1	Select the "*rpa-poc*" as the **Tenant**
2	Enter "*passw0rd*" (use a zero not a capital o)
3	Click on the **Login** button

5.2.4 Open the base script

We will open the base script with no exception handling and work on each type of exception handling using this base script.
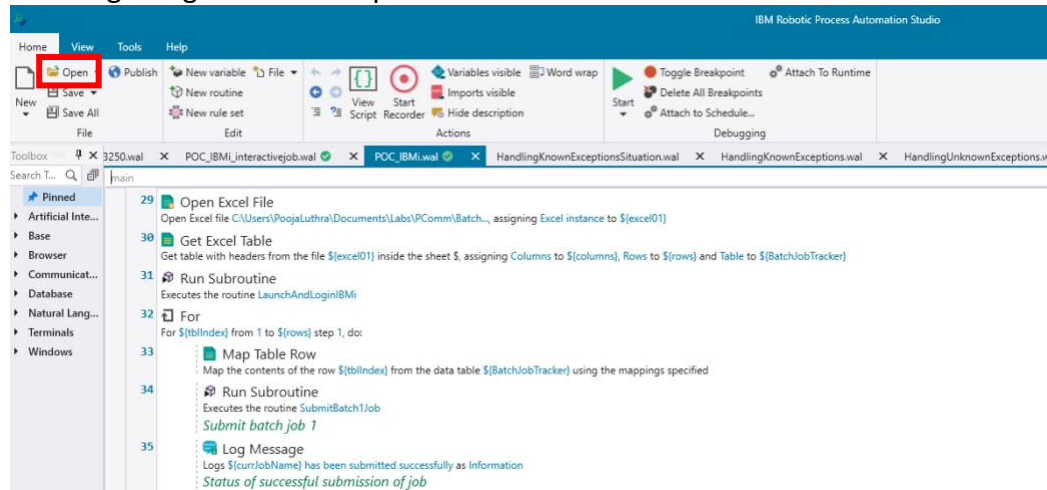
#	Description
1	Click on **Open** menu from **Home tab**.
2	Open the **Script folder** and got to **input folder** and select sales-lead-automation-NoExceptionHandling.wal and click on Open.
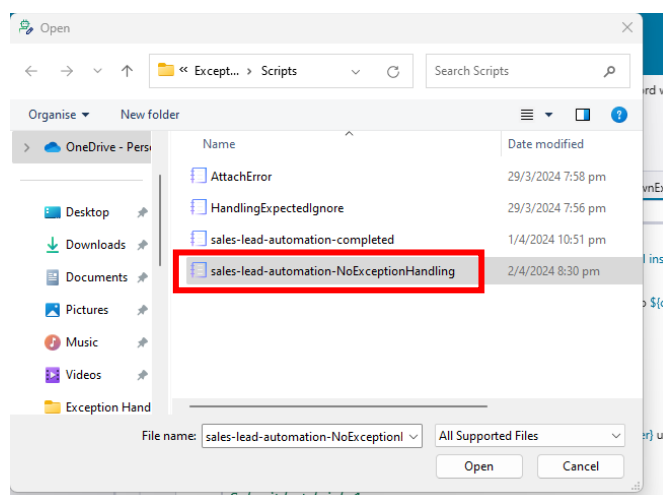
5.3 Setting up environment

5.3.1 Open Firefox and select Control Center under RPA folder.
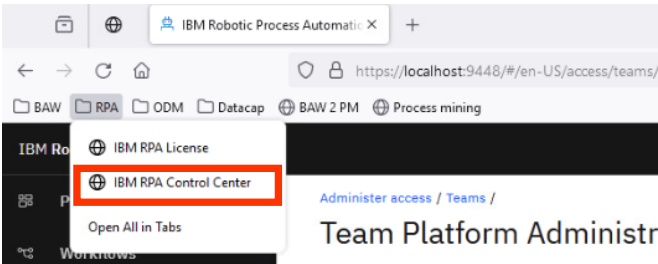
5.3.2 Enter username.

#	Description
1	Enter the "*admin@rpa-poc.com*" as the **Username**
2	Click **Continue**

5.3.3 Enter Tenant and Password
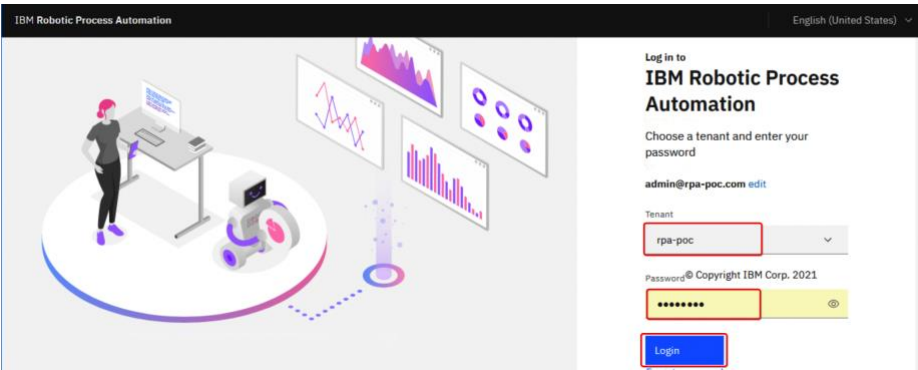
#	Description
1	Enter the "*rpa-poc*" as the **Tenant**
2	Enter "*passw0rd*" (make sure to use a zero not an uppercase o) as the **Password**
3	Click **Login**

5.3.4   Create a team with Vault access
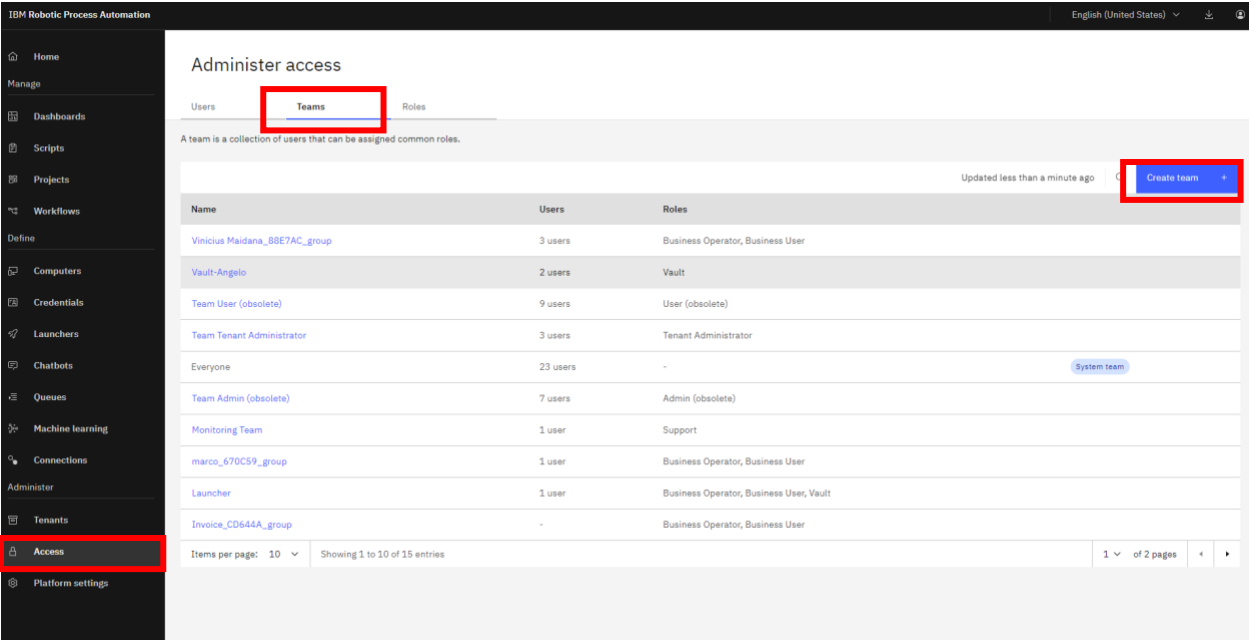
#	Description
1	Click on the **Access menu** from left panel.
2	Select **Teams** from right window.
3	Click on **Create Team.**
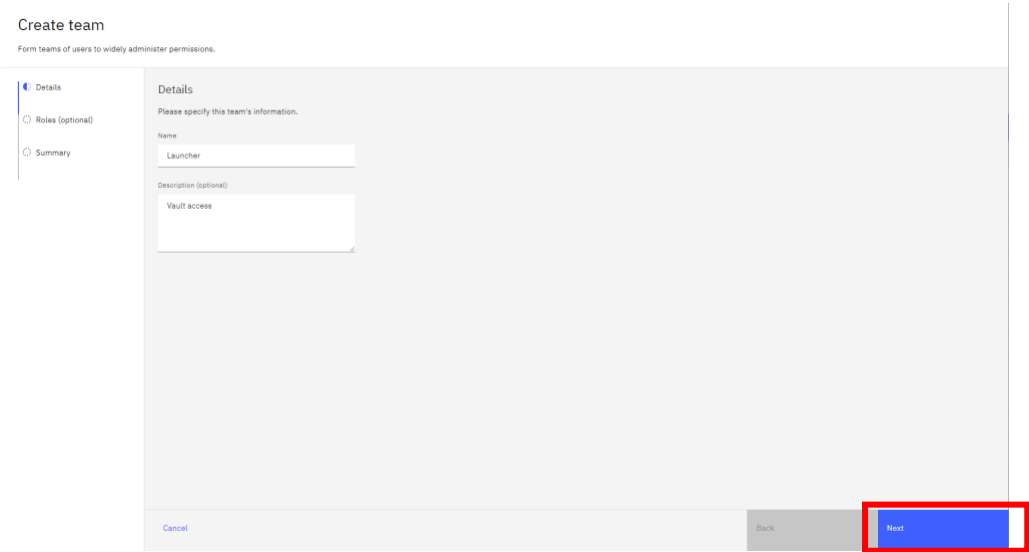
#	Description
1	Add **Name** *'Launcher'*.
2	In **description** add 'Vault access'.
3	Click on *Next button.*

#	Description
1	Window to add Roles will be opened.
2	Tick **all roles**.
3	Click on *Next button.*

#	Description

1	A summary will be loaded. Check the roles and other details and click on *Create*.
2	Team will be created.
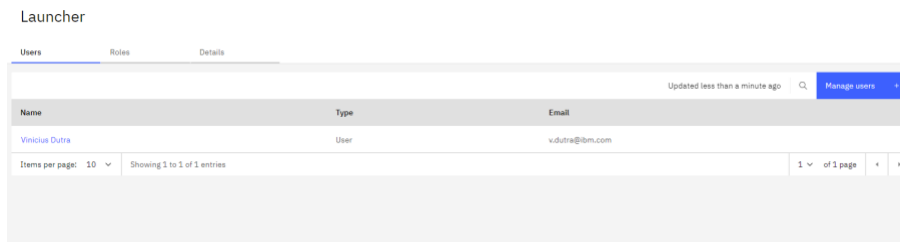
#	Description
1	Open **Launcher Team** and click on ***Manage users.***
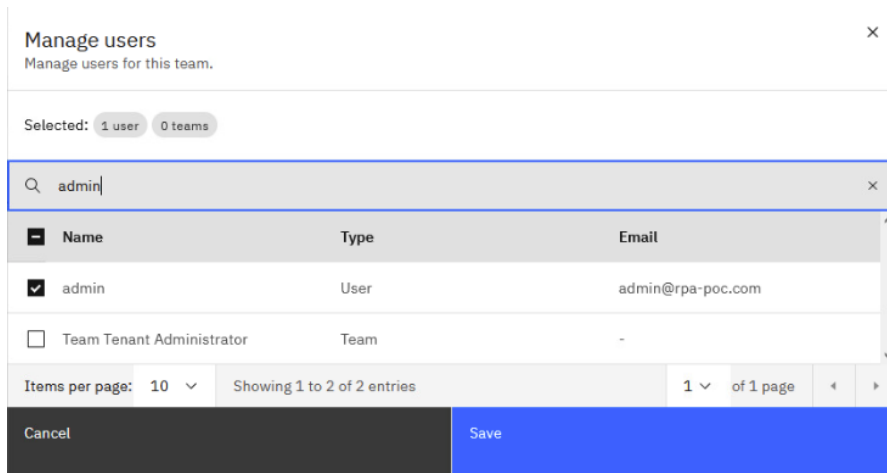
#	Description
1	Search the user by name of admin. Tick the one with admin name confirming email as admin@rpa-poc.com and click on Save.

