ACADEMY ONLINE

RPA Tutorial: Low-Code Image Recognition

Prerequisites

Before you begin this tutorial you will need:

- Development access to an Appian Cloud environment (21.1 or higher).
    - An admin will need to add:
        - A permission to your user.
        - A resource with the same permission.
- (optional) A remote machine to execute the robotic process

If you do not have access to Appian RPA within your organization, you can request a Free Trial.

Once you have the above, you're ready to get started! At the end of this tutorial, you will have created a new robotic process using the image recognition module.

Table of Contents

Image Recognition Module

In some scenarios, Appian RPA cannot access the logical elements that make up the user interface. You can use the low-code image recognition module to find screen regions by image. In this tutorial you will build a robotic process to open the Calculator application and perform a mathematical operation. You will capture screenshots using the Agent and send them to your robotic process.

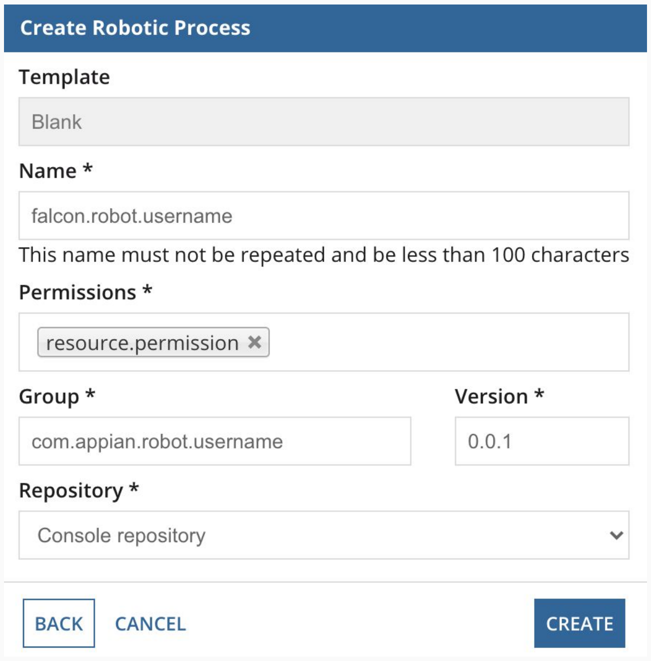
Before you get started, read through the Image Recognition Documentation.

Creating Your Robotic Process in the Console

The first phase of creating a bot consists in adding its definition to the Appian RPA console. To complete this tutorial, you will use a “Blank” template.

Use the following steps to complete this task:

1. Log into Appian.
2. Select **Appian RPA** from the navigation menu.
3. Click **Create robotic process,** located in the Dashboard toolbar.
4. Add details for your robotic process. For this bot, use the following:
    a. **Template**: Select ‘Blank’.
    b. **Name**: Your environment may be used by others. As a good practice, choose an identifier including your username.
    c. **Permissions**: Add the permission needed to launch the robotic process from your resource.
    d. **Group**: As the name field states, you should include the username .
    e. **Repository**: This element refers to the repository location. Use the ‘Console repository’ value.



Create Robotic Process

Template
Blank

Name *
falcon.robot.username
This name must not be repeated and be less than 100 characters

Permissions *
resource.permission

Group *
com.appian.robot.username

Version *
0.0.1

Repository *
Console repository

5. Click **CREATE** to create the robotic process and download the .zip file. In this tutorial, we won’t use the zip file.
6. You are now on the configuration page of the process you just created. You can change anything you set up earlier here if you need to.
7. Scroll down to the bottom of the page and click on the **Save** button and confirm.

Capture images

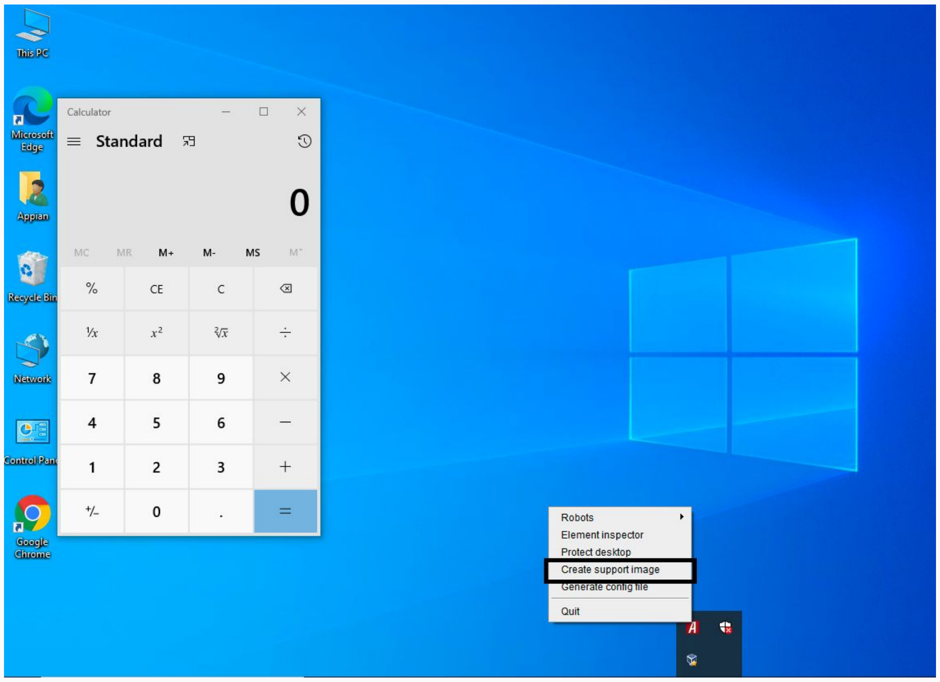
The images analyzed by the robotic process, can be obtained from the resource where the robotic process is executed. To capture an image, we will use the create support image function available in the Appian RPA agent. For further details, you can review the documentation.

Try to capture the most representative pixels of the image and, if possible, ensure these are small images. Applying these two best practices improves the image recognition performance.

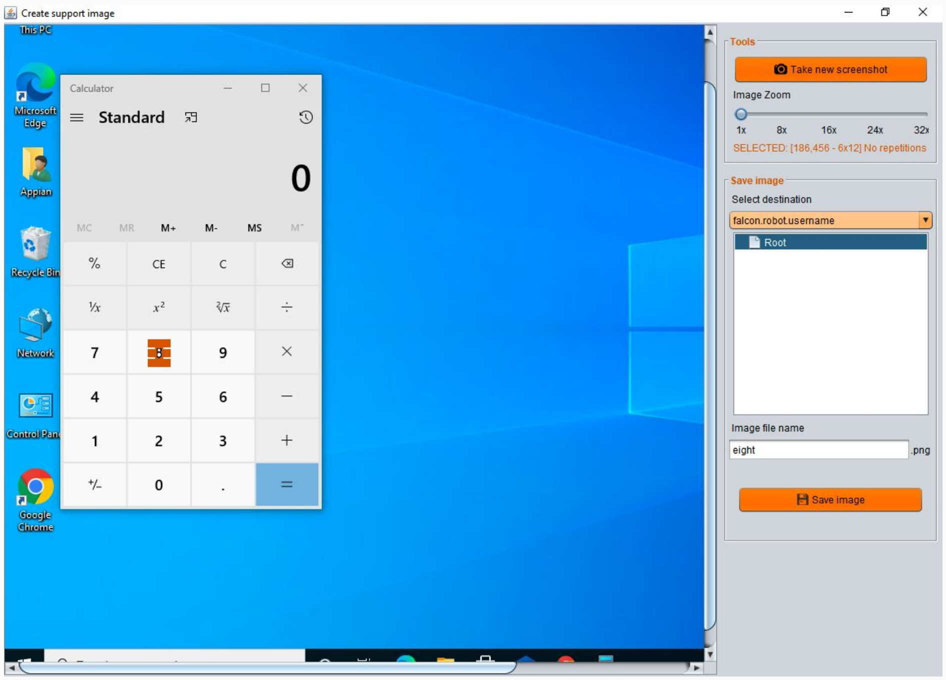
The first image to be captured is the “8” button.

