Blue Prism Labs

Lab 4: Process Best Practices

Document Revision 1.0

Trademarks and copyrights

Contents

Introduction

As mentioned in Lab 2, every organization will develop some of their own best practices and naming conventions dependent on their organization and structure. The Blue Prism portal (https://portal.blueprism.com) has some initial suggestions as well as templates to make building quicker and easier.

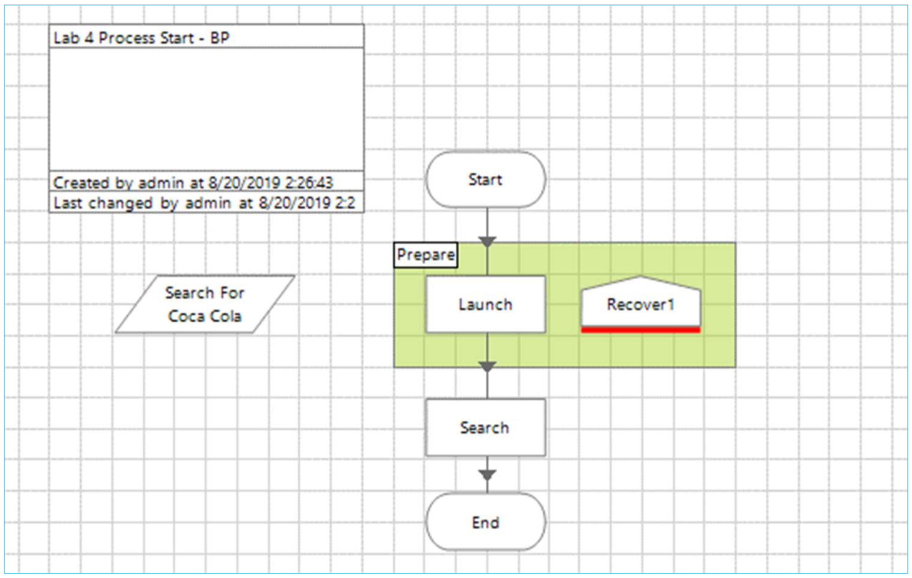
One best practice we will focus on here includes following a format of “Prepare”, “Work”, and “Cleanup” in your process’s flow. All processes start with readying the workspace for the work to be done and includes launching any required applications, logging in and navigating to the start page/screen. These steps fall under the “Prepare” section. Once complete, the work can be done – the “Work” section. Finally, after the work is completed, all applications can be closed – the “Cleanup” section. This process is the same whether it is done by a human worker or a digital worker.

You will utilize “Blocks” to isolate the stages within each section and identify the different sections of the process. This is the same process we used in Lab 2 – Object Best practices for “Verify”, “Do”, and “Confirm”.

We will also add in some exception handling to “catch” exceptions from the objects and either retry or terminate the process depending on what exceptions occur. In addition to organization, blocks also allow for defining different exception pathways. For example, if the “Prepare” section cannot be completed (i.e. the applications can’t be launched), then terminating the process may be appropriate. However, if there is an issue encountered in the “Work” section after the application was already running, a retry loop may be a better alternative than simply terminating the process.

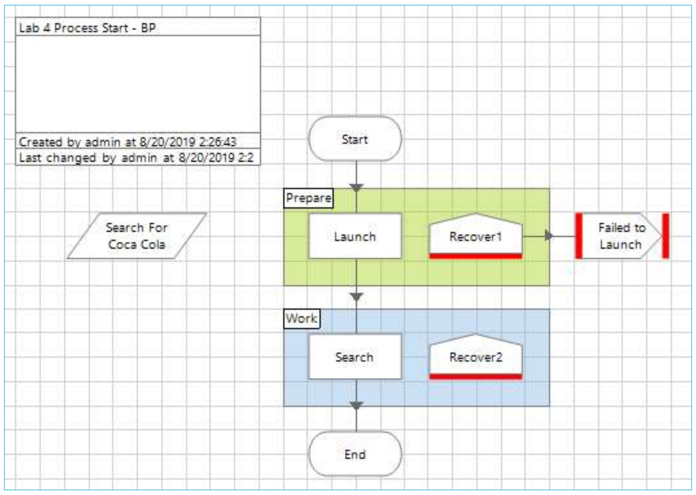
Lab 4: Process Best Practices

1) Open the Process called "Lab 4 Process Start - BP". Add a "Recover" stage next to the "Launch" stage. Add a block around both "Launch" and "Recover" and rename it "Prepare". Change the color to Lime. Blocks are useful for grouping shapes together, but their primary purpose is handling exceptions. If there is ever an exception, something that would cause an error message while in the debug mode, a Recover stage will "catch" it. This is VERY useful when making a resilient process. If an exception happens inside of a Block, and there is also a Recover inside of the same Block, that Recover will catch that exception. In this way, you can have multiple Recover stages, and design extremely resilient Processes.