5.3.5   Go to Vault credentials
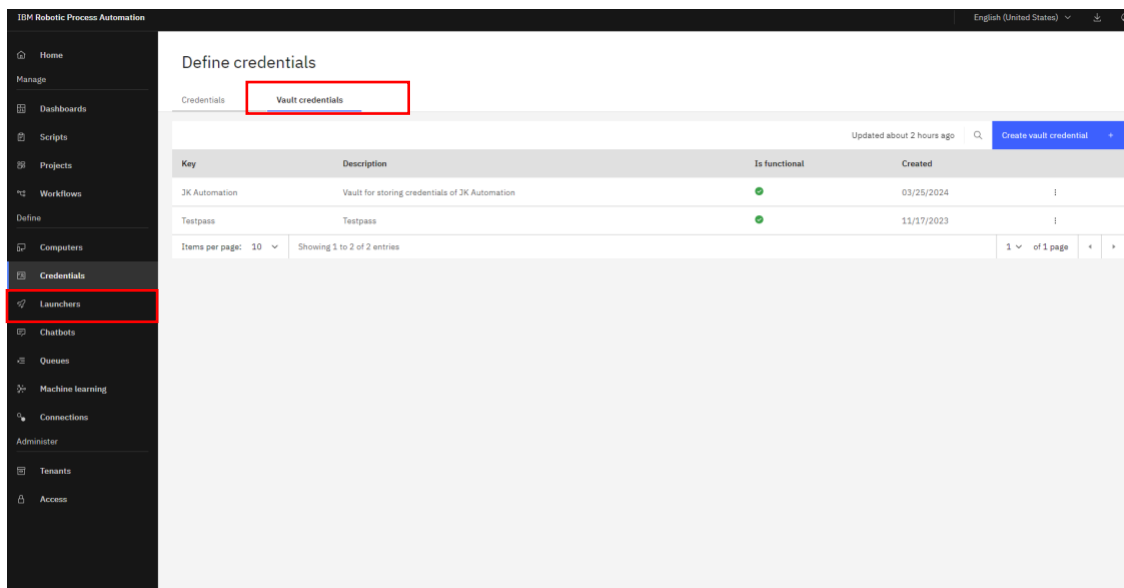
#	Description
1	Click on the **Credentials menu** from left panel.
2	Select **Vault Credentials** from right window.

#	Description
1	Click on **Create vault credential** and add the Key name '*JK Automation*' with description of '*Vault for storing credentials of JK Automation*'.
2	Associate a team to be assigned to this vault.
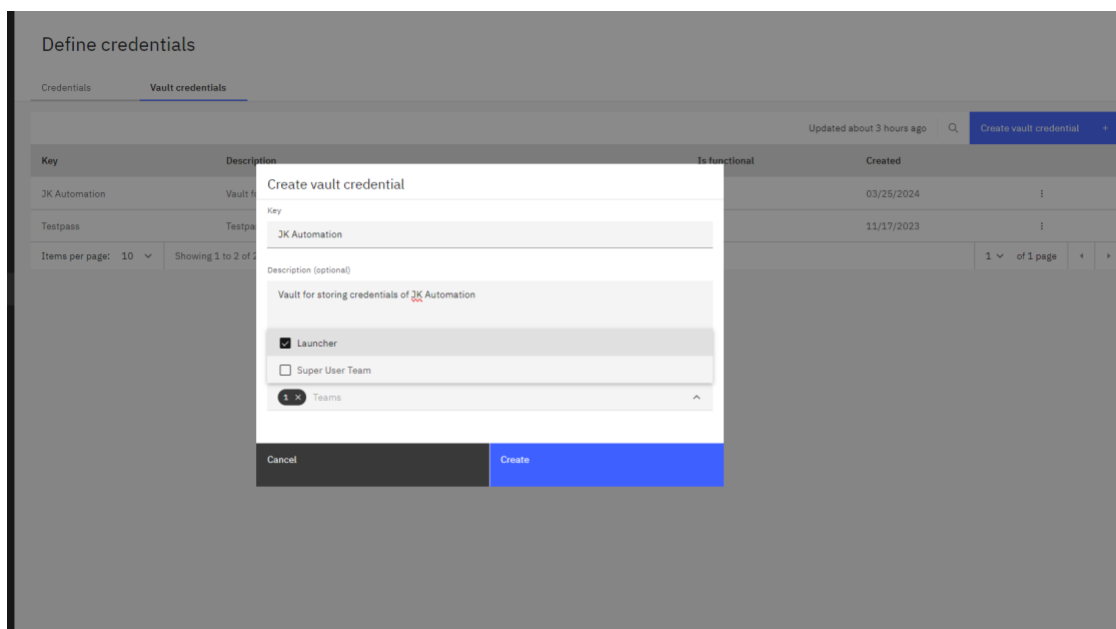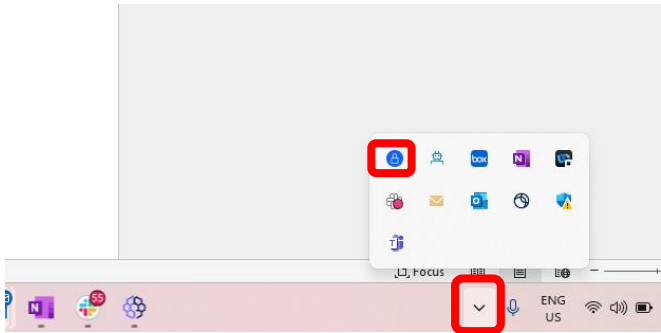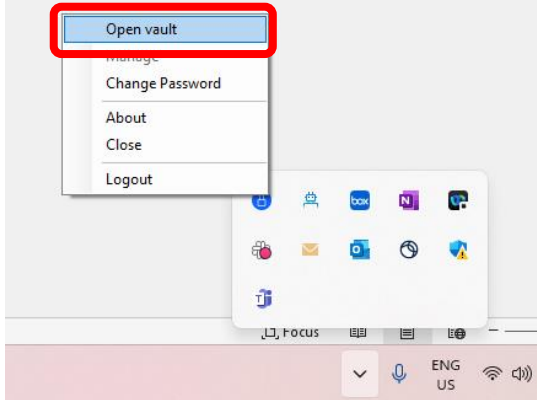
5.3.6 Set up Vault
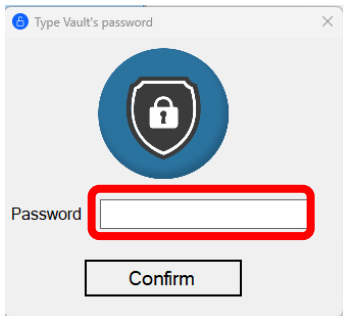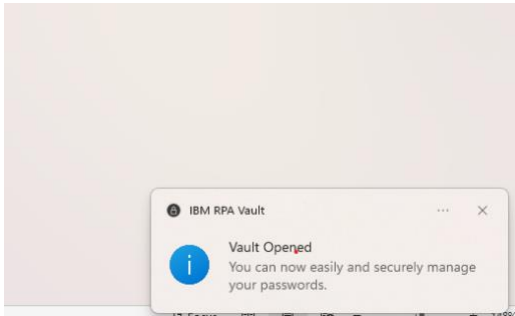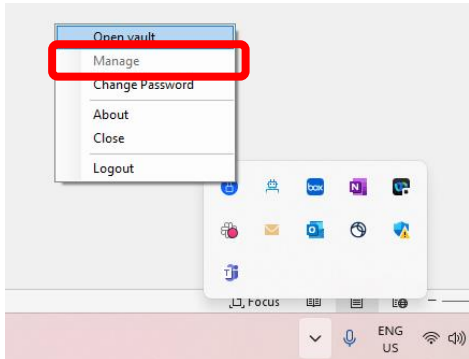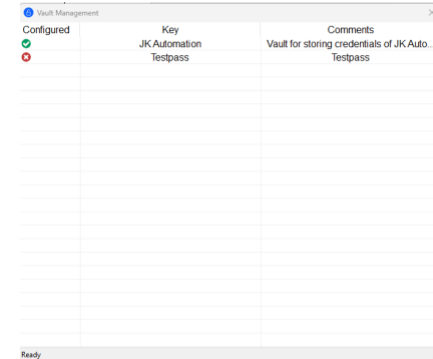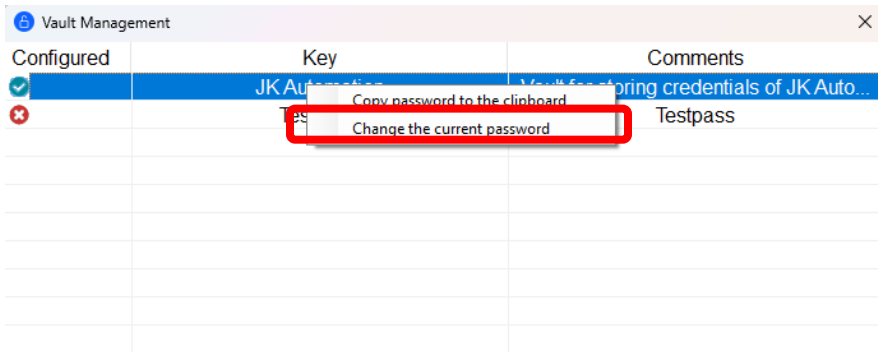
Check if the vault is already setup, then skip this step else follow the below steps to setup the vault:

   a. On your desktop, right click on IBM RPA Vault icon in System tray and click **Open Vault.** If the icon is not visible, the IBM RPA Vault may not be running. In this case, you can run the Vault.exe file located in the IBM Robotic Process Automation installation folder. **C:\Program Files\IBM Robotic Process Automation\IBM Robotic Process Automation Client**

   b. For first time login: configure the master password to be used to open the IBM RPA Vault. The IBM RPA Vault will open after configuring the master password.

   c. Open the IBM RPA Vault by typing the configured master password.

Your vault will open with **blue color** in the system tray.

5.3.7 Set up Vault Credential

Check if vault is already setup, otherwise setup the vault in section 5.3.6.

#	Description
1	Right click on IBM RPA Vault icon in System tray.
2	Right click on **Vault** icon.
3	Select *Open Vault* option from the menu.

4	Complete the setting of the vault by entering the configured password. You will receive a confirmation notification at taskbar mentioning that your Vault has opened. If vault is already opened, skip to step 5.
5	Right click on the Vault again from task bar and select **Manage** option. The Vault Management window will open.
6	Look for *JK Automation credential* which was setup earlier in Control Center.
7	 Right click on the same and select the option to *Change the current password.*
8	The window to setup the new password opens.

	Please enter your new password ✕ User Name [＿＿＿＿＿] Password [＿＿＿＿＿] Re-Type Password [＿＿＿＿＿] [Confirm] Enter User Name of *'admin'*.
9	Enter password of *'Password10'* and click on **Confirm** button.

This step will setup the credentials in vault. The credential appears Configured in Vault.

6 Build it yourself – Step-by-step instructions.

6.1 Overview

Keeping the focus on the proposal of this lab exercise, it will not be necessary to create the entire mapping of the systems involved in the process, the exercises are aimed at the practical use of Exception Handling in IBM RPA. See Exception handling with IBM RPA for more details on their features and usage.

Use case
In this exercise you will use a bot constructed to automate processing of sales leads and claims request that arrive in a CSV/Excel format. Each row of the file represents a separate sales lead/claim. The sales leads was a manual process earlier with entering (copy/paste) by an analyst into the online opportunity system of record (JK Automation Sales Leads).

We will be conducting exercises to add exception handling to this pre-built automation to add resilience to the process.
Based on the process, the lab will also be divided into following exercises:

- *Exercise No 1 – Use Input Validation*:
  This is the initial step of the process, where the script will be added with input validations to check the input leads data for validations of business data.
- *Exercise No 2 – Identification of Exception*:
  The error which is triggered due to unexpected flow of events is identified and categorized at this stage.
- *Exercise No 3 – Configure Business Exception Handler:*
  All the business exceptions faced during runtime are dealt with building logic at this stage.
- *Exercise No 4 – Configure System Exception Handler:*
  The logic for handling all system exceptions faced during runtime are configured in this stage.
- *Exercise No 5 – Ignoring Error and Resuming from next step:*
  The logic for ignoring error and resuming from the next step.

6.2 Exercise 1: Input Validation

The **best practices** around identifying the type of error and what action is to be taken is stated, to begin with, during **requirements gathering** activity. At the time of creating process **definition/design documentation**, a placeholder for exception handling is added to ascertain which all business exceptions can be expected and what action to be taken for either type of exception.

6.2.1 Exception Handling requirements in Process Definition Document

You will find below a sample artifact of Process Definition document containing requirements for adding exception handling catered to our scenario of JK Automation.

14.	Exceptions
14.1	**Business Exceptions** The login credentials of JK automation to be obtained from vault. If the vault's credentials get expired, the SME to be notified in attended run with a message stating the credentials are expired and if he would like to update the same in vault. Based on the action selected by the SME, the bot will either retry to add credentials in login page after confirmation from SME or throw a business exception detailing it to be a login issue wherein the process will re-run after credentials are updated.
14.2	**System Exception**
	The error information to be captured and details to be sent on email to SME.

At design phase, the details about the **contacts**, who will be emailed to intimate for business and/or system exceptions, will be provided and error information to be communicated will be finalized. The email group from where the intimation of exception is initiated will also be decided at this stage.

> **Best Practice**:
> Important to document for exception handling:
> - Business Exception –List the business scenarios and cases that are out of scope of automation ie the cases that will not be worked and will be passed to the business for reviewing manually.
> - Known system exceptions – Listing out of scope system responses.
> - Finalise on reporting mechanism wrt whom to inform, what information to provide, specifying email details etc.
> - Error information which will be shared to business users will be finalized.

6.2.2 Exercise for Input Data Validation

From the input file, we validate the data to bring out cases which are **out of scope** and treat them as **business exceptions**. It is important to differentiate between exceptions for 'in scope' cases and 'out of scope' cases. If a process was designed to ignore cases for applicants of a particular country, these sorts of exceptions should not be seen as errors.
In our exercise, we will look at two types of **data validation**:
  a. Input file is Empty
  b. Invalid Email address

For both cases, we will highlight them as business exception and after logging and capturing of error details, we will send an email to the user informing them about the invalid input suggesting them to correct the file and run again.

Now let us look at the exercise to configure input data validation.

6.2.2.1 Open Base Script

Open the base script creating sales lead and claim request with no exception handling added. Follow steps in **5.2.5 section.**

Save the script as **sales-lead-automation-inputvalidation-added.wal**.

6.2.2.2 Input File is Empty

We will make changes in subroutine *ValidateSalesLeadInput* in the above mentioned script.

