Lab Guide

Bot Design Lab Guide

Nigel T. Crowther
ncrowther@uk.ibm.com

Hands-on Lab

Version 1.0 for General Availability

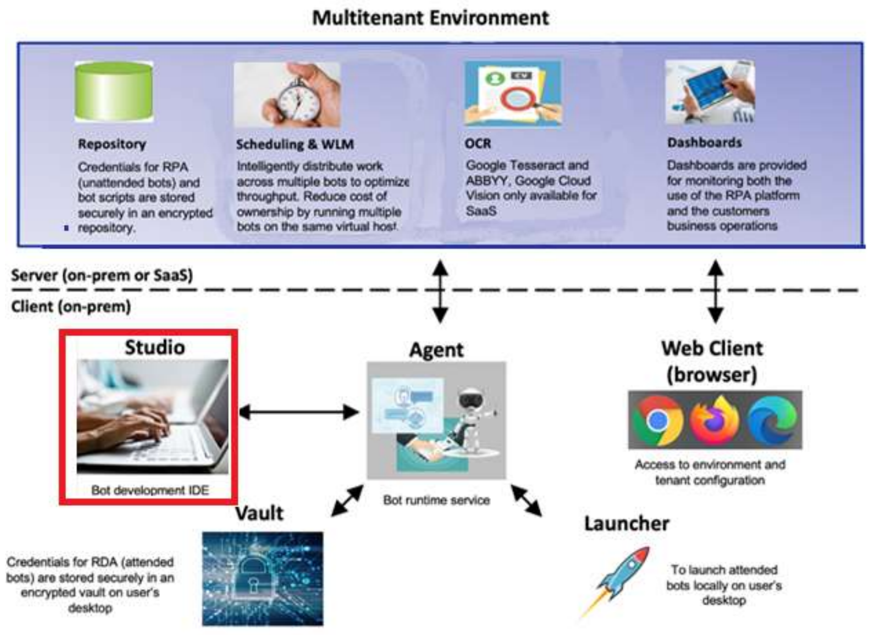
Table of Contents

1 Introduction

This lab will examine bot design and best practices.

We first design a bot and then build it. Each lab will enhance its successor until we have a bot that implements all best practices.

For the context of this lab see the highlighted area below.


Multitenant Environment
Repository
Credentials for RPA (unattended bots) and bot scripts are stored securely in an encrypted repository.
Scheduling & WLM
Intelligently distribute work across multiple bots to optimize throughput. Reduce cost of ownership by running multiple bots on the same virtual host.
OCR
Google Tesseract and ABBYY, Google Cloud Vision only available for SaaS
Dashboards
Dashboards are provided for monitoring both the use of the RPA platform and the customers business operations
Server (on-prem or SaaS)
Client (on-prem)
Studio
Bot development IDE
Agent
Bot runtime service
Web Client (browser)
Access to environment and tenant configuration
Vault
Credentials for RDA (attended bots) are stored securely in an encrypted vault on user's desktop
Launcher
To launch attended bots locally on user's desktop

1.1 What is good design?

The difference between a well-designed bot and a bad one is maintainability. A well-designed bot is understandable by others. A bad bot design is the opposite. It is hard to understand by others and maybe even by the author. This makes it hard to maintain and results in a short shelf life.

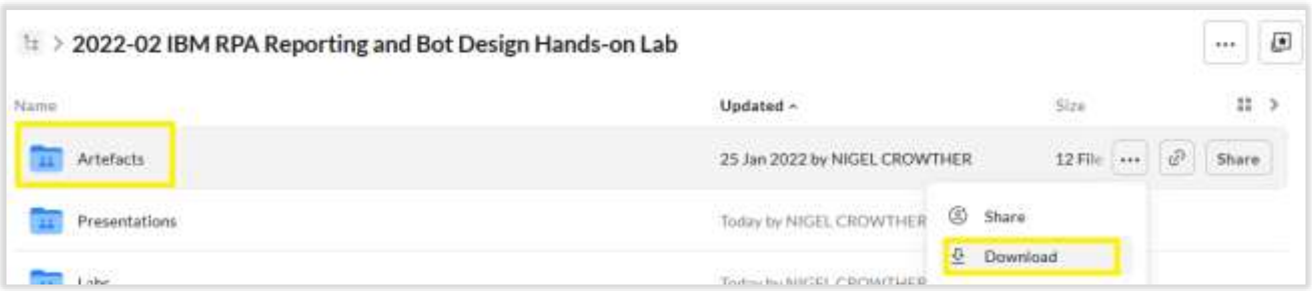
1.2 Prerequisites

RPA 21.0.2 or later installed with Studio.