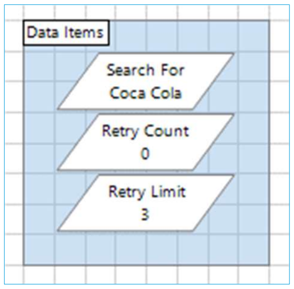


2) Add an Exception stage to the right of Recover1 and link Recover1 to it. Double click the Exception to open its properties. Change the name to "Failed to launch" and select the checkbox for "Preserve the type and detail of the current exception". This way, and exception that "bubbles up" from the Launch action will be preserved.

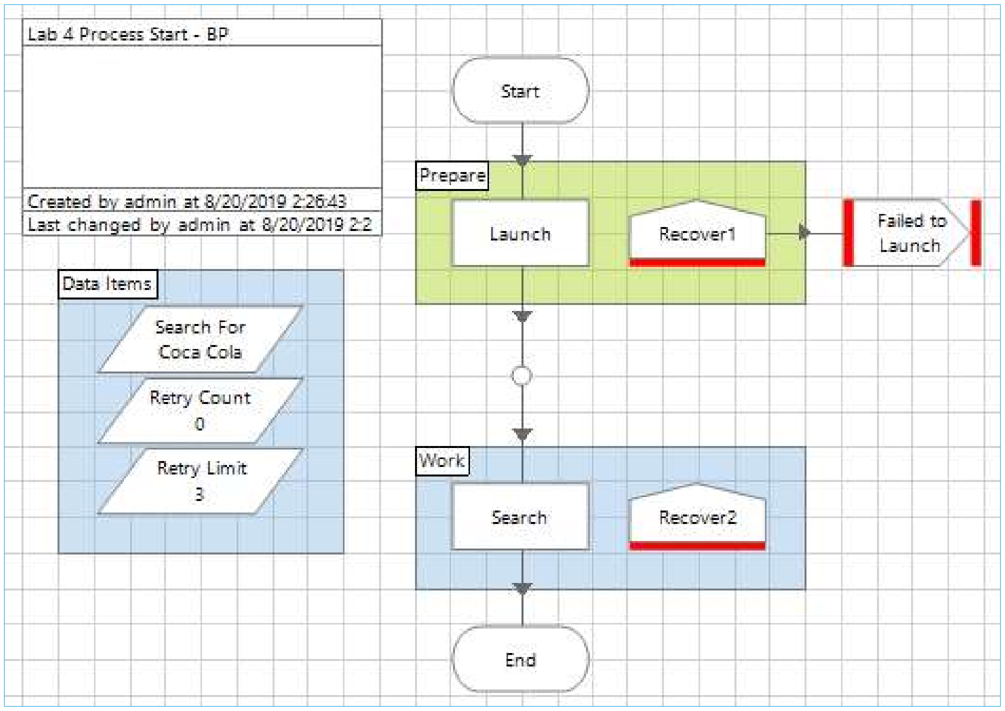
3) Add a "Recover" stage next to "Search" and a block around both "Search" and "Recover"; Rename it "Work".



