

Lab Guide

Exploring the RPA APIs

Nigel T. Crowther

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
2 The RPA API Overview.....	5
2.1 The Synchronous API.....	5
2.2 The Asynchronous API.....	6
2.3 Prerequisites	8
2.3.1 Download Lab Materials	8
2.3.2 Download Cygwin	9
2.3.3 Optional - Download SoapUI.....	10
2.3.4 Optional - Download Eclipse IDE	11
3 Exploring the RPA API	12
3.1 Create a bot process	12
3.2 Test the RPA API from PowerShell.....	12
3.2.1 Step 1 – Host Selection	12
3.2.2 Step 2 – Enter your tenant credentials	13
3.2.3 Step 3 – Select the tenant.....	13
3.2.4 Step 4 – Select the process	14
3.2.5 Step 5 – Edit the payload	14
3.2.6 Step 6 - Run the bot	15
3.2.7 Step 7 – View the Result	15
3.2.8 View the RPA API Curl.....	17
3.2.9 Run curl.....	17
3.3 Generating Open API Specifications	19
3.3.1 Start Eclipse.....	19
3.3.2 Set the perspective to Java	22
3.3.3 Build and run the Open API Generator	24
3.3.4 View the Asynchronous API	26
3.3.5 View the Synchronous API	27
3.3.6 Run the Synchronous API	29
3.4 Test the API with JUnit	31
3.5 Testing the API from SoapUI	34
3.5.1 Open SoapUI	34
3.5.2 Setting Project Variables	36
3.5.3 Run the test suite	37
3.5.4 Examine the test suite flow.....	40
3.6 Call Synchronous API from BAW	42
3.6.1 Import the signer certificate to BAW	42
3.6.2 Create BAW Process Application for testing the Synchronous RPA API	46
4 Integrate with the World!.....	54



1 Introduction

In this lab you will **explore the RPA APIs**. You will learn how to:

- Apply tools such as PowerShell, *curl* and *SoapUI* to test the RPA API
- Generate an *OpenApi* specification to integrate your bot to the modern world.
- Test the API using *Java* and *Junit*.
- Call the synchronous API from BAW.

For allocating an IBM VM and for calling the asynchronous API from BAW, see lab:

<https://www.ibm.com/docs/en/rpa/21.0?topic=automation-rpa-api-reference>

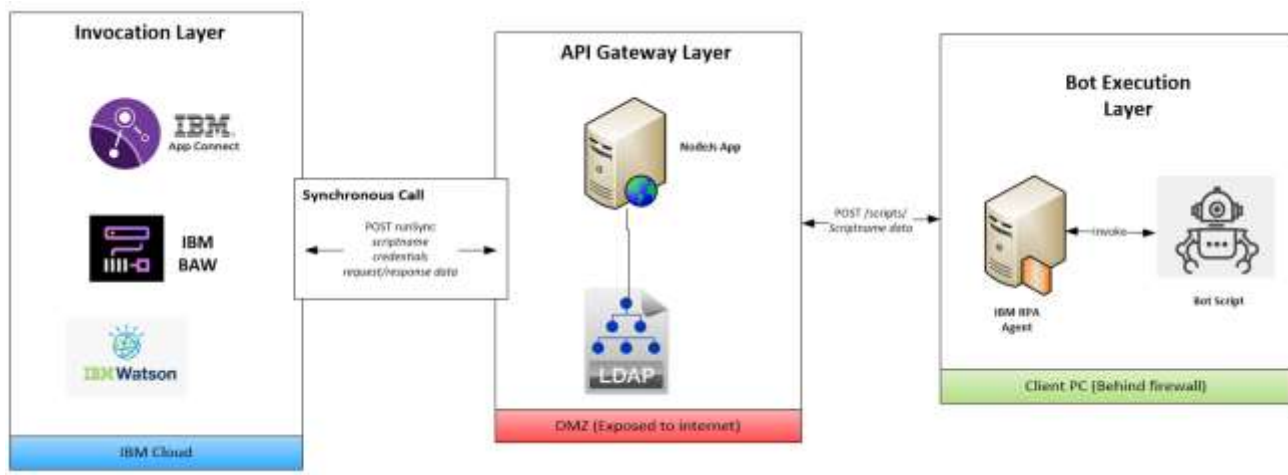


2 The RPA API Overview

This section summarizes the synchronous and asynchronous RPA APIs.

2.1 The Synchronous API

The synchronous API invokes a bot on a designated computer (running an RPA agent) and waits for the bot to complete. There is no authentication or built-in security. You may need to add a security gateway, where the invocation to the gateway would be made from external applications. See below:



The advantages of the synchronous API are:

- It requires just one API call
- It is easy to integrate with modern applications using an Open API specification.

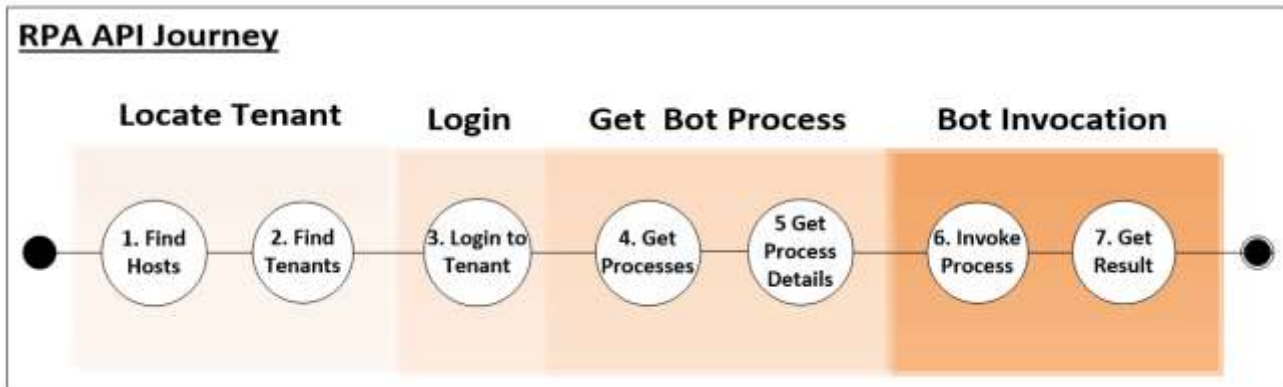
Disadvantages are:

- Lack of security. The API is not authenticated, opening your bot to abuse if not protected.
- Long running bots may cause time outs. Bots are generally long-lived processes, taking minutes or hours to complete. It is bad practice to write APIs that wait that long.
- Lack of resiliency. The API calls your computer directly, so if your computer is down the bot won't run.

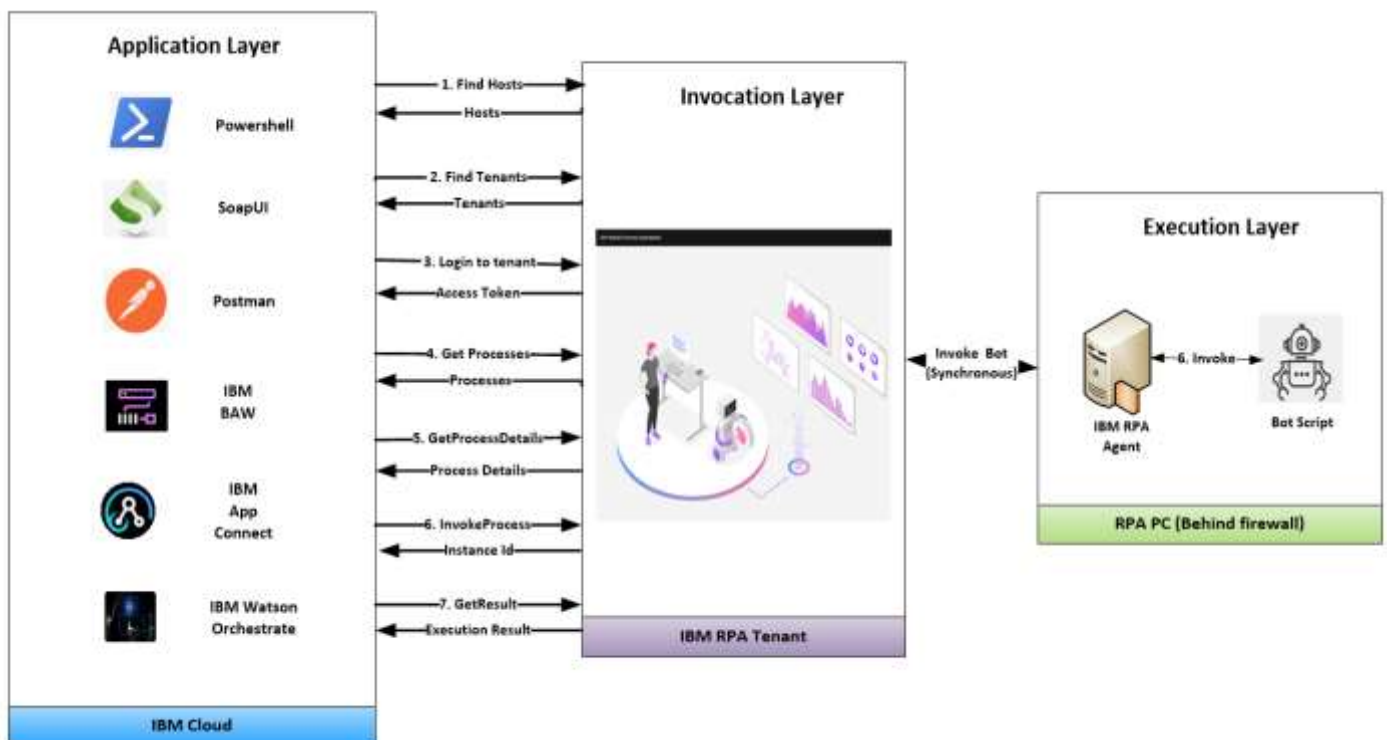


2.2 The Asynchronous API

The asynchronous API follows a journey. The journey starts with authentication and ends with retrieving the result. The following diagram describes this:



The API is invoked against the RPA tenant. The tenant authenticates the caller, locates the process in which the bot resides, decides the best computer on which to run the bot and then invokes the bot. The full interaction is depicted below:



The advantages of the asynchronous API are:

- It is authenticated
- It can be invoked anywhere on the world wide web (given correct credentials).
- Resilience. Bot invocation runs on the most available computer.



Disadvantages are:

- Bot invocation involves several API calls
- Extra development - the bot must be wrapped inside a process

There are also limitations in the current version of this API that may be rectified in future versions:

- For BAW, the Open API specification cannot be imported, which means low-level code must be written.
- The access token created by the bot needs to be regenerated every 24 hours.
- There is no two-factor authentication when generating access tokens.
- The API signature does not always get updated after bot parameters change

You will explore both the synchronous and asynchronous APIs in this lab. Let's get started!



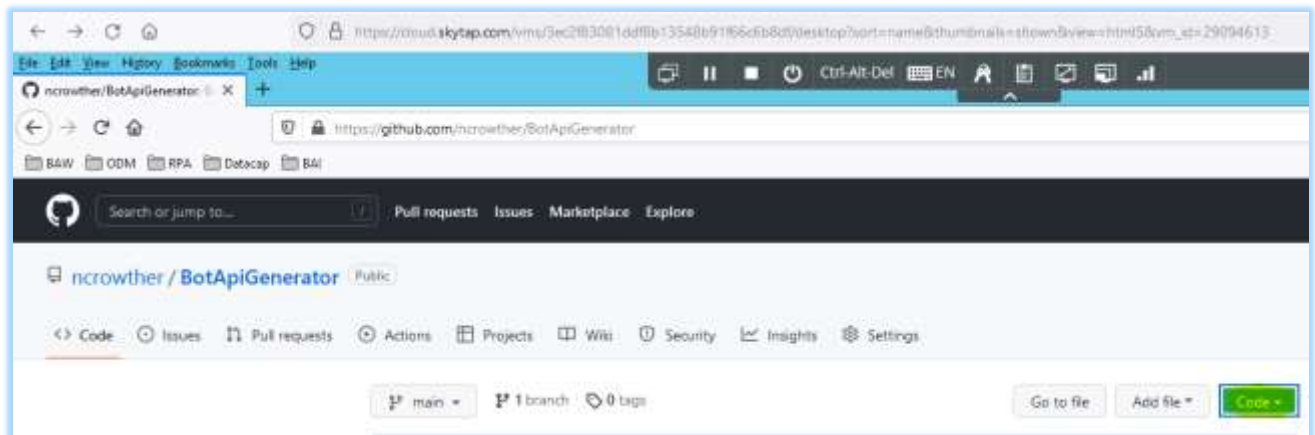
2.3 Prerequisites


2.3.1 Download Lab Materials

The prerequisite material for this lab is stored in **Github**. Use a browser to navigate to:

<https://github.com/ncrowther/BotApiGenerator>

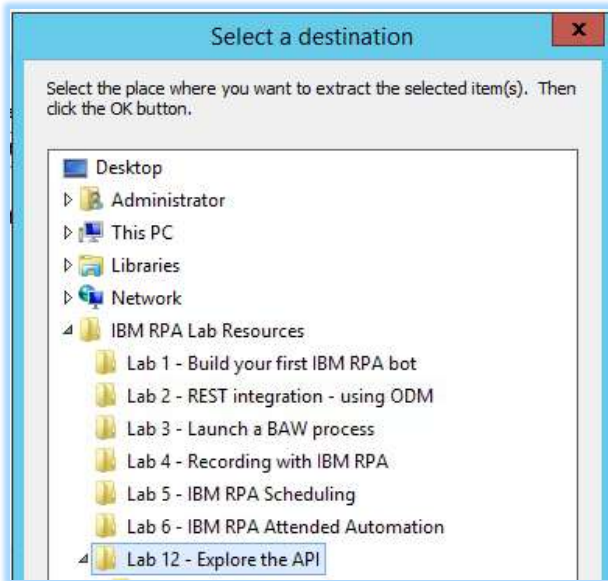
You should see the following:



Click on the  button, and then *Download Zip*. Save the file. Once downloaded, right-click the zip and select *extract all...* Extract the zip to a local folder. If using the Skytap image, create the folder below and extract the contents:

```
C:\Users\Administrator\Desktop\IBM RPA Lab Resources\Lab 12 - Explore the API
```

See below:



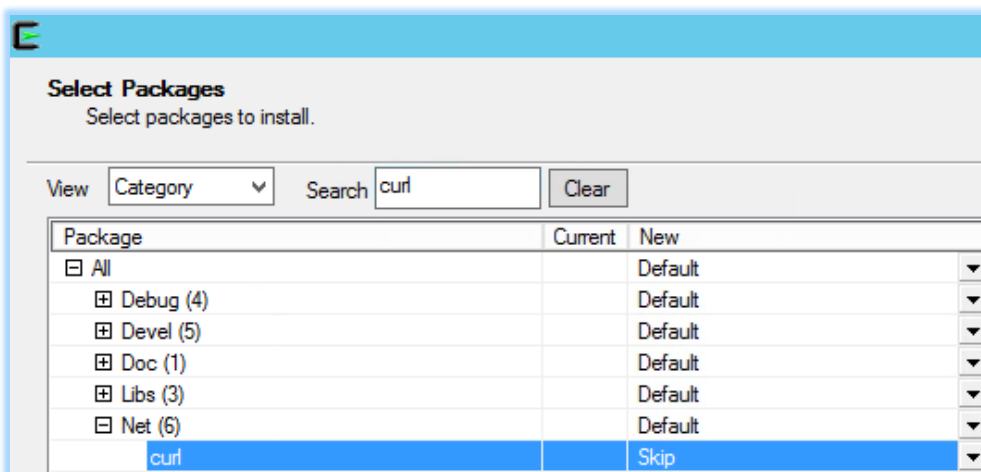
2.3.2 Download Cygwin

Using a browser, download and run:

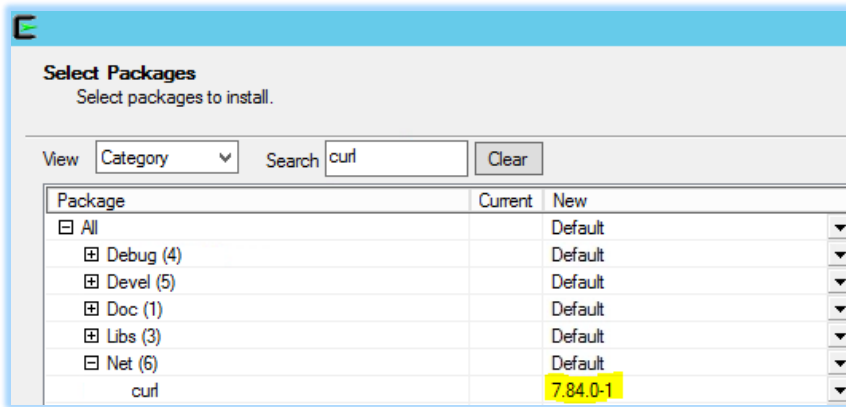
https://www.cygwin.com/setup-x86_64.exe

Use default installation settings. When it comes to selecting a download site, select your most local server.

