Blue Prism Labs

Lab 9: Exception Handling

Document Revision 1.0

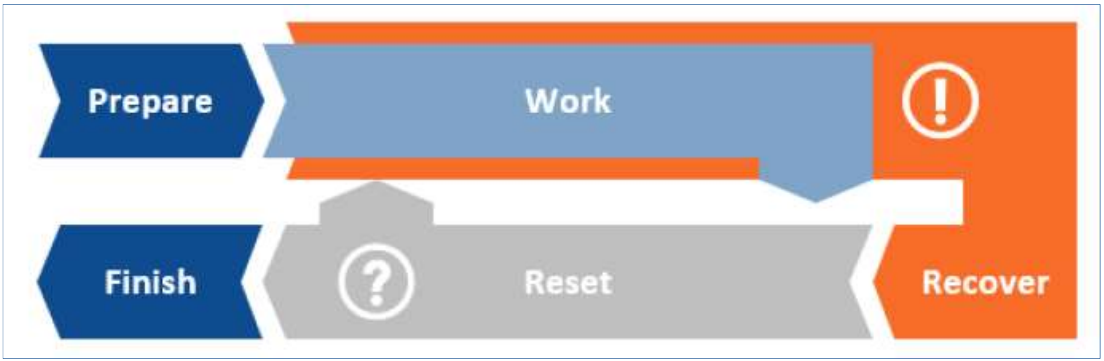
Trademarks and copyrights

Blue Prism Limited, 2 Cinnamon Park, Crab Lane, Warrington, WA2 0XP, United Kingdom
Registered in England: Reg. No. 4260035. Tel: +44 870 879 3000. Web: www.blueprism.com

Contents

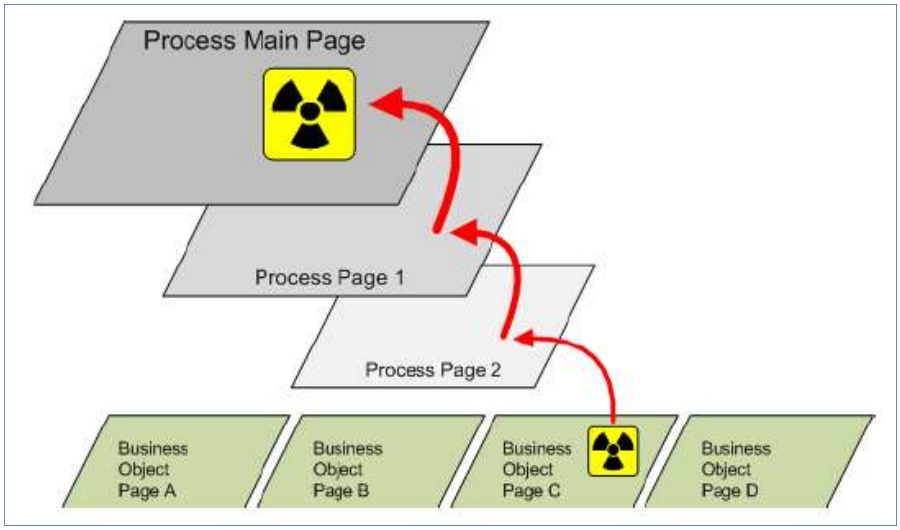
Introduction

A process is unlikely to always follow the "happy path" and resolve every case. In these instances, some cases will need to be identified as exception cases that will either need to be reviewed by a person or submitted to a different Work Queue to follow a different path for resolution. Recovery steps ought to be included to cater for the "unhappy path" logic as in the diagram below. This will make the entire process much more robust and resilient – able to handle multiple scenarios without failing completely.