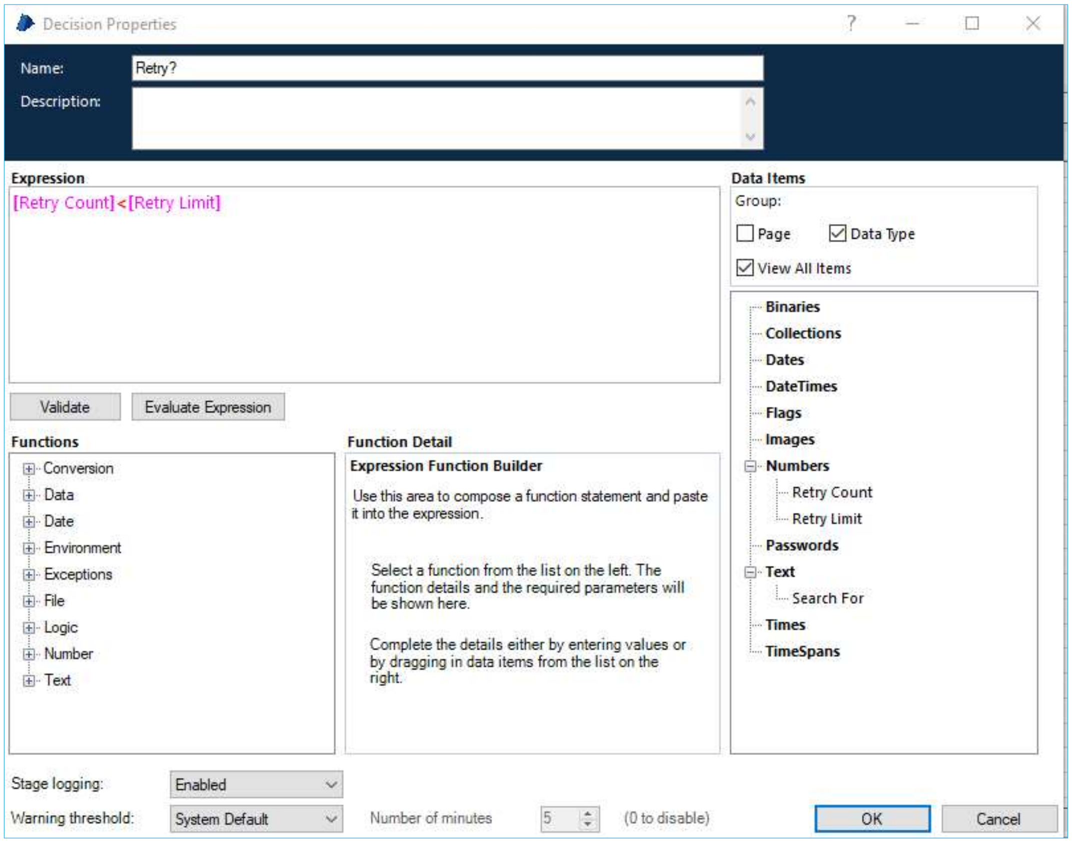
4) If something goes wrong with the "Search" action, we most likely don't want to simple crash process. Instead, let's set up a retry loop. First, add two Data Items, named "Retry Count" and "Retry Limit". Set the Data types to "Number" and set the initial values to 0 and 3 respectively. Add a block around them and name it "Data Items".



5) Delete the link between "Launch" and "Search". Add an "Anchor" between the two and link it together, as shown below. Note that you might have to move the shapes to make room. Anchors don't do anything other than help to organize links.



6) To the right of Recover2, add a "Decision" stage. Open its properties and change the name to "Retry?". Under where it says "Expression", enter "[Retry Count]<[Retry Limit]". You can do this by either typing it in or by dragging the Data Item names from the selection on the left. You can also add expressions using the built-in function wizard, located in the bottom right. The function wizard makes it simple to enter complicated expressions. Feel free to explore it! When you're done, press "OK".


Decision Properties

Name: Retry?

Description:

Expression

[Retry Count]<[Retry Limit]

Functions: Conversion, Data, Date, Environment, Exceptions, File, Logic, Number, Text

Function Detail

Expression Function Builder

Use this area to compose a function statement and paste it into the expression.

Select a function from the list on the left. The function details and the required parameters will be shown here.

Complete the details either by entering values or by dragging in data items from the list on the right.

