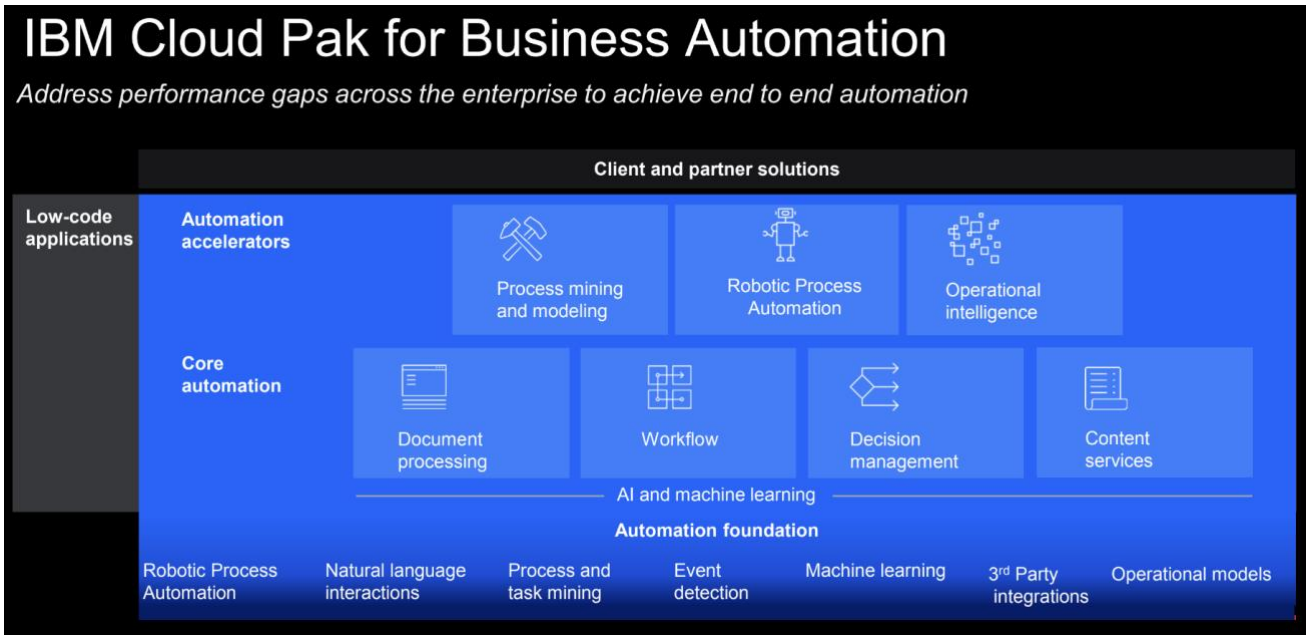
Using IBM RPA from BAW Processes - BAW Toolkit

Lab Guide V1.1 (2022-07-07)



Jukka Juselius

IBM RPA Technical Lead – EMEA

jukka.juselius@fi.ibm.com

Notices and disclaimers

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

This document is current as of the initial date of publication and may be changed by IBM at any time. Not all offerings are available in every country in which IBM operates.
Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information.

This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity. IBM products and services are warranted per the terms and conditions of the agreements under which they are provided. The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

IBM products are manufactured from new parts or new and used parts.
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled,
isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to ensure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers (Continued)

Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.** The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corporation, 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.

Table of Contents

1 INTRODUCTION 5

1.1 IBM BAW 5
1.2 IBM RPA 5
1.2.1 SYNCHRONOUS API 6
1.2.2 ASYNCHRONOUS API 7
1.3 COMMUNITY BAW TOOLKIT FOR USING IBM RPA 7

2 LAB SETUP 8

2.1 GETTING YOUR TEST ENVIRONMENT 8
2.2 STARTING BAW 12
2.3 CREATING AN RPA SCRIPT FOR TESTING AND PUBLISHING IT 12
2.4 SET UP RPA PROCESS 18
2.5 SET UP YOUR COMPUTER DEFINITION 22
2.6 IMPORT THE SIGNER CERTIFICATE TO BAW 25

3 CREATE BAW PROCESS APPLICATION FOR TESTING THE RPA API 29

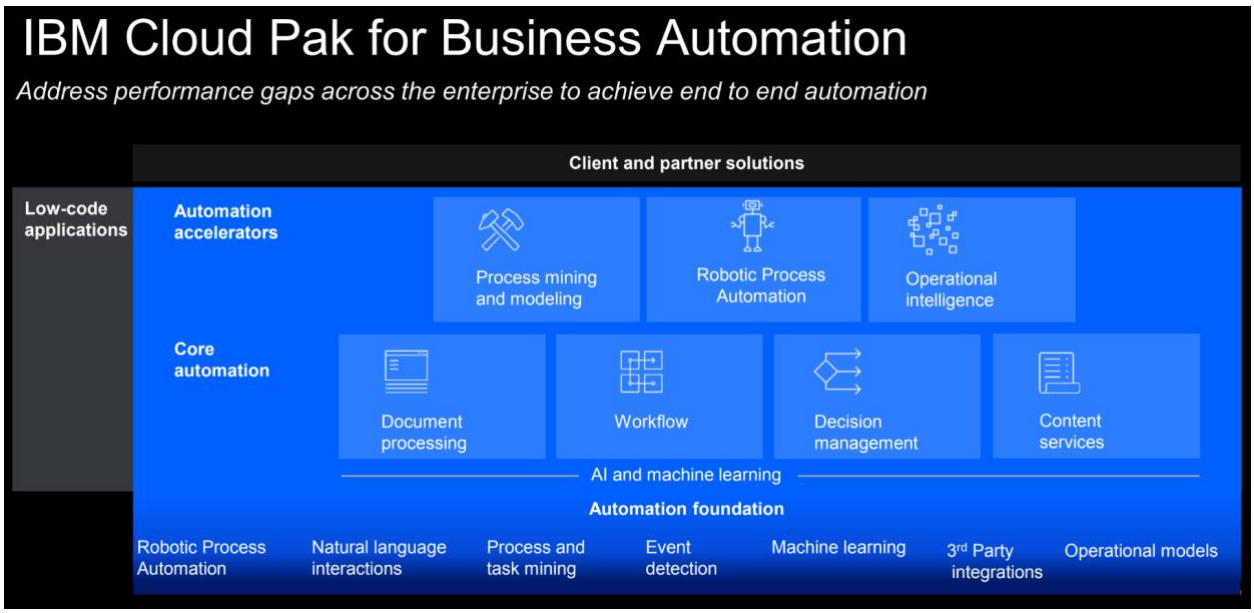
3.1 DOWNLOAD AND IMPORT THE TOOLKIT 29
3.2 IMPLEMENT PROCESS APPLICATION TO TEST IBM RPA INTEGRATION 32

4 TESTING THE INTEGRATION 47

4.1 RUN THE BAW PROCESS 47
4.2 PARSING THE BOT OUTPUT JSONSTRING (OPTIONAL) 51

1 Introduction

Both Workflow and Robotic Process Automation (RPA) are foundational business automation capabilities in IBM Cloud Pak for Business Automation. Synergy between these capabilities is clear, as RPA can trigger more complex end-to-end workflows when needed and Workflows to use RPA as part of end-to-end workflows to automate some of the activities that cannot be automated using the existing system services / APIs.



This lab will focus on how to use RPA as part of Workflows. How to trigger RPA bots as part of a workflow and how to get results back from them.

1.1 IBM BAW

IBM Business Automation Workflow (BAW) offers a holistic approach to manage and automate different types of workflows. It supports model-based processes for straight-through processing and human assisted workflows, but also case type of processes that are typically run and managed by a knowledge worker and might include a lot of different types of documents gathered and stored during the handling of the case.

More information on BAW can be found from the following links:

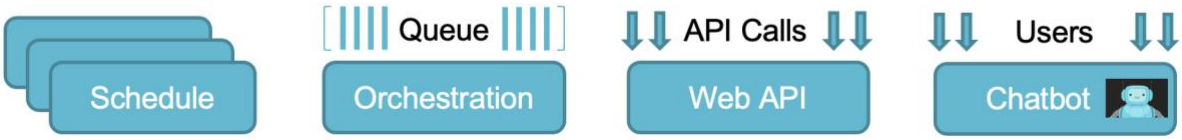
- Technical documentation: https://www.ibm.com/docs/en/baw/22.x
- Product page: https://www.ibm.com/products/business-automation-workflow
- IBM Community: https://community.ibm.com/community/user/automation/communities/community-home?CommunityKey=810abde6-3916-441b-aac3-b9105bb37e3c
- BAW in IBM Seismic: https://ibm.seismic.com/Link/Content/DCRFm2Cjhb3gHGcWJcjcFD7QD84P

1.2 IBM RPA

IBM RPA offers multiple different ways to run an RPA bot / automation. Perhaps the most common way is scheduling, but IBM RPA also allows users to create bot orchestrations – called *RPA Processes* – that will make use of underlying messaging infrastructure. Besides these options IBM RPA exposes two different APIs – synchronous and asynchronous – that can be used to trigger bots from external processes. Also running bots on-demand on business user's desktop (attended automation) and exposing bots as chatbots are supported.

IBM RPA offers several ways of starting unattended bots

- ✓ Scheduling
  - Defined using Control Center (tenant management console)
- ✓ Orchestration (Process)
  - Defined using Control Center, uses queues and arriving messages as triggers
- ✓ Web API x 2 (synchronous and asynchronous API)
  - Callable REST / SOAP API to trigger bot on-demand
  - Synchronous API exposed by every bot runtime agent
  - Asynchronous API exposed by RPA Server
- ✓ As exposed chatbot
  - Integrated to channel application and triggered whenever called



The two APIs that IBM RPA exposes are quite different in nature and therefore the usage of the APIs also differs.

1.2.1 **Synchronous API**

The synchronous API is exposed by any IBM RPA runtime agent you install. You can use it to run any bot that has been published to same IBM RPA tenant (RPA Server) the agent is connected to. Even though it is quite straightforward to use it has some limitations:

- No security. The API is open for anyone that reach the agent machine API port. Therefore, the security is totally dependent on the infrastructure and communication policies (networking and firewall rules).
- Limited load-balancing. The API does not really provide any means to distribute load it receives. The only thing that can be done is use Broker definition attached to the script that is being triggered. This can act as a simple failover mechanism that you can use to define list of computers (bot runtime environments) that should be used to run the bot triggered via API. This does not include any active load balancing algorithm, but it finds an available runtime from the list, if one exists, always reading the list though from start to finish.
- Synchronous invocation is not a proper way to invoke most of the RPA bots. It is quite normal that RPA bots will run for several minutes – depending on the work they do – and that might cause a timeout when the calling party is waiting for the bot to come back with some sort of a result. Synchronous API should be used to run bots that we know are executing in seconds and we're expecting to get a result back from them almost immediately. These can be for example bots retrieving or storing some simple data from one system.

1.2.2 **Asynchronous API**

IBM RPA asynchronous API is more flexible to use than the synchronous one. First, it is exposed by the RPA Server, not by the bot runtime environments. It has security and quite sophisticated workload management options that rely on IBM RPA orchestration process definition.

Secondly, as mentioned already earlier, some RPA bots can run for several minutes and make no sense to trigger them synchronously. Most of the RPA vendors support only asynchronous API for triggering bots, but as already explained, IBM RPA has both synchronous and asynchronous one.

In this lab, we will be focusing on using the asynchronous API.

1.3 Community BAW Toolkit for using IBM RPA

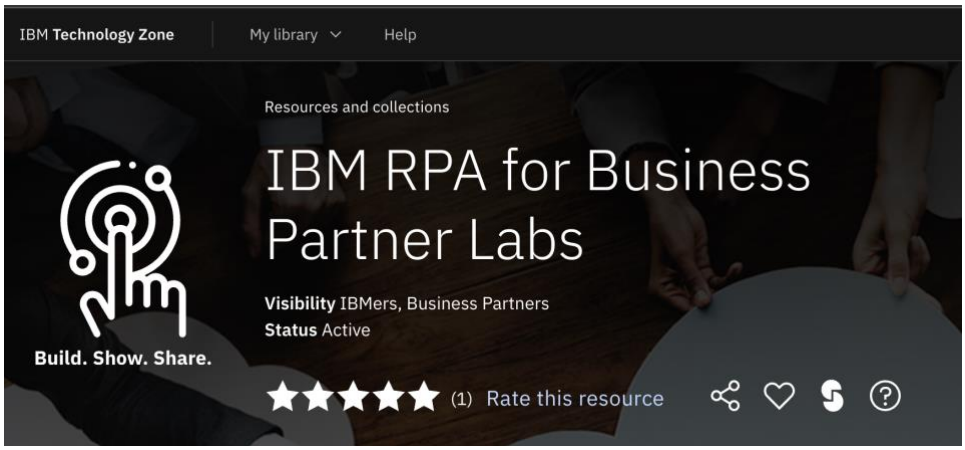
Currently (2022-07) we do not have any out-of-the-box integration functionalities to call out IBM RPA from IBM BAW. That's why we have provided a community asset in a form of a BAW Toolkit that can be used to establish the integration. We will use the toolkit in this lab and show how it can help to configure a working integration between BAW and RPA.

The toolkit itself is documented here: https://github.com/juseljuk/IBM-RPA-Toolkit-for-BAW

2 Lab Setup

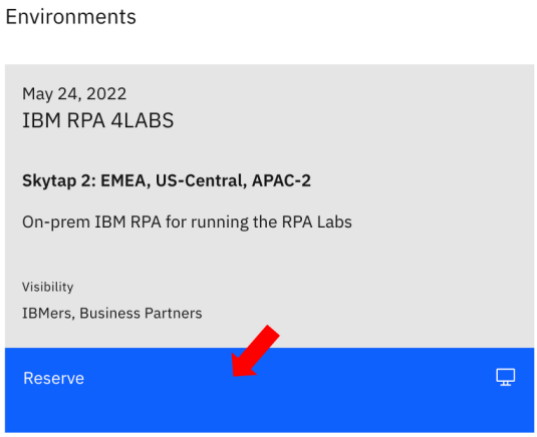
2.1 Getting your test environment

You will be able to provision your test environment from IBM Technology Zone. Just head to https://techzone.ibm.com/collection/ibm-rpa-for-business-partner-labs.

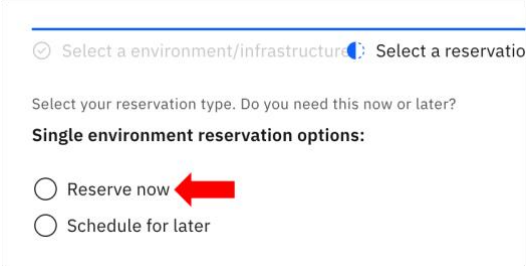


Note! All IBMers and IBM Business Partners that are linked to their partner profiles can access Technology Zone.

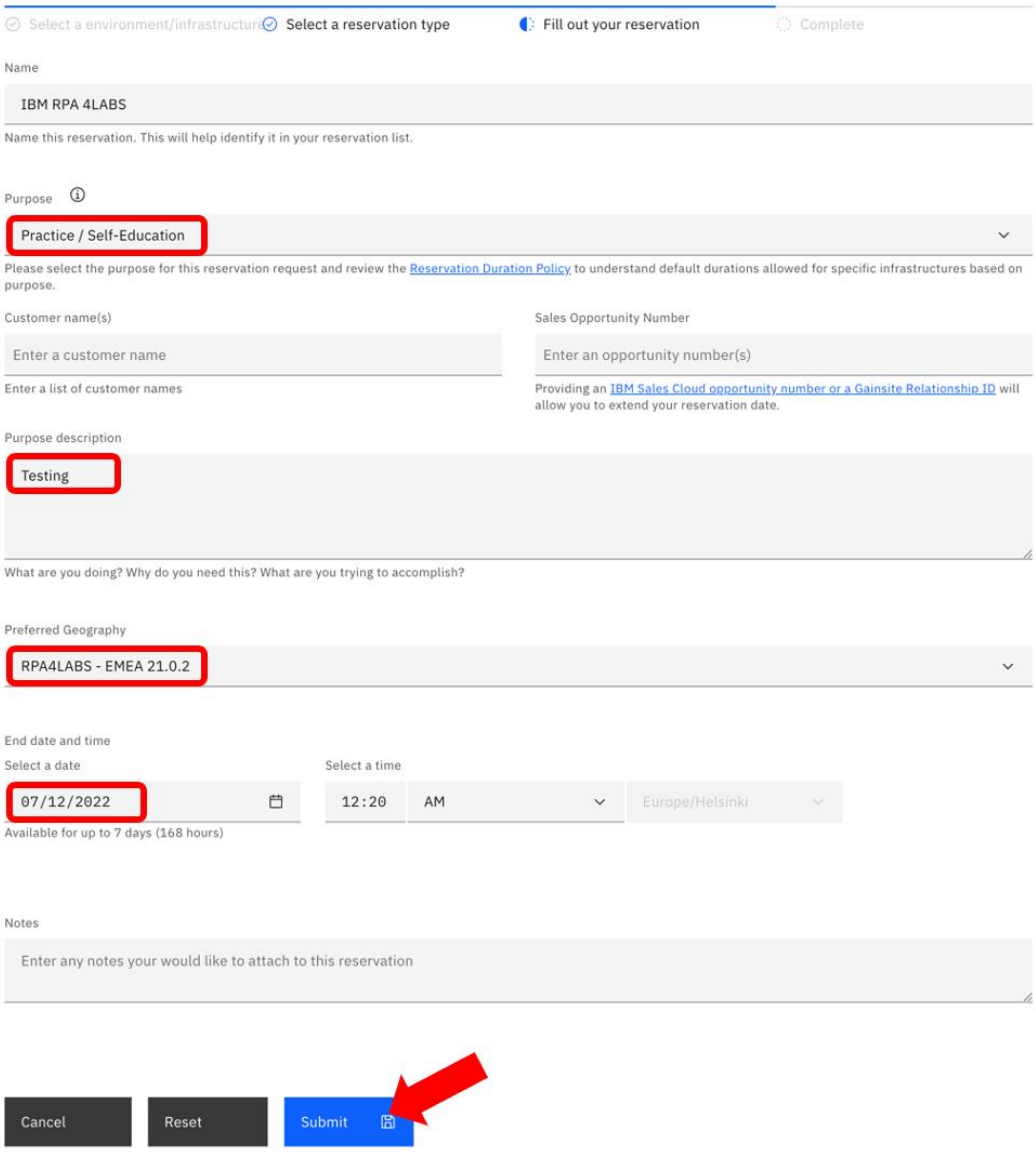
1. Scroll down to see the Environments section. Click **Reserve**. This opens *Create a reservation* page.



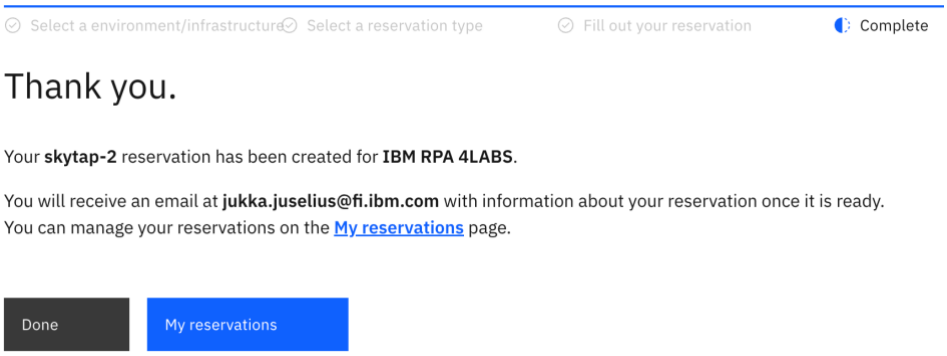
2. Select **Reserve now**.



3. Select the **purpose**, give it a **description**, select your **preferred geography** and **set the date and time** for your reservation to end. Then click **Submit**.

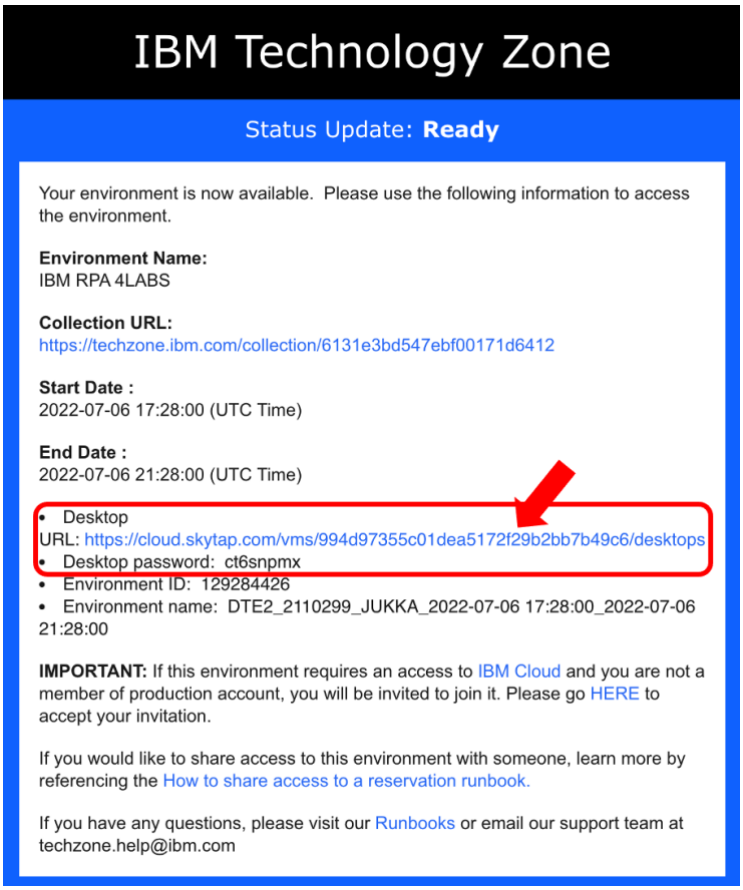


4. Submitting takes a couple of seconds, but you should soon see that your reservation has been created.



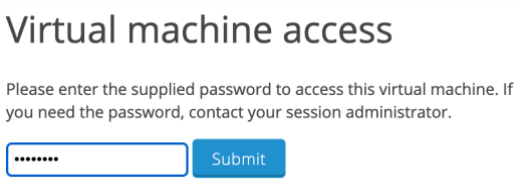
5. You will also instantly get an email informing you that your environment is ***provisioning***. It should not take more than 5 – 10 minutes to get another email with the needed information to access to access your environment once it has been provisioned. So, keep your eye on your emails 🙂

6. The second email looks like this:

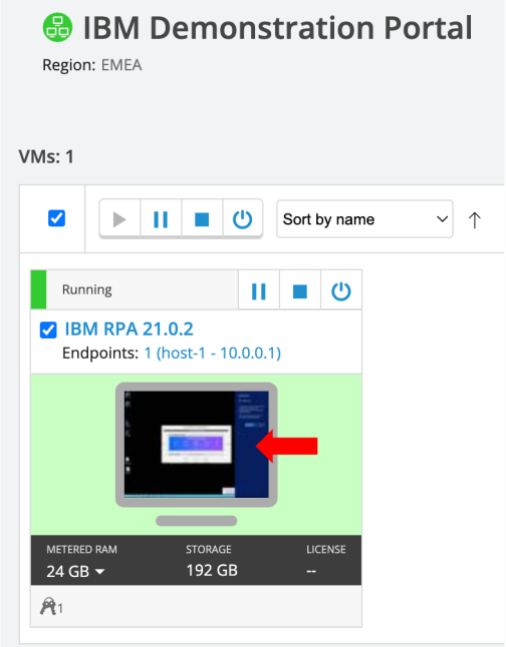


The most important pieces of information are the link to your virtual environment (Desktop URL) and the password (Desktop password) that you need when accessing it.