#	Description
1	In the subroutine, at **line 208**, add command of *Throw Exception* for handling case of input file being empty.
2	Add Message '*Business Exception - Input file is empty*'.
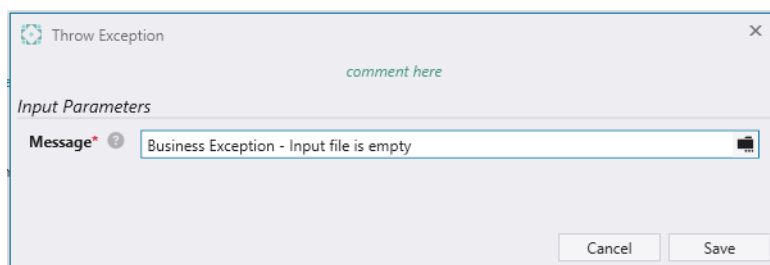
* Delete comment of Add Throw Exception at every script change.
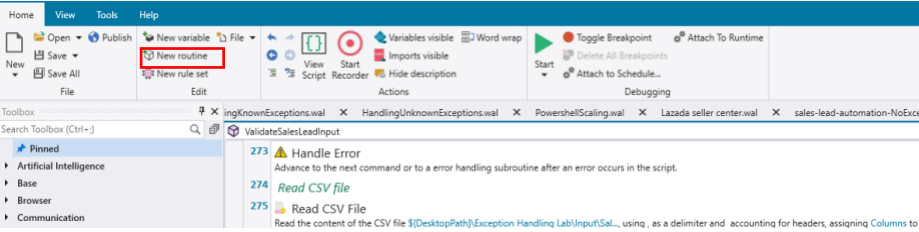
6.2.2.3 Invalid Email Address

#	Description
1	In the subroutine, at **line 214**, add command of *Throw Exception* for handling case of Invalid Email Address.
2	Add Message '*Business Exception - Invalid Email Address*'.
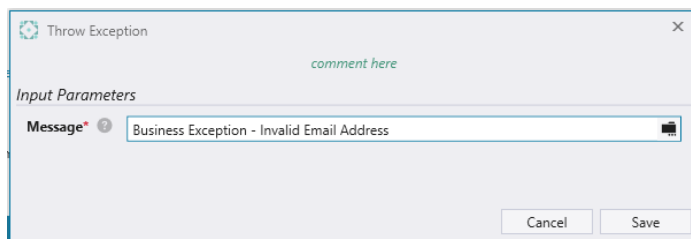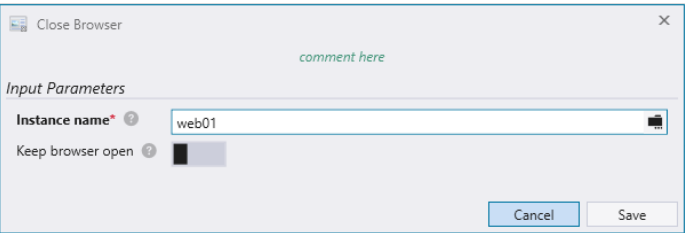
We will now create the **Input Validation Handler**.

6.2.2.4   Create new routine
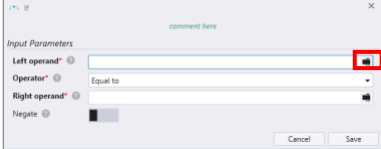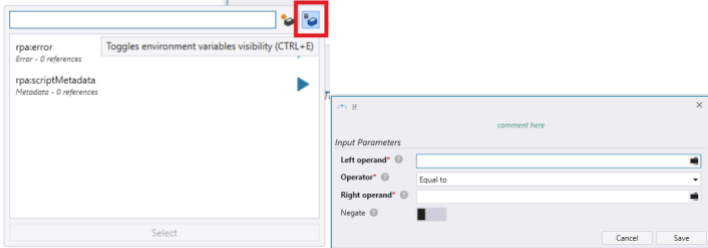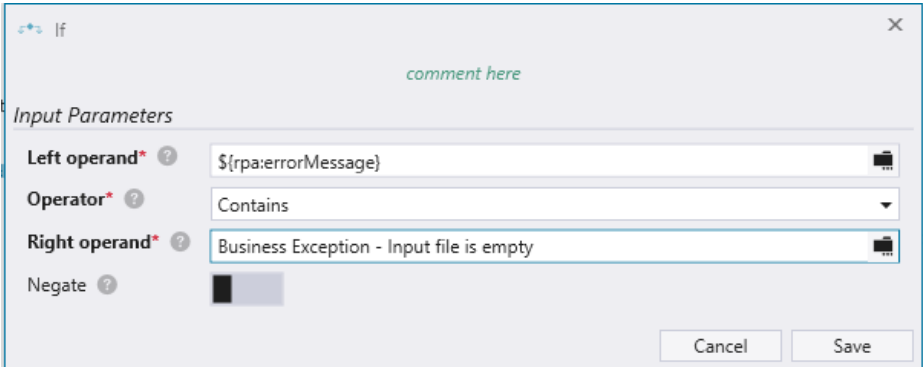
#	Description
1	On the homepage, click on *New Routine* option.
2	Add name of *Input_Validation_Handler* of the new subroutine and click on Save.
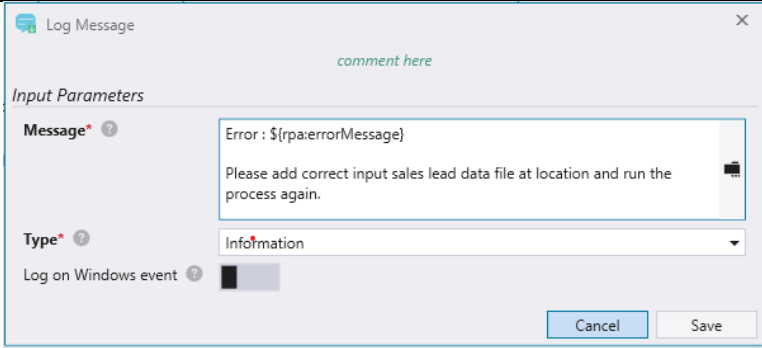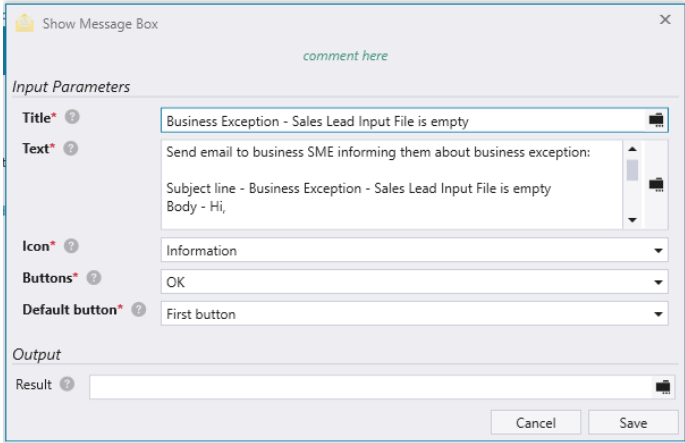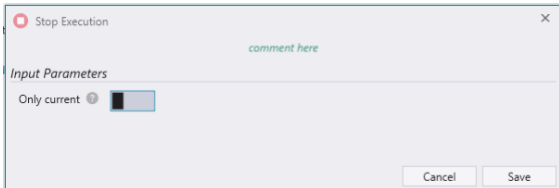
6.2.2.5   Close opened browser

#	Description
1	In the routine *Input_Validation_Handler*, add command *Close Browser*.
2	Add instance name of *web01* and click on Save.
3	Add command Log Message of 'Entered input validation handler'.
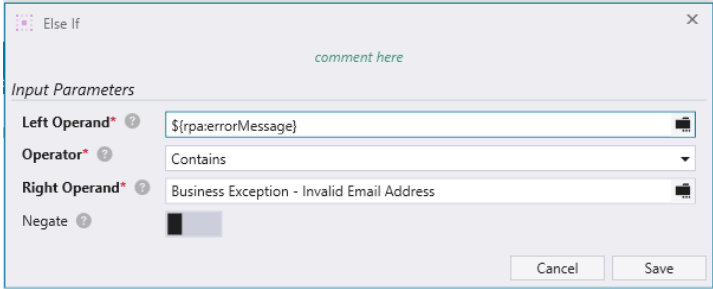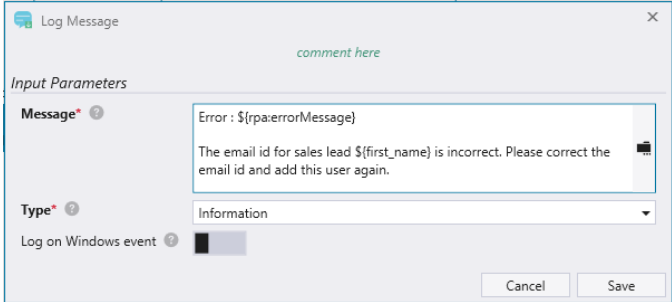
6.2.2.6   Handling Input file is Empty error
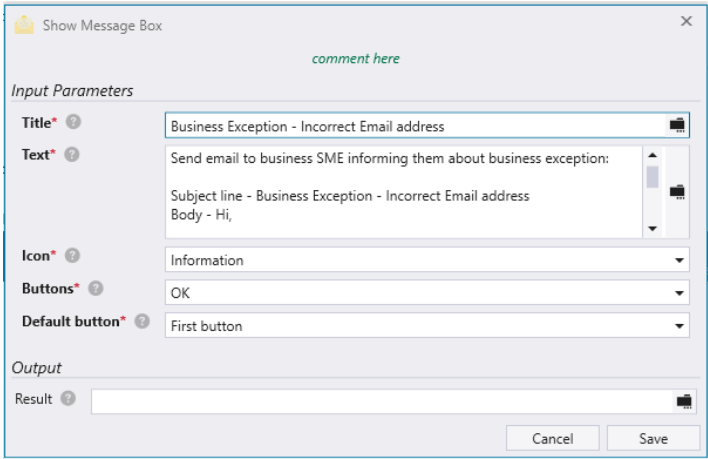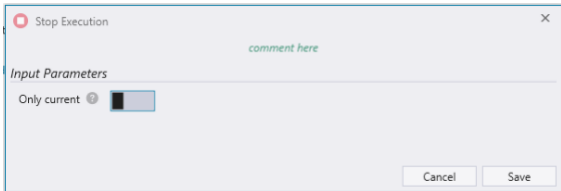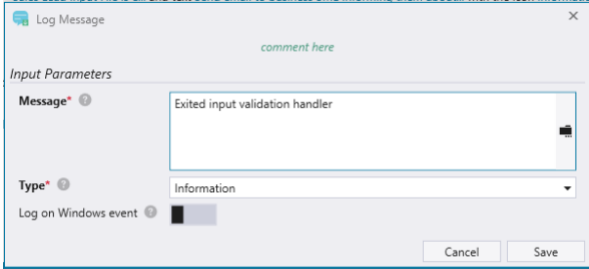
#	Description
1	Add **If condition** command.

2	Set criteria – Left operand:
	
	    c. Click on select a variable option.
	
	    d. In the window which opens, toggle environment variable visibility by clicking on the button.
	
	    e. Search error in search bar and select the option for rpa:errorMessage. This opens the error variable set. Click on Select.
	
	Operator: Contains (select from dropdown)
	
	Right operand: 'Business Exception - Input file is empty'
	
	An If and End condition appears.

3	Add command of **Log Message** inside **If condition**.
	Add **M*essage***: '${rpa:errorMessage} Please add correct input sales lead data file at location and run the process again.'

4	Add **Show Message** command.
	Add **Title** : '*Business Exception – Sales Lead Input File is empty*' Add **Text** : '*Send email to business SME informing them about business exception:* *Subject line - Business Exception - Sales Lead Input File is empty* *Body - Hi,* *Please add correct input sales lead data file at location and run the process again.* *Error : ${rpa:errorMessage}* *Thanks.* *RPA Bot Team'* Retain rest of the parameters and Click on Save.
5	Add command of **Stop Execution** and click on **Save.**

6.2.2.7 Incorrect Email address Error

#	Description
1	Add **Else If** condition at **line 228**. Set criteria – **Left operand**: '${rpa:errorMessage}' **Operator**: contains (select from dropdown) **Right operand**: 'Business Exception – Invalid Email Address'
2	Add command of *Log Message*. Add **Message**: '${rpa:errorMessage} The email id for sales lead ${first_name} is incorrect. Please correct the email id and add this user again.'
3	Add *Show Message* command. Add **Title**: 'Business Exception – Incorrect Email address' Add **Text**: Send email to business SME informing them about business exception: Subject line - Business Exception – Incorrect Email address Body - Hi, The email id for sales lead ${first_name} is incorrect. Please correct the email id and add this user again. Error : ${rpa:errorMessage}

Retain rest of the parameters and Click on **Save**.

#	Description
4	Add Command of **Stop Execution** and click on **Save.**

#	Description
1	Add command of **Log Message** after ending of If condition.
2	Add message *'Exited input validation handler'* in the message box and click on **Save**.