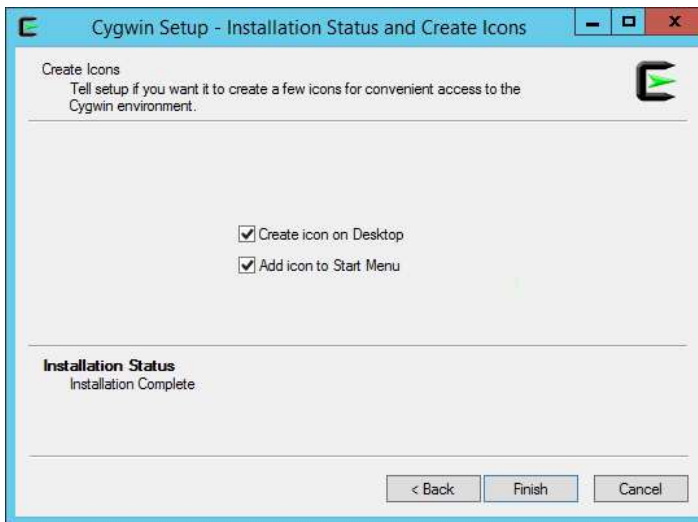
When selecting packages, enter **curl** as the search criteria and expand the package tree. You should see the following:



Curl is in *Skip* state which means it won't be installed by default. Change 'Skip' to the latest version as highlighted below.



Now click *Next* and continue pressing defaults until the installation window starts. After a minute you should see this:



Press Finish. The install is complete!

2.3.3 Optional - Download SoapUI

This step is not required if you are using the Skytap image.

You can download an open source version of SoapUI from the following location:

<https://www.soapui.org/downloads/soapui/>

Click on *Download SoapUI Open Source*.

Save the zip, extract and install the software using the default settings.



2.3.4 Optional - Download Eclipse IDE

This step is not required if you are using the Skytap image.

You can download Eclipse from the following location:

<https://www.eclipse.org/downloads/>



3 Exploring the RPA API

3.1 Create a bot process

IMPORTANT. To start your exploration, you need to create a bot process. Please download and follow steps **2.2 to 2.5** in this lab:

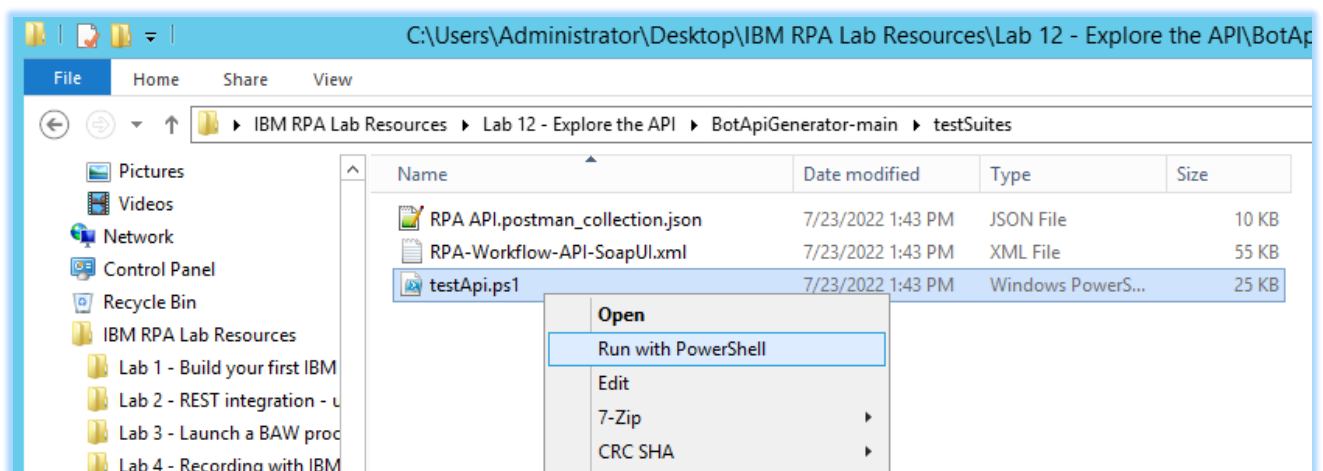
https://github.com/juseljuk/IBM-RPA-Toolkit-for-BAW/blob/master/downloads/Using%20IBM%20RPA%20with%20IBM%20BAW%201_1.pdf

3.2 Test the RPA API from PowerShell

Navigate to the Folder in which you unzipped the GitHub Repo in the pre-requisites step:

C:\Users\Administrator\Desktop\IBM RPA Lab Resources\Lab 12 - Explore the API\BotApiGenerator-main

Within the *testsuites* folder, Right-click *testApi.ps1* and select *Run with PowerShell*:

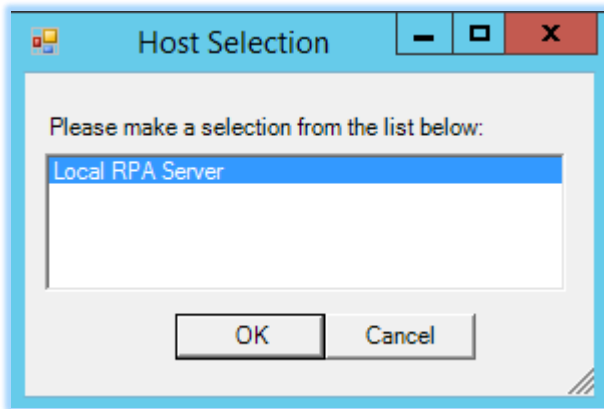


Ignore the security warning and press *Open*. A series of dialog boxes should appear. Let's follow the steps.

3.2.1 Step 1 – Host Selection

If you are running the Skytap image you will not be able to see your SaaS tenants. Press OK when the message “*Unable to find SaaS tenants*” appears.

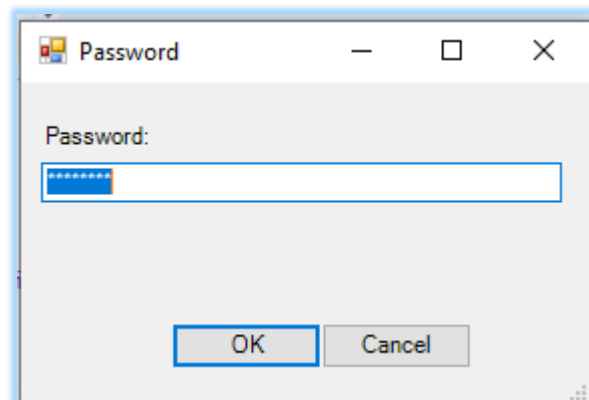
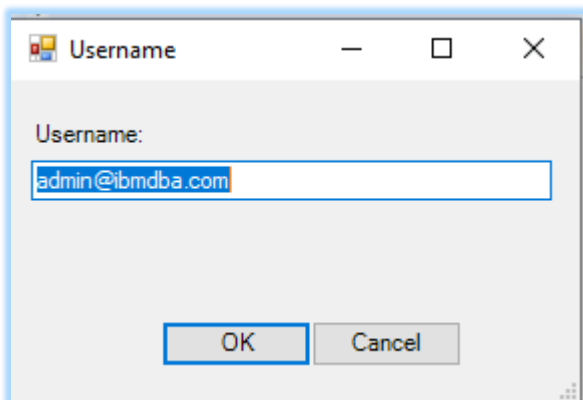
Select *Local RPA Server*



Tip: You need to select it so that it turns blue as shown above.

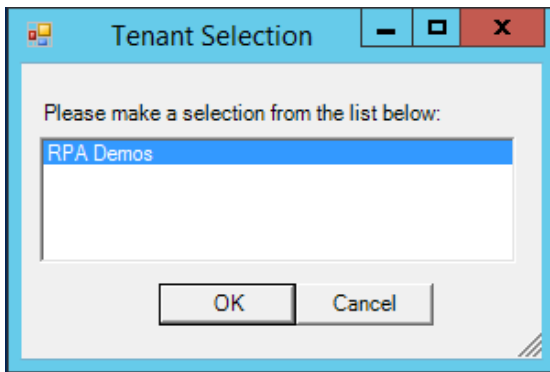
3.2.2 Step 2 – Enter your tenant credentials

If you are using the Skytap image the username and password are set as default values. If you are using your own tenant, these will be your tenant credentials.



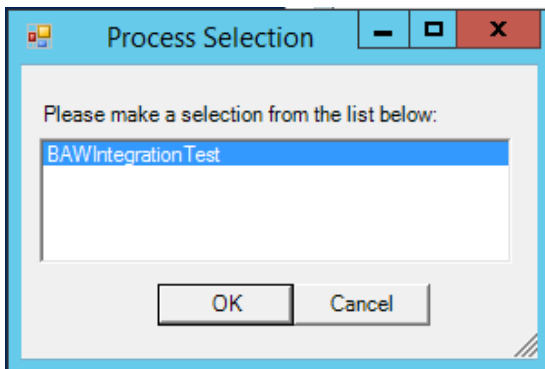
3.2.3 Step 3 – Select the tenant

Select tenant. If you are running within the Skytap image you only have one option. Select *RPA Demos*:



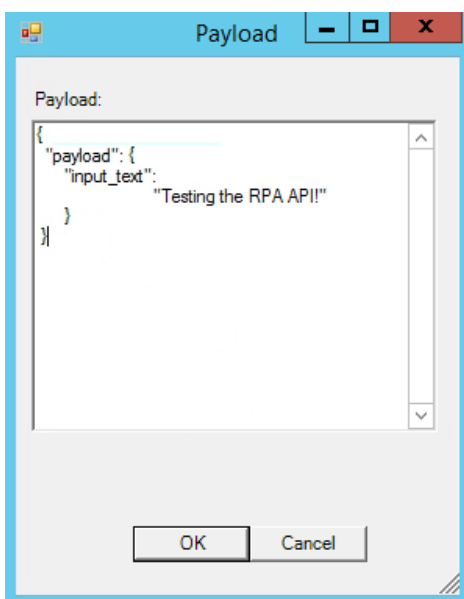
3.2.4 Step 4 – Select the process

Select the process you defined in the earlier step:



3.2.5 Step 5 – Edit the payload

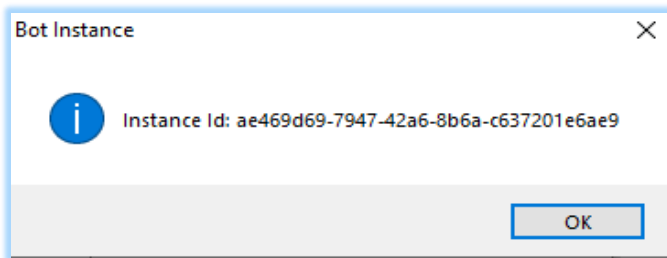
The payload will be generated from the bot input parameters. You can modify the input value if you wish:





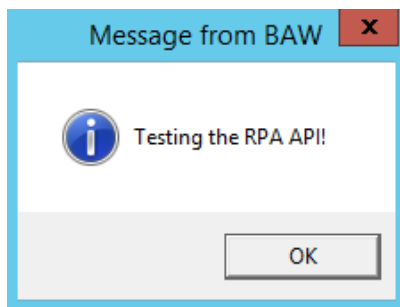
3.2.6 Step 6 - Run the bot

The bot runs and a dialog box appears showing the instance id of the running process

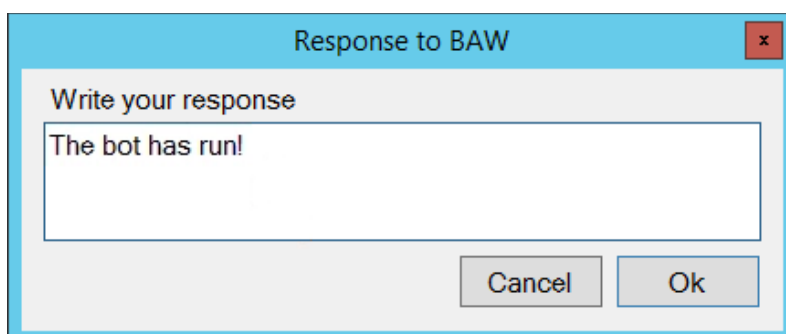


3.2.7 Step 7 – View the Result

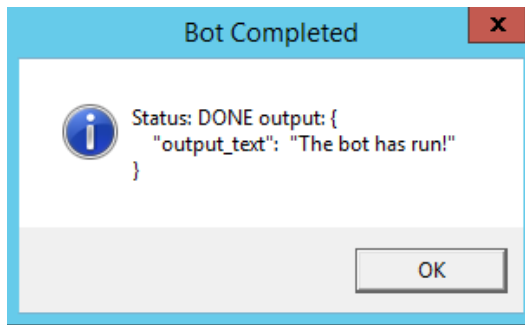
After a second, the bot runs, and you will see the following message:



Click OK and type your response:



The bot may take several seconds to run, in which case it will either be in *new* or *processing* state. When completed, the state will have a status of *done* and you will see the following dialog:



Press OK to finish.

Congratulations, you have run your first bot using PowerShell to drive the asynchronous API!

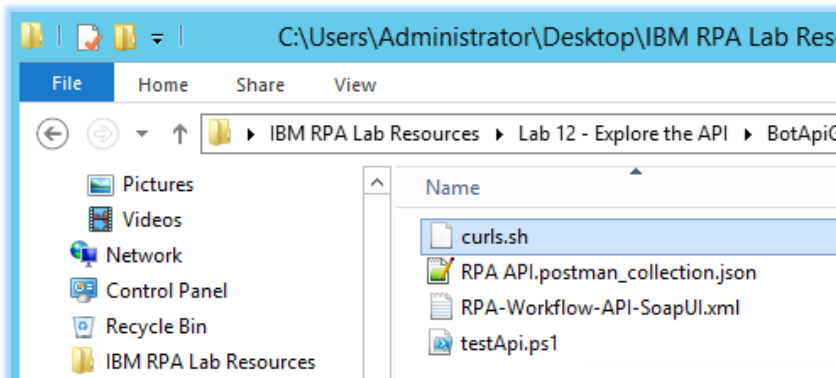
Next you will view the actual RPA API commands executed.



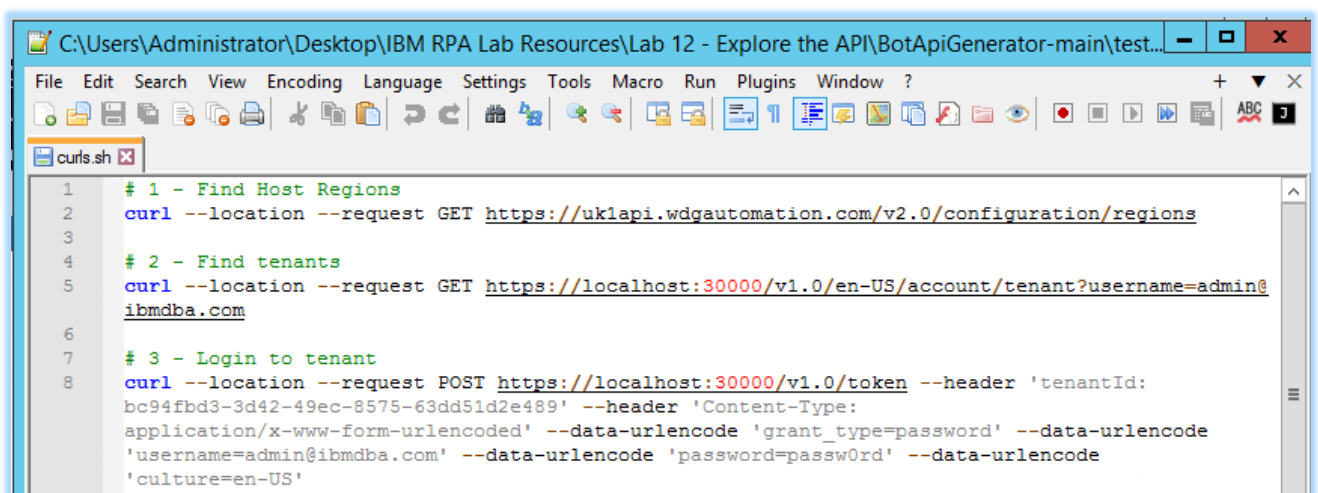
3.2.8 View the RPA API Curl

In this section you will use curl to call the RPA API. Curl is the de facto tool to test APIs.