IMPORTANT NOTE FOR SKYTAP USERS: Within your VM, open Firefox, navigate to the lab folder:

https://ibm.ent.box.com/folder/154727284550?v=2022-02RPAReportingBotDesign

Download the *Artefacts* folder as a zip. See below:

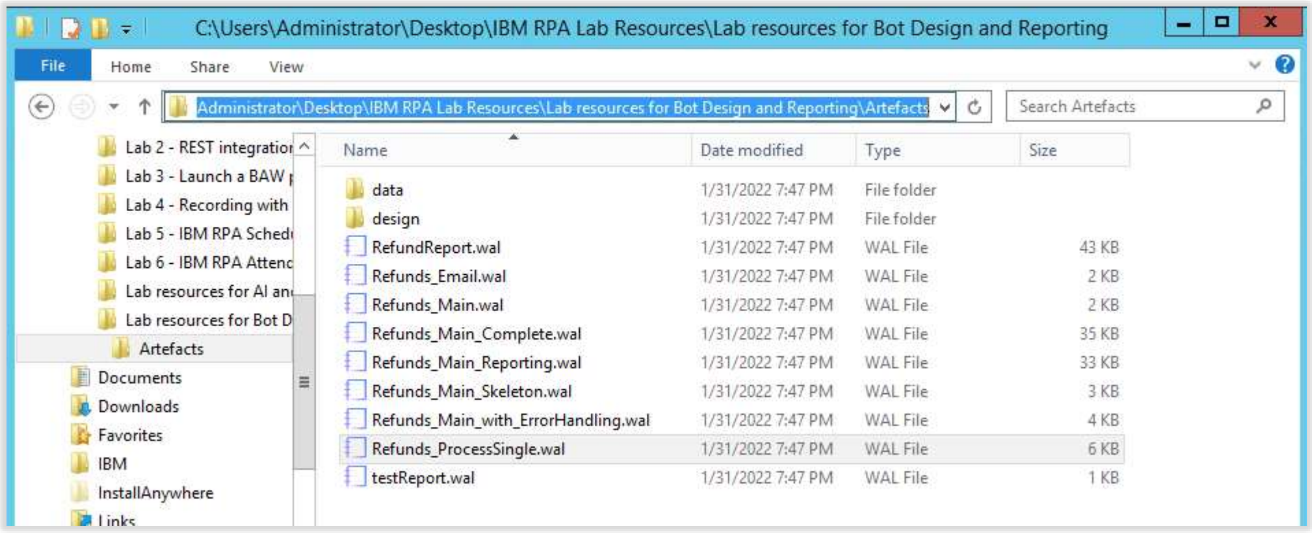

2022-02 IBM RPA Reporting and Bot Design Hands-on Lab

Name	Updated	Size
Artefacts	25 Jan 2022 by NIGEL CROWTHER	12 File
Presentations	Today by NIGEL CROWTHER	

The zip must be extracted to the following location:

```
C:\Users\Administrator\Desktop\IBM RPA Lab Resources\Lab resources for
Bot Design and Reporting
```

This location is important! If you copy to a different folder, you will experience problems. You should see the contents downloaded to the following folder:


C:\Users\Administrator\Desktop\IBM RPA Lab Resources\Lab resources for Bot Design and Reporting

Administrator\Desktop\IBM RPA Lab Resources\Lab resources for Bot Design and Reporting\Artefacts

Name	Date modified	Type	Size
data	1/31/2022 7:47 PM	File folder	
design	1/31/2022 7:47 PM	File folder	
RefundReport.wal	1/31/2022 7:47 PM	WAL File	43 KB
Refunds_Email.wal	1/31/2022 7:47 PM	WAL File	2 KB
Refunds_Main.wal	1/31/2022 7:47 PM	WAL File	2 KB
Refunds_Main_Complete.wal	1/31/2022 7:47 PM	WAL File	35 KB
Refunds_Main_Reporting.wal	1/31/2022 7:47 PM	WAL File	33 KB
Refunds_Main_Skeleton.wal	1/31/2022 7:47 PM	WAL File	3 KB
Refunds_Main_with_ErrorHandling.wal	1/31/2022 7:47 PM	WAL File	4 KB
Refunds_ProcessSingle.wal	1/31/2022 7:47 PM	WAL File	6 KB
testReport.wal	1/31/2022 7:47 PM	WAL File	1 KB

1.3 Scenario Description

You are an RPA developer working for a successful travel company. You have been approached by the customer service team. The team is working night and day cancelling customers after a pandemic. The team keep a list of customers in a spreadsheet. They manually refund each customer by entering details into a web site. Once the customer has been refunded, they update the spreadsheet and send a remittance to the customer. There is no API into the refund web site, and it randomly fails. Your task is to design and build a production-ready bot that refunds customers.

2 Best Practices

In this lab we will build a bot using the best practices found in the IBM RPA guidelines:

https://www.ibm.com/docs/en/rpa/21.0?topic=development-script-guidelines

2.1 Best Practice – Start with a Design

IBM RPA Studio has a built-in **workflow editor** to design and create bots. With this tool you can create diagrams that help structure your bot and share your understanding to team members.

