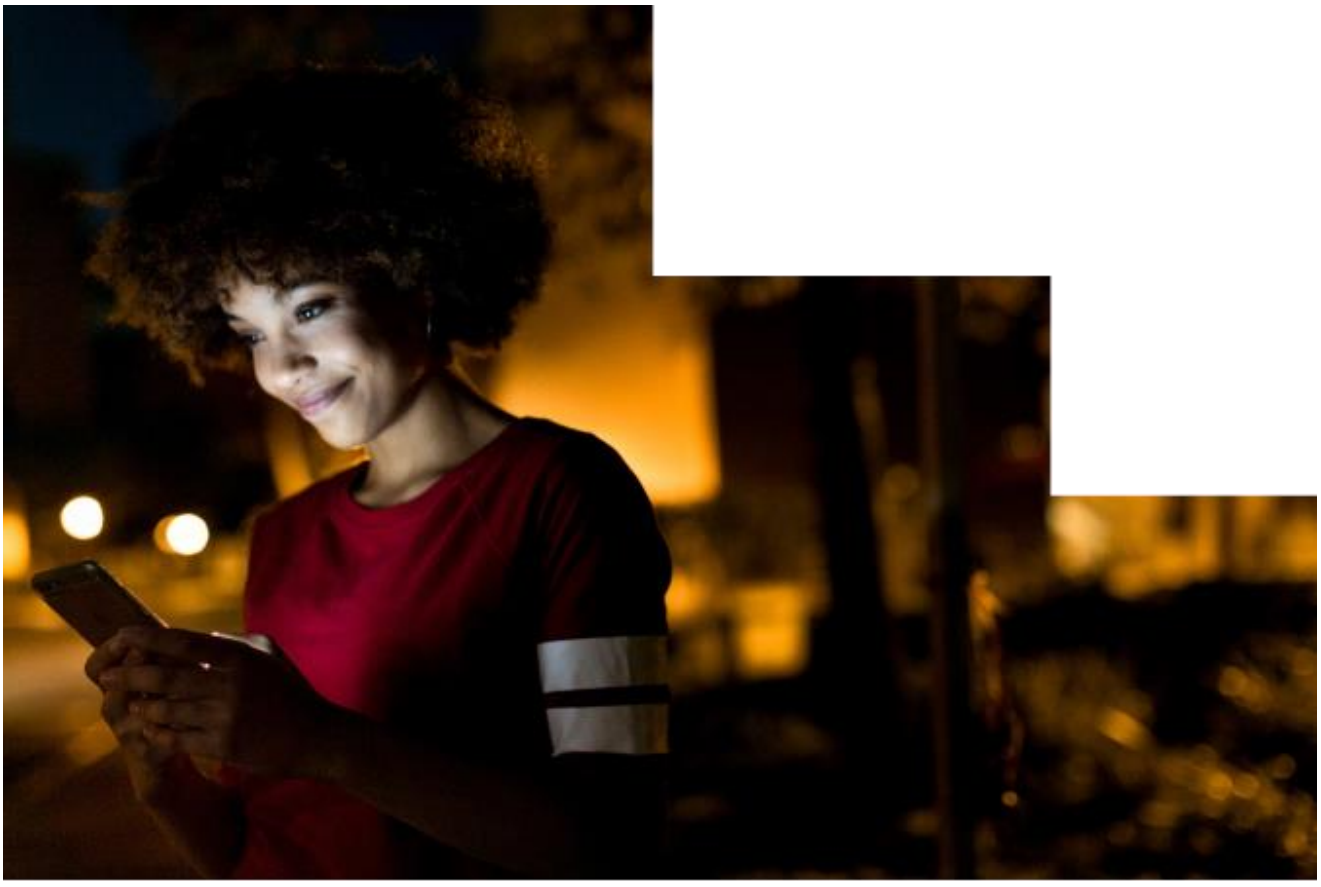
Microsoft

Robotic Process Automation in a Day

Lab 11 – Error handling (Optional)

30 mins

June 2021



This document is provided “as-is.” Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. Some examples are fictitious and are for illustration only. No real association is intended or inferred. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal reference purposes.

© 2021 Microsoft Corporation. All rights reserved.

Lab Overview

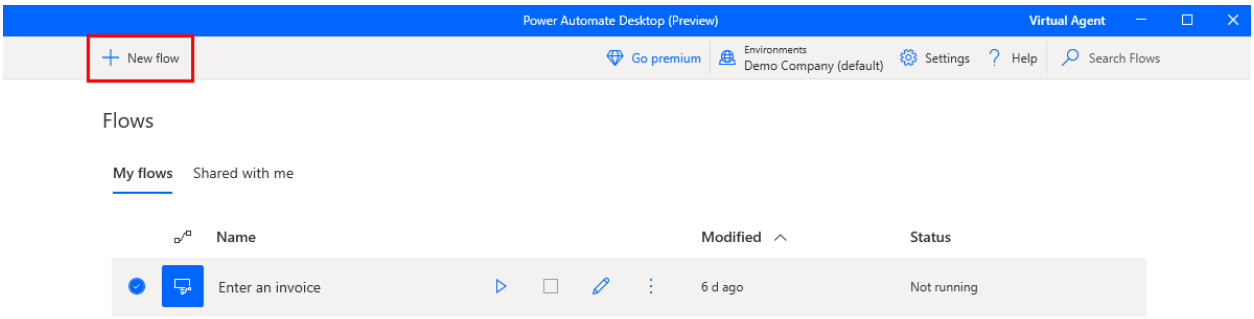
Power Automate provides error handling both at desktop flow and cloud flow level.

On block error action on the desktop flow acts like the **Scope** action in cloud flows. When an exception occurs in a block, we can configure it to perform subsequent actions like call a Subflow. This ensures all the exceptions are caught and handled gracefully. As a result, your automations become much more resilient and robust.

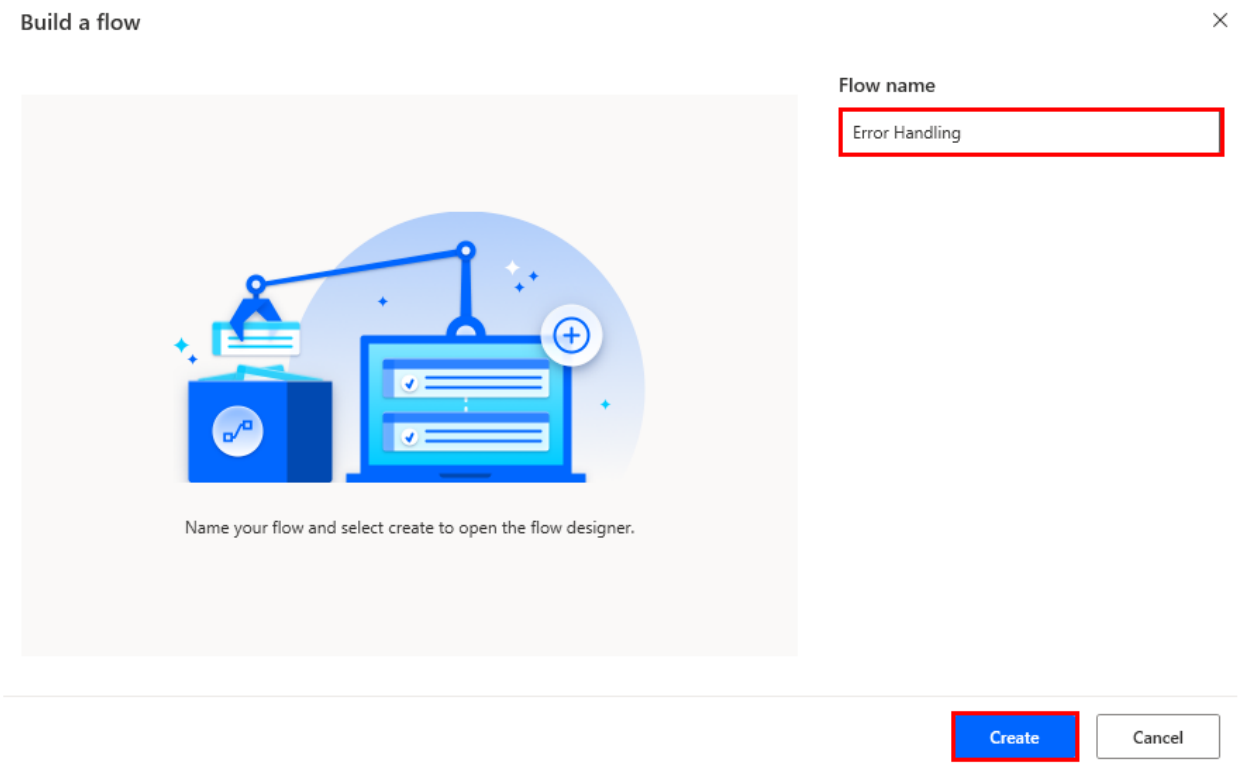
This lab we will walk you through how to do Error handling in desktop flow

Lab steps

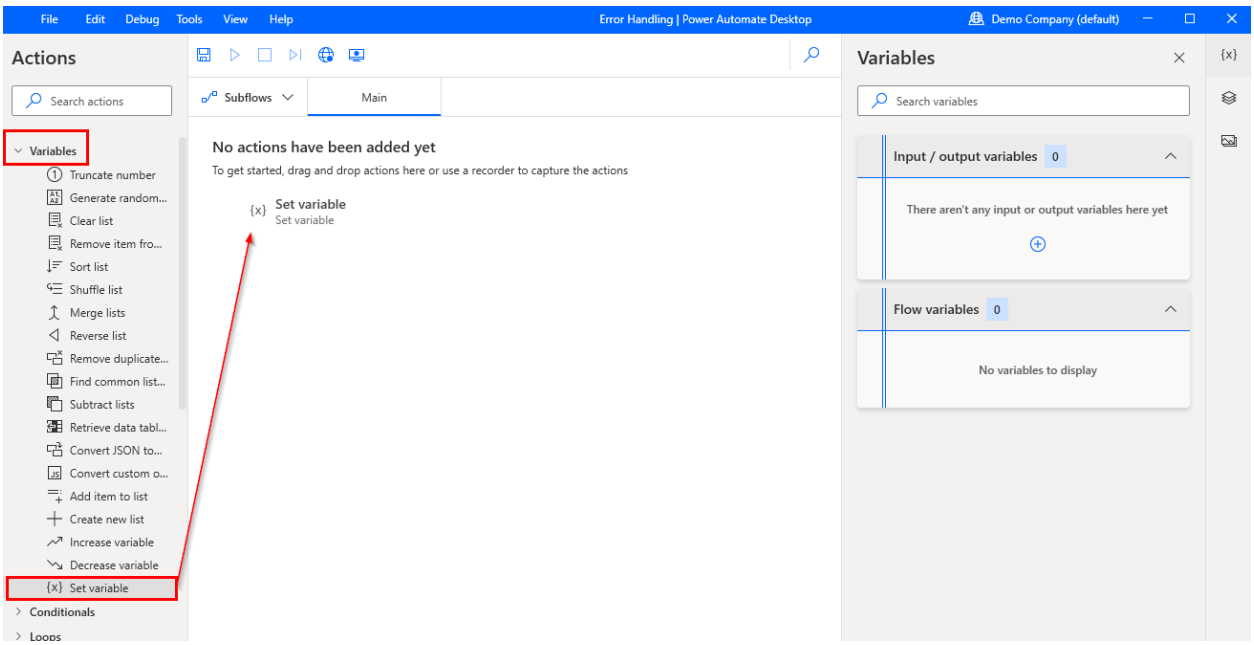
1. Let's first create a desktop flow to learn some error handling method. Open Power Automate desktop app and click + **New flow**



2. Name the flow **Error Handling** then click **Create.**

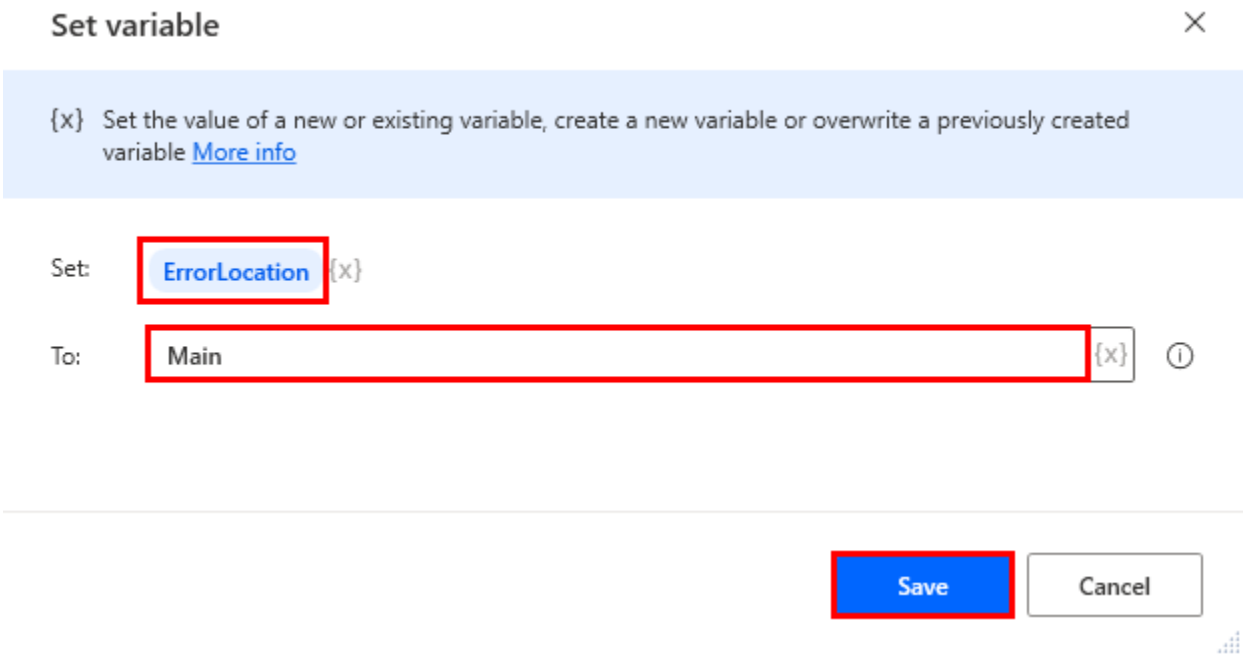


3. Under **Variables**, drag **Set variable** from the left drop down list to the right.

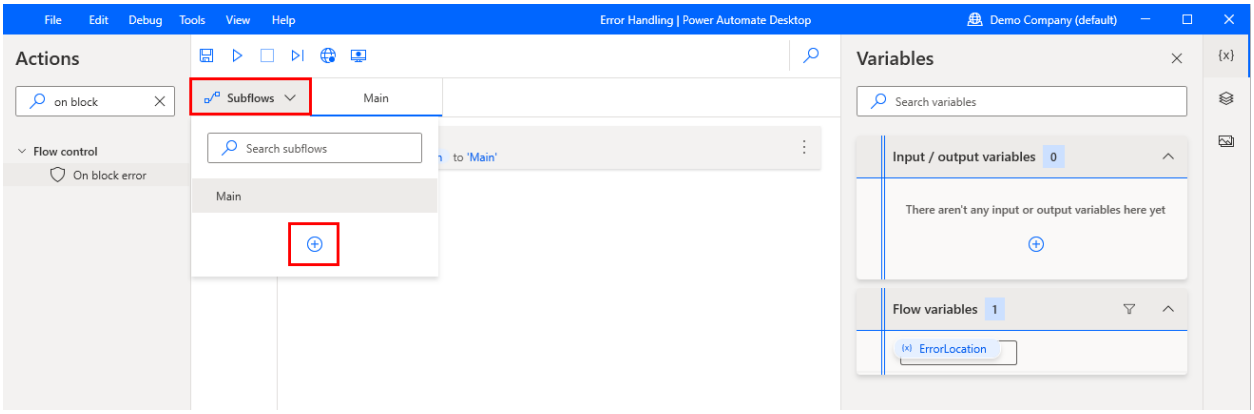


4. For the practice purpose, we will now set an ErrorLocation variable. Use below information to fill out **Set variable**:
    - **Set: %ErrorLocation%**
    - **To: Main**

