

Lab Guide

Unit Testing Bots Lab Guide

Nigel T. Crowther
ncrowther@uk.ibm.com

Hands-on Lab

Version 1.0 for General Availability





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 of 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.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.



Table of Contents

1 Introduction.....	4
1.1 Prerequisites	4
2 Unit Test Lab	5
2.1 Setup Lab.....	5
2.2 Run the Unit Test Template	5
2.3 Open the Refunds Script.....	7
2.4 Run the bot.....	7
2.5 Examine Code	7
2.6 Examine Script Parameters	8
2.7 Testing the Refund bot.....	9
2.8 Build the Test Cases	9
2.9 Successful Refund Test Case.....	9
2.10 Cash Refund Test Case	9
2.11 Invalid Amount Test Case	9
2.12 Run the bot.....	10
3 Spreadsheet Driven Testing	11
3.1 Why test with spreadsheets	11
3.2 Open the test scenario.....	11
3.3 Open the test script	12
3.4 Examine the script flow	12
3.5 Run the bot script.....	12
4 Conclusion	14

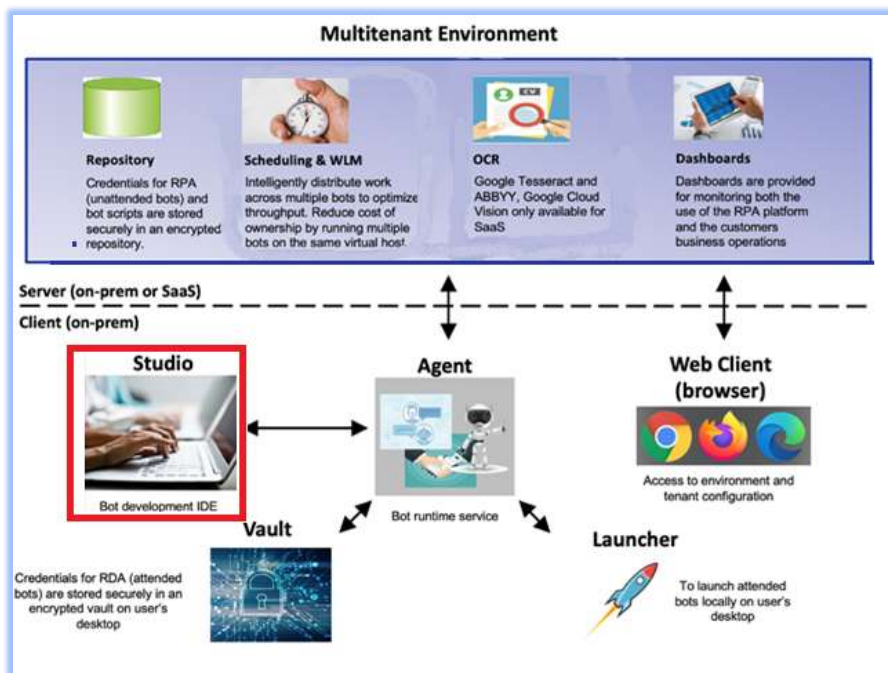


1 Introduction

Unit tests are essential for building production-quality bots. Unit tests give confidence that the bot behaves as expected and prevents regression. If you are automating bot builds, unit tests are a component of the build toolchain.

This lab will examine unit testing by revisiting the customer refunds bot covered in an earlier lab. We will first build 'hard coded' unit tests. Then we will look at extracting tests into a test scenario spreadsheet.

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



1.1 Prerequisites

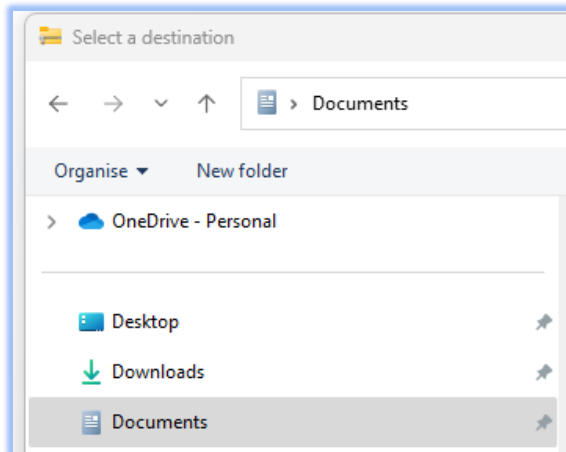
RPA 21.0.2.5 or later.