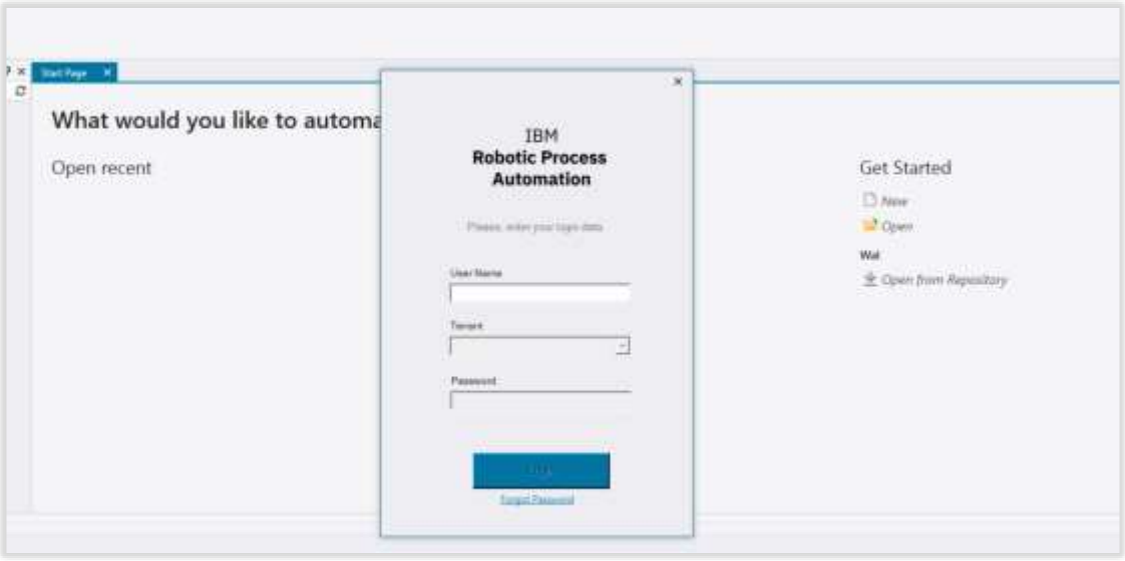
Note that the **workflow editor** can be used to build executable workflows. We will *not* be using this feature in this lab

Start RPA Studio

On your desktop find the IBM RPA Studio icon and launch it.

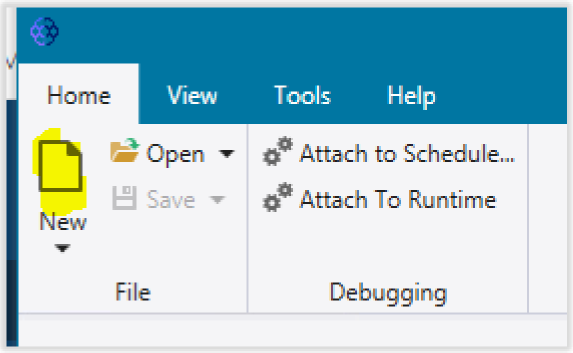
Log In

Login to RPA Studio with your credentials.
Tip: If you are using SkyTap, the credentials are:
User: *admin@ibmdba.com*
Password: *passw0rd*

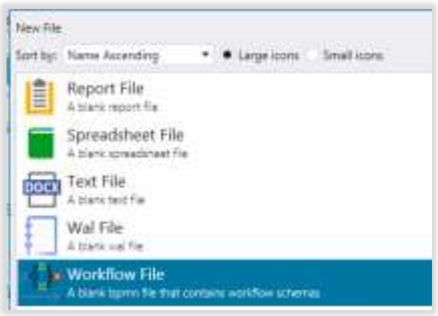


Create a Workflow

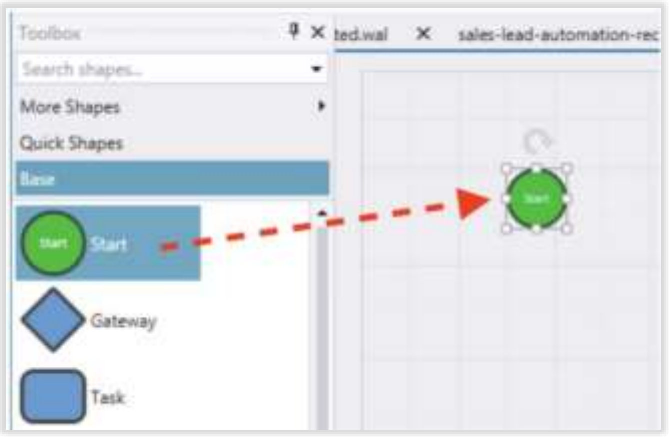
Once in RPA studio, select the *New* icon.
Tip: *Select the upper part of the icon inside the blank page.* See below:



Select *Workflow File* and press *Create*:



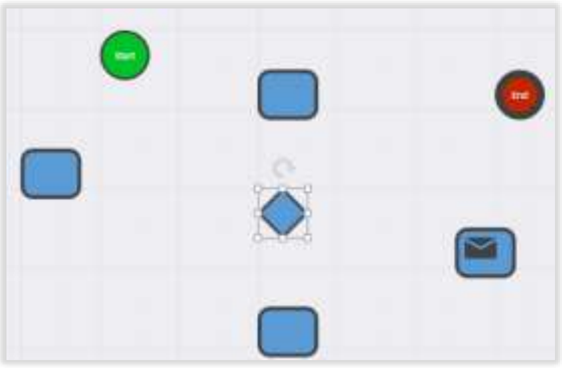
Drag and drop a *Start* element to your empty canvas.