Data Items

Group: [ ] Page [✓] Data Type

[✓] View All Items

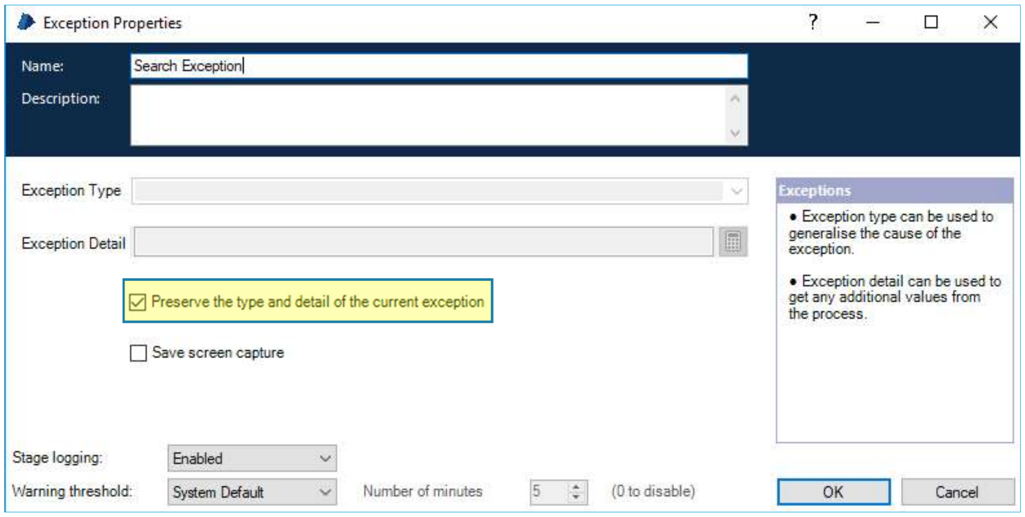
Binaries, Collections, Dates, DateTimes, Flags, Images, Numbers (Retry Count, Retry Limit), Passwords, Text (Search For), Times, TimeSpans

Stage logging: Enabled

Warning threshold: System Default

Number of minutes 5 (0 to disable)

7) Add an Exception stage to the right of "Retry?". Name it "Search Exception" and again select the checkbox for "Preserve the type and detail of the current exception"


Exception Properties

Name: Search Exception

Description:

Exception Type

Exception Detail

[✓] Preserve the type and detail of the current exception

[ ] Save screen capture

Exceptions

• Exception type can be used to generalise the cause of the exception.

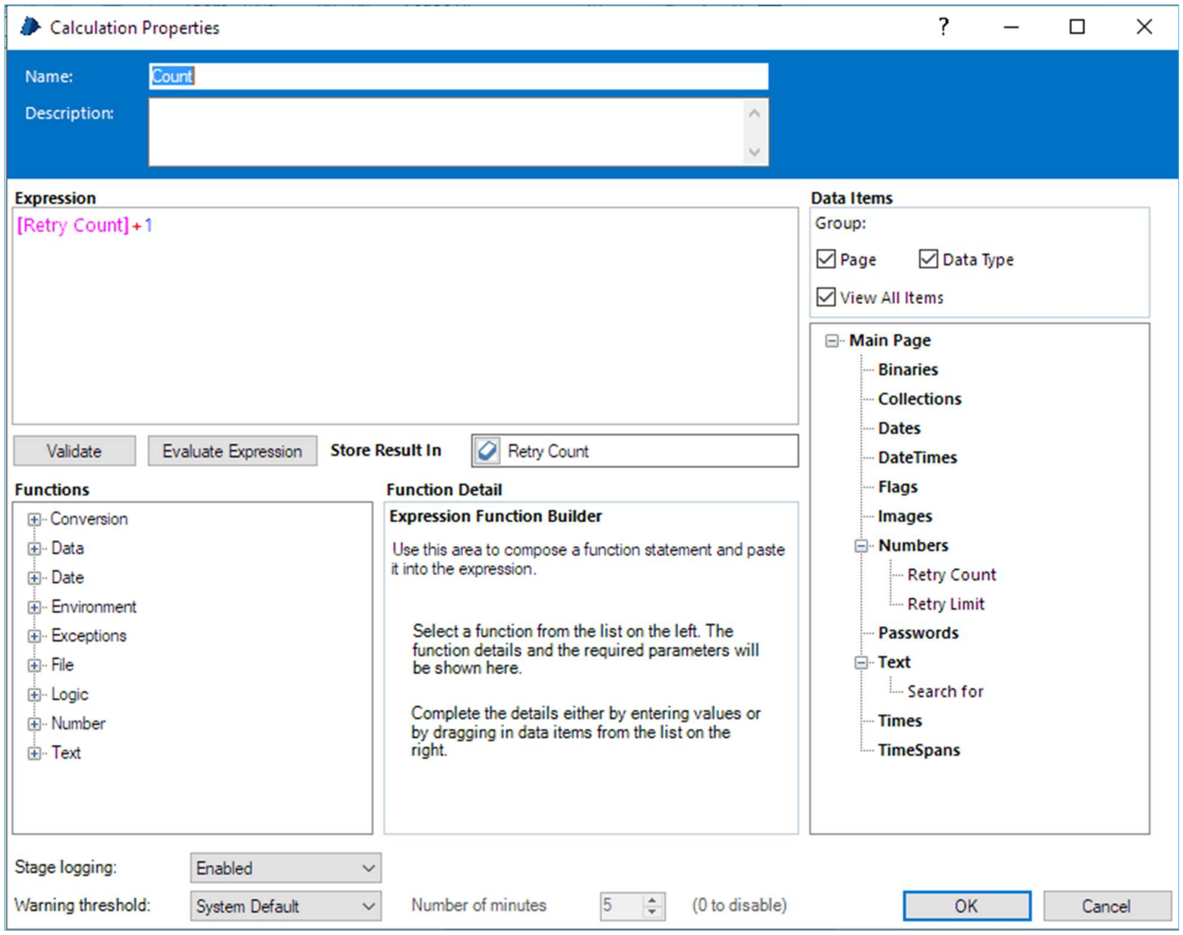
• Exception detail can be used to get any additional values from the process.