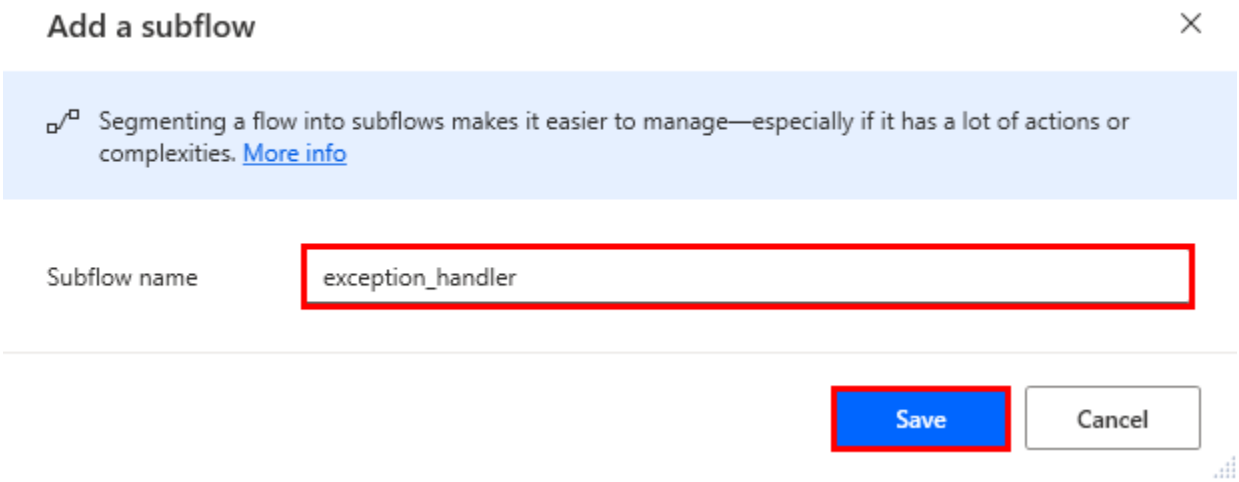
    Then click **Save.**



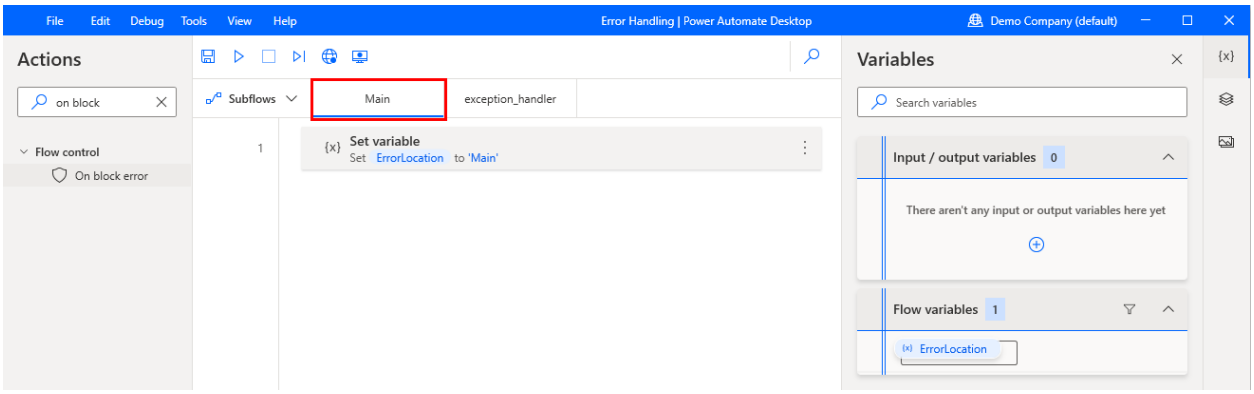
5. Create a new Subflow.



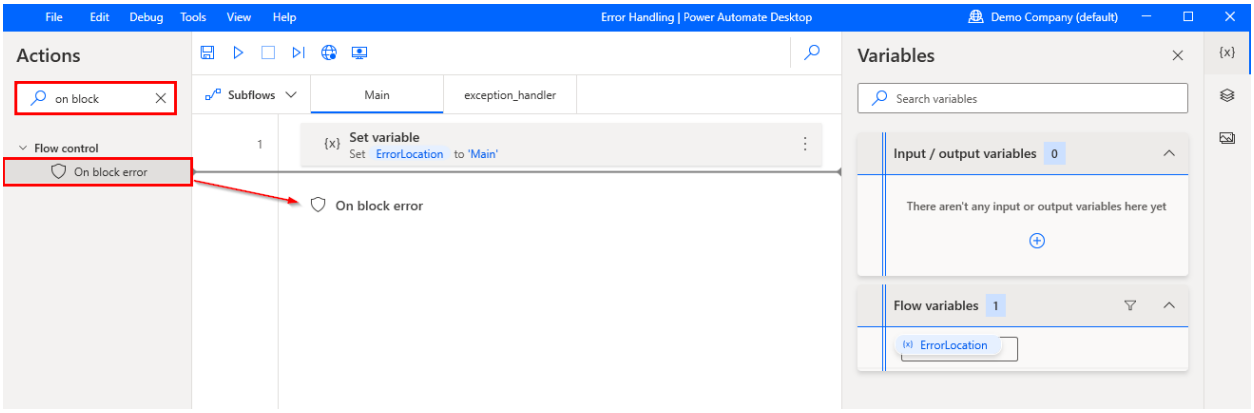
6. Name the subflow **exception_handler** then click **Save.**



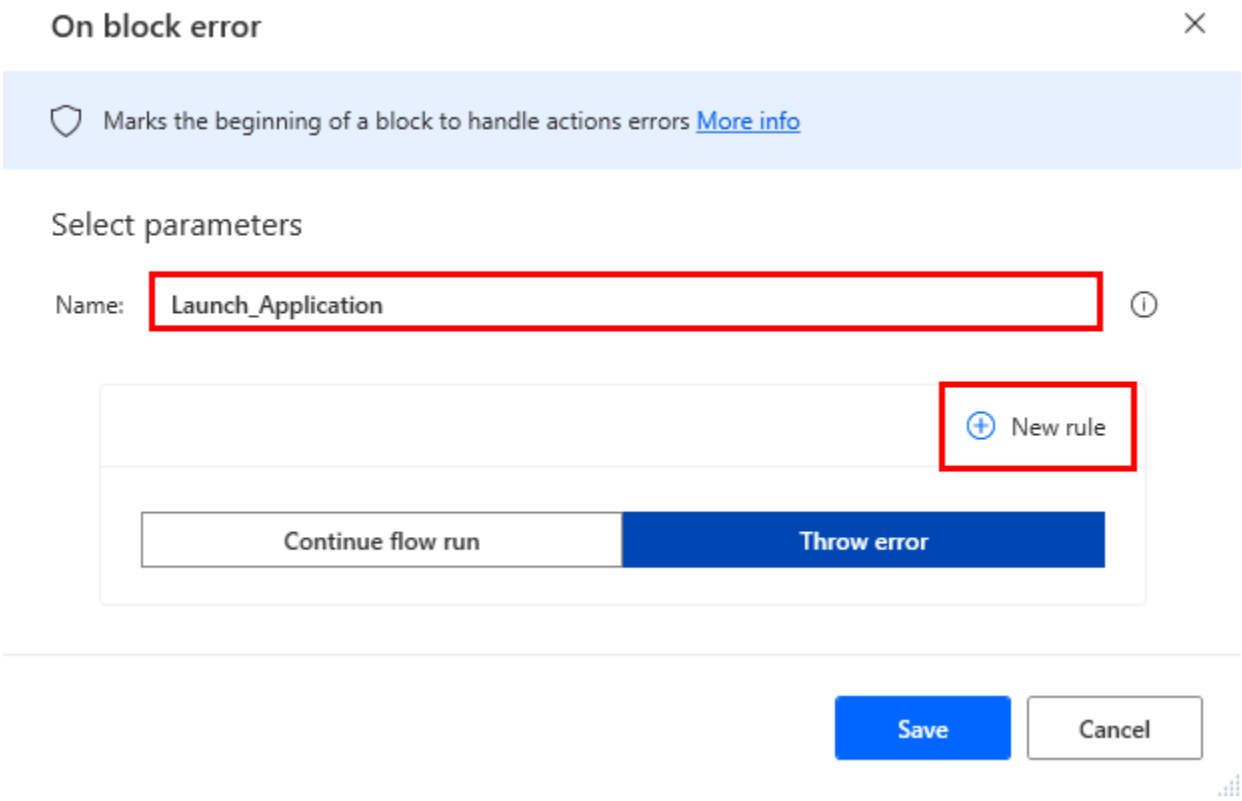
7. Go back to Main flow.



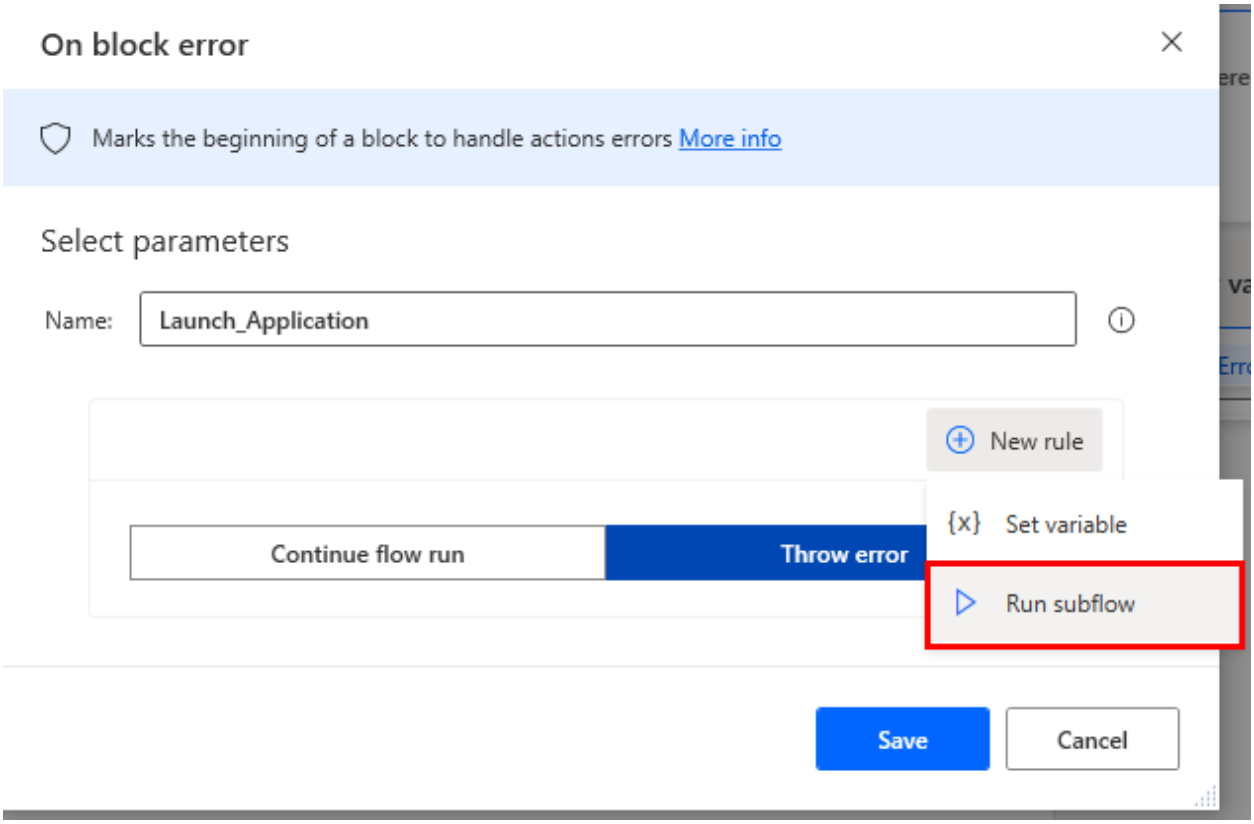
8. Now we will create a new error handling rule. Search **on block** on the search bar and drag the **On block error** action to the canvas list under the first step.



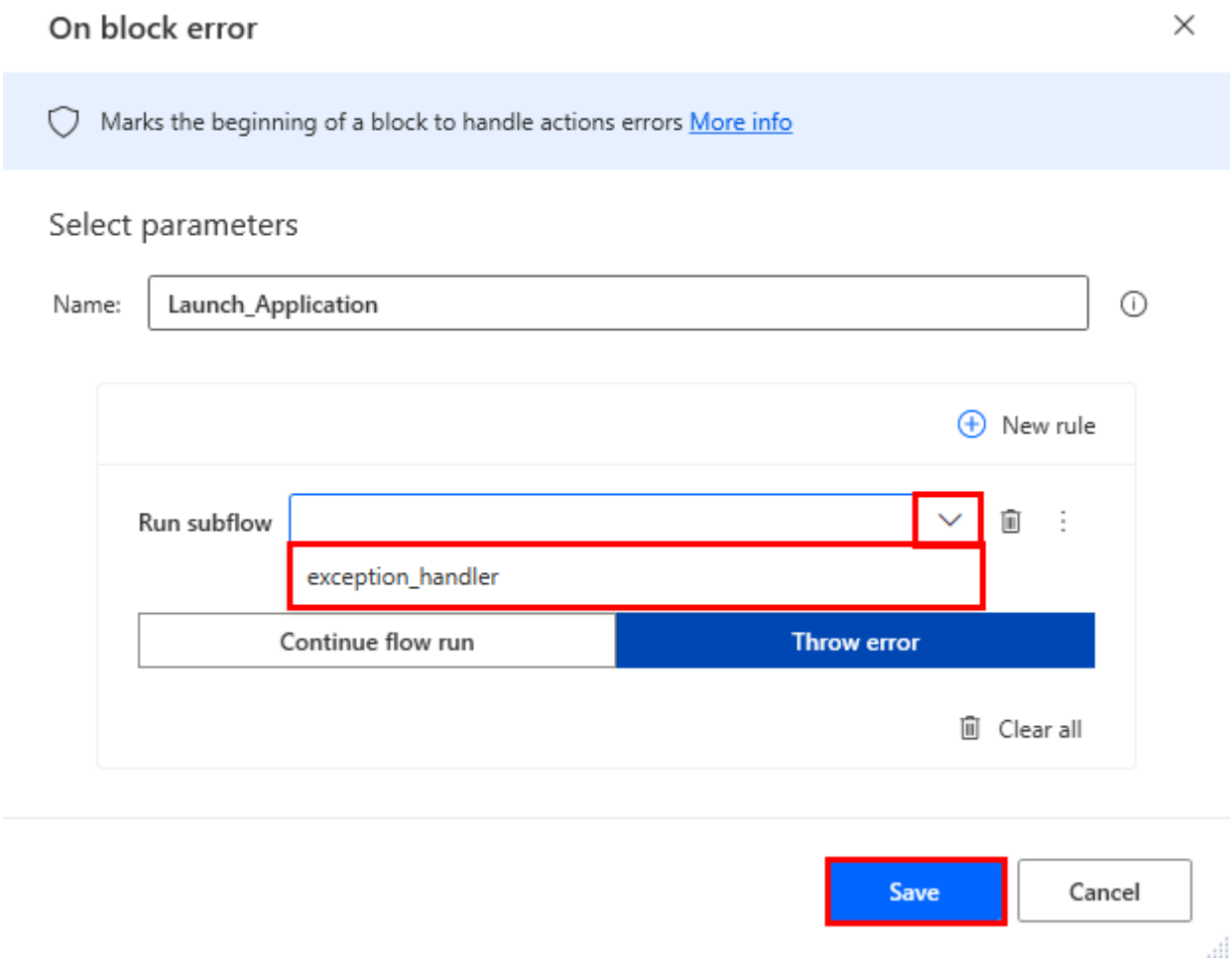
9. Name it **Launch_Application** the click **New rule.**



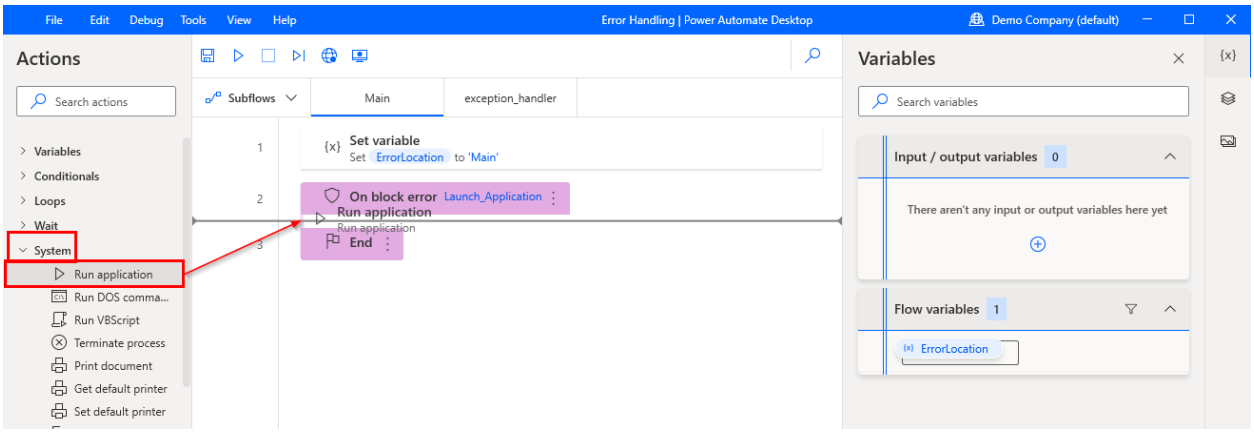
10. If an error occurrs within this block, we want to call our exception handler subflow. We can achieve this by creating a **New rule** that will **Run subflow** option.