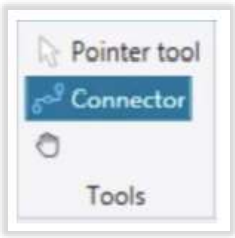
Add the following components to the canvas:

- Three *Empty Task* elements
- One *Send Task*
- One *Gateway*
- One *End*

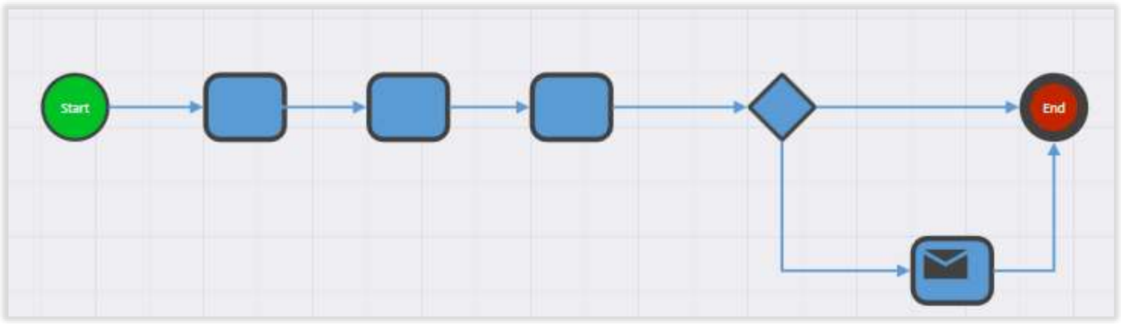
You should now have something like this:



Click the Connector from the top toolbar to select the connector tool.



Reorder the elements and then connect them to achieve the flow presented below.

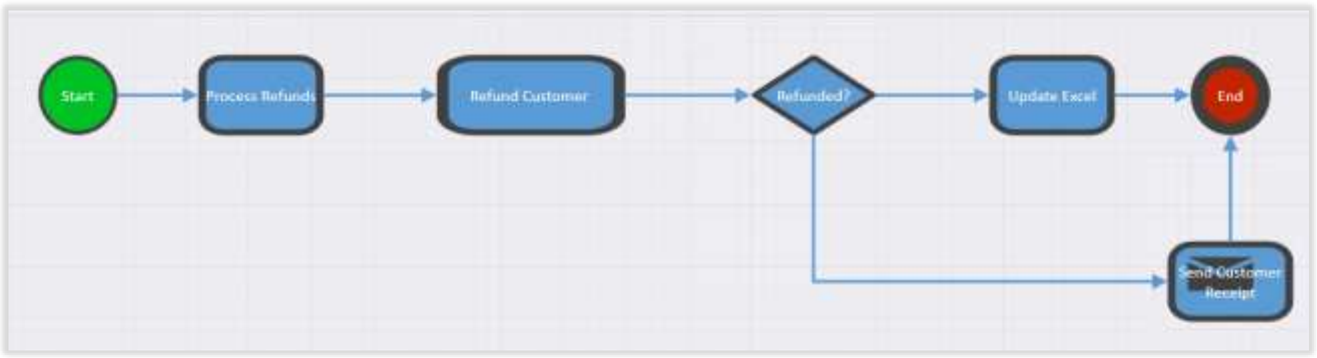


Deselect the Connector tool by selecting the *Pointer* tool just above the *Connector* in the *Tools* section of the top toolbar.

Next, we want to name our elements. For each element, right-click and select *Edit text.* Type the following names

- Process Refunds
- Refund Customer
- Refunded?
- Update Excel
- Send Customer Receipt

Once done, you should have the design below:



Now name your workflow. Click *Save* and then save as *refundBotDesign.bpmn*

In the next section we will build the bot based on this design.

Lesson learnt

Before you dive into build, consider creating a design first. This helps you understand the problem, structure your bot, and communicate your understanding to the stakeholders.

2.2 Best Practice – Build an Initial Framework

When building a bot, it is good practice to start with a framework. By framework we mean the subroutines, conditional branching, and sub-scripts within the bot. Once this structure is in place, you can build out the bot code.

Start RPA Studio

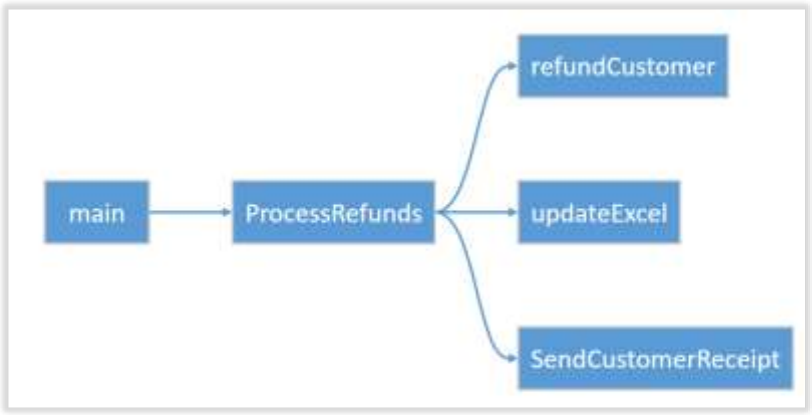
If not already started, launch IBM RPA Studio.

Import skeleton code

Open the bot script:

```
[bot_design_lab]\Artefacts\Refunds_Main_skeleton.wal
```

Within RPA Studio, press the call graph tab. You should see the following:



Compare the call graph to the BPMN bot design created in the previous exercise. The BPMN tasks have become subroutines. The decision point is implemented as an *if/then* condition in the *processRefunds* subroutine on line 40. Verify this in RPA Studio.

Now hit *Ctrl-F5* to run the bot. The log output should be:

```
[Info] Starting refunds.  Read excel file containing customers to be refunded
[Info] Excel Path …\refunds.xlsx
[Info] Refund individual customer using refund website
[Info] Update excel with the result of the refund operation
[Info] If refund was successful, send the customer a refund receipt
```

Note: If the script fails, make sure the project path is set correctly. See prerequisites at the start of this lab.

