IBM Cloud Pak for Business Automation Demos and Labs

IBM RPA and Workflow Integration

V 4.0

Bu Feng Hou
houbf@cn.ibm.com
Paul Pacholski
pacholsk@ca.ibm.com
Olaf Hahnl
olaf.hahnl@de.ibm.com
Aldo Justiniano
aldo.justiniano@ibm.com
Marco Crepaldi
marco.crepaldi@ibm.com

NOTICES

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

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any referenceto 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

Table of Contents

1 Introduction

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

- **Unattended bots**
  Use an RPA-driven digital workforce to automate repetitive tasks without human intervention.
- **Attended bots**
  Remote Desktop Automation (RDA) enables a human workforce to augment work using bots to perform repetitive tasks on demand.
- **Optical Character Recognition (OCR)**
  Process documents by extracting structured data from unstructured content.
- **Dashboards**
  Gain business insights into business operations.

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

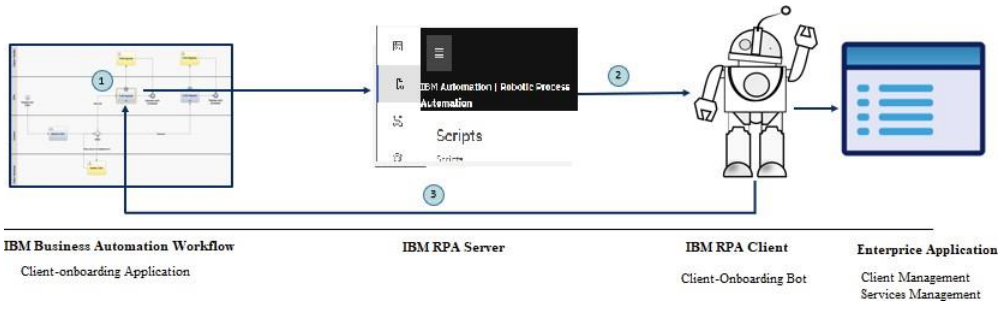
- **Faster time to value**
  Speed and simplicity of purchasing and deploying through easier licensing.
- **A comprehensive platform to automate all types of use cases**
  Tighter integrations between RPA and the rest of our platform.
- **Automate business and IT processes**
  Expand our automation mission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

You can explore the Documentation to understand more details about IBM RPA.

2 Overview

In this lab, you will learn how to invoke a bot authored in RPA Studio from a process developed with Business Automation Workflow.

The integration steps between Workflow and RPA are as follows:



1. Workflow reaches the activity where a bot should be invoked. The process designer models the activity to call a service flow that uses an external service that invokes an

RPA bot using the IBM RPA REST APIs. IBM RPA offers three types of APIs to start bots. One is a synchronous unprotected REST API to run the bot via a client-side API, this API has been deprecated since RPA v23.0.1. Another approach is to start the bot in an orchestrator process via server-side asynchronous APIs in a protected way. The 3rd approach is to start bot via bot APIs, please refer to documentation for details. In this lab, we will use the 2nd approach to start the bot from a workflow.

2. The RPA server deploys the bot by passing business data to the RPA Client.
3. The RPA client performs the actual work by executing the bot script. The bot script will add client-onboarding information and signed services into backend applications as part of the end-to-end client onboarding solution. Once the execution is completed, it will pass the output data back to the Workflow process with a status code indicating if the bot script execution was successful or not.

2.1 Pre-requisites

For this lab, you need to access:

- **IBM Robotic Process Automation**: You need to reserve a lab environment from IBM Technology Zone, as explained chapter 3.
- **IBM Business Automation Workflow Center**.

All the pre-requisites have been pre-installed/configured in the lab template. The information below is just for information purposes.

IBM Products:

- IBM Robotic Process Automation v23.0.x.
- IBM Business Automation Workflow v22.0.x.

Custom Solutions/Code:

- The IBM RPA Toolkit - contains the service flow to start the bot matching the information required by the two backend systems in an orchestrator process via server-side async APIs.

  This toolkit is just a sample implementation of invoking the RPA server-side APIs. It simplifies the workflow and RPA integration for this lab only. You may want to study how it is implemented, but please don't directly use it in any real customer project.

- A Java swing application simulating the backend, third-party system for the Client Management System.
- A web application simulating the backend, third-party Services Management System for managing the services a client has signed up to.

2.2 References

1. IBM Robotic Process Automation Documentation

2. IBM Robotic Process Automation Command Documentation

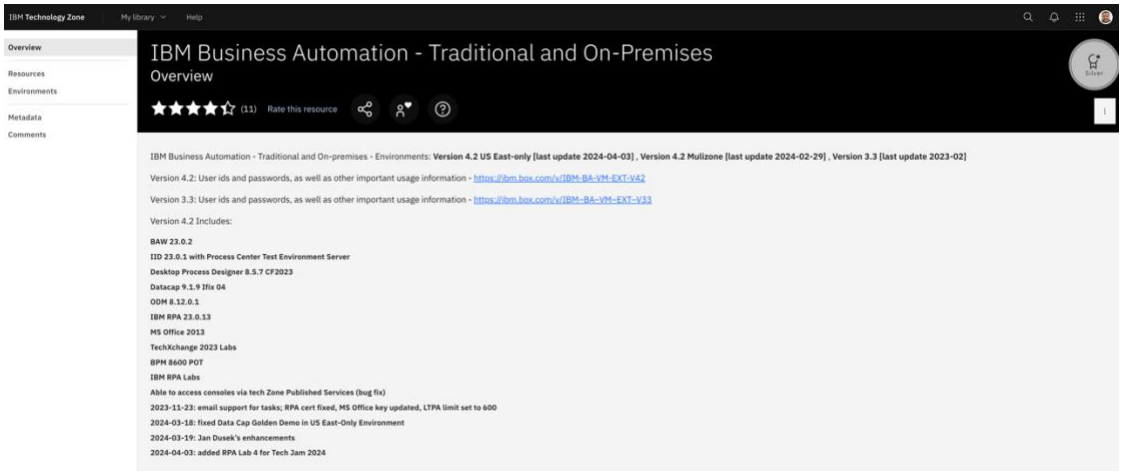
3 Accessing the Environment

If you have already reserved a lab environment from IBM Technology Zone, please go to Chapter 5 directly.

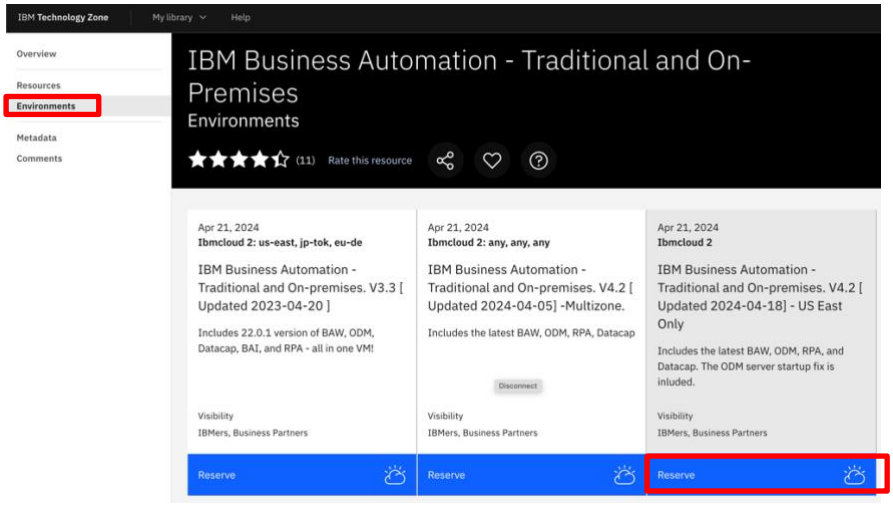
Reserve Environment

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

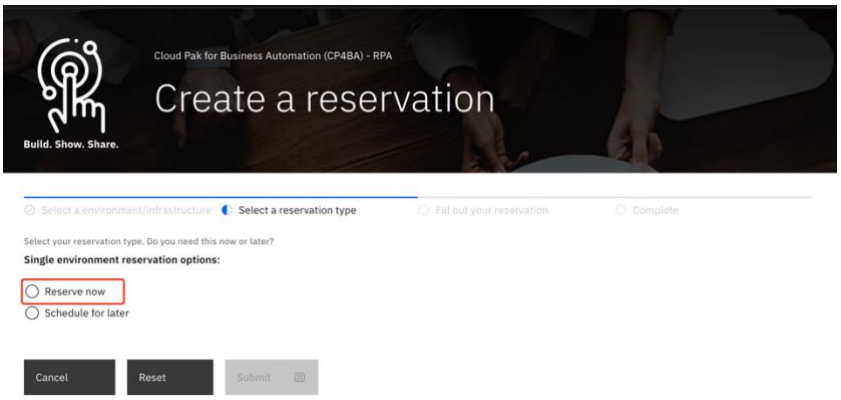
1. Click here to open IBM Technology Zone Reservation portal. You need to use your IBMID to login to the portal.



2. Click **Environments** on the left panel, and then reserve the last environment on click the blue button.



3. Select **Reserve for now**, then click **Submit**.



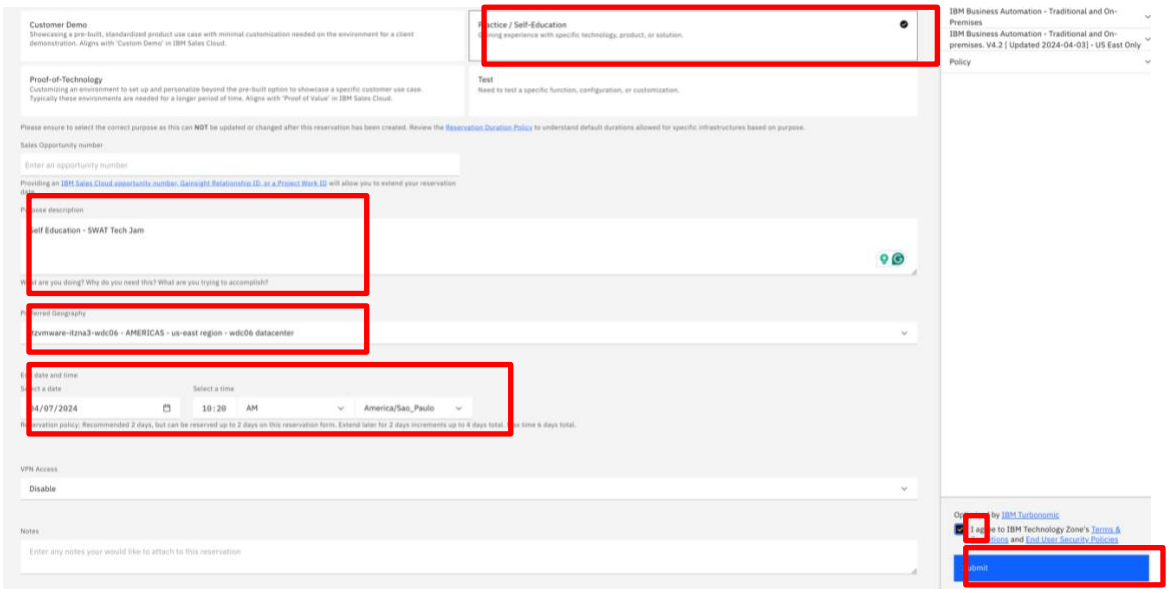
4. On the reservation page, make the appropriate selections as below. Once done, click **Submit**.

   Purpose: Select Practice/Self-Education.

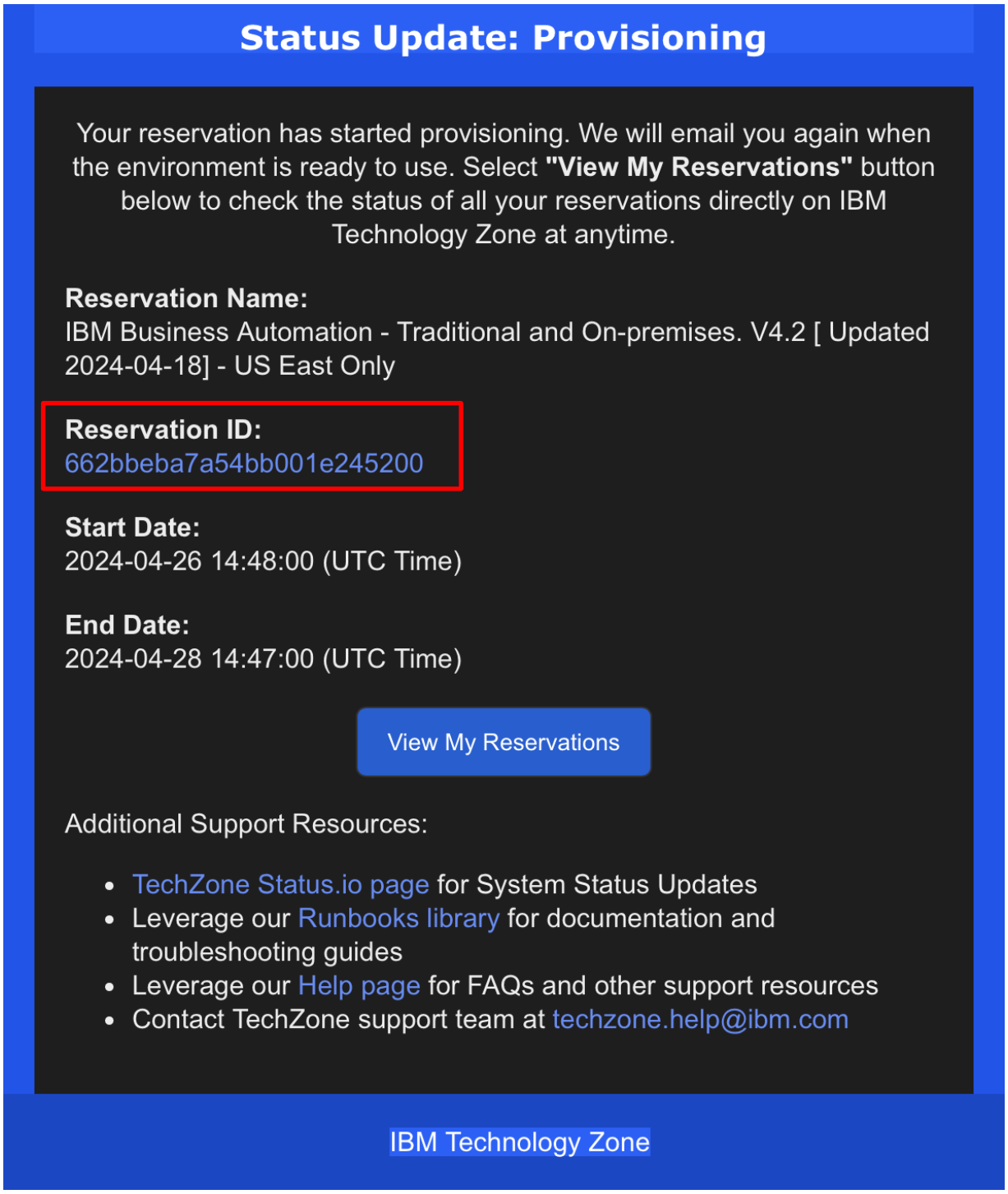
   Purpose description: Enter something like **Self Education**.

   End date and time: Select the end date and time that the environment will be deleted.

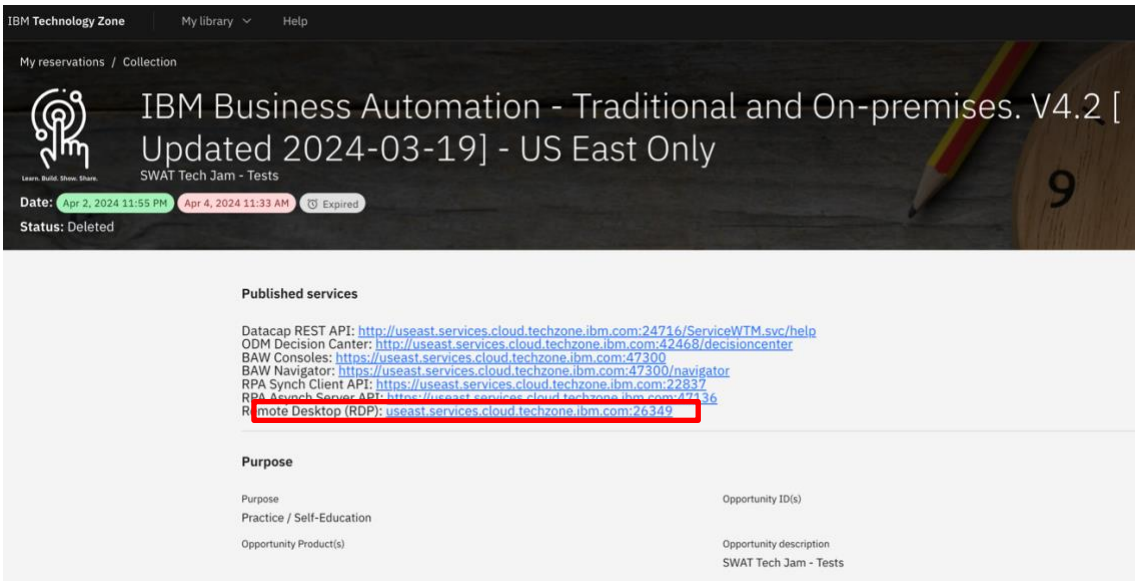
   Preferred Geography: Select the geography where your environment will be created. To get a better network connection, select the same geography as where you are located in.

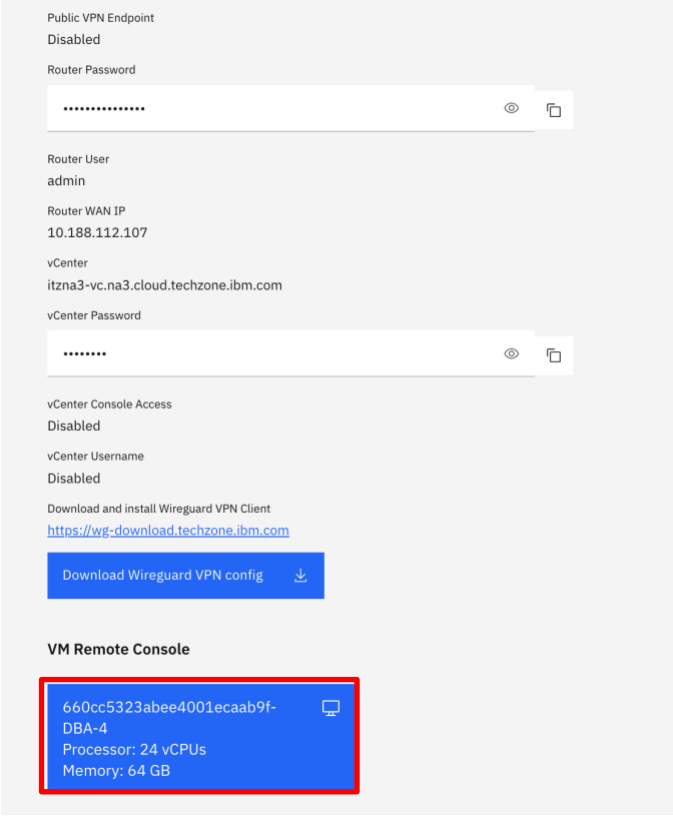


5. Once you have reserved an environment, you will receive an email with a link to access the environment's management console, click on Reservation ID.

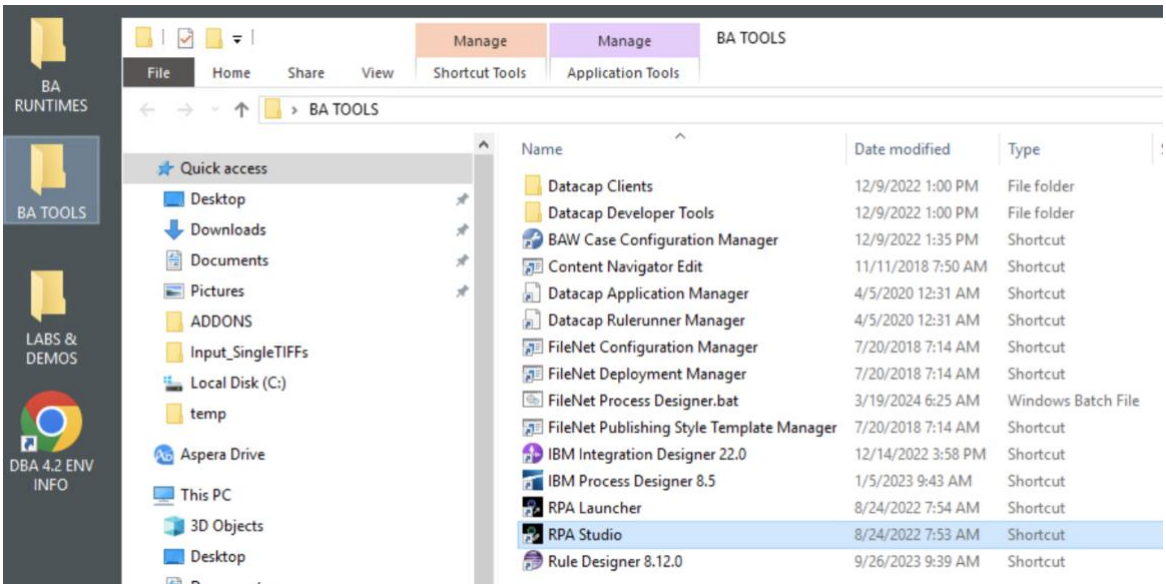


6. You can access the environment using Remote Desktop (RDP) or Remote Console (Web). Our recommendation is to use Remote Console (Web) for practicality. If you prefer to use the RPD, use the Remote Desktop (RDP) link, or keep rolling the page to access the Remote Console (Web).





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



4 Build it yourself – Step-by-step instructions

IBM RPA provides REST APIs for other applications to start bots. In this exercise, you will learn how a Workflow process activity can call an RPA bot to automate a swivel-chair task so far performed by a human. It will take about one and half hour to complete this exercise.

In the sequence of the scenario flow, it is assumed that the bot script is created first or is already available in the enterprise. You will model the business process and modify the implementation and data mapping accordingly to call the bot from the Process. In this exercise, you can use your script if you have performed the **Application Automation Using IBM RPA** lab. Or you can use the **ClientManagement** script, which has already been published to the tenant.

The IBM RPA Toolkit has been provided which provides various functionalities, including a data model that the client onboarding application uses and various service flows to start bots configured to run by an orchestrator process via server-side asynchronous APIs.

This lab contains two exercises that replicate the two steps required to start bots configured to run by an orchestrator process via server-side Async APIs. The first exercise is to create an orchestration process in the IBM RPA control center. The second exercise is to call RPA server-side APIs to create an orchestration process instance that will automatically start the bot execution.