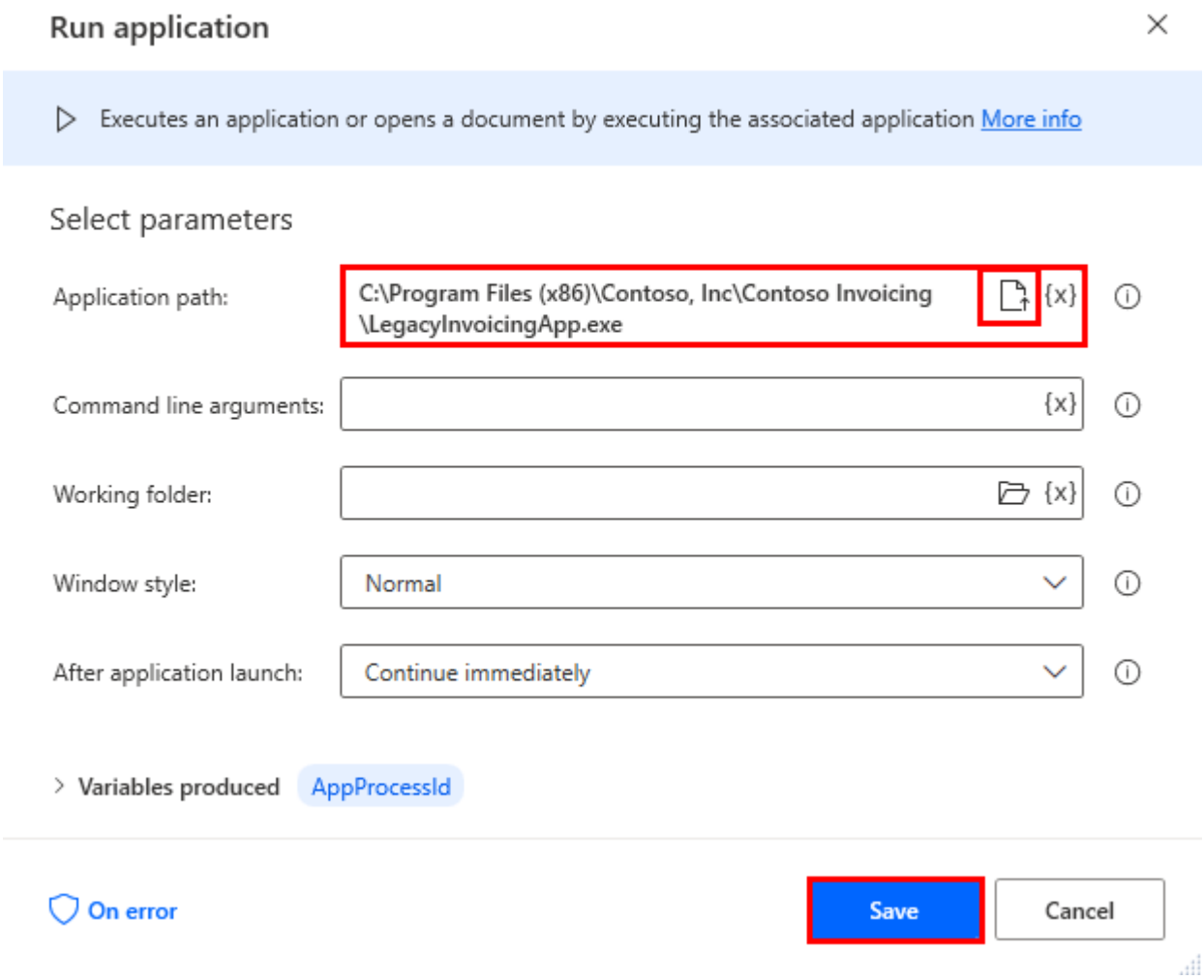
11. Select **exception_handler** subflow and click **Save**. Now if an error occurs, we will run the exception_handler subflow.



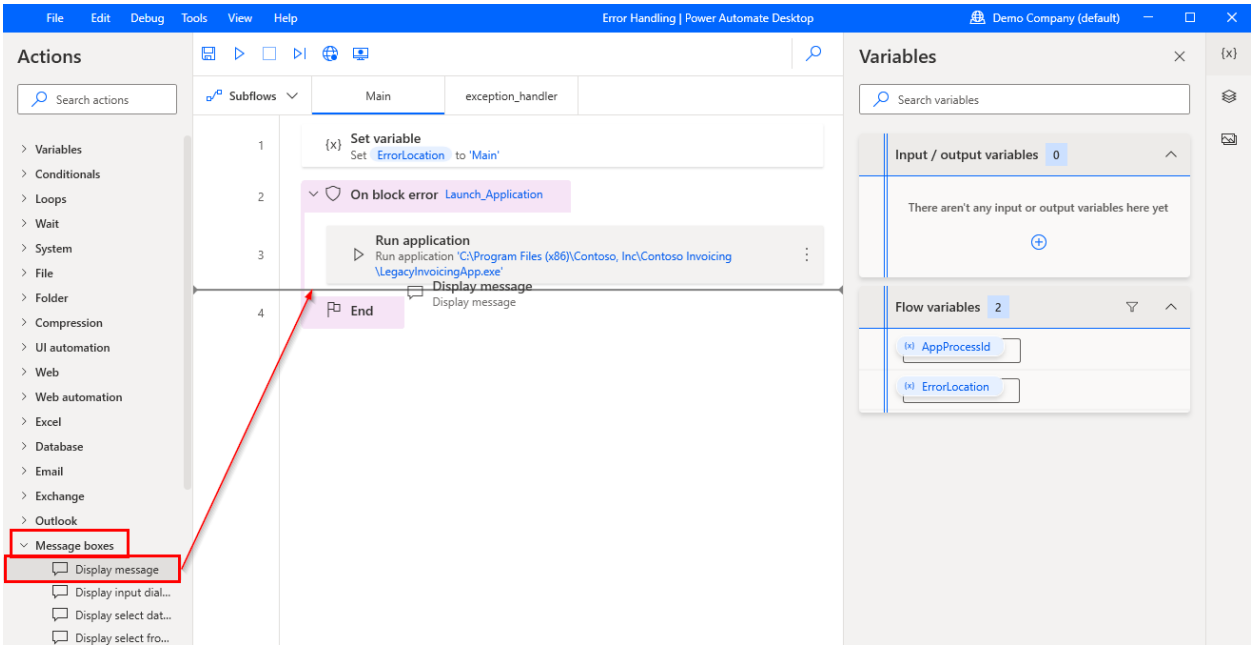
12. Under **System**, drag **Run application** to the canvas list under step 2.



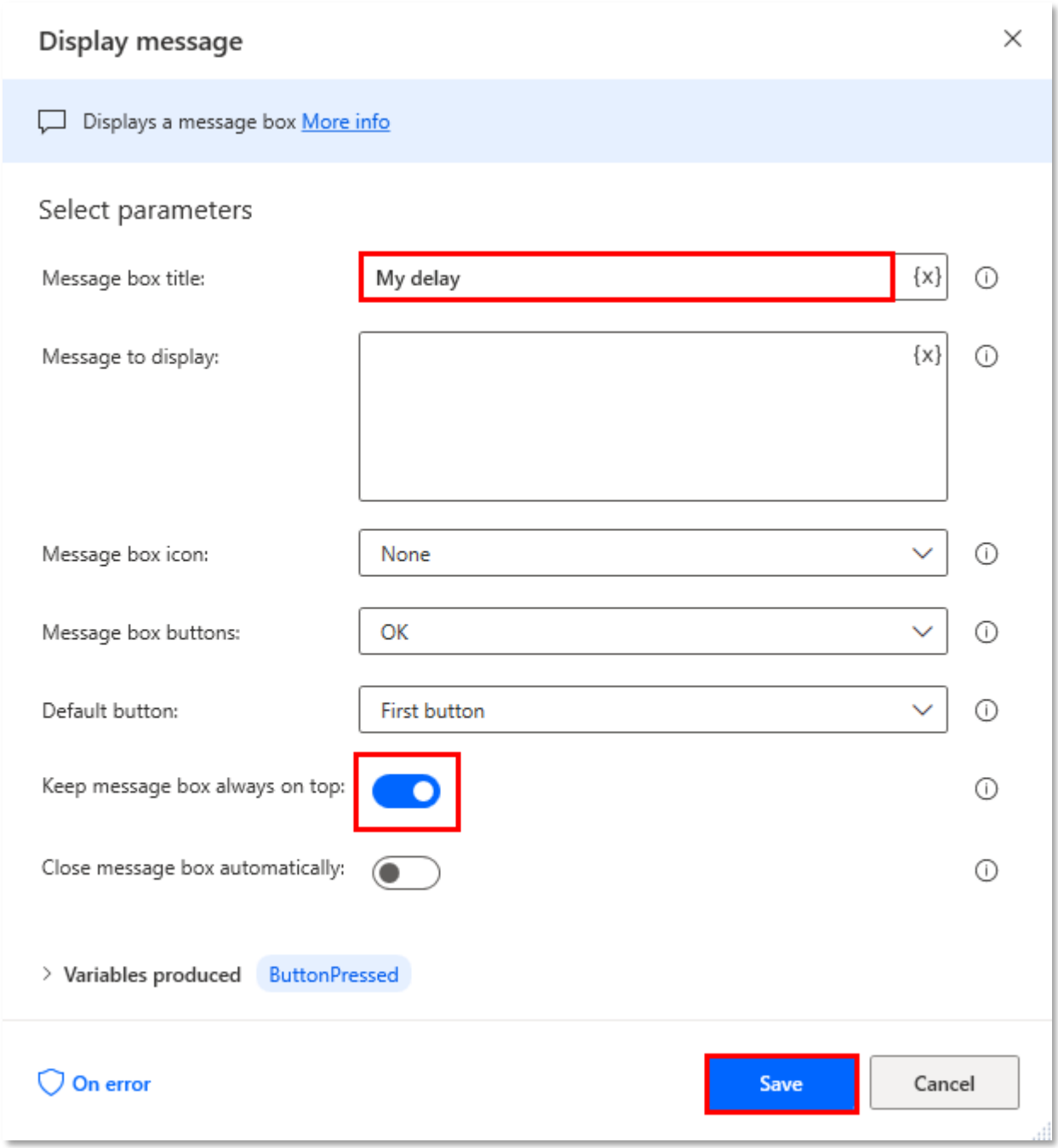
13. Enter the location path of Contoso Invoicing app under **Application path**. Then click **Save.**



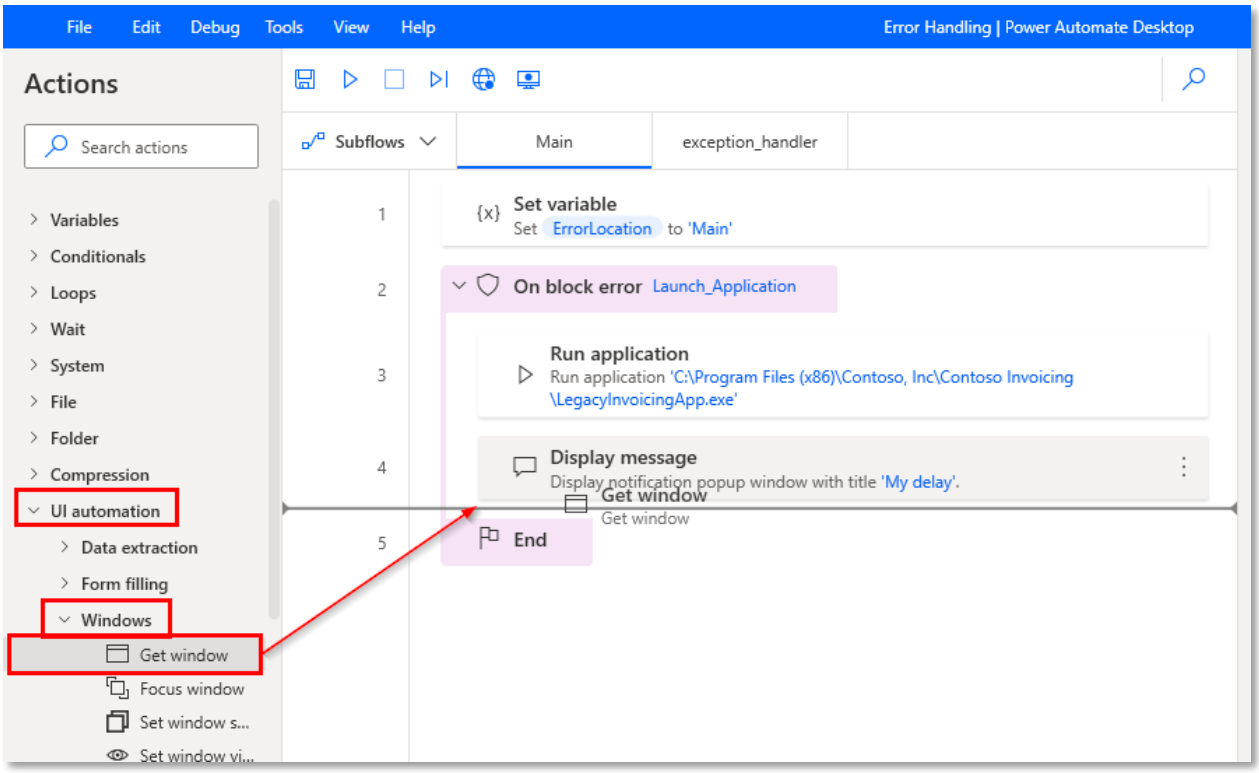
14. To create an error, so that our exception handler will be called for this lab, we will manually close the application so the following step cannot locate the window anymore. To do this we will use a message box step to allow us some delay time to close the application manually. Under **Message boxes**, drag **Display message** to the canvas list under step 3.



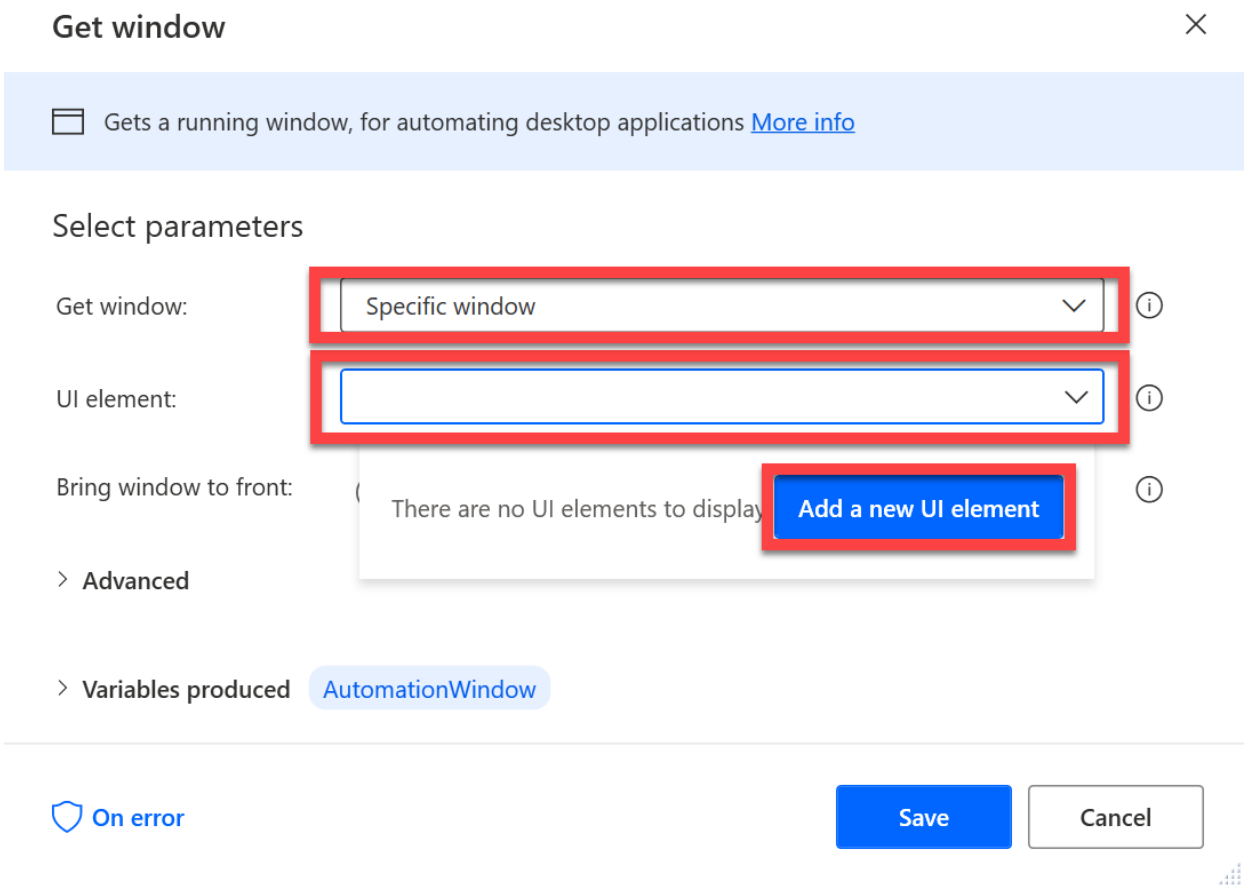
15. Enter **My delay** in **Message box title**. Turn on **Keep message box always on top**. Then click **Save.**