Lesson learnt

Start by building a framework to get the structure right. This helps readability and maintainability of your bot.

2.3 Best Practice – Use Error Handlers

Bots should have error handling. There are many reasons why a bot can fail, and when problems occur, administrators need to resolve them quickly. In this lab, we touch upon error handling practices that you should consider.

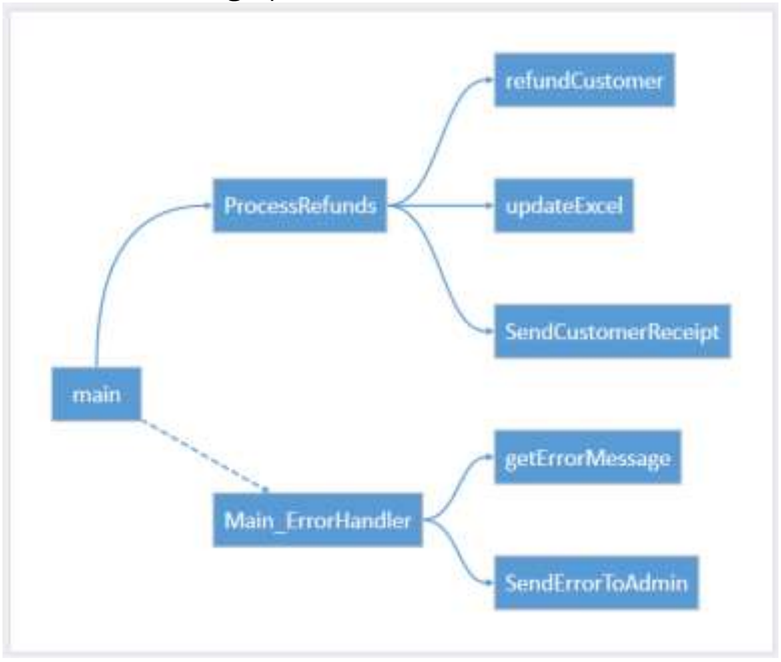
Tip: We use just one error handler in this example, but you could have an error handler for each subroutine to handle errors differently in each part of your script.

Examine Error Handler Routine

Open the following script in RPA Studio:

```
[bot_design_lab]\Refunds_Main_with_ErrorHandling.wal
```

Examine the call graph. Note the error handler:



Double-click the routines *Main_ErrorHandler* and *getErrorMessage* and *SendErrorToAdmin* to view the source code.

Test Error Handling

Test the error handler. Hit *Ctrl-F5* to run the bot without debug.

IMPORTANT: Do not run with debug (*F5*). If you do, the exception handler will not run.

The log output should be as follows:

```
Windows Host: IBMBAW
Error Message: Error reading excel
Subroutine: ProcessRefunds
Line Number: 72
[Info] ------------------------------------------------------------------------
```

```
Error executing command at line 31: failTest --message "Error reading excel"
```

The error handler has caught the exception and created a message containing:

- Machine name
- Bot name
- Error message
- Line number of the error

In a real-world implementation, this message could be emailed to an administrator to resolve.

Task: Using the error information, fix the problem.
Hint: The project path for the excel file does not exist on this machine. View the source of the error and work backwards to fix the path.

Lesson learnt

Adding error handling to your bot saves time finding problems, both in development and in production.

2.4 Best Practice – Separate Activities into Separate Scripts

If your bot performs distinct activities, it is good practice to split the bot into smaller scripts. This facilitates delegation of development between several developers and also allows for script re-use. Lastly, each script will be fully tested before it is integrated with the main script, leading to a more resilient and maintainable bot.

In our Refund bot, we are going to separate the web site interaction into a separate script. This script will be invoked from the main script.

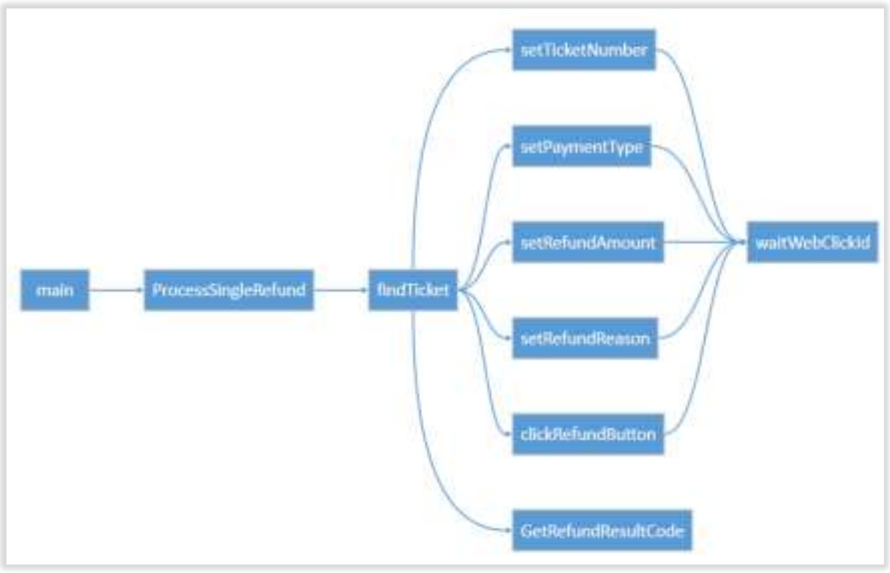
Open the Web Interaction Script

Open the following script in RPA Studio:

```
[bot_design_lab]\Artefacts\Refunds_ProcessSingle.wal
```