Complete the following steps to save an image for that number button:

1. Start the Appian RPA agent in your resource.
2. Open the Calculator application manually.
3. Select the **‘Create support image’** option from the Appian RPA agent menu.



4. Mark the new image.
5. Select the robotic process where it will be used.



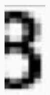
SELECTED: [186,456 - 6x12] No repetitions

Select destination: falcon.robot.username

Image file name: eight .png

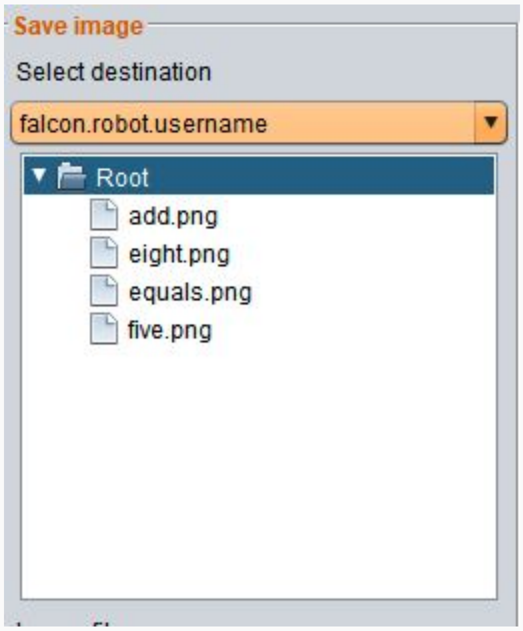
6. Save the image with the name eight.png. Click **Save image**.

The new image will be available in the support files section.



At first sight, it would be better to have the whole image. However, this is not necessary at all. We only need a representative image section to identify the region successfully.

You are going to implement the arithmetic operation "8+5=", so now you need to capture the rest of the needed images, "+", "5" and "=".

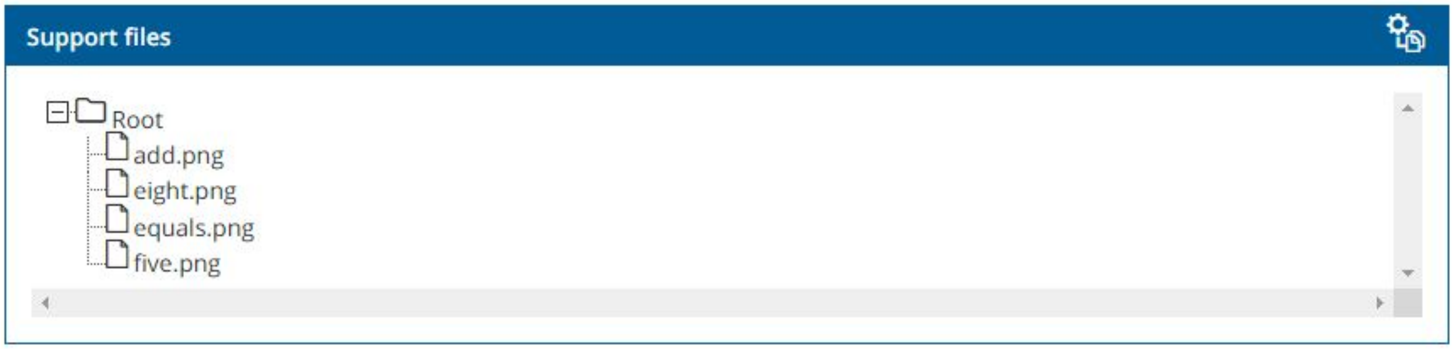


Save image

Select destination

falcon.robot.username

```
▼ Root
    add.png
    eight.png
    equals.png
    five.png
```



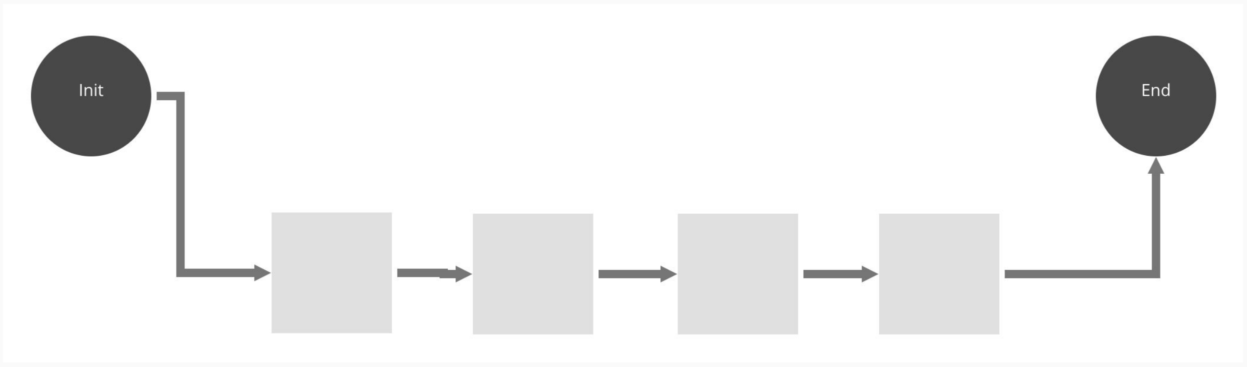
Support files

```
⊟ Root
  add.png
  eight.png
  equals.png
  five.png
```