7. Click the **Desktop URL link** and apply the **password** when asked. Click **Submit** to continue.



8. This will lead you to your IBM Demonstration Portal, where your virtual machine should already be up and running. If the VM is not running, use the controls (the “play button”) to start it. It will take a minute or two for the VM to start up and the status change to *Running*.



9. **Click the "display picture"** of your running VM to access the virtual desktop of the VM. The virtual desktop will be open in your browser. Recommended browsers to use are Firefox and Google Chrome.

10. If the Networks side bar is still visible, you can click **Yes** to close it.

11. **Notice** the toolbar on the top of the virtual image. You can use it to control some of the settings for your virtual desktop. More information here: https://help.skytap.com/accessing-vms-with-a-browser.html

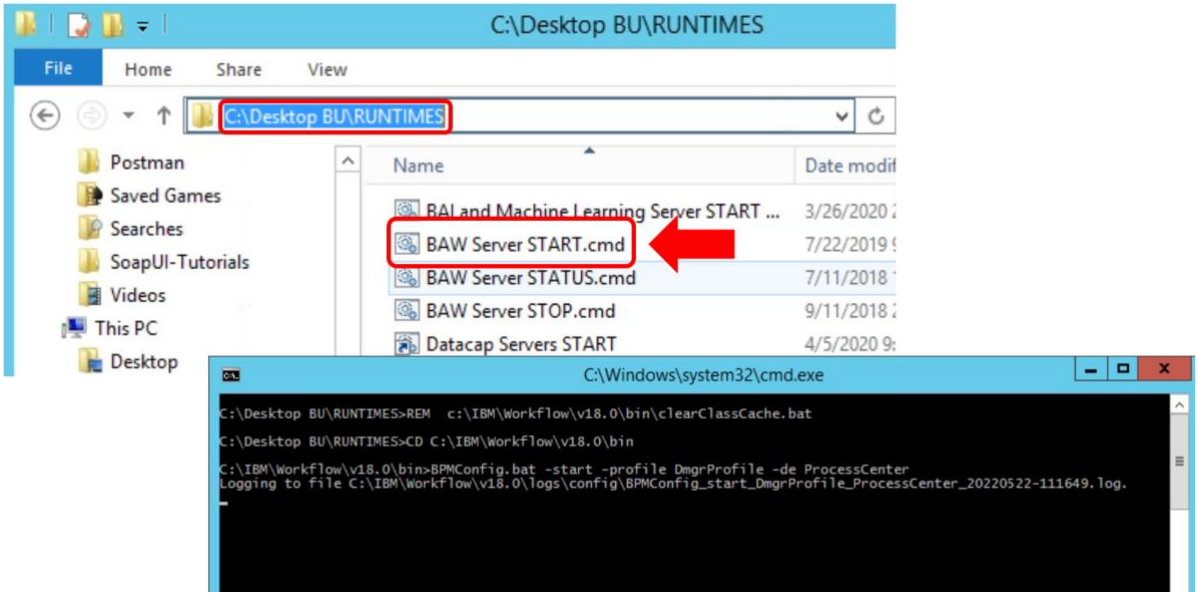
That's it, you're good to go!

2.2 Starting BAW

We will start BAW installed to the image.

12. Open **Windows File Manager** by clicking the icon from your taskbar.

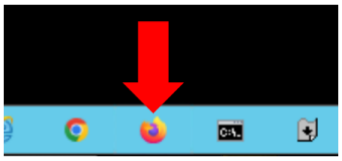
13. Navigate to **C:\Desktop BU\RUNTIMES** folder and double-click the **BAW Server START.cmd**. You will see a CMD window opening and BAW starting. Good, you can leave BAW to start (will take a couple of minutes) and move forward with the instructions. Just make sure **NOT** to close the CMD window.



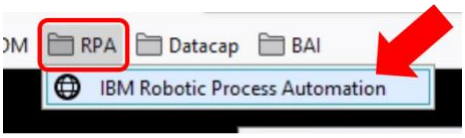
2.3 Creating an RPA script for testing and publishing it

Next, we will create a simple RPA script that we then use for testing the RPA API. We also show how to setup variables that are expected as inputs and variables that we want to use as outputs for the script.

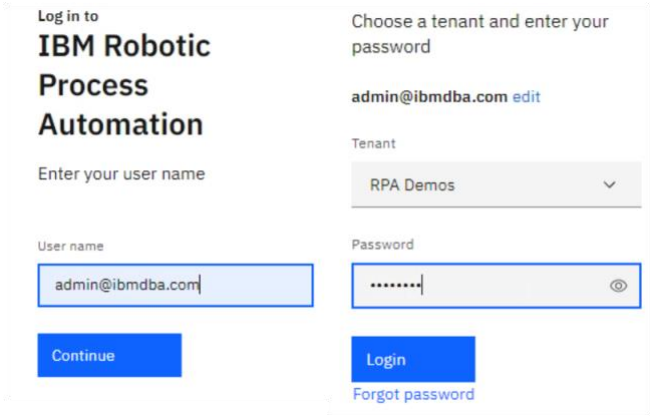
14. Before we start the actual bot development, we need to add correct roles for the user that we're using (admin@ibmdba.com). Open **Firefox** browser using the icon from your task bar.



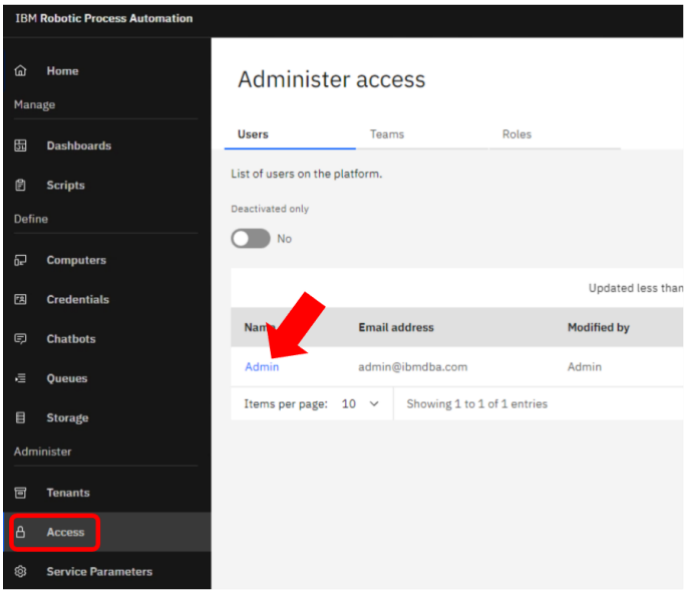
15. From the Bookmarks toolbar select **RPA → IBM Robotic Process Automation**.



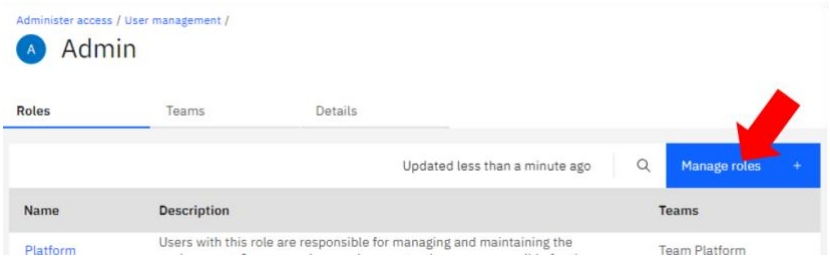
Log in as ***admin@ibmdba.com*** with password ***passw0rd*** (o as number zero).



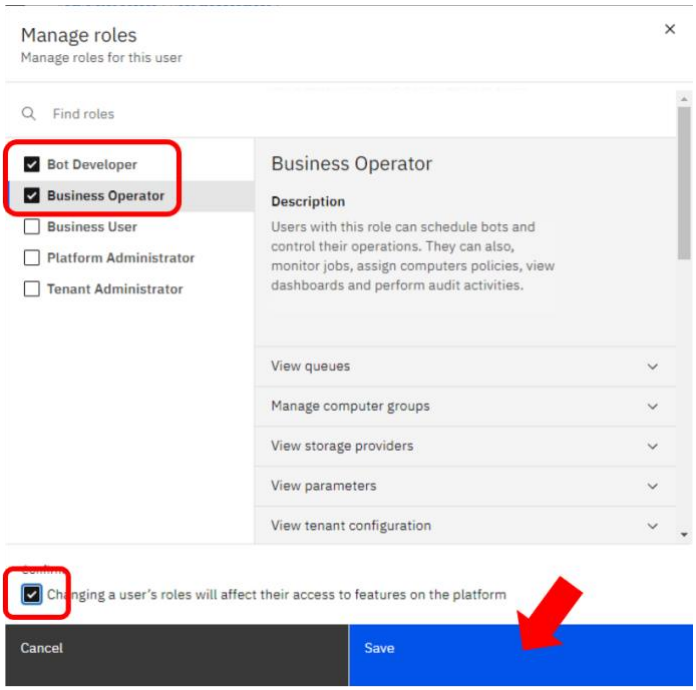
16. Select **Access** from the right-hand side menu bar and then click **Admin** from the listed users (we only have one user declared to this tenant).



17. This opens the user configuration for our Admin user. Click **Manage roles** button.



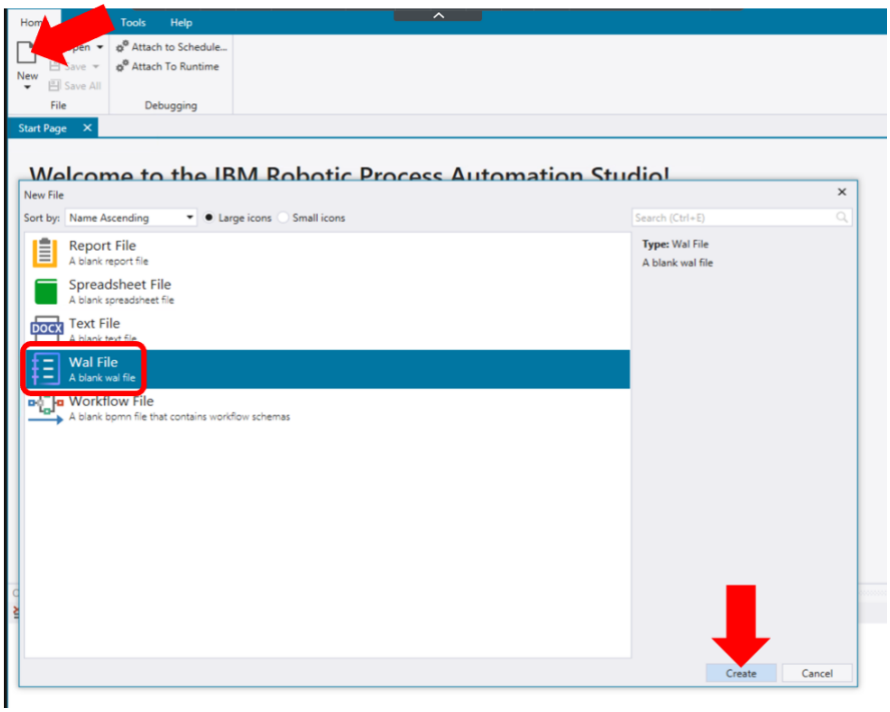
18. Select both **Bot Developer** and **Business Operator** roles for the user. Make sure to check **the check box under Confirm** and then click **Save** to save the user settings. Leave the browser open, you'll need it soon again.



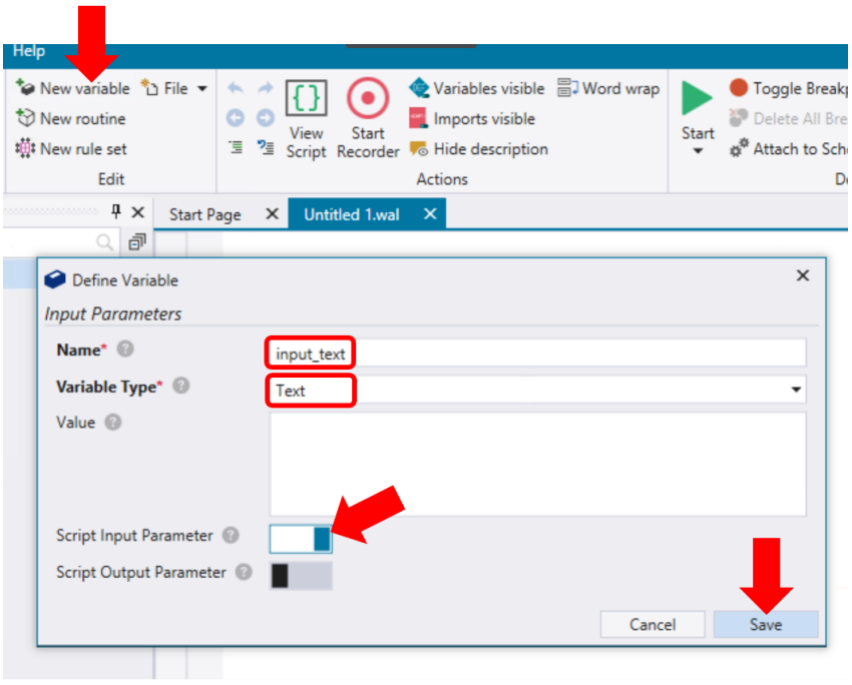
19. Start your IBM RPA Studio by double-clicking the shortcut on your desktop.

20. Login using ***admin@ibmdba.com*** as your username and ***passw0rd*** (o as number zero) as your password. **NOTE!** *After entering your username, you need to hit ENTER or click the login button in order to type in your password.*

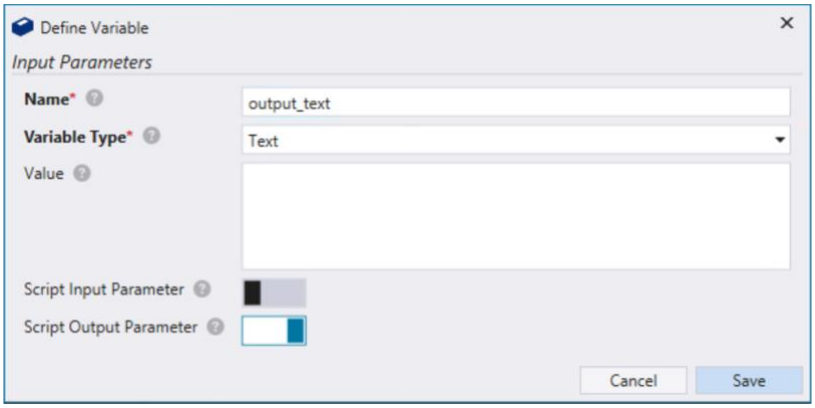
21. When Studio opened, click the **New** icon from your top toolbar, select **Wal File** and click **Create**. New and empty script will open.



22. Let's first create the variables for our automation. Click **New variable** selection from your top toolbar. Define variable name **input_text**, type **Text** and make sure to enable the **Script Input Parameter** switch. This will declare the variable as expected input variable for the script.



23. Now, repeat the previous step to create an output variable named **output_text** as well. Remember to enable **Script Output Parameter** switch.

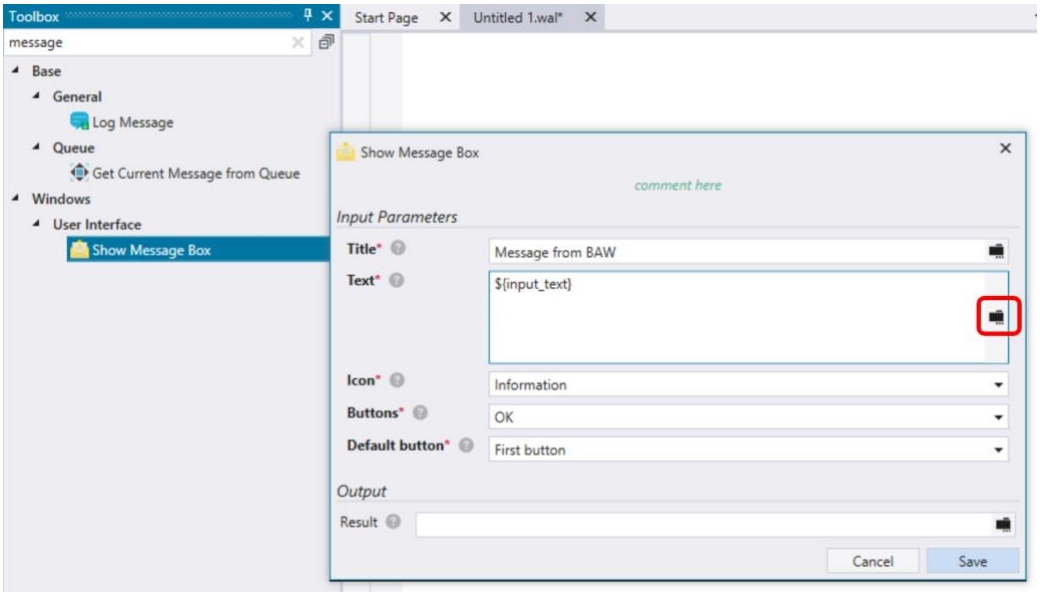


24. When the variables are defined, we can implement our test script logic. Let's keep it simple.

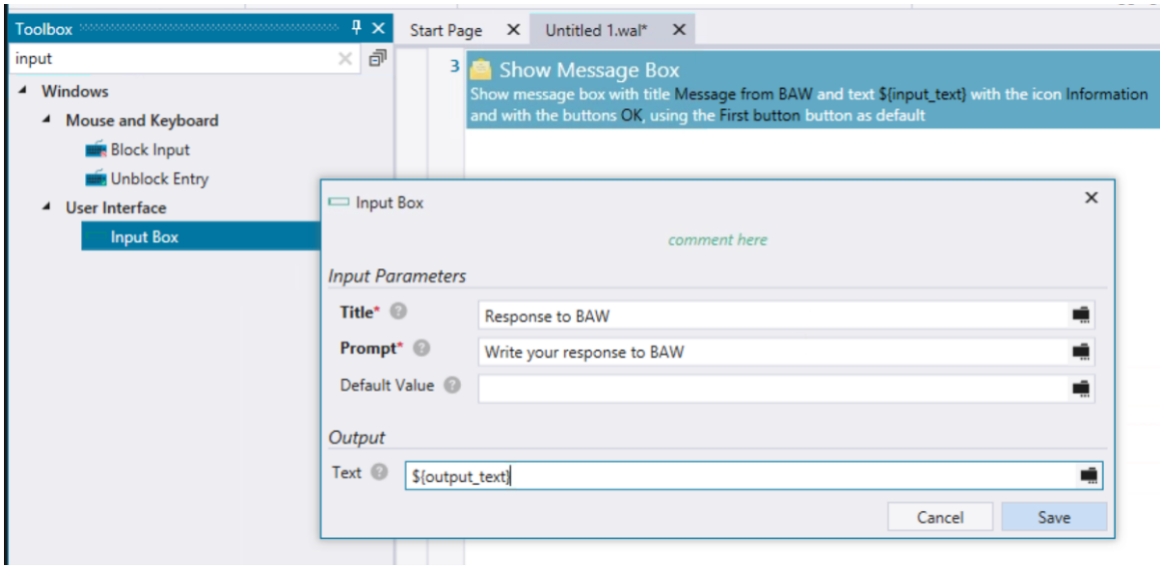
    Type in "**message**" to Studio's Toolbox to bring up the **Show Message Box** command. Drag and drop it to your empty script area or double-click it to open the configuration window for the command. Configure the command as follows (leave other configuration fields as they are):

    - Title: **Message from BAW**
    - Text: **${input_text}** (select the input_text variable by clicking the variable selection icon from the right-hand side of the Text input field)

    Click **Save** to add the command to your script.