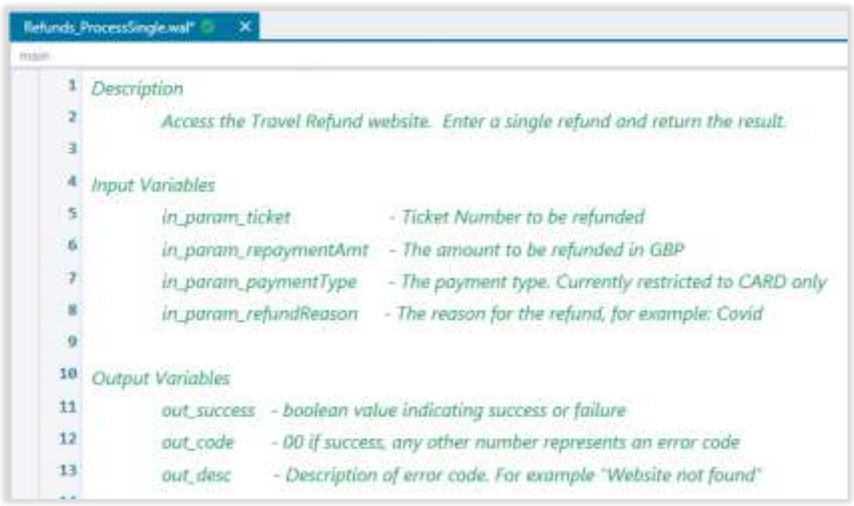
Examine Code

Press the *Call Graph* tab to view the bot structure. The script enters refund information into a web site. It then clicks the *Refund* button and returns the result. The bot uses the function **webWaitElement** which uses a time out rather than a delay to wait for HTML elements. Compare this to using the **Delay** function, which will always pause even if the element is available immediately.



Examine Script Parameters

Select the *Designer* tab. Examine the input and output variables**:**



```
Description
        Access the Travel Refund website.  Enter a single refund and return the result.

Input Variables
        in_param_ticket             - Ticket Number to be refunded
        in_param_repaymentAmt       - The amount to be refunded in GBP
        in_param_paymentType        - The payment type. Currently restricted to CARD only
        in_param_refundReason       - The reason for the refund, for example: Covid

Output Variables
        out_success   - boolean value indicating success or failure
        out_code      - 00 if success, any other number represents an error code
        out_desc      - Description of error code. For example "Website not found"
```

The script has four input parameters. These are:

In_param_ticket – the customer ticket number
In_param_repaymentAmt – the amount to refund
In_param_payment_type – the payment type. Only type *Card* is accepted.
In_param_refundReason – the reason for the refund

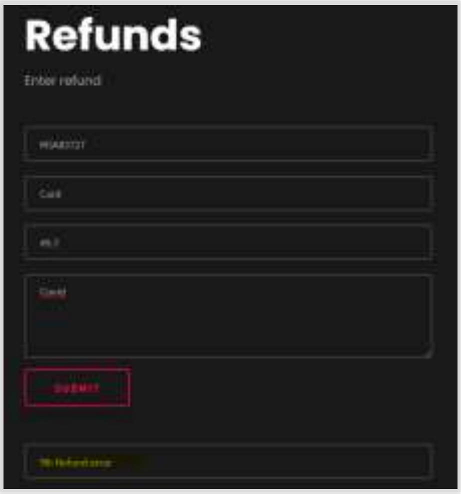
The values of the input parameters are defined with dummy test data. This allows us to test the script. When you come to call this script from another script, these parameters are overridden.

The script contains three output parameters. These parameters are generic and not specific to refunds. These are:

Out_success – a flag indicating returning whether the script ran successfully
Out_code – A number between **00** and **99**. **00** indicates success, any other number is an error
Out_desc – The description of the error code

Run the script

Run the bot with *Ctrl+F5*. You should see the following in the Refund web site:


Refunds

Enter refund

[illegible]

Card

[illegible]

Covid

SUBMIT

[illegible]

Lesson learnt

Separating your bot into smaller scripts allows for script re-use and independent testing of scripts.

2.5 Best Practice – Recover from non-fatal errors

The error code **98** was returned when running the refund bot above. This error simulates the real world where web sites don't always work as expected. Error code **98** is a recoverable error - it is an intermittent problem with the web site. It is good practice to make a distinction between recoverable errors and non-recoverable errors such as a missing file. Recoverable errors should be retried, whereas non-recoverable errors should throw an exception and stop.

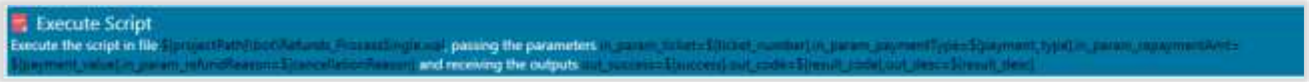
In this section we will show how to handle recoverable exceptions and retry them.