Stage logging: Enabled

Warning threshold: System Default

Number of minutes 5 (0 to disable)

8) Add a "Calculation" stage above "Retry?". Open its properties and change the name to "Count". Under where it says "Expression", enter "[Retry Count]+1". Finally, on the right hand side, expand "Text" to reveal "Retry Count". Click on "Retry Count" and drag it to the blank spot next to where it says "Store Result In". Press "OK". This stage will count the number of retry attempts.


Calculation Properties

Name: Count

Description:

Expression

[Retry Count]+1

Store Result In: Retry Count

Function Detail

Expression Function Builder

Use this area to compose a function statement and paste it into the expression.

Select a function from the list on the left. The function details and the required parameters will be shown here.

Complete the details either by entering values or by dragging in data items from the list on the right.

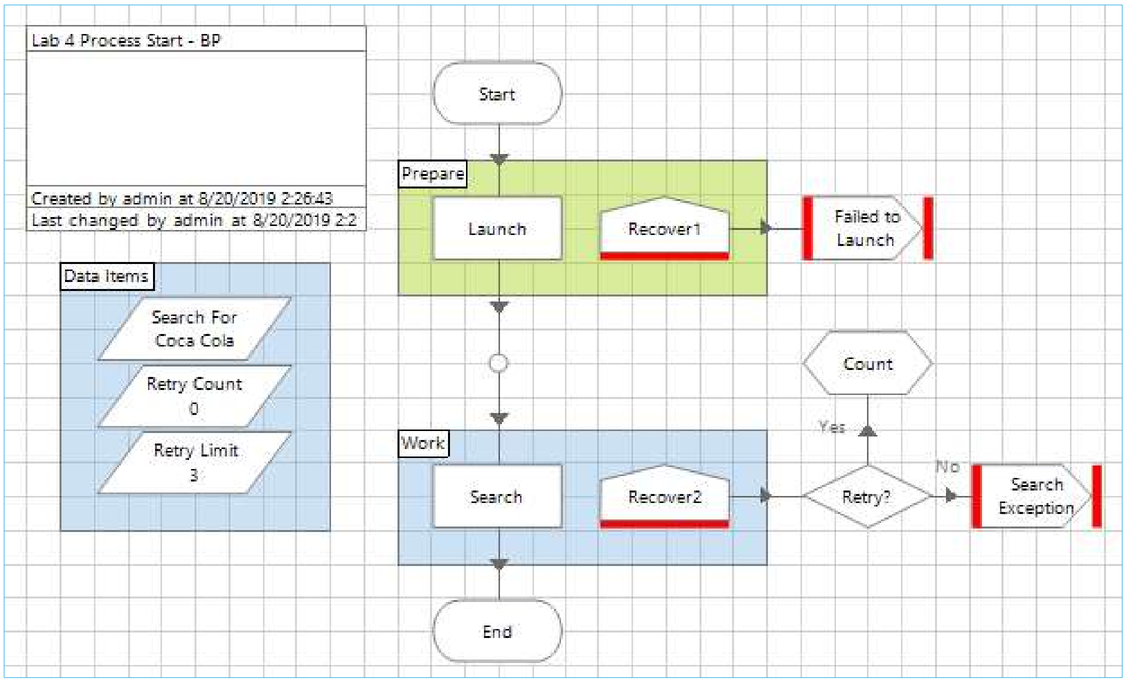
Stage logging: Enabled

Warning threshold: System Default

Number of minutes: 5 (0 to disable)