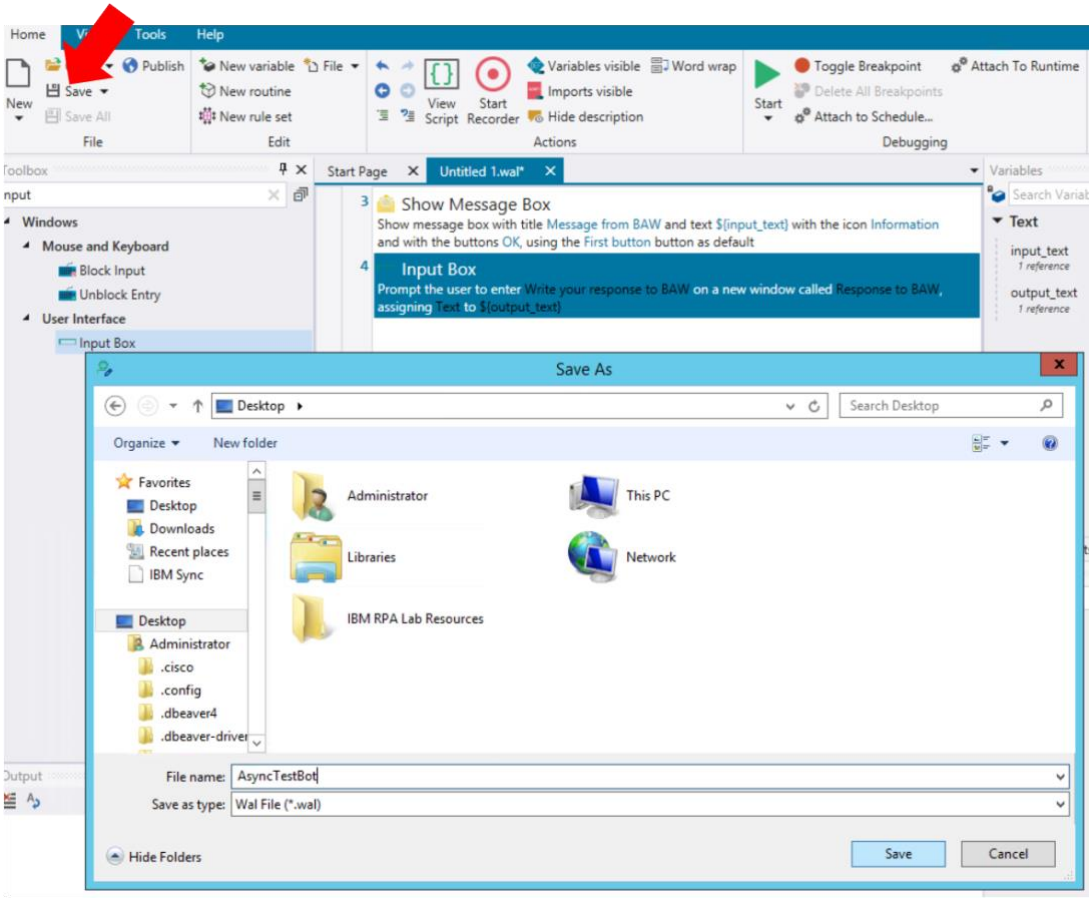


25. Add also the **Input Box** command to your script. Configure it as shown in the picture below.

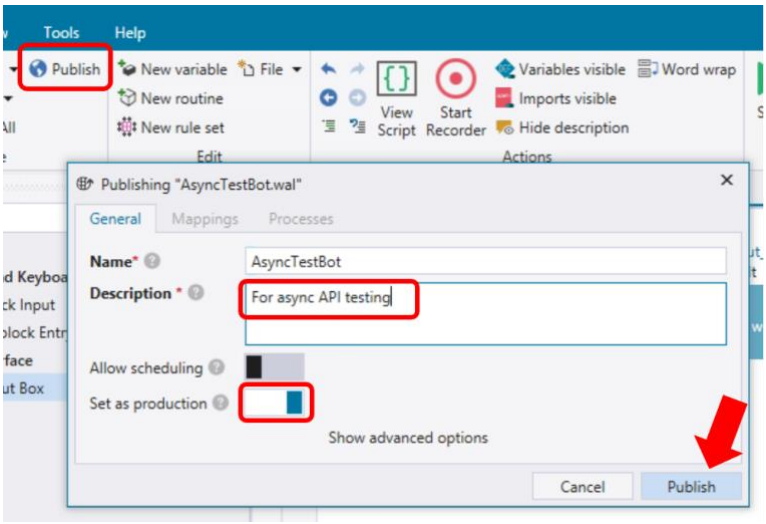


This command allows the user to type in a response back to BAW and therefore we store it as our script output variable (output_text).

26. Save the script by clicking the **Save** selection on the top toolbar. Save your script to your **Desktop** and name it **AsyncTestBot**.



27. Now need to publish the script to our RPA tenant that the Studio is connected to. Click **Publish** from the top toolbar, type in **a description** (For async API testing), enable **Set as production** switch and then click **Publish**.

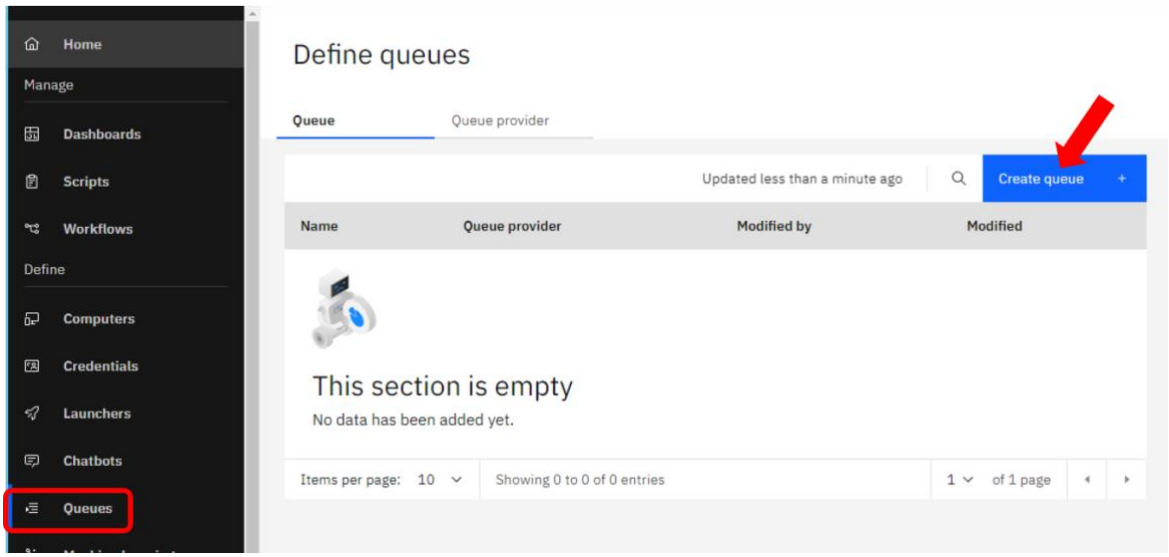


2.4 Set up RPA Process

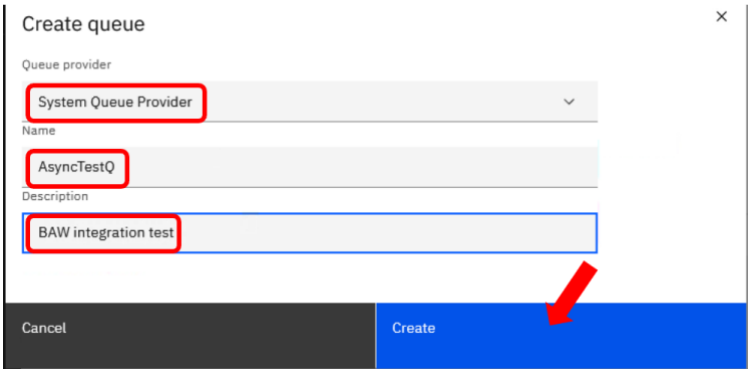
Since the IBM RPA asynchronous API is using RPA Process definitions, when we use the API, it creates an instance for defined RPA Process and then runs the automation script(s) that is defined to process step(s). So, in order to test the async API, we need to first define an RPA Process to be used. Let's get to it!

28. You should still have your RPA Control Center open in Firefox from the steps 7 – 11, but if you closed it, follow steps 7 and 8 to open it again.

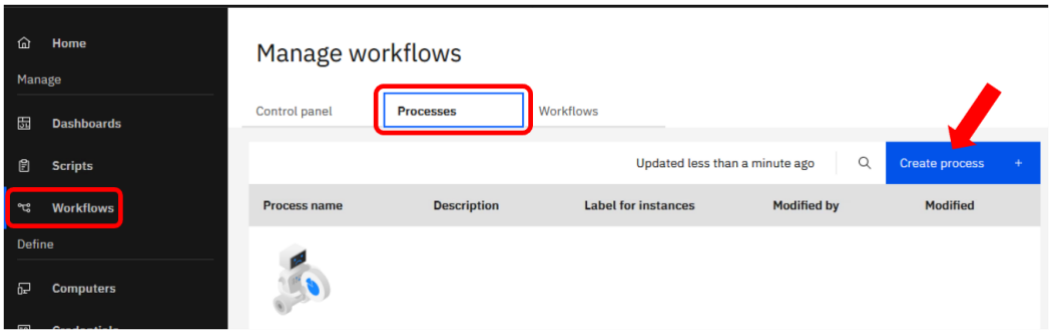
29. Select **Queues** from the right-hand side menu bar and then click **Create queue**.



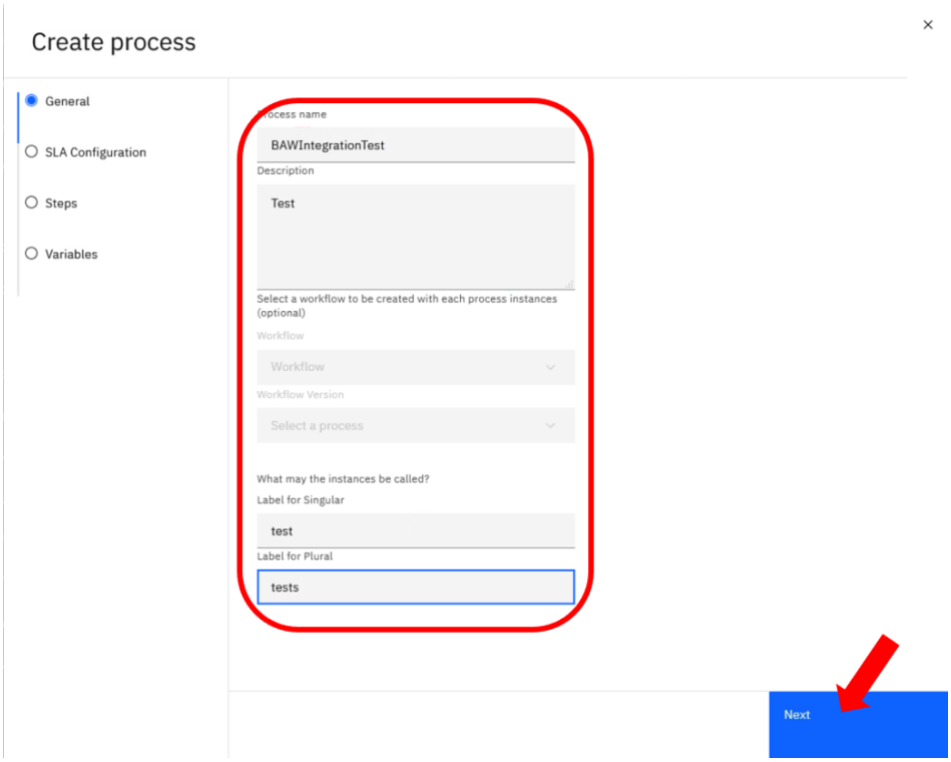
30. Select **System Queue Provider** as Queue provider, name the queue **AsyncTestQ**, give it a description and then click **Create**.



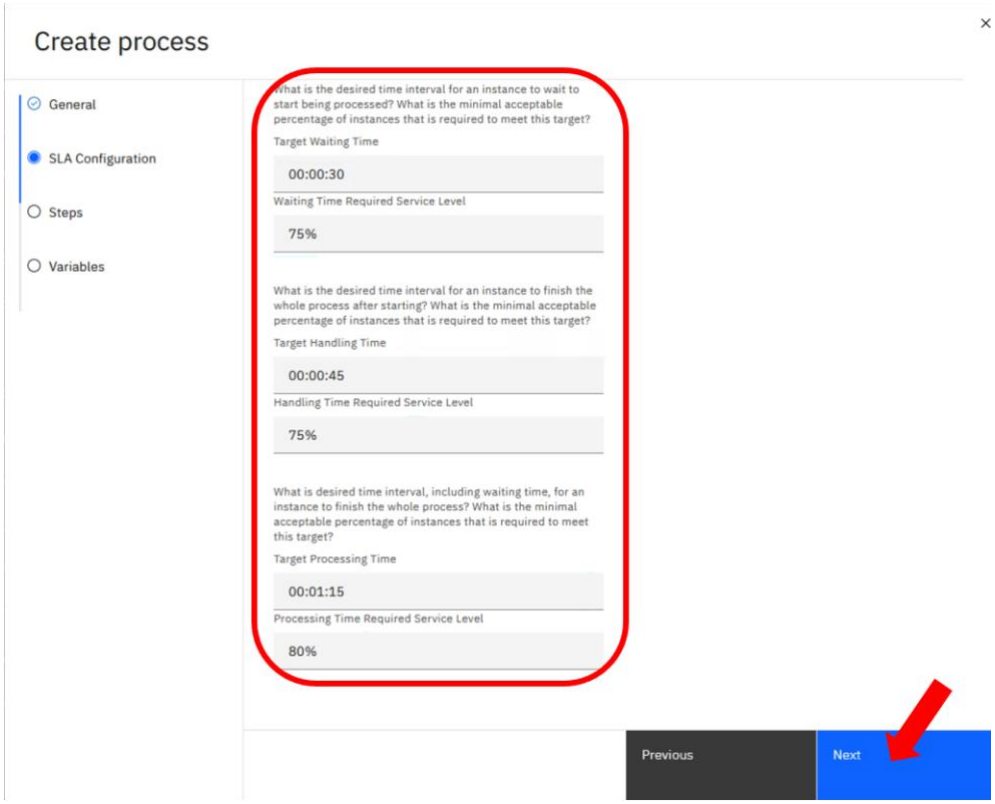
31. Next, select **Workflows** from the menu bar, move to **Processes tab** and then click **Create process**. NOTE! *If you do not see the Workflows option in the menu bar, just refresh your browser (this is because we just added the new roles for our user).*



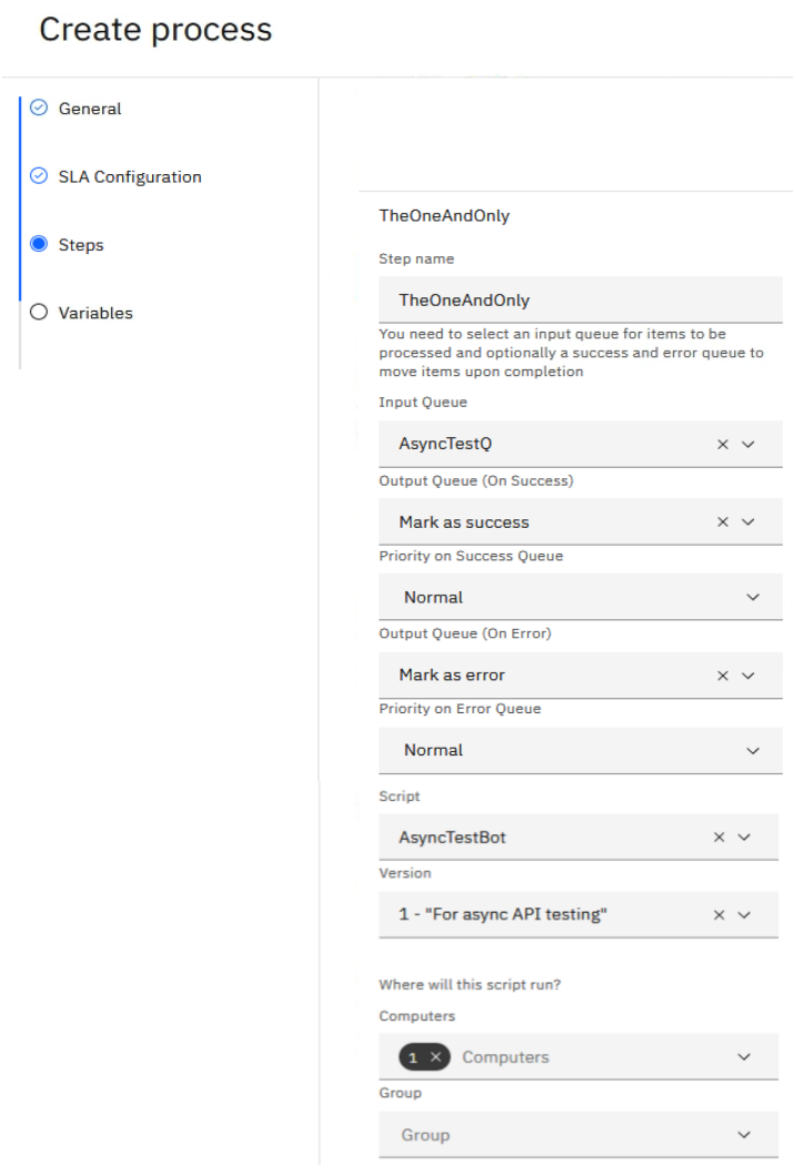
32. Name the process **BAWIntegrationTest**, give it a **description** and use **test** / **tests** for labels. Click **Next** to move forward.



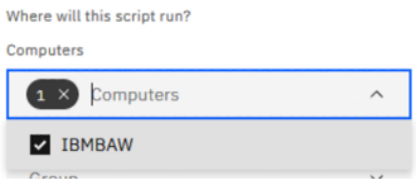
33. For SLA configuration use what is shown in the picture below or come up with our own values. Click **Next** to continue.
34.



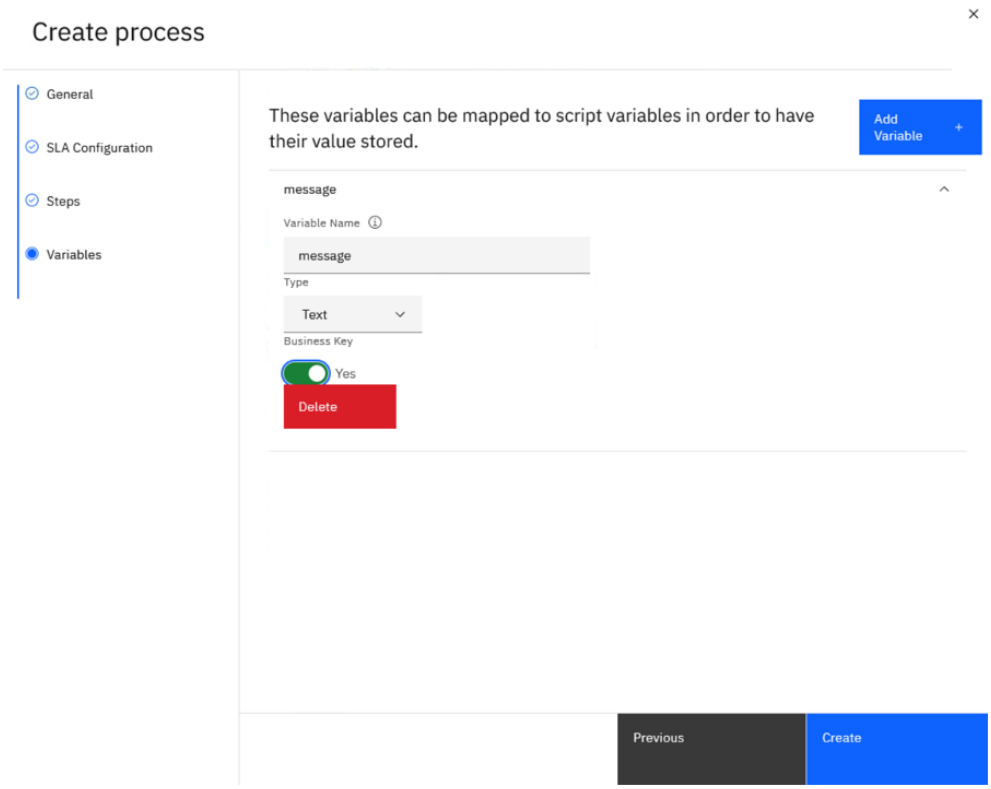
35. For steps, we'll need only one. Name it **TheOneAndOnly** and configure rest as shown in the picture below. Notice that we're using the script (AsyncTestBot) that we published to our tenant earlier. Click **Next** to continue.



Note! For computers select **IBMBAW** (the VM you're using).



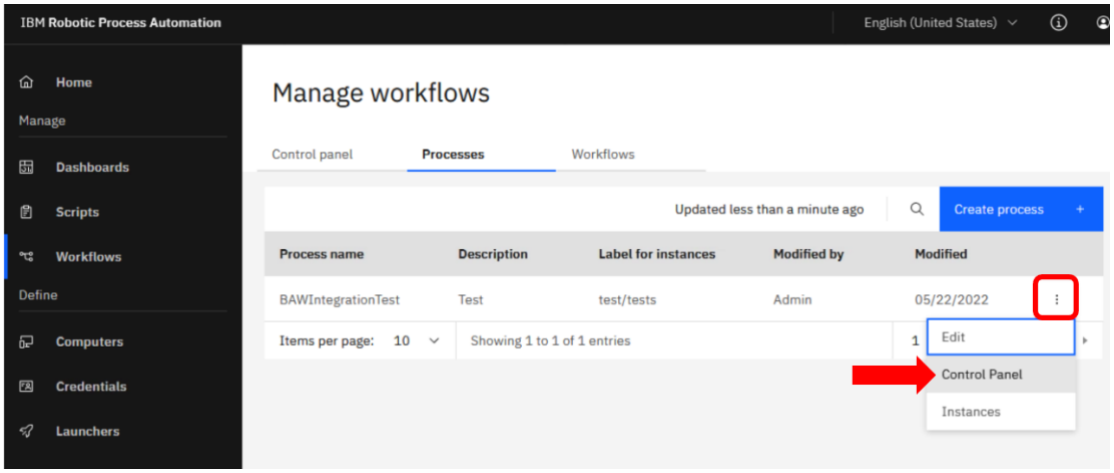
36. An RPA Process needs at least one variable that we can use to track the progression of a multistep process and relay information between the steps. We're not really using process variables in our demo, but we need to define one. See the picture below. When done, click **Create** to finish configuring the process and to create it.