4.1 Create the RPA Orchestration Process

The server-side API uses a so-called orchestration process in the RPA server to execute the bot script. You must create and configure the orchestration process before you can call the API to request the execution of the bot. In this exercise, you will learn how to create and configure an orchestration process that contains the following three steps,

1. Create a script in the RPA studio and publish it to the RPA server
2. Create queues in the RPA control center
3. Create an orchestration process

4.1.1.1 Create and Publish Your Script

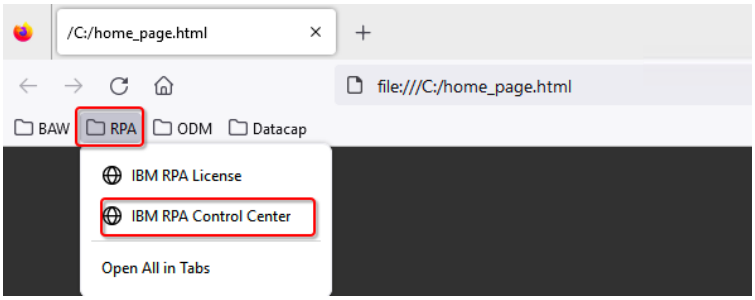
Bots in IBM RPA are scripts developed using IBM RPA Studio. Please refer to Script Development to learn how to use Studio to develop the bot script. In this exercise, you can use your script if you have performed the **Application Automation Using the IBM RPA** lab. Or you can use the **ClientManagement** script published into the RPA server already.

4.1.1.2 Create a Queue

1. Log into your environment using remote desktop if not yet, click **BA TOOLS** icon on the windows desktop.

2. Start the **Firefox** browser by clicking the Firefox icon from Windows toolbar.

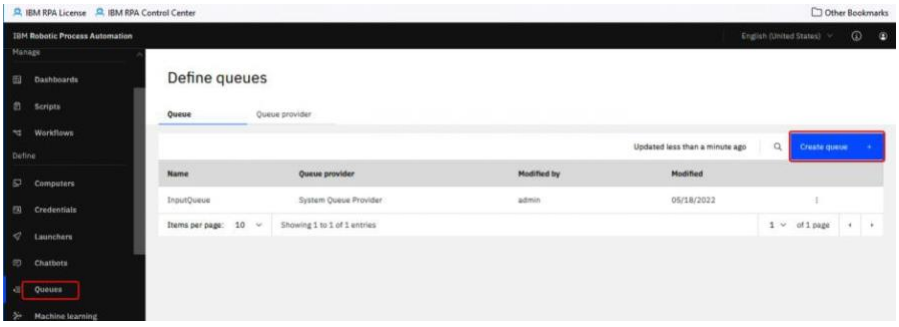
3. Click **RPA → IBM RPA Control Center** in the Firefox bookmark toolbar to launch the Control Center. You will first be taken to the login page.



4. Enter admin@rpa-poc.com as **User name**, then click **Continue**.

5. Tenant will be set to **rpa-poc** by default automatically, enter **passw0rd** (make sure to use zero as part of the password) as the password, and click **Login** to log in to the IBM RPA Control Center.

6. Click **Queues** in the left panel. Then click **Create queue**.

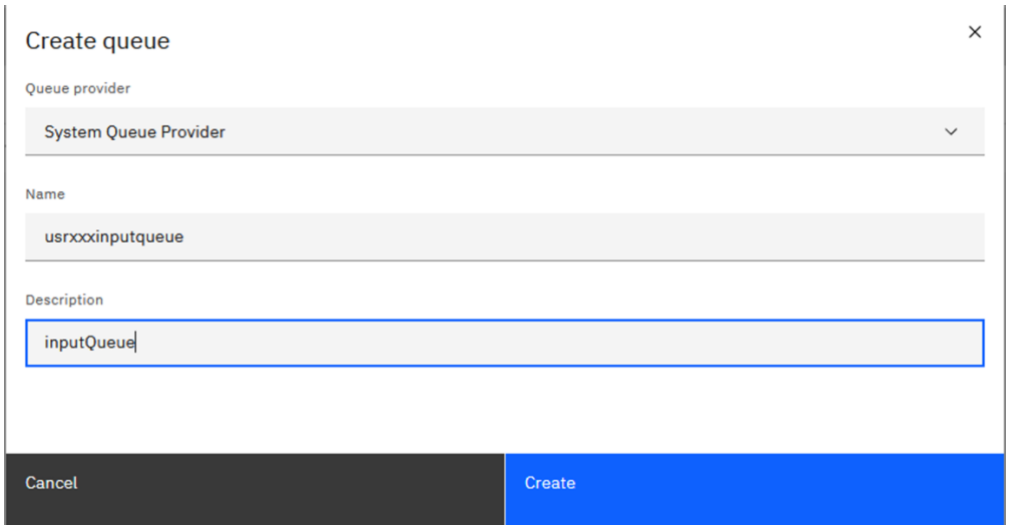


7. Configure the queue as below. Once done, click **Create**.

   Queue provider: Select **System Queue Provider**.

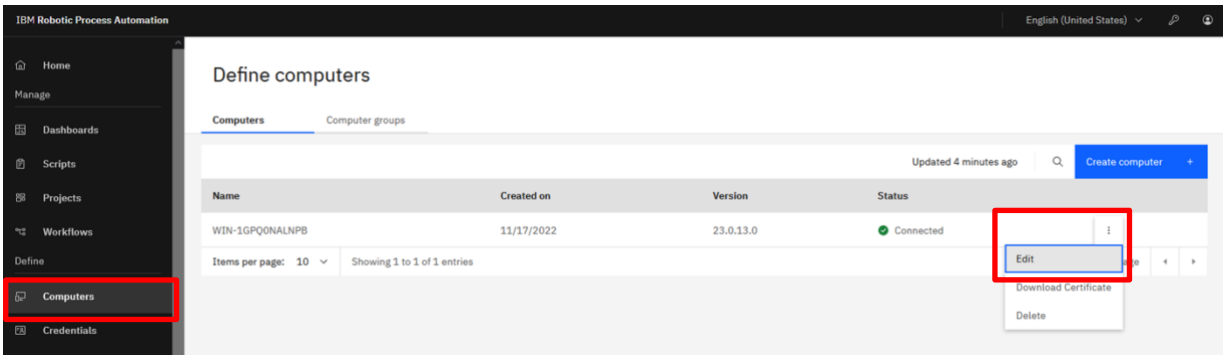
   Name: Enter a unique name; for example, prefix with your user name: **usrxxxinputqueue**.

   Description: Enter a description.



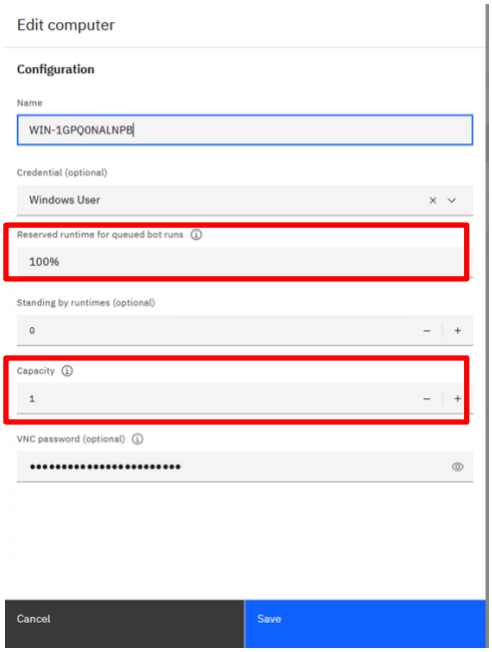
4.1.1.3 Configure computers for queue runs (no action)

1. *Suppose your environment has multiple computers, and you have selected them in the steps configuration, then for each message in the input queue. In that case, the bot will be started on one of the computers, providing it has an available runner license.*
2. *To check if your computer has allocated enough amount of runner licenses for an orchestrator process to execute the bot script, click* ***Computers*** *from the left panel, click the three-dot button next to your computer, then click* ***Edit****.*



3. *The* ***Reserved runtime for queued bot runs*** *number at the bottom is the number of total runner licenses allocated to this computer. The Queues runtime percentage represents the percentage of t*
4. *The* ***capacity*** *number at the bottom is the number of total runner licenses allocated to this computer. The Queues runtime percentage represents the percentage of total runner licenses allocated to execute the orchestrator process. The number of runner licenses allocated for an orchestrator process equals capacity multiply queue runtime percentage, it must be equal or greater than 1.* ***If the value of the capacity field is equal to zero, the items in the queue will not be executed.*** *In this lab, the fields are configured for the queue*

orchestrator to work.

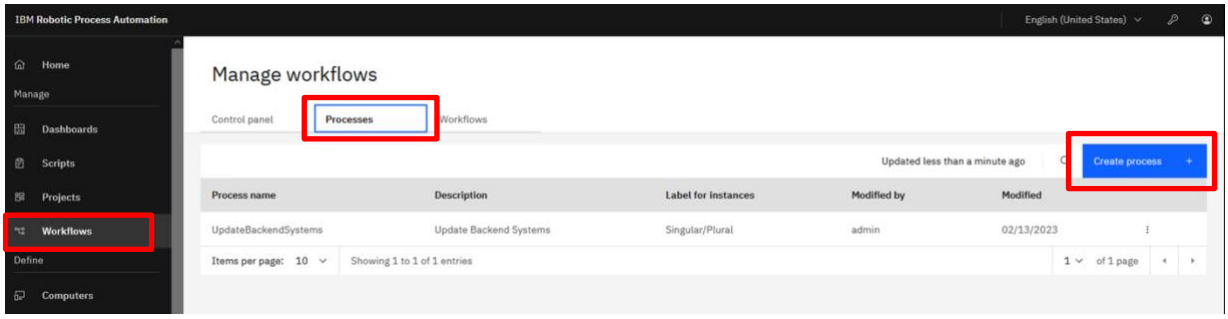


5. *It also requires credential to unlock the computer when bot executes on it. The encrypted credentials are stored in vault. IBM Robotic Process Automation provides two types of vaults, one is System Vault (or RPA Vault) which is recommended for unattended automation, another is User Vault (or RDA vault) which is recommended for attended automation. System vault will use public key to encrypt credentials and private key to decrypt credentials which are configured in the RPA control center. Please refer to Vault for details*

4.2 Create Process to Invoke RPA Bot Asynchronously

This exercise will continue in the IBM RPA control center.

2. Click **Workflows** in the left panel in the RPA control center. Select the **Processes** tab, and then click **Create Process**.



3. Configure the Process **General** information as below. Click **Next** once done.

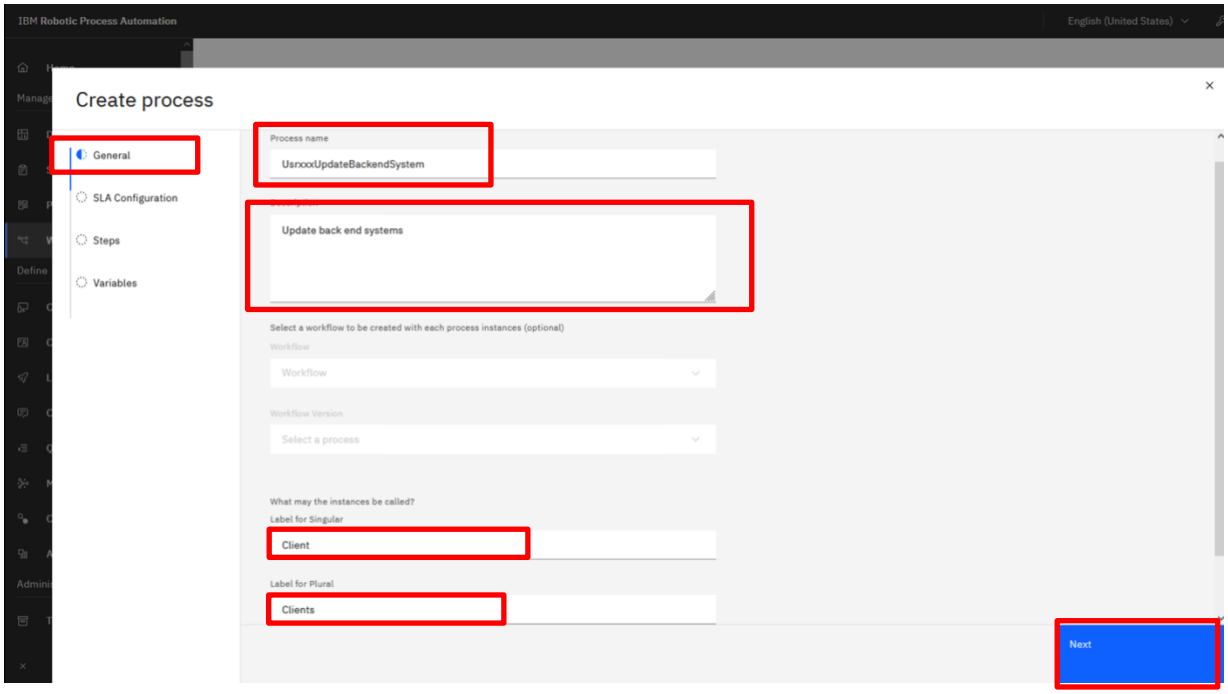
Process Name: Enter a unique name to identify the Process; for example, prefix it

with your user name **UsrxxxUpdateBackendSystem**.

Description: Enter a description of your Process.

Label for Singular: Enter the name to identify a single process instance; for example, **Client**

Label for Plural: Enter the name to identify multiple process instances; for example, **Clients**



4. Configure the process **SLA** settings as below. Click **Next** once done.

   Target Waiting Time: Enter the expected average time interval for a process instance to wait at most to start being processed, for example, 00:02:00.

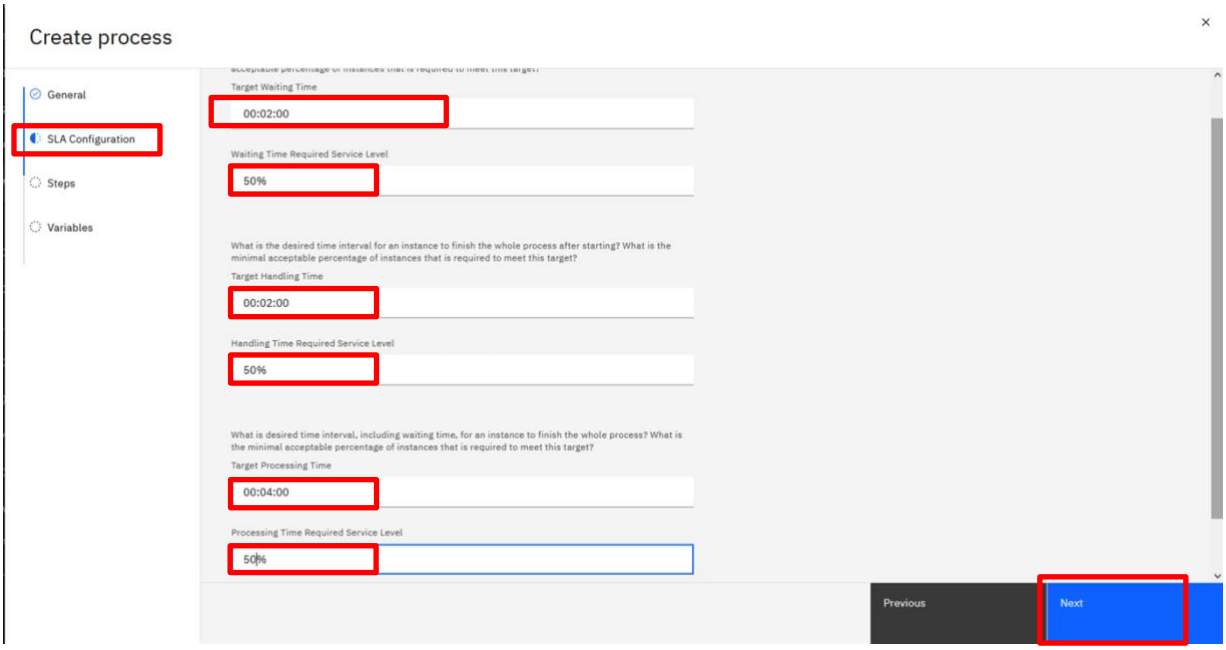
   Waiting Time Required Service Level: **E**nter the minimum acceptable percentage of instances required to meet this target, for example, 50%.

   Target Handling Time: Enter the expected average time it should take a process instance to finish after it got started, for example, 00:02:00.

   Handling Time Required Service Level: Enter the minimum acceptable percentage of instances required to meet this target, for example, 50%.

   Target Processing Time: Enter the expected average time interval, including the waiting time for a process instance to finish the entire Process, for example, 00:04:00.

   Process Time Required Service Level: Enter the minimum acceptable percentage of instances required to meet this target: 50%.



5. Configure the process **Steps** as below. Click **Next** once done.

   Step Name: Enter the step name, for example, **Update Backend Systems**.

   Input Queue: Select the input (**usrxxxinputqueue**) queue you created in the Create a Queue section. **Output Queue (On Success)**: Select **Mark as success**.

   Priority on Success Queue: Select **Normal**.

   Output Queue (On Error): Select **Mark as error**.

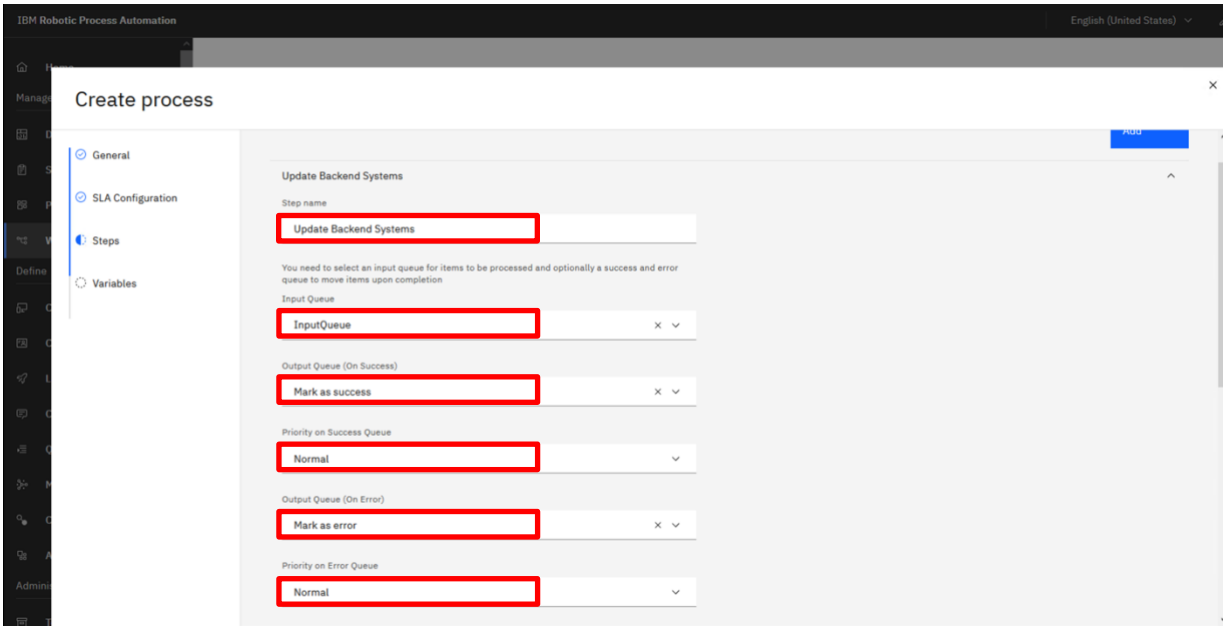
   Priority on Error Queue: Select **Normal**.

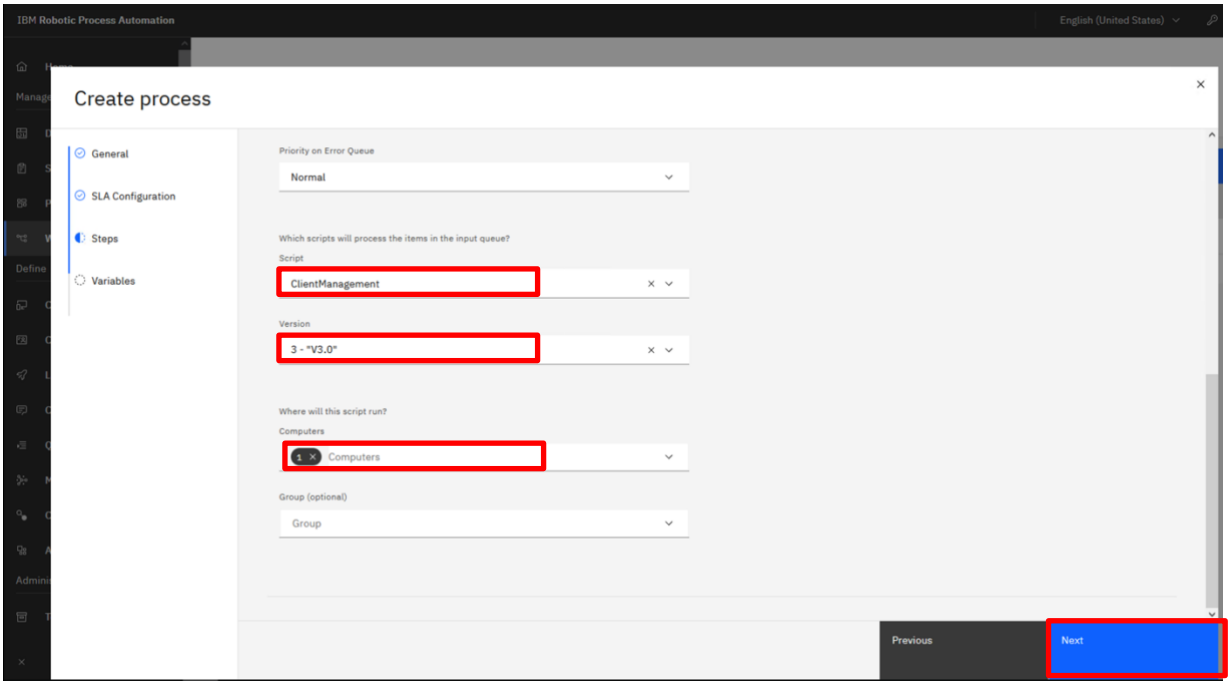
   Script: You can select your script if you have performed the **Application**

Automation Using IBM RPA lab or the **ClientManagement** script published to the server.

Version: Select the latest script version published to the server.

Computers: Select the only computer configured in your RPA environment to run bots.