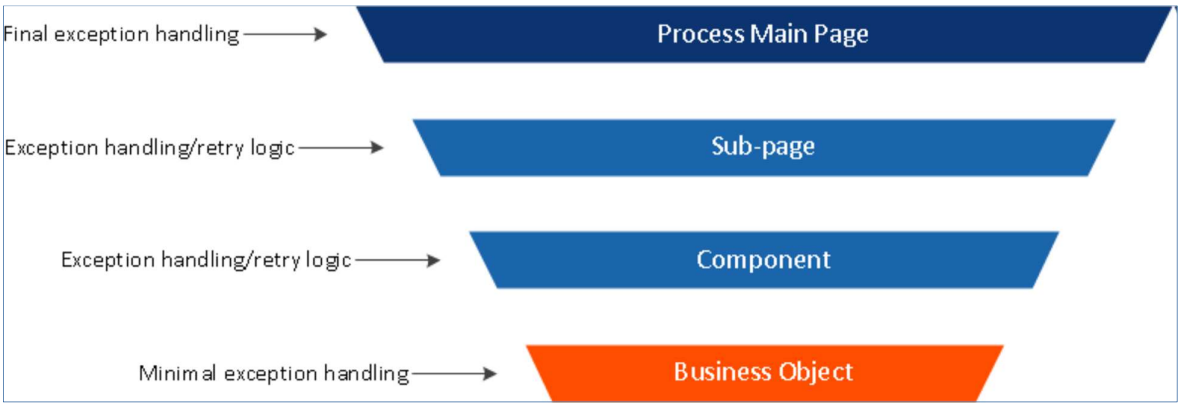


Here, exception handling is employed to make the process recover from unexpected application behavior, so that cases can be set aside for manual investigation or if necessary, reworked by the process. The recovery logic may also need the ability to restart applications for the process to continue working.

Blue Prism allows for designing an automation flow in layers – with objects, processes and pages. Any exceptions encountered and not resolved in any layer will bubble up to the top layer until they are handled.



Exception handling and retry logic tends to be done above the business object layer. This simplifies the business object design and avoids adding complexity that might hinder the reusability of the object. The top layer of a solution is the main page of the parent process, and often this page will use a block as part of a 'final' exception handler that protects lower layers and catches exceptions that bubble up. This serves as a "catch-all" to ensure that the process always completes handling any errors encountered. Additionally, care must be taken with retry logic to ensure that there are no nested retry loops (retry loops in one component as well as in the sub-page that could multiple the total loops. Ex. If layers 2 and 3 both have "retry loops" that retry 3 times, the process could end up performing actions 9 times working through the loops. This can slow down a process and take up excess digital worker time).



Blue Prism also allows for differentiation and customization of exceptions. Two suggestions of exception types to use are system exceptions and business exceptions. System exceptions include any exceptions where problems with an application were encountered. Business exceptions include any exceptions where there were issues with the data supplied for the case to be worked. This allows for different types of exceptions encountered to be handled differently! For reporting, it is a good idea to keep exception types minimal but enough to supply the required information for management of the digital workforce.

In the event of a serious problem with an application it may be better for a process to stop rather than carry on working. For example, if there are 1000 items in the queue and the target application is down, it makes no sense to work all items and have them all marked as exception 'application is unavailable'.

A better solution would be to count consecutive occurrences of an exception and bring the process to a stop when a sensible limit is reached. In other words, if say 3 cases in a row fail for the same System Exception, then treat this as a sign that something is wrong, notify administrator (maybe via email) and stop the process.

Note: Three new subpages have been added to the Lab 9 release: "Mark Item as Complete", "Mark Item as Exception" and "Reset Global Data" which support the tracking of consecutive exception types. These are available as part of the Process Templates and follow best practices.

The "Reset Global Data" page only includes data items that track current exception data and can be reset between cases. The "Mark Item as Complete" includes an action to reset any consecutive exception data items. If an item is completed appropriately, then there is no incidence of consecutive exceptions. The "Mark Item as Exception" will check for "system unavailable" exceptions, consecutive exceptions or business exceptions. These three best practice subpages allow you to maximize your digital workers' time by stopping a process if the same exception repeats itself 3 times.