2.5 Set up your computer definition

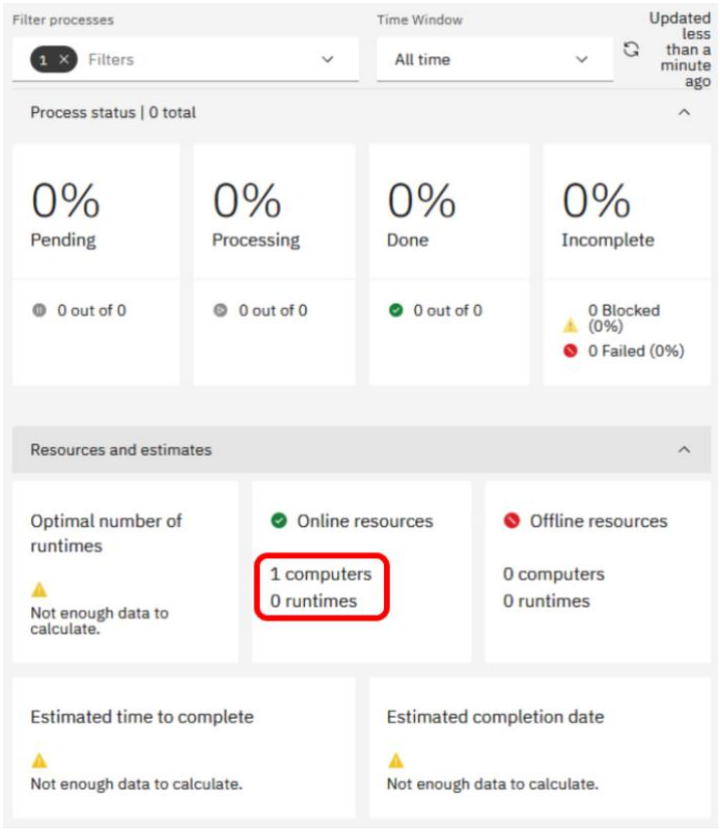
Now that we have configured an RPA Process to run our test script, we still need to make sure that your computer (IBMBAW that we defined as the computer to run the process) is ready to run processes, queue-based load i.e.

37. You should now see your newly defined Process under the Processes tab in Manage workflow view. Click **the ellipsis menu icon** at the end of the row and select **Control Panel**.

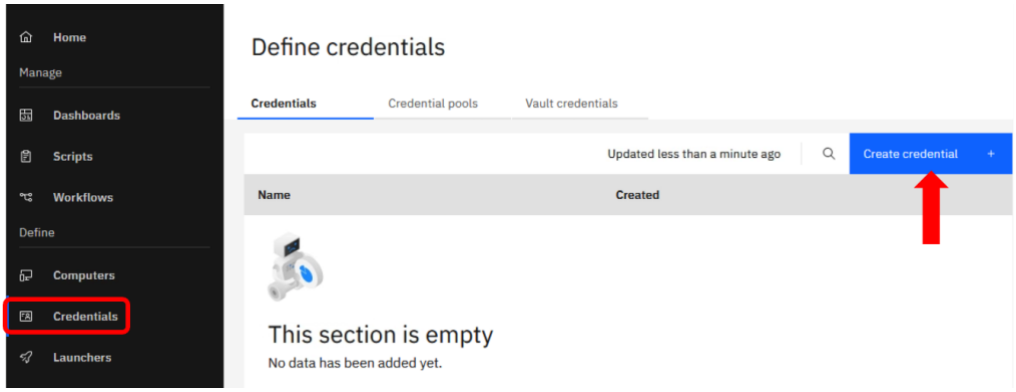


38. The process control panel is kind of a dashboard view to your process. It shows you the overall status of process execution and more. Here you can also check the computers assigned to the process and their available RPA runtimes.

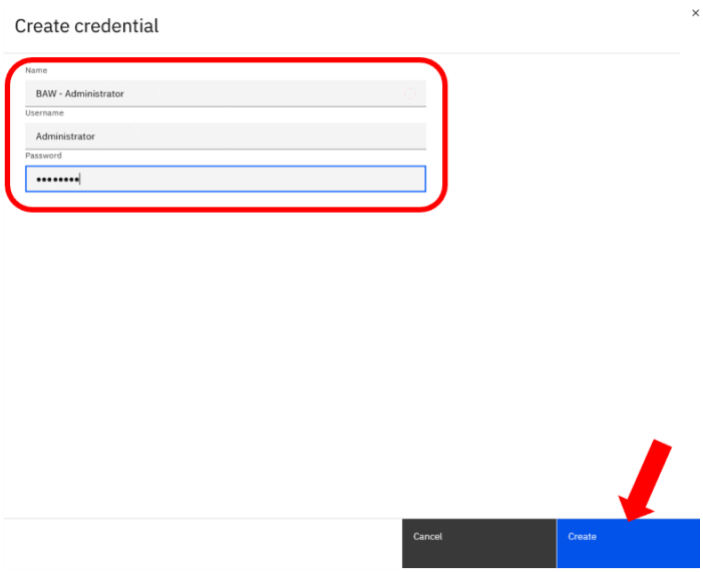
Notice the data under **Online resources**. It shows that 1 computer is online, but it has 0 runtimes available. Let's fix this.



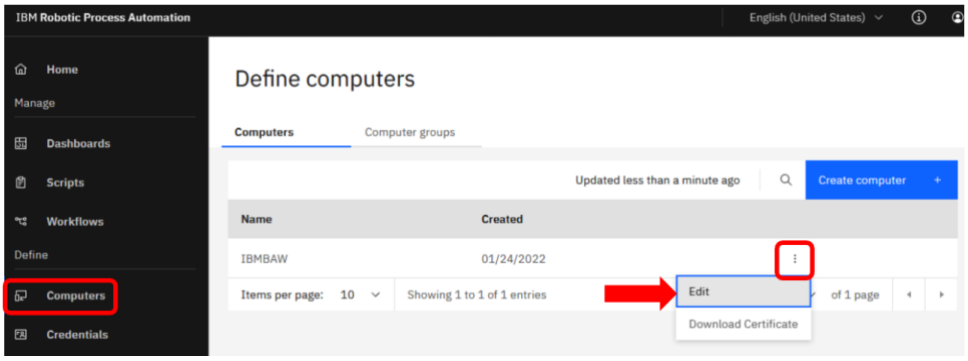
39. When running workload part of the defined Processes, the computer needs an assigned credential. This is to automatically login to computer if it's locked. Move to **Credentials** view from the left-hand side menu bar and click **Create credential**.



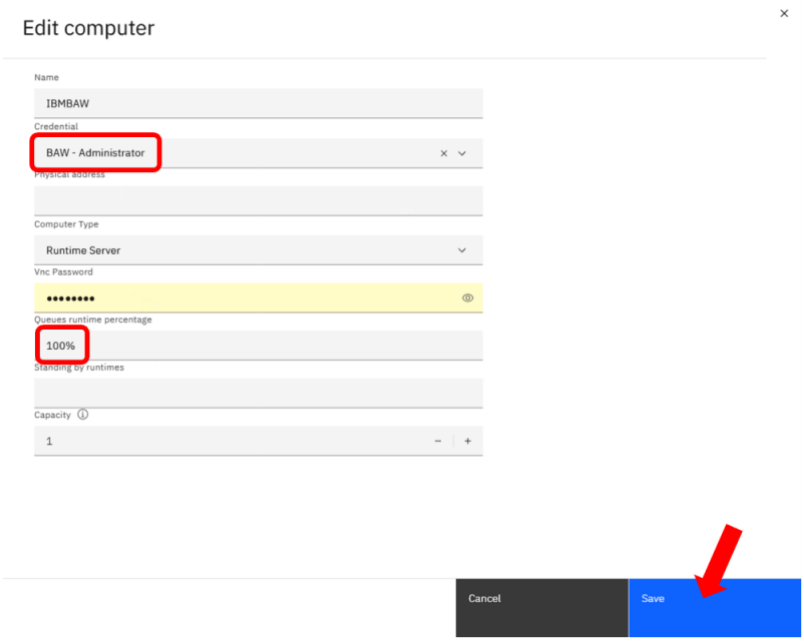
40. Name the credential **BAW – Administrator**. Type **Administrator** as the username and **passw0rd** as the password. That is the user account that you're actually logged in to your virtual machine. Click **Create**.



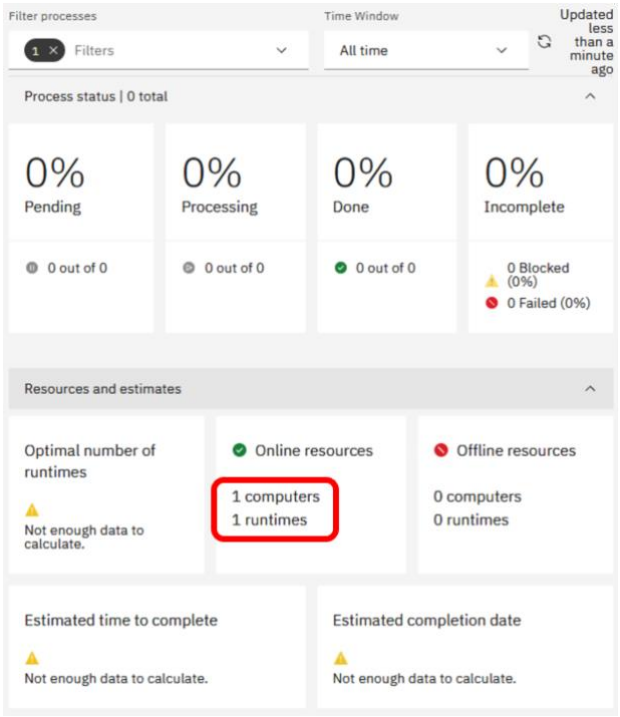
41. Move to **Computers** view from the left-hand side menu bar. Click on **the ellipsis menu icon** at the end of the IBMBAW row and select **Edit**.



42. Select **BAW – Administrator** as the credential and change the Queue runtime percentage to **100%**. Click **Save**.



43. Go back to BAWIntegrationTest process's **control panel**. You can see that the runtime number for Online resources has changed to 1.

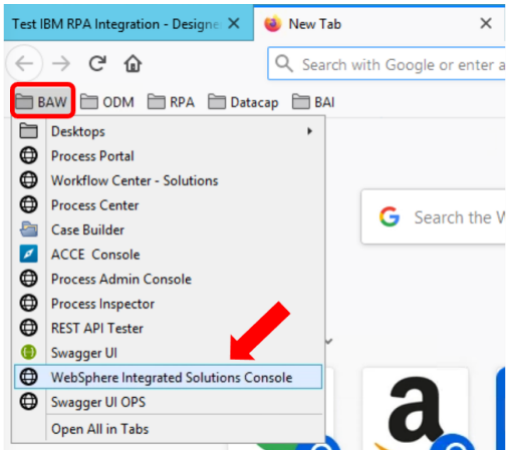


Nicely done! Now everything is set – when it comes to RPA environment – to test the asynchronous RPA API.

2.6 Import the signer certificate to BAW

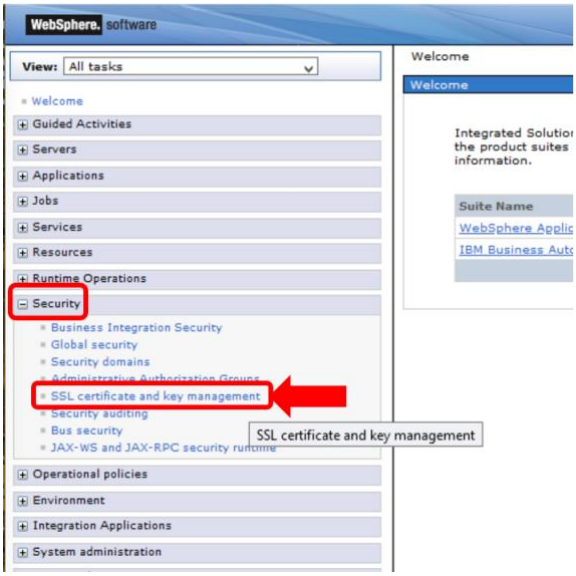
In order to BAW to call a trusted party – the IBM RPA Server in our case – we need to import the signer certificate from IBM RPA to BAW trust store. This is a security feature for BAW and underlying WebSphere application server. Note that if were running BAW on containers the procedure would be a bit different.

44. With Firefox web browser open the **WebSphere Integrated Solutions Console** from the bookmarks toolbar (BAW → WebSphere Integrated Solutions Console).

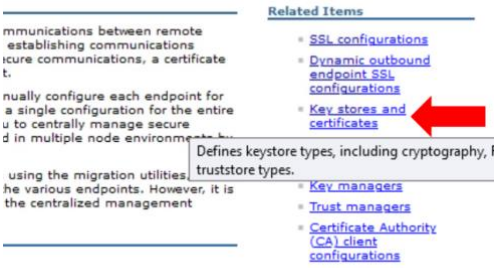


45. Log in with username ***admin*** and password ***admin***.

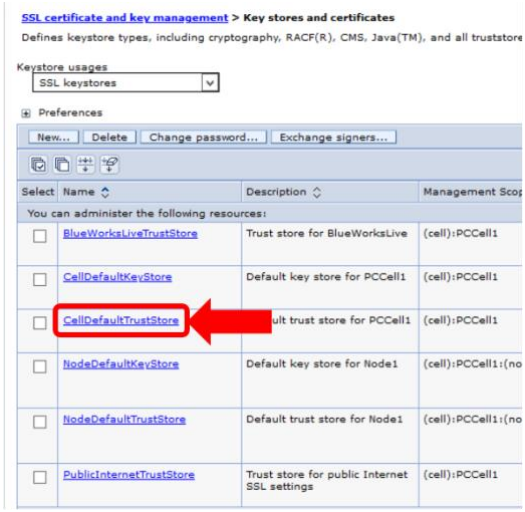
46. When the console opens, expand **Security** and select **SSL certificate and key management** from the left-hand side menu.



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



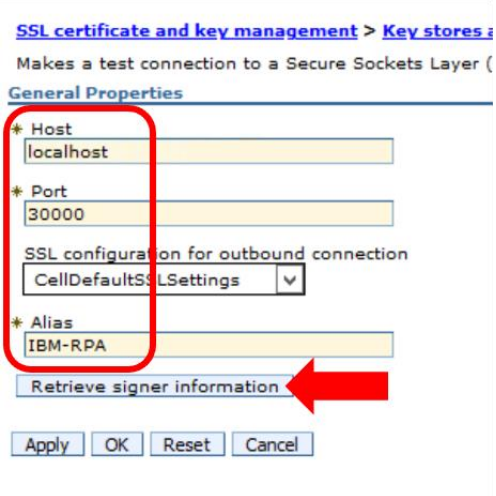
48. Next, click **CellDefaultTrustStore**.



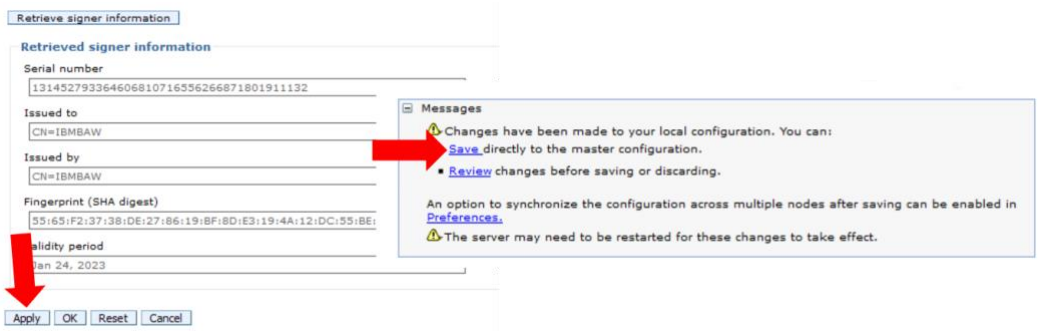
49. Click **Signer certificates** to open view for signer certificates for the CellDefaultTrustStore. As we see, we have already some previously imported certificates here.

50. Click the **Retrieve from port** -button.

51. This opens a wizard to get the signer certificate. Type in **localhost** for host, **30000** for Port and **IBM-RPA** 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. That’s it.

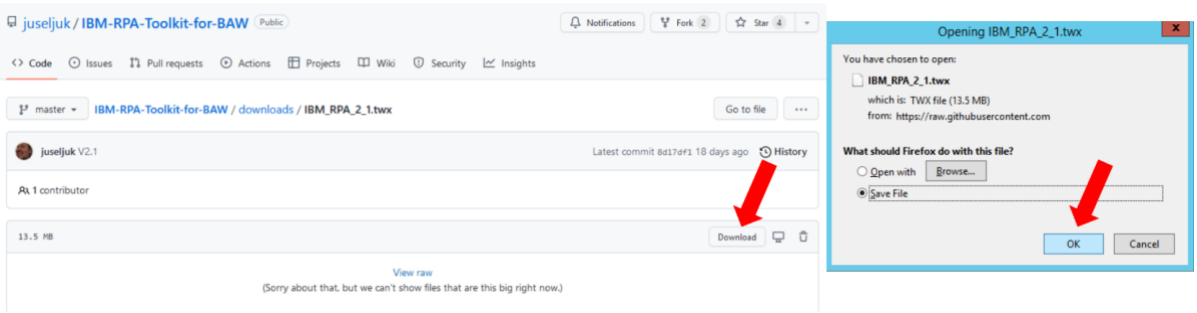
3 Create BAW Process Application for testing the RPA API

To test the RPA API, we will create a simple BAW Process Application. We will use the BAW Toolkit for IBM RPA to configure our process application.

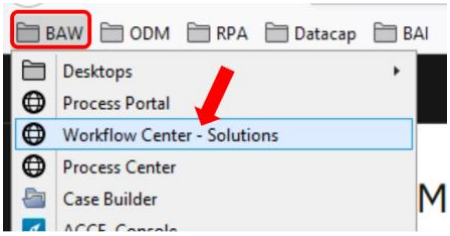
3.1 Download and import the toolkit

52. Navigate to https://github.com/juseljuk/IBM-RPA-Toolkit-for-BAW/blob/master/downloads/IBM_RPA_2_1.twx with your Firefox web browser.

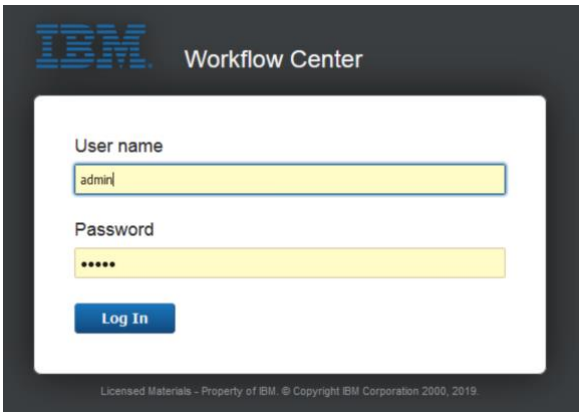
53. Click the **Download** button on the opened GitHub page, select **Save File** and click **OK**. The toolkit file will be saved to your profile Downloads folder.



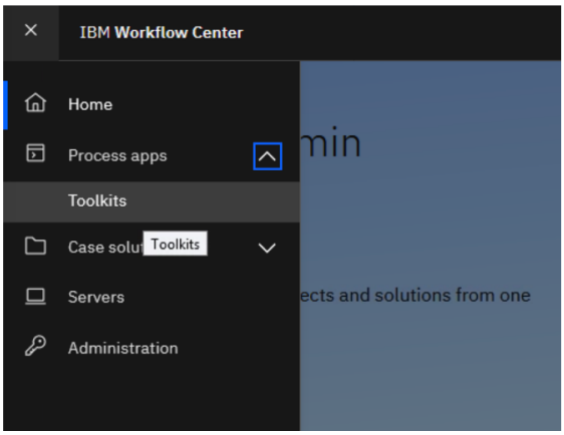
54. Use Firefox bookmarks toolbar to select **BAW → Workflow Center – Solutions** to open the BAW Workflow Center.



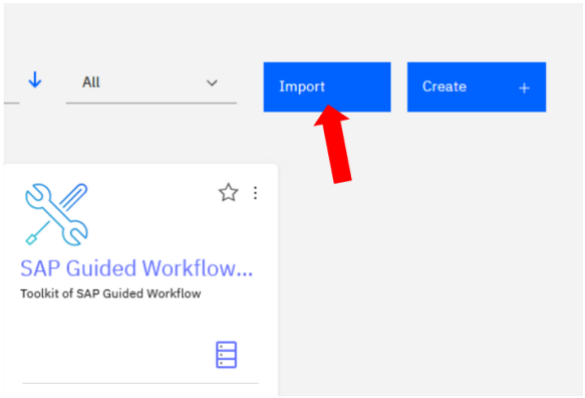
55. Log in as **admin** with password **admin**.



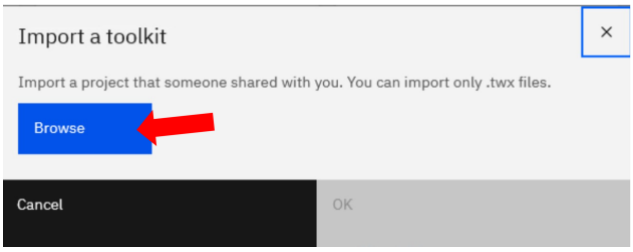
56. Once the Workflow Center page opens, select **Toolkits** under Process apps from the top left-hand side “burger menu”.