Navigate to the directory in which you ran PowerShell. Right click *curls.sh* and edit using *Notepad++*.



You will see the curl executed in the previous step.



3.2.9 Run curl

From the desktop, open



A Unix terminal should appear.



Paste the curl commands into the terminal. Examine the result of each command to verify it is working. Finally, your terminal should look like this:

```
Administrator@IBMBAW ~  
$ curl -k --location --request GET https://localhost:30000/v2.0/workspace/bc94fbd3-3d42-49ec-8575-63dd51d2e489/process/b6bfea68-7f55-43c6-821e-e5968a9f0c2f/instance/eb0a3dd3-2119-4ffe-a0c5-fbe724eae71b --header 'Authorization: Bearer 15PdWmZXG08krhJVyRmxDcW8KdH2fTHw-pu6cobTG1Tio9T2AiORIdhMAZngYL4iGDORWRhrm4RdcgeKKLDjjQke4er5Ex1d_wYPRZ-HmUqkyuhnMRpAq6VPPdkimV6dCZi6_Ix7d8bf18916XW9tmjW8NDDQ7SaxhyLtPpzAkVc3TEp9N9o9Pj55NZ0Sa-xgDGojaZuzwxc4Av6f8x1CLOIT12Rmxm5kV0zac9FgPCU7qzI96nLd3cC9kMxgPQ5wHM_9jkEffITDRj8wYGvk1NFUD9aw7FJxcl9o2d4ZTw6HiawOHpVYC65TGs0EJmMivVti6TFEyaKteDES_zKjVxuCFaz0-k5m121pCrF0pLsgtUqPD1SmrTuum_7b2gf40B71L58I4AeUdKB8CHC1iCu6x16zqp8xCwzTsxo0Xcd_XZAZqTyA1ntkKNi1WwTsw5LQjG6hounuG0wmB0WdcQnFTL37FwtLoA1IkvTsJTSkQfWn9krG7fbSJoHjYE-vSyVYUdRwXP2tpbrUJEmiCFxq_jp33PEauaynMp8xpdepbkskfhwtcv7LxDrTI2EdR20dti-gicDmZ1w-aYzJwwQLhxKYZY4nv_PtURuATVq9xBXj-L4In9qF-_augSSODK8AJe0628bFhJ-rbNhxGZ4Wv6MzDx5AtGsFL5_Q0801AJpE7zss7uVyhdNE2XRgAEgd-D7VWvwUAeEIA2hNN9pde2ueXD00u5BjbNp5e1wHZvn9PNPctCsMFFMgz1NIy0Y5nxxnMEY1q5vx2NRbQnX_7Vvp-6W9btFJr261nEb1gQq5m6rG1C7JLpciRq9Ccu_Aj6hYGo3watgtvw8Z4YqoGCU0NgAJD35pXAM1IjkiZy1HEcW0CRID59qdUFdS1U0VYX_M0oDpgsP9KU58g5tD5jak_2FIKYiIhqrFeXh5cDojUHQddq1HtikhWnpqEb3ZOG6MMwqL_8JUHR5kgd-uQPjPxqAyHXzdv1TzanjCkWK59R3558EN6LOtyc' --data '{"status": "done", "variables": [{"value": "string", "name": "input_text"}], "outputs": {"output_text": "ggg"}}'  
Administrator@IBMBAW ~  
$
```

Congratulations, you have run a bot using curl!



3.3 Generating Open API Specifications

Up until now you used the raw REST RPA API. But there is a better way! Modern applications allow API integration through the *OpenAPI* specification.

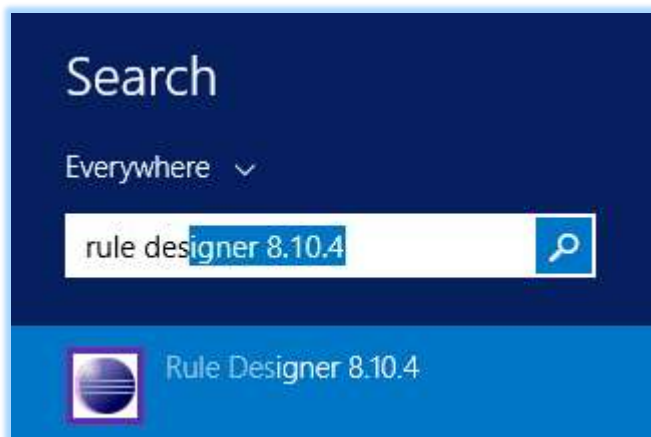
The advantages of using OpenAPI are:

- It's a universal API definition language understood by most modern applications
- It can be easily understood by humans in tools such as <https://editor.swagger.io/>
- The definition language is rigorous, so avoids the creation of sloppy APIs.

Let's go ahead and generate both a synchronous and an asynchronous OpenAPI spec for the bot you created.

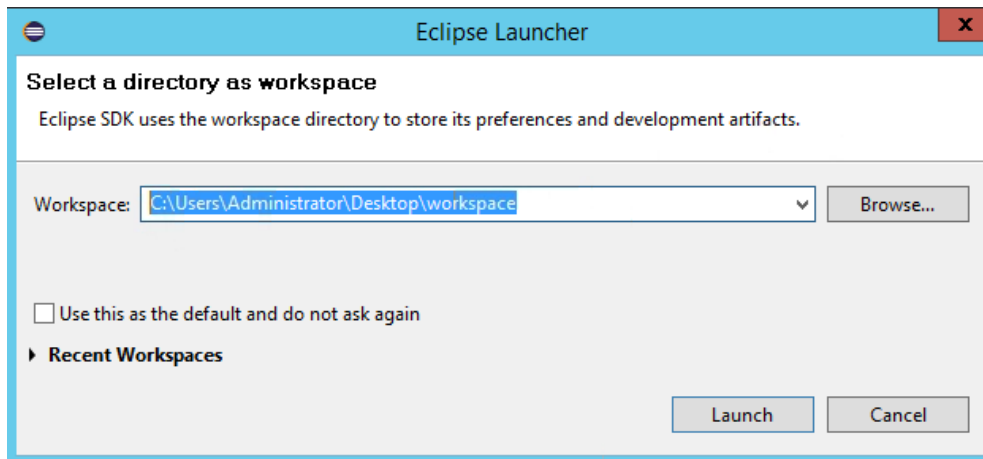
3.3.1 Start Eclipse

If you are running in the Skyap image, you can access Eclipse by running **Rule Designer**. Search for *Rule Designer* in the Windows search and then double click to start.



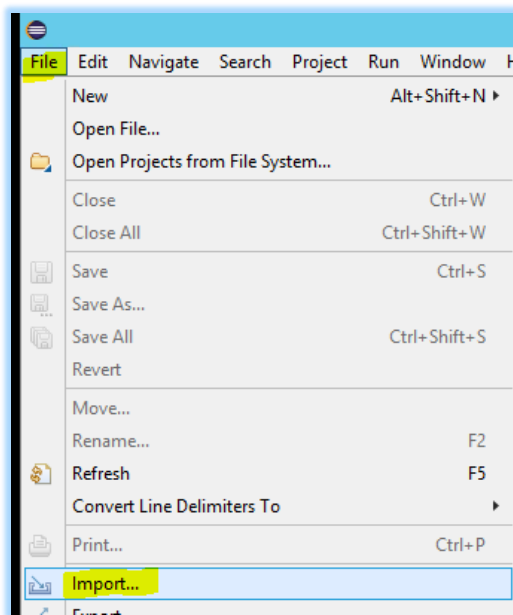
Create a workspace, for example:

```
C:\Users\Administrator\Desktop\workspace
```

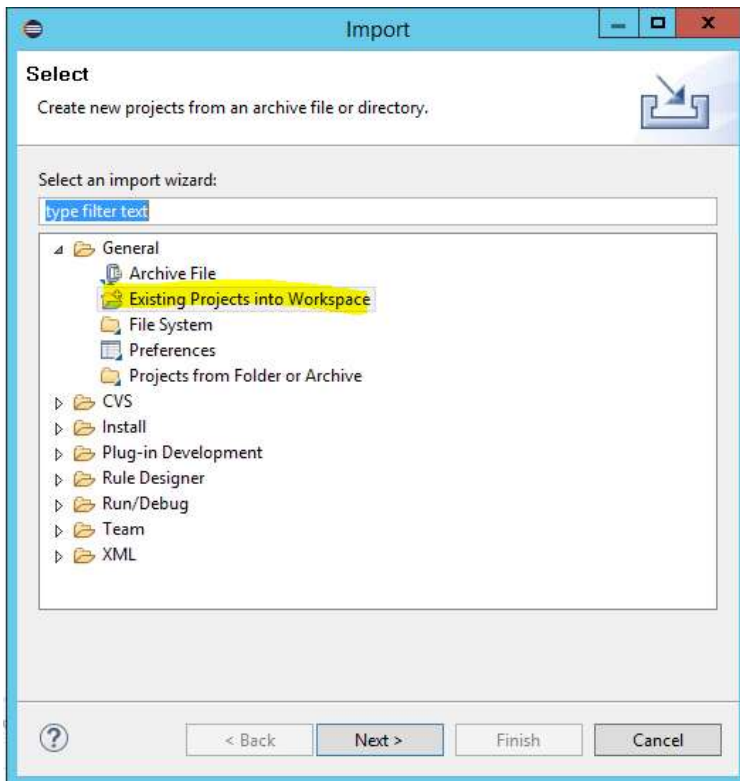


Press *Launch*.

Once Eclipse is open, select *File->Import*:



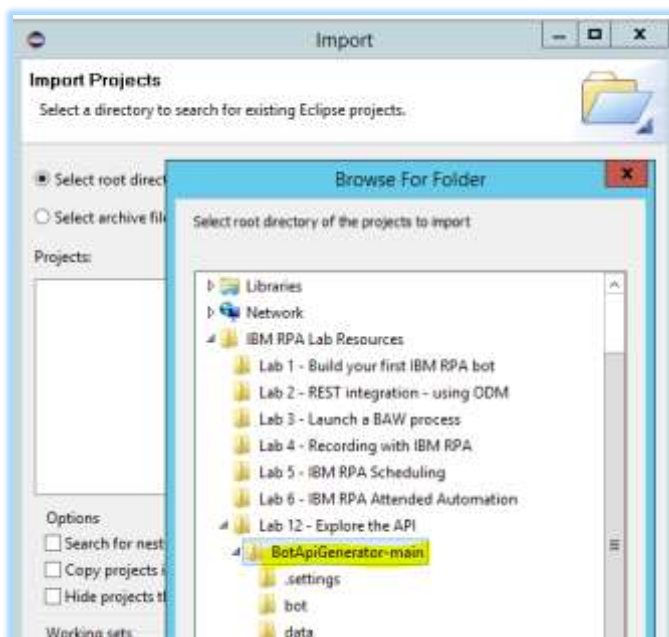
Select *General->Existing Projects into Workspace*:



Press *Next*. Select the folder

C:\Users\Administrator\Desktop\IBM RPA Lab Resources\Lab 12 - API\BotApiGenerator-main

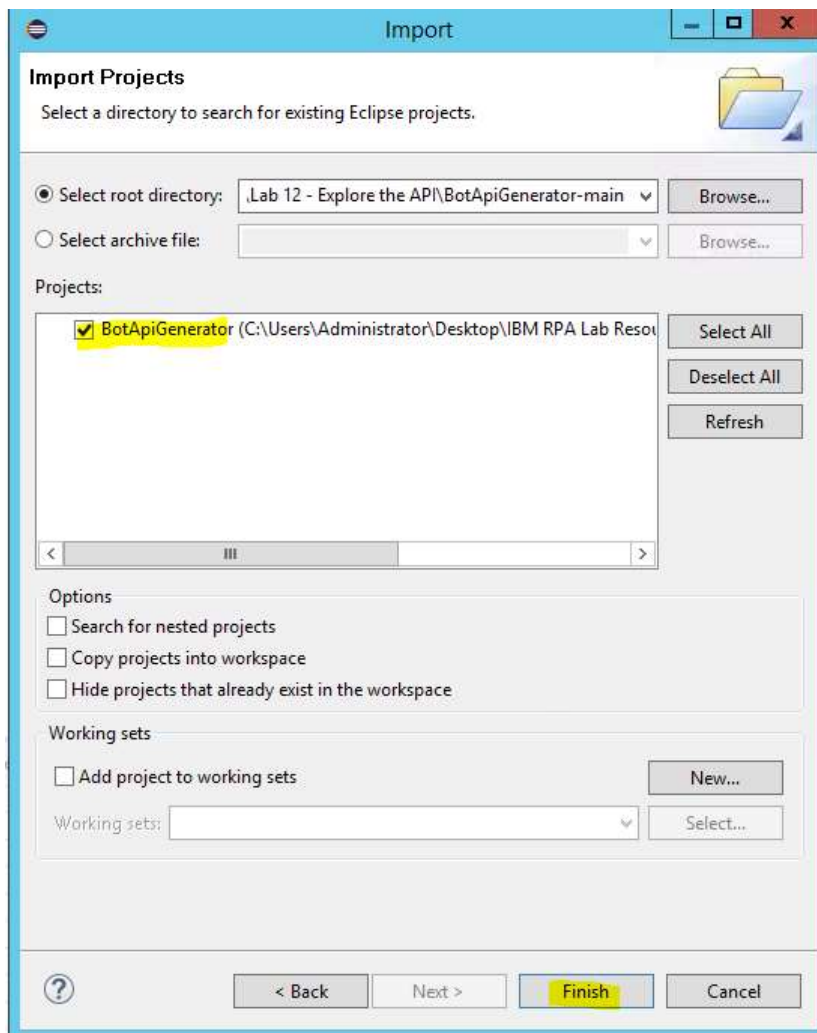
Note: This location may vary depending on where you downloaded your lab in step 2.3.1



Press *Finish*.