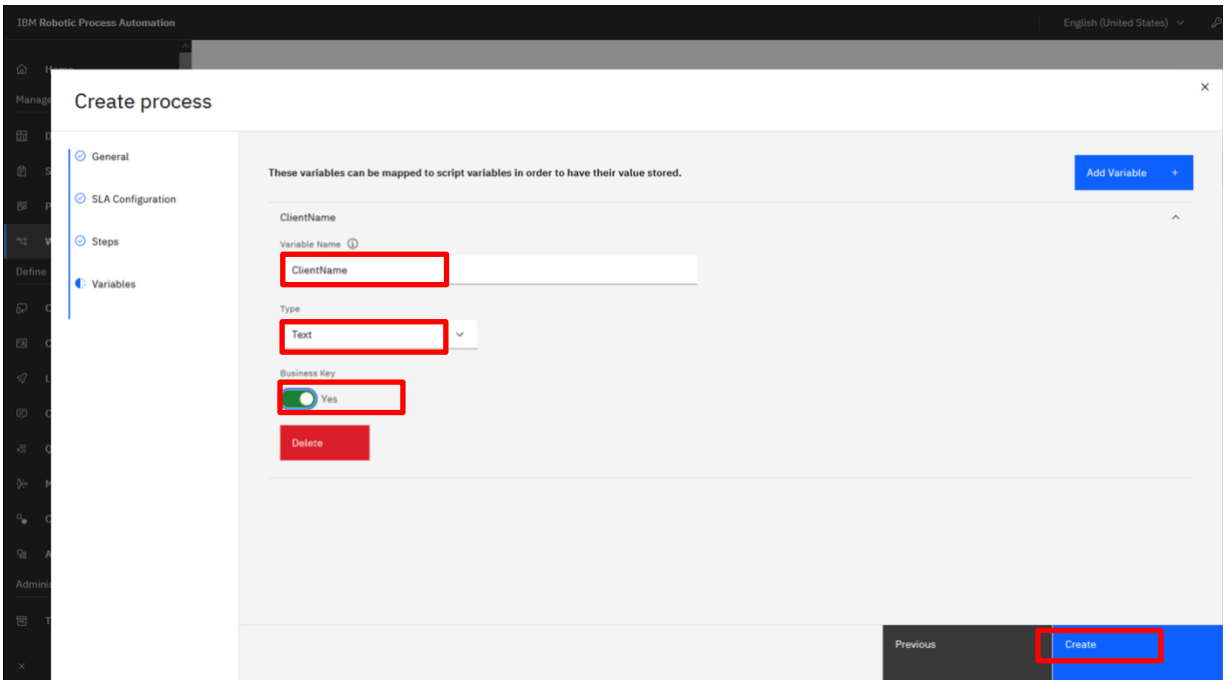
6. Configure **variables** as below. You can bind these variables to your scripts to persist data and track change history. During the process instance execution timeframe, you

can watch the process variables' values in real-time via Control Center's dashboard. Click **Create** once done.

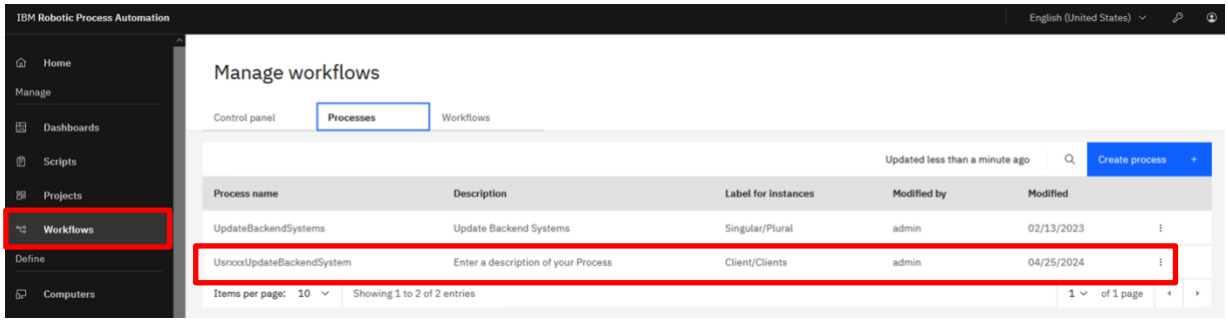
Variable Name: Enter the variable name, for example, ClientName.

Type: Select the variable type as Text.

Business key: set this variable as a business key. You need at least one variable set as a business key.



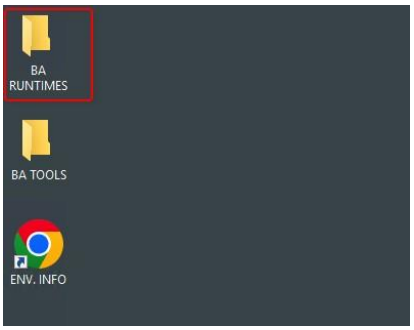
7. Your orchestration process should be created and listed in Manage Workflows view below.



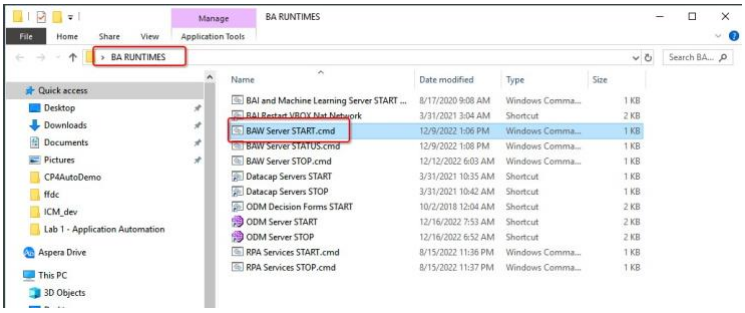
4.2.1 Explore the IBM RPA Toolkit

After creating an orchestration process in RPA control center, the next step is to call the RPA server-side APIs to create the process instance, which will automatically execute the bot script in the RPA client machine. As mentioned above, to simplify this lab, a sample IBM RPA toolkit has been implemented to simplify the integration of Workflow and RPA. Before creating the workflow process, explore the IBM RPA toolkit to familiarize yourself with the IBM RPA server-side APIs.

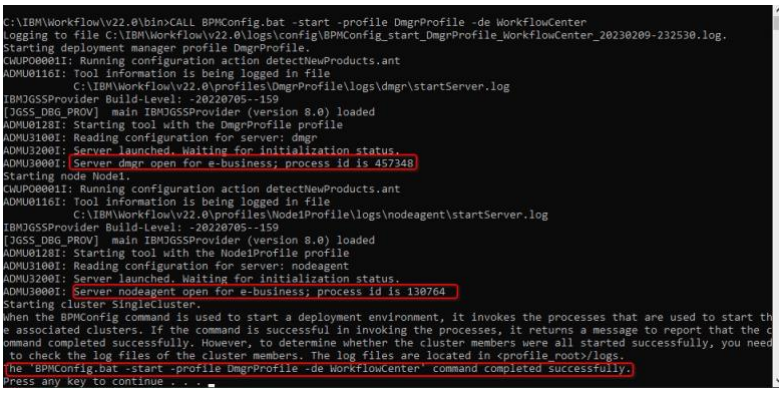
1. Access your environment VM using remote desktop if not yet, open **BA RUNTIMES** folder from windows desktop.



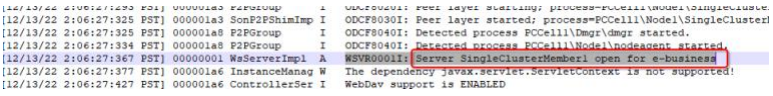
2. Double click **BAW Server START.cmd** shortcut to start IBM Business Automation Workflow server.



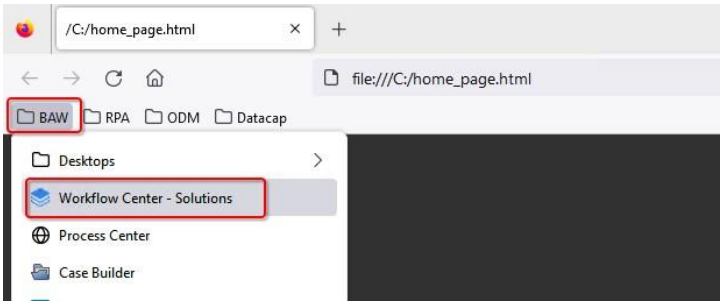
3. It will take a few minutes to start Business Automation Workflow server. Review the output and make sure server **dmgr** and **noteagent** are started successfully and the command to start a deployment environment is invoked successfully. Press **any key** to close the command window.



4. It will take a few minutes to start the cluster. To check if cluster is started successfully or not, you can check **SystemOut.log** located in *C:\IBM\Workflow\v22.0\profiles\Node1Profile\logs\SingleClusterMember1*.



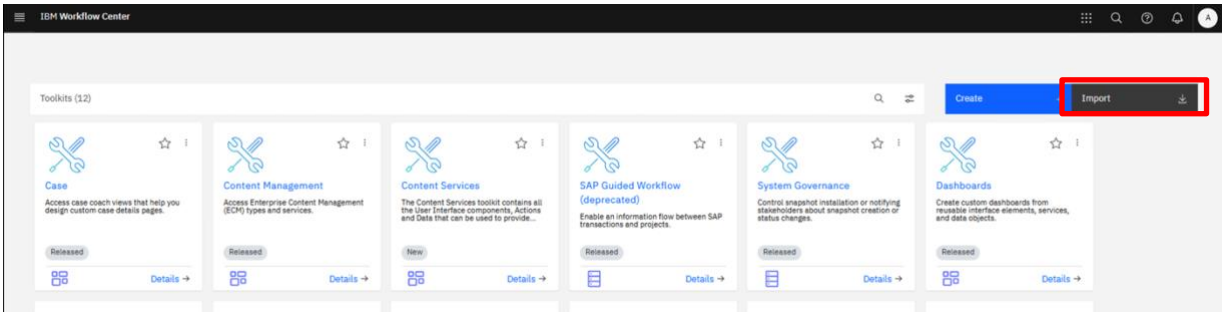
5. Start **Firefox** and select **BAW → Workflow Center - Solutions**.



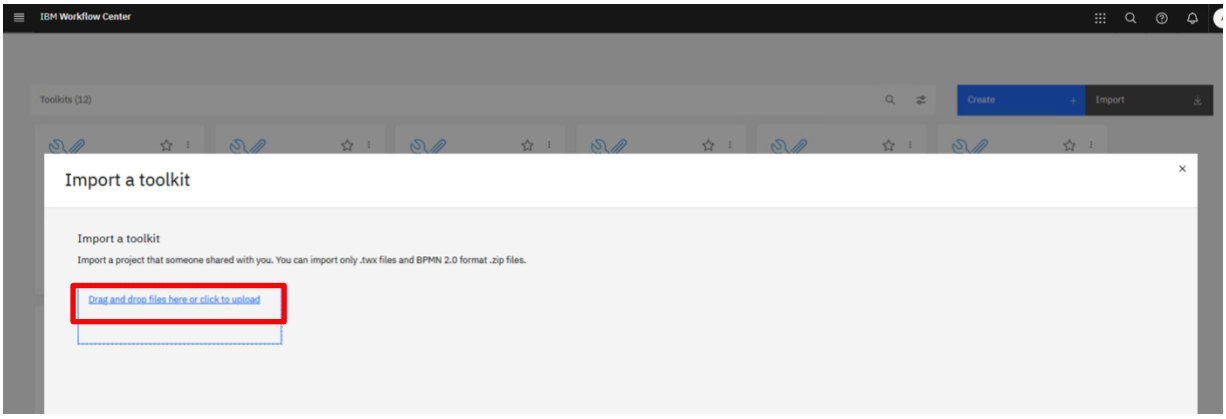
6. Enter **admin** as **username** and **admin** as **password**, then click **Log In** to log into Workflow Center. In case username or password are incorrect, please check the document mentioned in **step-6** at **page-9** to get correct user credential.

7. Click hamburg icon ☰ from the top-left corner in the IBM Workflow Center and select **Process apps→Toolkits**.

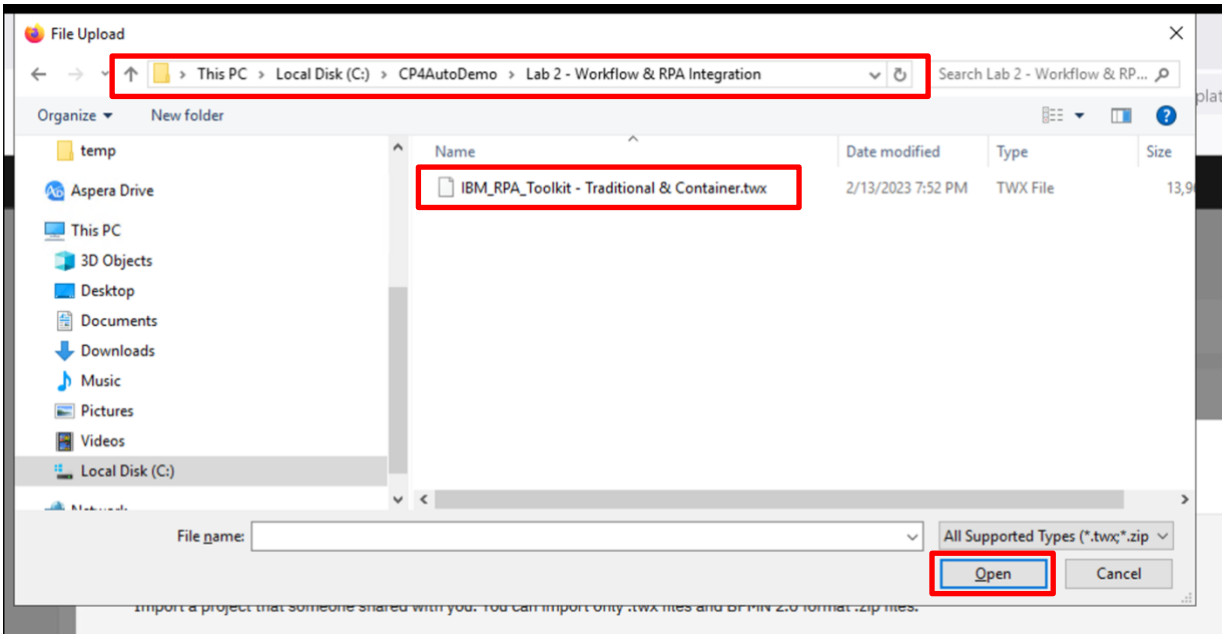
8. Click **Import**



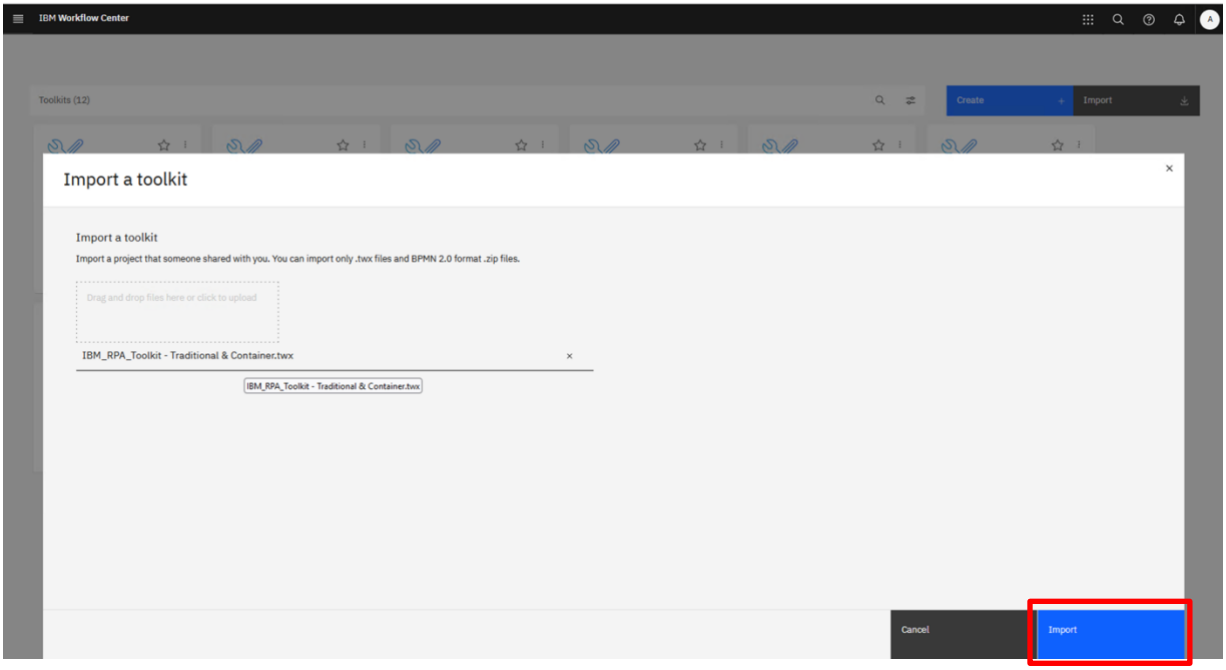
9. Click Drag and drops file here or click to upload



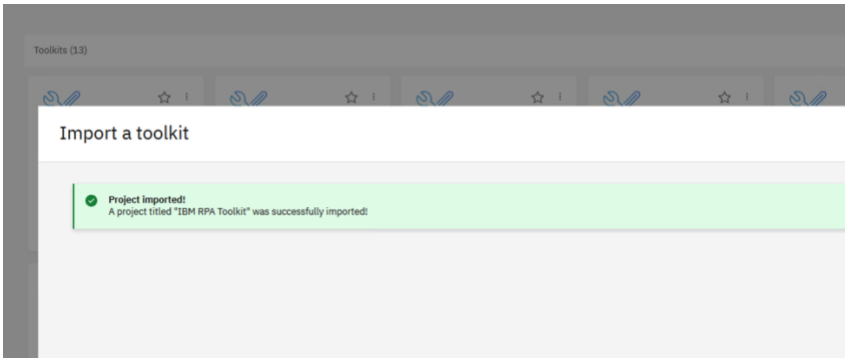
10. Select **IBM_RPA_Toolkit-Traditional & Container.twx** from **C:\CP4AutoDemo\Lab 2 – Workflow & RPA Integration** folder, click **Open** to import IBM RPA toolkit.



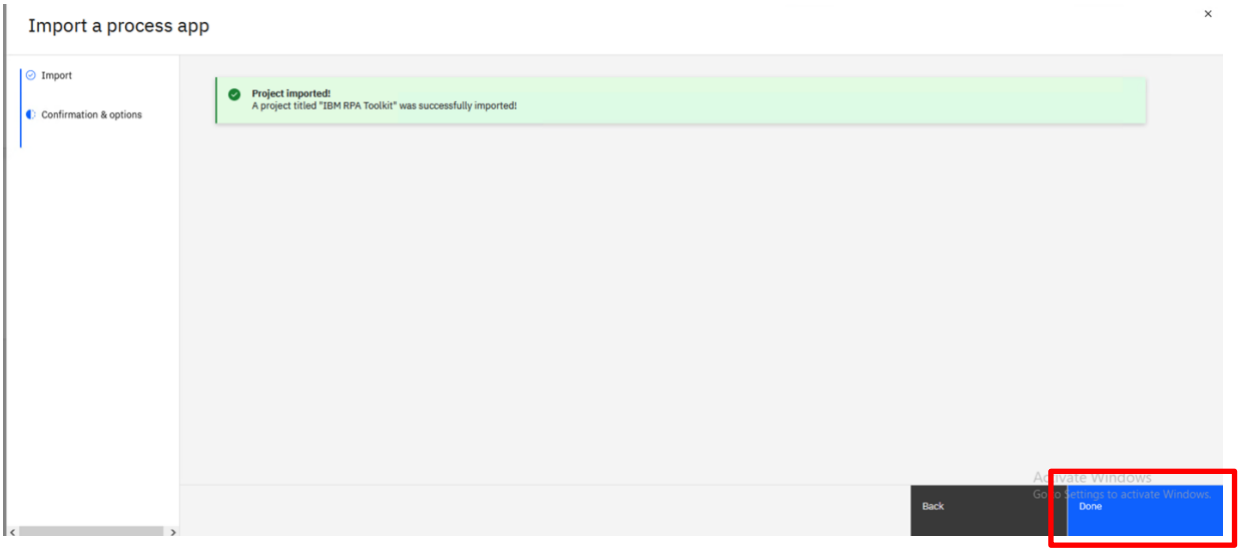
11. **Click Import**



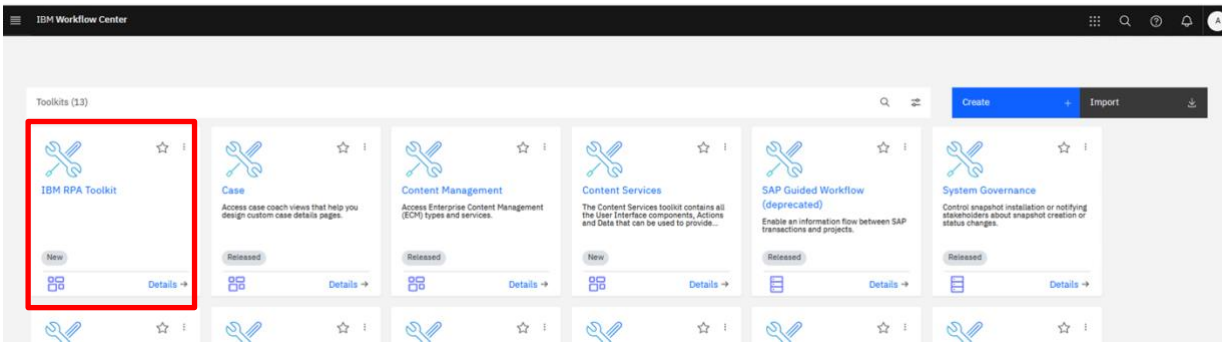
12. Wait for the message Project Imported



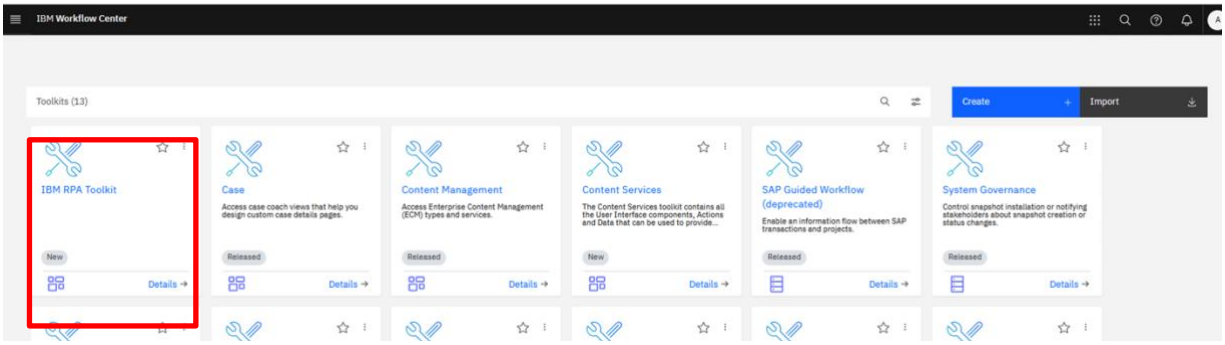
13. Click Done



14. Update the page to view the toolkit

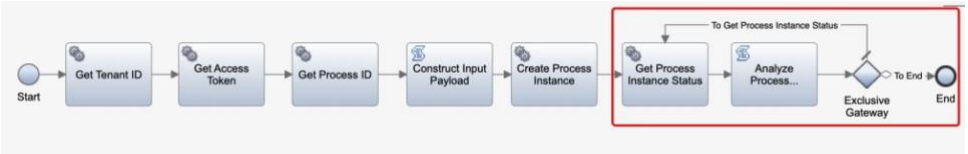


15. **IBM RPA Toolkit** will be imported into workflow center successfully. Click on the **IBM RPA Toolkit** icon to open the toolkit in Workflow Designer.