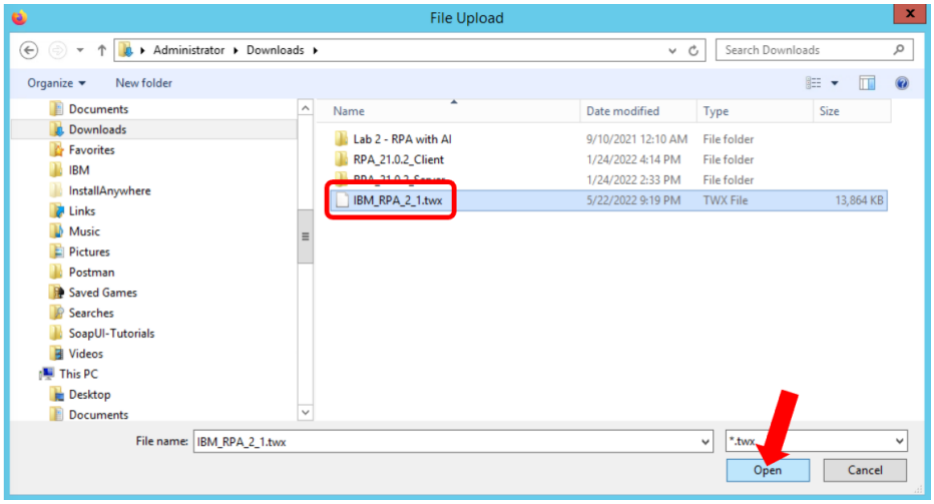


57. Click **Import** to start importing the BAW Toolkit for IBM RPA that you just downloaded.

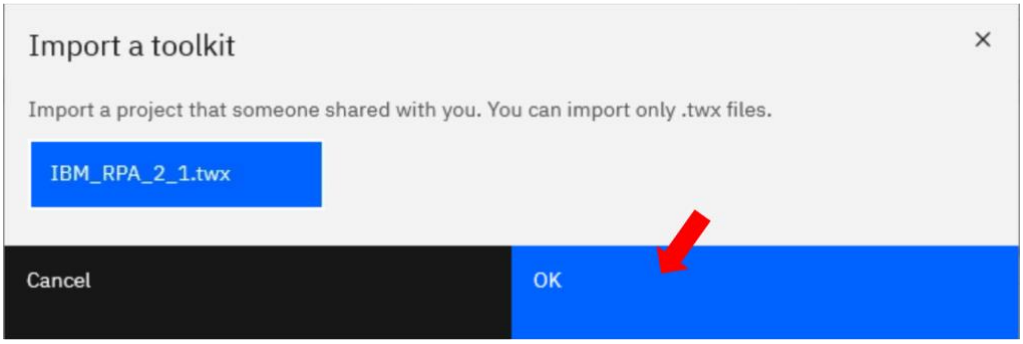


58. Click **Browse**, select ***IBM_RPA_2_1.twx*** from the Downloads folder and click **Open**.



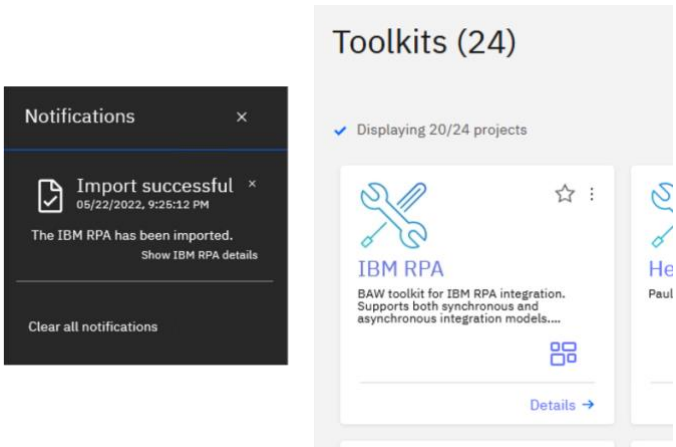


59. Click OK to import the toolkit.



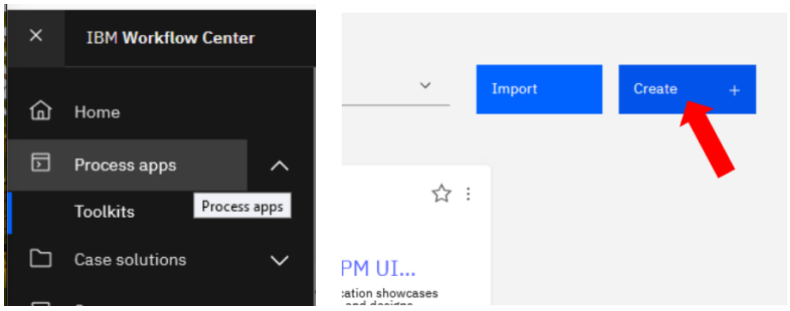
60. Importing the toolkit takes a couple of seconds and when done, you'll see a notification of successful Import. You should also see IBM RPA toolkit appearing to your list of available toolkits.

    Good! Now that the toolkit is imported and available, you can now implement a process application to test using it.

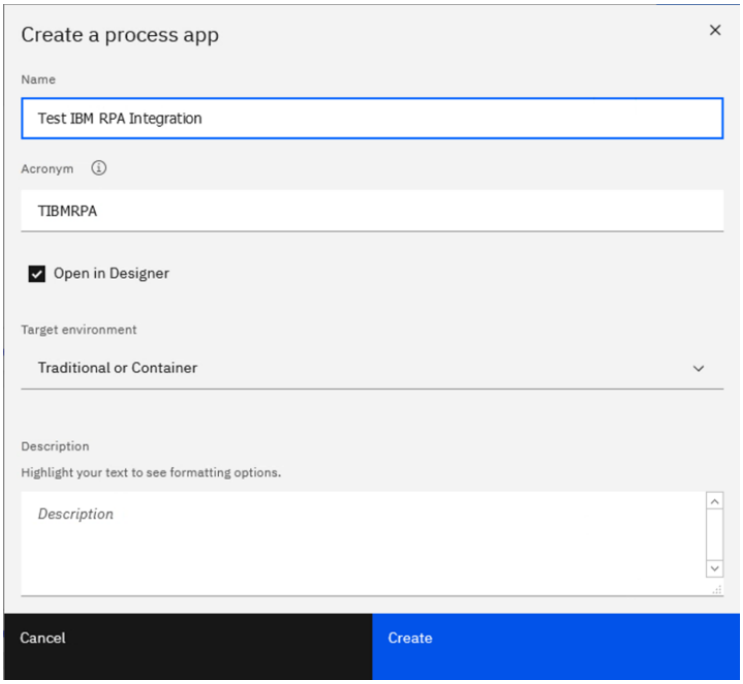


3.2 Implement Process Application to test IBM RPA integration

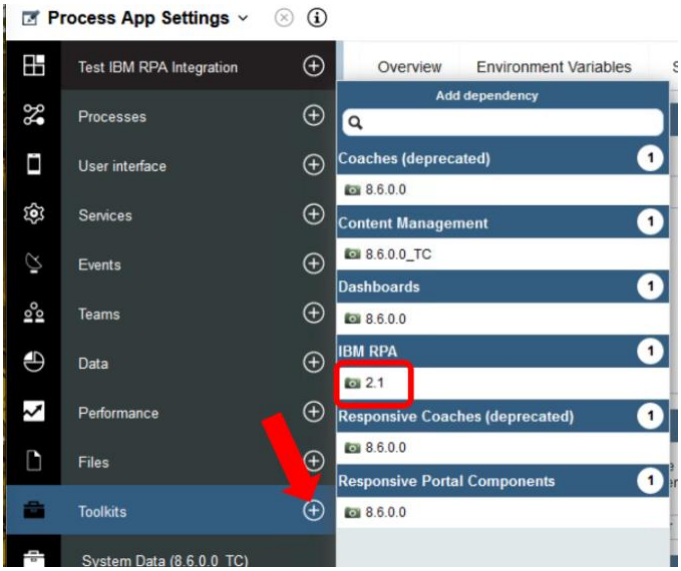
61. Select **Process apps** from the left-hand side menu and the click **Create**.



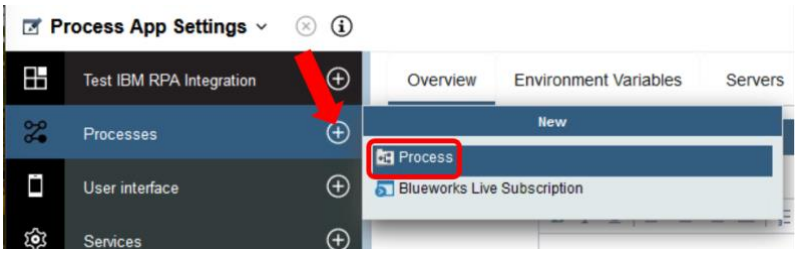
62. Name your process app **Test IBM RPA Integration**, make sure that “Open in Designer” is checked and click **Create**. The process application opens in web designer.



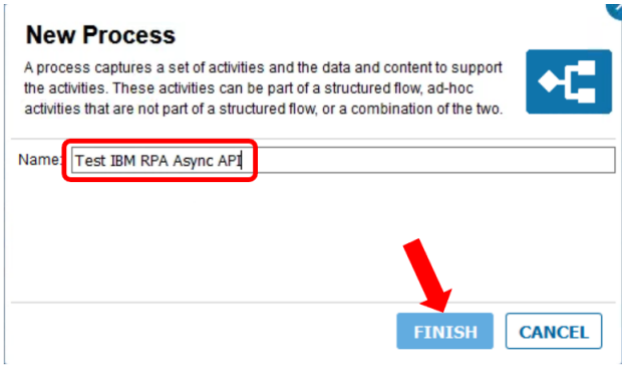
63. Click the **plus sign (+) besides Toolkits** from the left-hand side menu and select version **2.1** under IBM RPA (the toolkit version we just imported).



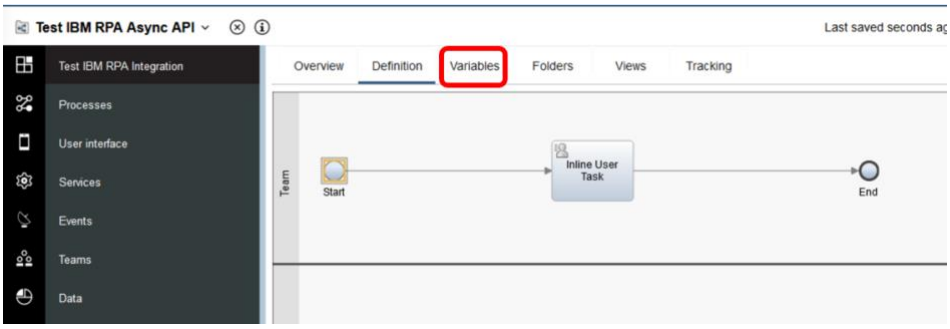
64. Next, create a new process by clicking **the plus sign (+) besides the Processes** in the left-hand side menu and select **Process**.



65. Name your process **Test IBM RPA Async API** and click **FINISH**.

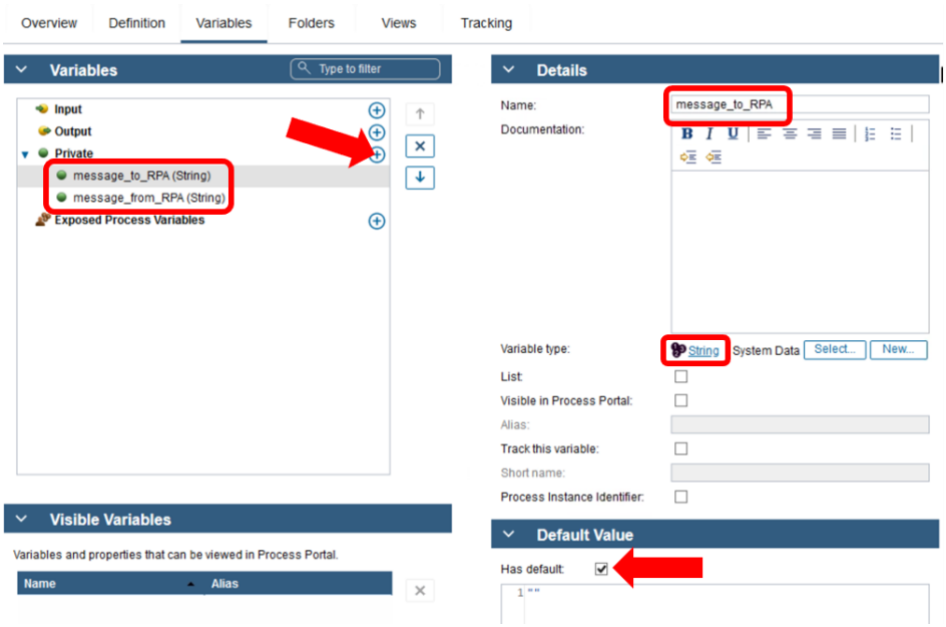


66. The process definition opens in the Definition view. First, we need to add some variables. Select the **variables tab**.

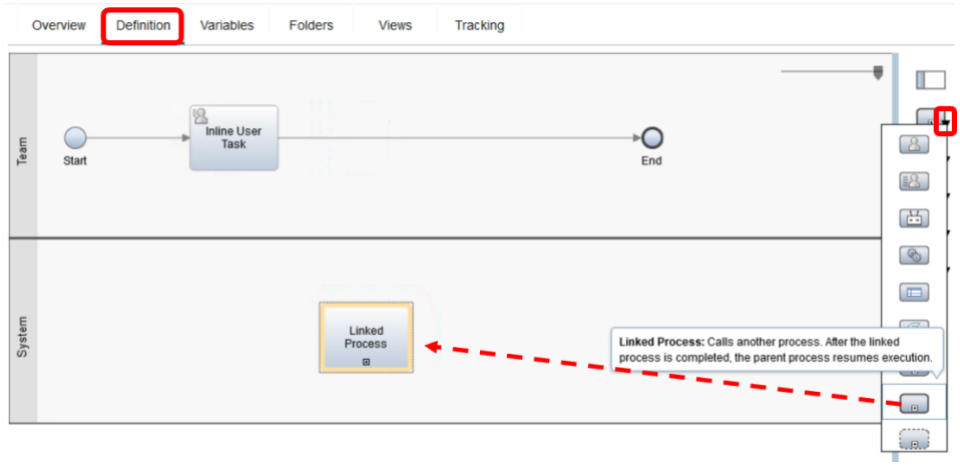


67. We need to create 2 private variables: **message_to_RPA** and **message_from_RPA**.

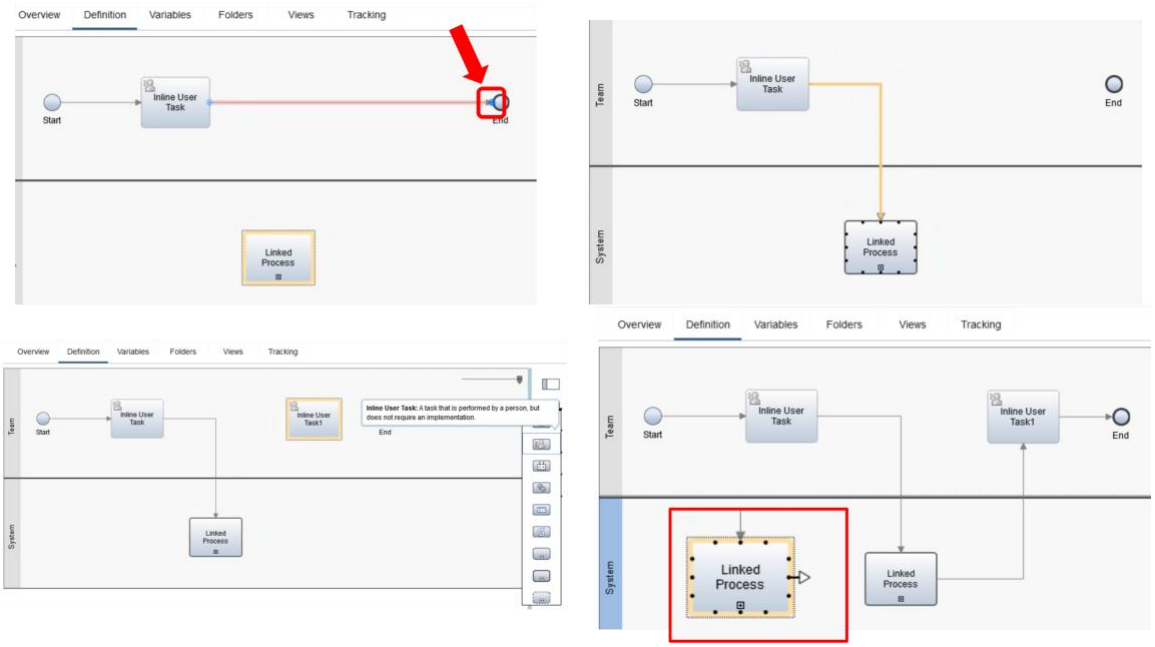
    To create the variables, first click **the plus sign (+) besides the Private** in the Variables section, then name the variable, make sure the type is **String** and that the "Has default" **is checked**. Repeat this to create both variables. Notice that web designer has auto-save, so you do need to save your changes.



68. Move back to **Definition** view, expand the selections for the Activity icon by clicking **the small arrow besides the icon** and finally drag and drop **Linked Process** activity to the **System** lane.



69. **Connect** the **Inline User Task** to **Linked Process** activity. You can do this by selecting the line between the Inline User Task and the End symbol. Then **hover over the connection point at the End symbol** so that you see the connection point changing to dark blue. You can now drag and drop the connection point to Linked Process activity as show in the picture below.

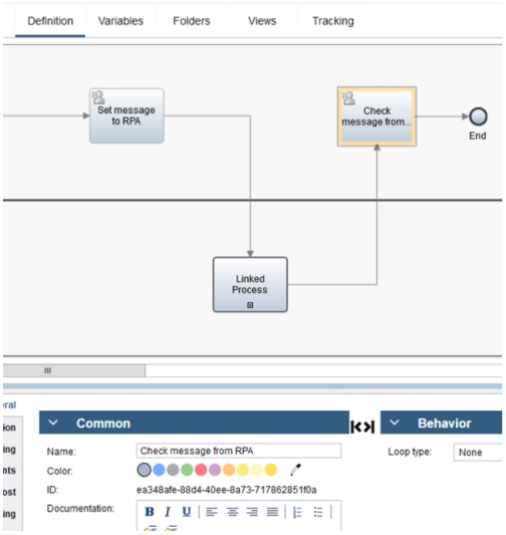


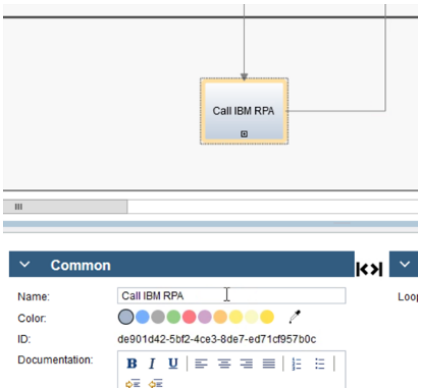
Next, **add** another **Inline User Task** form the drop-down selection menu for Activity to the **Team** lane and connect the Linked Process to it as shown in the picture above. You can do the connection by hovering over the border of the Linked Process activity until you see the arrow popping out from the side of the activity. You can now drag and drop the connection to new Inline User Task activity.

Finally, **connect** the new Inline User Task activity to the End symbol.

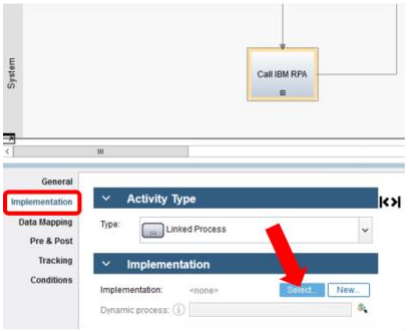
70. Select the first Inline User Task activity and change the name to **Set message to RPA**.

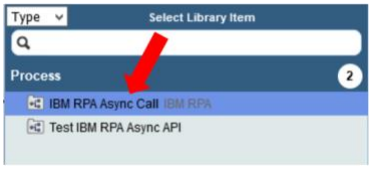
71. Repeat the previous step to change the name of the second Inline User Task activity to **Check message from RPA** and then the name of the Linked Process activity to **Call IBM RPA**.





72. Still the newly named **Call IBM RPA** activity selected, move to **Implementation** section of the properties section, under Implementation header click **Select button** for the implementation and select **IBM RPA Async Call**. This is the ready-made process to call IBM RPA from the toolkit that we earlier imported.