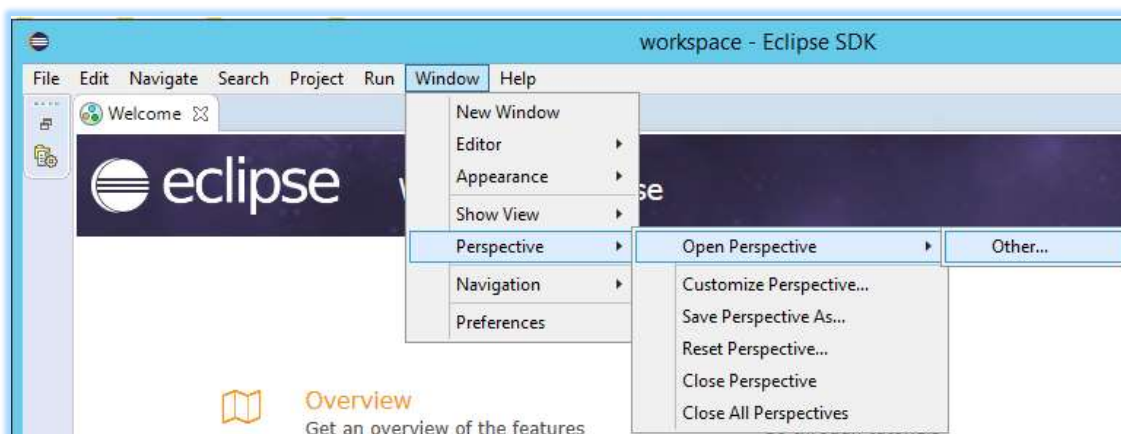


Select *BotApiGenerator* and then press Finish:



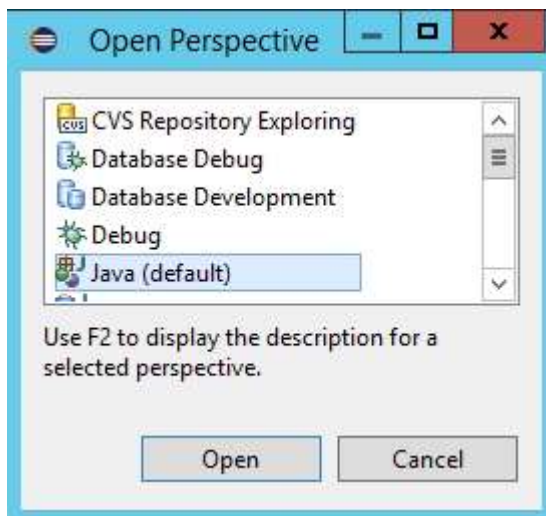
3.3.2 Set the perspective to Java

Navigate to Window->Perspective->Open Perspective->Other



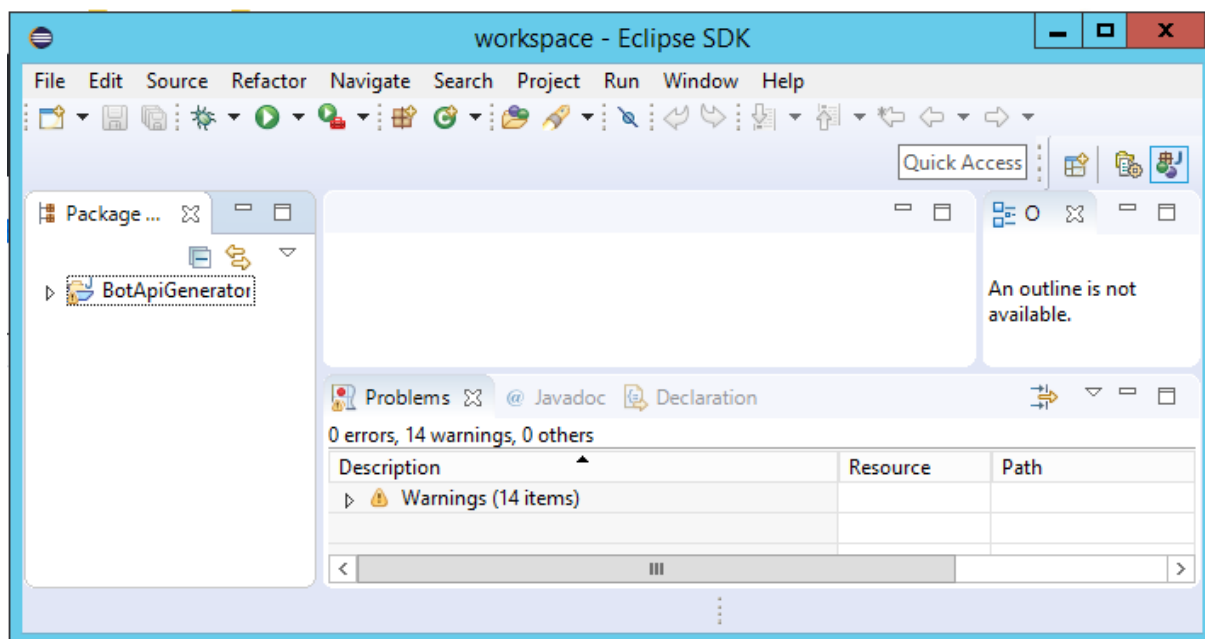


Then choose *Java (default)*:



Press *Open*.

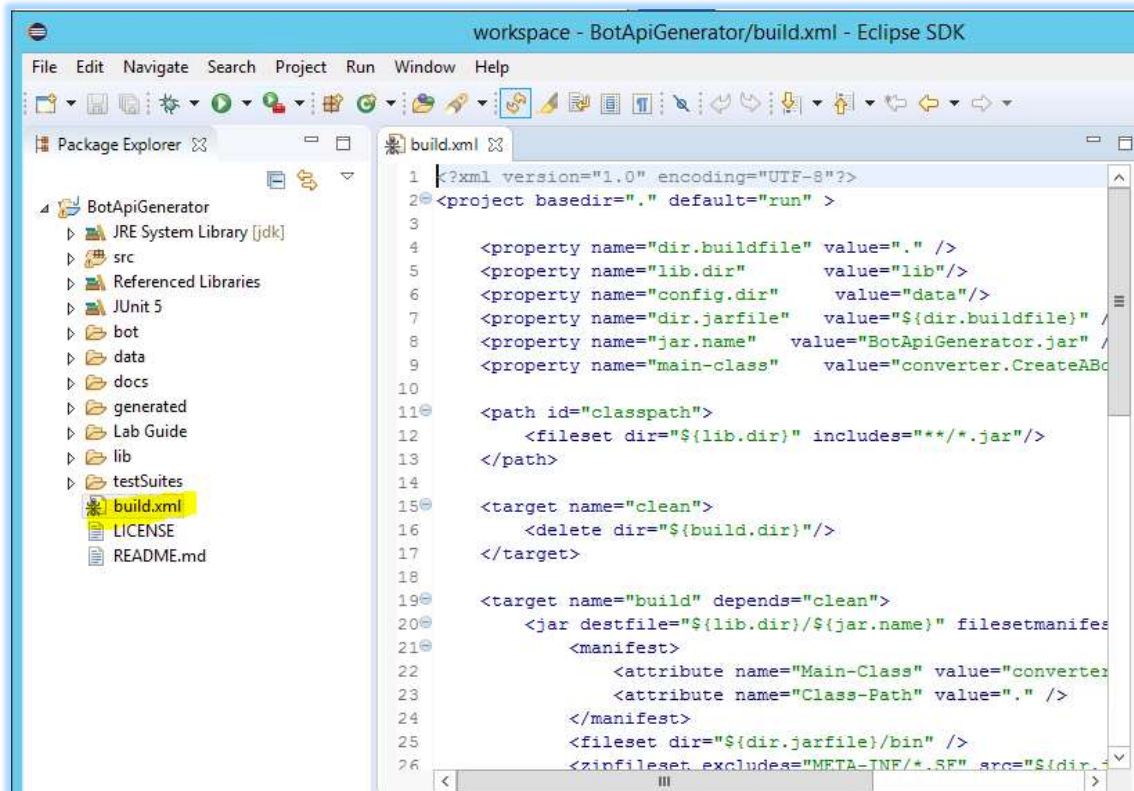
Close the *Welcome* tab. You should see this:



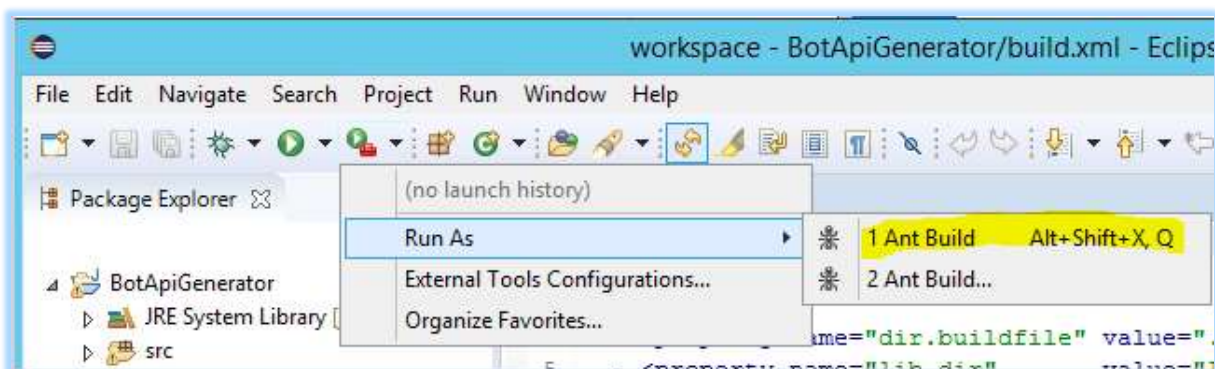


3.3.3 Build and run the Open API Generator

Expand the Package hierarchy on the left until you find *build.xml*. Double-click to open. You should see this:



Execute the *build.xml* by selecting the *External Tools* icon , then select *Run As->Ant Build*.



This runs the API generator. You should see the following output in the Console:

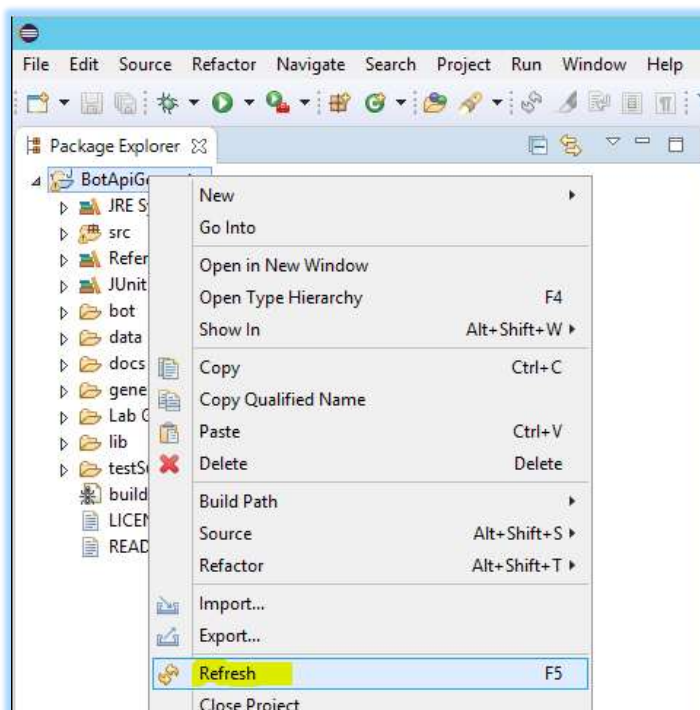


```
Problems @ Javadoc Declaration Console X
<terminated> BotApiGenerator build.xml [Ant Build] C:\IBM\ODM8103\jdk\bin\javaw.exe (Jul 26, 2022, 3:15:57 PM)

[java] apiHost:
[java] apiPayload:
[java] resUser:
[java] resPwd:
[java] RpaConfig [id=null, startId=null, name=BAWIntegrationTest, system=null,
[java] doRest: result is {"access_token":"GTQLjV7LRu9jT1lE3eMWYJhjoOROBq0g2OSK
[java] >> doRest: command=GET, urlString=https://ibmbaw:30000/v2.0/workspace/b
[java] doRest: result is {"results":[{"id":"b6bfea68-7f55-43c6-821e-e5968a9f0c
[java] >> doRest: command=GET, urlString=https://ibmbaw:30000/v2.0/workspace/b
[java] doRest: result is {"id":"b6bfea68-7f55-43c6-821e-e5968a9f0c2f","name":"
[java] BotInfo: BotInfo [workspaceId=bc94fbd3-3d42-49ec-8575-63dd51d2e489, pro
[java] **GENERATING CODE FOR BAWIntegrationTest
[java] Internal API generated in .\generated\Bawintegrationtest_internal.yaml
[java] **GENERATING CODE FOR BAWIntegrationTest
[java] External API generated in .\generated\Bawintegrationtest_external.yaml
[java] **GENERATING CODE FOR BAWIntegrationTest
[java] JUnit test generated in .\generated\BawintegrationtestTest.java
BUILD SUCCESSFUL
Total time: 1 second
```

Tip: If you see an error, it could be that the name of your process does not match that defined in the *data/config.json*. Edit this file to match your system.

Refresh the *BotApiGenerator* folder by hitting F5, or selecting the Refresh menu option:





3.3.4 View the Asynchronous API

Open `generated/Bawintegrationtest_external.yaml`:

```
1 openapi: 3.0.1
2 info:
3   title: BAWIntegrationTest RPA API
4   description: |-
5     This API presents an authenticated interface
6     The **IBM RPA API V2.0 API** provides s
7
8   * **The Authentication APIs** provide a JSON
9
10  * **The Tenant management APIs** provide s
11
12  * **The Process management APIs** provide
13
14  * **The Workspace APIs** provide endpoints
15  contact:
16    email: ncrowther@uk.ibm.com
17    version: 1.0.0
18  externalDocs:
19    description: Find out more about Swagger
```

The contents of this file contain a generated OpenAPI specification to run your bot asynchronously. Copy the contents of the entire file into the clipboard (Ctrl-A , Ctrl-C).

Open URL: <https://editor.swagger.io/>

Paste into the left panel. You should see the Open API specification for your bot:

BAWIntegrationTest RPA API 1.0.0 OAS3

This API presents an authenticated interface to invoke the BAWIntegrationTest RPA process asynchronously via the RPA tenant. Authentication is enforced through bearer token.

Application Layer | **Invocation Layer** | **Execution Layer**

The IBM RPA API V2.0 API provides support for external apps to the authentication flow, tenant management and configuration, workspace management, and process management APIs.

- The **Authentication APIs** provide a JSON Web Token (JWT) to be used as a bearer token to make API calls to the IBM RPA API Server.
- The **Tenant management APIs** provide endpoints to create and maintain tenants. This REST API supports creating, retrieving, modifying and deleting tenants. Only platform administrators can create tenants.
- The **Process management APIs** provide endpoints to create orchestration processes instances. This REST API supports retrieving information about processes and process instances and creating process instances.

Take a moment to explore the API.



3.3.5 View the Synchronous API

Let's revisit the synchronous API. Remember the synchronous API is not authenticated but it is easier to use and the bot can be invoked in a single call. Go back to eclipse and open

generated/Bawintegrationtest_internal.yaml:

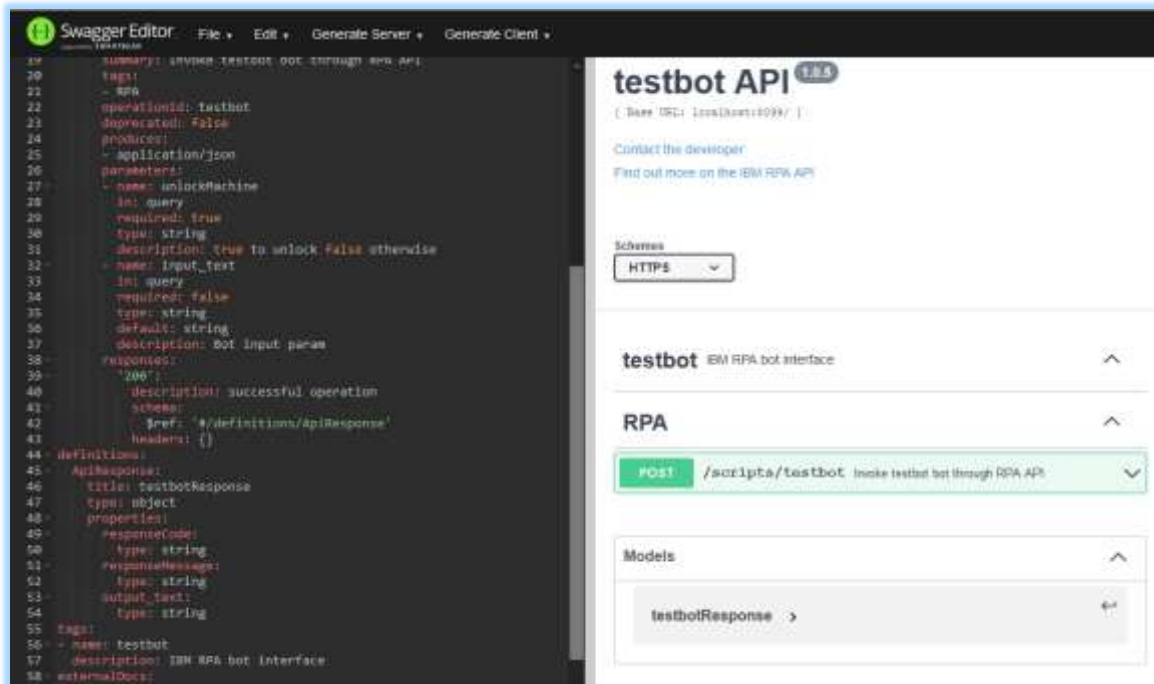
The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer displays the project structure for 'BotApiGenerator'. The 'generated' folder is expanded, showing 'Bawintegrationtest_external.yaml', 'Bawintegrationtest_internal.yaml' (highlighted), and 'BawintegrationtestTest.java'. On the right, the editor window shows the content of 'Bawintegrationtest_internal.yaml', which is an OpenAPI specification in YAML format.

```
1 swagger: '2.0'
2 info:
3   version: '1.0.5'
4   title: API
5   contact:
6     email: ncrowther@uk.ibm.com
7 host: localhost:8099
8 basePath: /
9 schemes:
10 - https
11 consumes:
12 - application/json
13 produces:
14 - application/json
15 paths:
16   /scripts/:
17     post:
18       description: This is an interface to RPA bot
19       summary: Invoke bot through RPA API
20       tags:
21         - RPA
22       operationId:
23       deprecated: false
24       produces:
```