16. The **IBM RPA Toolkit** contains two types of services that will be used in this lab. One is a data model that assigns values to the bot script's input variables. It is a string that will hold the client onboarding information business object, which will be added to the Client Management System as a JSON string. The second is a set of service flows corresponding to the RPA server-side APIs to retrieve information, including RPA tenant ID, process ID, access token etc., and create a process instance to start the RPA bot.

17. This toolkit contains various service flows corresponding to RPA server-side APIs. IBM RPA supports both on-premise and RedHat OpenShift deployment, different deployment model has different authentication mechanism. This toolkit provides two set of implementation to support both depoyments distinguished by its name. If a service flow name contains **OCP**, it indicates that this service is applicable to OpenShift deployment only. If its name contains **OnPremise**, it indicates the serivce is applicable to OnPremise deployment only. If its name doesn't contain OCP or OnPremise, it indicates the service supports both deployments. You can refer to the API Reference for a detailed specification of each API. In this lab, RPA is installed on premise Windows server, **Get Tenant ID OnPremise**, **Get Access Token OnPremise**, **Get Process ID, Create Process Instance,** and **Get Process Instance Status** will be used**.** The typical Process to start bot via those APIs is shown below.



Once the bot starts executing after the RPA process instance is created, the workflow process needs to query the bot execution result periodically until its execution is finished, and it also gets the output from bot.

Familiarize yourself with those service flows by clicking **Services** from the left panel and selecting the corresponding service flow,

Notes: since RPA is installed on Windows server in this lab, we will only introduce those service flows applicable to OnPremise deployment.

1. **Get Tenant ID Onpremise**: This is implemented using RPA API Authentication APIs. It returns the tenant ID given the tenant name and user name.

   Check the input and output variables by clicking the **Variables** tab.

Input:

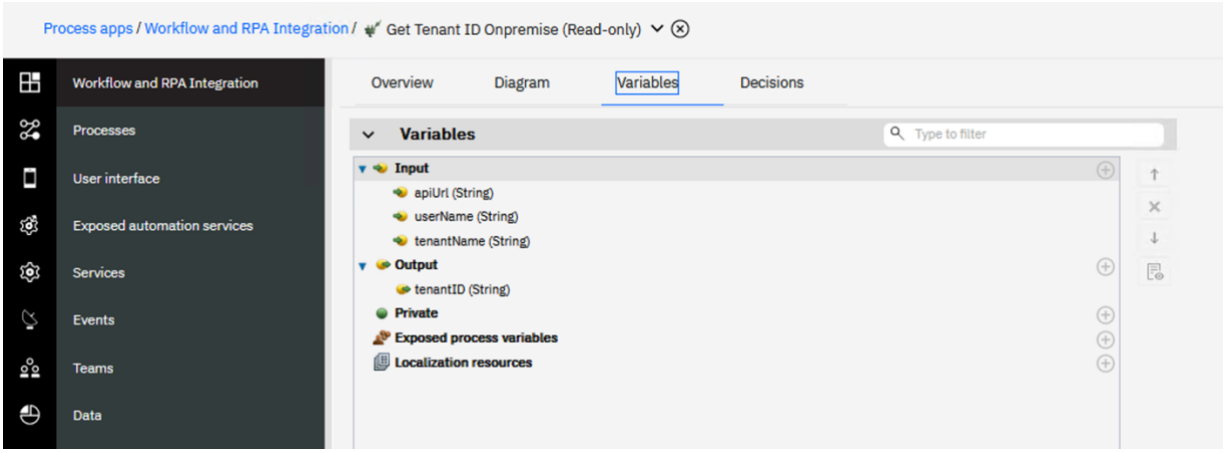
apiUrl: This is the API end-point. For SaaS deployment, you can get apiUrl by replacing "**app**" in the control center URL with "**api**". Taking South Central US region as an example, its control center URL is https://us1**app**.wdgautomation.com, so its API end-point is https://us1**api**.wdgautomation.com. For on-premise deployment, API port is set during the server installation, its API end point is **https://<your RPA Server>:<API Port>**.

userName: The user's email address to get the tenant ID that the user can access.

tenantName: This is the tenant name to retrieve its ID.

Output:

tenantID: Tenant ID if the given user belongs to the given tenant or "-1" if the user doesn't belong to the tenant.



2. **Get Access Token OnPremise**: This is implemented using RPA API Authentication APIs. It returns the authorization token, which is required in all following APIs.

   Check the input and output variables by clicking the **Variables** tab. Please note this API requires two additional input parameters, which are **grant_type** and **culture**. In this lab, we will use default grant_type which is **password**, and **en_US** for culture.

   Input:

   apiUrl: This is the API end-point.

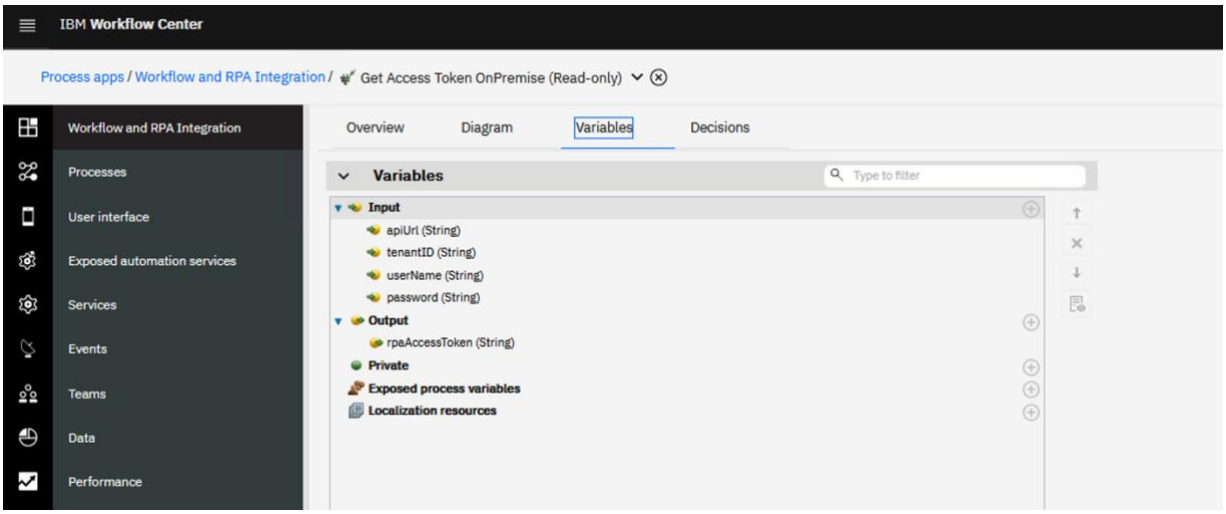
   tenantID : The tenant ID retrieved from **Get Tenant ID OnPremise**.

   userName: The user's email which has access to the tenant.

password: The user's password.

Output:

accessToken: The authorization token if user logs into tenant successfully or "-1" if login fails.



3. **Get Process ID**: This is implemented using RPA API [Process Management APIs.]
It returns the process ID for the specified Process defined in RPA control center.

Check the input and output variables by clicking the **Variables** tab.

Input:

apiUrl: This is the API end-point.

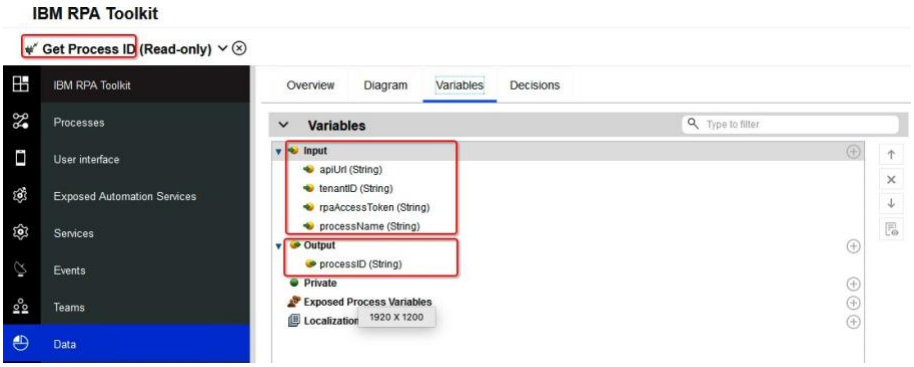
tenantID : The tenant ID retrieved from **Get Tenant ID OnPremise**.

rpaAccessToken: The authorization token retrieved from **Get Access Token OnPremise**.

processName: The process name as defined in the IBM RPA control center to retrieve its process ID.

Output:

processID: The processID of the specified Process defined in RPA control center or "-1" if the Process doesn't exist in the control center.



4. **Create Process Instance**: This is implemented using RPA API Process Management APIs. It creates a new process instance that will automatically trigger the bot execution if there is at least one available bot runner license.

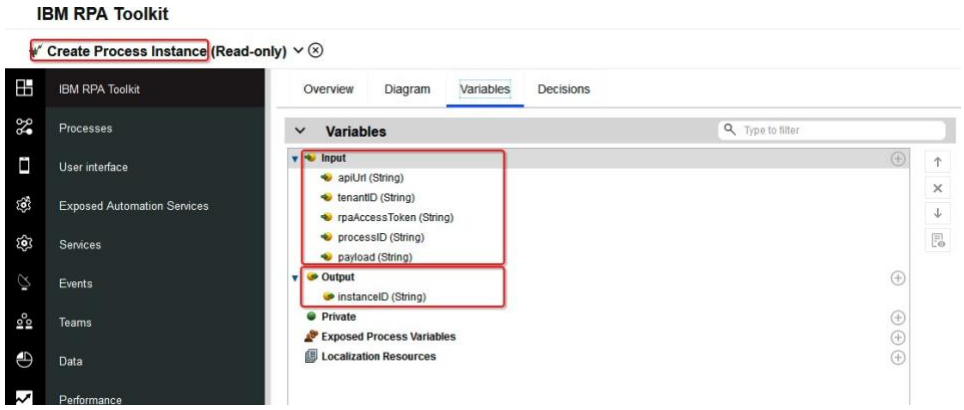
   Check the input and output variables by clicking the **Variables** tab.

   Input:

apiUrl:	This is the API end-point.
tenantID :	The tenant ID retrieved from **Get Tenant ID OnPremise**.
rpaAccessToken:	The authorization token retrieved from **Get Access Token OnPremise**.
processID:	The process ID retrieved from **Get Process ID**.
payload:	The input data passes to the script's input variables.

   Output:

instanceID:	The ID of the newly created process instance if a process instance was created successfully or "-1" if the creation fails.



5. **Get Process Instance Status**: This is implemented using RPA API Process Management APIs. It returns the process instance status.

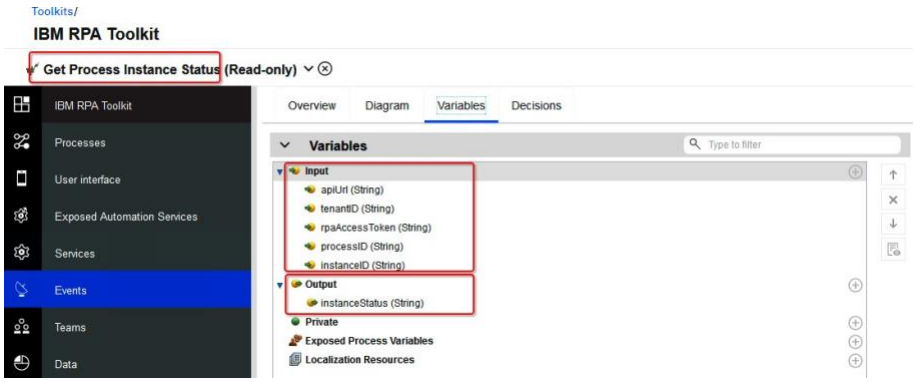
   Check the input and output variables by clicking the **Variables** tab.

   Input:

apiUrl:	This is the API end-point.
tenantID :	The tenant ID retrieved from **Get Tenant ID OnPremise**.
rpaAccessToken:	The authorization token retrieved from **Get Access Token OnPremise**.
processID:	The process ID retrieved from **Get Process ID**.
instanceID:	The instance ID returned from **Create Process Instance**.

Output:

instanceStatus: The process instance result as a JSON string. It contains three piece of information – **status**, **variables** and **outputs**.
status represents the status of the process instance which can be **new**, **pending**, **processing**, **done** or **failed**.
variables represents the process instance's input variables.
outputs represents the process instance's output variables.

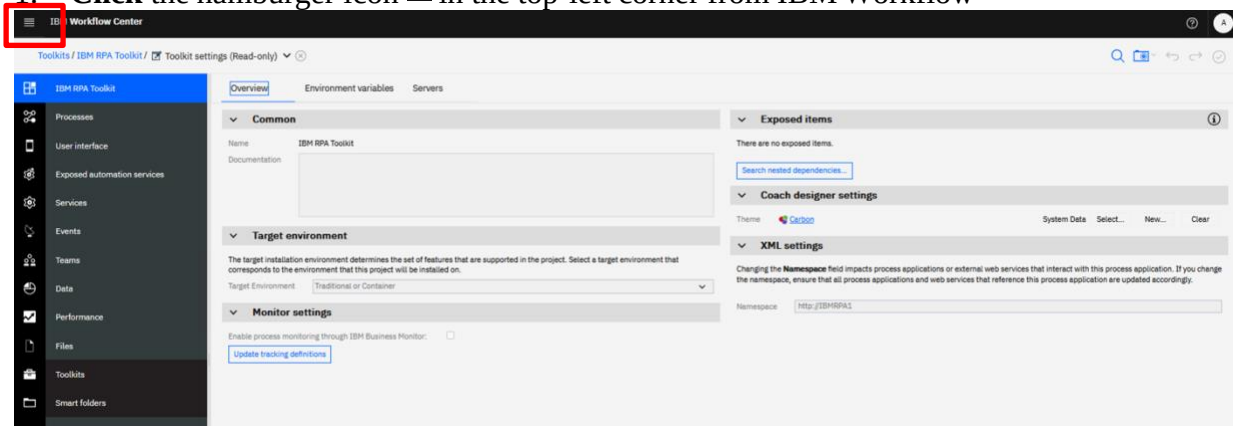


The toolkit also contains a few other service flows, including **Start RPA Bot OCP/OnPremise,** and **Query Bot Execution Status**. Since they are not used in this lab, we will not explain them one by one, you can explore them if you are interested.

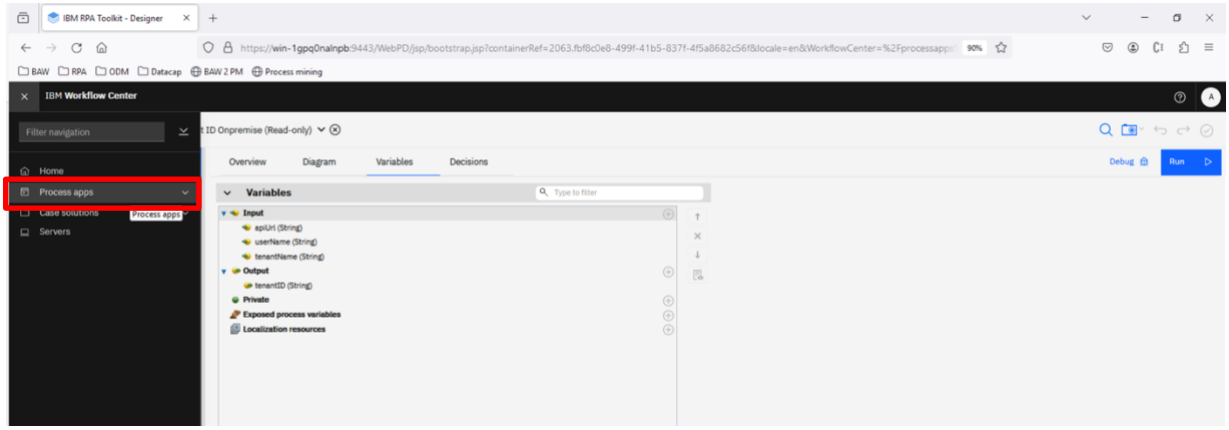
4.2.2 Develop a Workflow Process to start an RPA Bot

The entire end-to-end client onboarding solution involves many components: automation application, content management, automation decision service, and mobile capture. You can refer to the other labs to learn how to develop other parts of the client onboarding solution. The goal here is to showcase how activity in a Workflow process can call an RPA bot to add the client onboarding information to the backend applications. Therefore we will create a simplified process to illustrate how to call the RPA bot using the services introduced above.

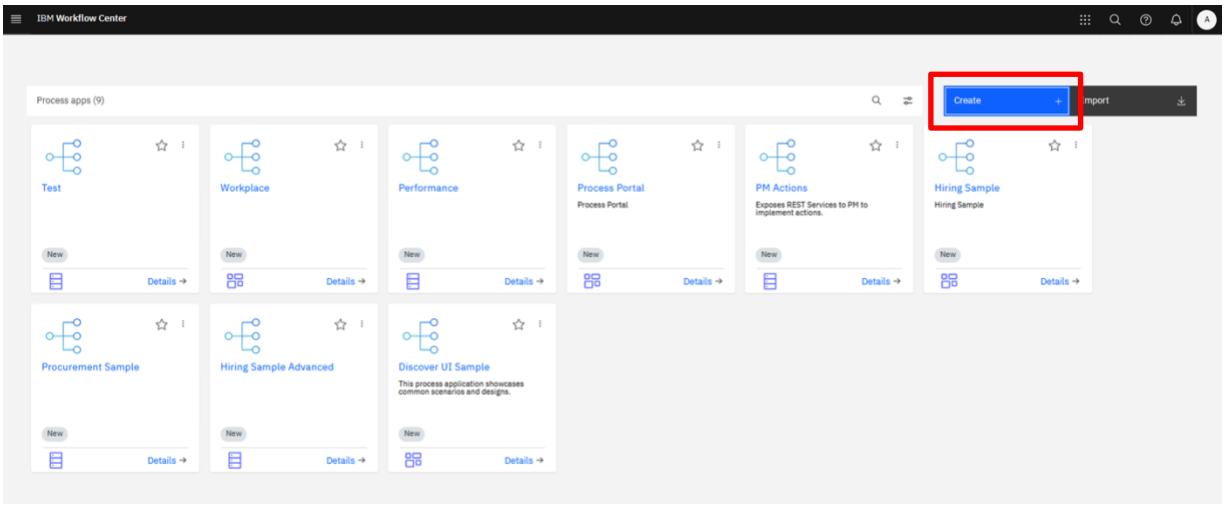
1. **Click** the hamburger icon ☰ in the top-left corner from IBM Workflow



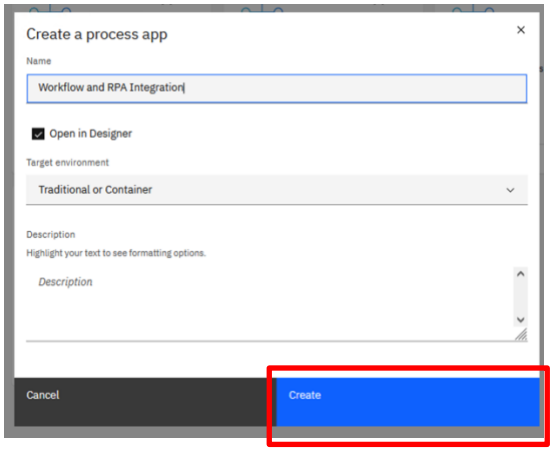
2. Clique the **Process Apps**.



3. Click **Create.**

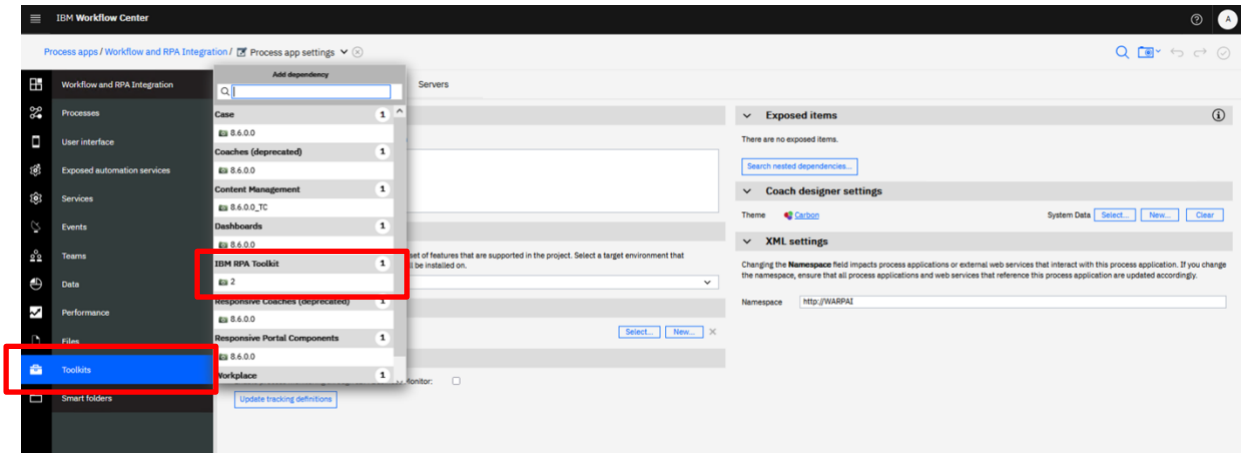


4. Enter a **name** for your process application, for example – **Workflow and RPA Integration**. Leave all others unchanged. Once done, click **Create**

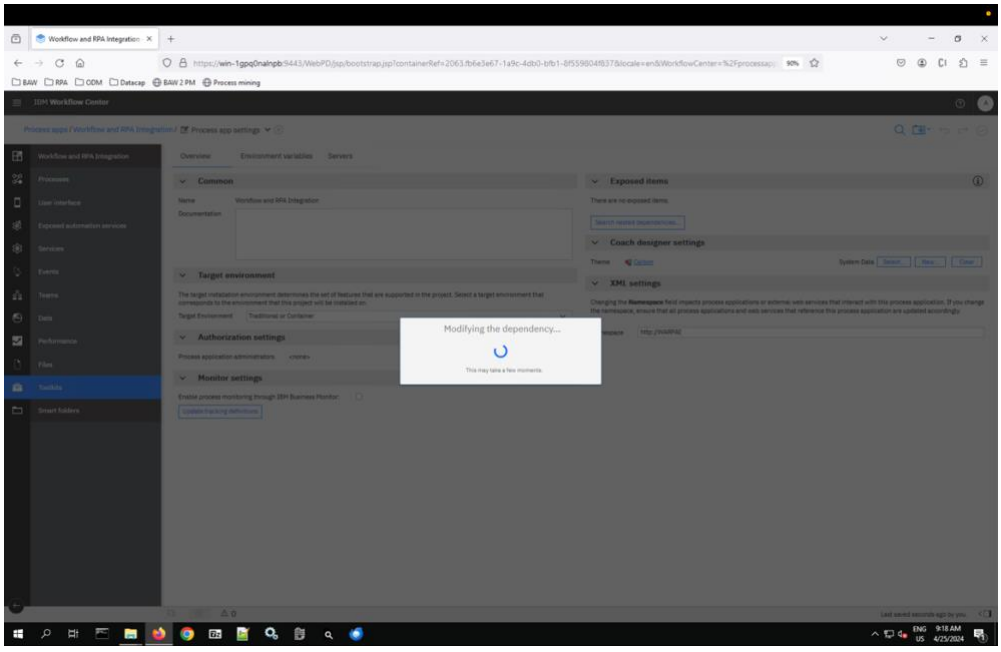


5. To use the data model and external services from the **IBM RPA toolkit**, the toolkit needs to be added as a dependency. Click the  icon on the right next to the **Toolkits** label. Then click on the latest version of the IBM RPA Toolkit to add it as a dependency.