Examine the completed script

Open the following script in RPA Studio:

```
[bot_design_lab]\Artefacts\Refunds_Main_Complete.wal
```

This script contains the complete implementation of the refunds bot. Take a while to examine the implementation. Note how each row in the excel file is read and then passed to `Refunds_ProcessingSingle.wal` using the *ExecuteScript* command. See below:

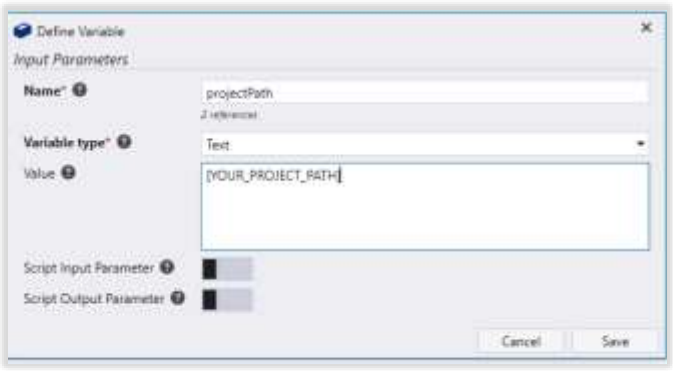

Execute Script

Execute the script in file ${projectPath}\bot\Refunds_ProcessingSingle.wal, passing the parameters in_param_ticket=${ticket_number}|in_param_paymentType=${payment_type}|in_param_repaymentAmt=${payment_value}|in_param_refundReason=${cancellationReason} and receiving the outputs out_success=${success}|out_code=${result_code}|out_desc=${result_desc}

The response is written back to the excel file. If the customer refund status code is **98** or **99** then the bot will retry the customer on a subsequent run.

Set project path

In RPA Studio, edit the *projectPath* variable to match your project folder (if not already):


Define Variable

Input Parameters

Name*: projectPath

2 references

Variable type*: Text

Value: [YOUR_PROJECT_PATH]

Script Input Parameter

Script Output Parameter

Hit *Ctrl-F5* to run the bot. You should see each customer is read, the web site starts, and the customer refund entered to the website. The script returns with a status code and excel is updated with the result. All eleven rows in the spreadsheet are processed. On completion, the log output should look like this:

```
[Info] Robot IBM RPA Refunds Robot ran at host NIGELSP52
Processed File: ..\refunds.xlsx
Total items processed - 10
Refunds Issued - 5
Refunds not issued - 5
Robot took 1 Minute(s) 41 Second(s) 219 Millisecond(s)
```

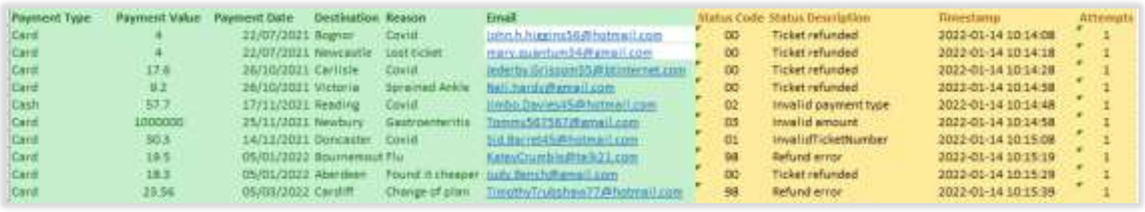
Examine Excel Data

Open the excel file containing the refunds:

```
[PROJECT_FOLDER]/Artefacts/Data/refunds.xls
```

Tip: You can open Excel files directly in RPA Studio.

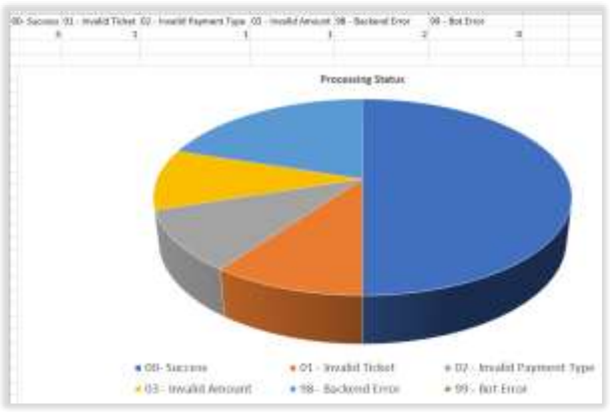
You should see the following:



Payment Type	Payment Value	Payment Date	Destination	Reason	Email	Status Code	Status Description	Timestamp	Attempts
Card	4	22/07/2021	Bognor	Covid	john.h.higgins56@hotmail.com	00	Ticket refunded	2022-01-14 10:14:08	1
Card	4	22/07/2021	Newcastle	Lost ticket	mary.quantum34@gmail.com	00	Ticket refunded	2022-01-14 10:14:18	1
Card	17.6	26/10/2021	Carlisle	Covid	jederby.Grissom35@btinternet.com	00	Ticket refunded	2022-01-14 10:14:28	1
Card	9.2	26/10/2021	Victoria	Sprained Ankle	Nell.hardy@gmail.com	00	Ticket refunded	2022-01-14 10:14:38	1
Cash	57.7	17/11/2021	Reading	Covid	Jimbo.Davies45@hotmail.com	02	Invalid payment type	2022-01-14 10:14:48	1
Card	1000000	25/11/2021	Newbury	Gastroenteritis	Tommy567567@gmail.com	03	Invalid amount	2022-01-14 10:14:58	1
Card	50.3	14/12/2021	Doncaster	Covid	Sid.Barrett45@hotmail.com	01	InvalidTicketNumber	2022-01-14 10:15:08	1
Card	19.5	05/01/2022	Bournemout	Flu	KateyCrumble@talk21.com	98	Refund error	2022-01-14 10:15:19	1
Card	18.3	05/01/2022	Aberdeen	Found it cheaper	Judy.Bench@gmail.com	00	Ticket refunded	2022-01-14 10:15:29	1
Card	23.56	05/03/2022	Cardiff	Change of plan	TimothyTrubshaw77@hotmail.com	98	Refund error	2022-01-14 10:15:39	1