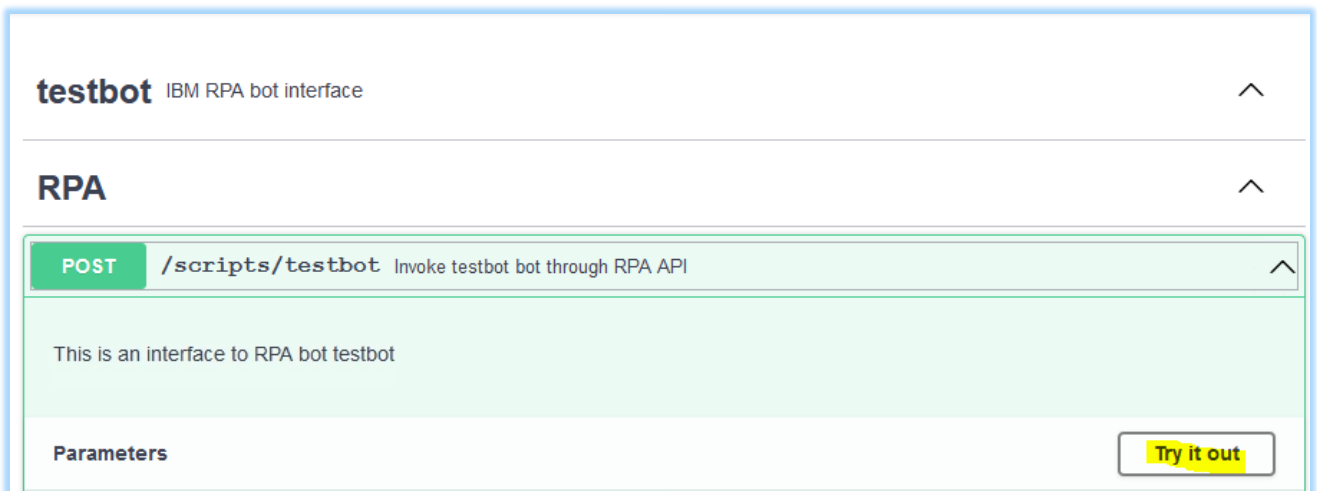
The contents of this yaml file contain the generated OpenAPI specification to run your bot synchronously. Copy the contents of the entire file into the clipboard (Ctrl-A, Ctrl-C).

Open URL: <https://editor.swagger.io/>

Paste into the left panel. You should see the Synchronous API specification:



Navigate to the *POST* command:



Expand the command. Press “try it out”. Enter *true* for *UnlockMachine* and the input text can be anything you like. Then press *Execute*. See below:



POST /scripts/testbot Invoke testbot bot through RPA API

This is an interface to RPA bot testbot

Parameters Cancel


Name	Description
unlockMachine * required	true to unlock false otherwise
string (query)	<input type="text" value="true"/>
input_text	Bot input param
string (query)	<input type="text" value="test"/>

Execute Clear

Responses Response content type application/json

Curl

```
curl -X 'POST' \
  'https://localhost:8099/scripts/testbot?unlockMachine=true&input_text=test' \
  -H 'accept: application/json' \
  -d ''
```

Copy the generated curl using the clipboard icon .

3.3.6 Run the Synchronous API

From the desktop, open a Unix terminal:



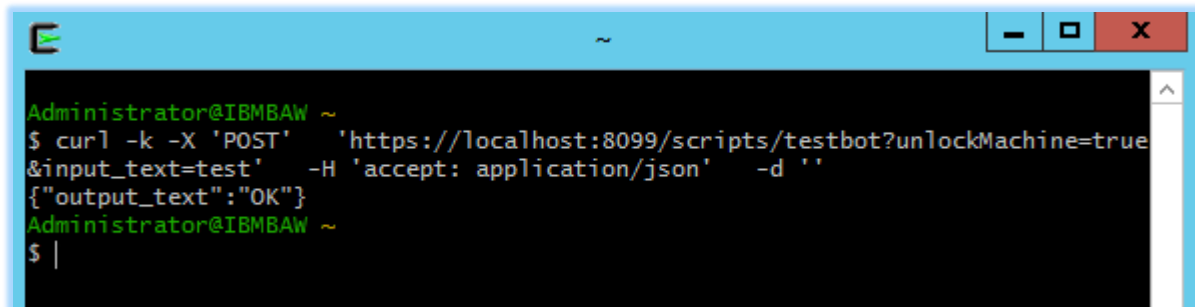
Paste the curl command into the terminal and hit return. If you are using Skytap you *must ignore certificate errors* by adding the **-k** (lowercase) flag to the beginning of the curl, command as shown below:

```
curl -k -X 'POST' 'https://ibmbaw:30000/v1.0/token' -H 'accept: application/json' -H 'tenantId: bc94fbd3-3d42-49ec-8575-63dd51d2e489' -H 'Content-Type: application/x-www-form-urlencoded' -d ''
```



```
'grant_type=password&username=admin%40ibmdba.com&password=passw0rd&culture=en-US'
```

Now the curl runs your bot synchronously and after a few moments you will see:



```
Administrator@IBMBAW ~  
$ curl -k -X 'POST' 'https://localhost:8099/scripts/testbot?unlockMachine=true  
&input_text=test' -H 'accept: application/json' -d ''  
{"output_text":"OK"}  
Administrator@IBMBAW ~  
$ |
```

Congratulations you have run your bot synchronously!

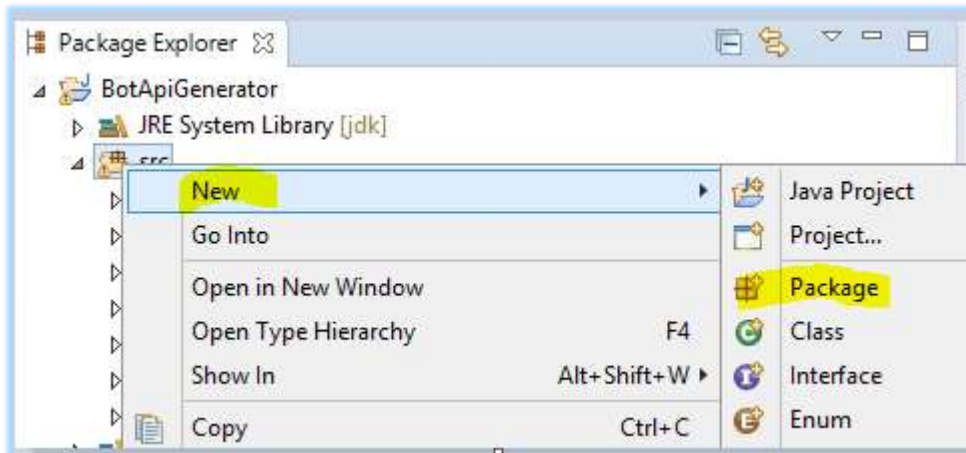


3.4 Test the API with JUnit

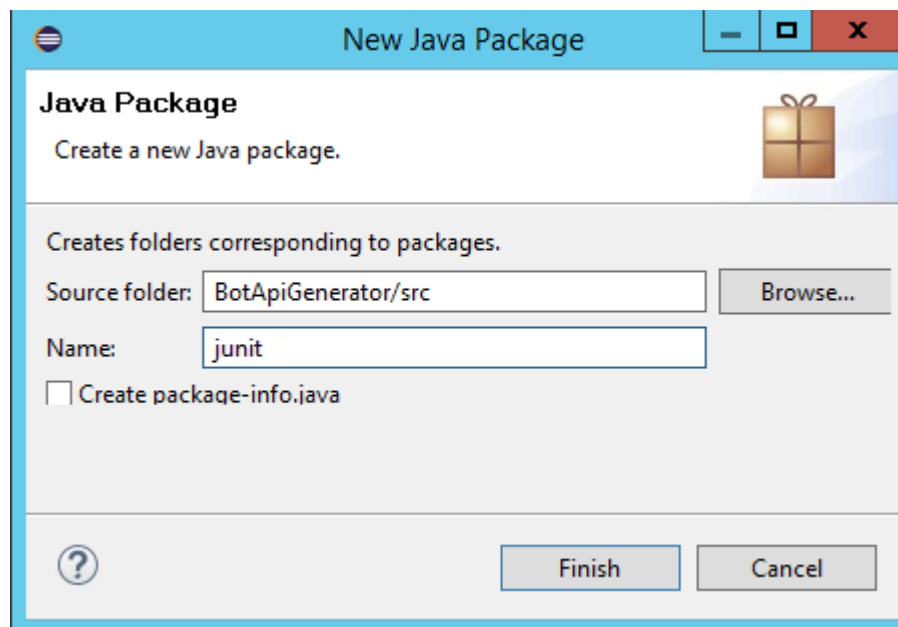
In this step you will test the RPA API using Junit. Junit is the original test framework and is still popular.

Revisit the Eclipse IDE you ran in 3.3.1.

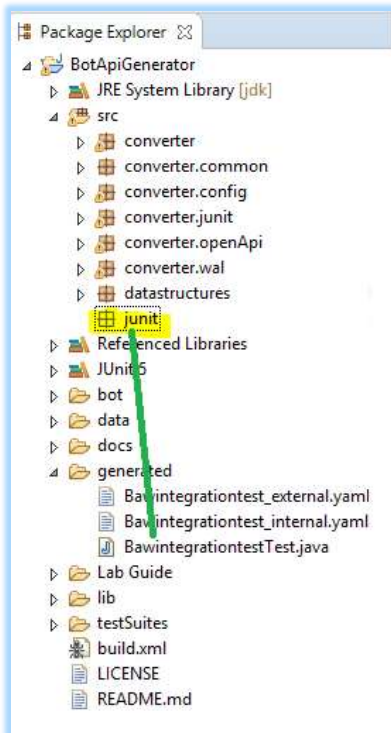
Create a new package under the *src* folder:



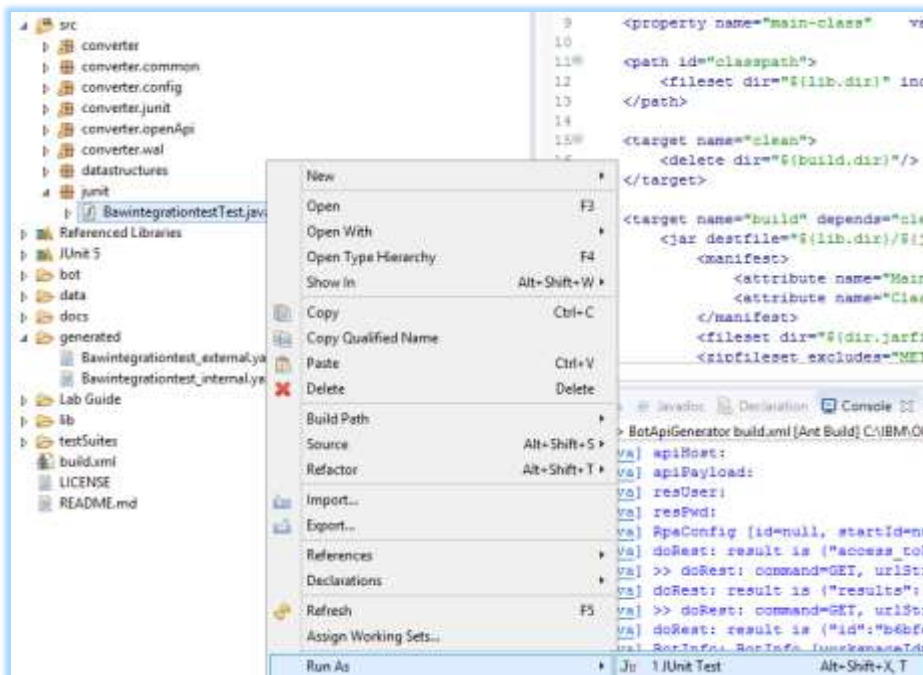
Set the name of the folder to *junit* and press Finish:



In the Package Explorer drag and drop *generated/BawintegrationTest.java* into the *junit* folder, as shown below:



Now right-click *BawintegrationTest.java* and run as a Junit test:



JUnit will test the bot using the RPA API.

After half a minute you should see this:

The screenshot shows the Eclipse IDE with the following components:

- Top Bar:** workspace - BotApiGenerator/src/junit/BawintegrationTestTest.java - Eclipse SDK
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse development tools.
- Left Panel:** Package Explorer showing the 'junit' package and the 'BawintegrationTestTest' class. Below it, a progress bar indicates 'Finished after 30.922 seconds' and 'Runs: 1/1', 'Errors: 0', 'Failures: 0'.
- Editor:** Displays the source code of 'BawintegrationTest.java'. The code includes imports for JUnit, static variables for 'baseUrl', 'tenantId', 'username', 'password', 'processName', 'payload', 'COMPLETED STATUS', and 'waitSeconds'.
- Right Panel:** Outline view showing the project structure and the methods of 'BawintegrationTestTest', including 'setUp()', 'testStartProcess()', and 'waitSeconds()'. It also lists various static variables.
- Bottom Panel:** Console view showing the execution output. It starts with a terminated message and then displays the command used to run the test: `doRest: result is ("access token": "3dsPRMmGmbowr-sQvDfAOngh8PmZaYwTlJRuU116Houp41VncmA76Wp^"`. The output continues with several REST API calls and their responses, including a successful status response: `doRest: result is ("status": "done", "variables": [{"value": "test", "name": "input_text"}], "outp`.

Note: Depending on how quickly you interact with the dialog boxes, you may see the test fail. The test failed because you took too long to enter a response. Rerun the test and make sure you enter a response within 10 seconds.

As you can see, there is a problem testing this bot. It relies on human input which fails if the human is too slow. Junit is not ideal for this scenario. It should be used for testing bots that do not require user input.

Nicely done! You have tested the RPA API with Junit.



3.5 Testing the API from SoapUI

Curl is great for initial testing of your bot, but it is not great for building test suites. SoapUI is an open-source tool that makes it much easier to build such suites. In this section you will apply SoapUI to test the async RPA API.

3.5.1 Open SoapUI

Find Soap UI and run it:



Once open, close the splash screen and click the *Import* button.



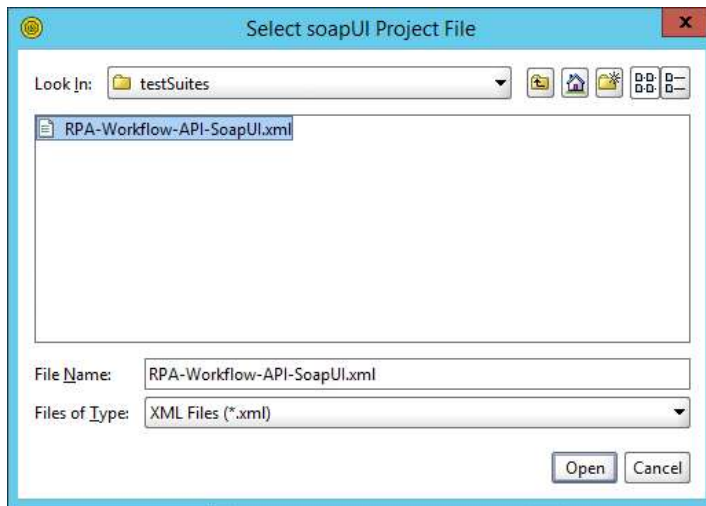


Navigate to folder:

C:\Users\Administrator\Desktop\IBM RPA Lab Resources\Lab 12 -
API\BotApiGenerator-main\testSuites

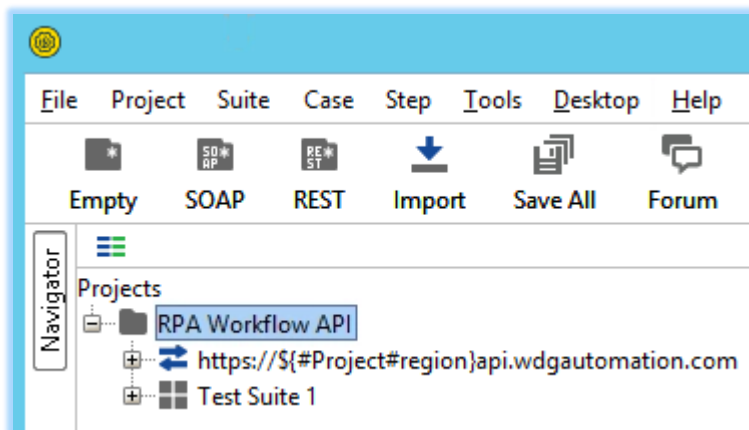
Note: This location may vary depending on where you downloaded your lab in step 2.3.1

Select *RPA-Workflow-API-SoapUI.xml* and press *Open*:



Ignore the version warning.