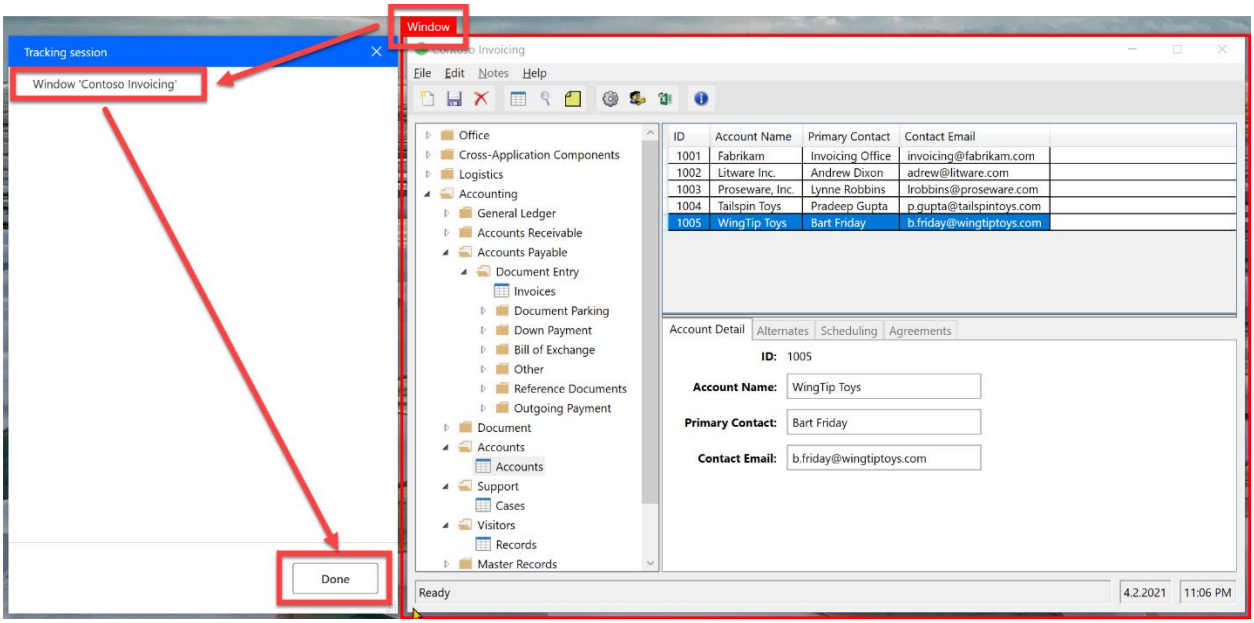
16. Now let's create a **Get window** step that will fail because we manually close the application. Open Contoso Invoicing App. Next expand **UI Automation**, under **Windows**, drag **Get window** to the canvas list under step 4.



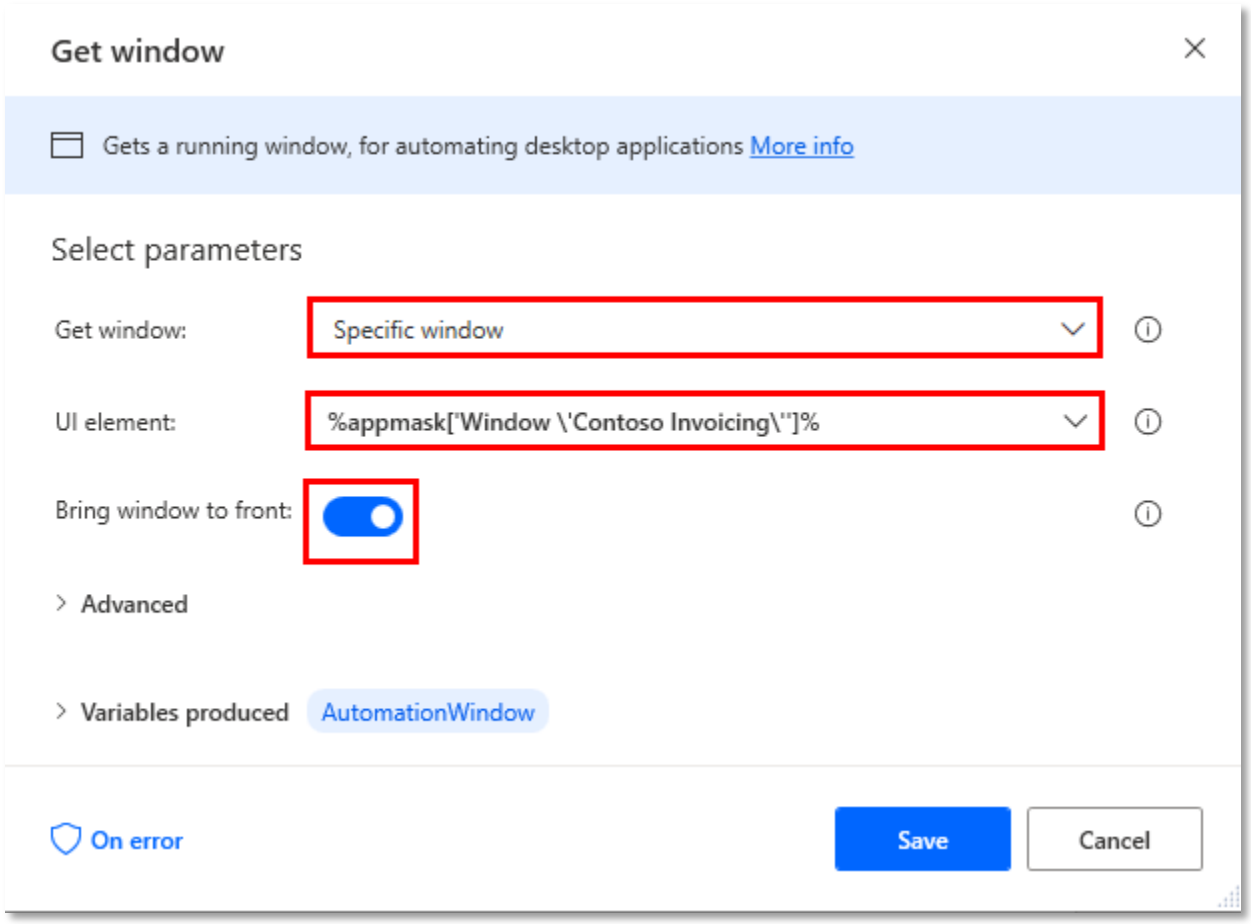
17. Select **Specific window** in **Get window** dropdown list. Select **UI element** in the **UI element dropdown list**. Because this process hasn't recorded any UI elements yet, it will display text "There are no UI elements to display" together with the button to add new one. Click the button.



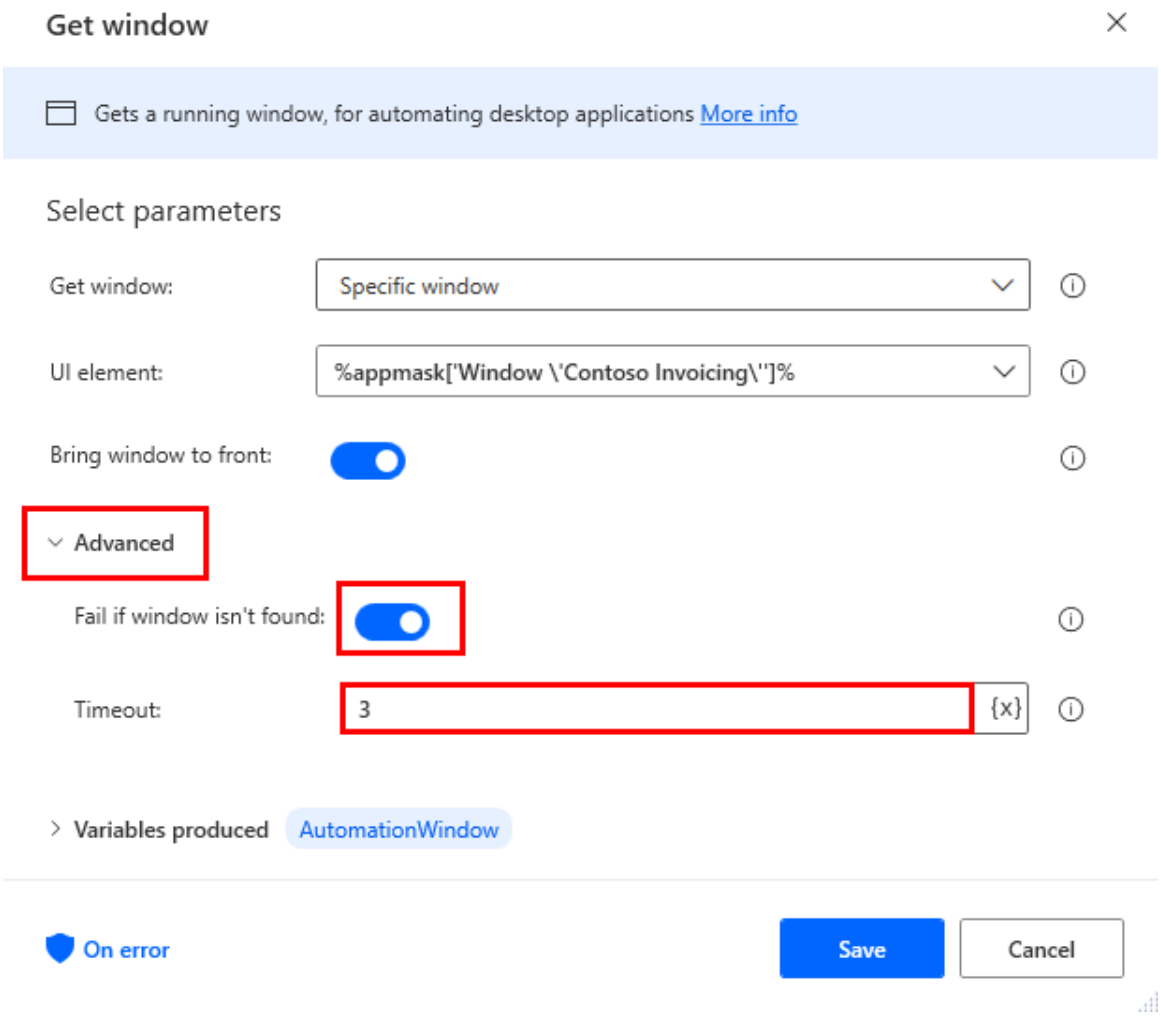
18. Then once the **Tracking session** window appears, hover mouse cursor over the **Contoso Invoicing** app, so that the **"Window" selector** appears (application window is surrounded with red border) and then, while having **"Control" (Ctrl) keyboard key pressed**, click **left mouse button**, what will add new UI element to the list. Then click "Done" button.



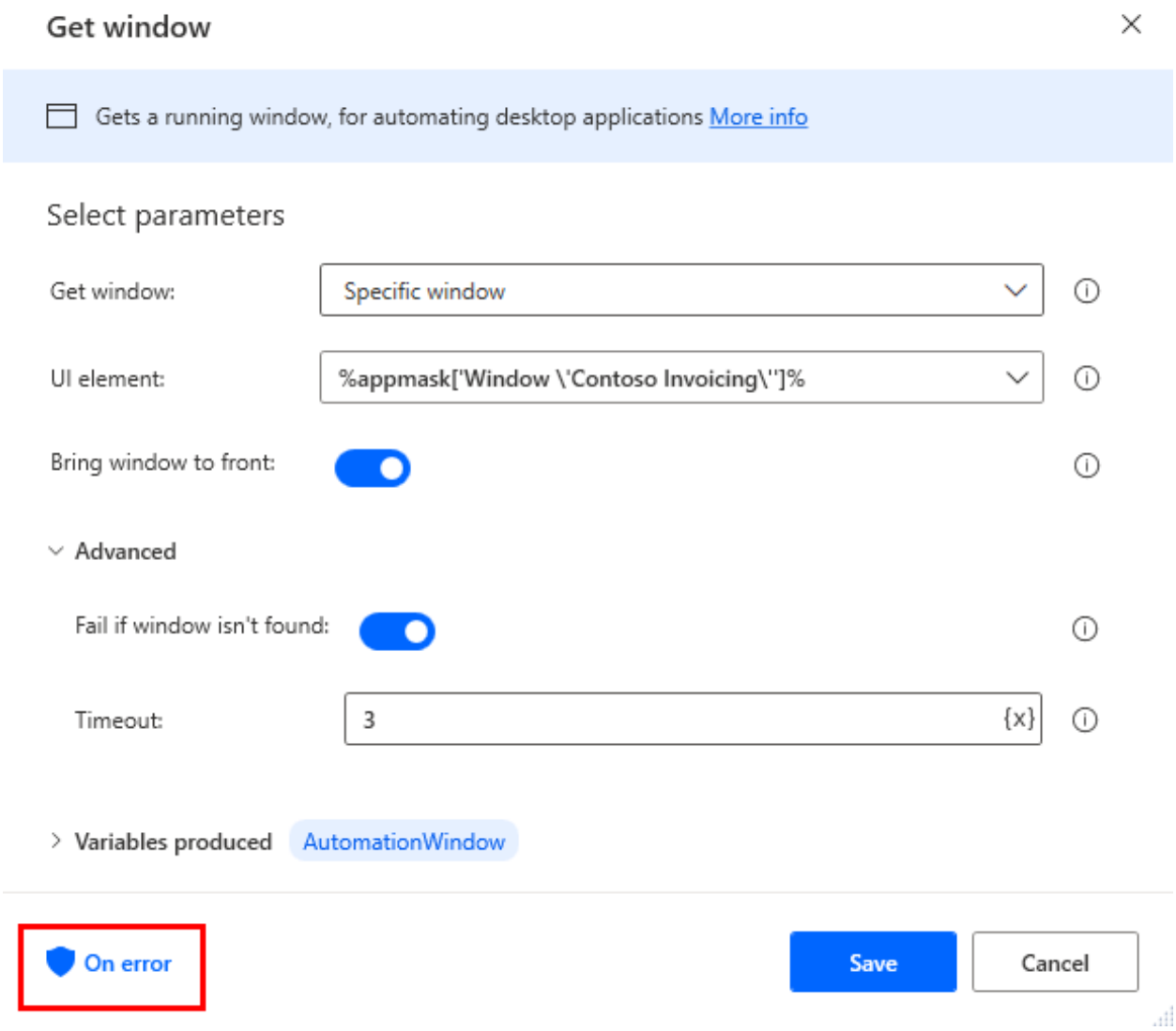
19. Now select newly added UI element. Also, turn on **Bring window to front** option.



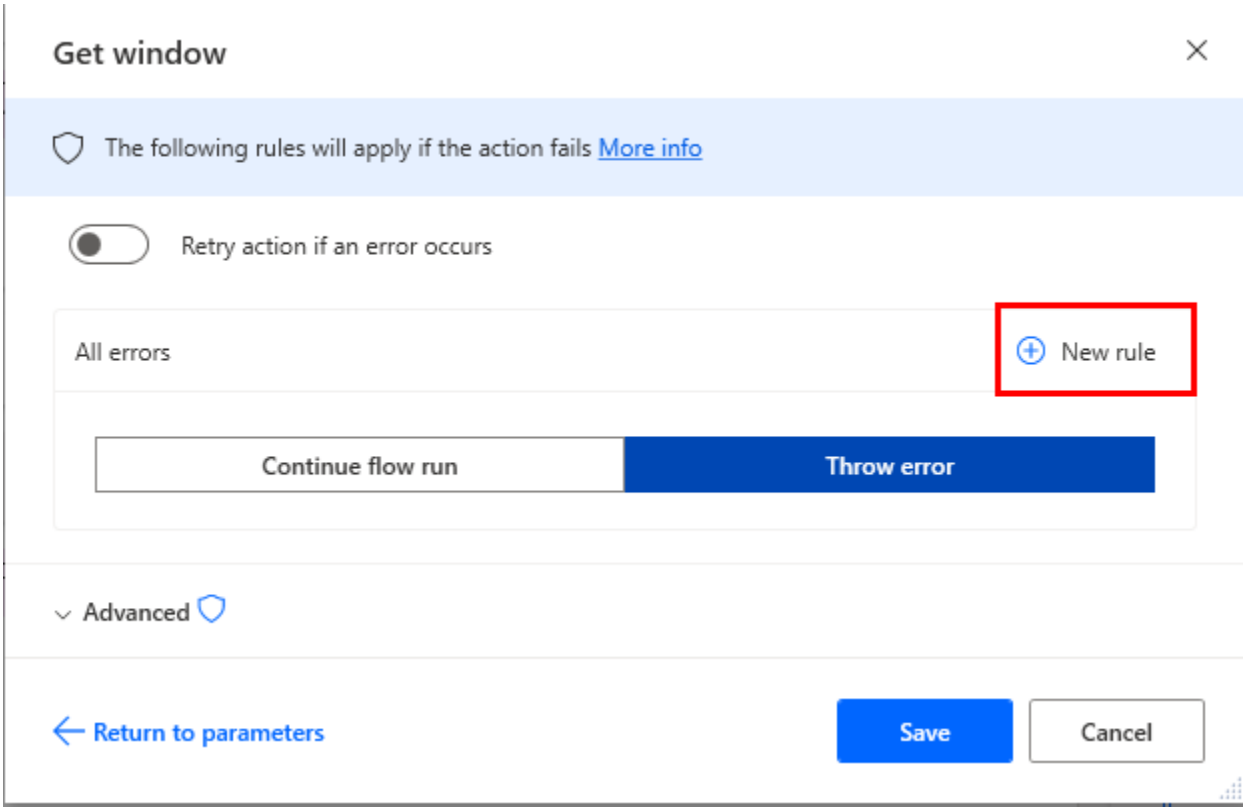
20. Expand **Advanced**. Turn on **Fail if window isn't found** option and set **Timeout** as **3** seconds.