Lab 9: Exception Handling

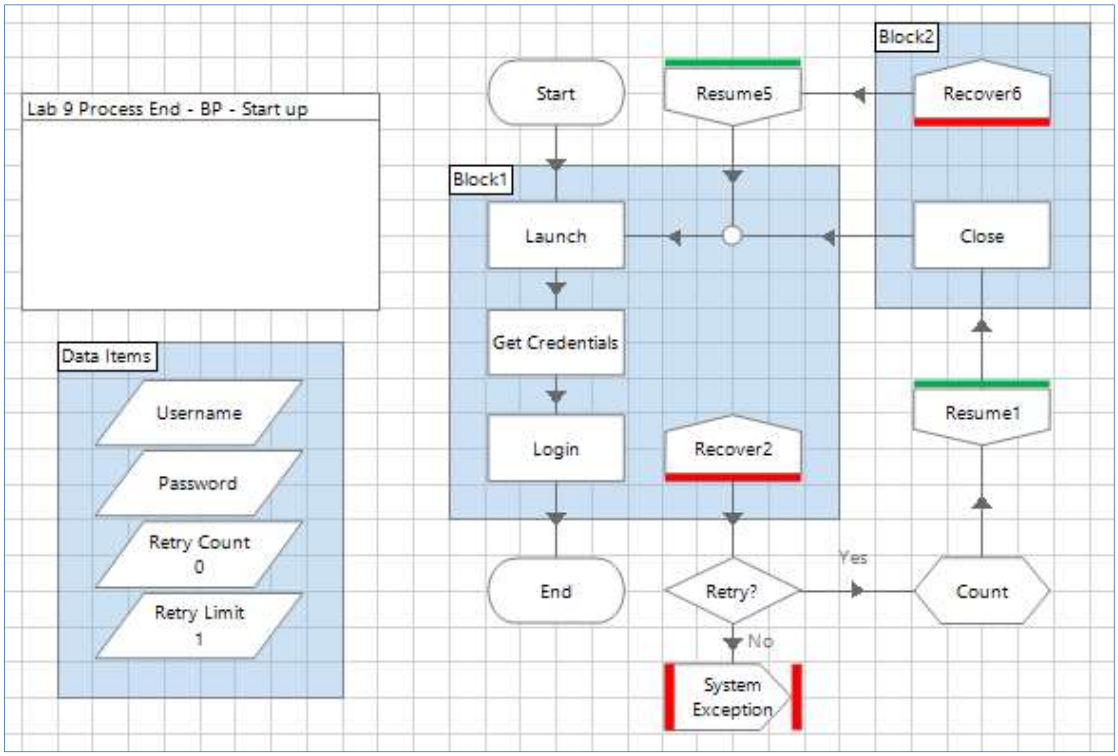
Note: Retry loops and blocks have been added to the "Record in Excel", "Populate Queue" and "Start up" subpages just as was done in Lab 4 (Process Best Practices). Review these pages for the changes.

1) Open the Process called "Lab 9 Process Start - BP". In addition to retry loops, exception handling also may require resetting the applications between the retry loops. The paths for "Populate Queue" and "Record to Excel" already include resetting excel by stopping any excel that is running before performing any work. However, the "Start Up" and "Search" paths may need to have the application closed and relaunched before trying again.

   Start with the "Start Up" subpage. Add an "Action" stage above the "Resume" stage (move the "Resume" stage down to make room). Name this new stage "Close", set the "Business Object" to "Lab 6 Object End - BP", set the "Action" to "Close". Press "OK" to close.

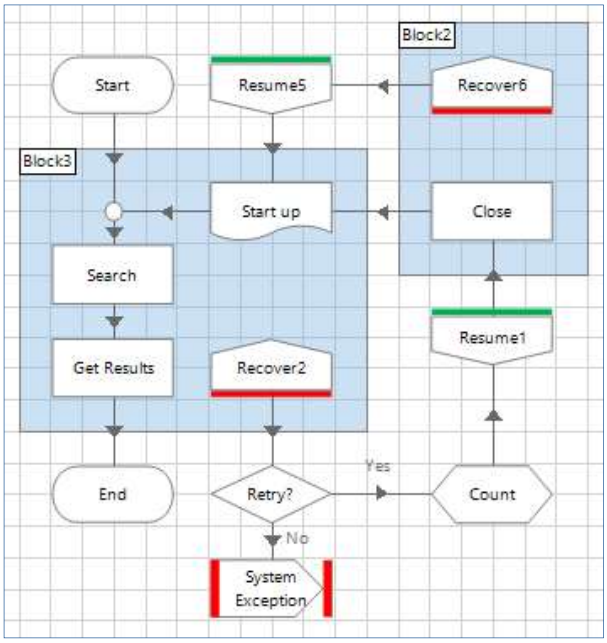
   The "Close" stage should be outside of the block for the recovery path since it is part of the recovery path, not the happy path. However, the "Close" could also experience issues (i.e. if the application is already closed) so it needs its own recovery path that is simple – recover then resume. Place a block around the "Close" stage with an additional "Recover" stage inside the block. Place a "Resume" stage linking from "Recover" to the "Launch" stage (add an anchor to help align blocks like below). Connect all the new stages.