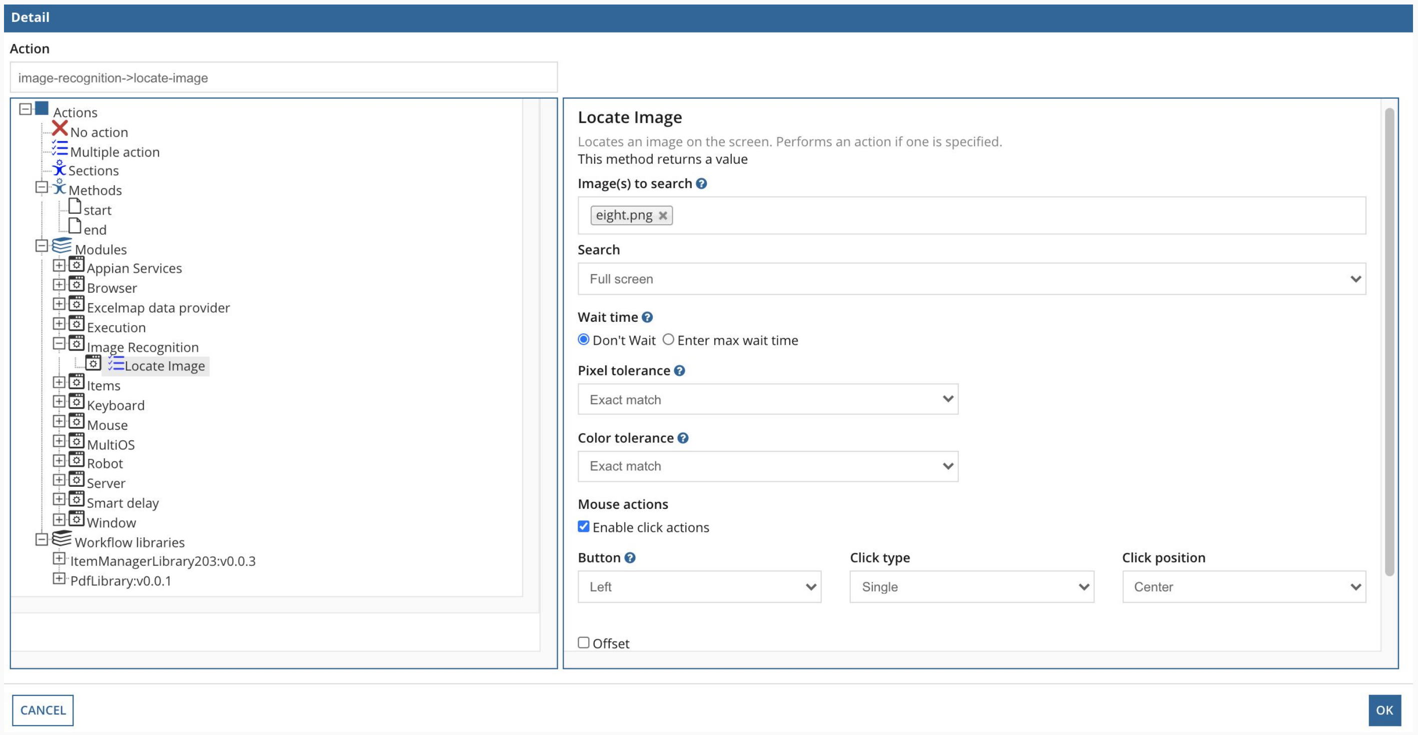
Add low code actions

Once you have the images, you need to add the low code actions to click the buttons.

1. Go to your robotic process configuration and add four new actions into the workflow, one for each image.



2. Select the first action, and add it to the Locate Image action. You must choose:
   a. Image to search : eight.png
   b. Enable click actions.



Action: image-recognition->locate-image

Locate Image

Locates an image on the screen. Performs an action if one is specified.
This method returns a value

Image(s) to search: eight.png

Search: Full screen

Wait time: [✓ Don't Wait] Enter max wait time

Pixel tolerance: Exact match

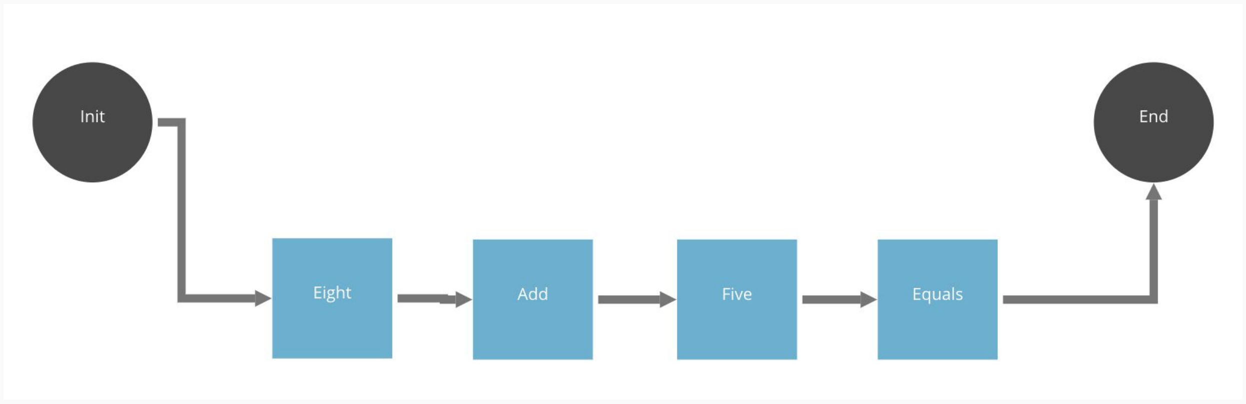
Color tolerance: Exact match

Mouse actions: [✓ Enable click actions]

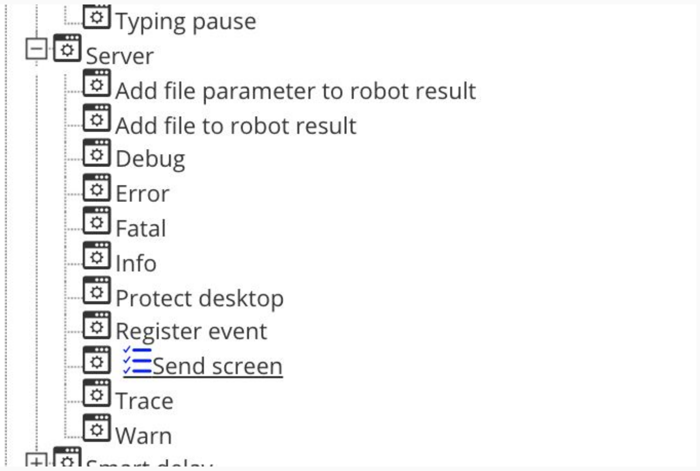
Button	Click type	Click position
Left	Single	Center

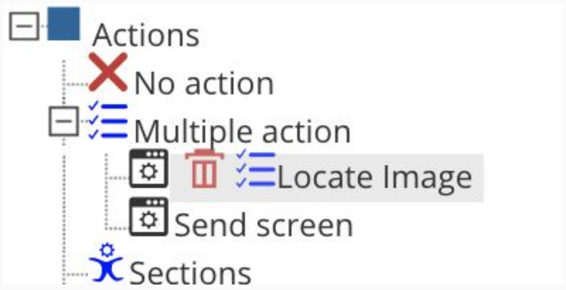
☐ Offset

3. Add the name to the action and repeat the steps with "+" , "5" and "=". The final result should be:



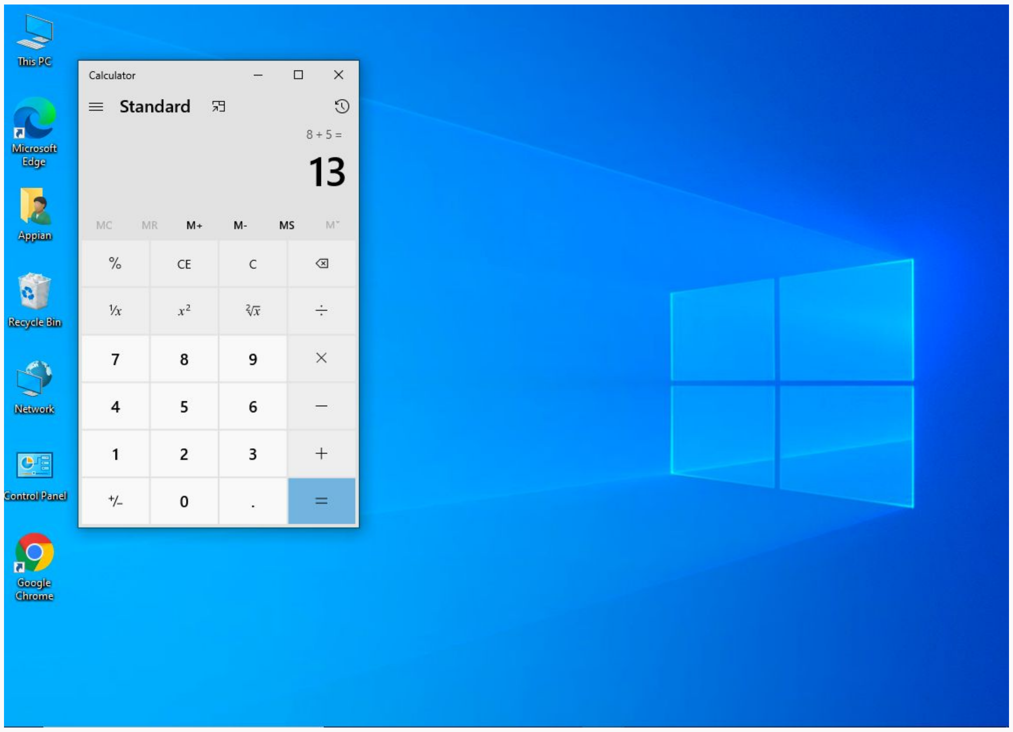
4. Before running the process, you need to add a screenshot to see the result in the log. You can add the action Server -> Send Screen" to the last action, and configure a Multiple Action.