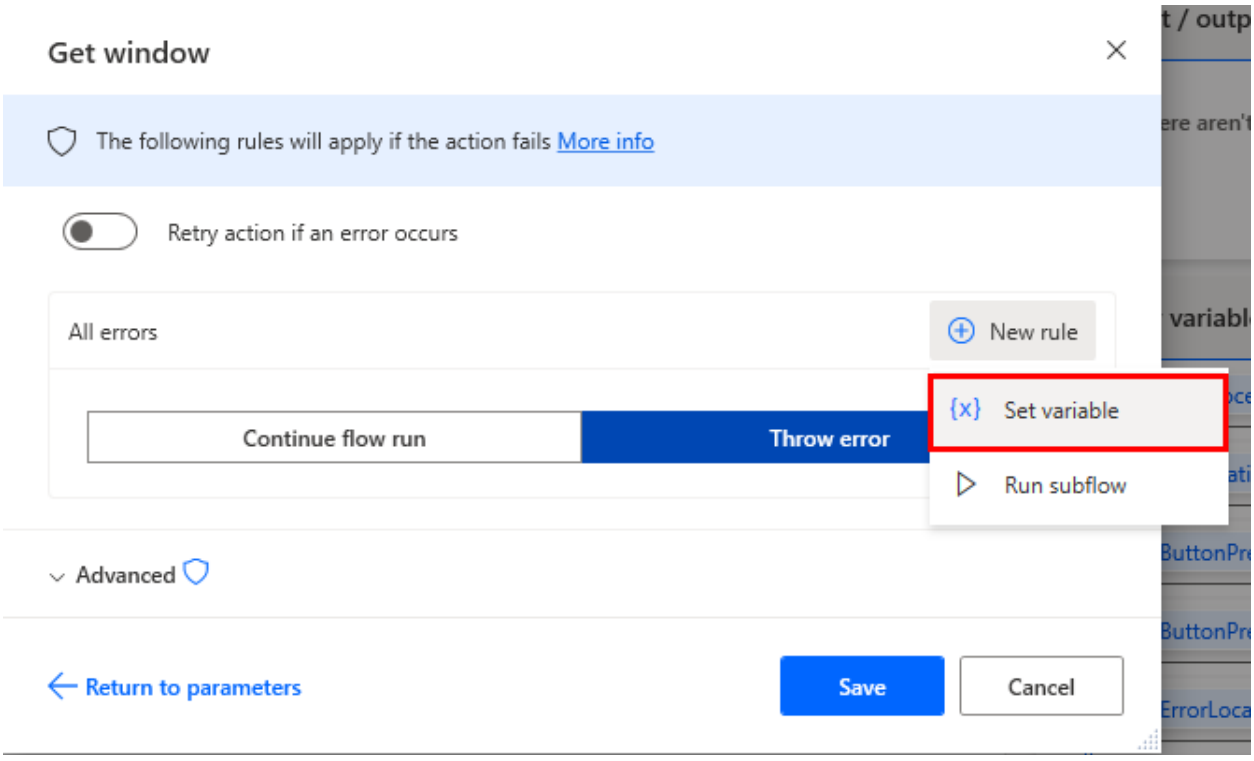
21. To show that this is the step where the automation fails, we will set the **ErrorLocation** variable here. To do that, click **On error** button.



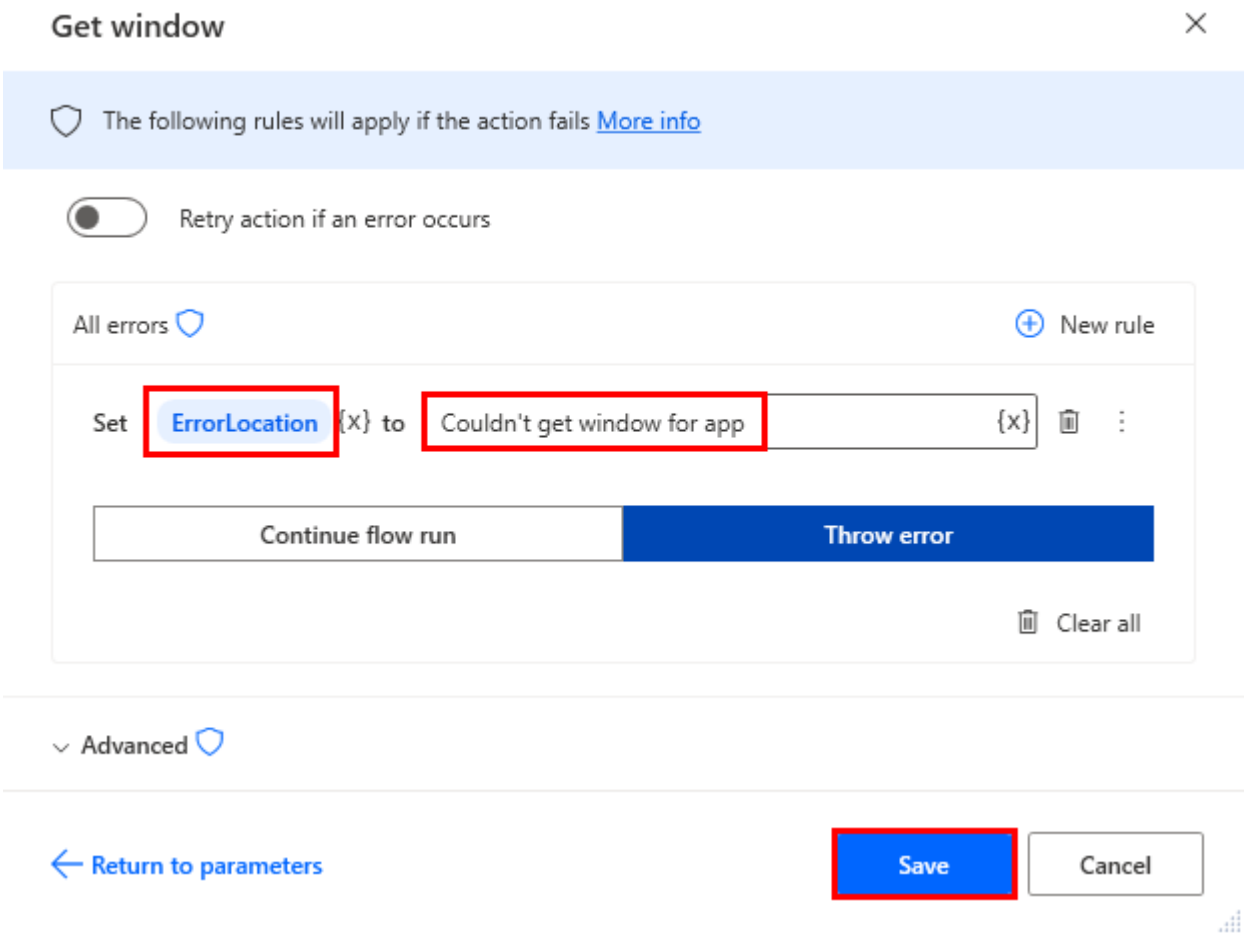
22. Click **+New rule**.



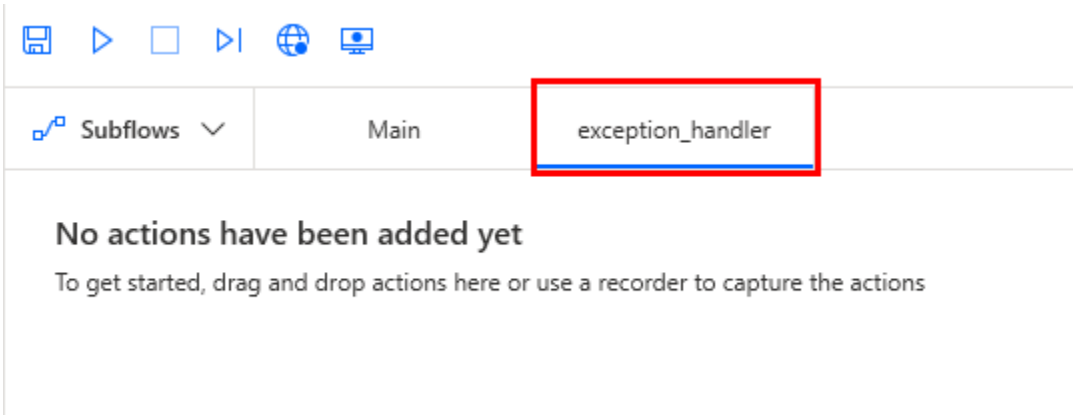
23. Select **Set variable.**



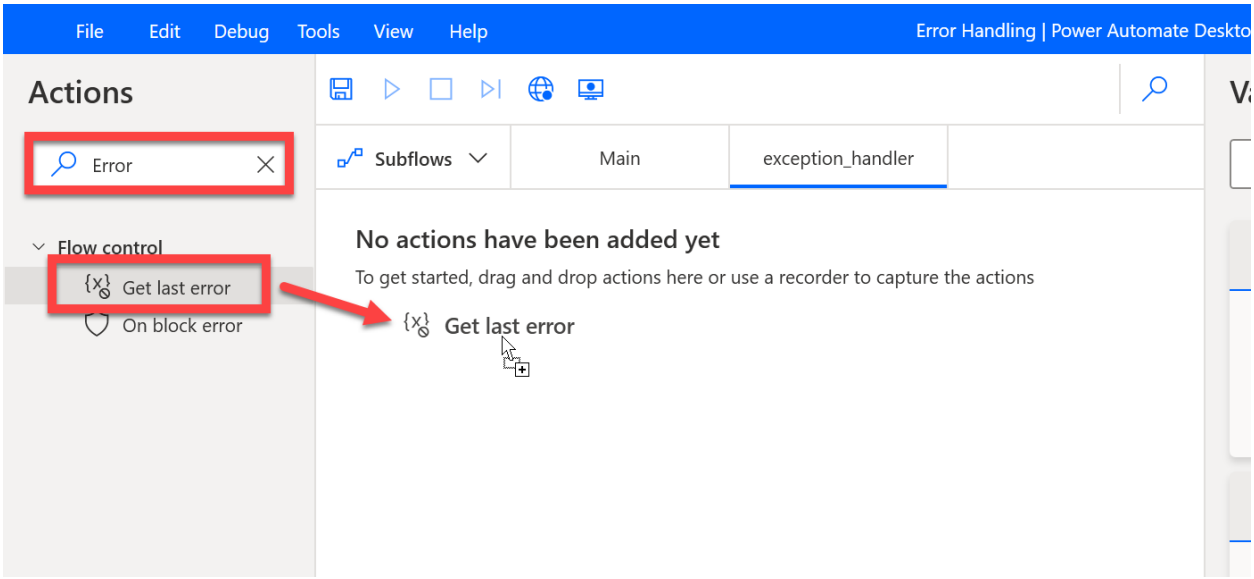
24. Set **%ErrorLocation%** to "**Couldn't get window for app**". Then click **Save**.



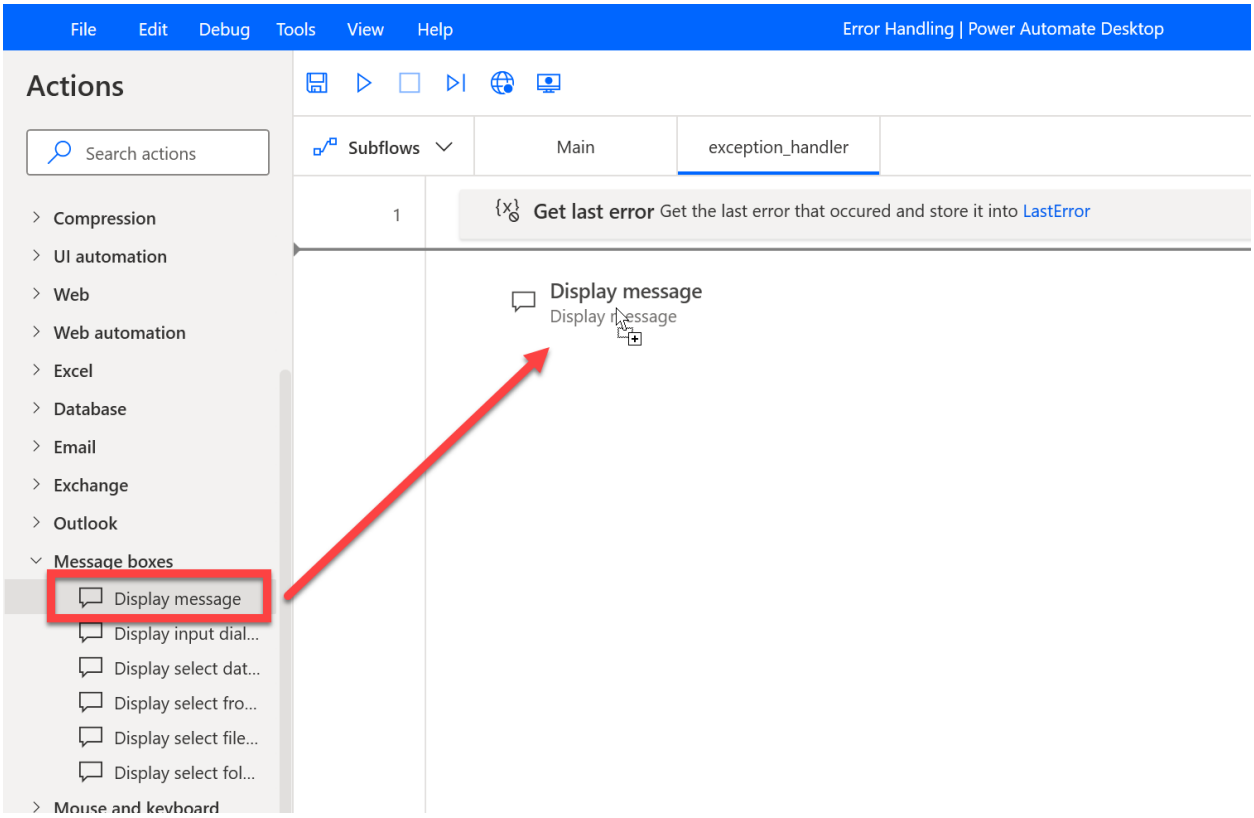
25. Now let's write what should happen in the exception_handler subflow. In this exercise we will just display the ErrorLocation value supplemented with details about the last exception. In the future you can create any of your own error handling logic. Go to **exception_handler** subflow.