Verify the file is imported:

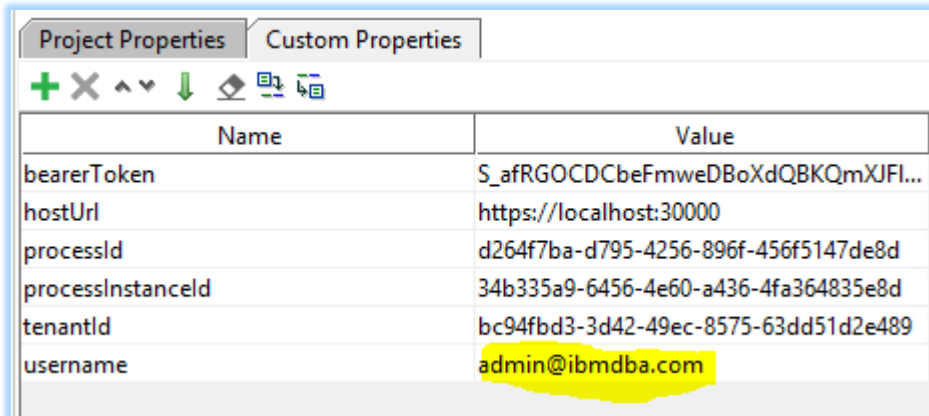




3.5.2 Setting Project Variables

Note: If you are using Skytap you can skip this step as the variables have been pre-configured.

The test suite retrieves information from project variables. Click on the project *RPA Workflow API* and select *custom properties* tab.



Name	Value
bearerToken	S_afRGOCDCbeFmweDBoXdQBKQmXJFI...
hostUrl	https://localhost:30000
processId	d264f7ba-d795-4256-896f-456f5147de8d
processInstanceId	34b335a9-6456-4e60-a436-4fa364835e8d
tenantId	bc94fbd3-3d42-49ec-8575-63dd51d2e489
username	admin@ibmdba.com

You can edit these variables to your own tenant and processes. The two key variables are *username* and *hostURL*.

The default for *username* is

admin@ibmdba.com.

If you are not using Skytap, you need to set this to your RPA user.

The default for *hostURL* is

<https://ibmbaw:30000>

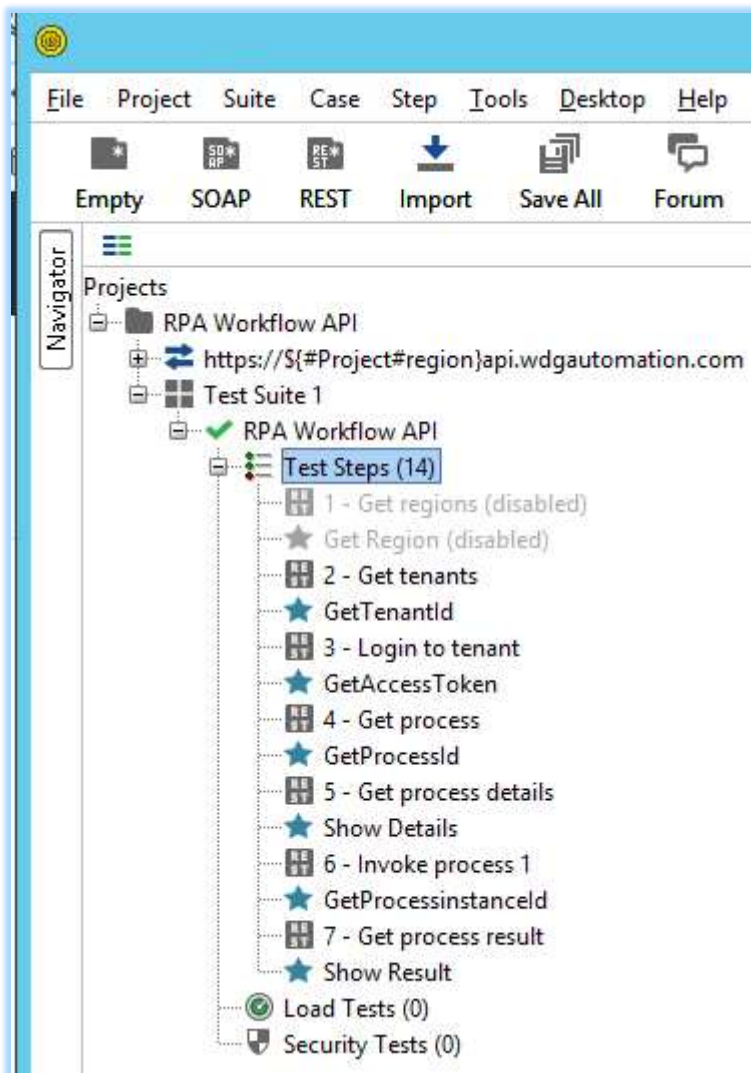
If you are not using Skytap, you will need to set this to your RPA server. If you are using SaaS, you can use the tenant below:

<https://us1api.wdgautomation.com>

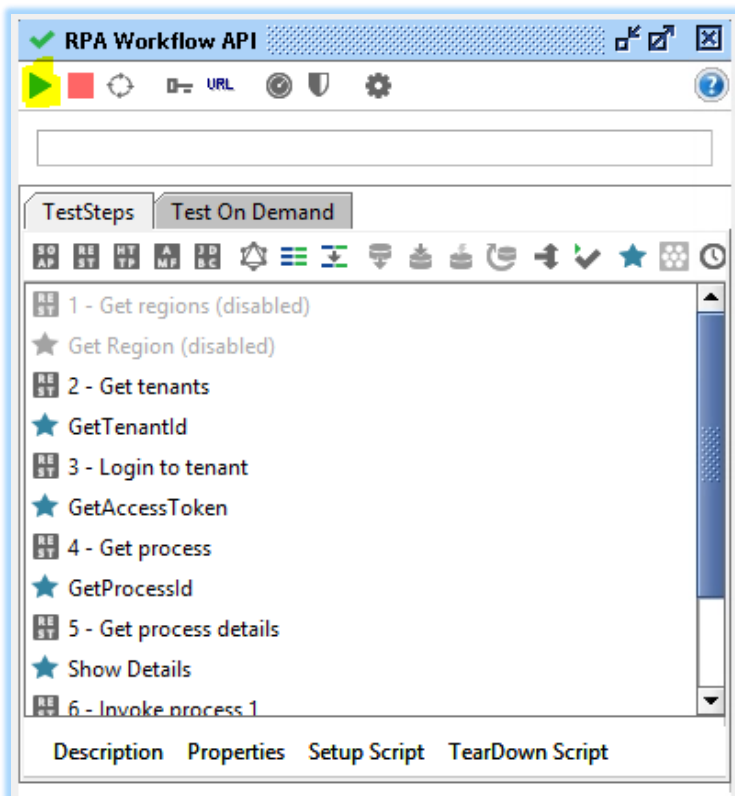


3.5.3 Run the test suite

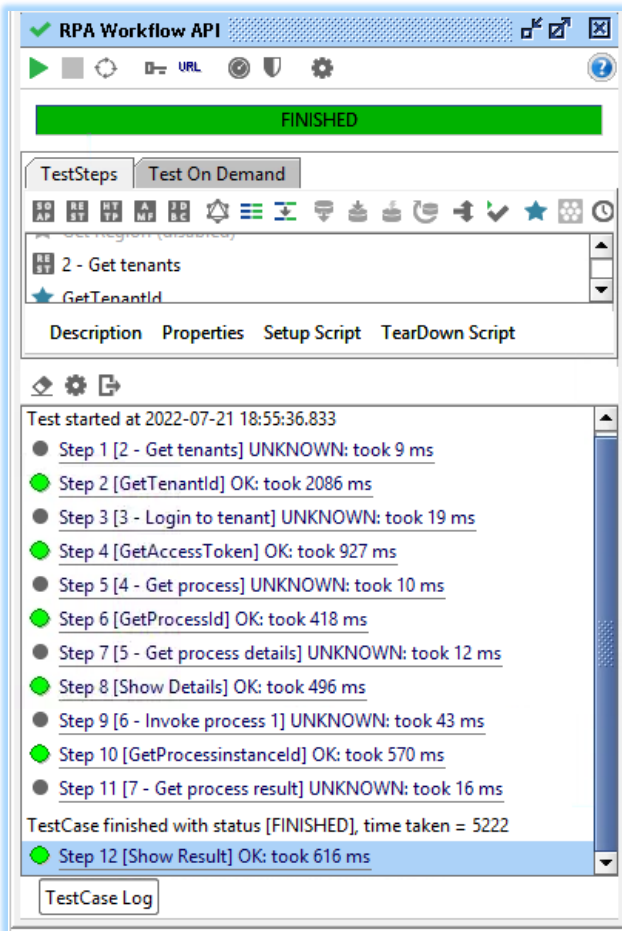
In the Projects tree, navigate to *Test Steps*. See below:



Double-click *Test steps* to open the *RPA Workflow API* test runner. Click *Run* to invoke the RPA API test suite. See below:



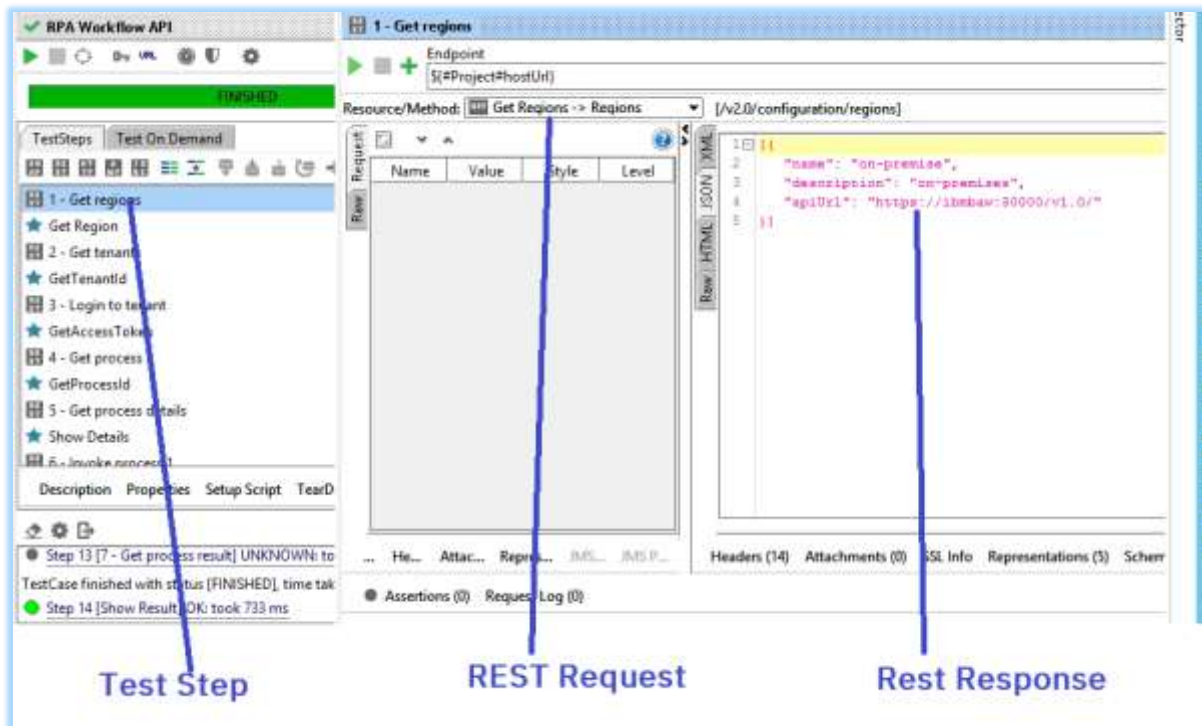
When running, you will be prompted for a series of dialog boxes showing you the API journey. When complete, you should see the following:





3.5.4 Examine the test suite flow.

The SoapUI test suite orchestrates the flow of execution by setting variables in one step and reading them in the next. You can verify this by double-clicking **step 1 – Get Regions**:



You can see the invocation request in the center panel and the invocation result on the right panel.

Tip: The result is JSON, so you need to select the JSON tab on the right panel.

Now select the next step, **Get Region**. This step is written in *Groovy Script*. It parses the JSON response from previous step and sets a project variable to be used in the next step. See below:



The screenshot shows the SoapUI interface with the 'Get Region' test step selected in the Test Steps list. The Groovy code for this step is displayed in the main editor. Two yellow boxes highlight specific parts of the code: one for parsing the JSON response into variables and another for setting a project property. Blue arrows point from text labels to these highlighted sections.

```
1 import com.eviware.soapui.model.testsuite.TestCaseRunContext
2 import net.sf.json.groovy.JsonSlurper
3
4 //Check if the response
5 assert context.response, 'Response is empty or null'
6
7 log.info("context.response: " + context.response)
8
9 def jsonResonponse = new JsonSlurper().parseText(context.response)
10
11 region = 0
12 regionName = jsonResonponse[region].name
13 regionUrl = jsonResonponse[region].apiUrl
14
15 if (regionUrl) {
16     regionUrl = regionUrl.replace("/v1.0/", "")
17     log.info("Found Region: " + regionName)
18     log.info("Found region Url: " + regionUrl)
19     testRunner.testCase.testSuite.project.setPropertyValue( "hostUrl", regionUrl )
20 }
21
22 def ui = com.eviware.soapui.support.UISupport
23 ui.showInfoMessage( "selected region is: " + regionUrl )
```

Parse Json Response into variables

Set project variable

Congratulations, you have just tested the RPA API with SoapUI.



3.6 Call Synchronous API from BAW

This section will build a BAW process to call the **synchronous** API from section 3.3.5. This lab reuses content from the async API lab. The bot from the async lab is called **ayncTestBot**. This is a misnomer for this section as we will be calling it synchronously.

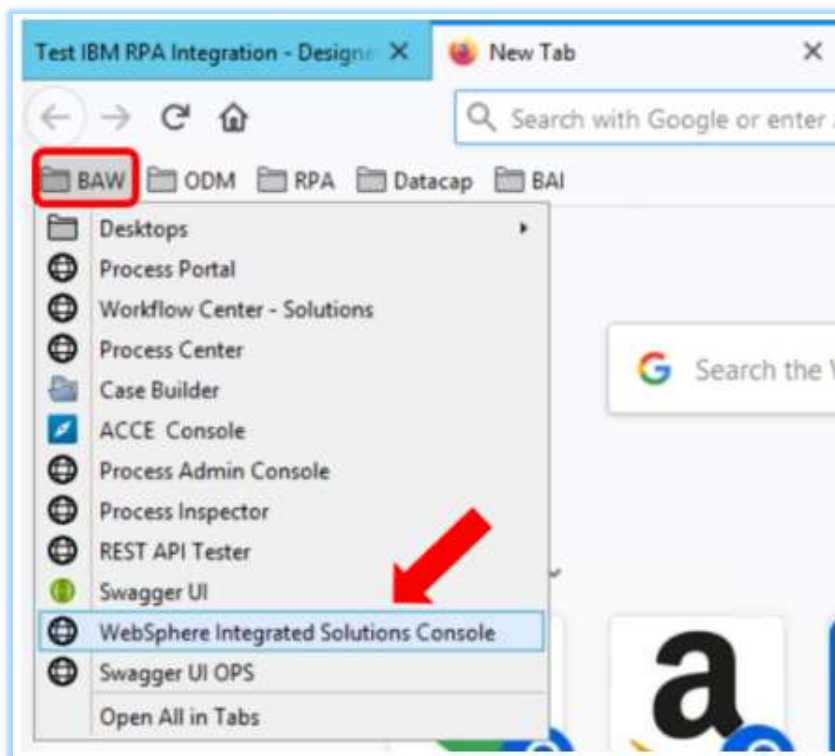
3.6.1 Import the signer certificate to BAW

If using the Skytap image we need to import the RPA certificate into BAW so that it is trusted.