Running Your Robotic Process

Once the new images have been captured and the actions defined, you should have the AppianRPAagent.exe running on the resource. Leave the calculator application open and run the robotic process from the console. After the execution, you should see the number 13 displayed.



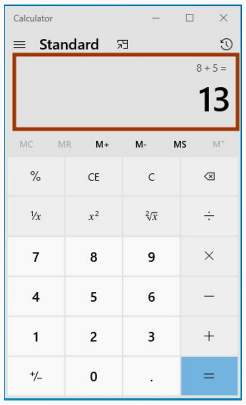
If you have any issues during the previous steps, review the [Troubleshooting]() page.

Capture the result

Now, you need to capture the arithmetic answer and set it as the result of the item. To do this, you need to do two things:

- Capture the answer using the clipboard.
- Use the Item low code module.

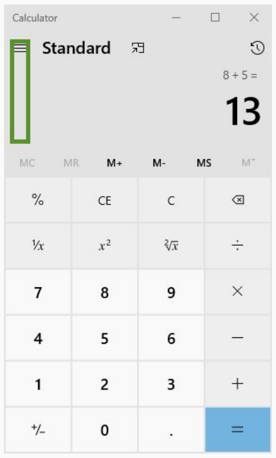
If you're uncertain of the steps to capture the answer, you can try do it manually. In this case, you can click anywhere inside the orange rectangle and press CTRL + C on your keyboard.



Now, you need to create a low code action to do the same. First, the bot needs to click inside the rectangle, so you have to capture a representative image. But, what image should you capture? You can't get the whole rectangle or a part of it because:

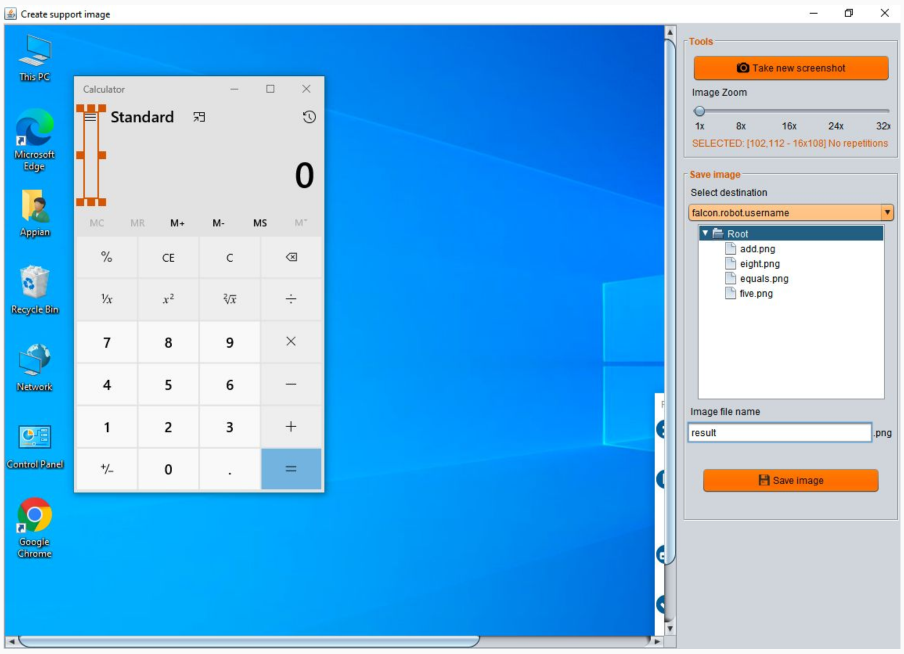
1. If the captured image includes the arithmetic answer, you can only use the image if you know the answer is 13.
2. If you capture just the gray rectangle, that is not a representative image, and the bot could find the image elsewhere

What can you do? Select a representative image and include part of the target region. Once the image is found, you can click a relative position of the image. For example, this green rectangle.



This is a typical case when using image recognition. For example, when you need to fill in a text field you must include the label of the field in your image because all the empty text fields look the same.

You can capture the image on the resource the same way you captured the other images.



SELECTED: [102,112 - 16x108] No repetitions

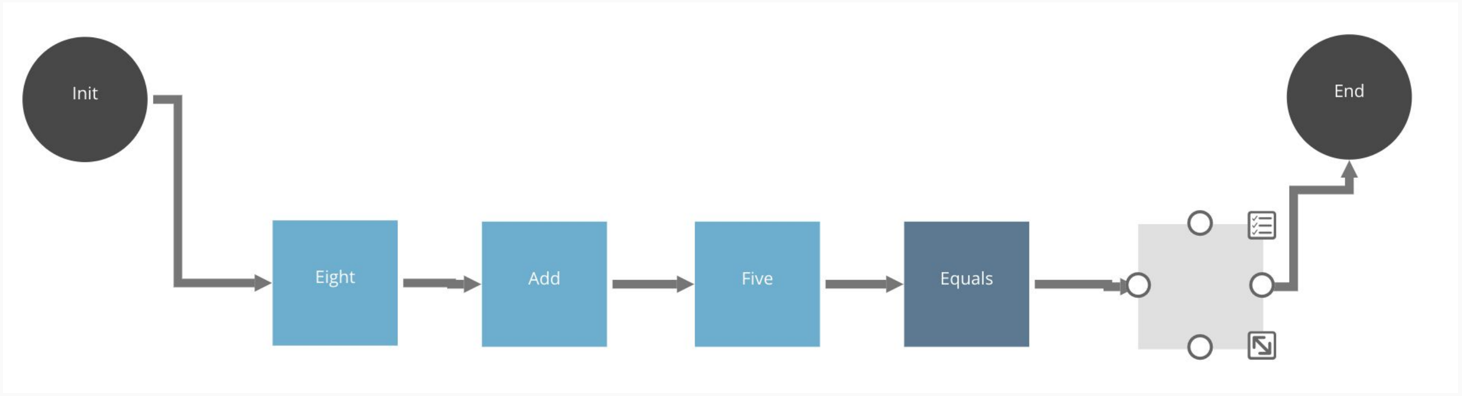
Select destination: falcon.robot.username

Image file name: result .png

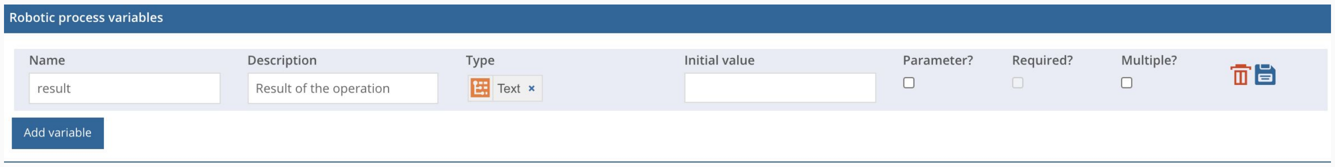
Next, create a new action.