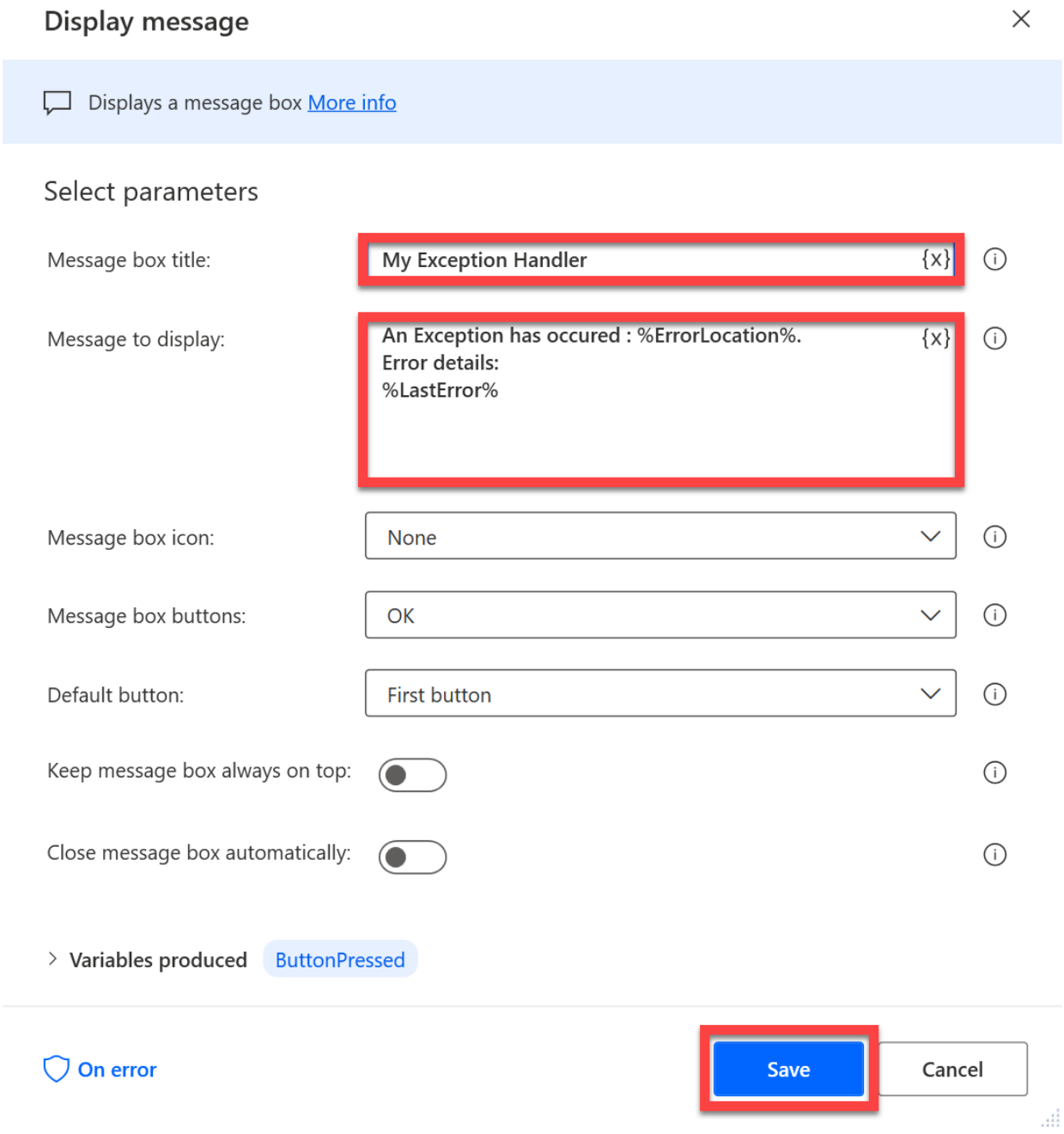
26. In **Search actions** type **Error**, then drag **Get last error** action to the canvas list. It will produce the **LastError** variable.



27. Next, under **Message boxes**, drag **Display message** to the canvas list.

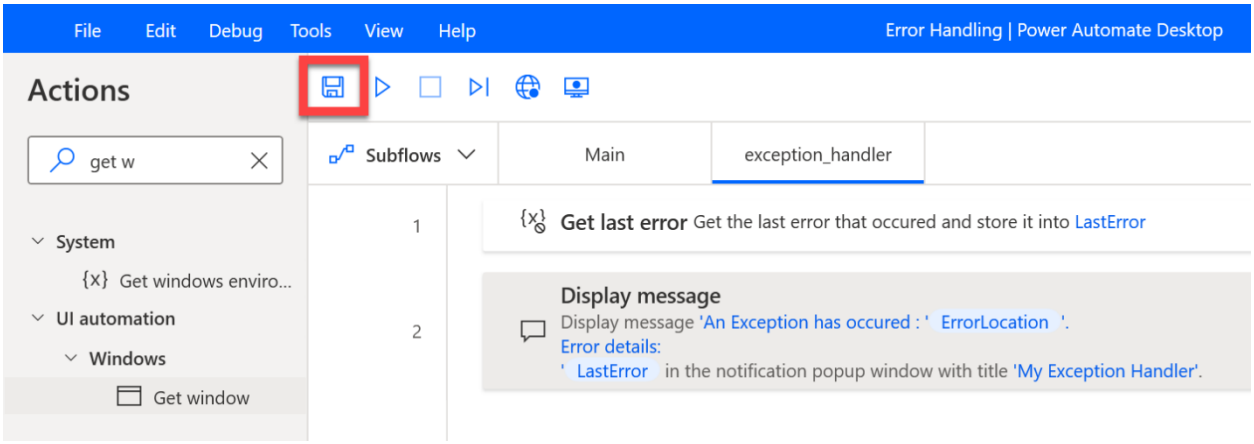


28. In **Message box title** column enter **My Exception Handler**, and in **Message to display** column enter:
An Exception has occured : %ErrorLocation%.
Error details:
%LastError%.
Then click **Save.**



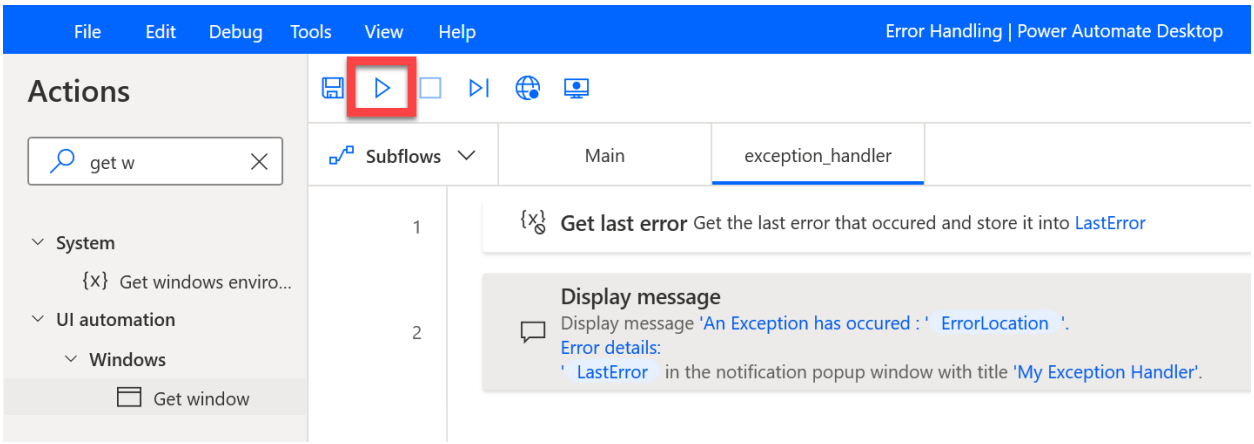
29. Click **Save.**

Note: We are using a **Display message** action for illustrative purposes. In a real-world scenario you can include additional logging/debugging information here including calling custom APIs or sending emails to a support team supplemented with screenshots.



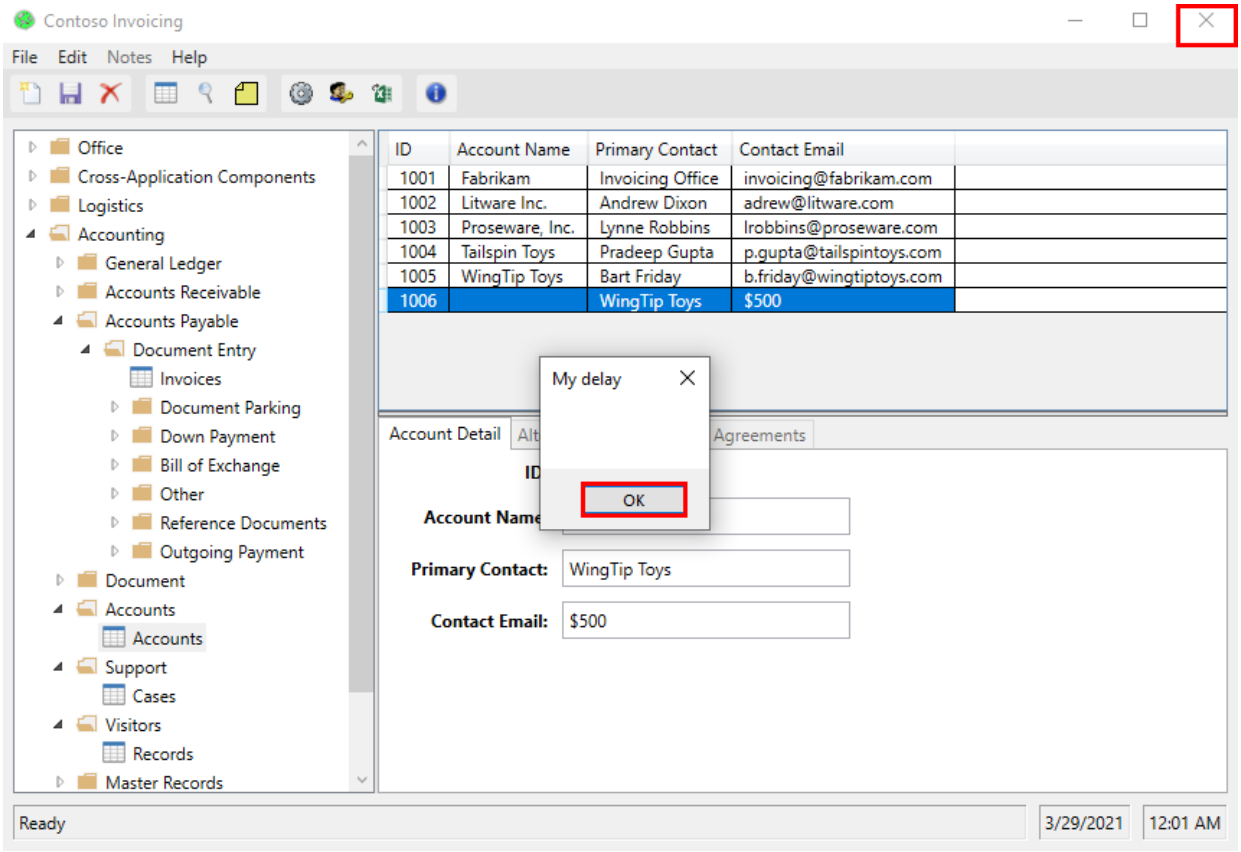
30. **Test** the desktop flow.

Note: you need to close all opened Contoso Invoicing apps page before testing the desktop flow.

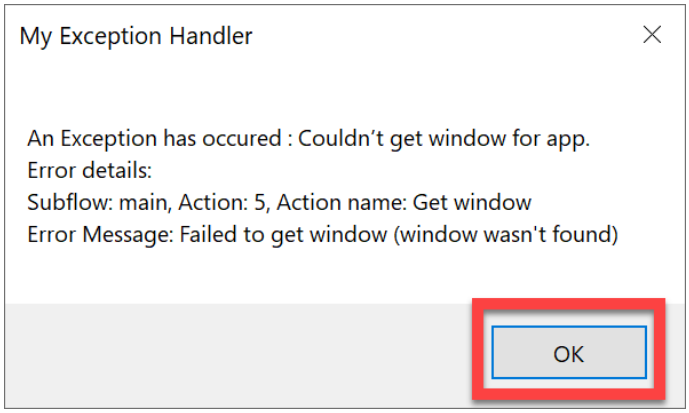


31. Contoso Invoicing application will automatically start and you will see the My delay message box pop up. Here you will first close the **Contoso Invoicing application** to ensure that the following step will fail. After you closed the application, click **OK** to close **My delay** message box and continue the process.

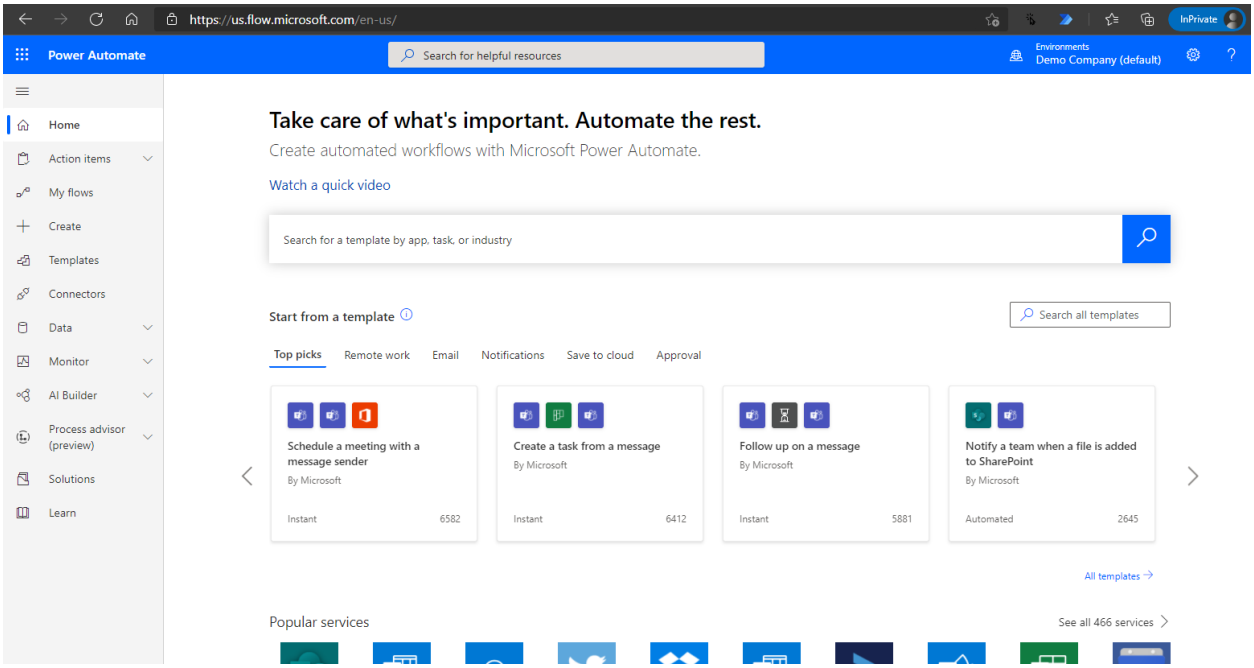
Note: you need to close Contoso Invoicing app before you close the message box.



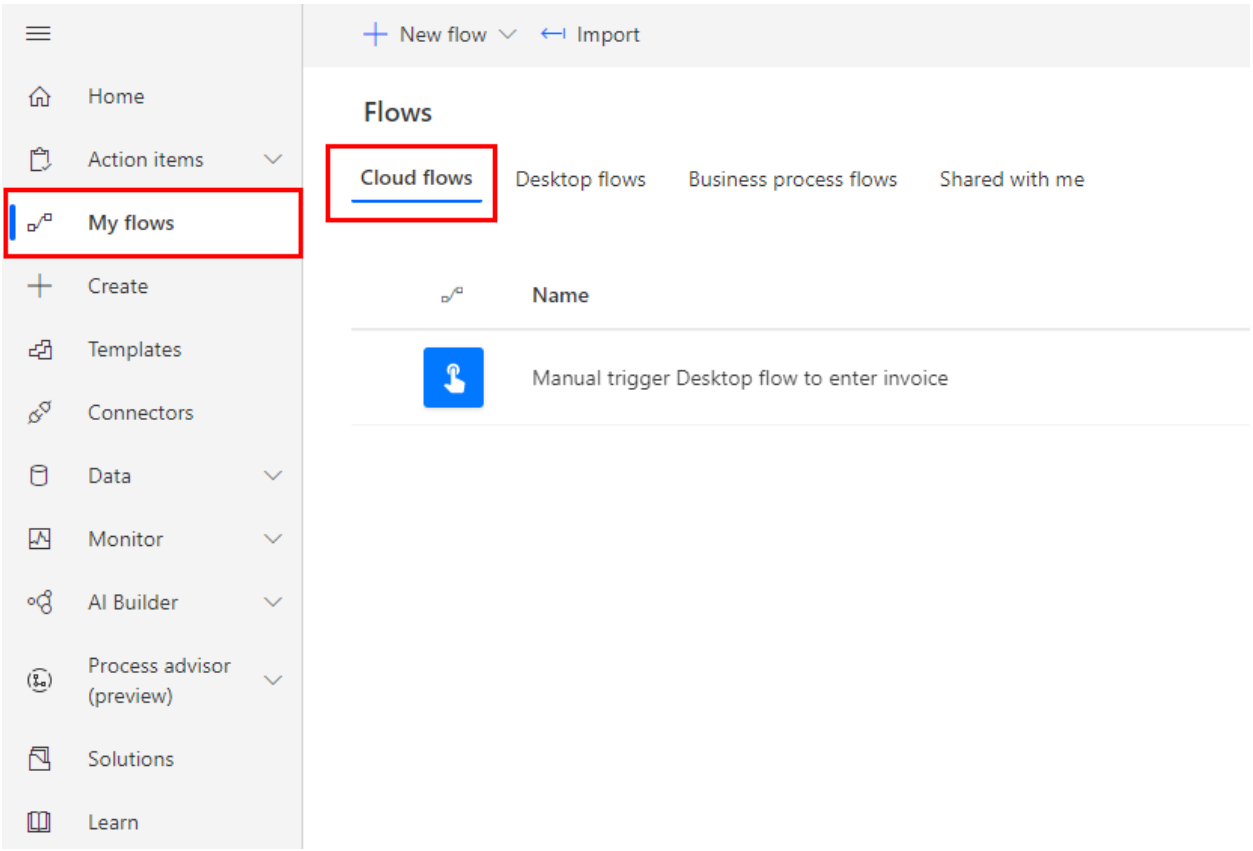
32. Now the flow will run the next step to **Get Window**, but since you have closed the application, this step will fail and will set the **ErrorLocation** variable value to be **Couldn't get window for app**. Then the **On Block Error** will handle this error automatically by running the **exception_handler** subflow. You will verify the exception handling run as expected if you see this prompt message: **An exception has occurred: Couldn't get window for app** together with detailed error information. You can click **OK** to close it.