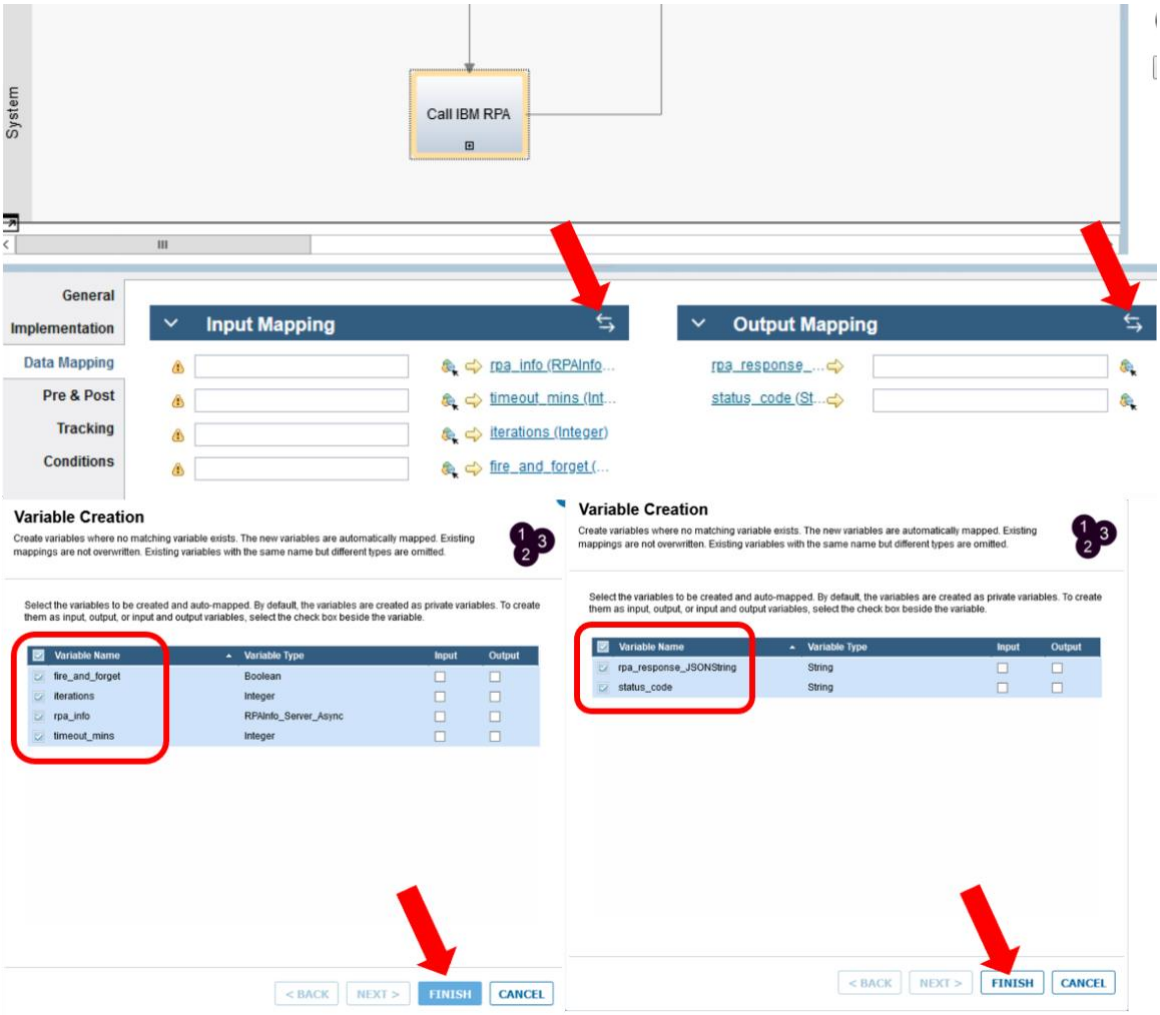




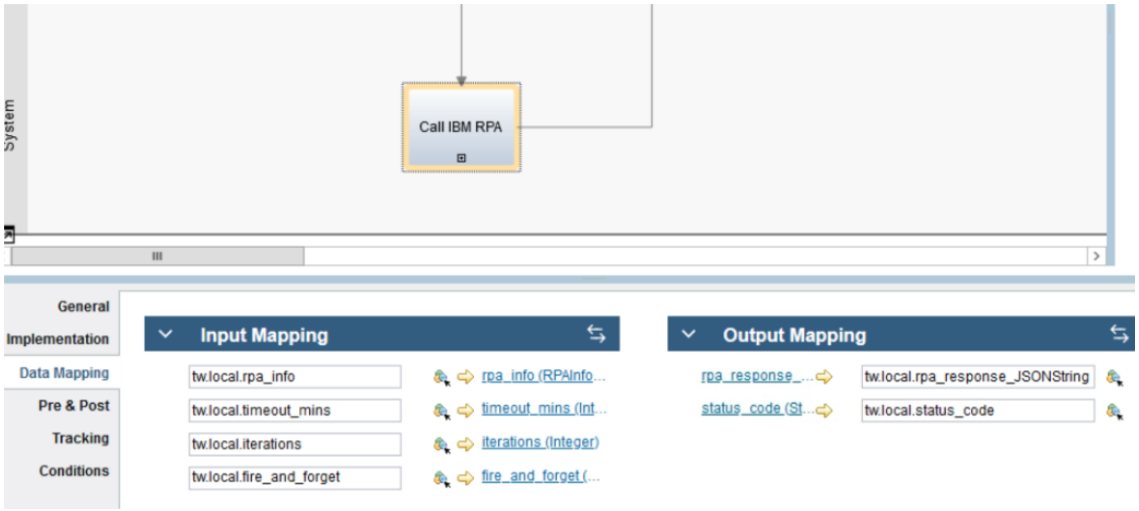
73. Next, select the **Data Mappings**. We'll create the needed input and output variables for the process that we want to call. Can generate them automatically based on the signature of the process – or service – we want to call.

    First, click on **the two arrows at the end of the Input Mapping header** to open the Variable Creation wizard. Make sure that each variable is selected (look the picture below) and click **FINISH** to close the wizard.

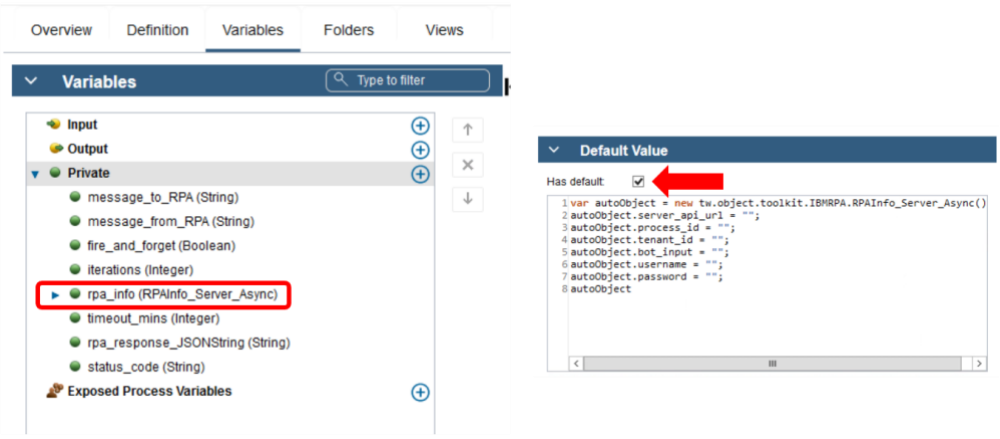
    Repeat the previous also for the Output Mapping.



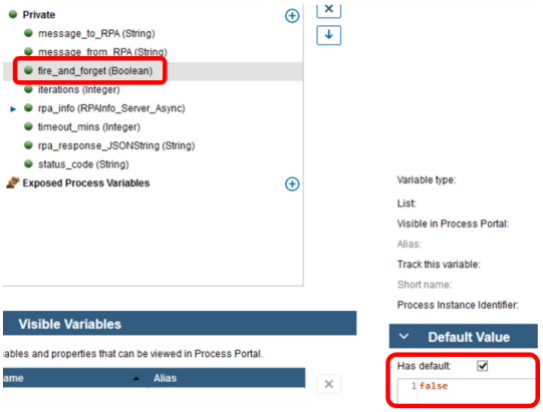
When done, you should see variables automatically also assigned under Input and Output Mapping:



74. Go to **Variables** tab/view and make sure that **rpa_info** variable has "**Has default**" option checked.

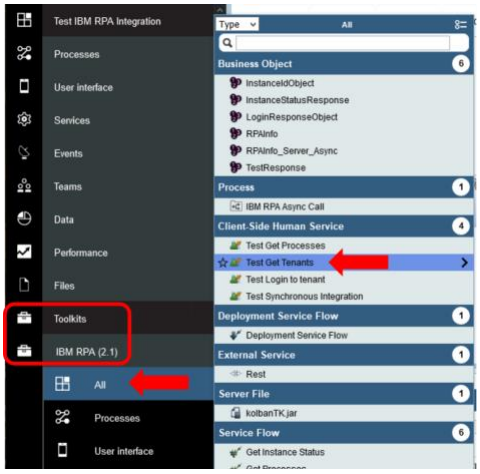


75. Similarly, check the **fire_and_forget** variable and see that the value for it is **false**. *Note! This variable can be set to true to omit waiting for any response back from the RPA triggered, hence the name "fire and forget". In this exercise we want to wait for the answer / result back from the RPA bot, so we set it to false.*

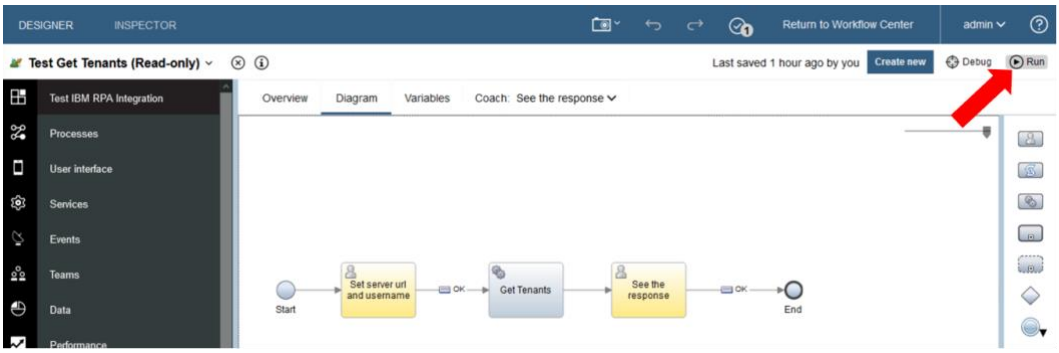


76. Now that we have our test process and the variables in place, we can use some of the helper services inside the RPA Toolkit to find out the needed IDs (tenant and process) in order to test the integration.

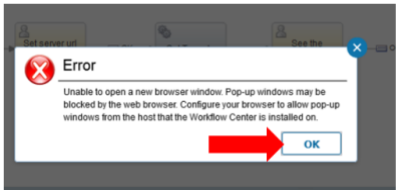
    Expand the **Toolkits → IBM RPA (2.1)** from the left-hand side menu (just click on the names), click **All** to show all the artefacts of the toolkit and then click **Test Get Tenants** to open that Client-Side Human Service.

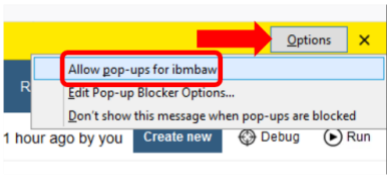


77. When the service opens, click the Run selection to run the service. We can use the service to get our tenant ID that we need to complete the configuration for triggering the bot. **NOTE!** *Tenant IDs are immutable, so you just need to get the ID once since it will not change after it is set.*



78. You will probably see an Error since the pop-ups for this site are blocked by default. Close the message box by clicking **OK** and then go to **Options** now displayed on the top of your browser window and select **Allow pop-ups for ibmbaw**.





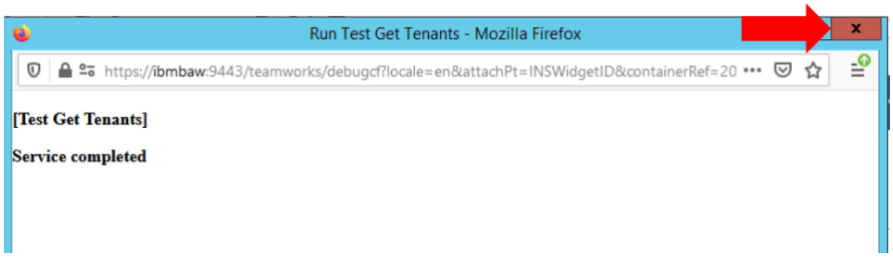
79. **Close** the opened window and **run** the service again. This time you should see our test UI window opening. Set ***localhost:30000*** to Server_url, ***admin@ibmdba.com*** to Username and then click the **OK**-button.

80. The service should run and after a couple of second you should see the results presented. The service is calling the part of the API that you do not need to authenticate to. With this call, we're getting all the tenants for our user admin@ibmbda.com.

    As you can see, we have just one tenant in our locally hosted IBM RPA Server called RPA Demos. Now, you need to copy and store the tenant id (*bc94fbd3-3d42-49ec-8575-63dd51d2e489*) to yourself, because we need that when we move forward.

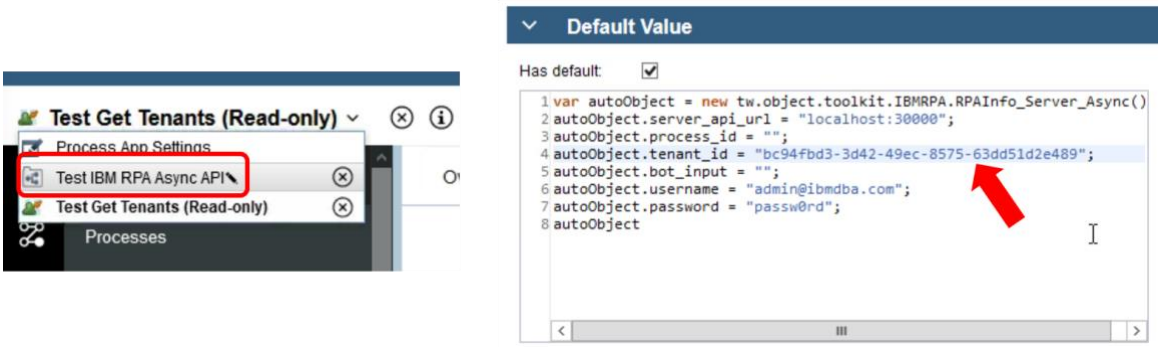
    NOTE! *You might need to make the output section a bit larger to see the hole output message. Use the control on the bottom right-hand side of the output section.*

81. Click **OK** to close the UI and you can also **close** the window that says, “Service completed”.



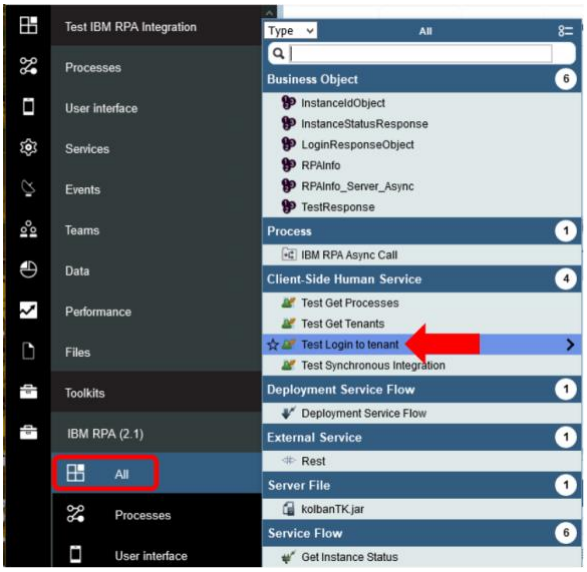
82. Move back to your process **Test IBM RPA Async API** by selecting it from the dropdown menu where it currently says, “Test Get Tenants (Read-only)”. This will bring you back to your process definition.

    Move to **variables** tab and set the ***tenant_id*** attribute of your **rpa_info** variable using the value we just got using the Test Get Tenants -service.



83. We still need to get the process id in order to invoke our bot through the IBM RPA asynchronous server API. For this, the toolkit offers two additional helper service.

    Expand **All** from the IBM RPA toolkit again and this time open **Test Login to tenant**.



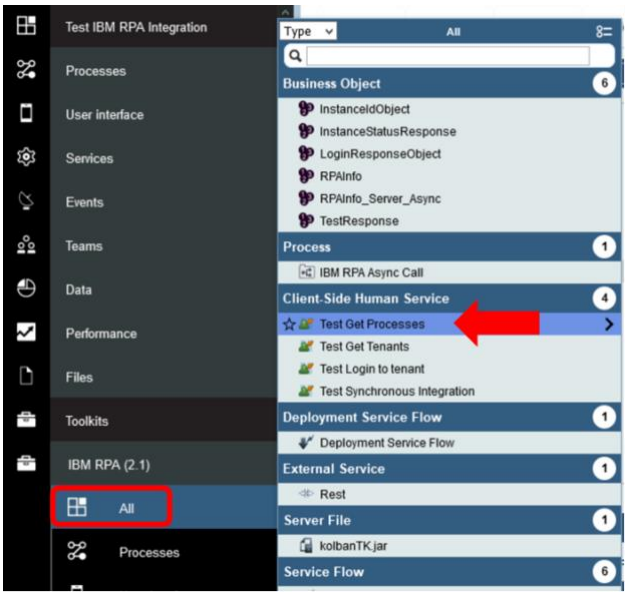
84. Run the service and fill in the UI form with values:
    - Server_url: **localhost:30000**
    - Username: **admin@ibmdba.com**
    - Password: **passw0rd**
    - Tenant_id: *<use the value you got and stored from the step 70>*

    Click the **OK** button.

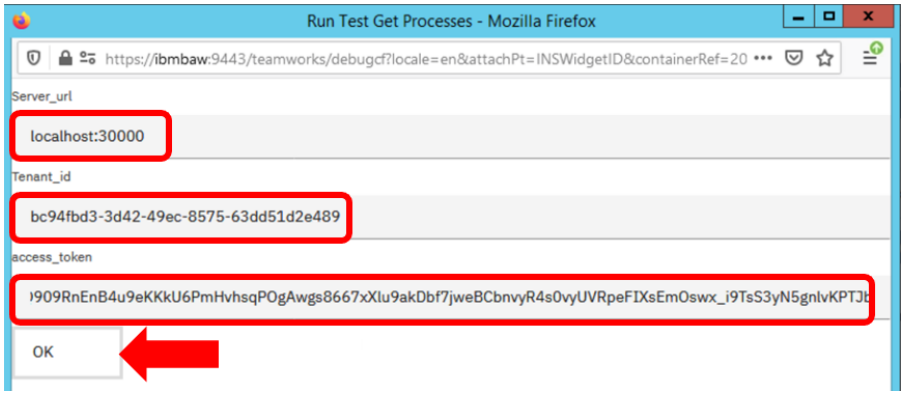
85. The service runs and will present the results of calling the RPA API to get an access token in new UI screen. The *Login_response* shows the hole JSON string response back from the RPA server and for your convenience, it also shows the access token in *Access Token* field. **Copy and store the access token** we need it during the next steps.

86. Click **OK** to close the service and you can also close the window that says, “Service completed”.

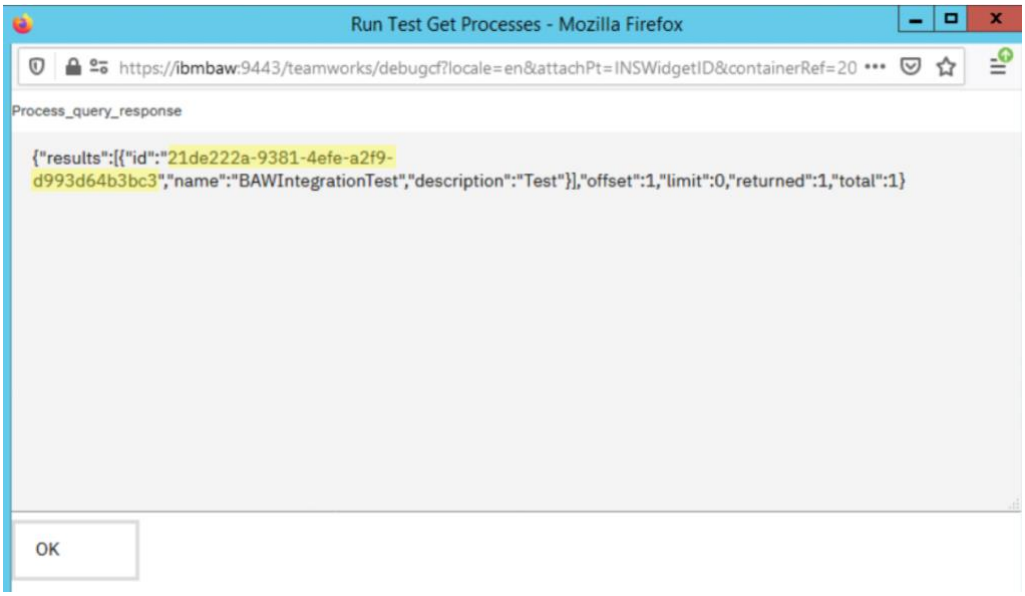
87. Next, we want to use the **Test Get Processes** helper service from the toolkit.



88. **Open** and **run** the service. For *Server_url* use **localhost:30000**, for *Tenant_id* the **tenant id value that you obtained earlier** and for *access_token* the **value you just obtained in step 75**. Click **OK**.