The final subroutine appears as follows:
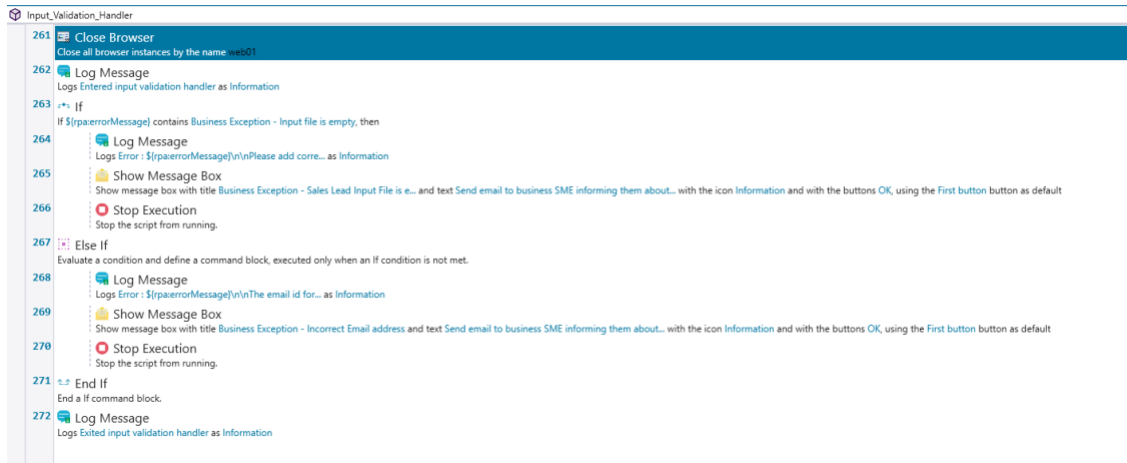
Now, we will add the error handling to the subroutine.

6.2.2.8   Adding error handler to subroutine.

#	Description
1	Open subroutine **ValidateSalesLeadInput**.
2	At the top of this subroutine at **line 203**, add command **Handle Error**.
3	Select subroutine '*Input_Validation_Handler*' and click on Save.
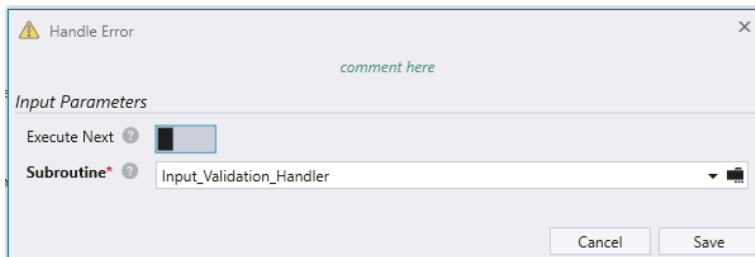
The final subroutine for ValidateSalesLeadInput looks as follows:
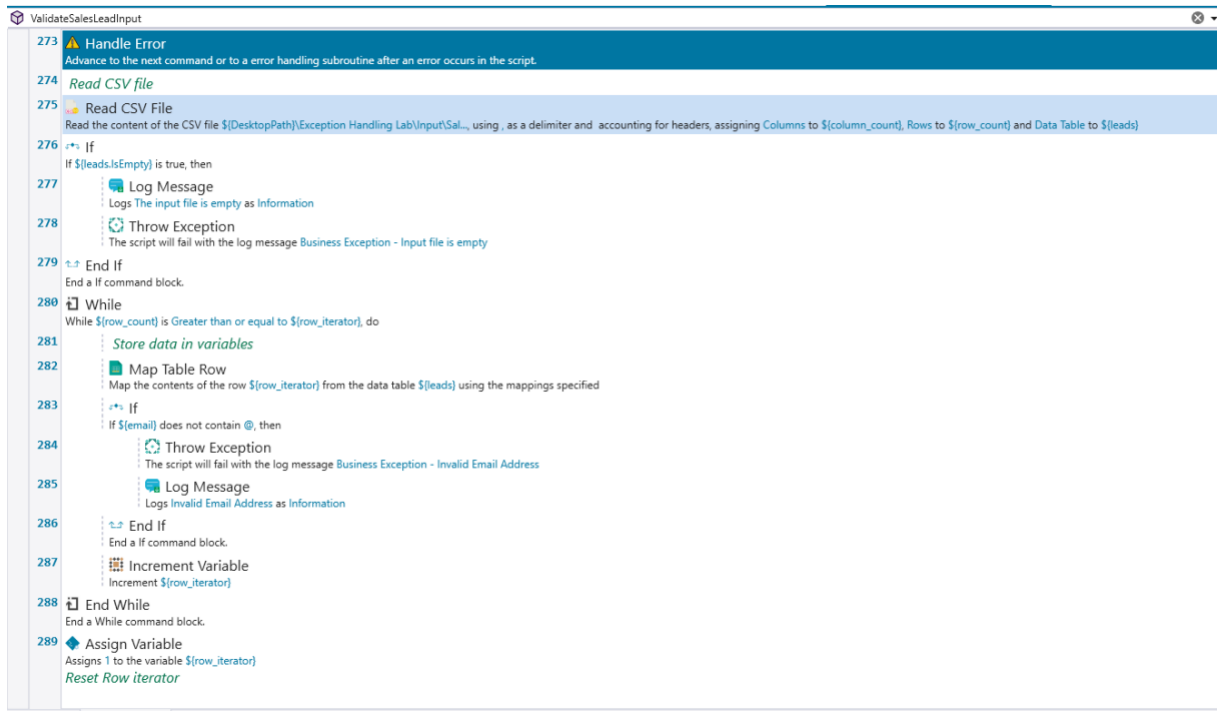
6.2.2.9 Testing Input Validation:

Test No 1

Steps	Create a copy of SalesLead.csv and empty the original file SalesLead.csv and run the process, by selecting **Start without Debugging.**
Output	**Business Exception – Input File is Empty will be thrown,** email will be sent to the user informing them about the file issue, suggesting them to correct the file and run again. The process will be stopped at this stage.

Test No 2

Steps	Open SalesLead file and change the email address of Dave to dwakemanus.ibm.com ie removing @ and run the process, by selection **Start without Debugging.**
Output	**Business Exception – Invalid Email Address,** email will be sent to the user informing them about the incorrect data issue, suggesting them to correct the file and run again. The process will be stopped at this stage.

6.2.2.10 Best Practise

> **Best Practice**:
> - When you decide to label exceptions, it is important to differentiate between exceptions for 'in scope' cases and 'out of scope' cases. Business scenarios which are encountered due to input validation should be treated as 'out of scope' and will not be considered as failure of process.
> - List all input validations with business users, which may be out of scope eg juvenile or could cause problem in running of process.
> - Place validation checks by adding decision stages on input data.
> - Input validation step to be performed at the starting of the process.
> - After encountering business exception, the handling should be closed with **Stop Execution**

6.3 Exercise 2: Isolate exception handling

The exception handling can itself cause problem, where one issue handling can send handler in infinite loop. Isolating your code to better handle specific issue is ideal for implementing better error handling.
Now we will see how we can achieve this with IBM RPA.
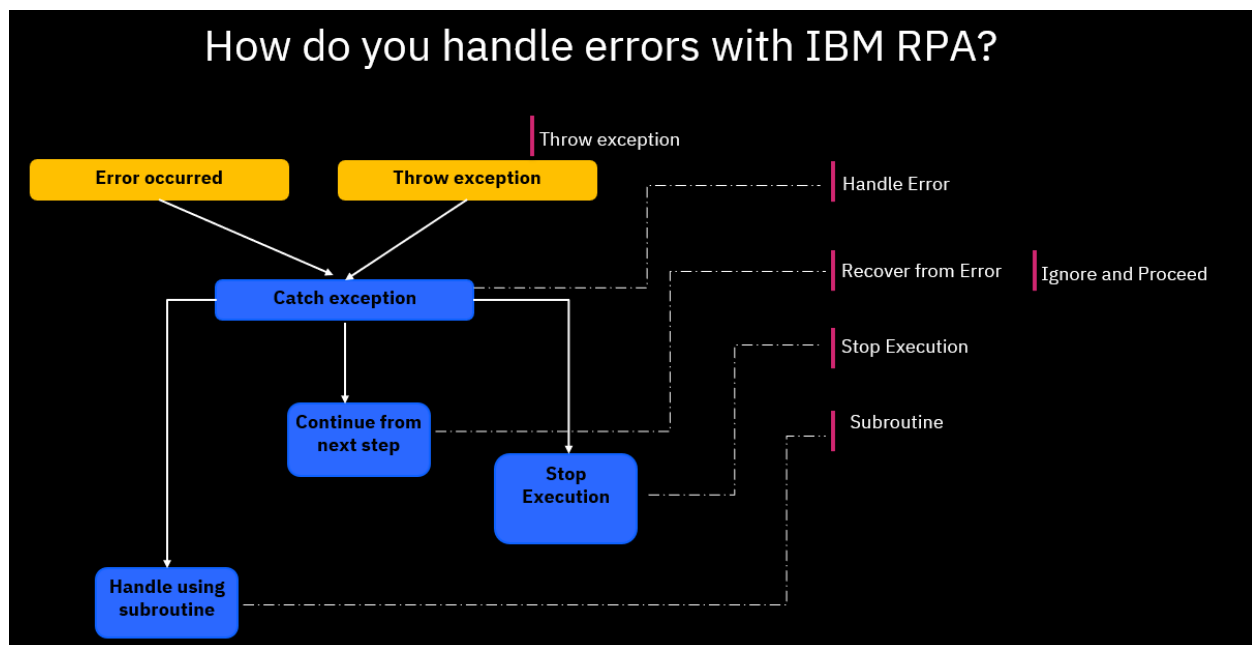
The best practise is to keep your **handle error** above specific command or create smaller subroutines to handle the error handling better for that part of code. The error handling subroutines can be added in such a way that they check based on conditions which type of exception the same is. After isolating the exception type, the handler also contains resolution mechanism which will vary based on exception type.

Handle Error command scope
The scope of the main routine encompass other subroutines if you call them with the command Run Subroutine (goSub), Run Subroutine If (goSubIf), or Execute

[Subroutines](#) (goSubs). The following image shows the **Call Graph** tab from IBM RPA Studio of a script with four subroutines.
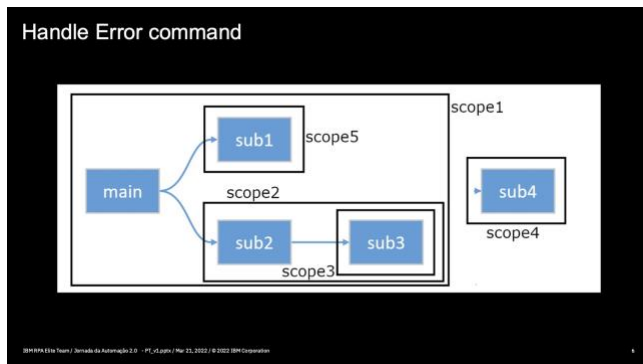
The **Handle Error** (onError) command watches for *Runtime Exceptions* in the subroutine scope where you call the command and the subroutine's inner scopes. The command will watch for raised Runtime Exceptions from the moment the command runs until:

- The end of the innermost subroutine scope
- The appearance of a new **Handle Error** (onError) command call

If a subroutine does not have Handle Error command specifically configured for it, the errors will *bubble up* to the parent subroutine or main page, as the case may be.

Error Environment Variable

The environment variable for IBM RPA to record error data is shows as follows:

It has the following parameters:

The error variable has the following parameters:

Message

Holds the exception message.

LineNumber

Holds the line number where the exception event took place.

Command

Holds the command name that raised the exception.

Script

Holds the script name that raised the exception.

Routine

Holds the subroutine name where the exception took place.

StackTrace

Holds the call stack trace of the exception. The call stack trace is the hierarchy of subroutines and script calls until the command that raised the exception.

Count

Holds the number of elements in StackTrace.

redTime

Red Time configured for script version that started the execution.

Best Practices for Logging exception data

It is good practice to record enough data to identify the reason for the unknown exception. Record as many data as you can by:

- Logging event messages and variable values
- Taking screenshots when automating GUIs
- Recording the screen during the last attempt when automating GUIs

6.3.1   Best Practise

Best Practice:
- Build smaller subroutines to represent small usable business function and add error handling for as many conditions as can be imagined for each subroutine.
- Building exception handler inside each subroutine can help in handling error specifically faced in that subroutine.
- If an error occurs in a subroutine and there is no error handling added there, the exception is going to *bubble up* and will be caught by the next Handle Error command present in **parent or main subroutine**.
- The scope of error handling and subroutines can be seen in *Call Graph*. It is **best practice** to check the scope.

6.4 Exercise 3: Business Exceptions

The cases appearing in a flow may sometimes contain some parameter that may make the case impossible to work with. By placing some *validation logic*, you may want to filter out cases which are **out of scope** of current process operation.

For example, the ID to be provided for Claims process requires 10 digit, it would make no sense for bot to attempt a case if it has only been given 7 digits.

These types of exceptions, based on *'business rules'*, should be seen as different from a technical error. When a case is not completed due to such logic, it should not be seen as a problem. The bot is behaving according to the rules. Though, a case that did not flout any business logic but could not be completed should be seen as an error.

In our example, input validation and login issues could be categorized as business exceptions and dealt with accordingly.

6.4.1 Exercise for Login error

As presented in the Architecture Overview, the entire mapping of artifacts involved in the process are already ready. So, in this exercise, we will edit each of the scripts to include only the unique commands for using the IBM RPA Exception Handling:

6.4.1.1 Open Base script

Open the base script creating sales lead and claim request with no exception handling added. Follow steps in **5.2.5 section**.

Save the script as **sales-lead-automation-businessexception-added.wal**.