This will be a multiple action, where the bot will:

- Look for the image, and click on a relative position
- Copy and paste the result in a Robotic process variable



Before you add the methods, you must create the result process variable.

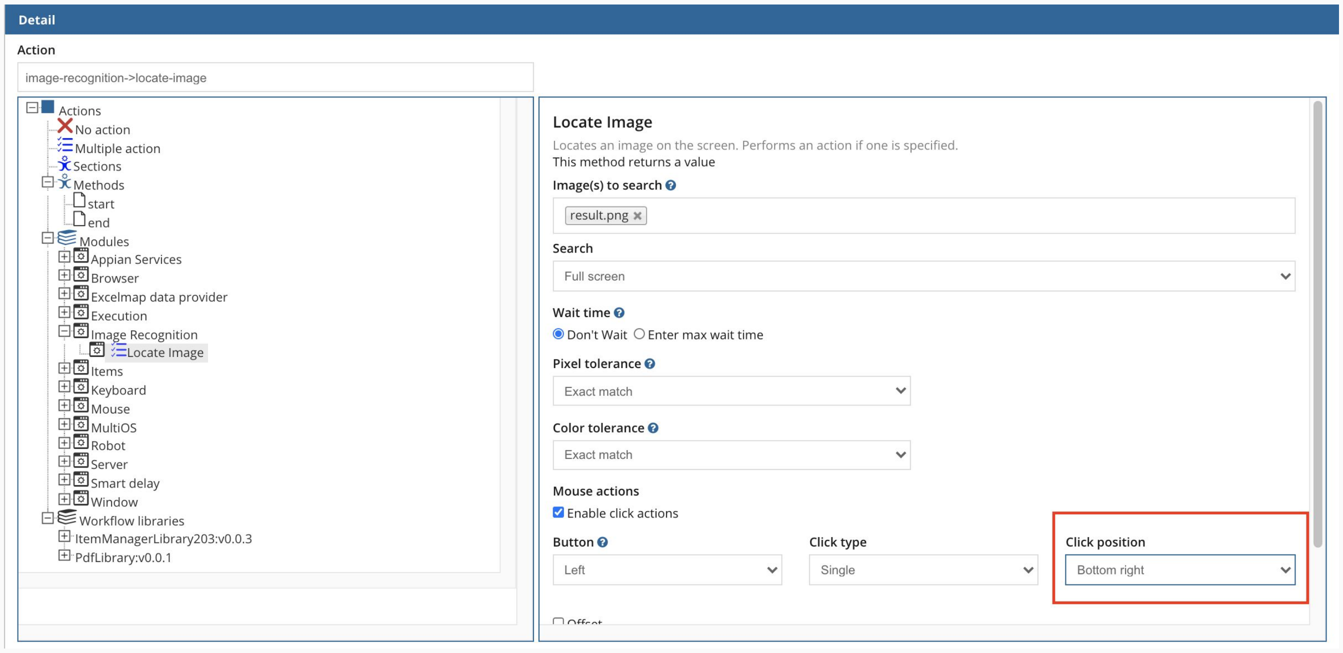


Robotic process variables

Name	Description	Type	Initial value	Parameter?	Required?	Multiple?
result	Result of the operation	Text		☐	☐	☐

Add variable

Then you will use the Locate image method to add the result.png, but this time you will indicate a different click position.



Detail

Action: image-recognition->locate-image

Locate Image

Locates an image on the screen. Performs an action if one is specified.
This method returns a value

Image(s) to search: result.png

Search: Full screen

Wait time: [✓ Don't Wait] Enter max wait time

Pixel tolerance: Exact match

Color tolerance: Exact match

Mouse actions
☑ Enable click actions

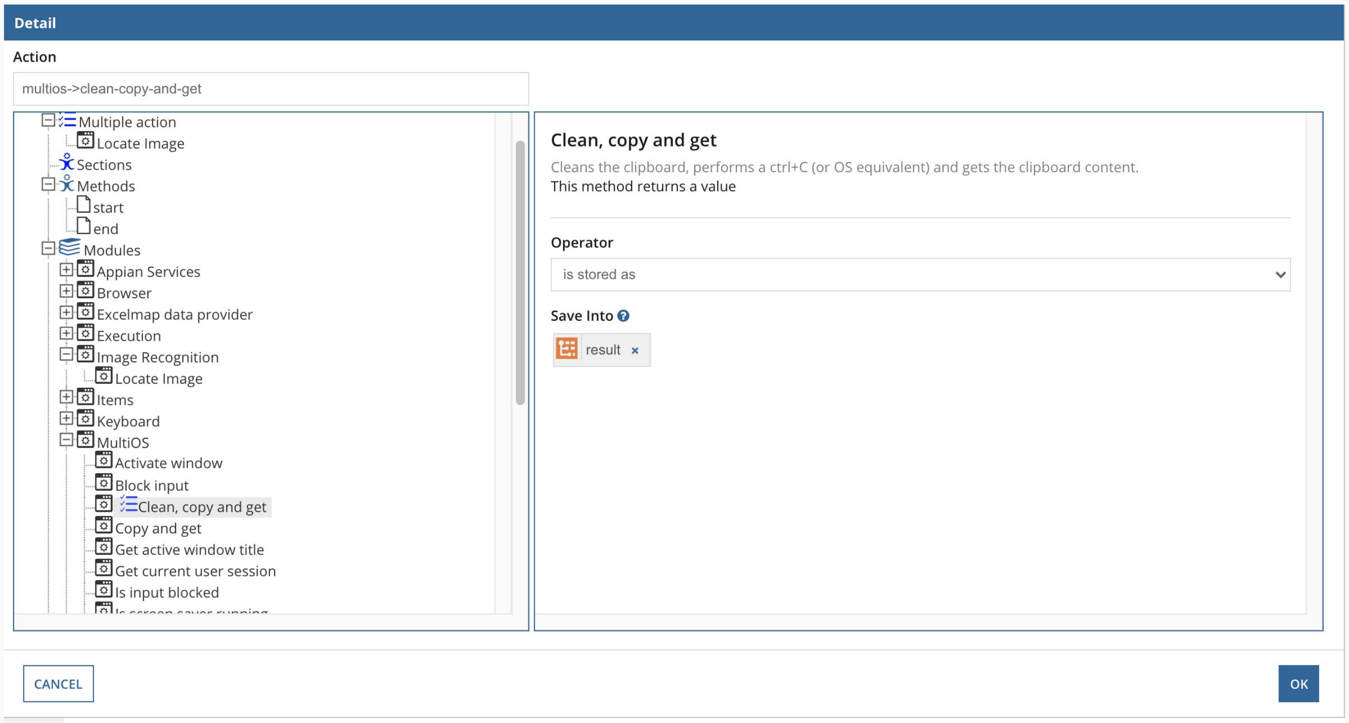
Button: Left

Click type: Single

Click position: Bottom right

☐ Offset

Add the action MultiOS -> Clean, copy and get



Detail

Action

multios->clean-copy-and-get

- Multiple action
  - Locate Image
- Sections
- Methods
  - start
  - end
- Modules
  - Appian Services
  - Browser
  - Excelmap data provider
  - Execution
  - Image Recognition
    - Locate Image
  - Items
  - Keyboard
  - MultiOS
    - Activate window
    - Block input
    - Clean, copy and get
    - Copy and get
    - Get active window title
    - Get current user session
    - Is input blocked