2 Unit Test Lab

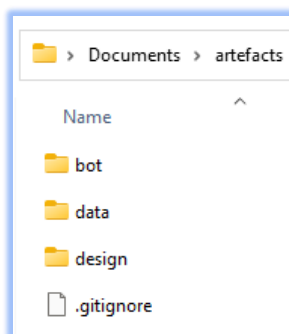
2.1 Setup Lab

Download the *Artefacts.zip*. Extract the zip to the *documents* folder.



This location is important! If you copy to a different folder, the supplied scripts won't run

You should see the contents downloaded:



2.2 Run the Unit Test Template

We will start with a simple unit test to demonstrate principles. Then we will build on these principles to test the refund bot.

Start RPA Studio

If not already started, launch IBM RPA Studio.

Tip. If you are using the IBM lab environment, your credentials are:

Username: admin@ibmdba.com

Password: passw0rd

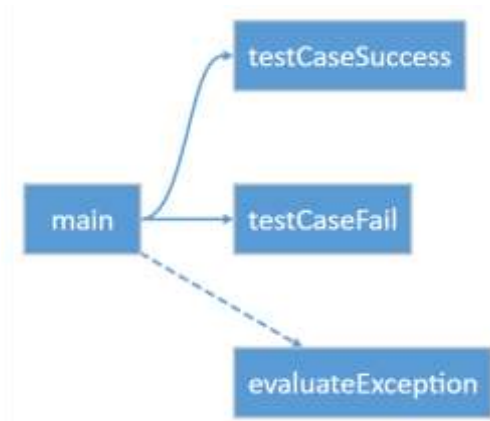


Import Template code

Open the bot script:

```
[Documents]\Artefacts\bot\test_template.wal
```

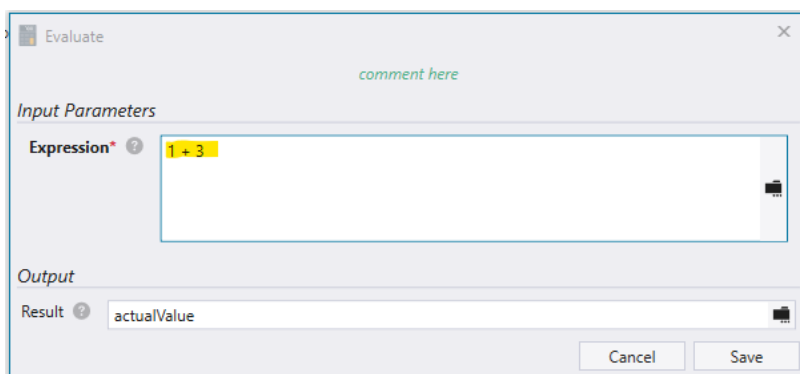
Within RPA Studio, press the call graph tab. You should see the following:



Now hit *Ctrl-F5* to run the bot without debugging (**important**). The output should be:

```
[Info] Starting Unit tests  
[Info] Unit tests completed with 0 failures
```

Now edit the evaluate expression on line 31 so that the test case will fail:



Save and rerun. This time you will see the following test failure:

```
[Info] Starting Unit tests  
  
[Info] Failed testCaseFail, Expected value: 3, actual value: 4.  
Assert failed at line 34: testCaseFail  
  
[Info] Unit tests completed with 1 failure
```

You can see the test captured the error and gave the reason. It also gave additional information to help diagnose the failure.



2.3 Open the Refunds Script

Let's revisit the refund bot. Open the following script in RPA Studio:

`Dopcuments\artefacts\bot\Refunds_ProcessSingle.wal`

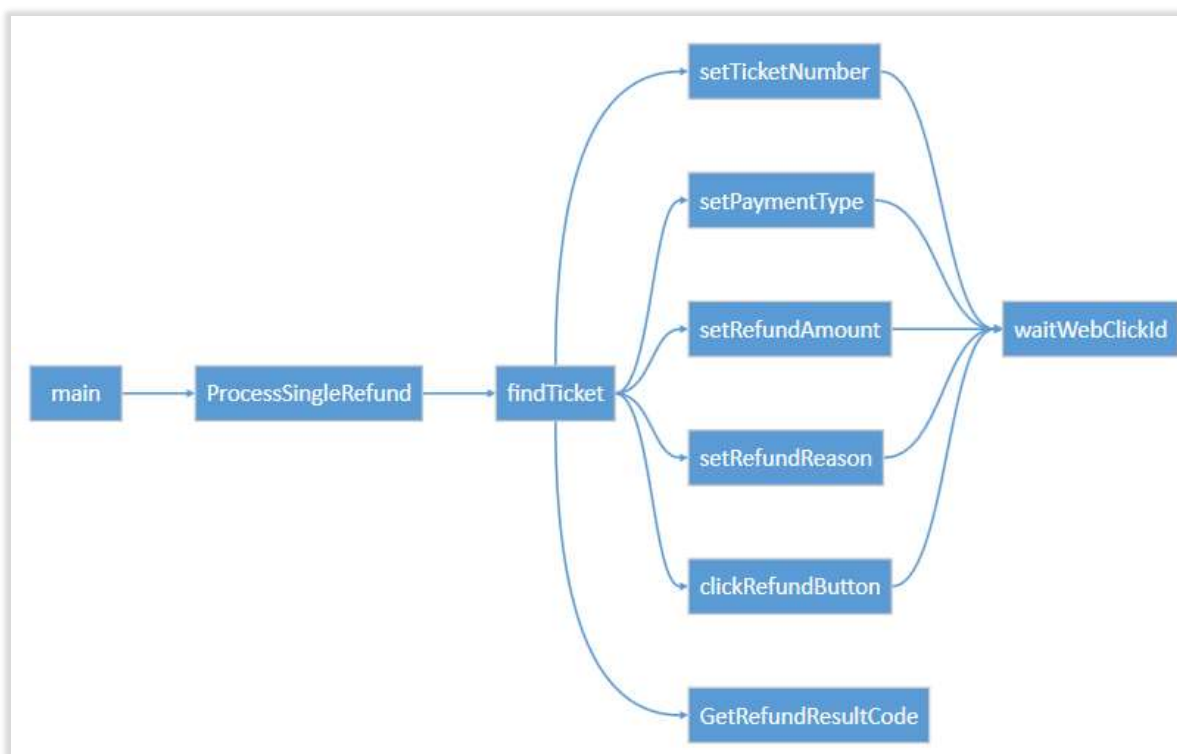
2.4 Run the bot

Press *Ctrl+F5* to run the bot. See how it issues a single refund on a web site, and then returns a status code and description.

2.5 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 result will be on of:

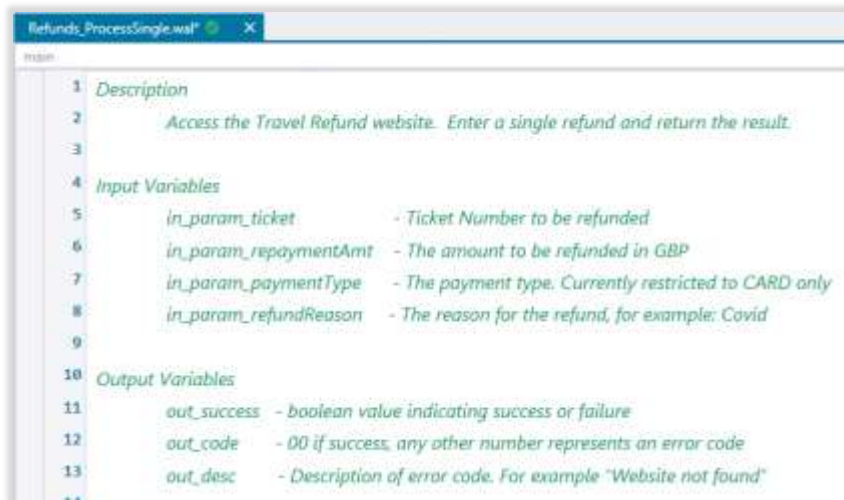
- 00 – A successful refund
- 02 – A Failed refund due to invalid payment type
- 03 – A failed refund due to a too large refund amount
- 98 –Processing error (this is randomly generated)
- 99 –Website error





2.6 Examine Script Parameters

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



The input parameters are:

In_param_ticket – the customer ticket number

In_param_repaymentAmt – the amount to refund. Amounts over 10000 are not valid.

In_param_payment_type – the payment type. Only type *Card* is accepted.

In_param_refundReason – the reason for the refund

The output parameters 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



2.7 Testing the Refund bot

Now let's create a test script for the refund bot.