6.4.1.2 Open **Login** Subroutine from the subroutine panel on the right side.

6.4.1.3 Add logic for throwing business exception.

The logic for checking if the login is successful is already added in the script as shown below:

```
Login
  90  handle error command to be added here
  91  ◆ Assign Variable
      Assigns 1 to the variable ${index}
  92  ✎ Insert Label
      Add a marker in your script to use with the Go to command.
  93  ▦ Get Vault Item
      Gets the vault whose name is ${VaultKeyName}, assigning Password to ${password}, User Name to ${username} and Success to ${vaultRetrieved}
  94  ▦ Set Value to Field
      Set the value ${username} on an element with CSS value body > div:nth-child(4) > form > div:nth-child(..., waiting for a response for default timeout
● 95  ▦ Set Value to Field
      Set the value ${password} on an element with CSS value body > div:nth-child(4) > form > div.form-group..., waiting for a response for default timeout
  96  ⟲ Click on Web Page
      Perform a Click on element selected by, body > div:nth-child(4) > form > div:nth-child(... Css
      Click on Submit button
  97  ⊕ Wait for Element in a Web Page
      Wait 00:01:30 for an element with CSS value body > div > div.col-md-2.leftNavColumn > div > ..., assigning Result to ${LoginSuccessFlag}
  98  •⊷ If
      If ${LoginSuccessFlag} is false, then
  99      ☁ Show Message Box
          Show message box with title Change Vault Credentials and text Change Vault Credentials with the icon Question and with the buttons OK/Cancel, using the Second button button as default, assigning Result to ${ResultVaultButton}
 100      •⊷ If
          If ${ResultVaultButton} is Equal to OK, then
 101          •⊷ If
              If ${index} is Less than or equal to 1, then
 102              ⊡ Navigate
                  Navigate to https://jk-automation-container.12kvw4kay3ug.us...
 103              ◆ Assign Variable
                  Assigns 2 to the variable ${index}
 104              ≫ Go to
                  Go to the label named LoginAgainLabel
 105          ⥅ Else
              Define a command block executed when the If condition is not met.
 106              add Throw exception here
<> Script   ⛶ Designer   ▣ Call Graph
```

As can be seen, the logic waits for the JK Automation main page to appear after setting values of **username** and **password** on **login** page. If the login is unsuccessful, the script prompts a message box which informs the user about the **invalidity of the credentials** and ask if they could update the same. The user will decide whether he would like to change the credentials now or at a later stage and re-run the process. Accordingly, our logic will check for vault credentials again or it may not.

We will now add the logic for throwing a **business exception** where **login is not successful**.

Normally when we encounter a **business exception**, we are not supposed to retry the same as the **error** has happened due to **deviation from normal flow**. In some circumstances, such as, in case of login, it may result in application getting locked. However, in our circumstance, since this is attended mode automation, we may give an option to correct the credentials in vault and run again.

#	Description
1	In the Else condition at **line 107**, add command of **Throw Exception**.
2	Add Message as '*Business Exception - Login credentials invalid*'
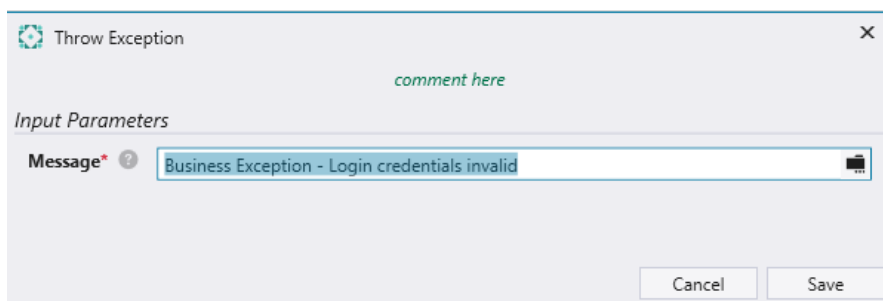
This throws a business exception when the user does not enter correct credentials in vault.

#	Description
1	In the **Else** condition at **line 110**, add command of ***Throw Exception***.
2	Add Message as '*Business Exception - Login credentials invalid*'

This throws a business exception when the user cancels the request for resetting password.

6.4.1.4 Create Login Exception Handling routine
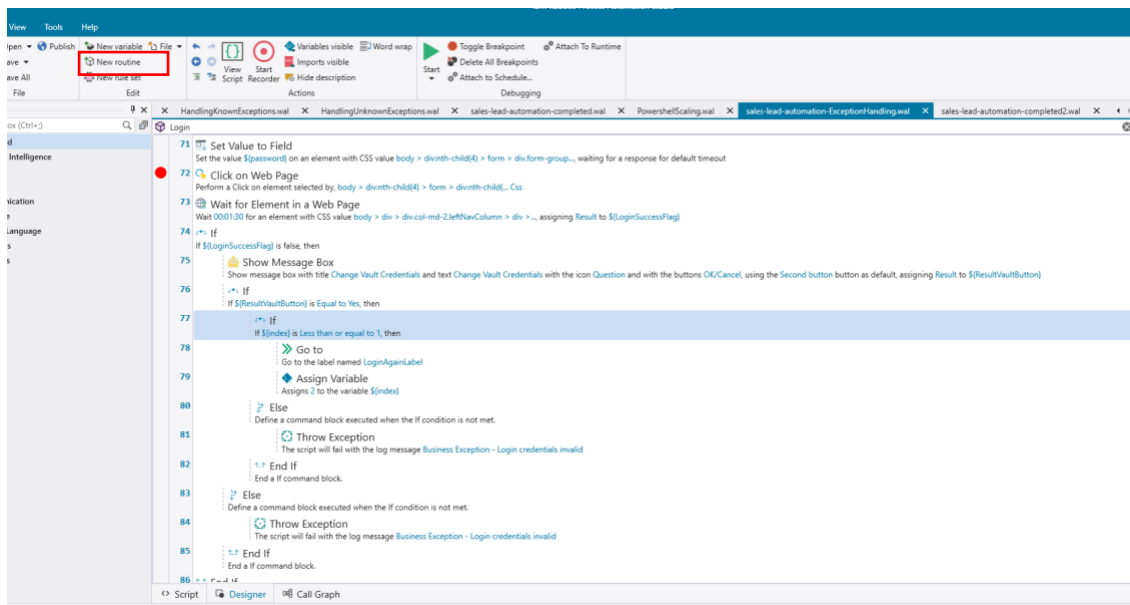
#	Description
1	In the Home tab, click on ***New routine.***
2	Add **Name** of the subroutine as '*LoginExceptionHandler*' and click on ***Save*** button.
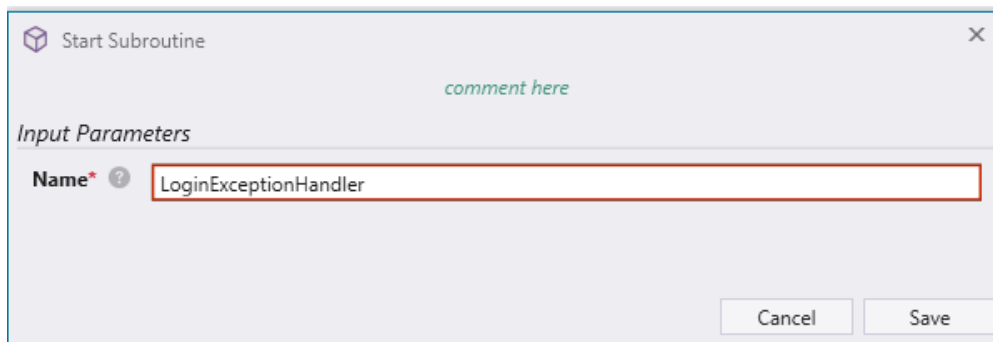
6.4.1.5 Add condition for checking Login error

#	Description
1	In the subroutine, add **If condition** command.

2	Place the condition checking if *${errorMessage} contains 'Business Exception – Login credentials invalid'* and click on **Save** button.

6.4.1.6   Add Logging of Error message

#	Description
1	For the **If condition** is *True*, add **Log Message** command.
2	Add message *'Please update the credentials in the vault JK Automation and rerun the process.* --- *${rpa:errorMessage}'* in log message command with **Type** as **Information** and click on **Save** button.

6.4.1.7   Email to SME

After the login error has been identified by the exception handler, an automated process needs to apply **error notifications** as formalized during define and design phase.

In case of business exception, the **business SME** will be informed about the issue through email to aid him in taking appropriate action to correct the issue and handle the running of the process.

We added in **Message box**, the details which can be sent on email applying best practices around notification.

#	Description
1	Add command **Show Message Box**.
2	Add **Title** as '*Business Exception - Login credentials to be updated*'.
3	Add **Text** as '*Send email to business SME informing them about business exception:* *Subject line - Business Exception - Login Credentials invalid* *Body - Hi,* *Please update the credentials of JK Automation in vault. After updation, run the process again.* *Thanks.* *RPA Bot Team*'
4	Add Icon as '*Information*'.
5	Click on **Save** button.

6.4.1.8 Add Stop Execution command

#	Description
1	Add **Stop Execution** command and click on **Save** button.

This completes the error handler for login business exception.

6.4.1.9 Add Handle Error command

#	Description
1	Go to subroutine of **Login**
2	Add command **Handle Error** at the top of the subroutine
3	Select subroutine of 'LoginExceptionHandler' from dropdown.
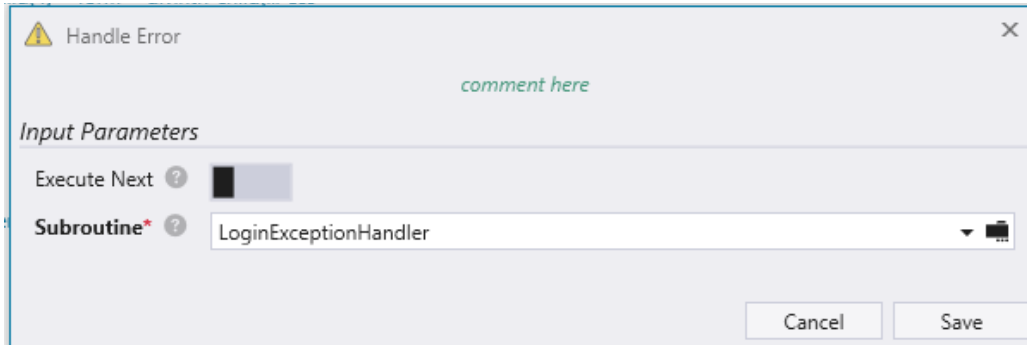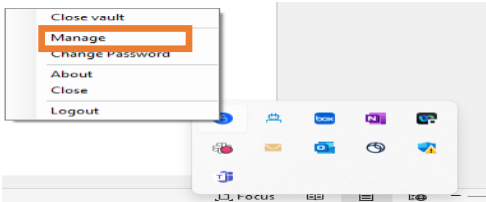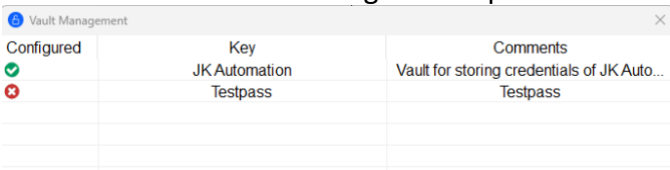
6.4.1.10 Testing the Login error
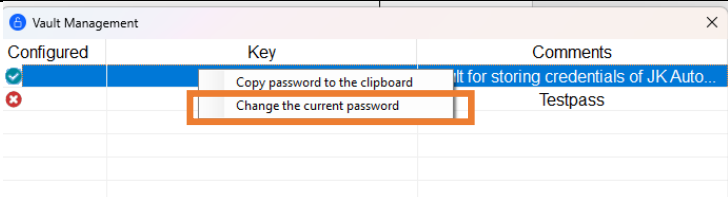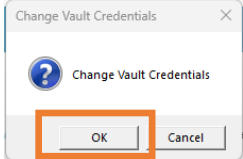
We can test out the Login error by two ways:

Test No 1 – Providing correct vault credentials

Steps	Check if vault is logged in and run the process.
Output	The process will run fully adding cases for sales lead and claims requests.

Test No 2 – Providing incorrect vault credentials and correcting them

Steps	a. Right click on vault icon on taskbar. b. Click on Manage option. c. The window for Vault Management opens. d. Right click on **JK Automation credential** and click on **change the current password**.

e. Enter username as admin and Change the *password* to 'Password2'.
f. Run the process **Start without debugging**.

Output	a. The process will throw a message box saying 'Change Vault Credential'? If the message does not appear, minimise windows to see the message. b. Go to Vault and change the password back to Password10. c. After changing credential in Vault, click on **OK**. The process will run again successfully.

Test No 3 – Cancelling request to change vault credentials

Steps	a. Add incorrect password in Vault as per Test 1. b. Check if vault is logged in and Run the process from Studio and select **Start without Debugging.** c. Cancel the message popup for Changing Vault Credentials.
Output	The process will stop with Business Exception of 'Login Credential Invalid'

Go to Vault and change the password back to Password10 for running of next exercises.

6.4.2 Best Practice

> **Best Practice**:
> - When you decide to label exceptions, it is important to differentiate between exceptions for 'in scope' cases and 'out of scope' cases. Business scenarios which are encountered due to input validation should be treated as 'out of scope' and will not be considered as failure of process.
> - After encountering business exception, the handling should be closed with **Stop Execution** command as we cannot proceed further with current processing.
> - Business Exceptions are not supposed to be retried as you already know that the business logic is different from the agreed flow.

6.5 Exercise 4: System Exceptions

The errors which occur due to behavior of any application result in **system exceptions**. The applications sometimes behave in not flawless manner, crashing, freezing, stalling, running slowly or performing in a way they are not supposed to. In a production environment, for an

uninterrupted automation, we must cater in to accommodate for **unpredictable behavior** of the applications the bot is interfacing with.