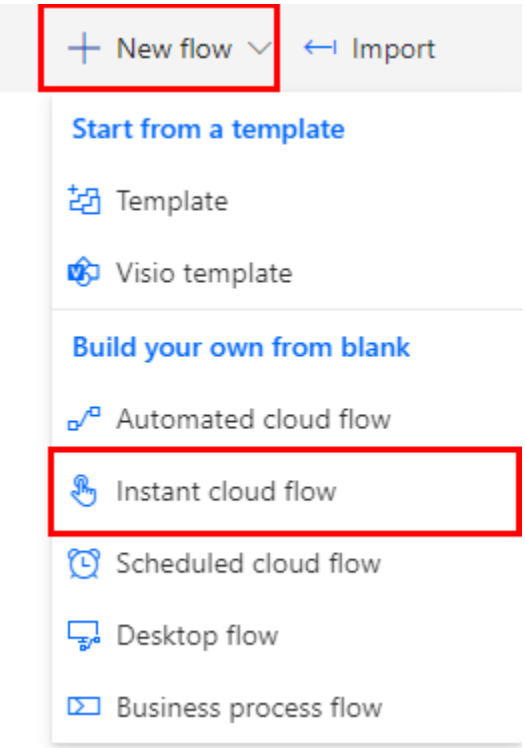
33. Now let's see what the expected behavior from cloud flow side is. Navigate to https://powerautomate.microsoft.com



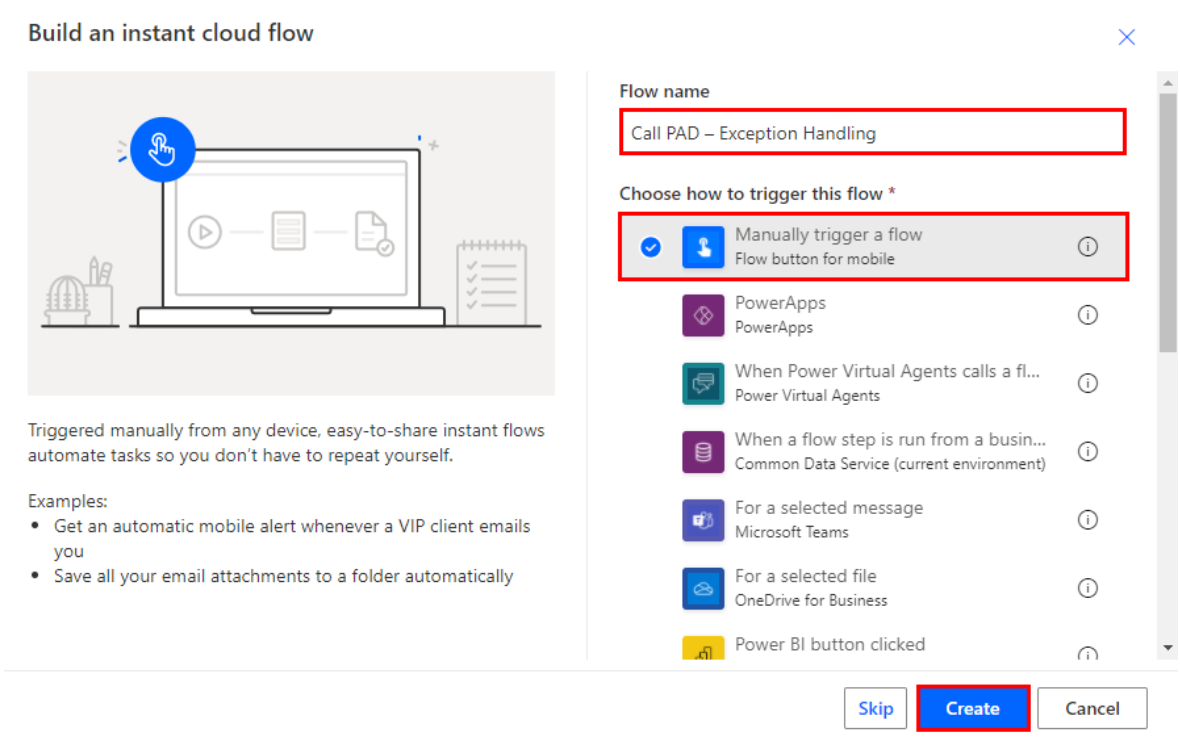
34. Go to **My flows – cloud flows**



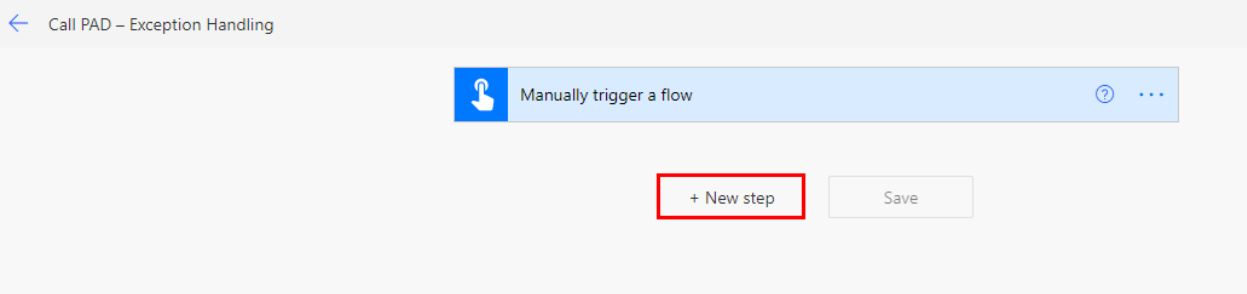
35. Click **New flow** – **Instant cloud flow**.



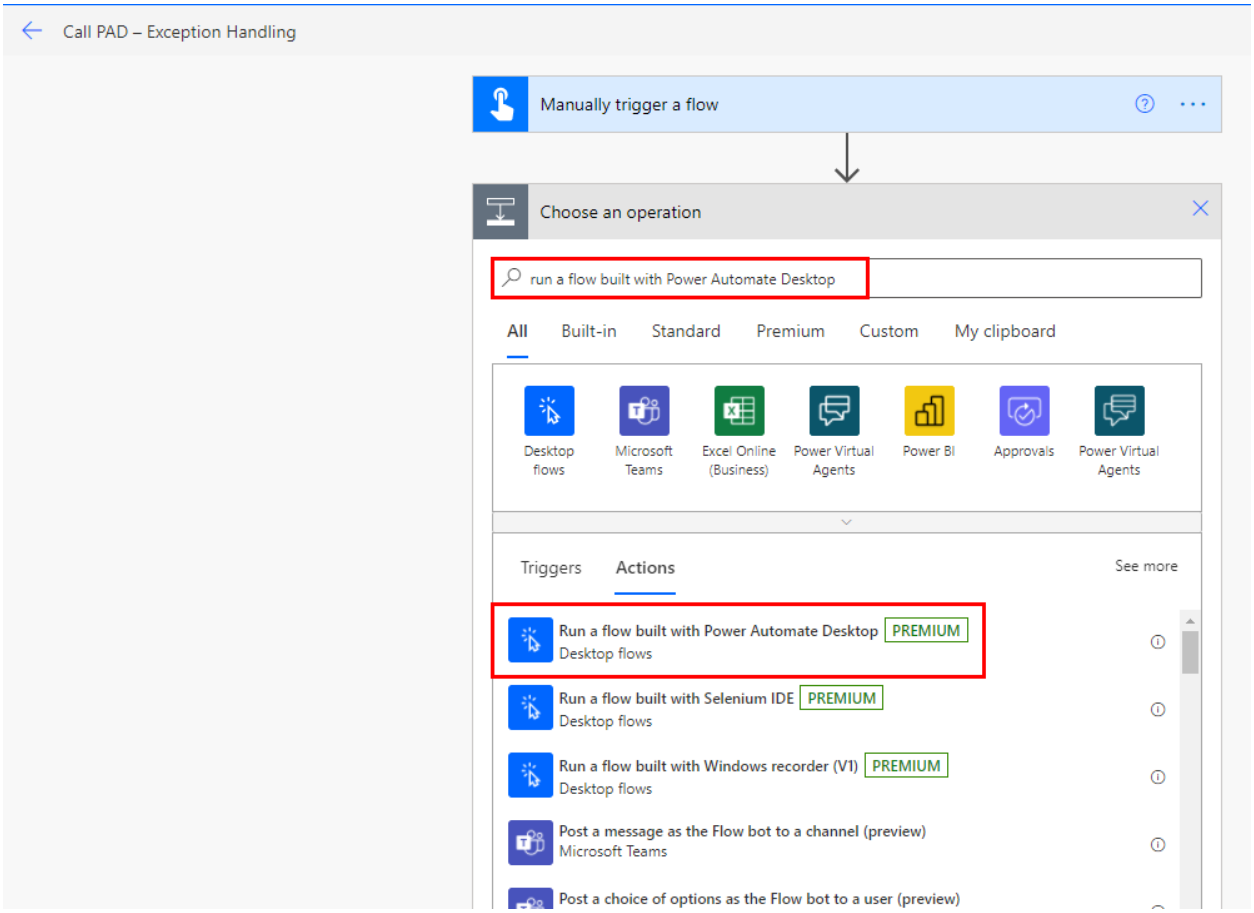
36. Name the flow **Call PAD – Exception Handling.** Select **Manually trigger a flow.** Then click **Create.**



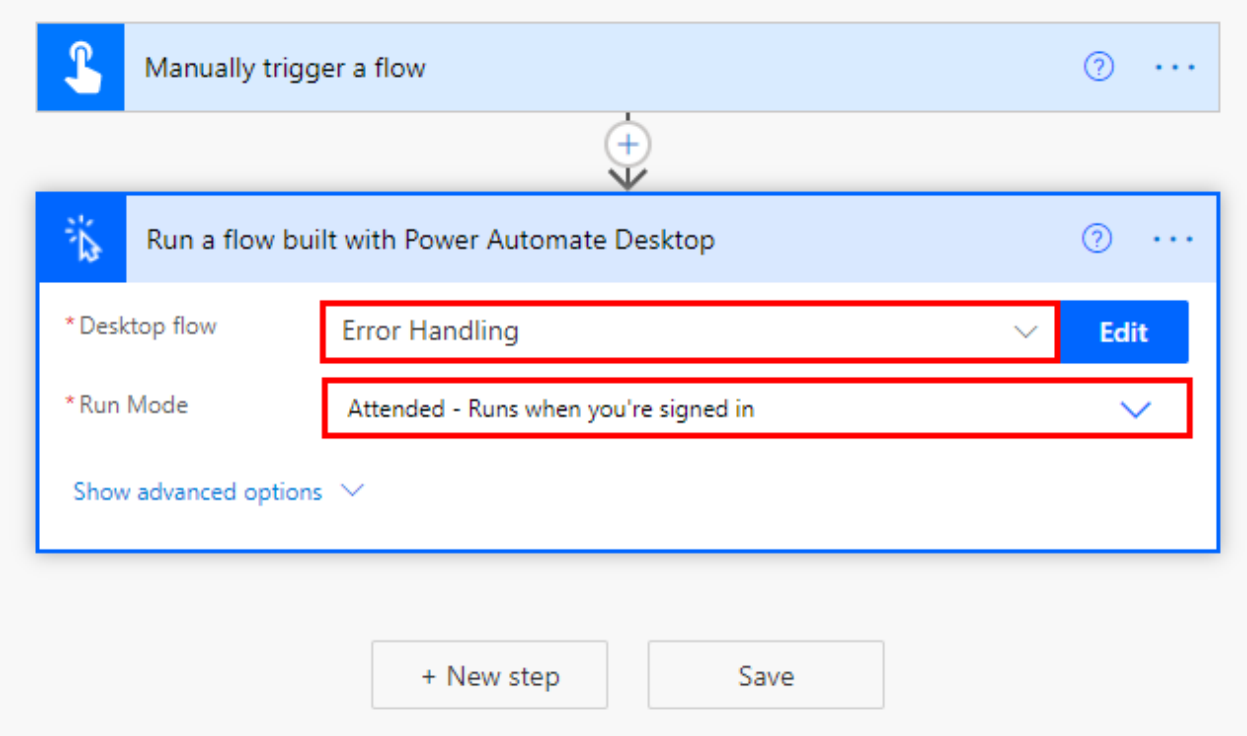
37. Click **New step**.



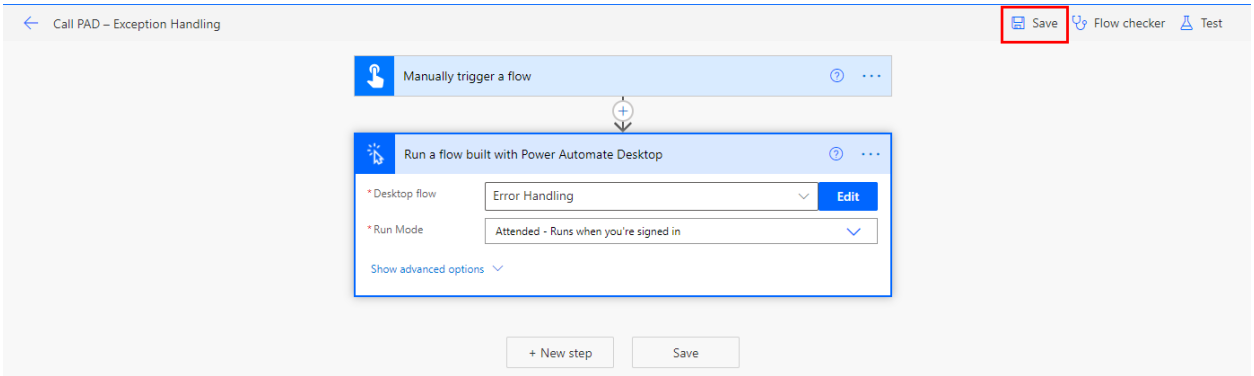
38. Select **Run a flow built by Power Automate Desktop** using search bar.



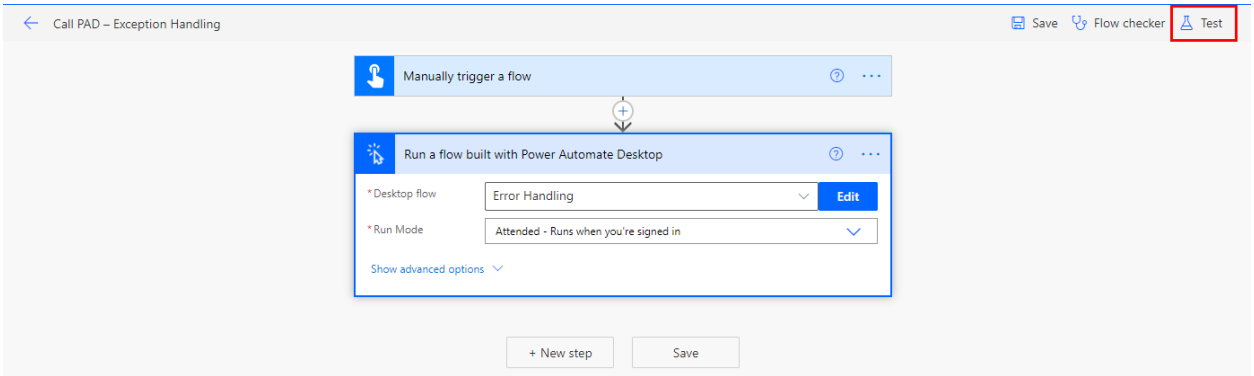
39. Select **Error Handling** desktop flow and in **Attended** run mode.



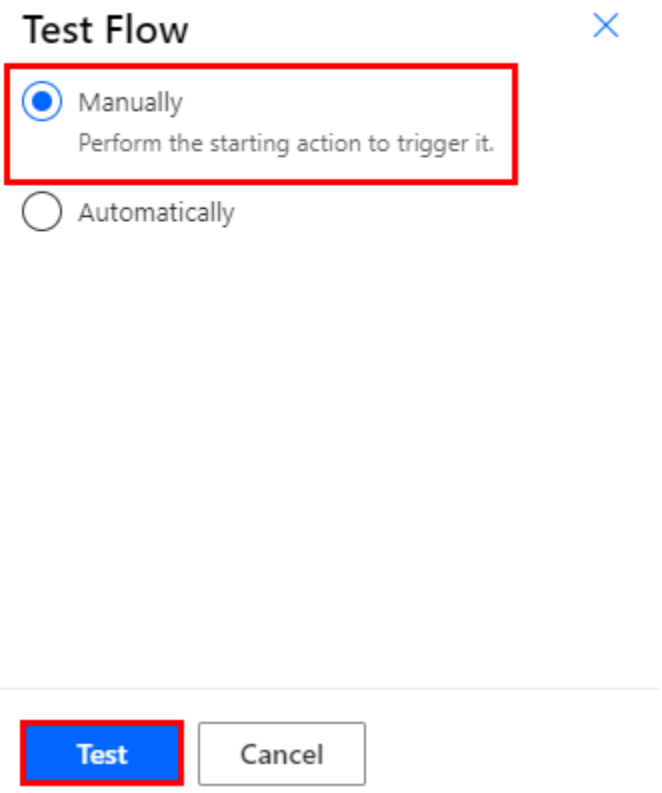
40. Click **Save**.