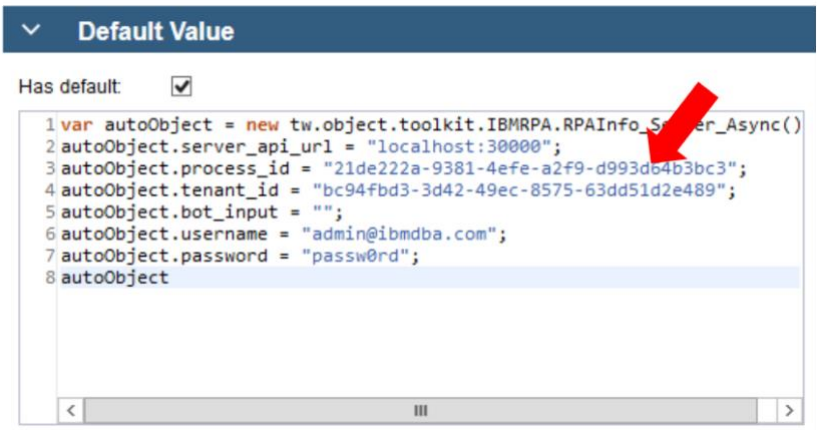


89. This helper service will get you all the one step processes that have been defined to the RPA tenant. As it should, we only have one of those defined, BAWIntegrationTest. **Copy and store the id** for the process (highlighted in yellow in the picture below). NOTE! *The id for your process will be different than in the picture below, because they are unique.*



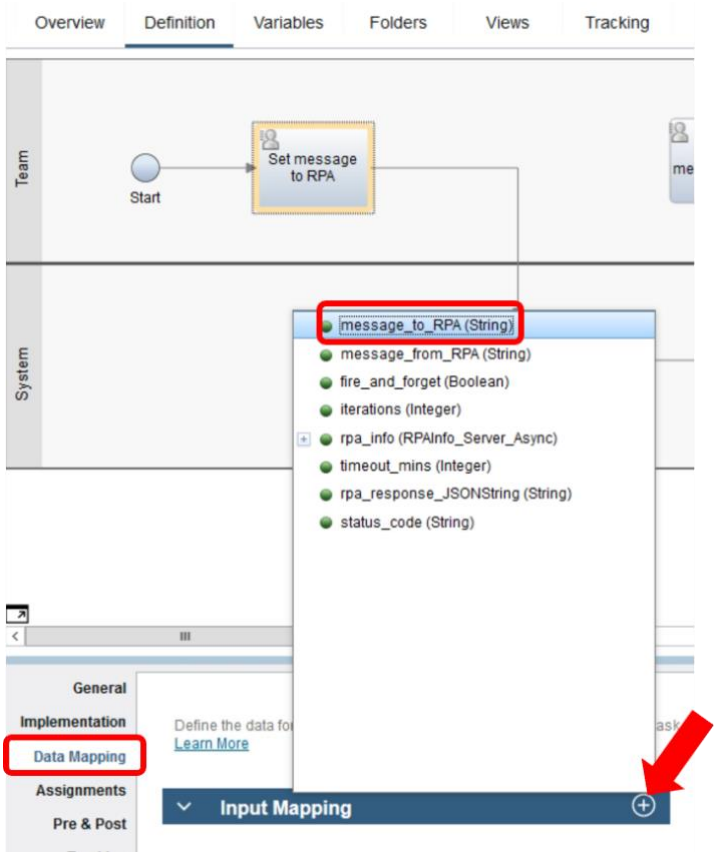
90. Click **OK** to close the service and close the window stating that the service has completed.

91. Go back to view your process definition and set the **process_id** attribute for your rpa_info variable with the value you just obtained in step 79.

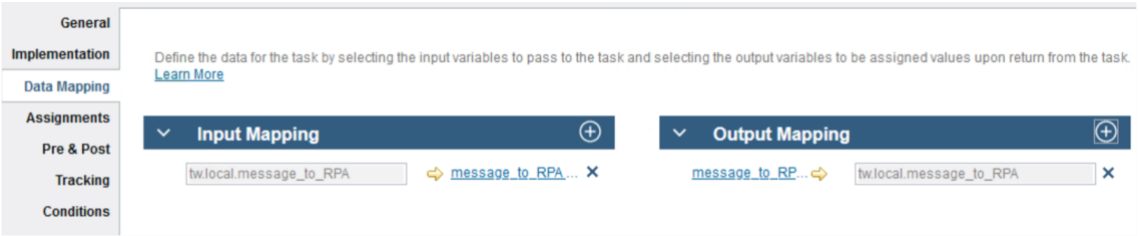


92. **Fill in also the other variable attributes** as shown in the picture above. Attribute *bot_input* is to be left empty.

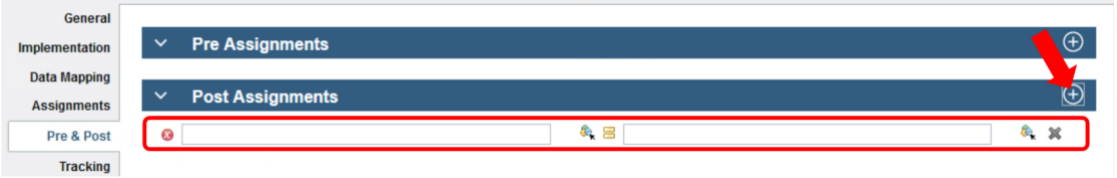
93. Next, Select the **Set message to RPA** task in your process definition. In configuration section move to **Data Mapping**, click the **plus sign (+)** for the Input Mapping section and select **message_to_RPA**.



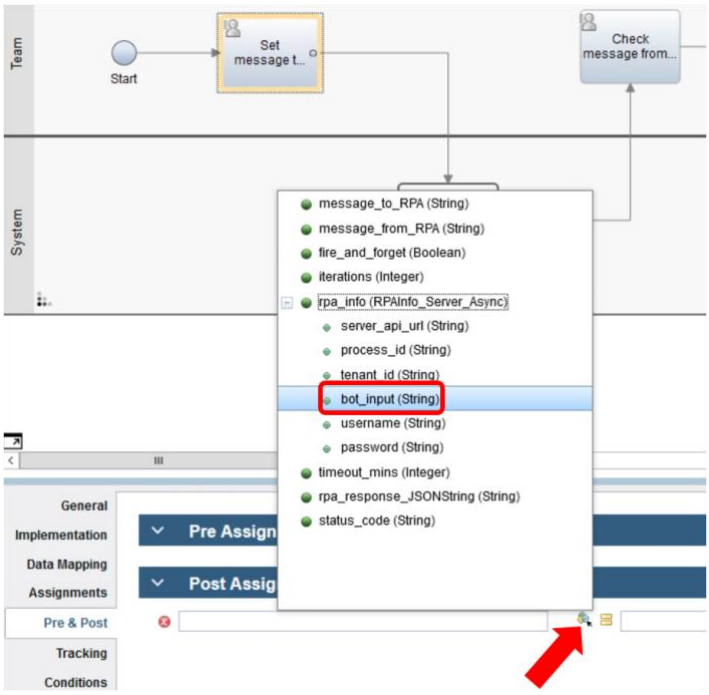
94. **Repeat** the previous also for the **Output Mapping** of the Set message to RPA activity so that you have the variable *message_to_RPA* mapped for both input and output.



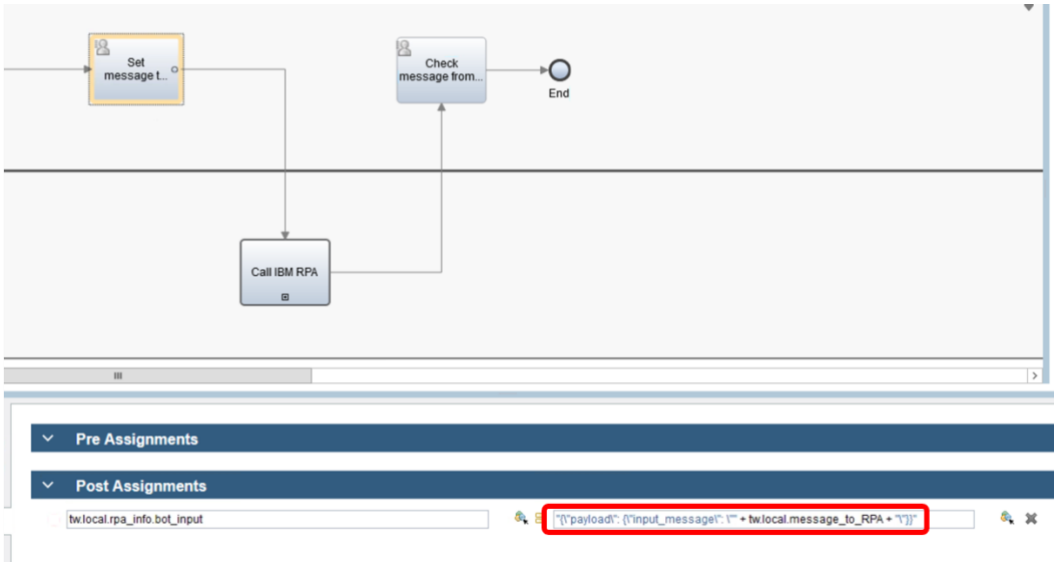
95. Next, when **Set message to RPA** activity still selected, move to move to **Pre & Post** assignments and click the plus sign (+) at the end of the **Post Assignments** section to create an empty mapping as show in the picture below.



96. Click the **variable selection icon** (look the picture below) and select the **bot_input** attribute of the rpa_info variable.

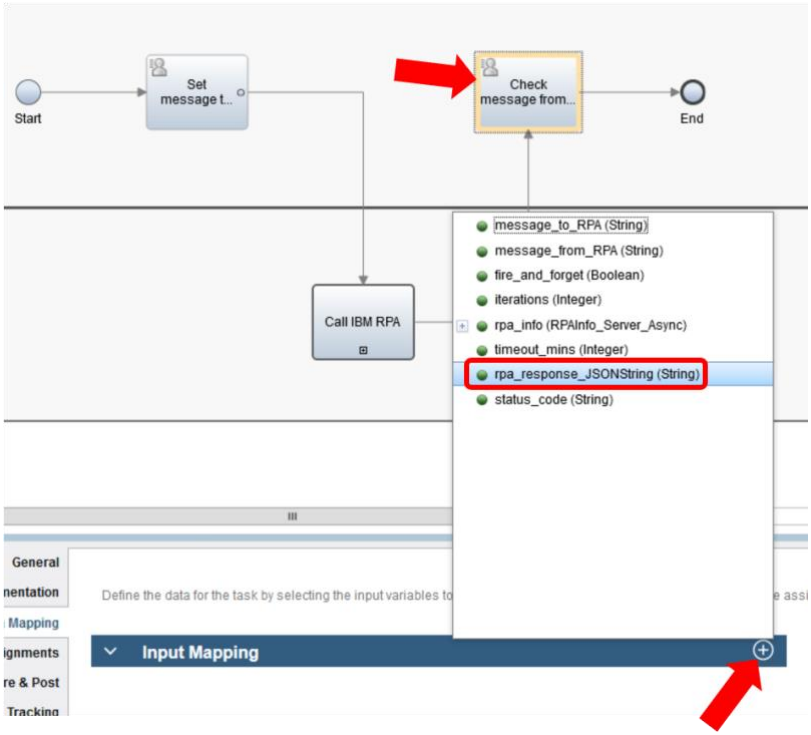


97. Set the assignment value to:
    "{\"payload\": {\"input_text\": \"" + tw.local.message_to_RPA + "\"}}"



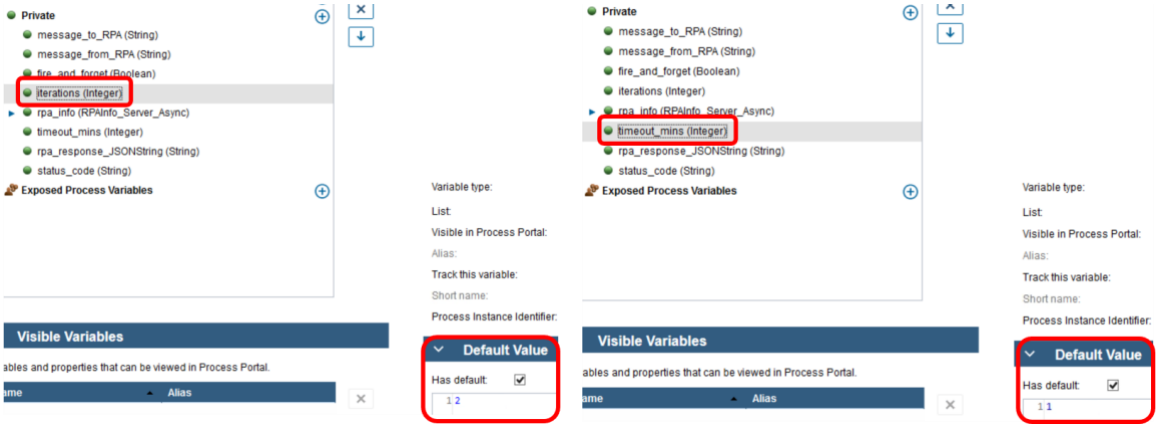
    This will assign the value we set in the Set message to RPA activity to the bot_input attribute of the rpa_info variable. When calling the IBM RPA API, the input variables for the bot needs to be placed inside a payload object presented as JSON string.

98. Since we also want to observe the response back from RPA, select the **Check message from RPA** activity and under **Data Mapping** add new **Input Mapping** selecting **rpa_response_JSONString** variable (look picture above).



99. One last thing before testing, we need to assign some values to our variables that control the execution of the RPA integration. Move to **variables** view and make sure that also **iterations** and **timeout_mins** have "Has default" selected and following values assigned as default values:

    - iterations: **2**
    - timeout_mins: **1**



*The value for **iterations** determines how many times BAW will try to get the response back from the RPA and **timeout_mins** the time (in minutes) between each try.*

Nice! We're now ready to test the integration 👍

4 Testing the integration

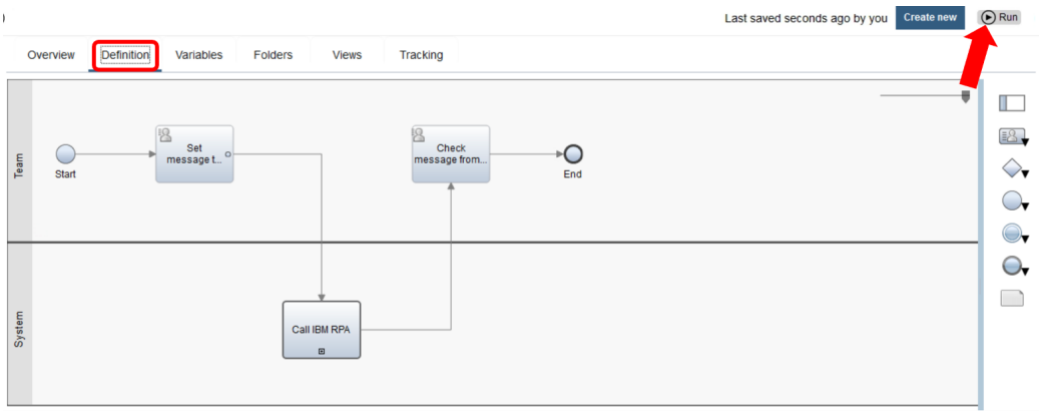
Finally here! You are now ready to test the integration. Let's just briefly recap what you needed to do before getting here.

- Started and set up BAW (imported the signer certificate and the RPA Toolkit)
- Created a simple RPA bot and published that to RPA server tenant
- Created a one-step RPA process definition (and all the related configurations) that uses the bot you created
- Created a BAW process application with one process definition to call the IBM RPA using the imported RPA Toolkit for BAW

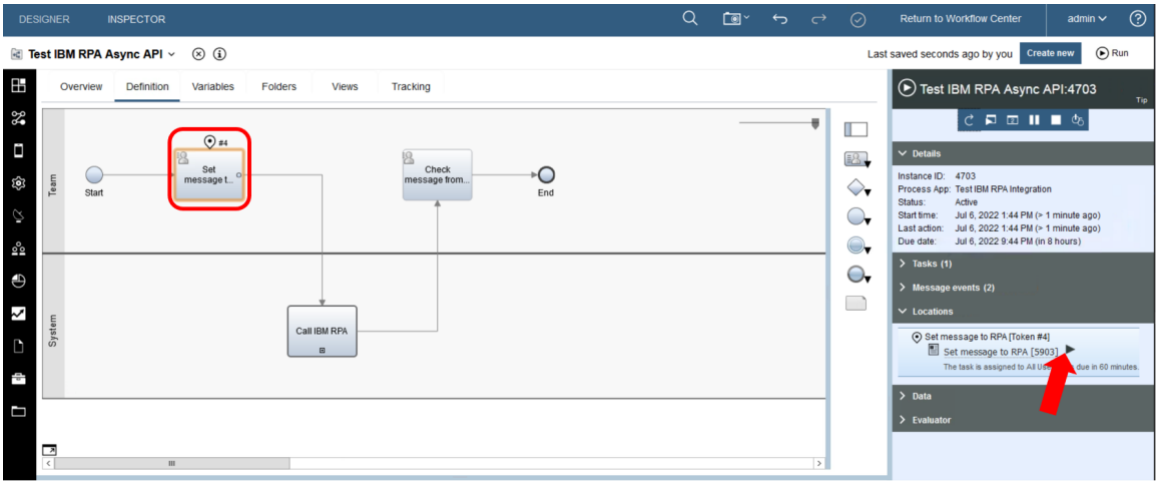
As you already have seen, there are many things to set up and configure in order to call out IBM RPA asynchronous API from IBM BAW. The toolkit helps quite a lot, but there's still a lot to consider. But you're now ready to test the integration, so let's do it!

4.1 Run the BAW process

100. Move back to Definition view of your process and run it as you have done earlier with toolkit services.

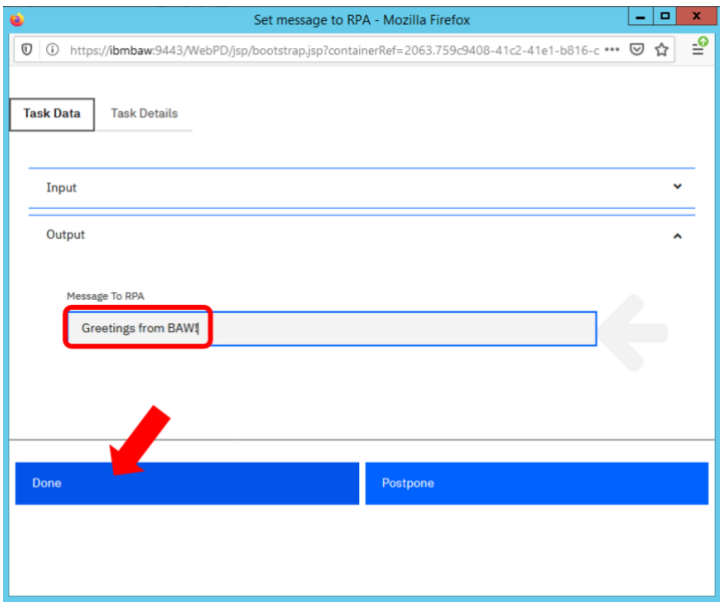


101. You will see process inspector / debugger view and that we're now ready to run the first activity (Set message to RPA). Click the "play button" under Locations from the right-hand side control section.

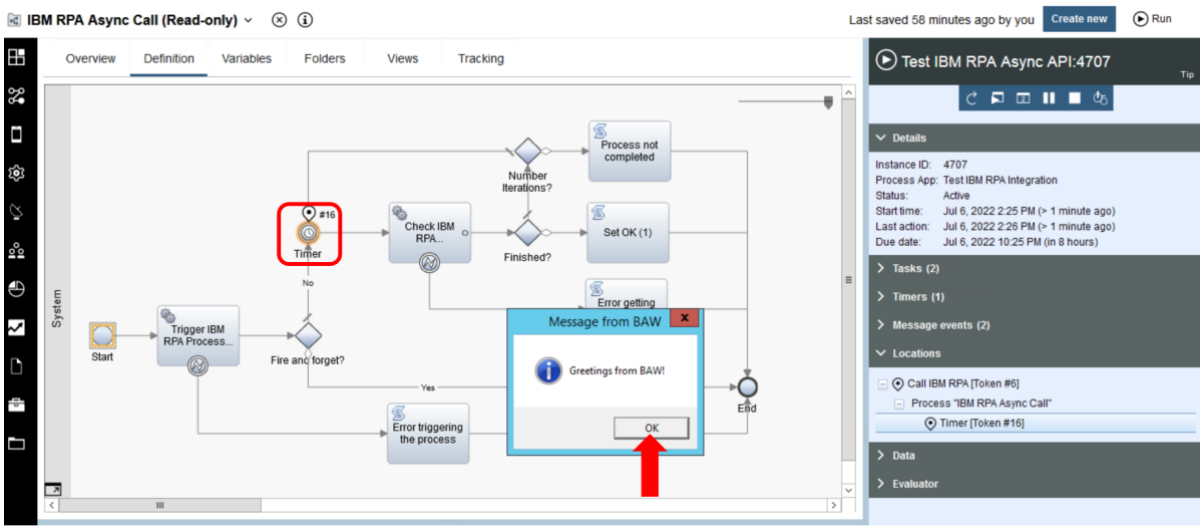


102. This will open our first Inline User Task activity UI, which are great for testing since they just let you to see and modify the mapped input/output variables defined to them.

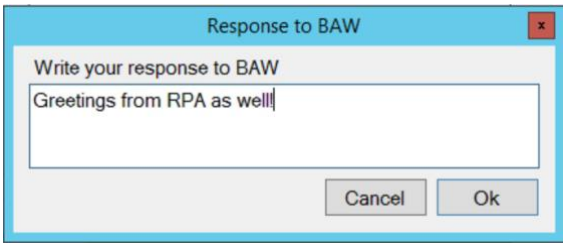
Now, **set the message** you want to use as the input_text variable to your RPA Bot and then click **Done**. Close the window when it says, “Service completed”.



103. Since the RPA is running in the VM, you should see the RPA bot running, since it displays the message box with the message you just set as the input for the bot. Also, the process inspector moves to show the implementation in the RPA toolkit. As you can see, we already triggered the bot by starting the RPA process definition we earlier configured and now the toolkit process has stopped to timer activity waiting for 1 minute, before trying to retrieve the results.