Based on which type of system exception is encountered, proper handling needs to be in place as these can cause the process to fail.

At a broad level, we can classify system exceptions to be of two types:

1. ***System Exception – Try Only Once*** – These are cases of process failure where retrying may not be an option either owing to the process being risky and may have financial implications or any other situation.
   For example, the bot fails at the command of submit button where you are submitting business critical data and we are not sure if the update was done or not (ie the system becomes unresponsive) and should therefore not retry this case.

2. ***System Exception – Systems Unavailable*** exception if occurs due to some system or internal exception, the process should check, if possible, if the required systems are still available.
   If the system is *not available*, an **exception** must be **thrown** and the process should attempt to start the systems periodically until it is available again. However, retrying to gain control of systems should be done ***upto a limit*** post which a ***final exception*** can be **thrown** resulting in **process failure** but all users will be informed with *proper error detail* to assist them in handling the cases manually and returning process back to normal.
   In this exercise, we will create the System exception – Handler subroutine first and then associate the subroutine as per the situation.

6.5.1   Exercise Overview

In this exercise, we will work with the same script sales-lead-automation-NoExceptionHandling.wal script.

6.5.1.1   Open the base script creating sales lead and claim request with no exception handling added. Follow steps in **5.2.5 section.**

Save the script as **sales-lead-automation-SystemExceptionHandling-added.wal**.

6.5.2   System Exception – Exception Handler
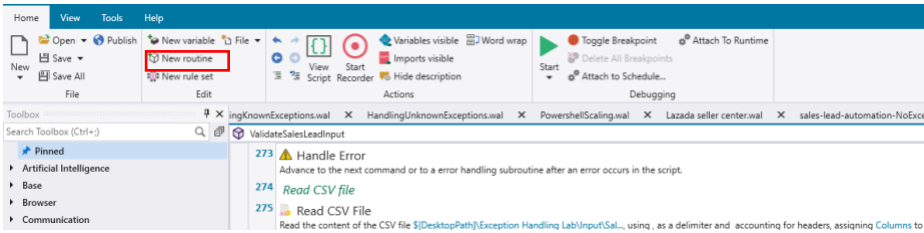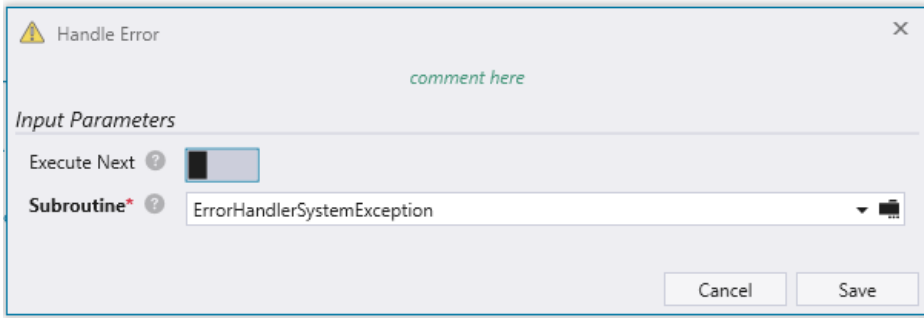
Now, we will create the ***exception handler*** to handle all scenarios of system exceptions. We will answer the following important factors when constructing exception handling for system exceptions.

- Which cases can be retried and which cannot be?
- What will be the maximum times that we will retry the case.
- What communication channel(s) we will use to communicate about the error faced. Will it be via email, reports etc.

- What is important information that the business user will need in order to handle the case manually or assess the problem of system exception. Eg, screenshot, error message, command line, subroutine name etc.
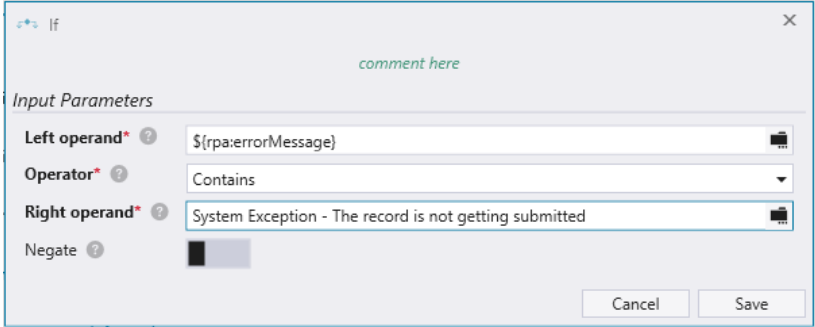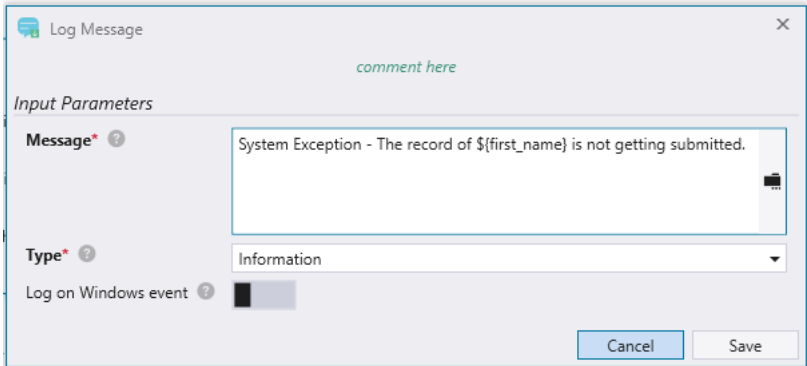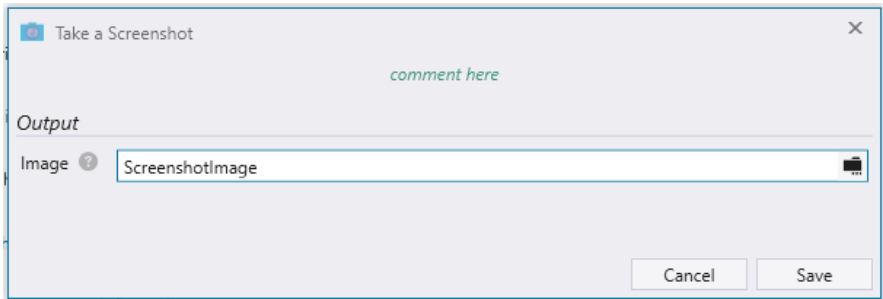
Exercise for creating handler subroutine to handle the error.
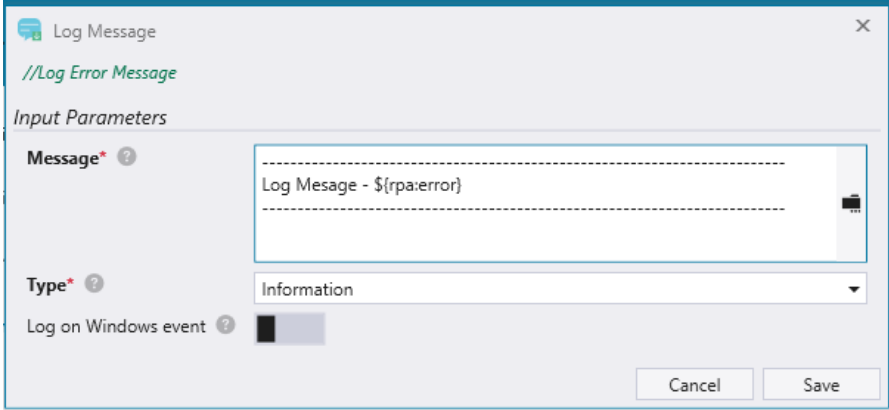
6.5.2.1 Open System Exception handler subroutine

#	Description
1	On the homepage, click on ***New Routine*** option.
2	Add name '***ErrorHandlerSystemException***' of the new subroutine.