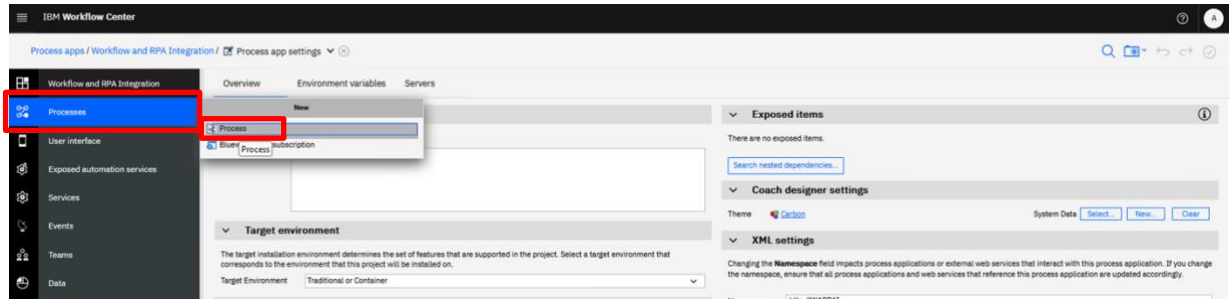
Notes: The version number may be different. Please always select the latest version/version with the highest version number.



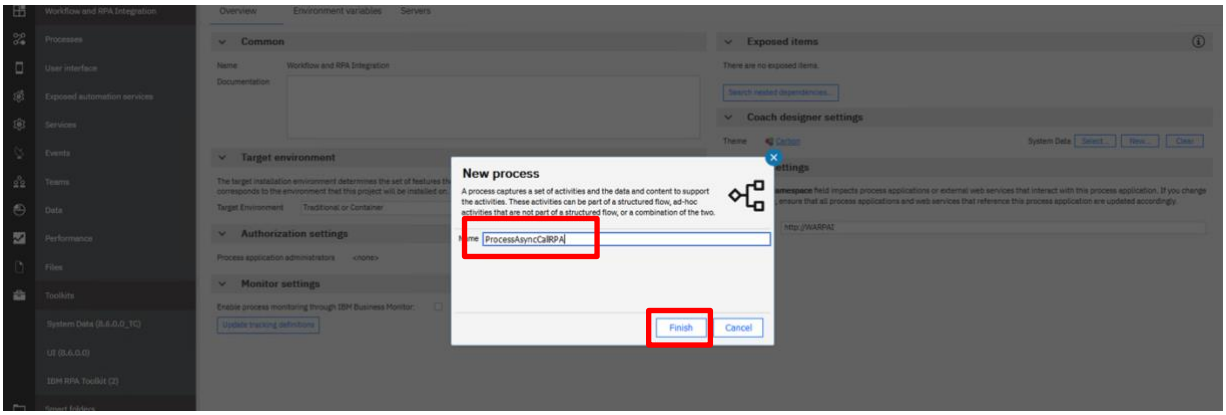
6. After clicking, please wait while modifying the dependency."



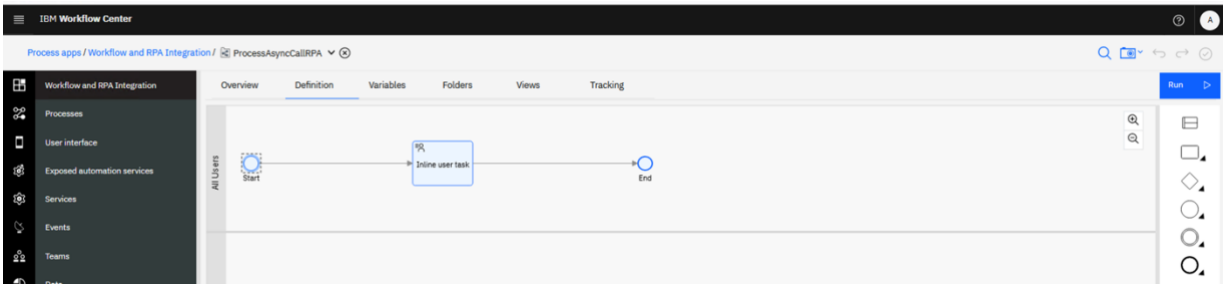
7. Click the ⊕ icon on the right of the **Processes** label and then click **Process**.



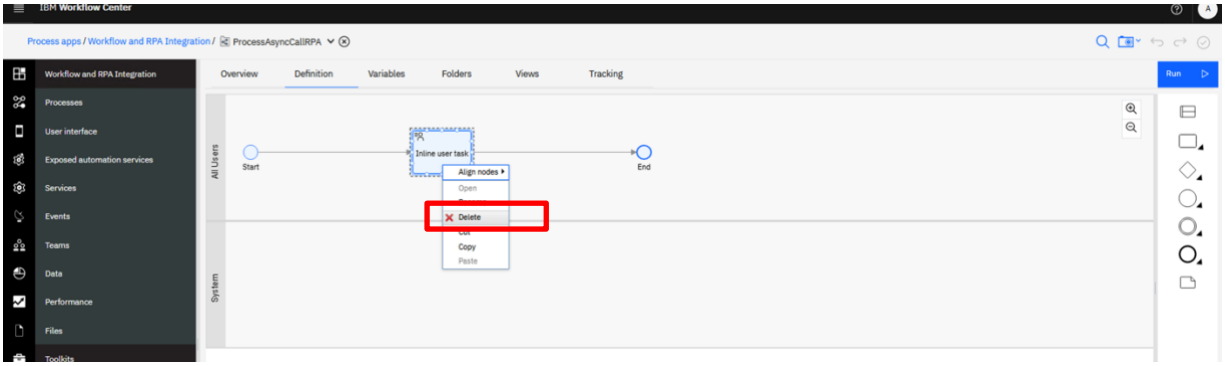
8. Enter a name for the new Process, for example – ProcessAsyncCallRPA, then click **Finish.**



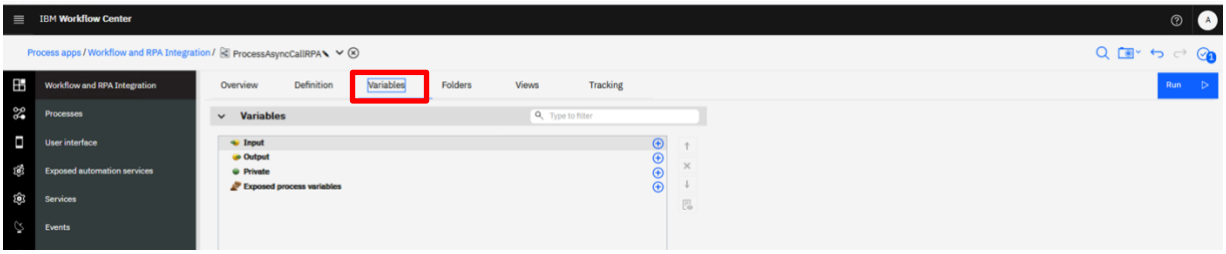
9. The newly created ProcessAsyncCallRPA process is opened in Workflow Designer. It initially contains one inline user task. We will change its implementation to start the RPA bot through the service flows provided in the IBM RPA Toolkit.



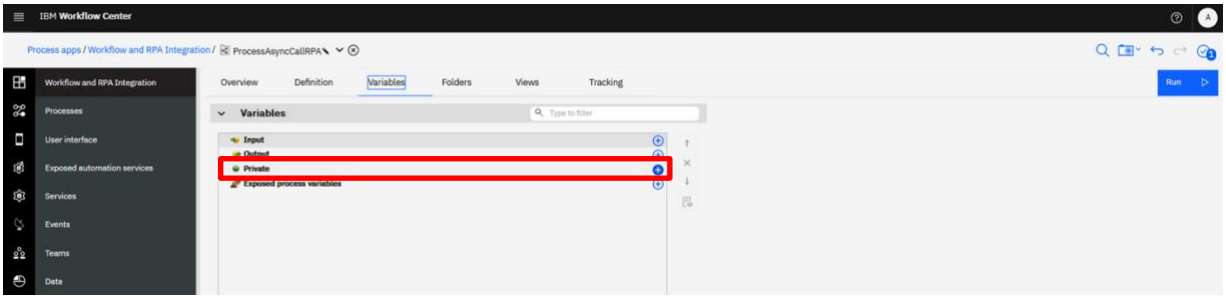
10. Right-click the Inline User Task and select **Delete** to remove it.



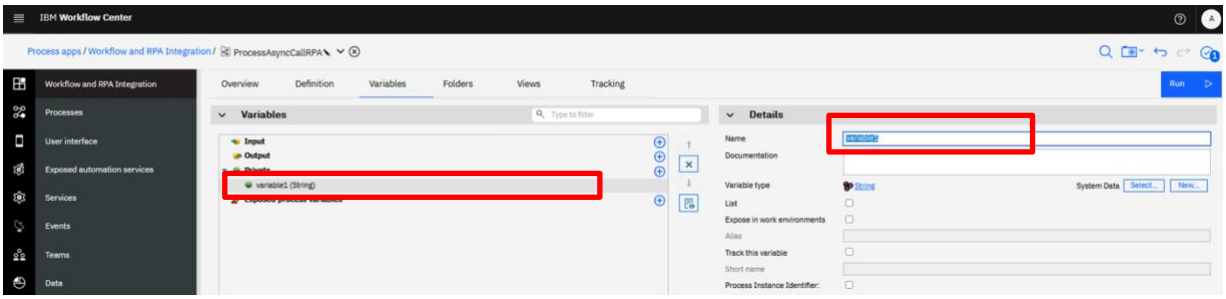
11. Click on the **Variables** tab to switch to the Variables view.



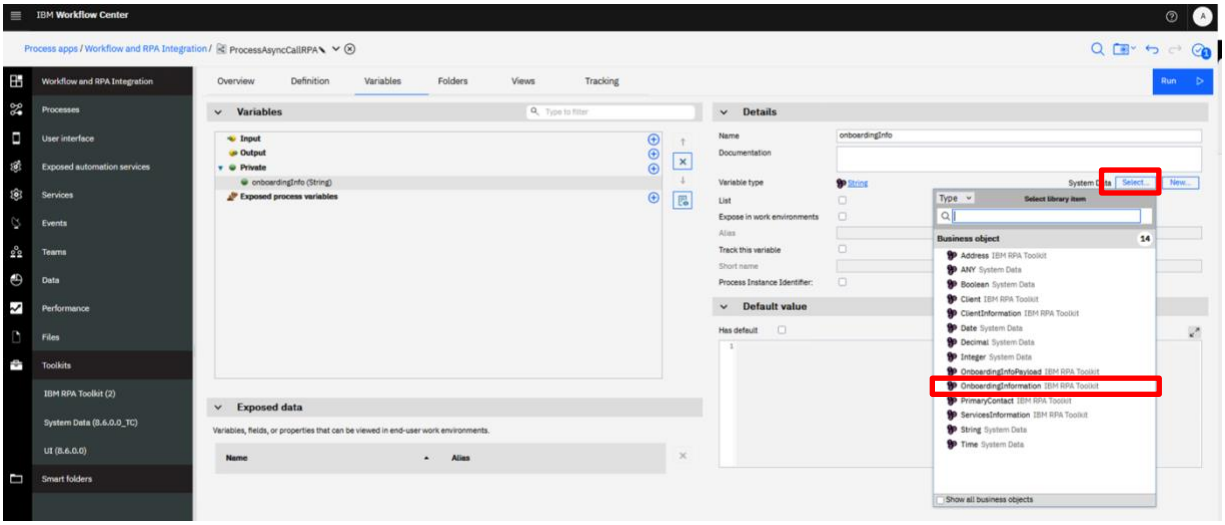
12. Click the ⊕ icon to add a **Private** variable



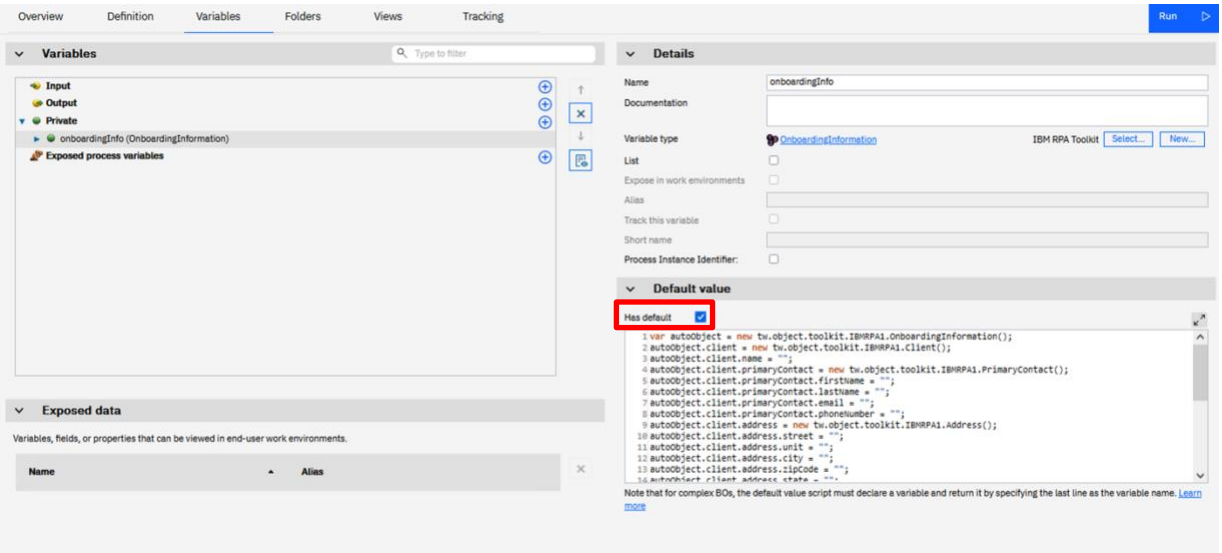
13. Click the variable Change its name to **onboardingInfo**



14. Click in the **select button** and find to OnboardingInformation (IBM RPA Toolkit) and click to select



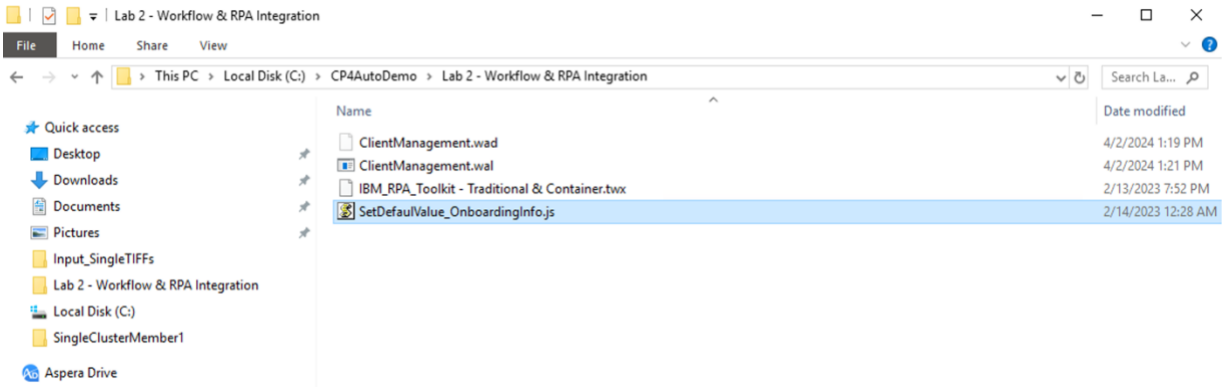
15. Click on Default Value



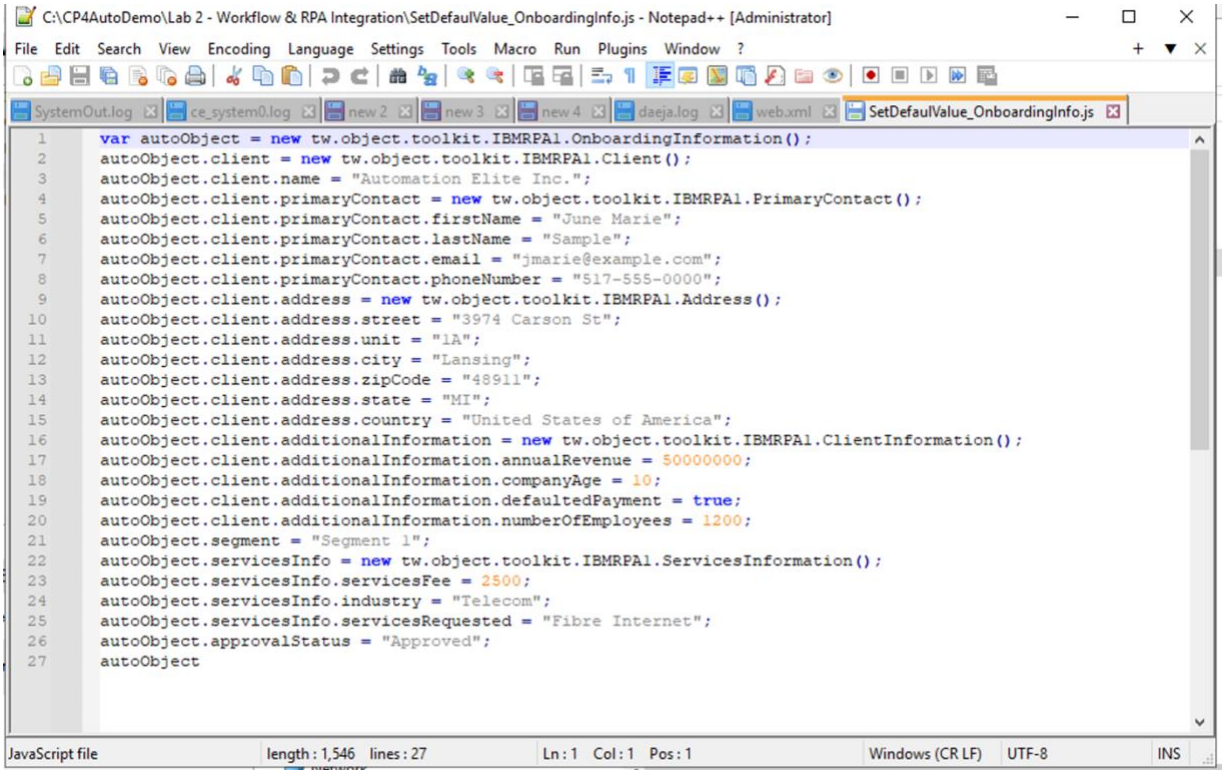
The auto-generated JavaScript constructs the business object structure and sets the default values to blank.

16. Change the values of default value

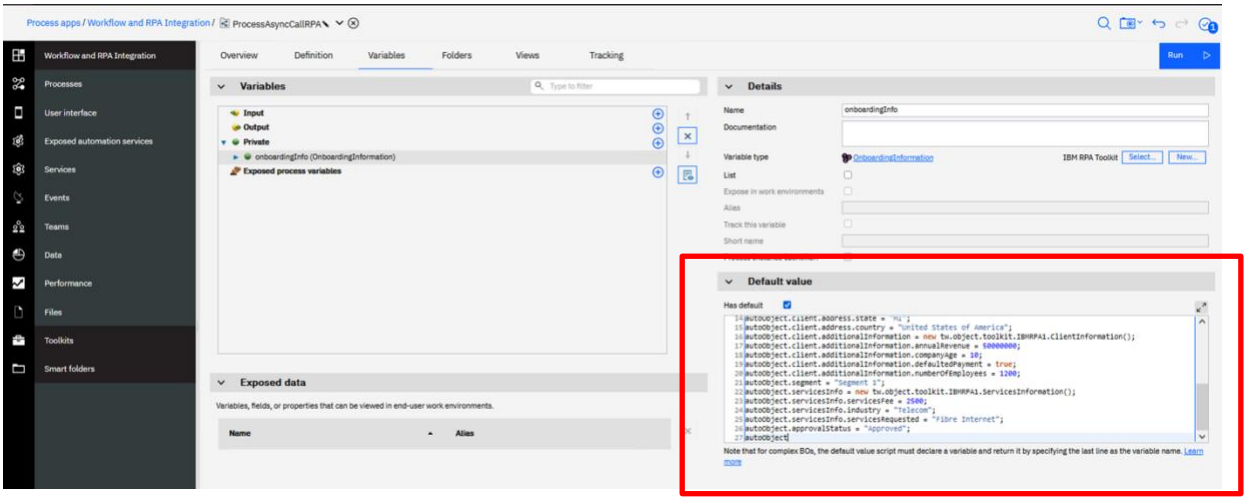
We need to change its default value. Replace the auto- generated JavaScript code with the code from **SetDefaulValue_OnboardingInfo.js** located in **C:\CP4AutoDemo\Lab 2 – Workflow & RPA Integration** folder.