With Firefox, open the **WebSphere Integrated Solutions Console** from the bookmarks toolbar (BAW → WebSphere Integrated Solutions Console)

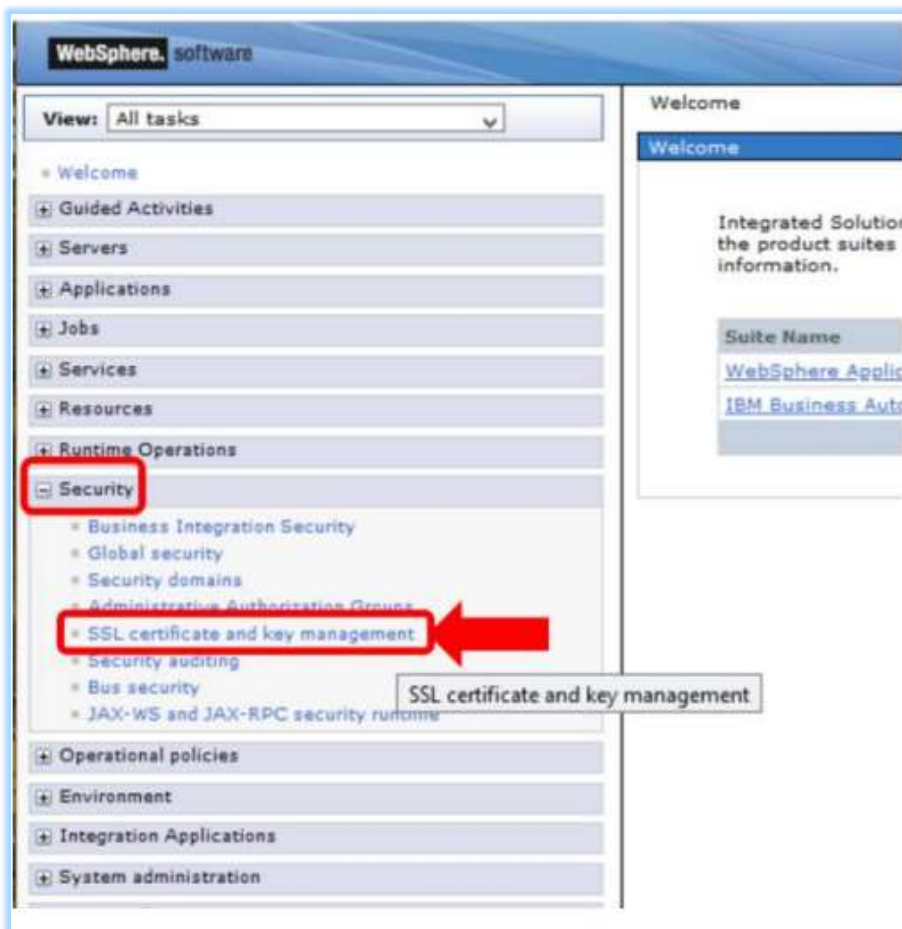
Tip: If you get an **Unable to connect** error, make sure you start BAW with the command:

```
c:\Desktop BU\RUNTIMES\BAW Server START.cmd
```

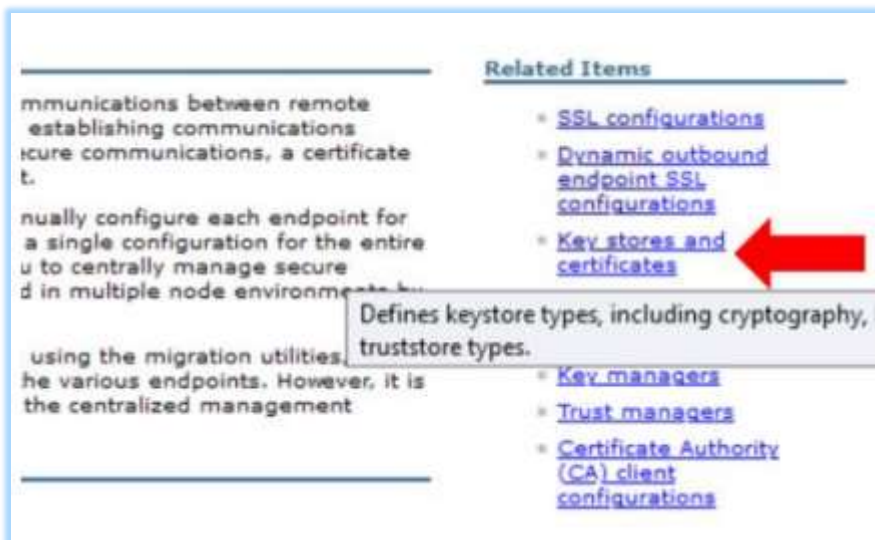


Log in with username **admin** and password **admin**.

When the console opens, expand **Security** and select **SSL certificate and key management** from the left-hand side menu.



From the page opened, click **Key stores and certificates**.



Next, click **CellDefaultTrustStore**.



SSL certificate and key management > Key stores and certificates
Defines keystore types, including cryptography, RACF(R), CMS, Java(TM), and all truststore

Keystore usages
SSL keystores

Preferences

New... Delete Change password... Exchange signers...

Select Name Description Management Scope

You can administer the following resources:

<input type="checkbox"/>	BlueWorksLiveTrustStore	Trust store for BlueWorksLive	(cell):PCCell1
<input type="checkbox"/>	CellDefaultKeyStore	Default key store for PCCell1	(cell):PCCell1
<input type="checkbox"/>	CellDefaultTrustStore	Default trust store for PCCell1	(cell):PCCell1
<input type="checkbox"/>	NodeDefaultKeyStore	Default key store for Node1	(cell):PCCell1:(no
<input type="checkbox"/>	NodeDefaultTrustStore	Default trust store for Node1	(cell):PCCell1:(no
<input type="checkbox"/>	PublicInternetTrustStore	Trust store for public Internet SSL settings	(cell):PCCell1

Click **Signer certificates** to open view for signer certificates for the CellDefaultTrustStore. We have already some previously imported certificates here.

CellDefaultTrustStore
Java(TM), and all truststore types.

Addition of Properties

- Signer certificates
- Private key
- Public key
- Private key alias
- Public key alias
- Custom attributes

Manages signer certificates in key stores.

Preferences

Add Delete Extract Retrieve from port

Select Alias Issued to Fingerprint (SHA Digest)

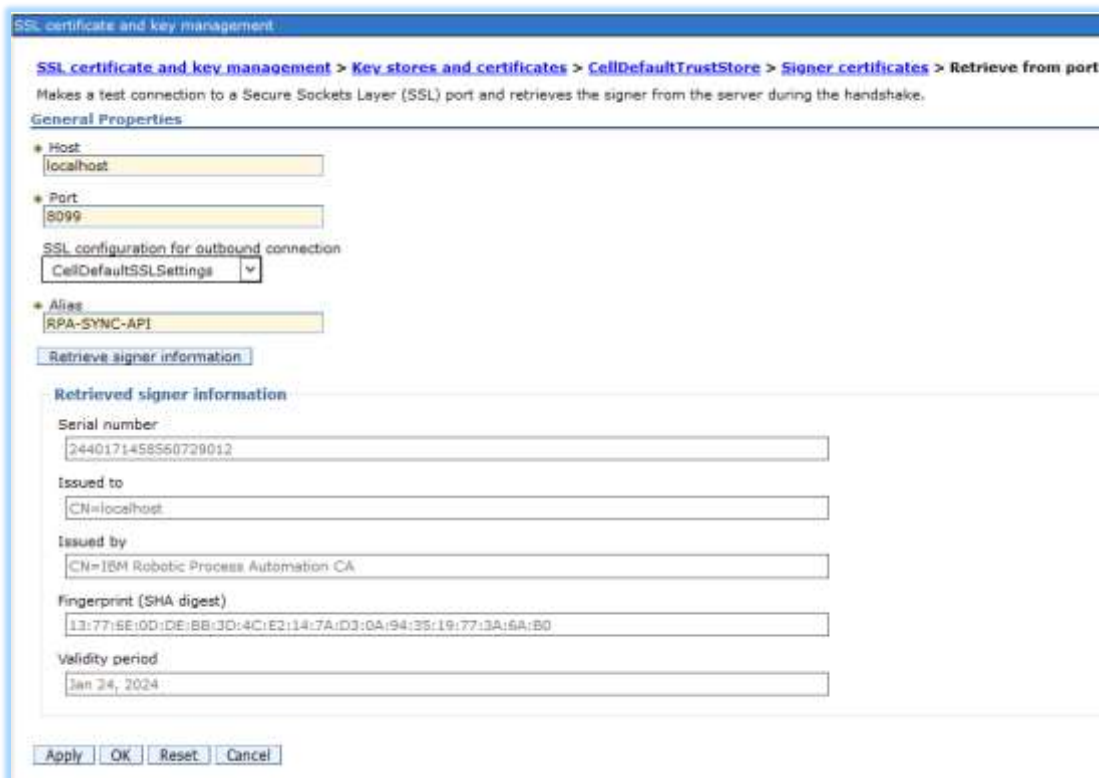
You can administer the following resources:

<input type="checkbox"/>	cell.default.truststore	CellDefaultTrustStore	A5:43:C1:5C:5B:1B:6C:EE:8D:3F:83:8D:1A:FB:4B:FD:0F:C1:6B:43
--------------------------	-------------------------	-----------------------	---

Click the Retrieve from port -button.



This opens a wizard to get the signer certificate. Type in **localhost** for host, **8099** for Port and **RPA-SYNC-API** for Alias. When the values are set, click the **Retrieve signer information** button. The signer information is shown below. Click **Apply** and then **Save** to save the signer certificate.



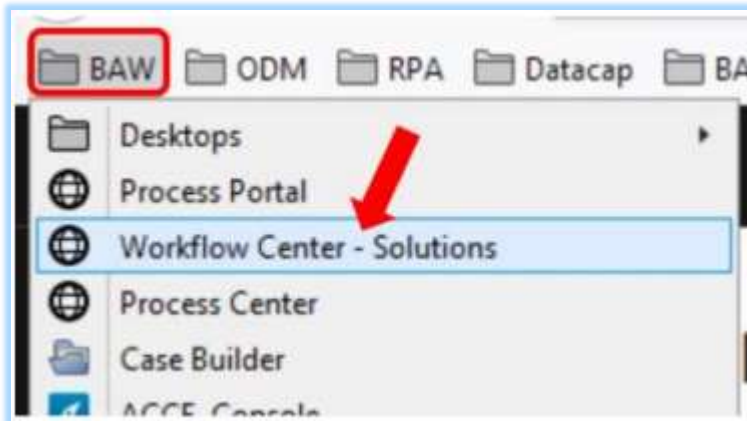
You can now close the WebSphere console.



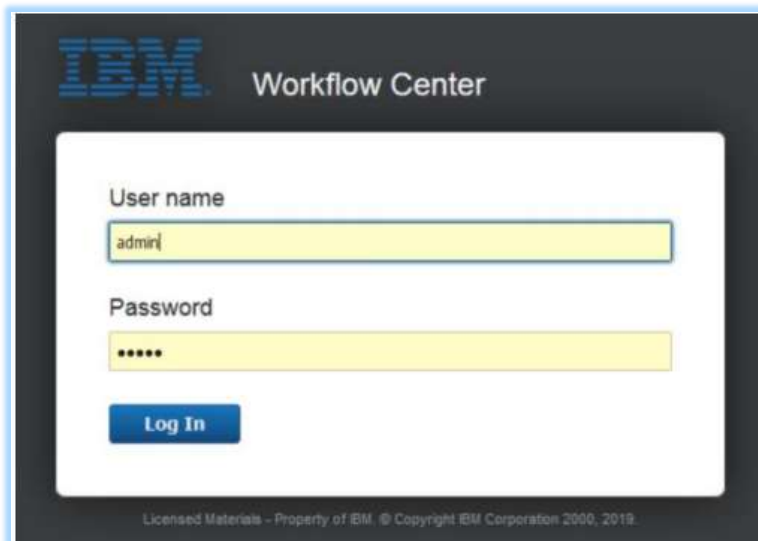
3.6.2 Create BAW Process Application for testing the Synchronous RPA API

To test the RPA Synchronous API, we will import a simple BAW Process Application.

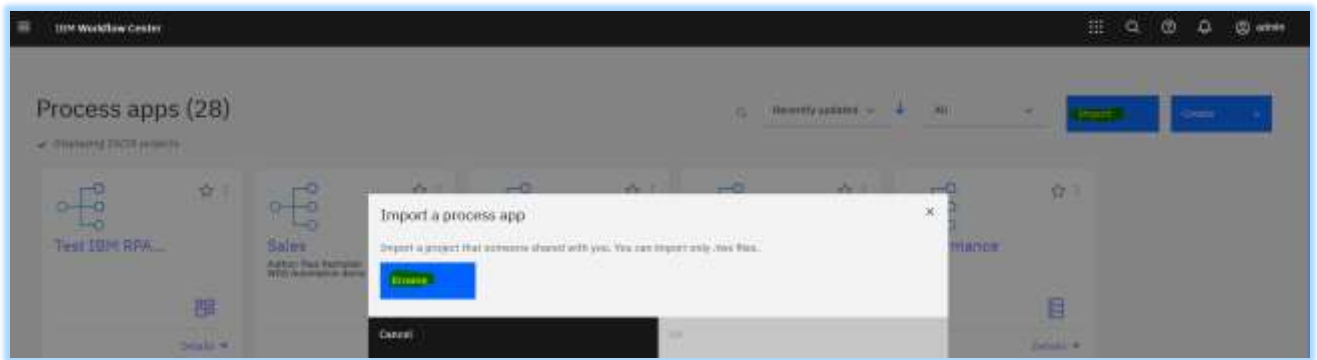
Use Firefox bookmarks toolbar to select **BAW → Workflow Center – Solutions** to open the BAW Workflow Center.



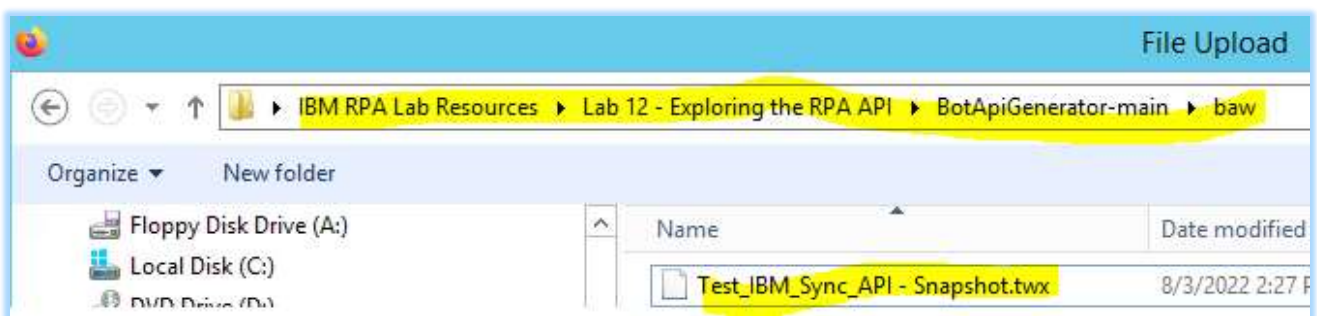
Log in as **admin** with password **admin**.