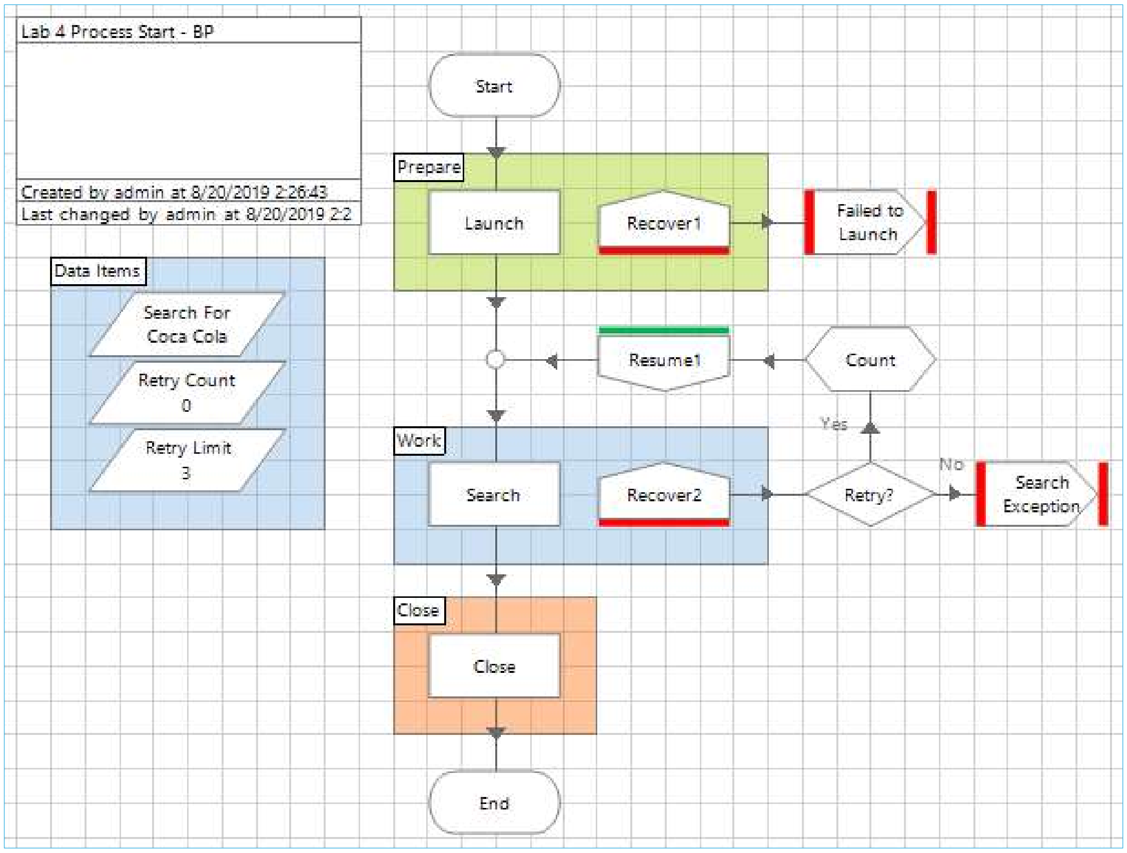
Note: If you are ever unsure how to set up a function, the "Expression Function Builder" (directly below where you enter the Expression) is an excellent tool! Select a function from the list on the left and then use the function builder on the right to enter in the values. When you're finished, press "Paste" and it will be added above as the Expression. The "Validate" button will then allow you to make sure the Expression is valid, and the "Evaluate Expression" button will help you easily test it. With these tools, it becomes easy to create even complex Expression!

9) Link the process together as shown below. Make sure that the "Yes" coming out of "Retry?" points to "Count". If it does not, right click "Retry?" and select "Switch".



10) Add a "Resume" stage between "Count" and the Anchor. This will exit recovery mode and allow the process to continue. Link "Count" to "Resume" and link "Resume" to the anchor.

11) The last piece we must do is the clean-up. Add an action between "Search" and "End". Name the action "Close" and make it call the "Close" action from the Object "Lab 3 Object End - BP". Put a Block around "Close" also called "Close" and change the color to orange.



12) Try it out! When finished, feel free to close both the browser and the process.

Note: You've now completed your first bit of Exception Handling! While traditional programming requires you to attempt to define every negative, Blue Prism exception handling work in the opposite way, capturing anything at all that goes wrong inside of the exception block. Full processes use this inside of loops to prevent a process from ever crashing. Instead of crashing, the process will mark that item as an exception, to be reviewed later, and move on to the next item in the queue. This resiliency is why customers are comfortable running mission critical processes via Blue Prism RPA.