   This will allow the process to continue even if the browser was already closed within the retry attempts.



2) Next, add the same recovery path to the "Search" page with one extra addition. Once we close the application, we need to start it again before we can retry the search, so we need to add the Launch and Login portions by calling the "Start Up" subpage. Add the same "Close", "Recover", "Resume" stages and the block as in the previous step (remember you can copy/paste)!

Also, add a "Page" stage that calls the "Start Up" page. This stage should reside in the original block so that if that stage experiences an error, it will throw an exception before trying to search. The block with the "Close" stage will simply recover and resume which will not work if the "Start Up" stage fails. See below for the layout.

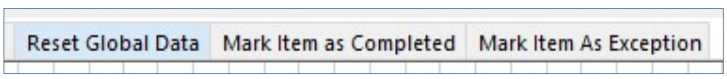


3) Now that we have our exception handling all set in our subpages, we can move up to the "Main Page" and decide how we want to deal with the exceptions that bubble up from subpages or objects.

One place where the exceptions will bubble up is in the "Work" block which will catch exceptions from the "Search" and "Record in Excel" subpages. Currently, there is no exception handling defined so the whole process will terminate when the first case encounters an error. A better path would be to catch the error with a "Recover" stage, mark the case with the exception information and move on to the next case.

This is a good method for dealing with each case. However, what if the same exception occurs over and over for every case (as in the case of an application outage)? We may want to track the exceptions for consecutive cases and if the error occurs multiple times, we can stop the process from continuing to work the queue.

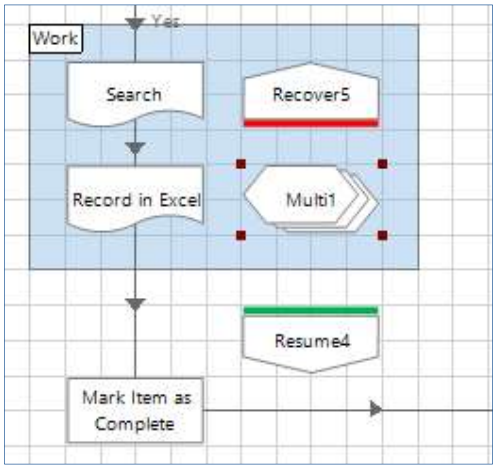
This logic is built in to the subpage "Mark Item as Exception". The "Mark Item as Complete" adds a step to reset consecutive data items to support this logic as well. This path also makes use of global data items (data items that are available outside of the page they are configured). "Reset Global Data" is the page to use to reset variables before starting on the next case. Review each of these pages to gain a better understanding of their functions.