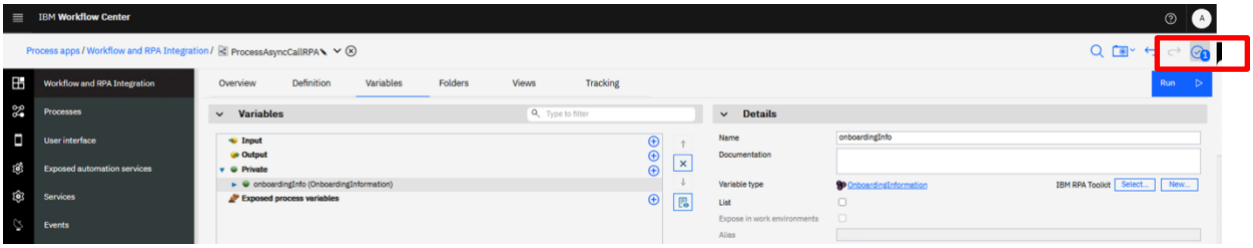
17. Edit with Notepad++ and select value



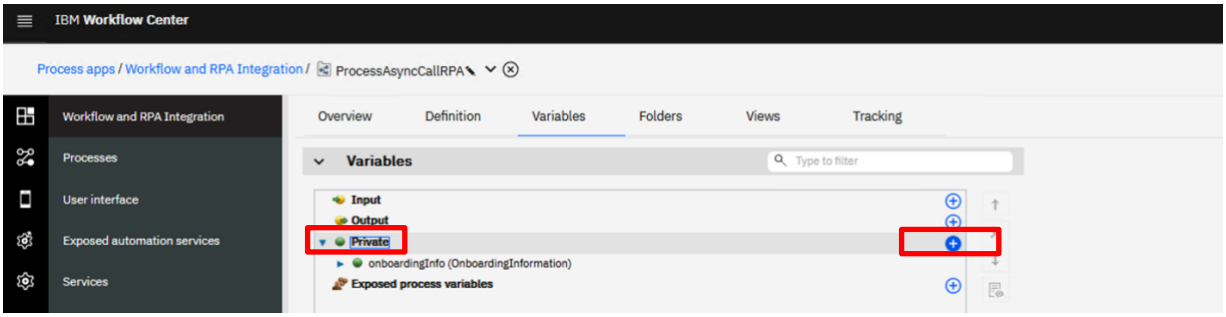
18. Paste the copied content into the Default value



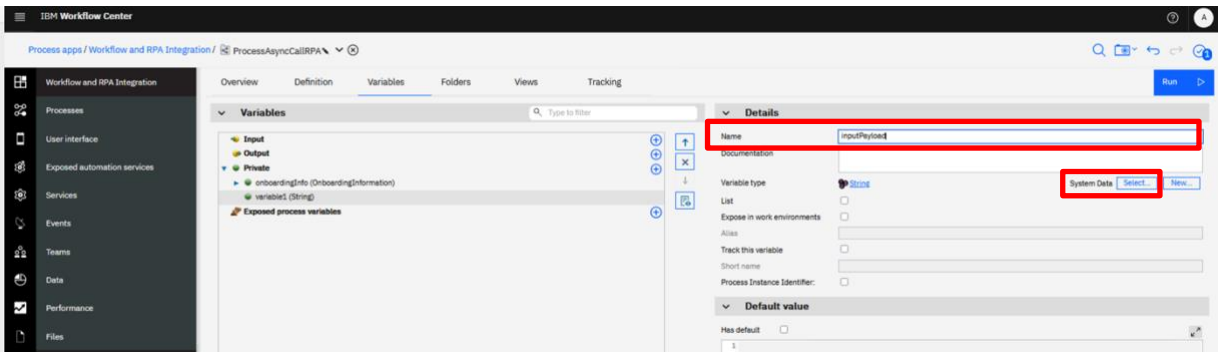
19. Clique icon  right side to save.



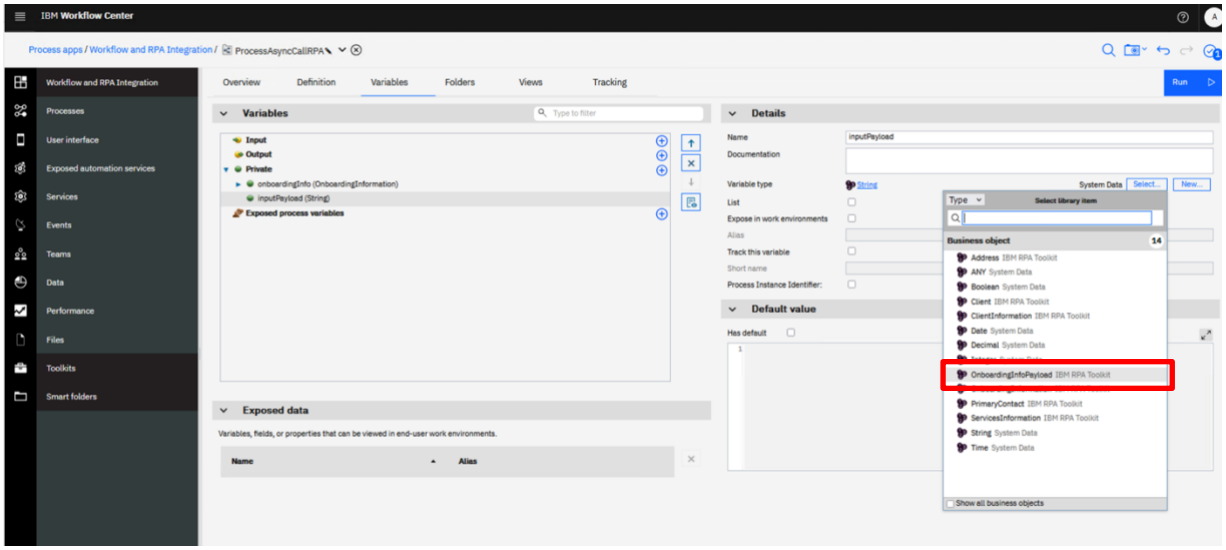
20. Click on the Private label and click the ⊕ icon twice next to add additional **private**



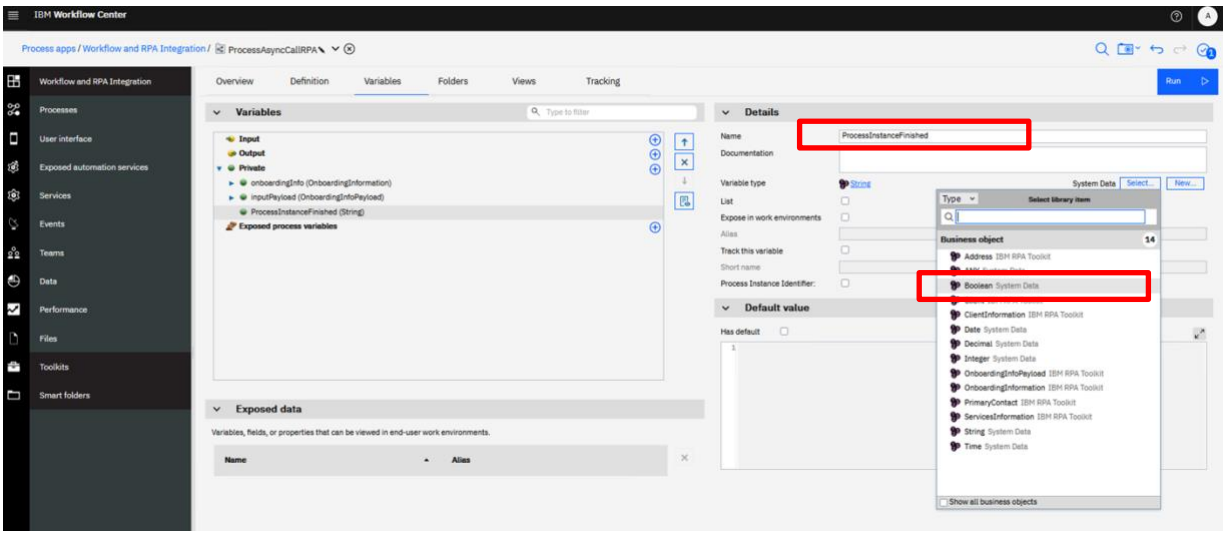
21. Change the name to **inputPayload** and click select



22. Select OnboardingInfoPayload from IBM RPA Toolkit as the Type.

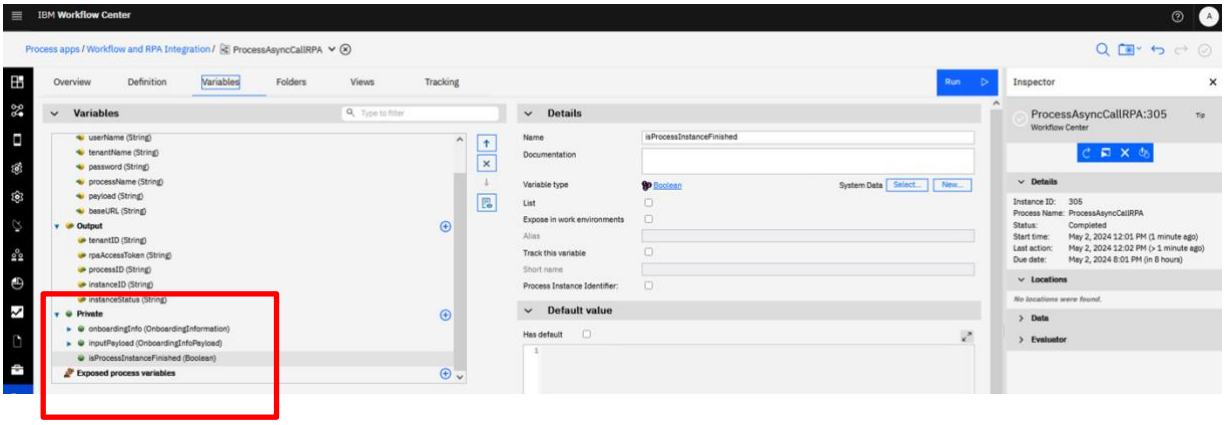


23. Click on the Private label and create **isProcessInstanceFinished** variable and select Boolean as the type



24. In this step we will have 3 variables created

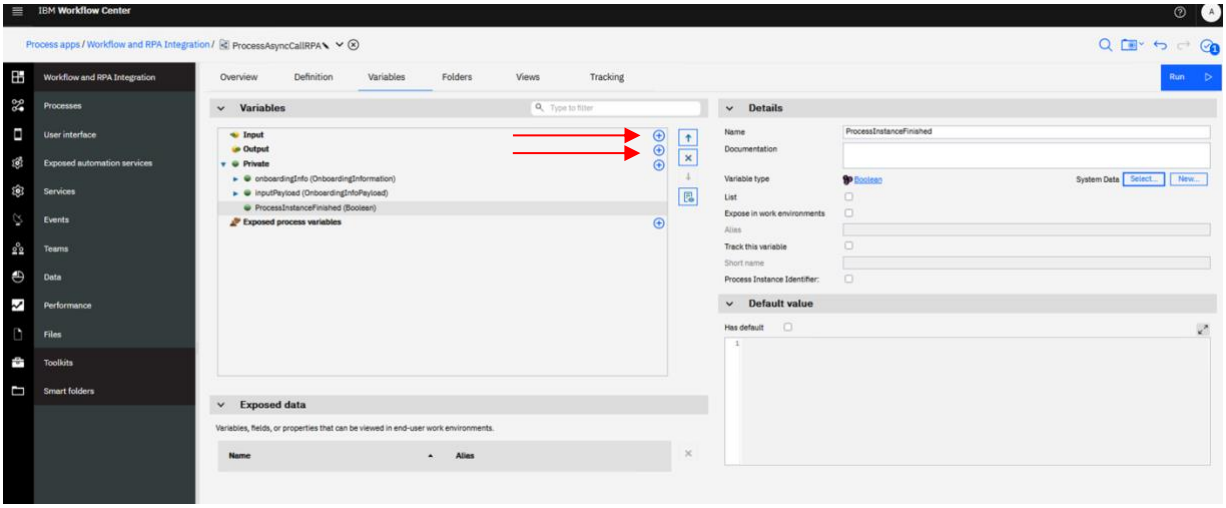
- **onboardingInfo**
- **inputPayload**
- **isProcessInstanceFinished**



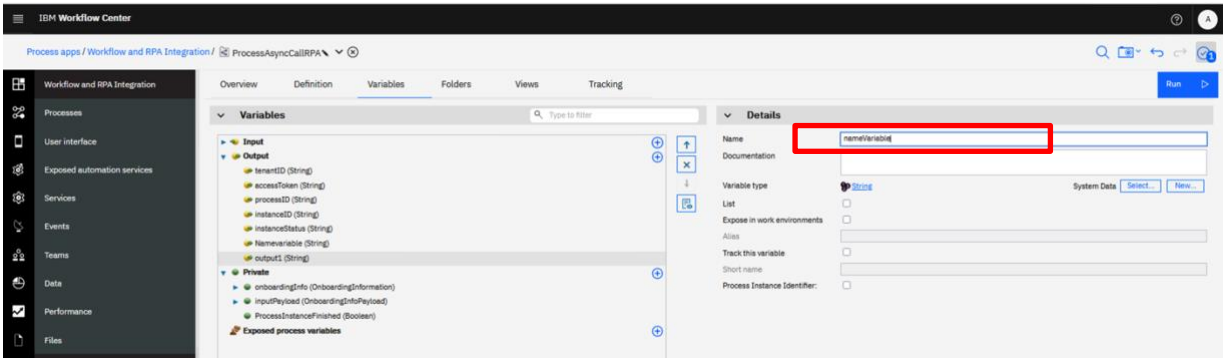
25. Create all variable input/ouput bellow: you should create a total of 7 input variables of type String and 5 output variables of type String. Later on, we will define some values for these variables.

Variable Name	Input/output	Type
apiUrl	input	String
userName	input	String
tenantName	input	String
tenantID	output	String
password	input	String
rpaAccessToken	output	String
processName	input	String
processID	output	String
payload	Input	String
instanceID	output	String
instanceStatus	output	String
baseURL	input	String

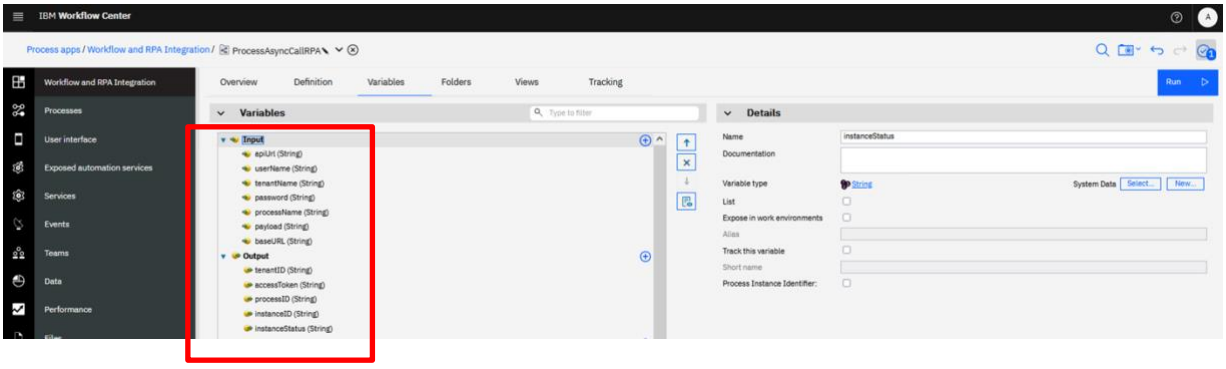
26. Click on input or Output



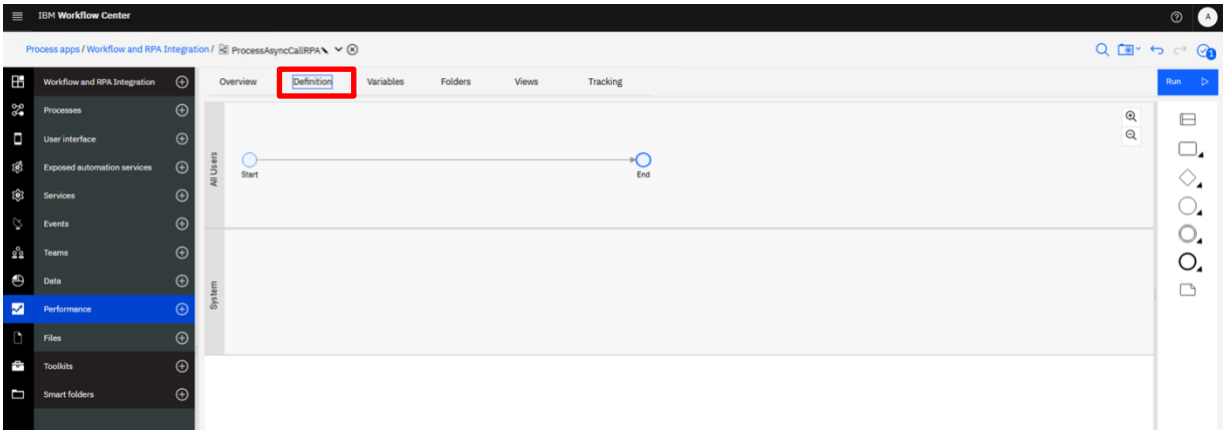
27. Click on the Name field and set a variable name, then click on save". 



28. The final result will be this.

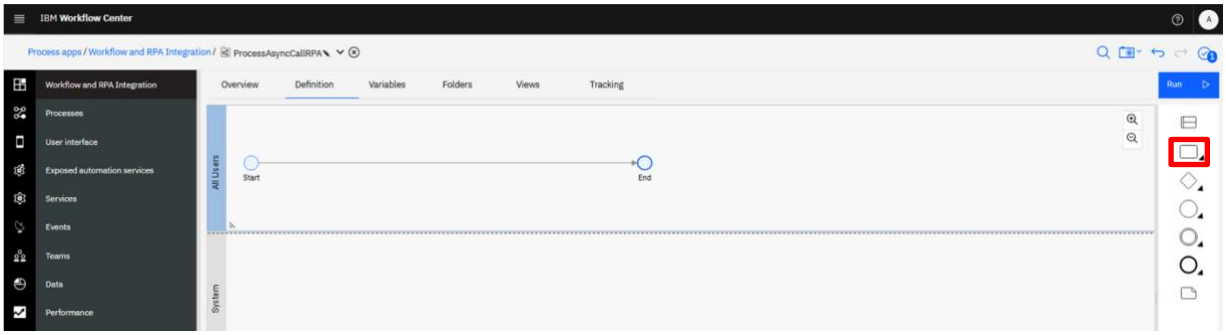


29. Click the **Definition** tab to switch back to the process diagram view.

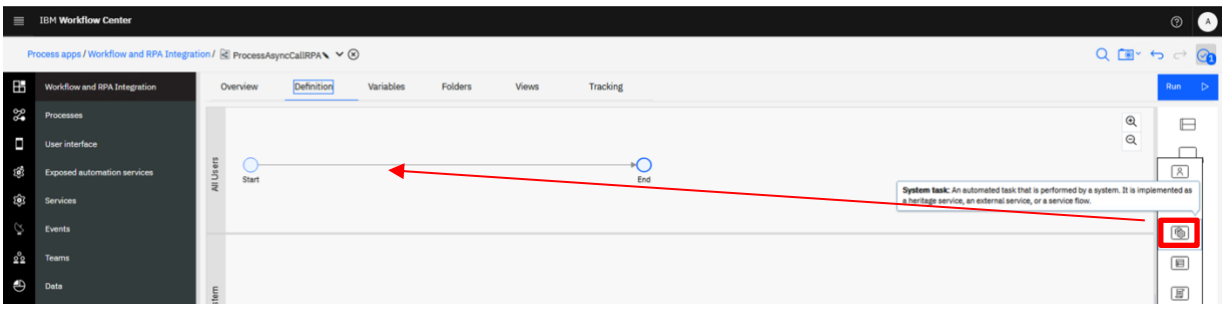


30. Drag a **System Task** activity from the right-hand palette and drop it onto the line between the **Start** and **End** activity.