6.5.2.2 Logic for System exception which cannot be retried (Try Once)

#	Description
1	Add **If condition**. In the input parameters, add ***Left operand***: '${rpa:errorMessage}' ***Operator***: Contains (select from dropdown) ***Right operand***: 'System Exception - The record is not getting submitted.'
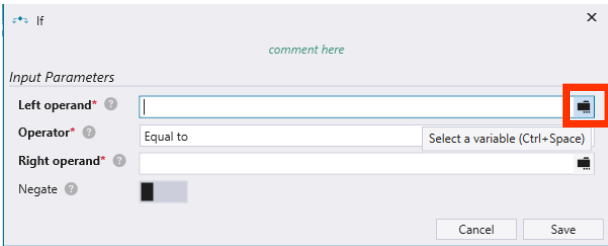
2	Inside the **If condition**, add following commands:
3	**Log Message** **Message**: 'System Exception - The record of ${first_name} is not getting submitted.' **Type**: Information
4	Add command of **Take a Screenshot** In **Output**, add parameter for Image: ScreenshotImage
5	Add command of **Log Message** **Message**: '--- Log Message - ${rpa:error}'
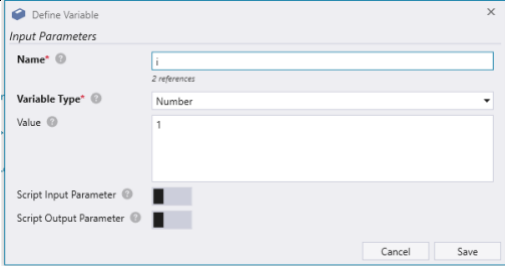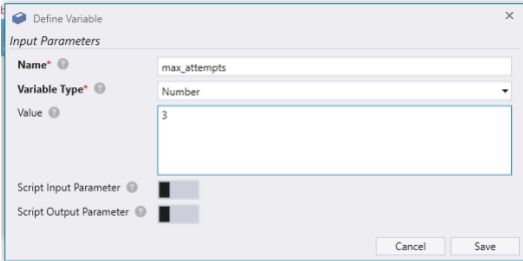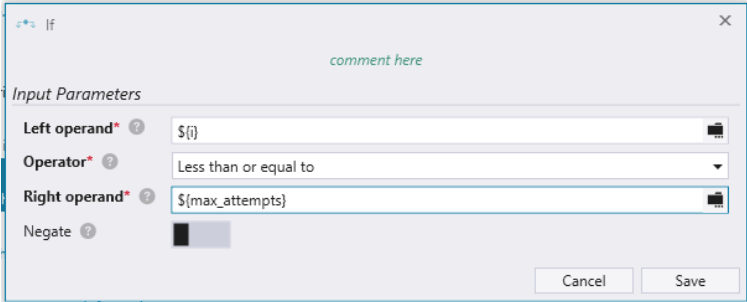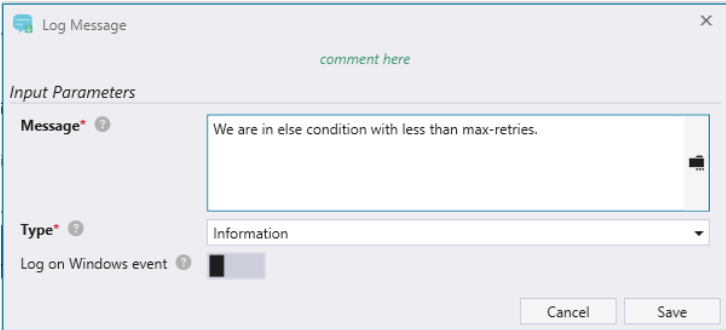
6	Add command *Recover From Error* and click on *Save button.*
	This command is added when we want the bot to **recover and resume** after an error takes place and resuming operations. Whenever an error occurs, it results in a *runtime exception event.* Then we would like to pursue normal operations after an error occurs, we should always use this command of recover from error to come out of error conditions.
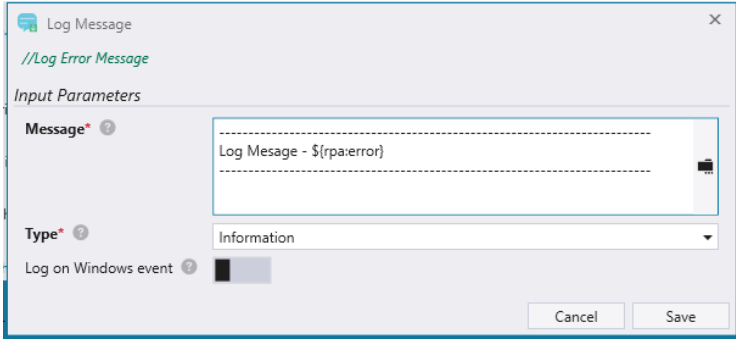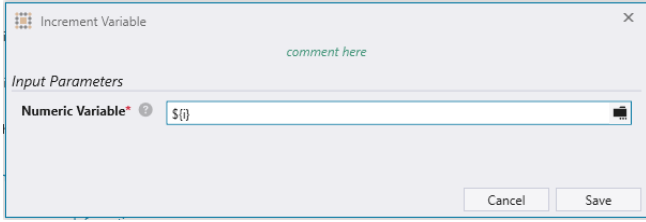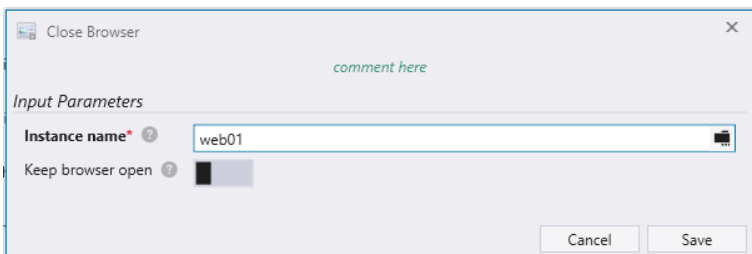
6.5.2.3 Logic for attempting to retry upto maximum limit
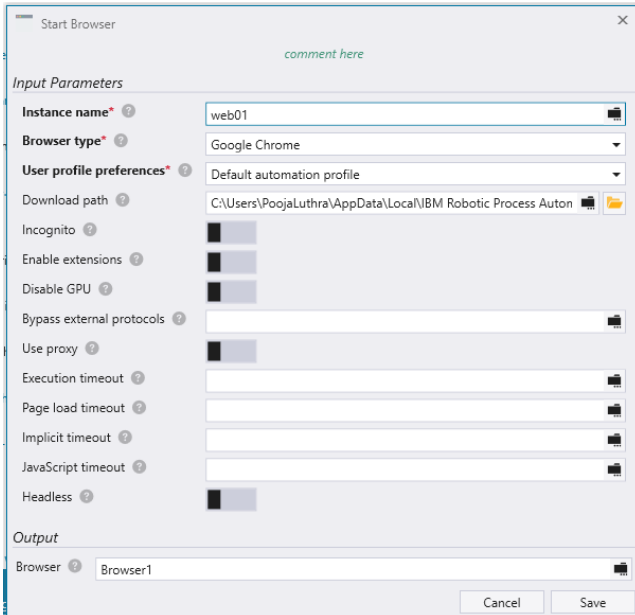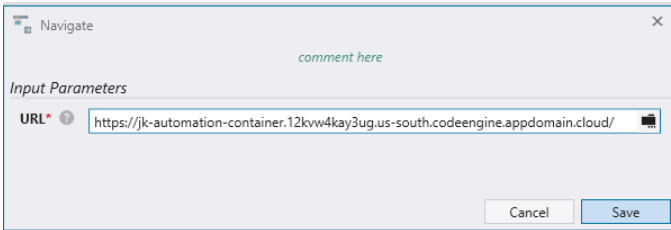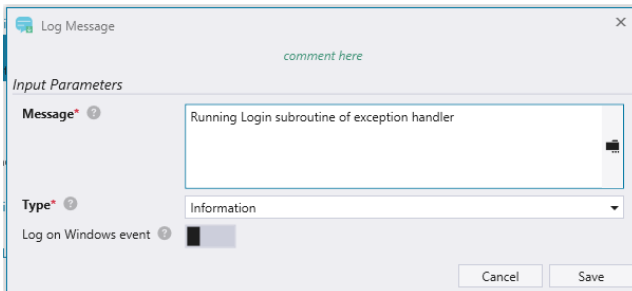
Next we will add logic for attempting to retry upto maximum limit.
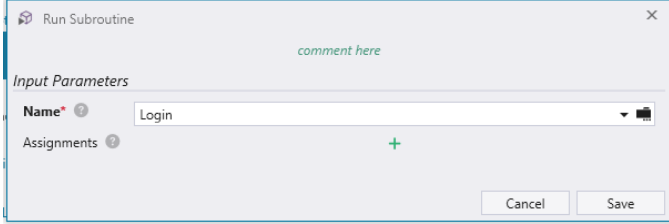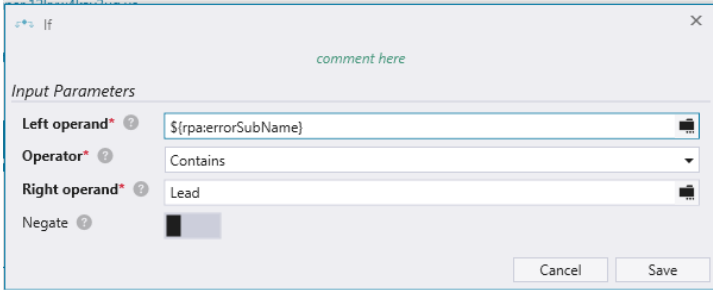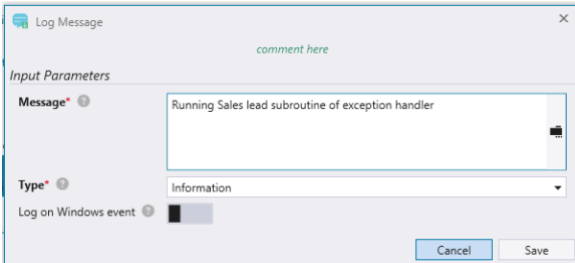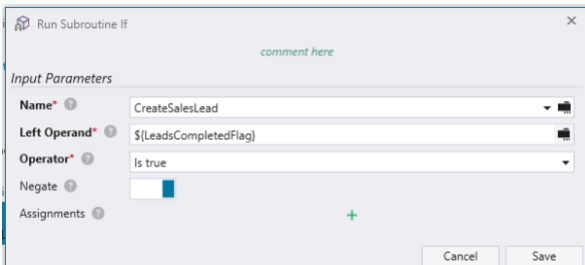
We will place this piece of instructions in the **Else condition**.
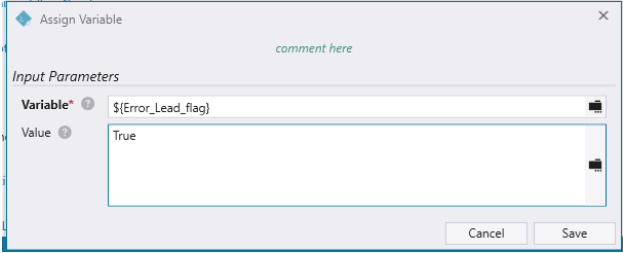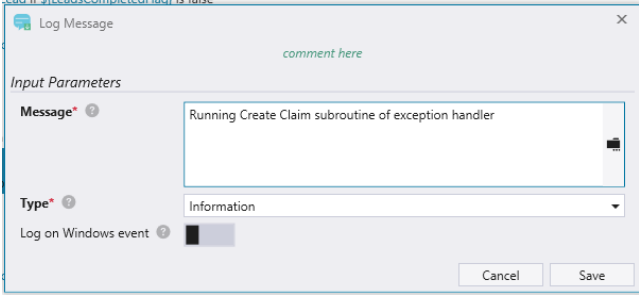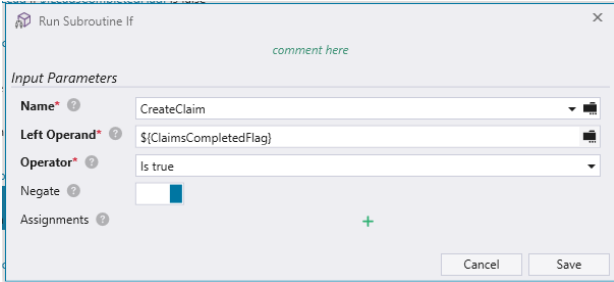
#	Description
1	Logic has been added here to check if current try for resuming is less than max limits.
	From the commands panel, add **Else condition at step no 227.**
2	Inside *Else condition,* add another *If condition*.
	In the *input parameters*, add
	Left operand:
	a. Click on select new variable.
	b. Click on new variable icon.
	c. Add variable **i** with *variable type* as Number and *Value* as 1.
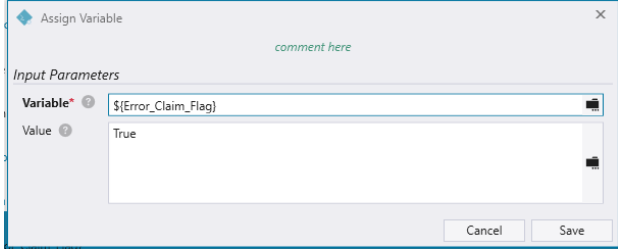
The left operand is added with '${i}'

Operator: Less than or equal to

Right operand: ${max_attempts}

Create max_attempts in similar manner as *i assigning value of 3.*

Final If condition is as follows:

2	Add ***Log Message*** in the If condition, with ***Message*** – 'We are in else condition with less than max-retries'.
3	Add ***Log Message*** command ***Message***:

'―――
Log Mesage - ${rpa:error}
―――'

4	Add command of *Increment Variable.* In *Numeric value*, add '${i}' and click on *Save*.
5	Add command of *Close Browser*. Enter *Instance name* as '*web01*' and click on **Save**.
6	We will add the logic to return applications to normal state, to attempt to *retry* running the *transaction/case.* Add command of *Start Browser* **Input Parameters** - *Instance name* – web01 *Browser Type* – Google Chrome **User profile preferences** – Default automation profile **Download path** – check the path to IBM RPA's downloads, eg C:\Users\PoojaLuthra\AppData\Local\IBM Robotic Process Automation\downloads

	Output – Browser – Browser1 On adding this, a prompt will appear to add **Close Browser**, click on **Cancel**.	
7	Add **Navigate command**. Add input parameter of URL '*https://jk-automation-container.12kvw4kay3ug.us-south.codeengine.appdomain.cloud/*' 	
8	Add **Log message** command. Add message '*Running Login subroutine of exception handler*'. 	
9	Add command **Run Subroutine**. Select the routine '*Login*' from dropdown.	
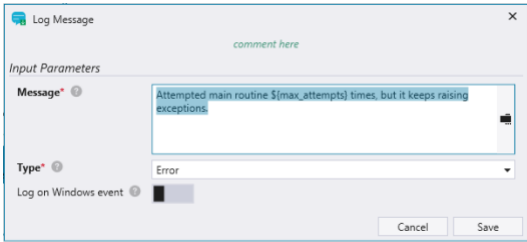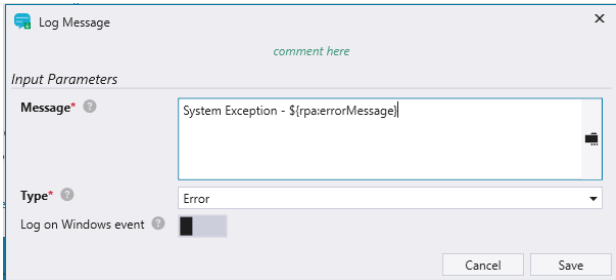
10	We will configure commands to add logic for detecting if the system exception occurred while creating sales lead or claim.
	Add **If condition**, with **Input Parameters**:
	Left operand: '${rpa:errorSubName}'
	Operator: Contains
	Right operand: Lead
11	In the **2ⁿᵈ If condition** add the following commands:
	a. Add **Log Message** with Message of '*Running Sales lead subroutine of exception handler*'
	b. Add command **Run Subroutine If** with parameters:
	Name: CreateSalesLead (select from dropdown)
	Left operand: ${LeadsCompletedFlag}
	Right operand: Is True (select from dropdown)
	Turn **Negate** option as **ON**.
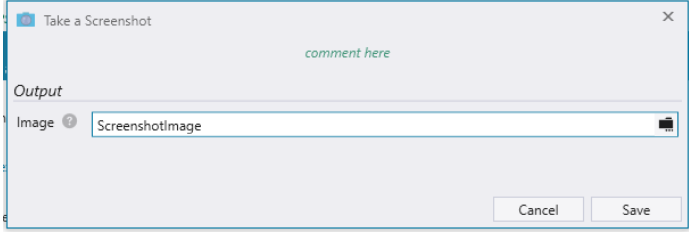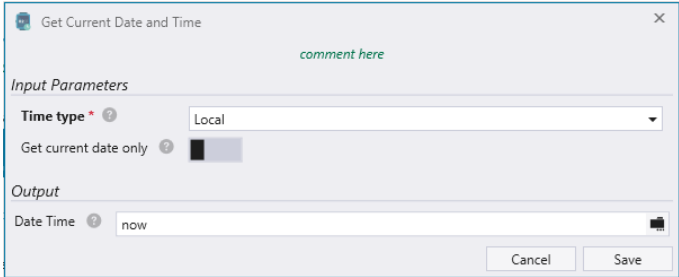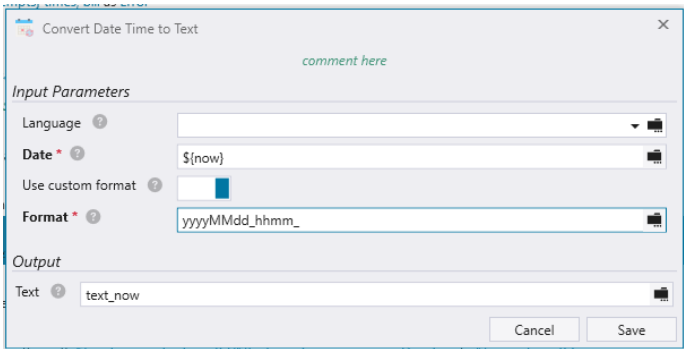
c. Add command of *Assign variable* with Input parameters:
Variable: ${Error_Lead_flag}
Value: True

d. Add command *Recover From Error*.