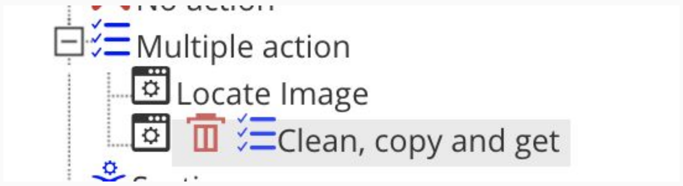
Clean, copy and get

Cleans the clipboard, performs a ctrl+C (or OS equivalent) and gets the clipboard content.
This method returns a value

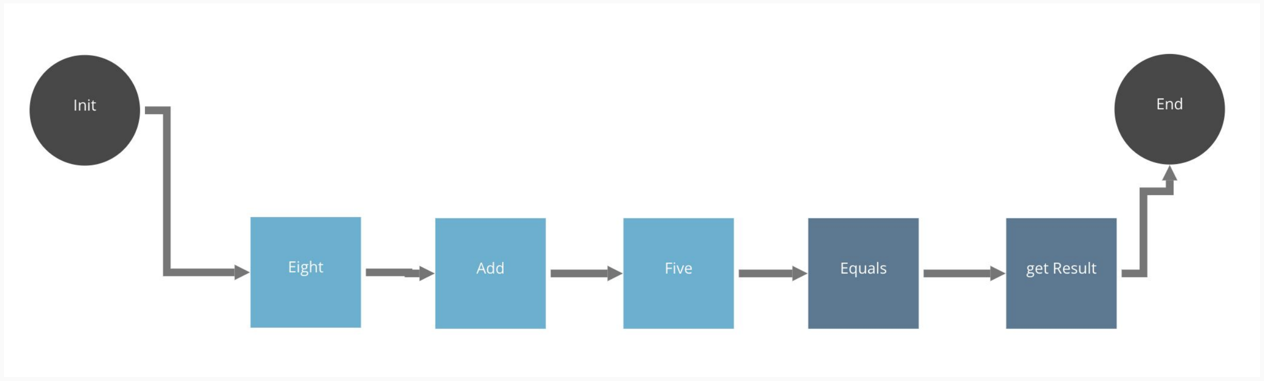
Operator: is stored as

Save Into: result

Be sure that the action contains both methods:

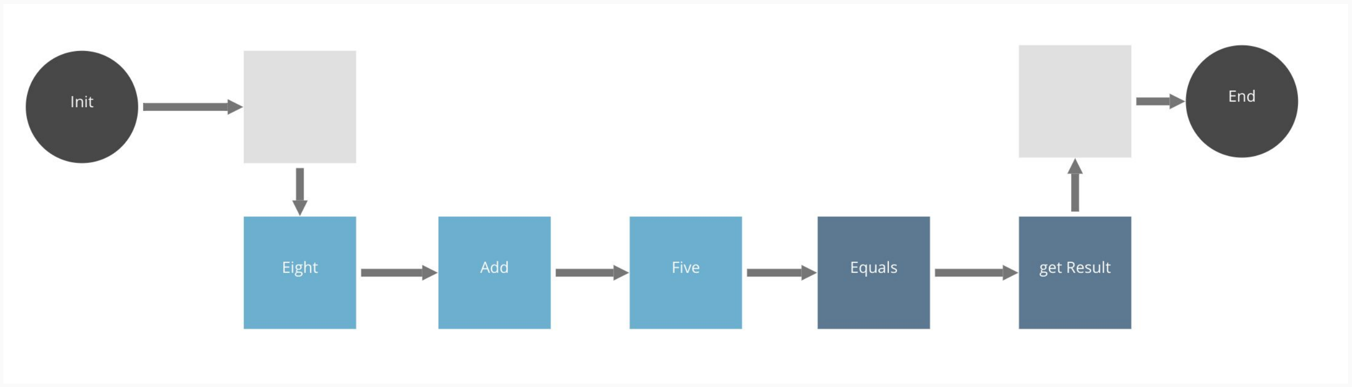


- Multiple action
  - Locate Image
  - Clean, copy and get

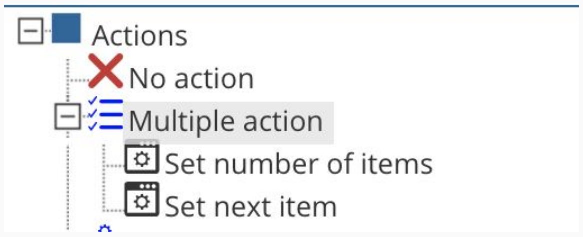


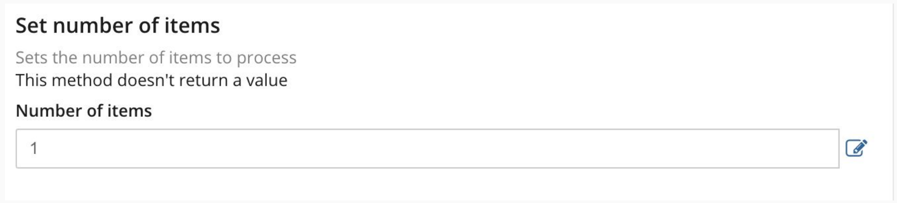
Finally you are going to use Item low code action to show the result of the execution in the console.

You need to add two new low code actions:



In the first one, you will set the number of items and the next item action. In this example, you will add constants, but remember that normally you will use these actions with Expression Rules to iterate over several items