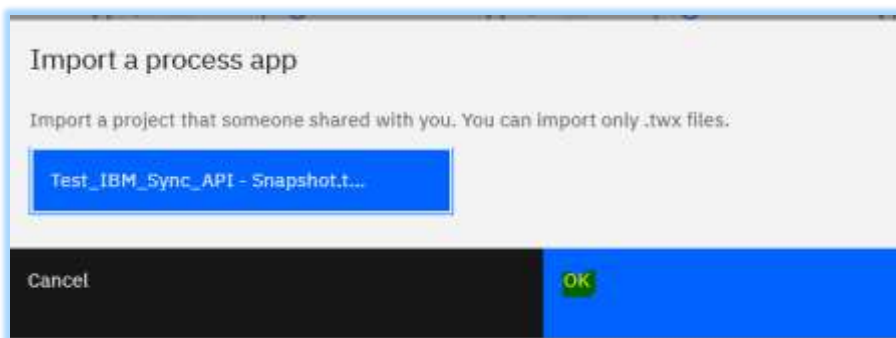
Select **Process apps**. Click **Import** and then **Browse**:



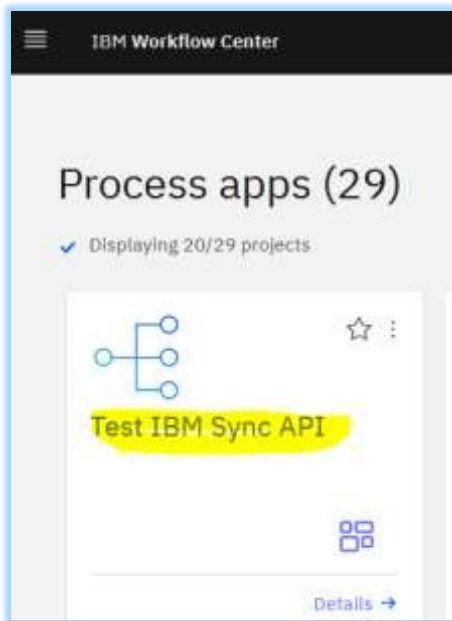
Select the twx file from the Git folder you extracted earlier:



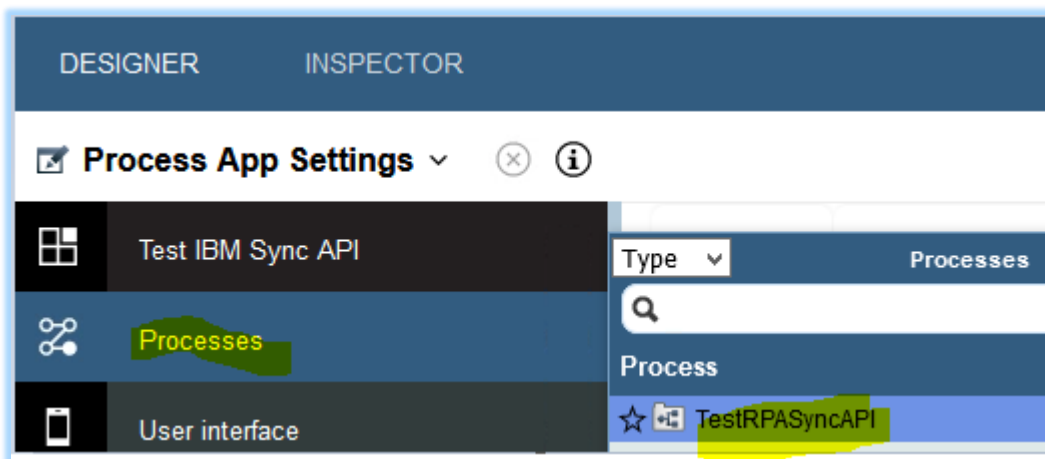
Press OK



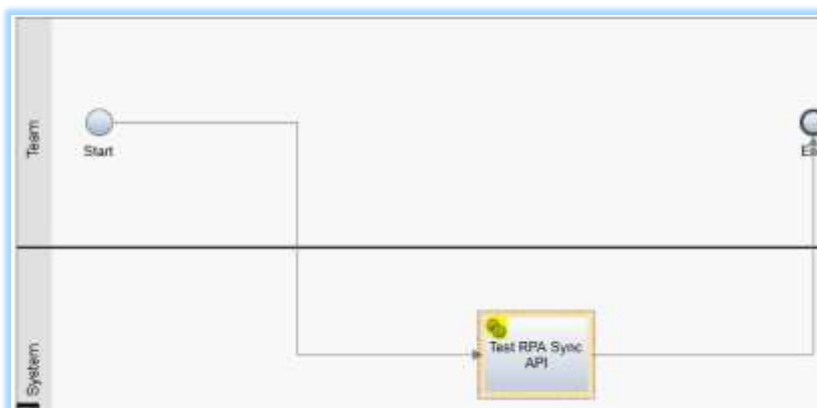
Select the process to edit it:



Select **Processes->TestRPASyncAPI**:



Drill into the **Test RPA Sync API** task by double clicking the cog icon on the top left



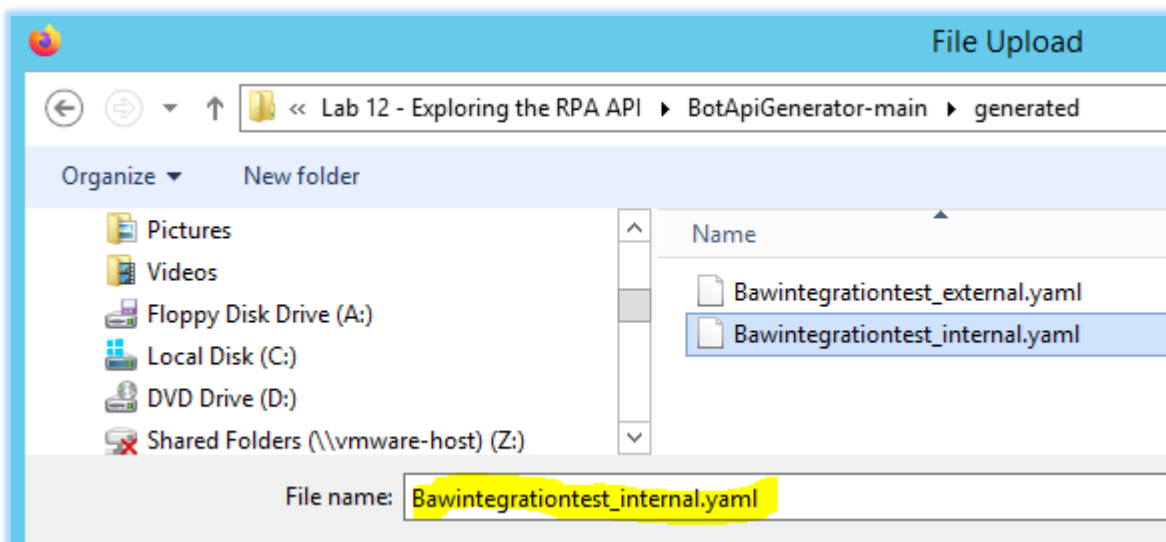


Now select the **Invoke RPA Sync API Service Task** and select the **Implementation** tab on the left. You should see this:



Select **New...** as highlighted on the above diagram.

Select the **Bawintegrationtest_internal.yaml** Open API we generated in section 3.3



Click **Next>**



New External Service

An external service lets you call a service or application that is external to IBM Business Automation Workflow. [Learn More](#)

Select a method to discover the service.

REST service from local file

External service name: Bawintegrationtest_internal

File name: Bawintegrationtest_internal.yaml

< BACK NEXT > FINISH CANCEL

Click **Next** again and then **Finish**

You should see the API has been successfully imported into BAW. Expand **BAWIntegrationTest**. Verify that the data structures to invoke the API have been created.

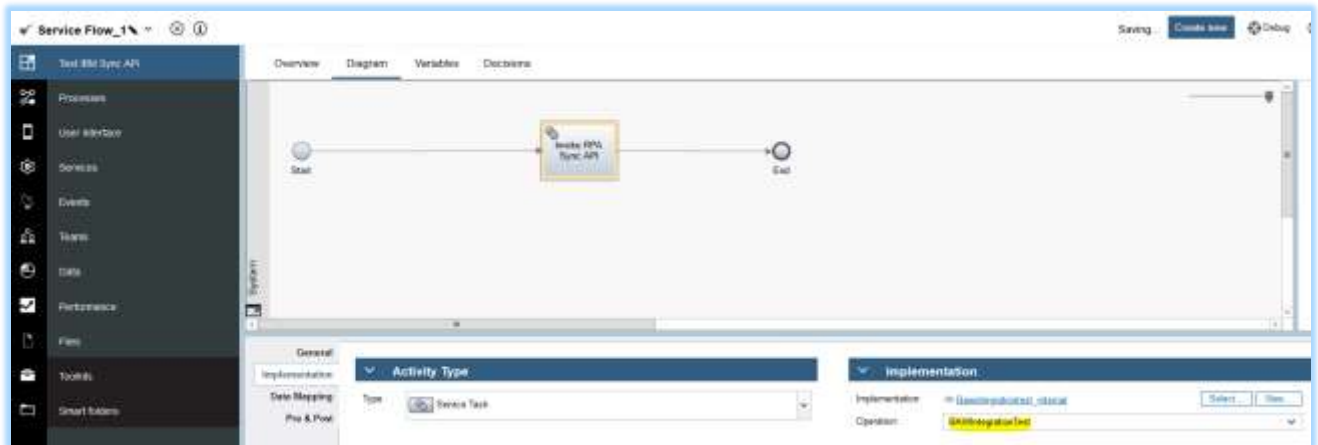
External Service

- ▼ Bawintegrationtest_internal
 - ▼ BAWIntegrationTest
 - ▼ Input
 - unlockMachine (String)
 - input_text (String)
 - ▼ Output
 - BAWIntegrationTest_200 (ApiResponse)

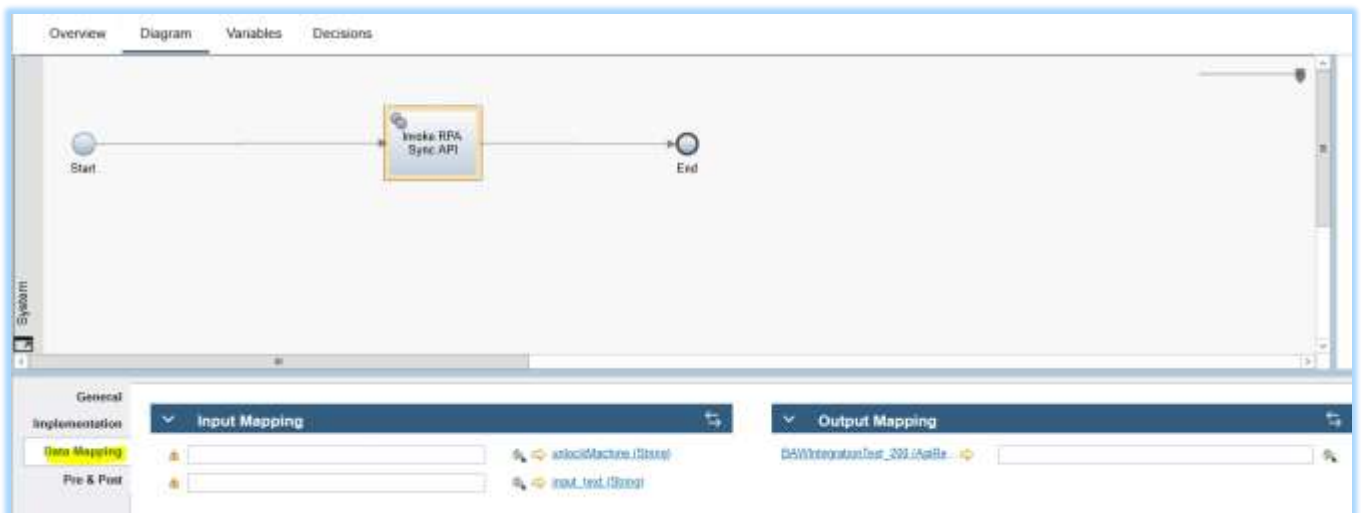
The final step is to test the API. Navigate back to the Service Flow. The quickest way is to close the current pane;l:

Bawintegrationtest_internal ▼ × i

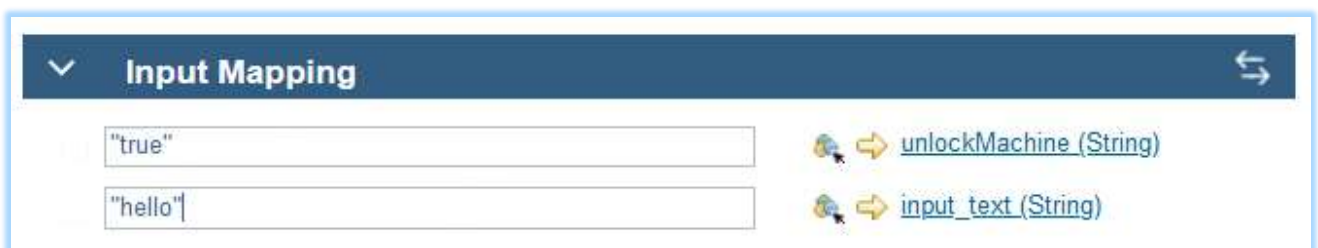
Back in the Service Flow, select the Operation as **BAWIntegrationTest**:



Select the Data Mapping tab as shown below:



Set the input mapping:



Now generate the output variable by clicking on the double arrow icon highlighted:



Click the output box and press **Finish**



Variable Creation

Create variables where no matching variable exists. The new variables are automatically mapped. Existing mappings are not overwritten. Existing variables with the same name but different types are omitted.




Select the variables to be created and auto-mapped. By default, the variables are created as private variables. To create them as input, output, or input and output variables, select the check box beside the variable.

<input checked="" type="checkbox"/> Variable Name	Variable Type	Input	Output
<input checked="" type="checkbox"/> BAWIntegrationTest_200	ApiResponse	<input type="checkbox"/>	<input checked="" type="checkbox"/>

To test the API, press Debug:

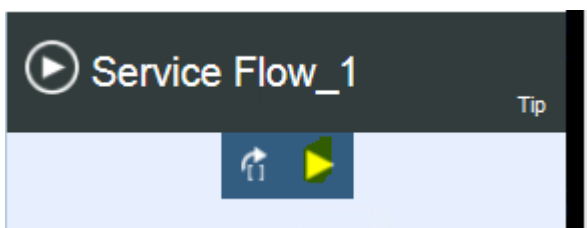
Last saved seconds ago by you

Create new

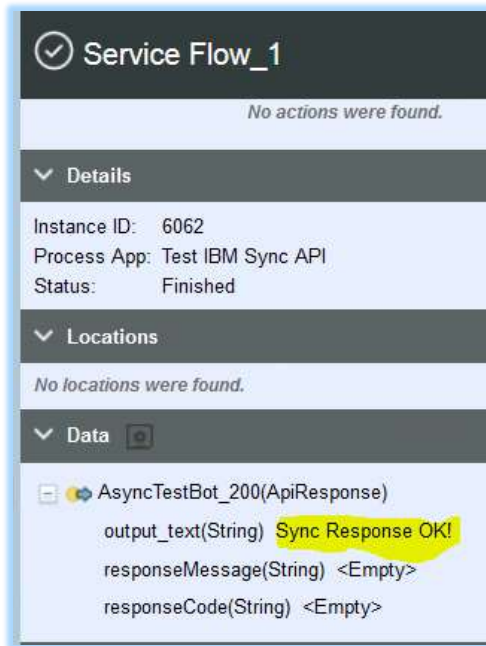
 Debug

 Run

Now press **Run**:



Enable popups. You should see the message “hello” appear from the Bot window. In the bot output window, type “Sync Response OK!” After pressing OK, check the response:



Tip: If your bot does not run, you may need to bounce the BAW server by performing:

```
c:\Desktop BU\RUNTIMES\BAW Server STOP.cmd
```

Wait until complete, then:

```
c:\Desktop BU\RUNTIMES\BAW Server START.cmd
```

Congratulations, you have invoked a bot from BAW using the Async API.



4 Integrate with the World!

In this lab you examined the synchronous and asynchronous RPA API. You called the API using OpenApi, PowerShell, Curl, SoapUI and JUnit.

Now that you have completed this lab, you can start creating your own integration. The generator provided in the lab could help create an API framework for your bots.

If you have feedback, please contact:

ncrowther@uk.ibm.com

This concludes the lab.