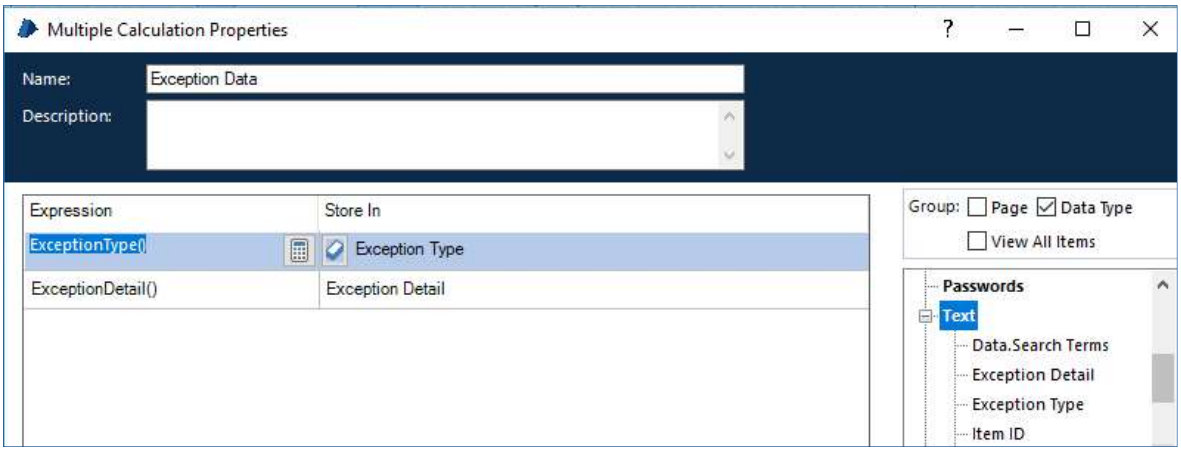
Go back to the "Main Page" to build the logic that will follow the path described in the previous step.

First, add a "Recover" stage inside the "Work" block and a "Resume" stage beneath it outside of the block.

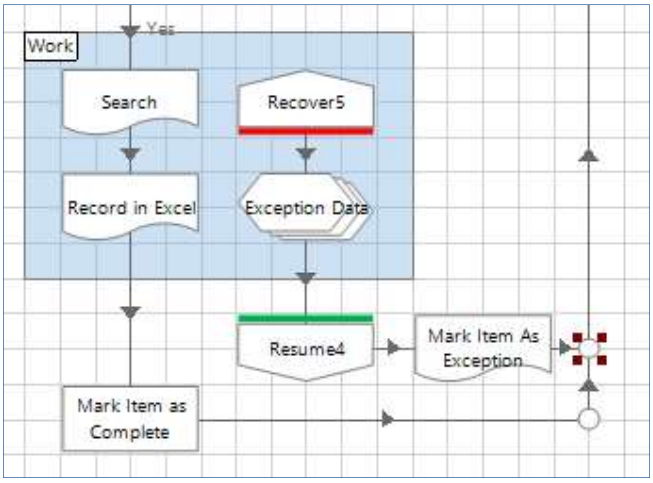
Exception Type and Exception Detail can only be captured in between a "Recover" and "Resume" stage (once the "Resume" occurs, exception detail is lost unless it is captured in a data item). Now add a "Multi Calc" stage in between the "Recover" and "Resume" to capture this exception info. (This can be inside the "Work" block as space allows; drag the "Mark Item as Complete").



Name the "Multi Calc" stage "Exception Data" and add 1 additional row to the one existing in the main box so that there are 2 rows. In the first row, type "ExceptionType()" (or use the expression editor/calc icon to set it). Drag the "Exception Type" data item from the panel on the right to the "Store In" column. For the 2nd row, type "ExceptionDetail()" and store in "Exception Detail" data item.