Note: Your response data may be different as the website generates random errors.

Still in Excel, click on the *Processing Status* tab. You should see the following chart:



00- Success	01 - Invalid Ticket	02 - Invalid Payment Type	03 - Invalid Amount	98 - Backend Error	99 - Bot Error
5	1	1	1	2	0

The chart has two Backend errors (both error code 98). If you run the bot again, these rows will be reattempted until they disappear. The bot will *not* retry rows with non-recoverable errors such as *Invalid Payment Type*.

Lesson learnt

Making your bot retry non-fatal errors adds resiliency.

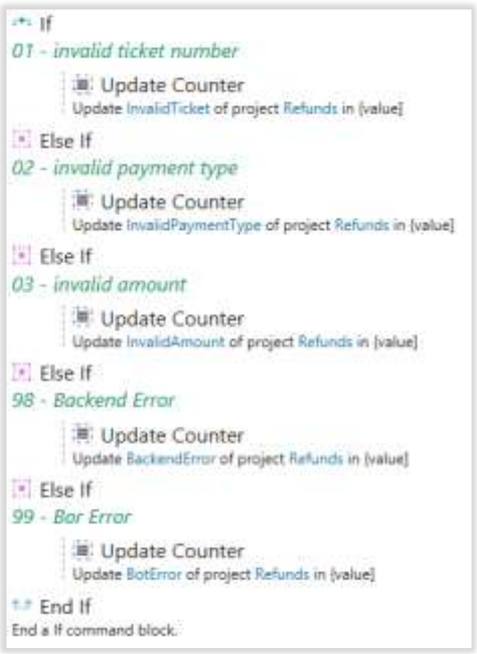
2.6 Best Practice – Use Dashboard Counters

The Dashboard is essential for monitoring the health of your bot. Not only can the Dashboard show status of bots, but it can also show the inner workings of bots in real time. Each time your bot updates its counters they can be displayed in on the dashboard. This allows RPA administrators to monitor the internal state of your bot and react to problems.

Create counters in the Bot
Open the following script in RPA Studio:

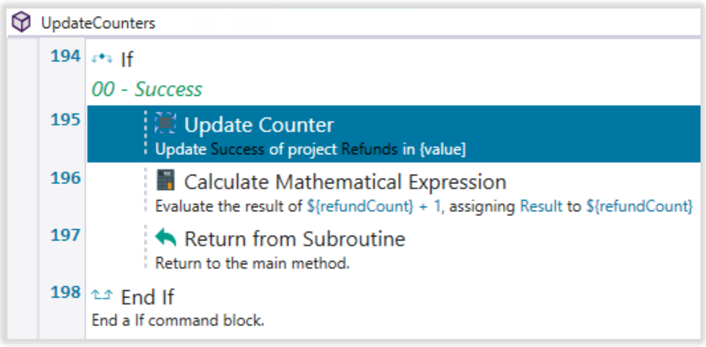
```
[bot_design_lab]\Artefacts\Refunds_Main_Complete.wal
```

Navigate to the *updateCounters* subroutine and comment out the *updateCounter* calls. When done you should see the following:



```
If
01 - invalid ticket number
    Update Counter
    Update InvalidTicket of project Refunds in {value}
Else If
02 - invalid payment type
    Update Counter
    Update InvalidPaymentType of project Refunds in {value}
Else If
03 - invalid amount
    Update Counter
    Update InvalidAmount of project Refunds in {value}
Else If
98 - Backend Error
    Update Counter
    Update BackendError of project Refunds in {value}
Else If
99 - Bot Error
    Update Counter
    Update BotError of project Refunds in {value}
End If
End a If command block.
```

Make sure you uncomment the success counter as well:



```
UpdateCounters
If
00 - Success
    Update Counter
    Update Success of project Refunds in {value}
    Calculate Mathematical Expression
    Evaluate the result of ${refundCount} + 1, assigning Result to ${refundCount}
    Return from Subroutine
    Return to the main method.
End If
End a If command block.
```

This code updates counters in your dashboard so that the refund count can be seen in real time.

There is one crucial step missing. We need to define the counters and the dashboard in the tenant. We will do this in the next guide, Dashboards in IBM RPA. Please refer to this guide to proceed.

Lesson learnt

Adding counters allow administrators to see the internal workings of your bot in real time and react to issues quickly.

3 Advanced Exercise

Examine the best practices identified in the RPA script guidelines:

https://www.ibm.com/docs/en/rpa/21.0?topic=development-script-guidelines

What are the best practices *not* followed in the refund bot? Identify and rectify.

Nicely done! This concludes the lab.