In RPA Studio, open *test_refunds_start.wal*

2.8 Build the Test Cases

We will now create test cases for the following scenarios:

- 00 – A successful refund
- 02 – A Failed refund due to invalid payment type
- 03 – A failed refund due to a too large refund amount

2.9 Successful Refund Test Case

Let's start with successful refund. Copy & paste the following code directly under the start of the *successfulCardRefund* subroutine (from line 32):

```
setVar --name "${ticket_number}" --value 567567
setVar --name "${payment_type}" --value Card
setVar --name "${payment_value}" --value 87
setVar --name "${expectedValue}" --value 00
```

2.10 Cash Refund Test Case

Now add a test case for cash refunds. This refund should return error code 02. Copy & paste the following code directly under the start of the *cashRefundNotAccepted* subroutine:

```
setVar --name "${ticket_number}" --value 453445
setVar --name "${payment_type}" --value Cash
setVar --name "${payment_value}" --value 8
setVar --name "${expectedValue}" --value 02
```

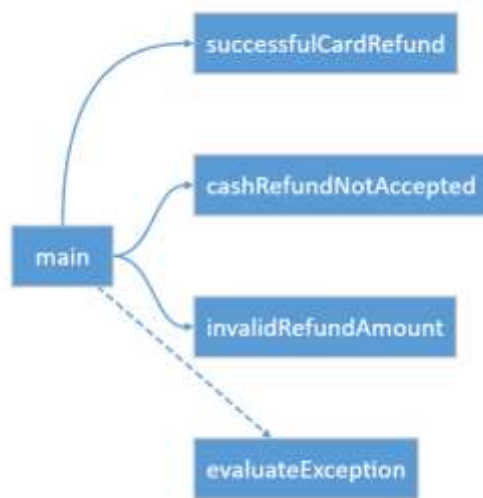
2.11 Invalid Amount Test Case

Now add a test case for an invalid refund amount. In this test case, the refund amount is too large and should return error code 03. Copy & paste the following code directly under the start of the *invalidRefundAmount* subroutine:

```
setVar --name "${ticket_number}" --value 087877
setVar --name "${payment_type}" --value Card
setVar --name "${payment_value}" --value 870000
setVar --name "${expectedValue}" --value 03
```



You should now see the following call graph:



2.12 Run the bot

Run the bot without debug (Ctrl+F5)

You should see the following output:

```
[Info] Finding ticket 567567
[Info] Finding ticket 453445
[Info] Finding ticket 087877
[Info] Unit tests completed with 0 failures
```



3 Spreadsheet Driven Testing

In this section we will demonstrate how to test bots with spreadsheets.

3.1 Why test with spreadsheets

The tests built in the previous section were 'hard coded'. In other words, they were coded directly into the bot. It is more agile to define test scenarios in a spreadsheet. The bot runs against spreadsheet data and results are written back to the spreadsheet.

Use test spreadsheets when:

- Business users need to create tests.
- There are many tests
- Tests are changed frequently

3.2 Open the test scenario

Using RPA Studio, open the test scenario:

```
[Documents]\artefacts\data\refundTestScenarios.xlsx
```

You should see the following:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2													
3	1	Valid ticket refund	E3A98987	John Higgins	Card	4	22/07/2021	Bognor	Cancellation	00	Ticket refunded		
4	2	Invalid ticket	X123456	Sidney Barrett	Card	30.3	14/12/2021	Doncaster	Booking Error	01	InvalidTicketNumber		
5	3	Invalid payment type	E3B00293	Jim Davies	Cash	57.7	17/11/2021	Reading	Cancellation	02	Invalid payment type		
6	4	Invalid payment amount	E3B00362	Tom Cotton	Card	1000000	25/11/2021	Newbury	Illness	03	Invalid amount		

The green area defines the test scenarios. The yellow area contains the expected results. In this case, the expected result attributes are status code and status description.

Close the spreadsheet without making changes.