12	Add *Else condition* to the current If condition at **line 242**. Add the following commands in **Else condition**: a. Add command *Log Message* with message '*Running Create Claim subroutine of exception handler*'.

    b. Add command *Run Subroutine If* with parameters:
Name: CreateClaim
Left operand: ${ClaimsCompletedFlag}
Right operand: Is True
Turn *Negate* option as ON.

    c. Add command of *Assign variable* with Input parameters:
Variable: ${Error_Claim_flag}
Value: True
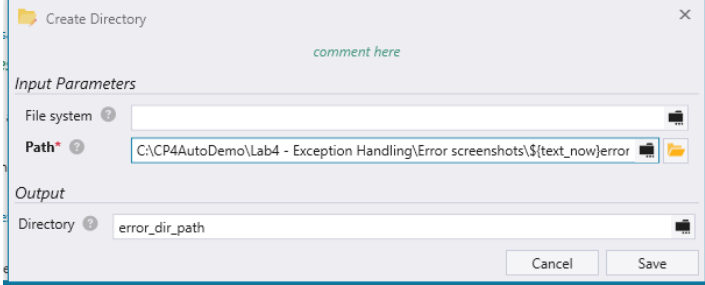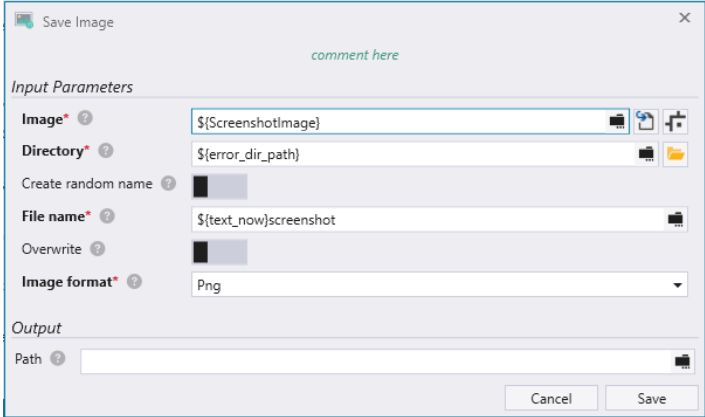
    d. Add command ***Recover From Error***.

6.5.2.4 Logic for cases where the transaction has been retried for max-retries

In this scenario, we will collect as much evidence as possible to aid the business to identify the unknown exception cause and maybe even prevent this in future.

#	Description
1	Add **Else condition** at *line no 248*. In the ***2ⁿᵈ Else condition***, Add ***Log Message*** command with **Message**: 'Attempted main routine ${max_attempts} times, but it keeps raising exceptions.'
2	Add ***Log Message*** command with **Message**: 'System Exception - ${rpa:errorMessage}'. With **Type**: Error
3	Add command ***Take a Screenshot*** and add **Image Output** : ScreenshotImage

	Take a Screenshot dialog with comment here, Output Image field showing "ScreenshotImage", Cancel and Save buttons
4	Add command **Get Current Date and Time** Select **Time type**: Local **Output**: **Date Time** – now Get Current Date and Time dialog with Input Parameters: Time type* = Local, Get current date only toggle; Output: Date Time = now; Cancel and Save buttons
5	Add command of **Convert Date Time to Text** Add **Input Parameters**: **Date**: ${now} Toggle **custom format** **Format**: yyyyMMdd_hhmm_ **Output – Text** : text_now Convert Date Time to Text dialog with Input Parameters: Language, Date* = ${now}, Use custom format toggle, Format* = yyyyMMdd_hhmm_; Output: Text = text_now; Cancel and Save buttons
7	Add command **Create Directory** In **Input parameters,** in **Path** - C:\CP4AutoDemo\Lab4 - Exception Handling\Error screenshots\${text_now}error **Output – Directory**- error_dir_path

8	Add command **Save Image**. **Input parameters**, **Image**: ${ScreenshotImage} **Directory**: error_dir_path **File name**: ${text_now}screenshot **Image format**: Png 	
9	After this step, there will be End If.	

Final subroutine script appears as follows:

ErrorHandlerSystemException

134 *System exception which is not to be retried.*

135 🖼 **Log Message**
Logs Entered System exception handler as Information

136 ⟳ **If**
If ${rpa:errorMessage} contains System Exception - The record is not getting su..., then

137 🖼 **Log Message**
Logs System Exception - The record of ${first_name}... as Information

138 📷 **Take a Screenshot**
Take a screenshot of the current screen, assigning Image to ${ScreenshotImage}

139 🖼 **Log Message**
Logs --... as Information
Log Error Message

140 ⚠ **Recover From Error**
Recover from an error an resume the current script session.

141 ⤵ **Else**
Define a command block executed when the If condition is not met.

142 ⟳ **If**
If ${i} is Less than or equal to ${max_attempts}, then

143 🖼 **Log Message**
Logs We are in else condition with less than max-ret... as Information

144 🖼 **Log Message**
Logs --... as Information
Log Error Message

145 ▦ **Increment Variable**
Increment ${i}

146 🖥 **Close Browser**
Close all browser instances by the name web01

147 ▭ **Start Browser**
Start web01 in Google Chrome, assigning Browser to ${Browser1}

148 ▦ **Navigate**
Navigate to https://jk-automation-container.12kvw4kay3ug.us-...

149 🖼 **Log Message**
Logs Running Login subroutine of exception handler as Information

150 🐿 **Run Subroutine**

151 ⟳ **If**
If ${rpa:errorSubName} contains Lead, then

152 🖼 **Log Message**
Logs Running Sales lead subroutine of exception hand... as Information

153 🐿 **Run Subroutine If**
Executes the routine CreateSalesLead if ${LeadsCompletedFlag} is false

154 ◆ **Assign Variable**
Assigns True to the variable ${Error_Lead_flag}

155 ⚠ **Recover From Error**
Recover from an error an resume the current script session.

156 ⤵ **Else**
Define a command block executed when the If condition is not met.

157 🖼 **Log Message**
Logs Running Create Claim subroutine of exception ha... as Information

158 🐿 **Run Subroutine If**
Executes the routine CreateClaim if ${ClaimsCompletedFlag} is false

159 ◆ **Assign Variable**
Assigns True to the variable ${Error_Claim_Flag}

160 ⚠ **Recover From Error**
Recover from an error an resume the current script session.

161 ↩ **End If**
End a If command block.

162 ⤵ **Else**
Define a command block executed when the If condition is not met.

163 *Add here any commands to help you collect as many evidence as possible to identify the unknown exception cause*

164 🖼 **Log Message**
Logs Attempted main routine ${max_attempts} times, b... as Error

165

166 🖼 **Log Message**
Logs System Exception - ${rpa:errorMessage} as Error

167 *The following block only makes sense if bot is automating GUIs.*

168 📷 **Take a Screenshot**
Take a screenshot of the current screen, assigning Image to ${ScreenshotImage}

169 🔲 **Get Current Date and Time**
Get current Local date and time, assigning Date Time to ${now}

170 🗓 **Convert Date Time to Text**
Convert a date time to text, assigning Text to ${text_now}

171 📁 **Get System Folder Path**
Get the path of the system's special folder My Documents, assigning Path to ${mydocuments_dir_path}
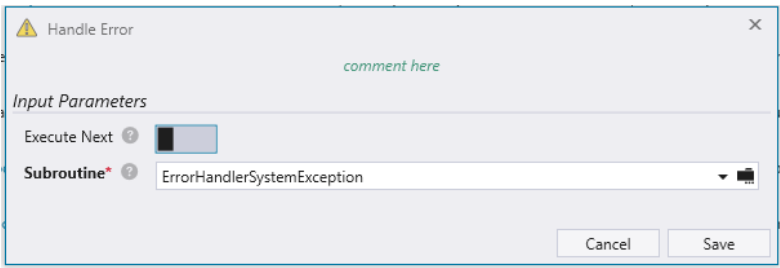
6.5.3   System Exception Try Once

We will be conducting exercise for the **System Exception Try Once**.

Scenario

When we fill in sales lead data in JK Automation website, we click on Submit button to submit a lead. At this stage, there is a possibility that the website becomes unresponsive and when trying to open again, we will not know if that sales lead was submitted or not. Therefore, we will not try to submit the sales lead again.

Please follow the instructions:

> **6.5.3.1**   Open subroutine **CreateSalesLead** from subroutines panel on the right. In this subroutine, we are submitting the sales lead input for each case. Before clicking on submit button, we will add **error handler** to take care of any issue which may happen after clicking the said button.

#	Description
1	Open subroutine **InsertLeadData**. At **line no 85** in subroutine before clicking **Submit** button, add command of **Handle Error**.
2	Choose subroutine 'ErrorHandlerSystemException' from dropdown and click on **Save** button.

> **6.5.3.2**   Open subroutine **CheckStatusOfLeadData**.

In this subroutine, we are checking if the lead data case which was submitted is appearing in the submitted sales lead table. We will perform this check by **enabling a flag if sales lead data is submitted successfully**.
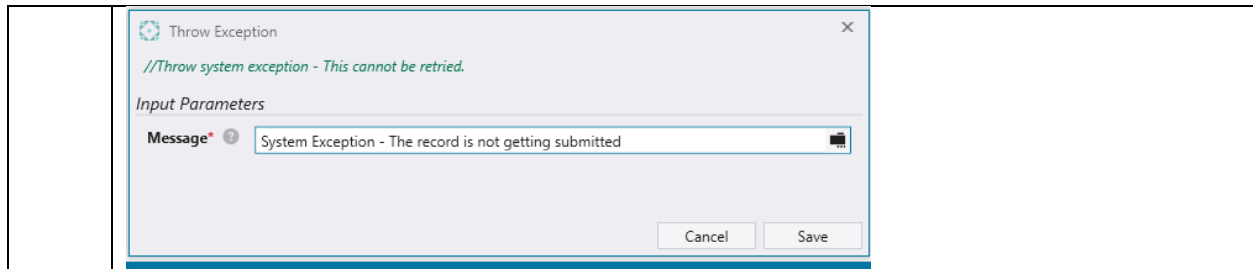
#	Description
1	At *line no 127*, add *Throw exception* command in **If condition**.
2	Add **Message** of 'System Exception - The record is not getting submitted' and click on *Save* button.

6.5.3.3 Add Error Handler in stage of inserting Claims Data

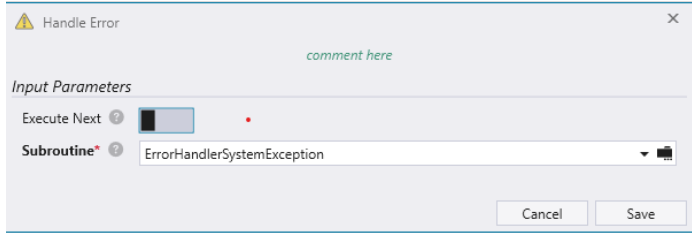| #	Description
1	Open subroutine *InsertClaimsData*. At *line no 185*, before clicking **Submit button**, add command of *Handle Error*.
2	Choose subroutine 'ErrorHandlerSystemException' from dropdown and click on *Save* button.

6.5.3.4 Add Throw exception at stage of checking status of Claims Data

#	Description
1	Open subroutine *CheckStatusOfClaimsData*. At *line no 200* in subroutine, add *Throw exception* command at If condition.
2	Add **Message** of 'System Exception - The record is not getting submitted' and click on **Save** button.

6.5.4 Add error handling to Create Sales Lead subroutine(to cover General exceptions)

Open subroutine CreateSalesLead.

#	Description
1	In the first line of subroutine, add command Handle Error. Select subroutine ErrorHandlingSystemException from dropdown.

6.5.5 Testing System Exception

Since system exceptions are exceptions which are caused due to unexpected responses of application, which is the application turns unresponsive due to latency, crash etc, we will be testing out this exception handling by failing the application manually.

Test scenario : Start to run from studio, **'Start without Debugging'** once the bot has completed entering sales lead information of the 2nd person ie Dave, we will close the browser manually.

Result: The bot will open the chrome window and JK automation website again and start from after where it failed earlier.
To check 3 retries part, we can fail for three times by closing manually again and again, after which the bot will not try to execute again. It will end in failure of process.

6.5.6   Best Practice

> **Best Practice**:
> - In case of any exception, the type and detail of exception to be checked by the exception handling routine before proceeding with performing any action.
> - *System exception* can be retried, prescribed upto 3 times.
> - System exception to also be checked with *previous case's exception* details to confirm that the problem is not appearing due to application's unresponsiveness or unavailability.
> - Where possible **retry system exception** within the process. This may require special navigation or even a restart of the system.
> - You will need to *cleanup* the application and navigate ready to perform first action again.

6.6   Ignoring Error and resume from next step

In some scenarios, we may decide to *ignore the error and resume operations*. We may place checks in our process to check for probable scenarios.

For eg, when an application is already attached but you are not sure at the next stage of process whether it is attached or not. You would like to be sure by attaching again, if it is not already attached, as otherwise the next step of commands may fail.

So, you will try attaching, it will throw an error which can be captured and resumed normally by ignoring as the application would already be launched, you can proceed with the next steps safely after that.

6.6.1   Exercise

In our exercise scenario, we open a word document, opening a new file. At some stage of running the process, we try to open file again, in case we need to do that to be sure.

However, since that New button is not present, as the file is already opened, it will throw an error stating it cannot find the element on screen.

In such a scenario, we will utilize **Ignore Error** option in *error handler*.

The steps of configuring the exercise are as follows:

#	Description
1	Import script **AttachError.wal** and save it as **HandlingExpectedIgnore.wal**.
2	Add the *Handle error* command at *line no 17* with *Execute Next* option selected.
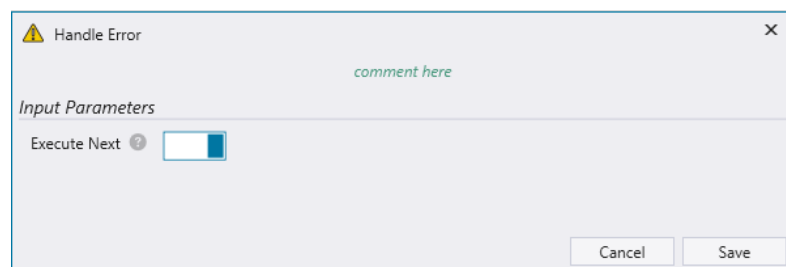
6.6.2 Testing

Step 1	Run the script AttachError.wal
Output	Finishes with error
Step 2	Run script HandlingExpectedIgnore.wal
Output	A log message will be generated saying it has handled the error and the bot will proceed with executing the next step.

Congratulations, you have successfully completed this lab!!!