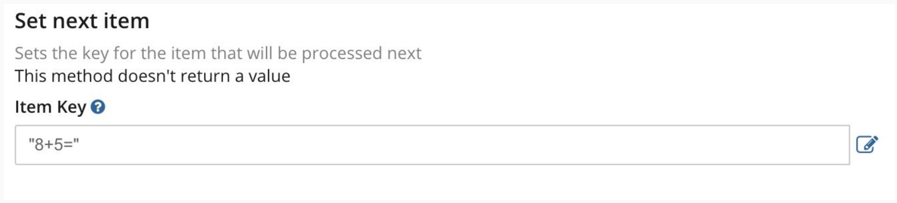


Set number of items

Sets the number of items to process
This method doesn't return a value

Number of items

1



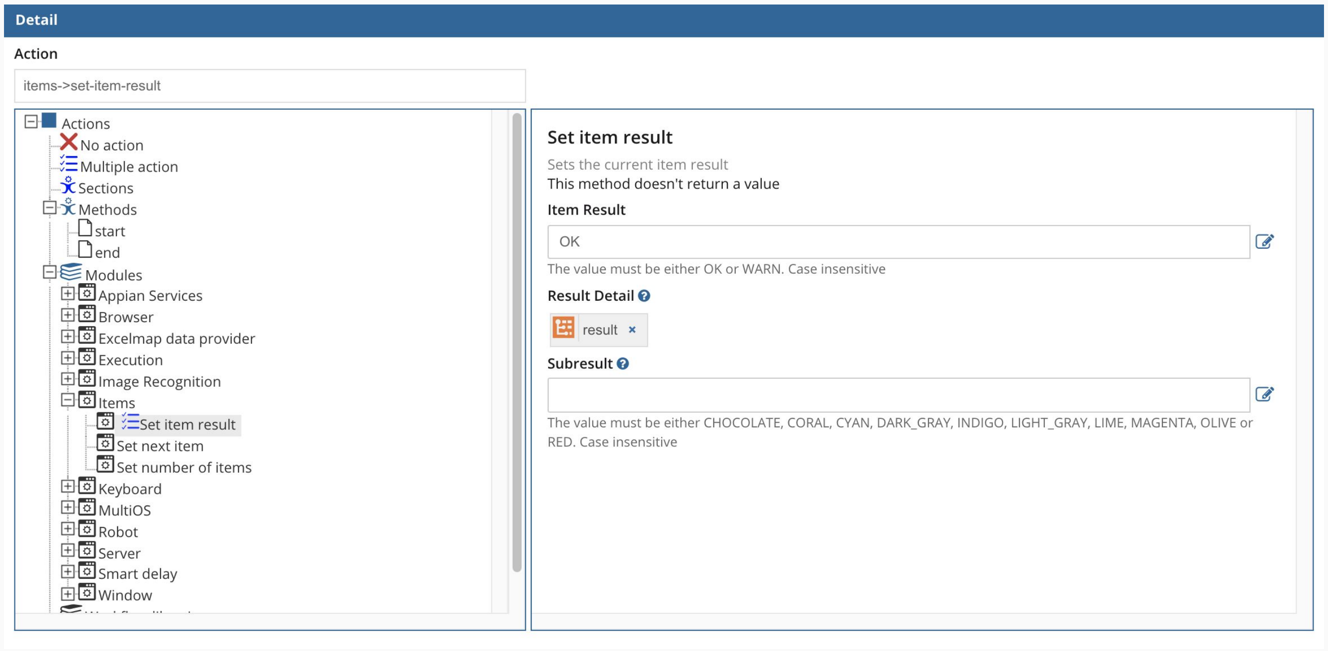
Set next item

Sets the key for the item that will be processed next
This method doesn't return a value

Item Key

"8+5="

The last action, will set the result of the item with the value of the variable.



Detail

Action

items->set-item-result

Set item result

Sets the current item result

This method doesn't return a value

Item Result

OK

The value must be either OK or WARN. Case insensitive

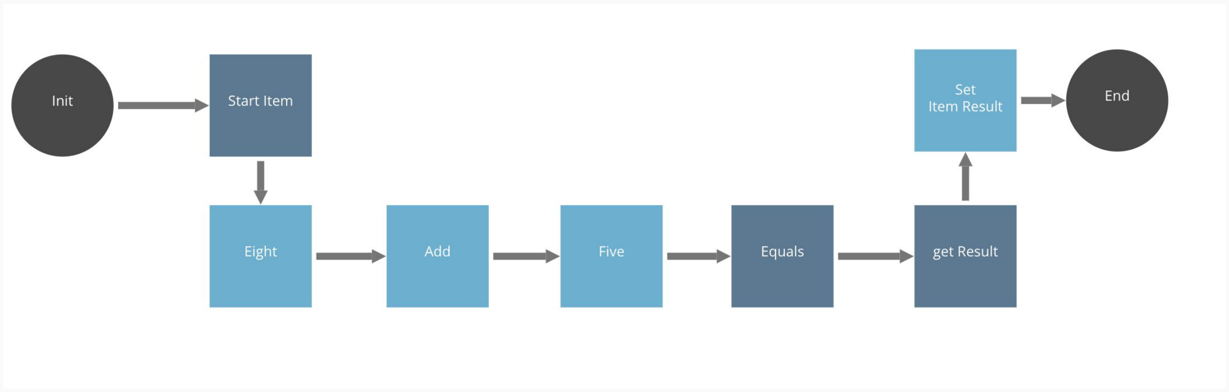
Result Detail

result

Subresult

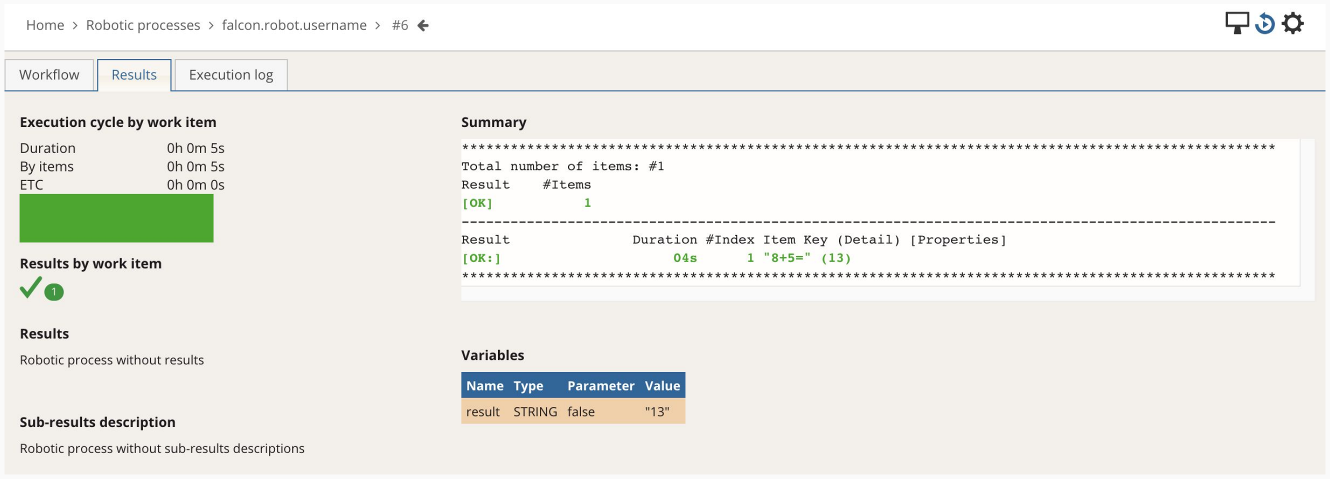
The value must be either CHOCOLATE, CORAL, CYAN, DARK_GRAY, INDIGO, LIGHT_GRAY, LIME, MAGENTA, OLIVE or RED. Case insensitive

The final workflow will be:



Running again the Robotic Process

The robotic process will execute the same operation, but now you can review the result in the console, and most important, you will be able to use this result in another robotic process or Appian process model.



Execution cycle by work item

Duration	0h 0m 5s
By items	0h 0m 5s
ETC	0h 0m 0s

Results by work item

✓ 1

Results

Robotic process without results

Sub-results description

Robotic process without sub-results descriptions

Summary

```
****************************************************************************************************
Total number of items: #1
Result    #Items
[OK]           1
----------------------------------------------------------------------------------------------------
Result               Duration #Index Item Key (Detail) [Properties]
[OK:]                     04s       1 "8+5=" (13)
****************************************************************************************************
```

Variables

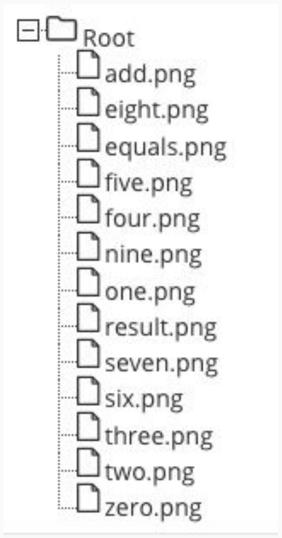
Name	Type	Parameter	Value
result	STRING	false	"13"

Challenge Activity

Now that you know how to work with image recognition, you can try to automate the entire calculator, and implement a robotic process that performs several operations.