    22.1 – Click  on the task option

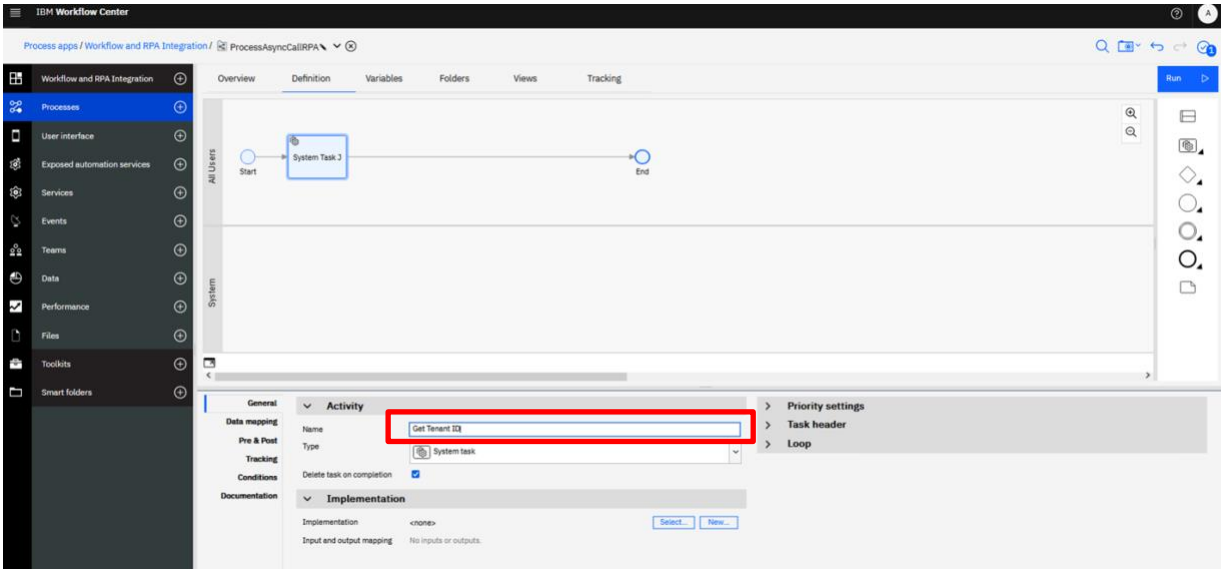


    22.2 Drag a System Task

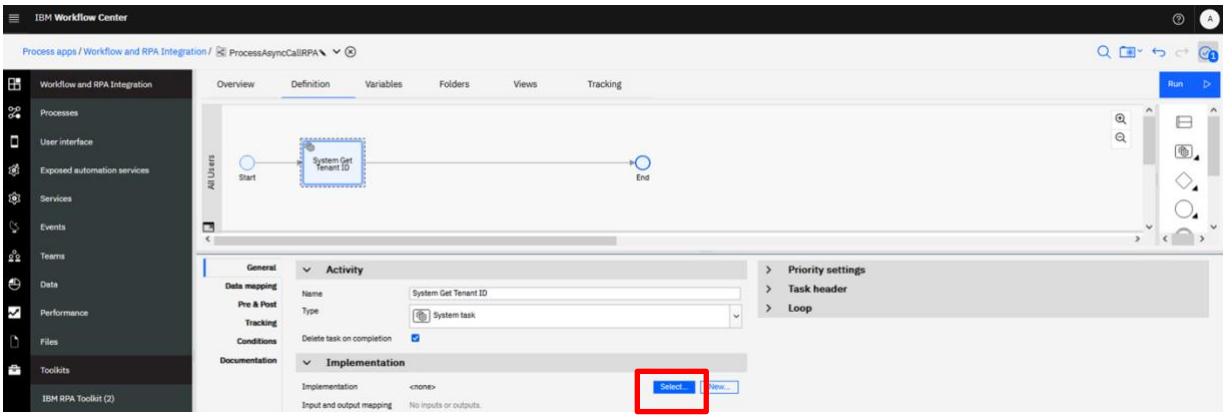


31. Configure the **System Task** activity as below:

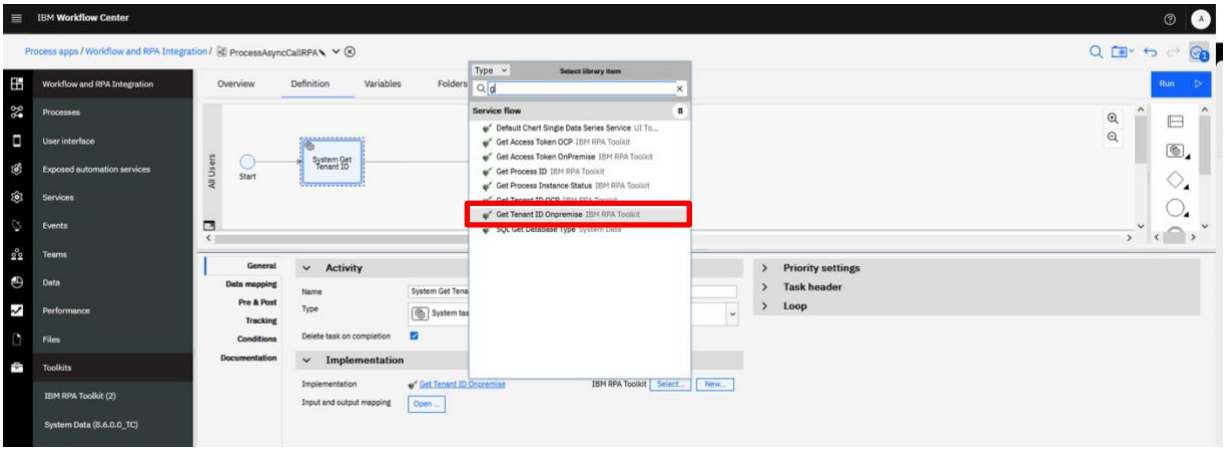
26.1-Select task and change the name to **Get Tenant ID**



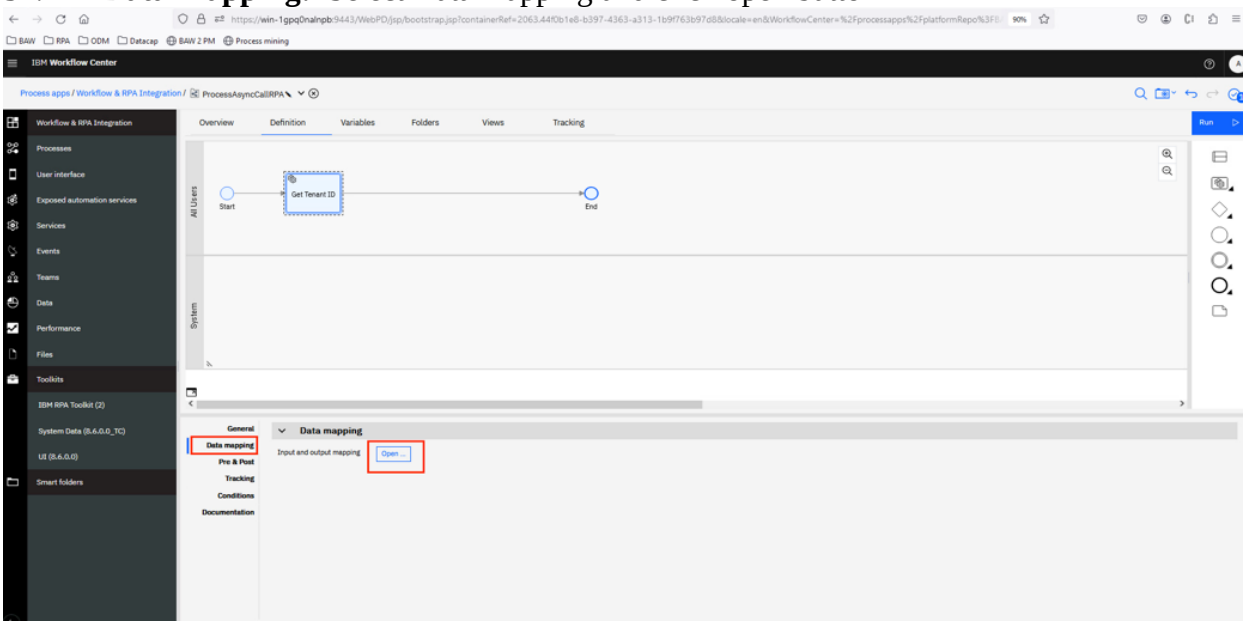
32. **Implemention:** Click the **Select** button and select **GET Tenant ID OnPremisse**



33. **Implemention**: select GET Tenant ID Onpremisse

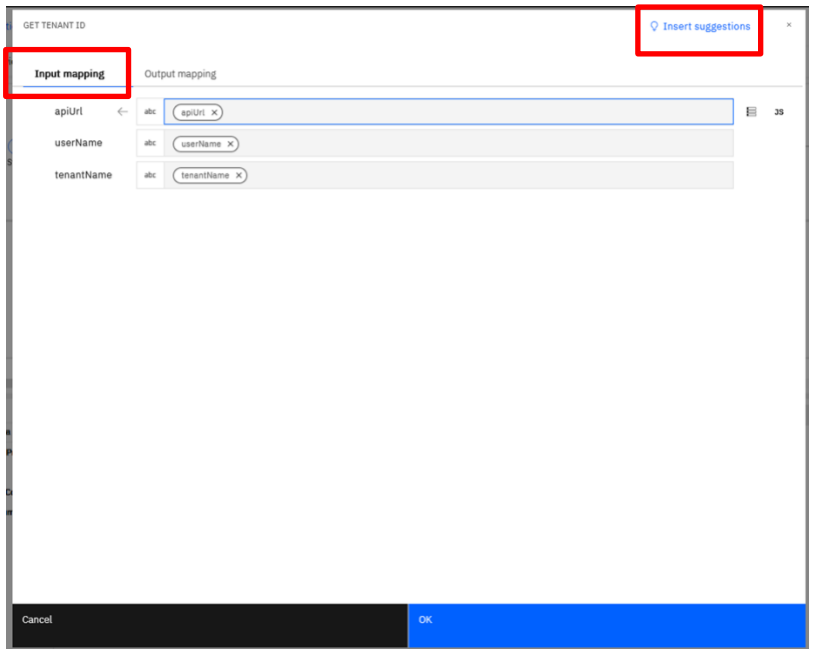


34. **Data Mapping:** Select Data mapping and click open button

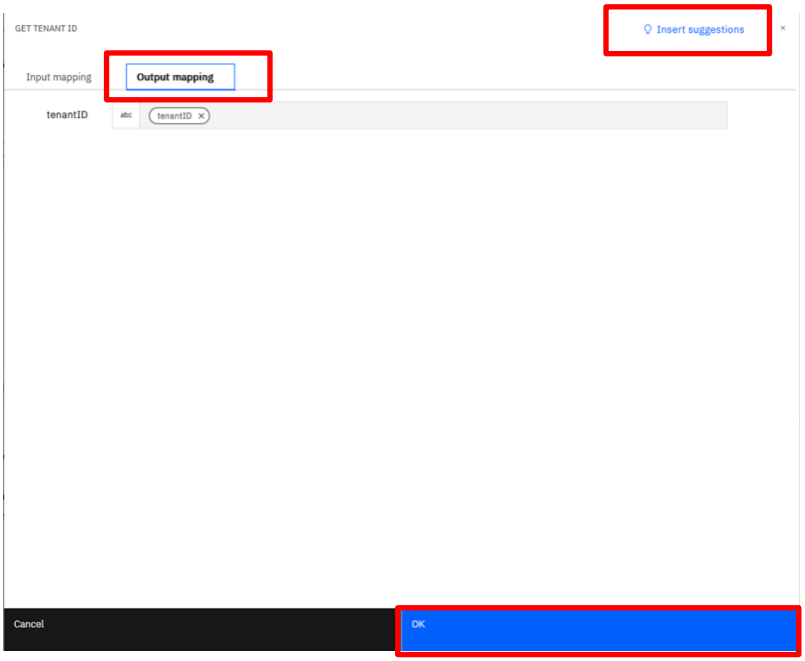


35. Click on Insert Suggestions and note that the variables are automatically binded. Repeat this step for the input and output variables.

Input mapping



Output mapping

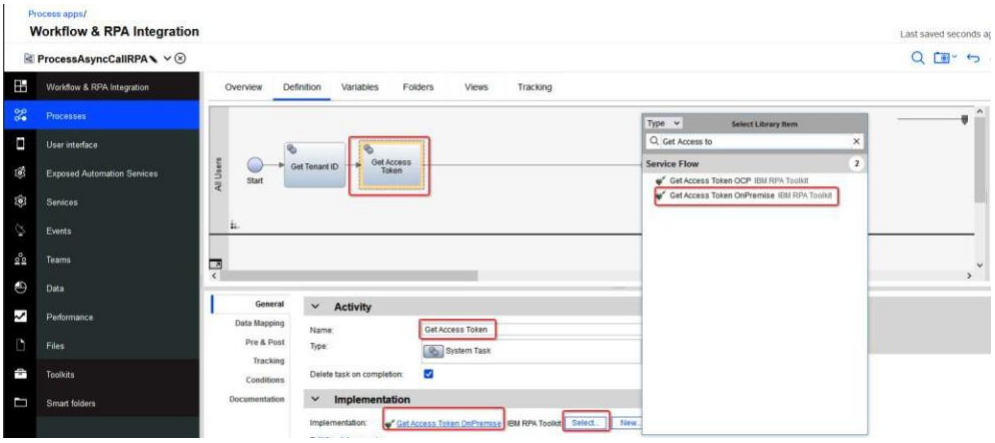


36. Follow the same approach to add a second system task and configure it as below:

    Name: Change its name to **Get Access Token**

    Implementation: Select the **Get Access Token OnPremise** service flow as its implementation. (Step 24)

    Data Mapping: Create and bind all input and output variables.

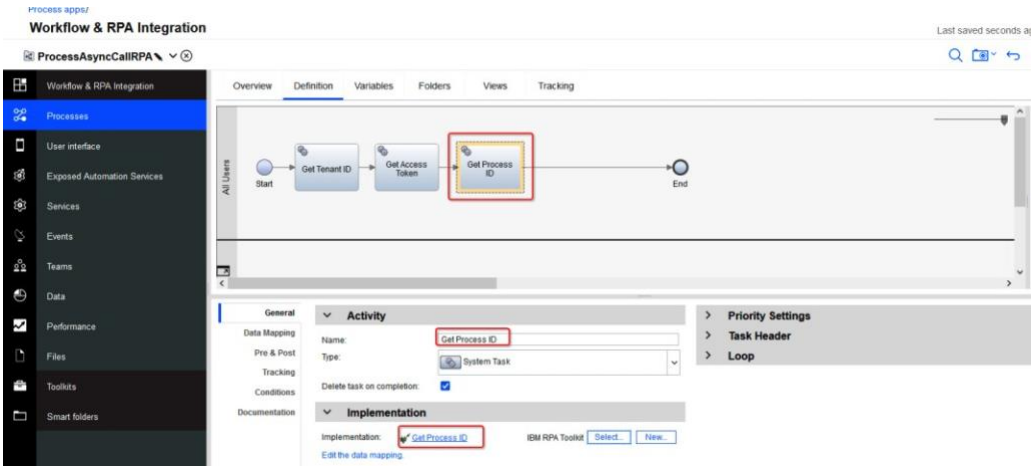


37. Follow the same approach to add a third system task and configure it as below:

    Name: Change its name to **Get Process ID**

    Implementation: Select the **Get Process ID** service flow as its implementation.

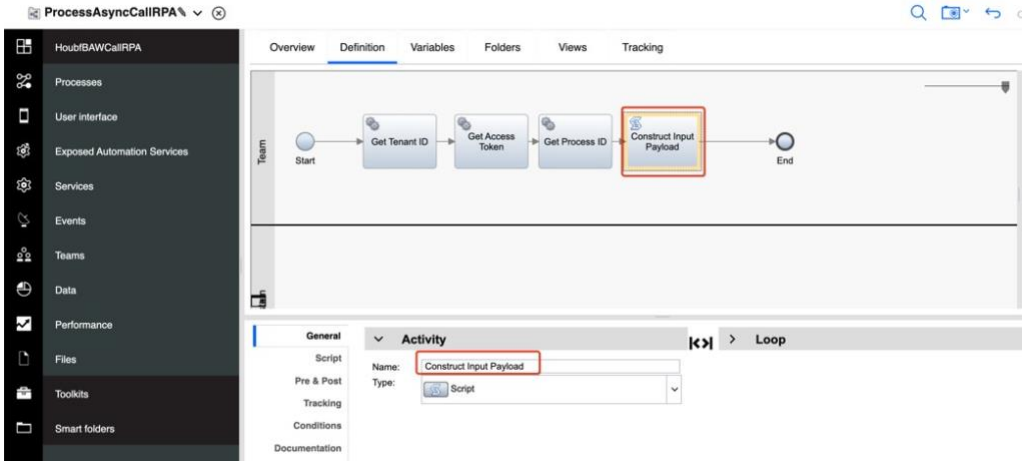
    Data Mapping: Create and bind all input and output variables.



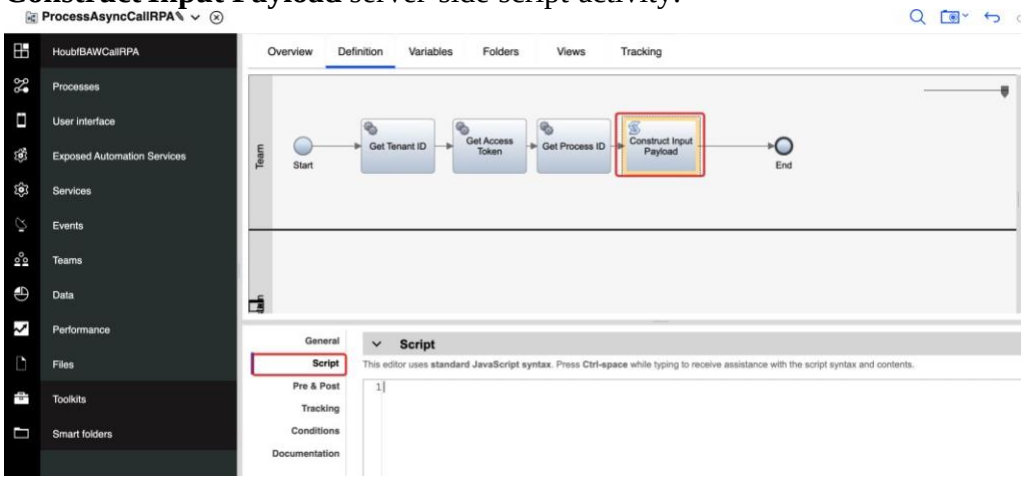
38. Before we can call the **Create Process Instance** API to create a new process instance on the RPA server to execute the bot, we need to construct the bot input payload. The client onboarding information is stored in the variable **onboardingInfo**, a business object. We need to convert it to a JSON string and pass that to the bot as a payload. Drag a **Server Script** activity and drop it onto the line

between Get Process ID and End activity as below.

39. Change the name of the Server-Side script activity to **Construct Input Payload**.

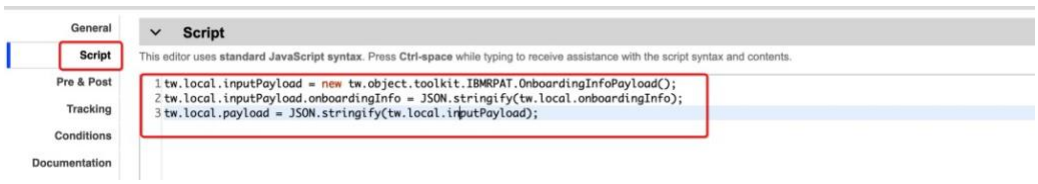


40. Click the **Definition tab** to switch back to the process diagram definition. Select the **Construct Input Payload** server-side script activity.



41. Click the **Script** tab of the activity to switch to the Script view, copy and paste below JavaScript snippet into the Script editor. This creates an instance of the OnboardingInfoPayload business object from the RPA integration toolkit, sets the JSON representation of the onboardingInfo variable in the instance, and assigns the JSON representation to the payload variable.

```
tw.local.inputPayload = new tw.object.toolkit.IBMRPA1.OnboardingInfoPayload();
tw.local.inputPayload.onboardingInfo = JSON.stringify(tw.local.onboardingInfo);
tw.local.payload = JSON.stringify(tw.local.inputPayload);
```

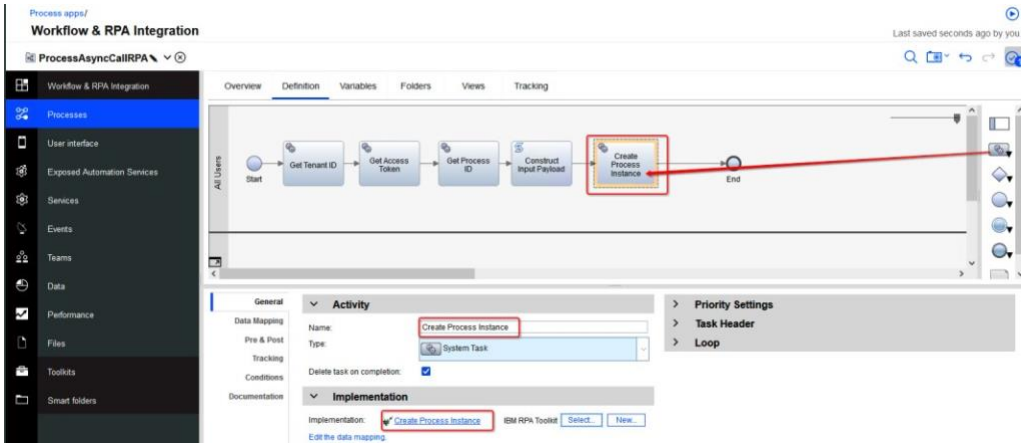


42. Drag a **System Task** activity and drop it onto the line between the **Construct Input Payload** and **End** activities, and configure it as below:

    Name: Change its name to **Create Process Instance**.

    Implementation: Select the **Create Process Instance** service flow

    Data Mapping: Create and bind all input and output variables.

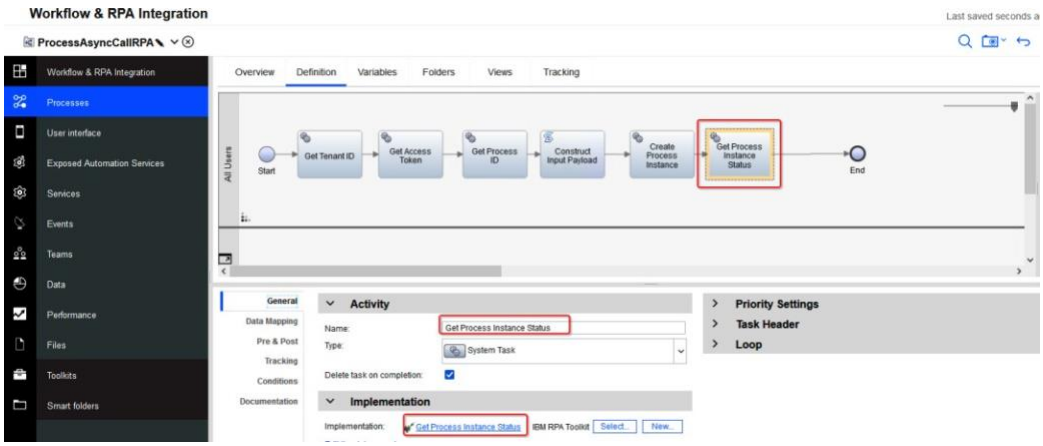


43. After the process instance has been created, it needs to check if it has been processed or not periodically. This can be achieved using the **Get Process Instance Status** API. Add another **System Task** activity to query process status, and configure it as below:

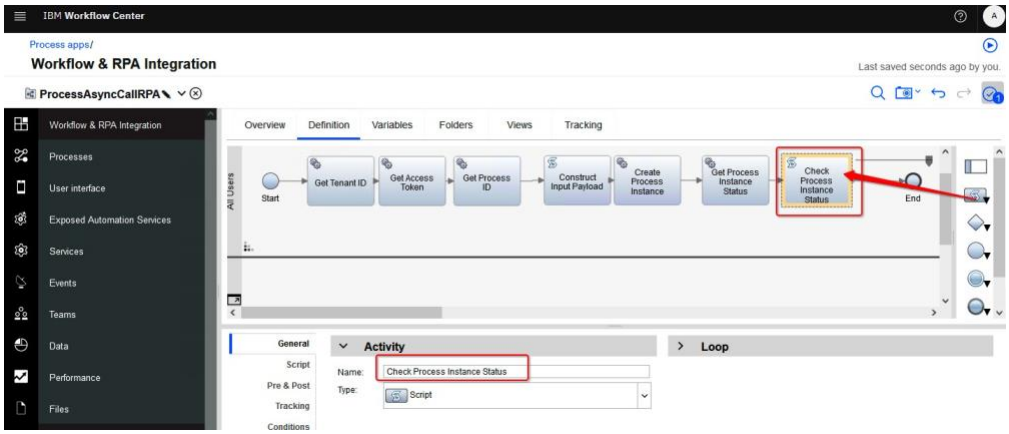
    Name: Change its name to **Get Process Instance Status**.

    Implementation: Select the **Get Process Instance Status** service flow.

    Data Mapping: Create and bind all input and output.