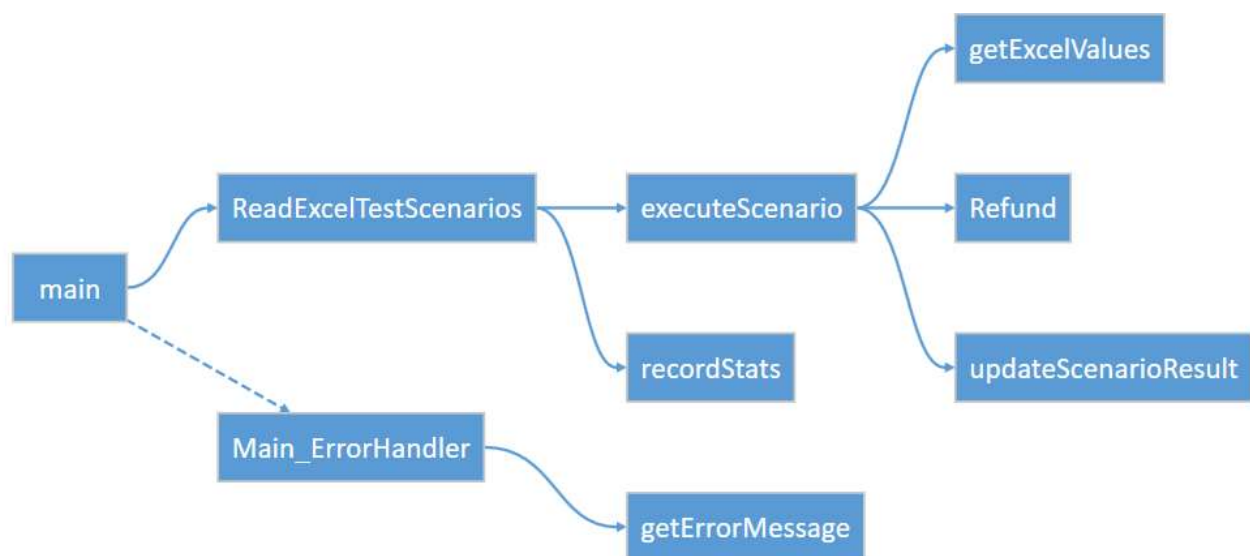
3.3 Open the test script

Within RPA Studio, open:

`[Documents]\artefacts\bot\refundTestScenarios.wal`

3.4 Examine the script flow

Go to the Call Graph tab. You should see the following:



You can see the excel scenarios are read, each scenario executed, and the results written back to excel.

3.5 Run the bot script

Hit **Ctrl+F5** to run the bot without debugging. After a while you should see the Refund website pop up four times, once for each test scenario defined in the spreadsheet. When finished, you should see the following in the console.

```
12/15/2022 3:22:56 PM - [Info] Refunds Unit Tests Started
12/15/2022 3:22:56 PM - [Info] Excel Path - ...\artefacts\data\refundTestScenarios.xlsx
12/15/2022 3:22:56 PM - [Info] Begin tests
12/15/2022 3:22:56 PM - [Info]
Ticket Number - E3A98987
Status -
Status Code - 00
Payment Type - Card
Payment Value - 4

12/15/2022 3:22:59 PM - [Info] Finding ticket E3A98987

12/15/2022 3:23:06 PM - [Info]
Ticket Number - X123456
Status -
```



Status Code - 01
 Payment Type - Card
 Payment Value - 30.3

12/15/2022 3:23:08 PM - [Info] Finding ticket X123456

12/15/2022 3:23:15 PM - [Info]
 Ticket Number - E3B00293
 Status -
 Status Code - 02
 Payment Type - Cash
 Payment Value - 57.7

12/15/2022 3:23:17 PM - [Info] Finding ticket E3B00293

12/15/2022 3:23:24 PM - [Info]
 Ticket Number - E3B00362
 Status -
 Status Code - 03
 Payment Type - Card
 Payment Value - 1000000

12/15/2022 3:23:25 PM - [Info] Finding ticket E3B00362

12/15/2022 3:23:32 PM - [Info] Robot Refunds Unit Tests ran at host XXXP52
 Processed File: ...artefacts\data\refundTestScenarios.xlsx
 Total items processed - 4
 Robot took 0 Minute(s) 36 Second(s) 605 Millisecond(s)

Now re-open the test scenario spreadsheet. You should see each scenario has changed to reflect the result of the test:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		TEST CASE								EXPECTED RESULT			
2	#	Test Scenario	TicketNumber	Customer	Payment	Value	Date	Destination	Reason	Code	Status Description	Timestamp	Pass/Fail
3	1	Valid ticket refund	E3A98987	John Higgins	Card	4	22/07/2021	Bognor	Cancellation	00	Ticket refunded	2022-12-15 10:48:04	PASS
4	2	Invalid ticket	X123456	Sidney Barrett	Card	30.3	14/12/2021	Doncaster	Booking Error	01	InvalidTicketNumber	2022-12-15 10:48:12	PASS
5	3	Invalid payment type	E3B00293	Jim Davies	Cash	57.7	17/11/2021	Reading	Cancellation	02	Invalid payment type	2022-12-15 10:48:20	PASS
6	4	Invalid payment amount	E3B00362	Tom Cotton	Card	1000000	25/11/2021	Newbury	Illness	03	Invalid amount	2022-12-15 10:48:29	PASS

Verify that all tests passed.



4 Conclusion

In this lab we presented two methods for unit testing bots. The first used the *Assert* method to unit test inside bot scripts. The second method extracted tests to a spreadsheet for more flexible testing.

Use the first method if you want to keep your test suite simple and have just a handful of tests. Use the spreadsheet if you have many test scenarios that may need to be maintained by the business.



Nicely done! This concludes the lab.