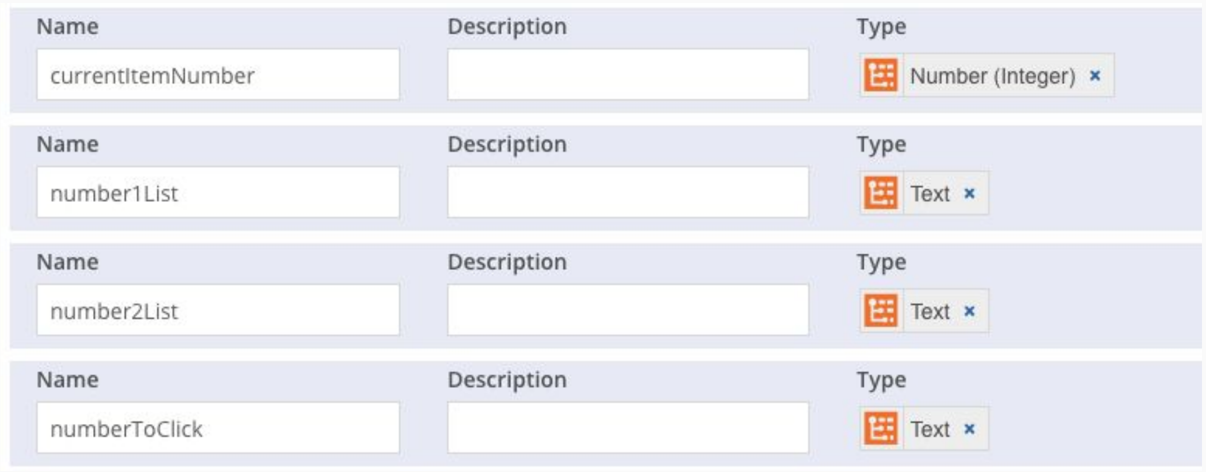
Here are some suggested steps:

1. Capture the images for all number



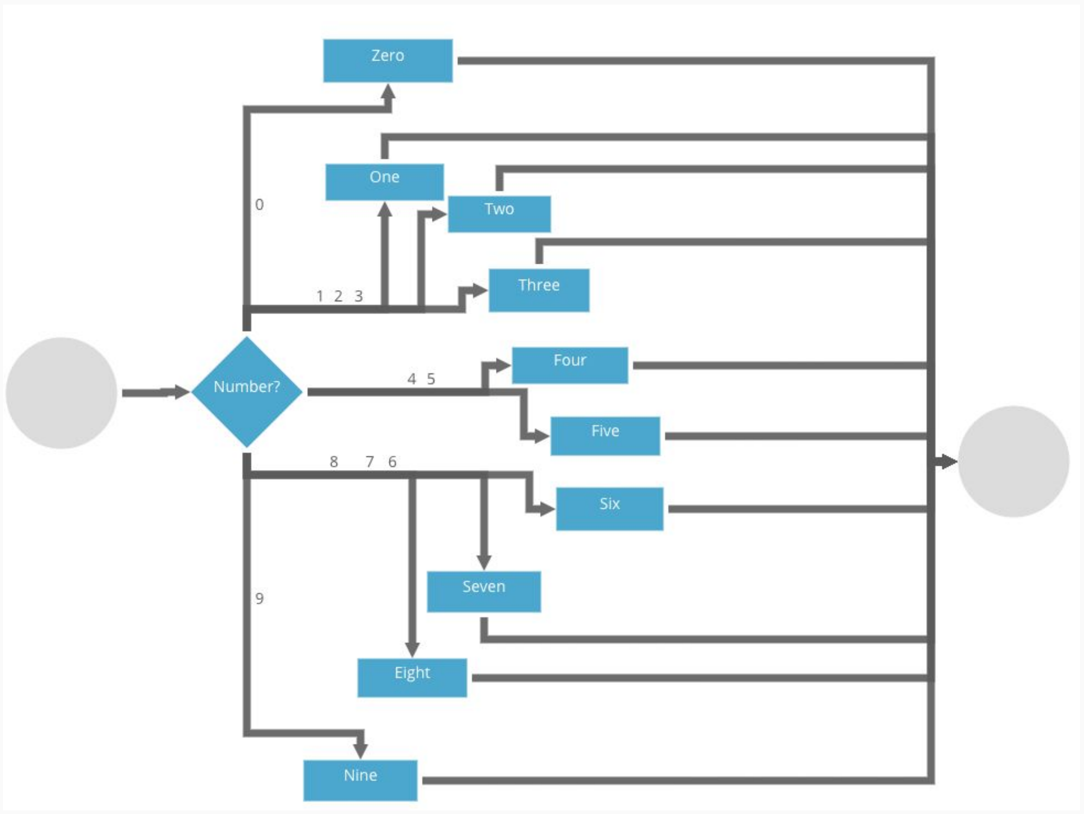
```
Root
  add.png
  eight.png
  equals.png
  five.png
  four.png
  nine.png
  one.png
  result.png
  seven.png
  six.png
  three.png
  two.png
  zero.png
```

2. Define the variables to iterate. In this case, you can use two lists of numbers. In a more realistic bot, the input would be dynamic.

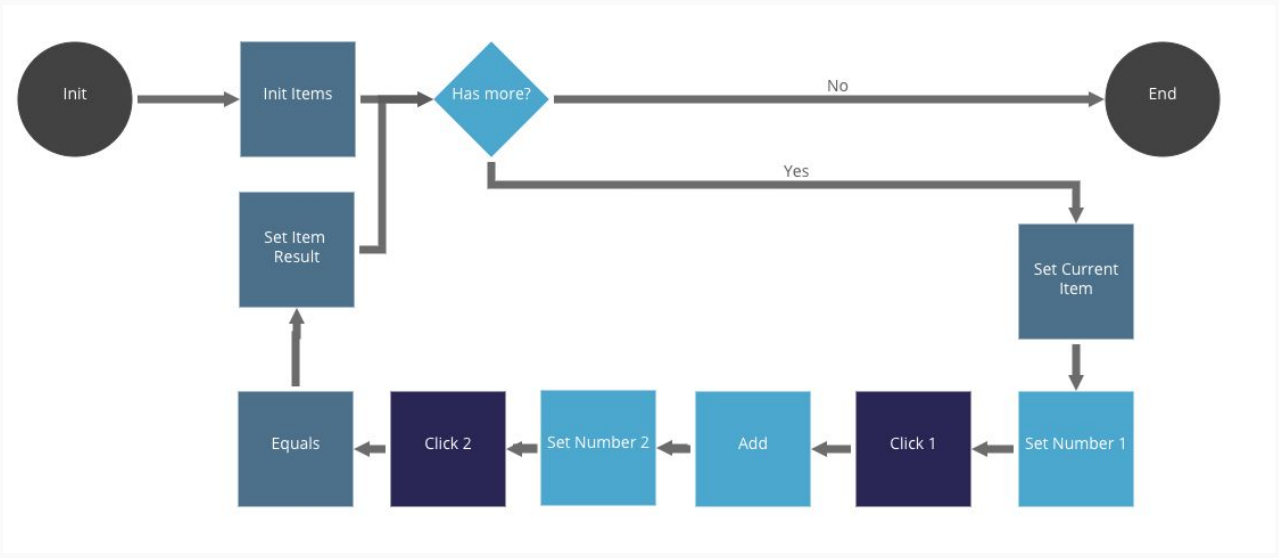


Name	Description	Type
currentItemNumber		Number (Integer)
number1List		Text
number2List		Text
numberToClick		Text

3. Create a subsection to click a button



4. Change your workflow to iterate and click each button.


Init
Init Items
Has more?
No
End
Yes
Set Item Result
Set Current Item
Equals
Click 2
Set Number 2
Add
Click 1
Set Number 1