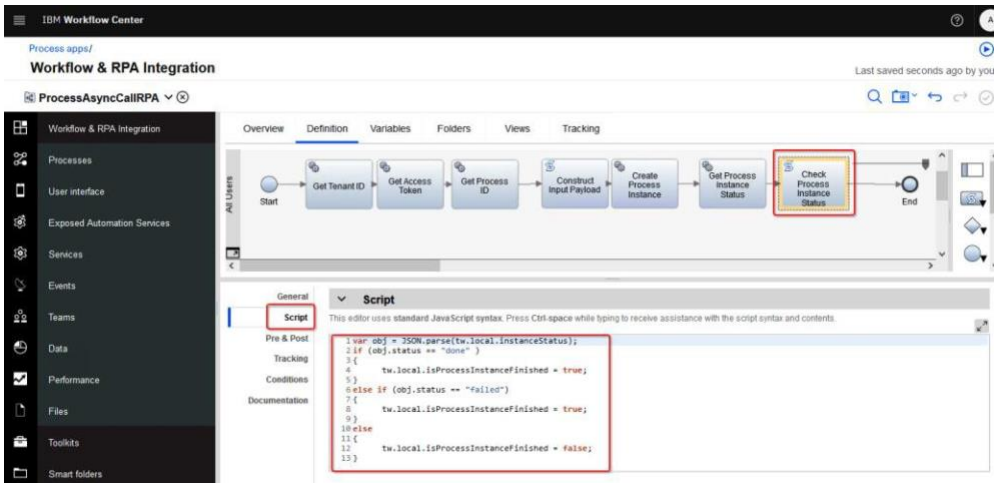


44. Once we get the process instance status, we need to check if the process instance has finished or not. Drag a server-side script activity and drop it onto the line between the **Get Process Status** and **End** activities. Change its name to **Check Process Instance Status**.

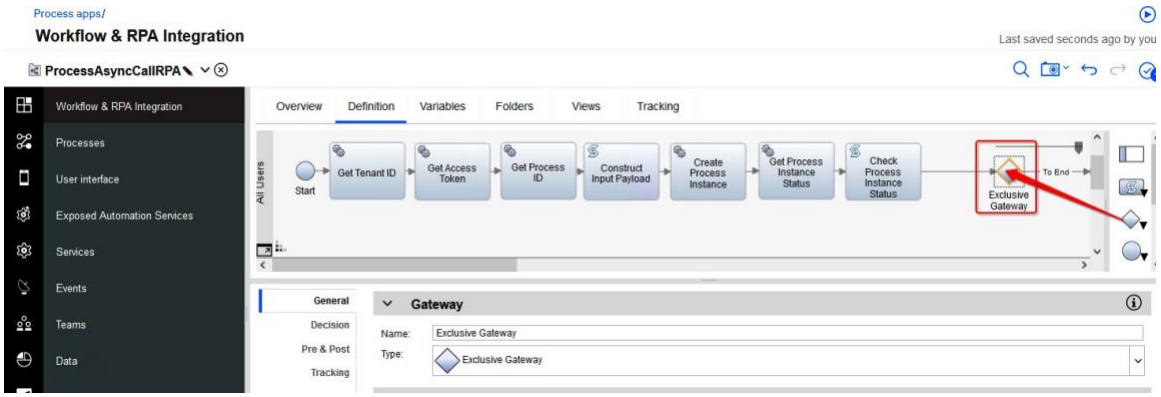


45. Click **Script** to switch to Script tab, copy and paste below JavaScript snippet into script editor.

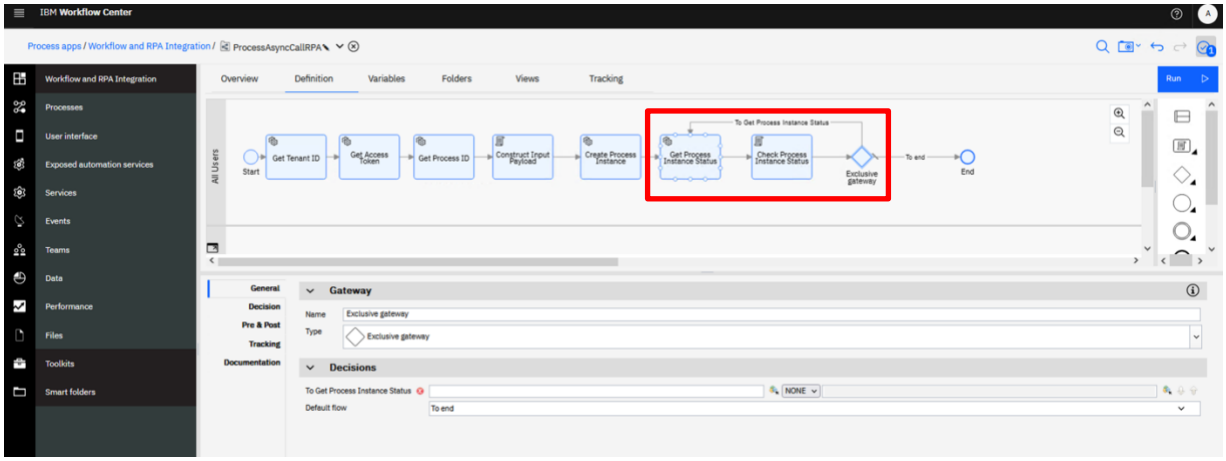
```
var obj = JSON.parse(tw.local.instanceStatus);
if (obj.status == "done" )
{
    tw.local.isProcessInstanceFinished = true;
}
else if (obj.status == "failed")
{
    tw.local.isProcessInstanceFinished = true;
}
else
{
    tw.local.isProcessInstanceFinished = false;
}
```



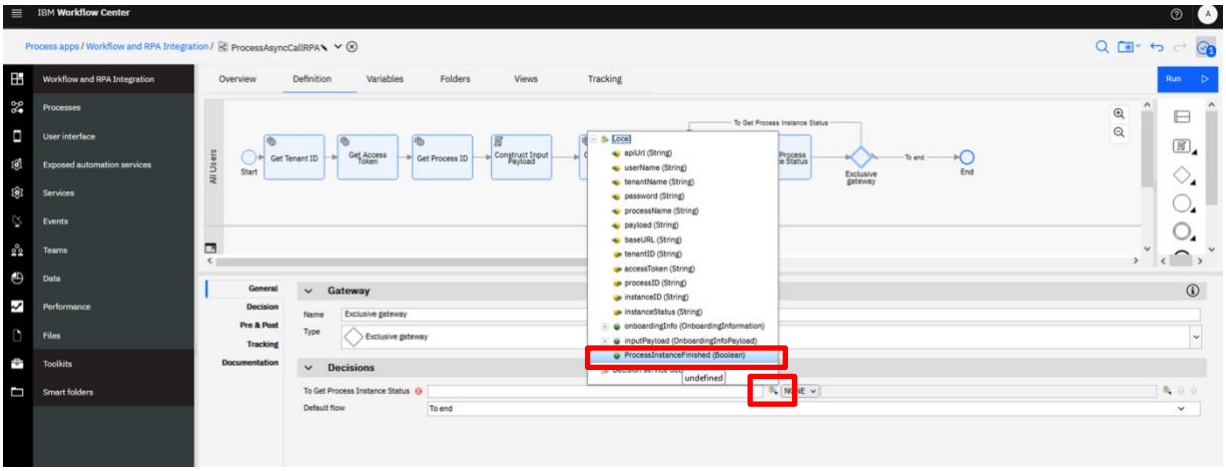
46. If the process instance has finished processing, the Workflow process will continue to the **End** activity. If it is still being executed the Workflow process needs to go back to the **Get Process Instance Status** activity to recheck the process instance status. Drag and drop a **Gateway** activity onto the line between the **Check Process Instance Status** and **End** activities.



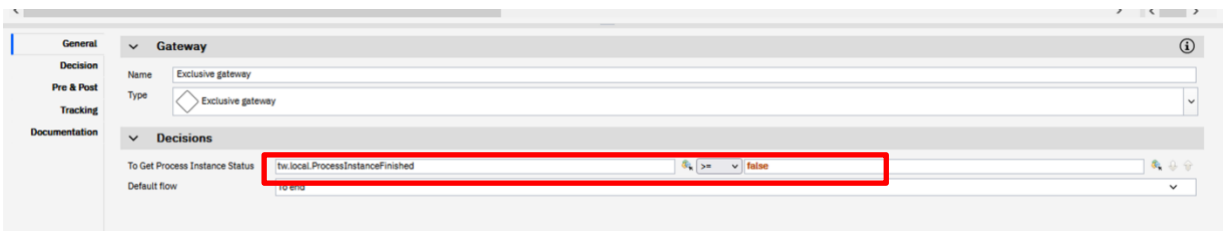
47. Connect the **Gateway** with the **Get Process Instance Status**



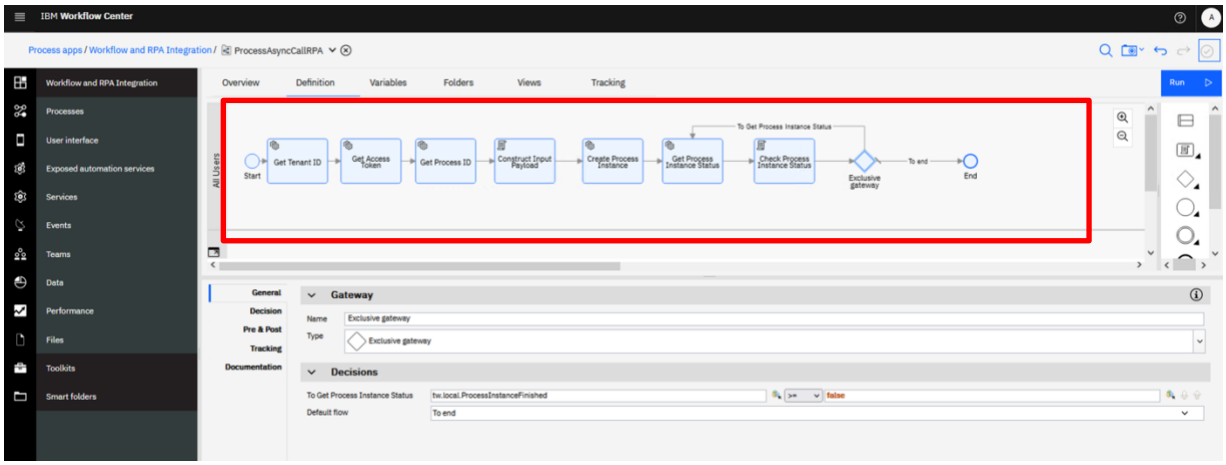
48. Click on the icon  and select the variable ProcessinstanceFineshed (Boolean)



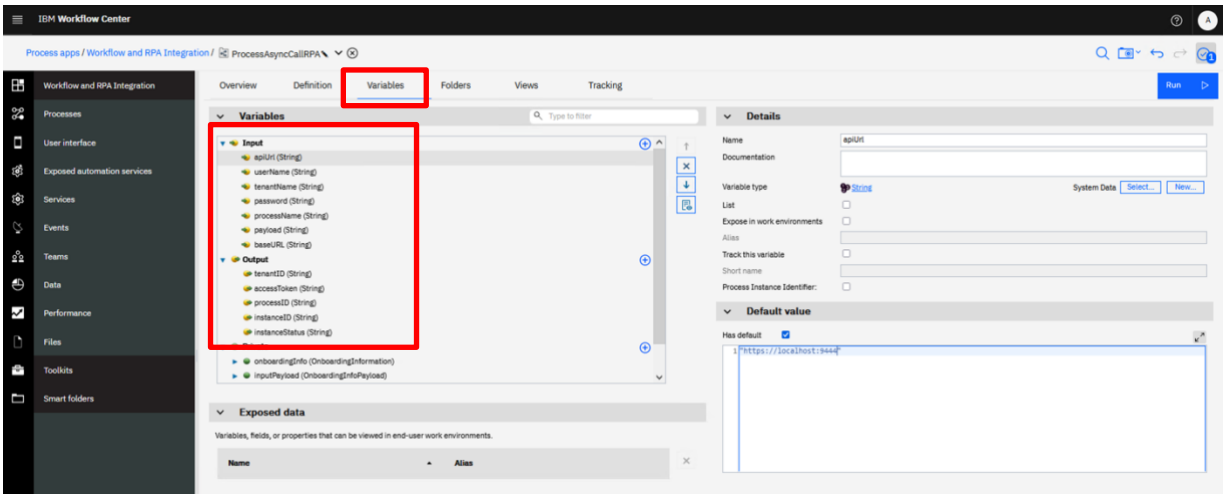
49. Click on button “none” and set condition ++ value “false”



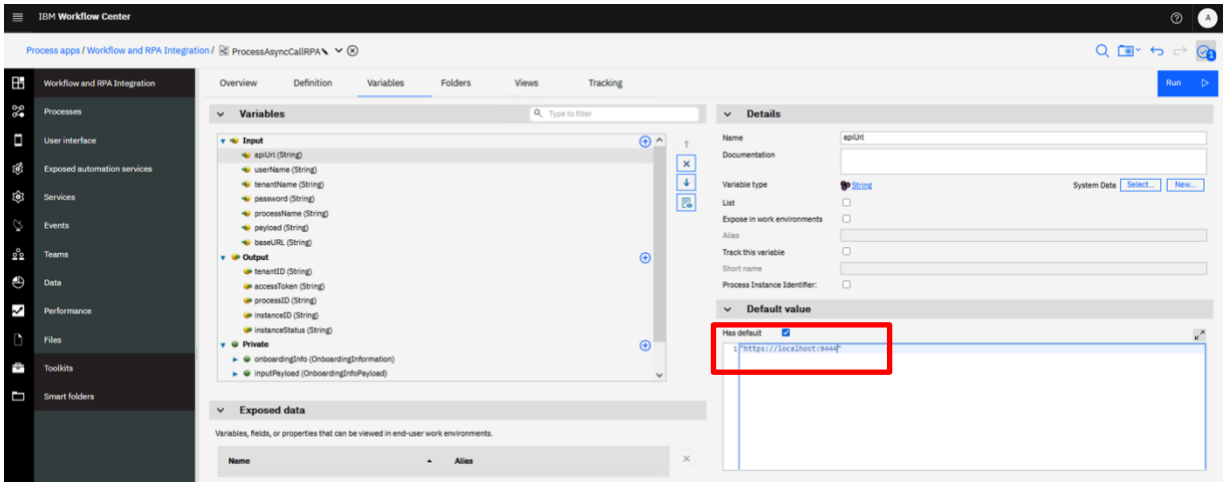
50. Your Process should be similar to the one below. It contains 5 system task activities and 2 server-side script activities. Before we can test the Process, we also need to set the RPA tenant and process information.



51. Click on the **Variables** tab to switch to the variables view. A couple of input variables have been created as part of mapping the input values of some of the system activities. To execute the Process, values for **apiUrl**, **tenantName**, **userName**, **password,** and **processName** need to be set.



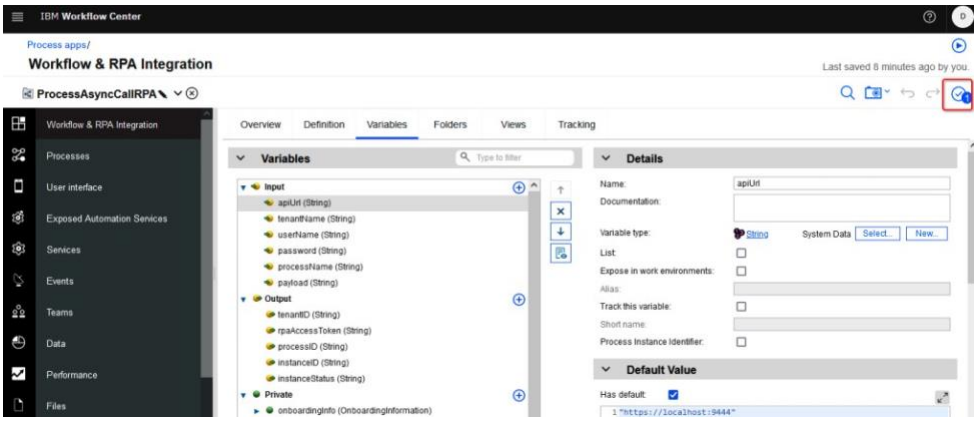
52. Select the **apiUrl** input variable, check **Has default** in the bottom-right corner and**: "https://localhost:9444"**



53. Repeat the same steps to set the default value to **tenantName**, **userName**, **password,** and **processName** as below:

    tenantName: rpa-poc
    userName: admin@rpa-poc.com
    password: passw0rd (make sure to use a zero as part of the password)
    processName: Enter the process name you defined in the exercise Create an orchestration process or use the pre-configured process name: **UsrxxxUpdateBackendSystem**.

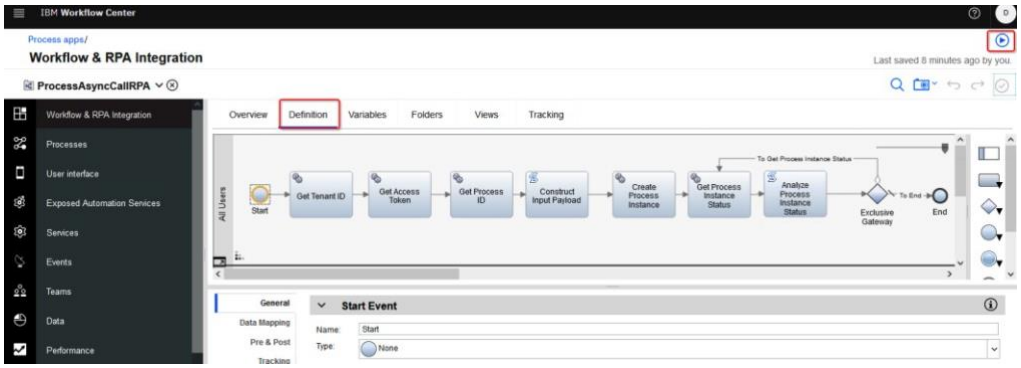
54. Click the icon in the top-right corner to save your process application.



4.2.3 Verification Instructions

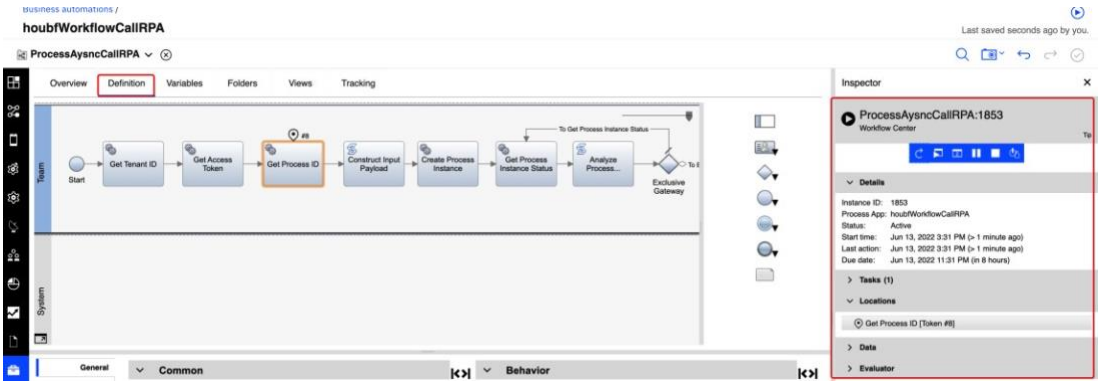
Using the Playback and Inspector, you can quickly test the Process directly from the development environment without publishing it to a Workflow server. We will use it to validate if the Workflow process authored above can successfully start the bot.

1. Click **Definition** to switch to the process diagram view, then click the icon in the top-right corner in the Workflow Designer window. It will start a new process and show it in the Inspector.



*If you see a warning message indicating Firefox prevented this site from opening a pop-up window, click the **Options** and then select "Allow pop-ups for…" to allow Firefox to open a pop-up window.*

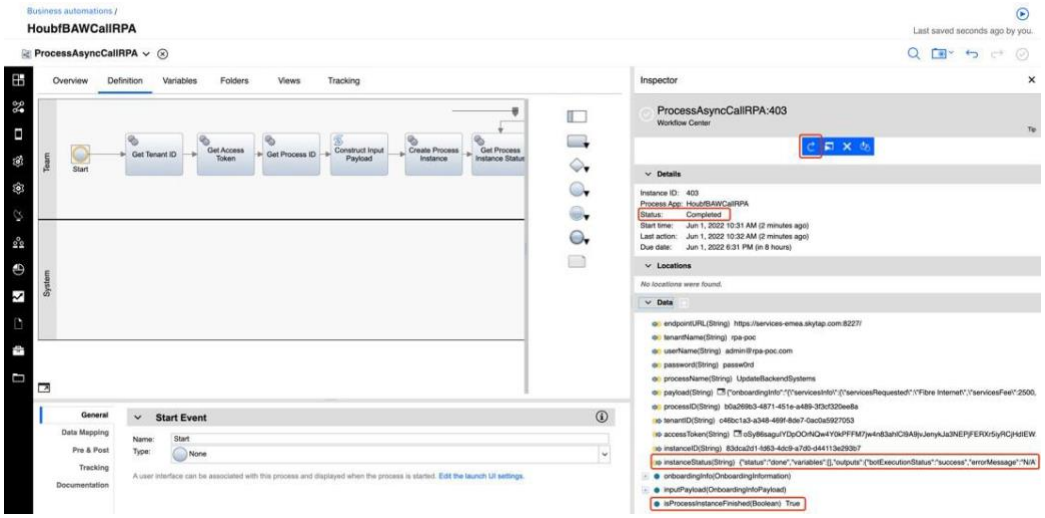
2. A new process instance will be started as below. It will call the service flows that calls the RPA server-side APIs to create process instance and execute the bot script.



3. **Watch the Windows desktop**, but don't touch the keyboard or mouse. The bot will be started. It will first start the Client Management System Java application to add the client information and grab the client ID. Next, the Service Management System web application will start to add the signed services for the client. Please return to the **Inspector** once the bot execution finishes.

   *Notes: you may not be able to see the bot execution if you access your environment using remote desktop since bot will be executed in a different user session window. If you access the environment using **VM remote console** from browser, you can see the bot execution.*

4. Click the icon in the Inspector window to refresh the process instance status. You may need to refresh it several times until its status becomes **Completed**. Check the bot execution status by reviewing the value of **isProcessInstanceFinished** in the data section. It should be "**true**" indicating that the bot has been executed successfully. Please also check the **instanceStatus** which contains three piece of information – **status**, **variables** and **outputs. outputs** hold the output response from the bot.



Summary

In this exercise, you have learned how to:

- create a queue;
- create and configure an orchestration process in the RPA control center;
- implement a workflow process that calls a series of service flows that use external services to interact with the RPA control server to run a bot;
- start an RPA bot through the IBM RPA server-side Async REST API and get its status.

Congratulations, you have successfully completed this lab!!!