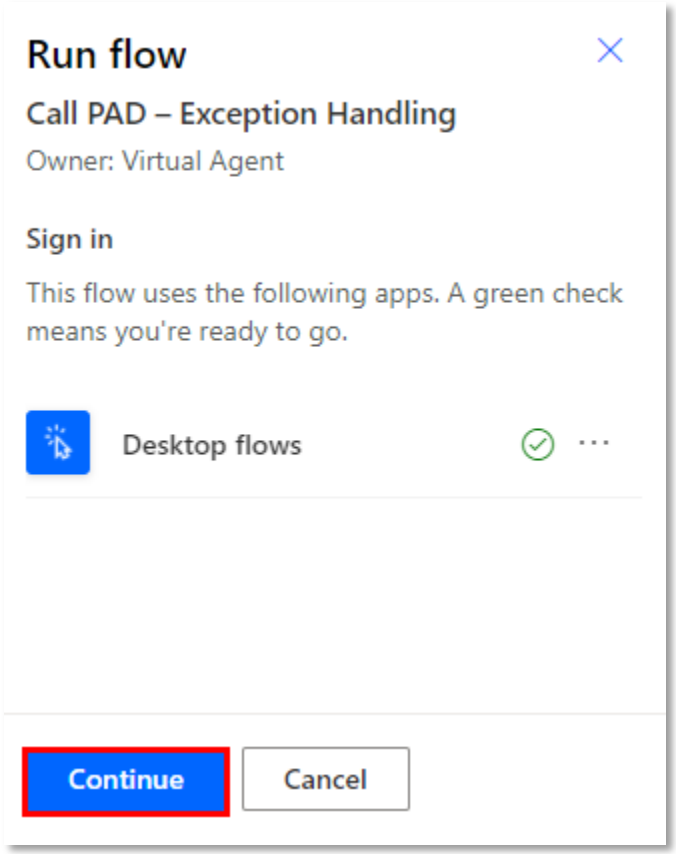
41. Click **Test**.



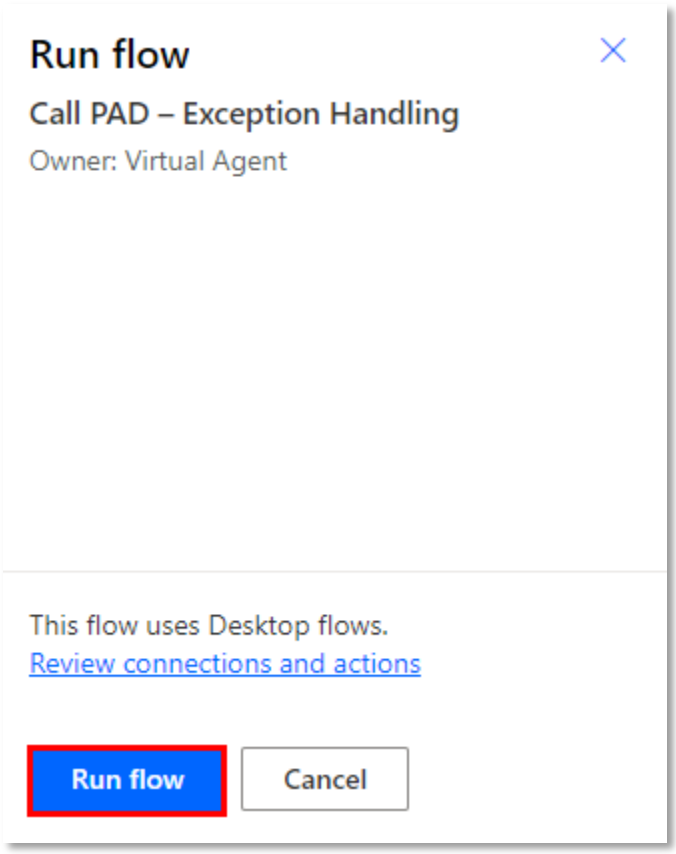
42. Select **Manually** and click **Test**.



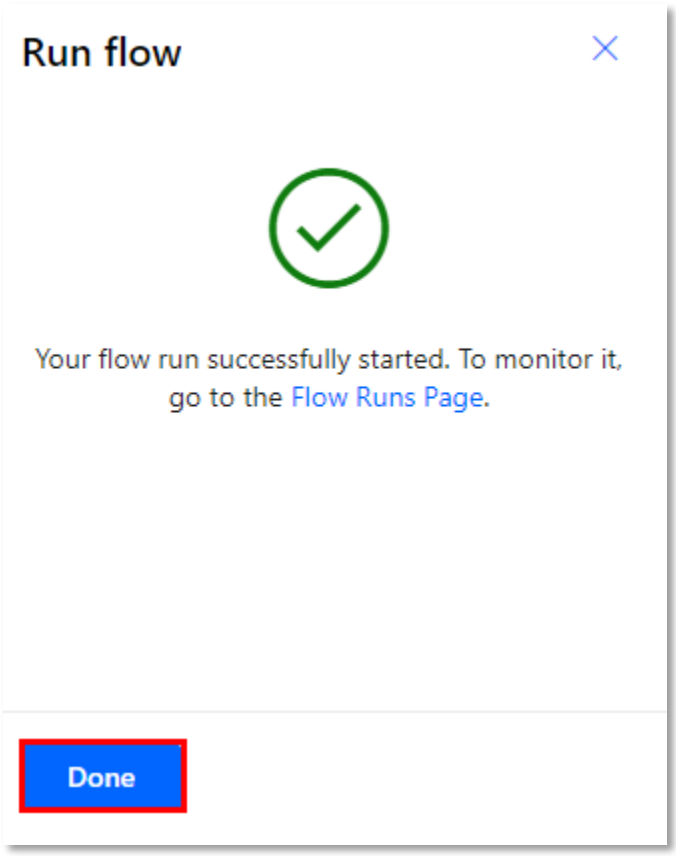
43. Click **Continue**.



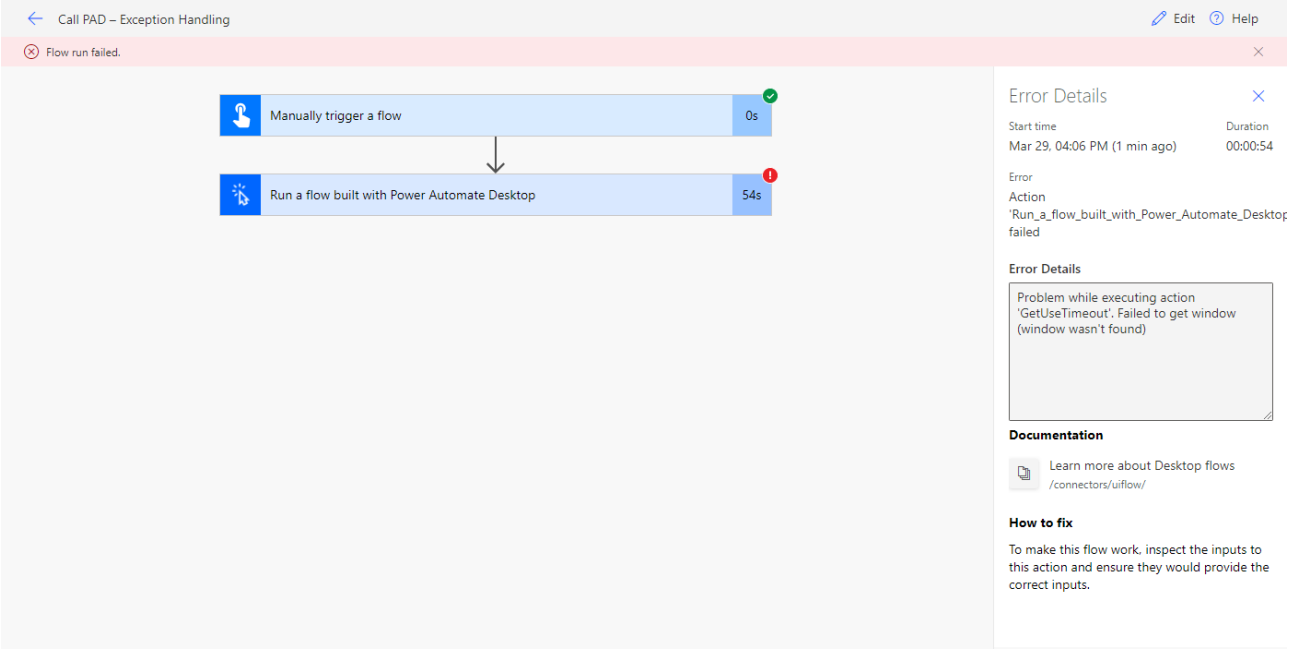
44. Click **Run flow**.



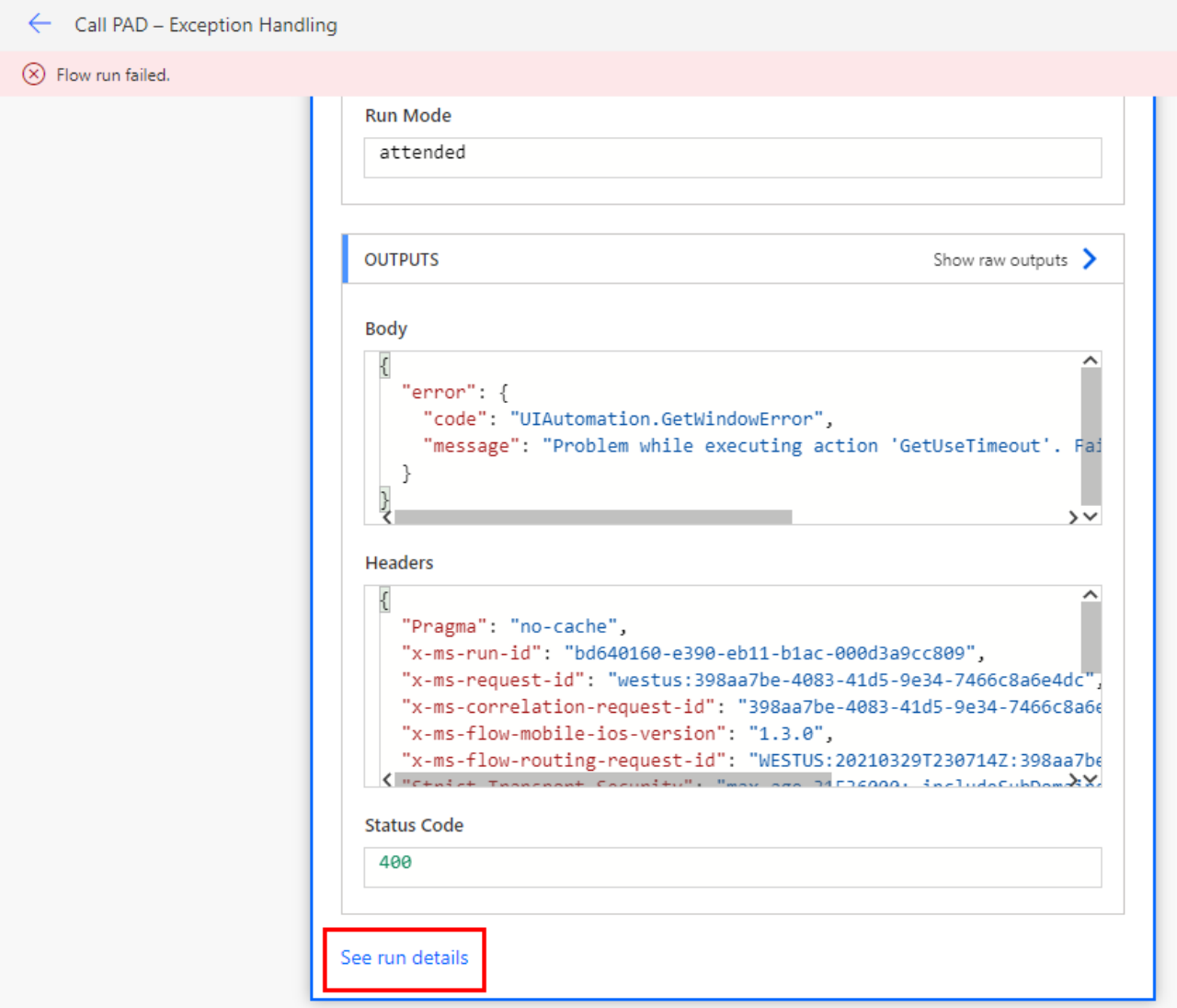
45. Click **Done**.

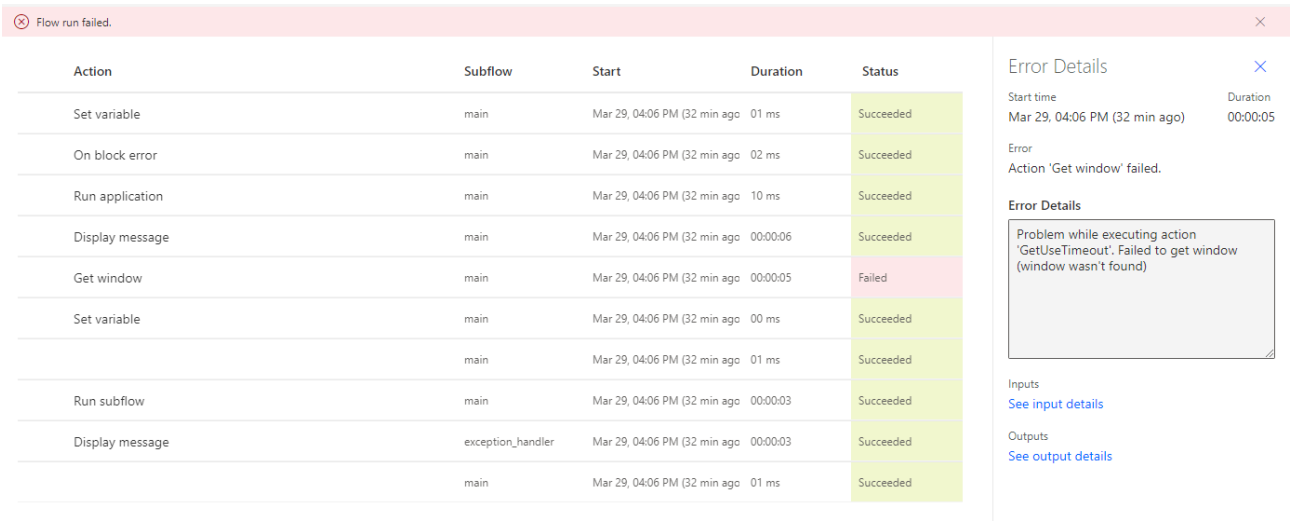


46. You will notice that the cloud flow run failed. This is important for users to get accurate telemetry send from the desktop back to the cloud. Although the exception has been gracefully handled within the desktop flow, we still want all the exception errors to surface in the cloud flow and let users be aware what error has occurred and where it occurred.



47. From the cloud flow Run history, expand on the **Run a desktop flow** action to see more failure details, and lick the **See run details** link at the bottom to browse to the detail desktop run history.





48. From this detailed steps' log, you can see the Get window step has failed with error details, and then the exception handler subflow was run as designed.

Note: if your cloud flow run did not report as failed, it could be because you do not have the latest Power Automate Desktop version (v2.6 or later). In that case, please update to the latest Power Automate Desktop and re-test.

Check your knowledge

Lab 11

5 mins

1. You need to close all opened Contoso Invoicing apps page before testing the desktop flow.

   A. True
   B. False

   Answer: A. True - Contoso Invoicing application will automatically start when your desktop flow start running

2. In order to create an error case for this lab we will manually _____ the application so the following step cannot locate the window anymore.

   A. Open
   B. Close
   C. Either A or B
   D. None of the above

   Answer: B. Close – Once we closed the application, our desktop flow cannot locate the window anymore

3. If your desktop flow run did not report as failed on step 46, it could be because you _____.

   A. You do not have the latest Power Automate Desktop version (v2.6 or later)
   B. You do not have the latest Power Automate Desktop version (v2.4 or later)
   C. None of the above

   Answer: A. You do not have the latest Power Automate Desktop version (v2.6 or later)

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations or warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2021 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at https://www.microsoft.com/enus/legal/intellectualproperty/Trademarks/Usage/General.aspx are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.