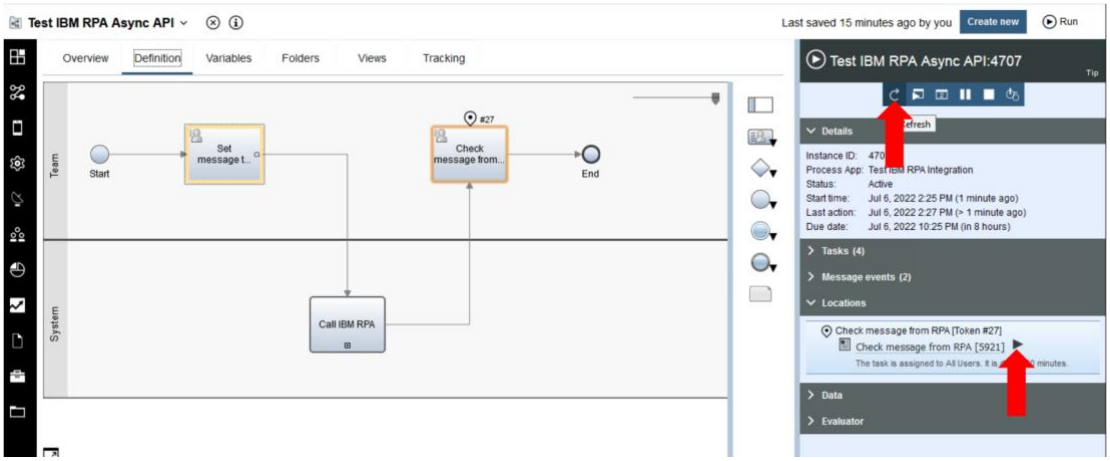


104. Click **OK** to close the Message from BAW message box. RPA bot will next display the input box we defined to our script. **Type in your response** back to BAW and click **OK** to finish the RPA bot.



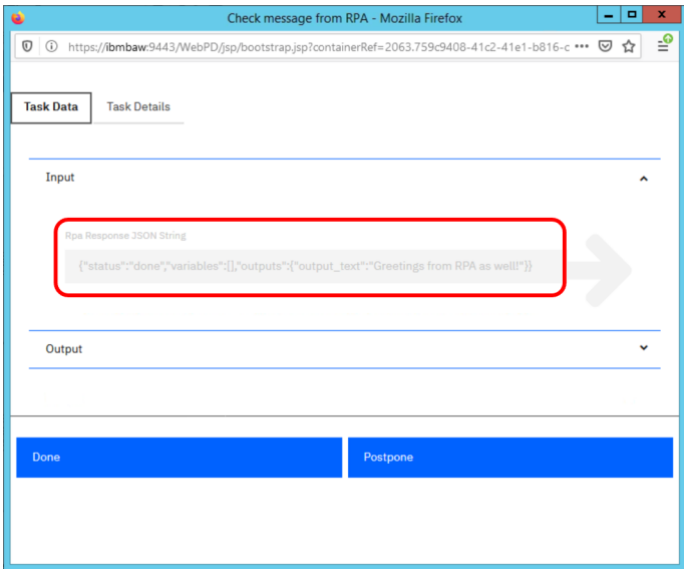
105. You can click the refresh icon in the inspector control section to see the latest process status. *Note that the process will wait until one minute has passed before trying to get a response back from the RPA server*.

When a minute has passed and you refresh, you should see the focus back on your main process with **Check message from RPA** activity activated. **Run** the activity as you did earlier by clicking the "play button" under locations.



106. The out-of-the-box UI for the activity is opened and you can see the response under Input section. The text might be hard to read because of the light colors of the default theme, but if look closely, you can see that the response is in JSON String format and the output_text element has a value of

"***Greetings from RPA as well!***". In order to extract just the output_text from the JSON String, you would need to do some extra parsing. If you're up for it, check out the optional 4.2. Parsing the bot output JSONSting.



107. Click **Done** to close the UI and as always, you can close the window saying, "Service completed".

Congrats! This ends the lab exercise. You went through all the needed steps to configure integration between IBM BAW and IBM RPA. As mentioned already earlier, there are quite many things to consider and hopefully this lab has given you enough information to understand the things we need to pay attention to when integrating BAW with RPA.

All feedback is welcome! You can reach me via email jukka.juselius@fi.ibm.com, don't be strangers!

THANKS FOR PARTICIPATING!

4.2 Parsing the bot output JSONString (optional)

The async RPA API returns the bot "results" in certain structure that is described in the API documentation. Let's have a look at our response JSONString that represents a data object.

```
{"status":"done","variables":[],"outputs":{"output_text":"Greetings from RPA as well!"}}
```

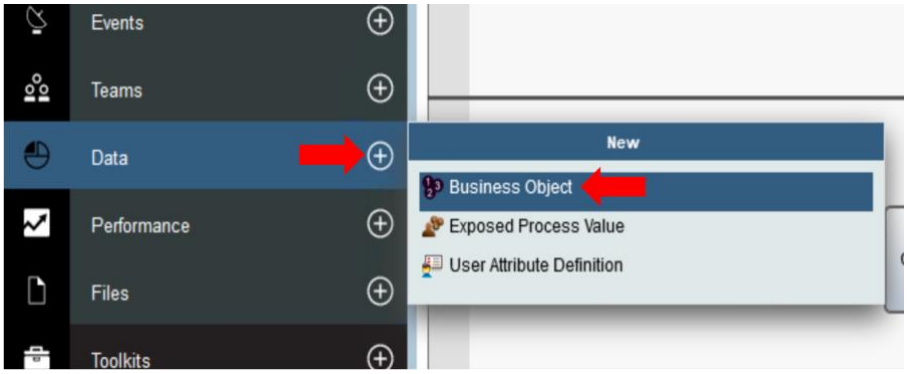
First element in the response is the **status** of the process that was executed. The toolkit logic uses this information to see if the process has already executed or still running.

Next element is the list of **variables**. These would be the variables that we have defined into our RPA Process definition and used them as part of our bot logic. We did not do this, hence the list is empty.

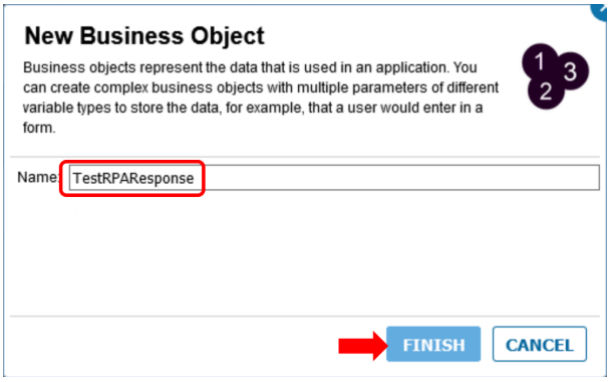
The last element is called **outputs**. This an object itself that holds all the output variable names and their values, in our test just "output_text" that we set to "Greetings from RPA as well" with the input box that was opened for us when the bot was running.

Since the response is a JSONString we can easily model it as BAW business object and then use JSON.parse() to map the JSONString to a business object that has the same elements. Let's do it!

108. **Click the plus sign (+) for Data** from the left-hand side menu and then select **Business Object**.

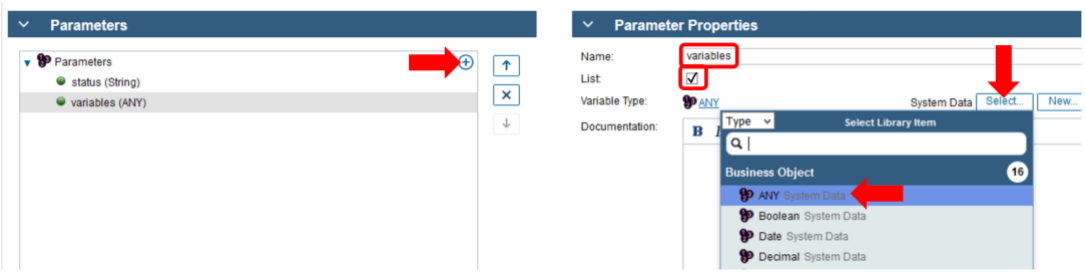


109. Name the business object **TestRPAResponse** and click FINISH.

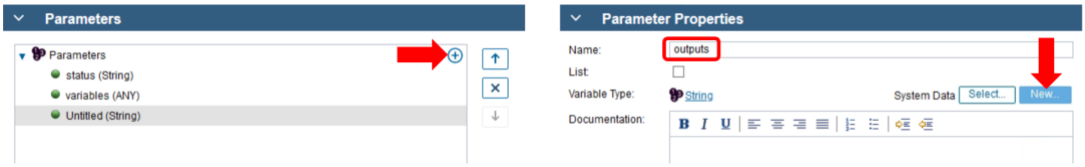


110. This will open the business object editor. Click the plus sign (+) to add new parameter. Name it **status**. Note! When typing the name, hit Enter to make the name set and shown also under parameter list.

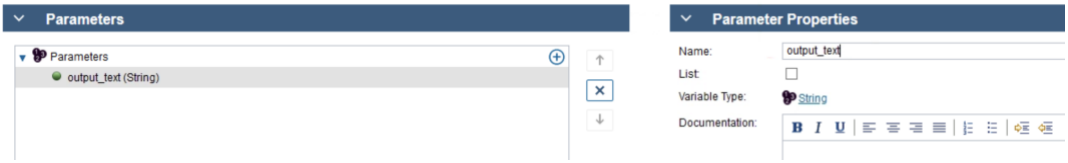
111. Add another parameter for **variables**. Since it was a list, *make sure to check the **List** option!* We can use variable type ANY in the situations where we do not care, or we do not know the type of the parameter. Click the **Select button** for the variable type and **select ANY**.



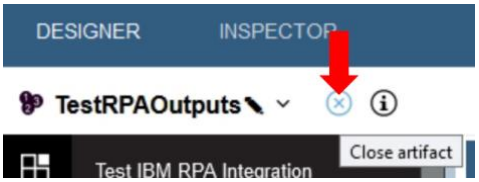
112. The final element in the JSONString response was called **outputs** and it is an object also itself. So, we need to create an embedded business object to represent it. **Add a new parameter** for it, name it **outputs** and then click the **New button** for the variable type.



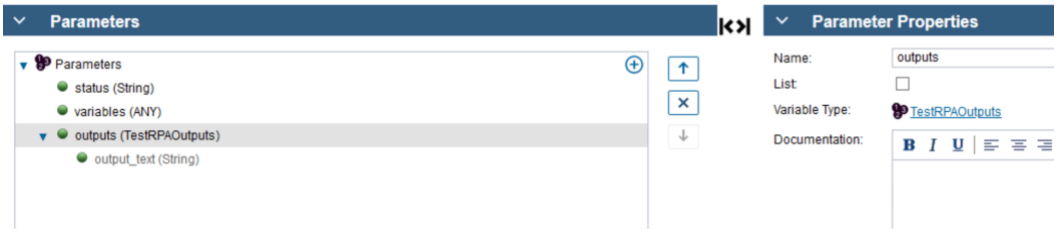
113. Name the new business object **TestRPAOutputs** and click **FINISH**. This opens an empty business object editor. Now, the name of the attributes we need to add here really comes down to what our RPA bot has defined as output variables. Since our test bot has only one – *output_text* – we only need to add that. Go ahead and add **output_text** (String) parameter.



114. You can close the editor for TestRPAOutputs by **clicking the X sign** on the top of the editor.

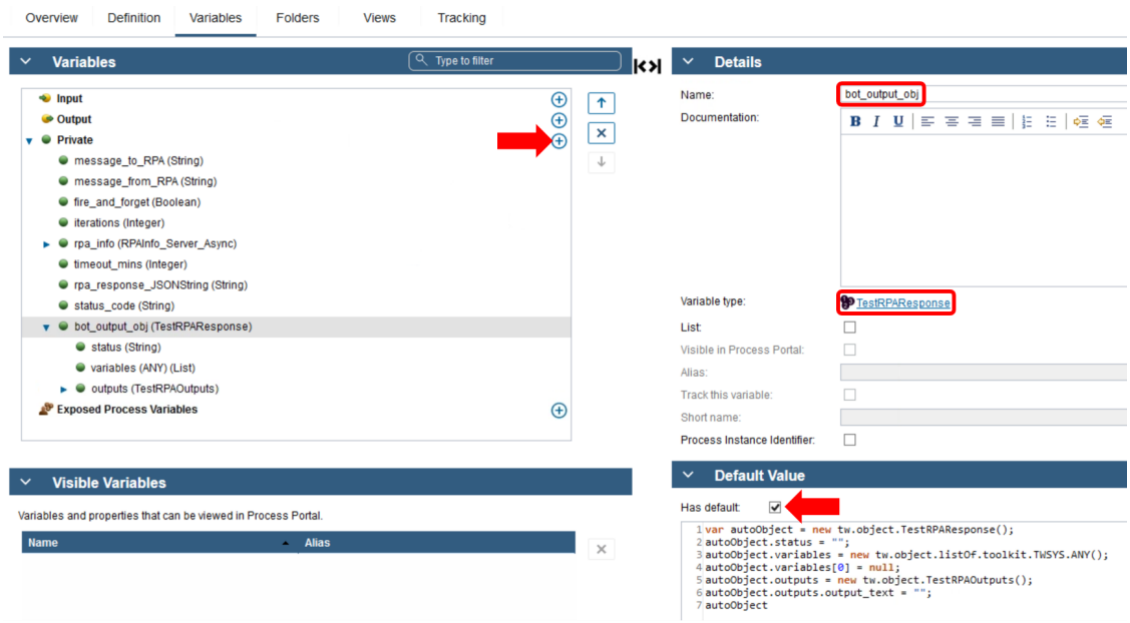


115. You again see the main business object that we were defining – *TestRPAResponse* – with **outputs** of type **TestRPAOutputs** has been added as the last parameter.

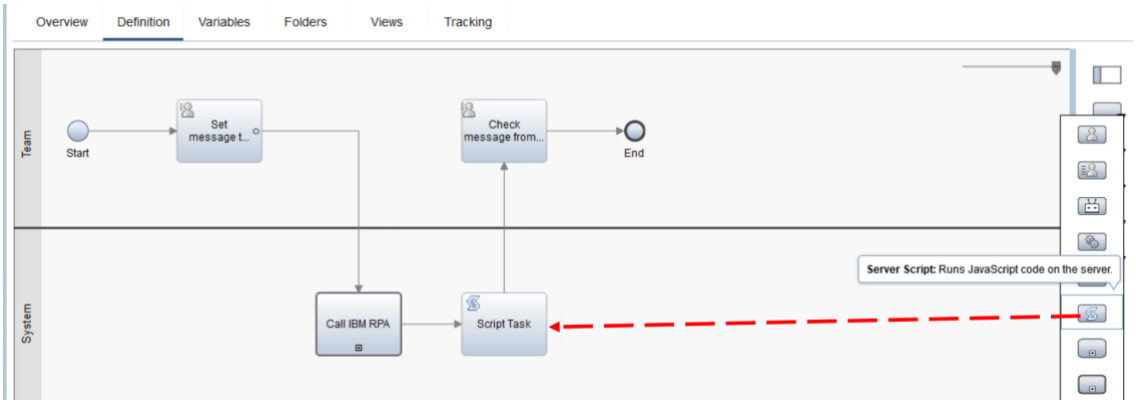


Go ahead and **close** this one as well since we're done defining it.

116. Now we need to add a new variable for this new business object in order make use of it in our process. Move to variables view of your process and **add a new Private variable** named **bot_output_obj** type of **TestRPAResponse**. Also **check the "Has default" option** to initialize the business object for our use.

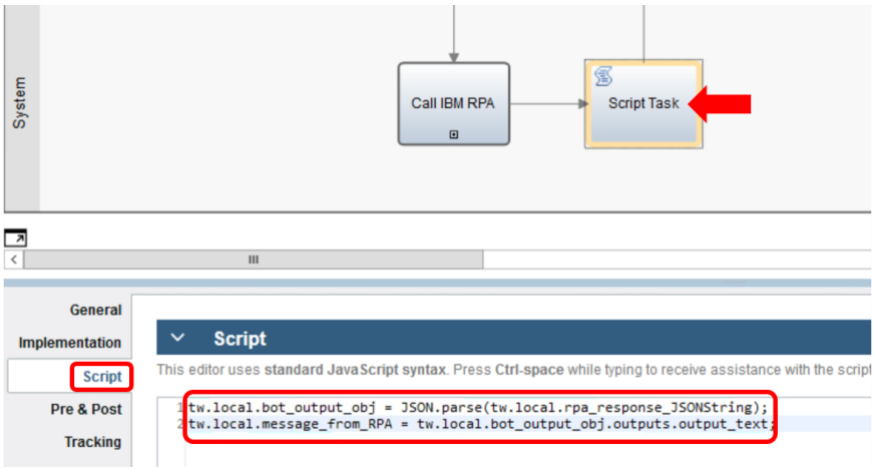


117. Back in Definition view, drag and drop a **Server Script activity** to System lane and **connect it** as shown in the picture below. We will use to map the JSONString response to our new variable.



118. Click the **Script Task** to select it and then write / copy-paste the following script to the **Script section** of the activity configuration.

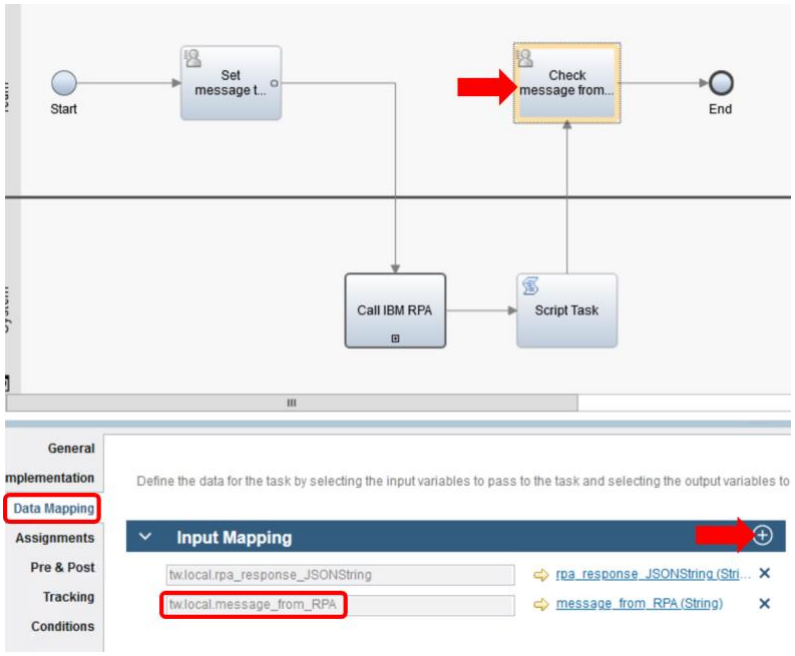
```
tw.local.bot_output_obj = JSON.parse(tw.local.rpa_response_JSONString);
tw.local.message_from_RPA = tw.local.bot_output_obj.outputs.output_text;
```



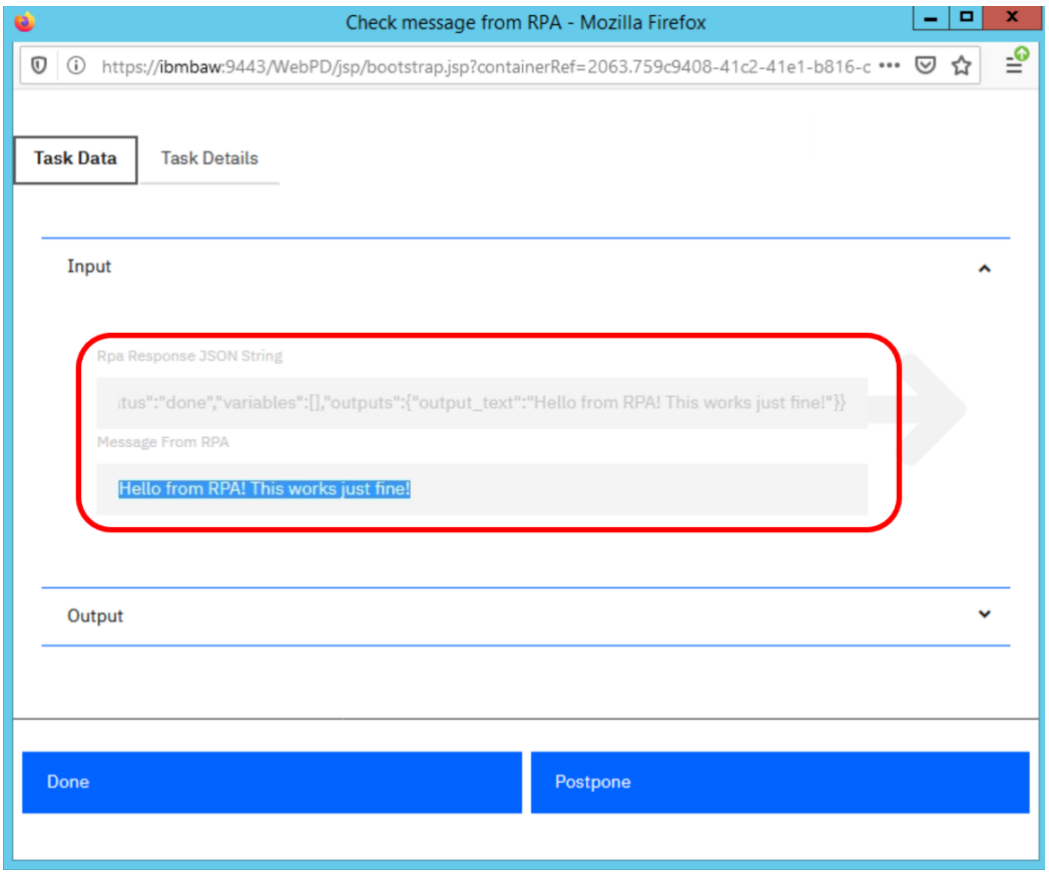
What we're doing here, is to first assign the data from the RPA response JSONString directly to our newly created business object / variable using JSON.parse() method. ***NOTE!*** *This only works, if the structure and the parameter names of the business object matches with the data in JSONString*.

Secondly, we're using the variable *bot_output_obj* – that we just populated from the JSONString reply from RPA – to assign the value to our *message_from_RPA* variable, using just the output_text value.

119. **Add** a new **Input Mapping** to the **Check message from RPA** activity for the **message_from_RPA** variable.



120. Nice! You can now run the process again. This time you should see the both the JSONString response AND just the output_text from the bot in your Check message from RPA activity when you reach it.



You can use this approach to quite easily extract just the RPA output variable values to be used in your BAW process. Hope you enjoyed the lab!