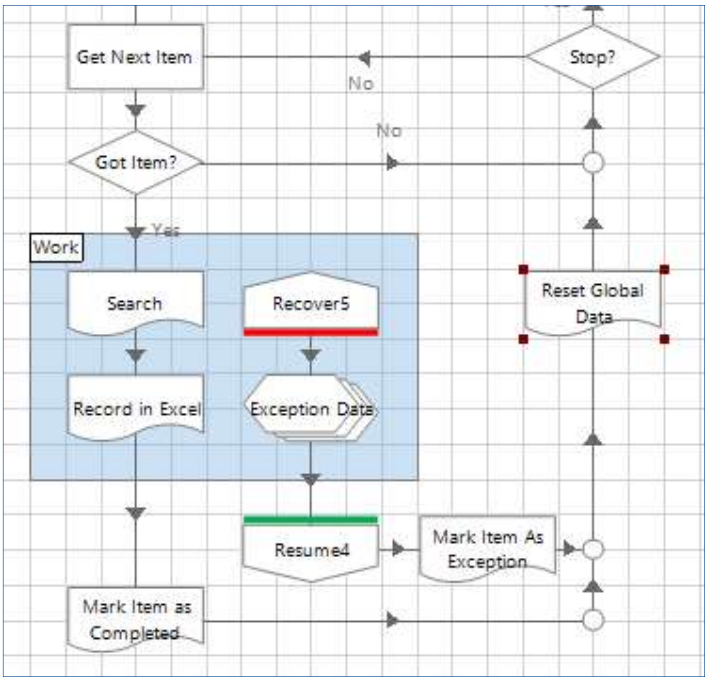


4) Next, add a "Page" stage to the right of the "Resume" stage which references the existing page "Mark Item as Exception". Link the new recovery path together and into the main cycle (adding an anchor point to do so). Now, if any exceptions bubble up from the subpages or objects, they will be caught and used to mark the case as an exception. This digital worker can then move on to the next case seamlessly!

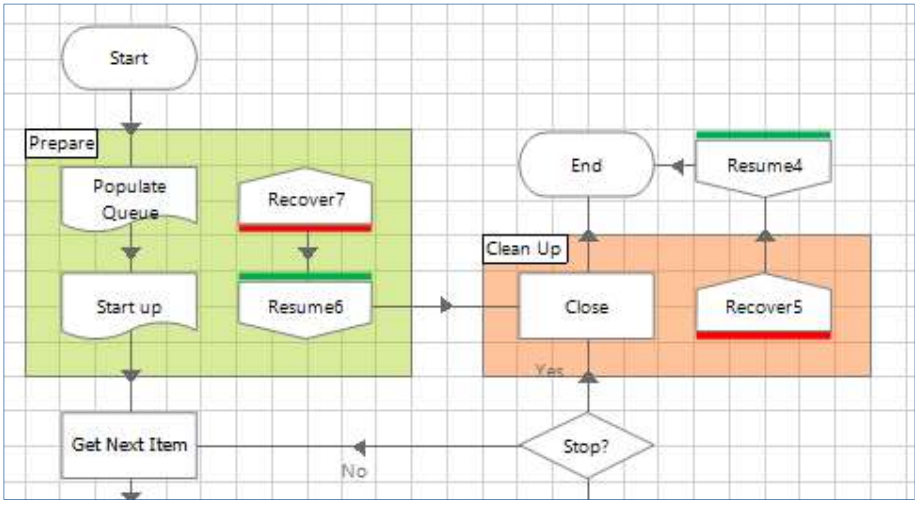


5) Previously, your "Mark Item as Complete" on your main page is an action using the built in functionality with work queues. Note that this action has already been removed. The "Mark Item as Completed" subpage has an extra step to reset consecutive variables. We'll use that page instead.

6) Add another "Page" stage in the path between anchor points up going up toward the "Stop?" decision stage. Reference the page "Reset Global Data" and link everything together to complete the full cycle.



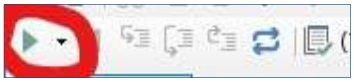
7) Finally, we need two more "Recovery" and "Resume" paths for the "Prepare" and "Clean up" blocks. The "Prepare" recovery path should flow to the "Close" stage because if the queue can't be populated or the applications can't be started, then no work needs to be done. The "Clean up" recovery path should flow directly to the "End" stage.



8) Now you can run your process from start to finish! However, this time, close the browser in the middle of the process and watch the error handling paths that it follows. You can test this in each page or in each block by using the "Pause button (if you used the "Go" button to run) or "Step"/"Step over" buttons

Click the "Reset" button in the upper left:

9) Click the "Go" button in the upper left or the "Step" or "Step over" buttons to step through each stage:



10) Watch your process run and check to see if the error handling is working as expected! When finished, feel free to close both the browser and the process.

Note: In this lab, you learned about the various features available with exception handling. In addition, you learned about how exceptions work together through the various layers of a process and how/where to place different types of recovery steps. These features will allow the digital workers to complete many more processes than the simple "happy path" and be much resilient and robust!