

Logic Apps | Re-execute an action when it fails

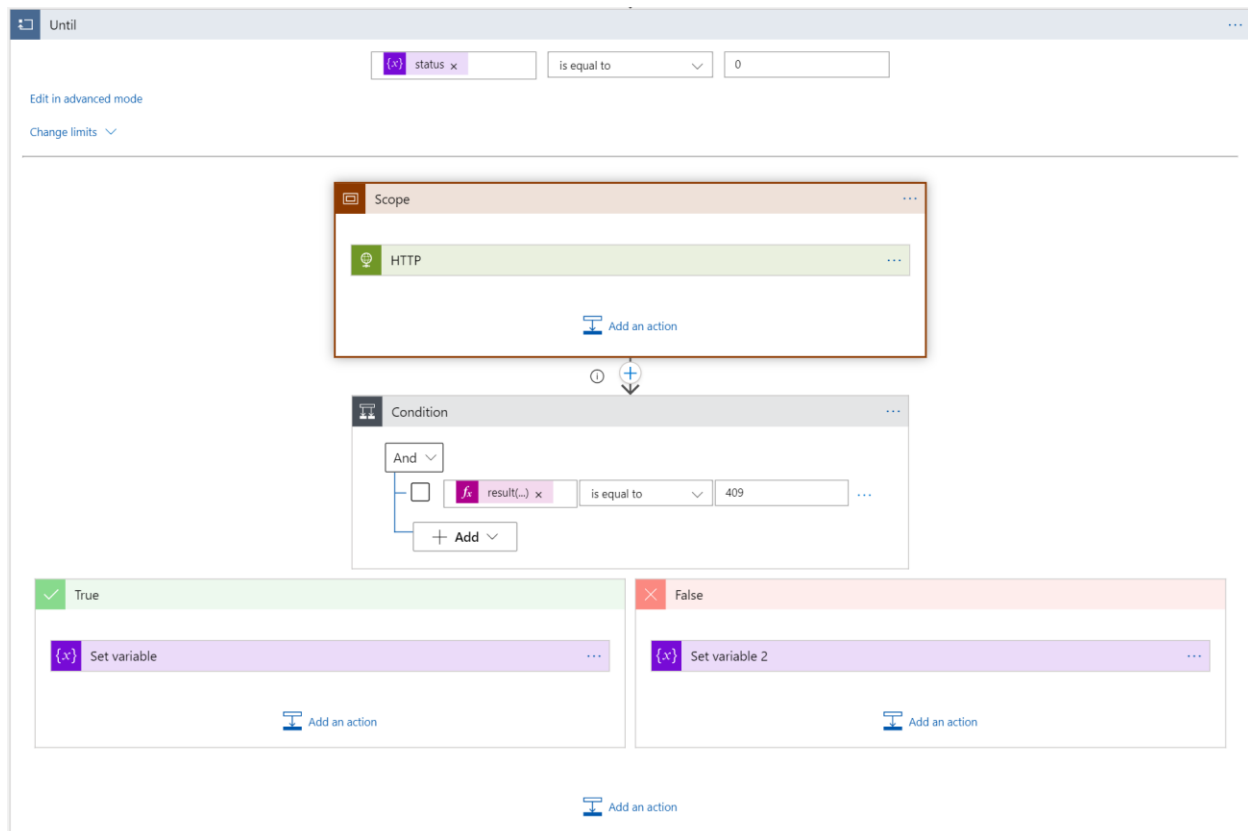
Logic Apps allows to retry failed action executions with the retry policy described in the below link:

<https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-exception-handling#retry-policies>

“A retry policy specifies whether and how the action or trigger retries a request when the original request times out or fails, which is any request that results in a 408, 429, or 5xx response”

However, there could be some scenarios where could be needed to retry the action execution for other error codes than the ones mentioned (408, 429, or 5xx). This document provides a work around for intermittent issues to allow to re-execute an action when the status code is not covered by the retry policy.

Design overview

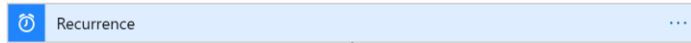


Steps to follow and related documentation:

Add A trigger

In this example we use a recurrence trigger

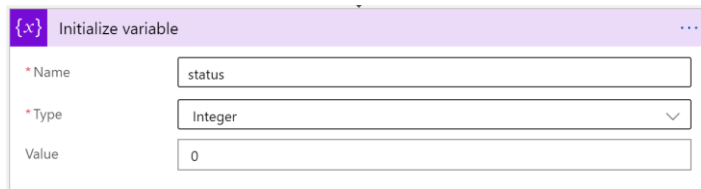
<https://docs.microsoft.com/en-us/azure/logic-apps/tutorial-build-schedule-recurring-logic-app-workflow#add-the-recurrence-trigger>



Initialize a variable

Initialize an integer variable that we can call 'status' and will be used to handle the success or failure of the action in question

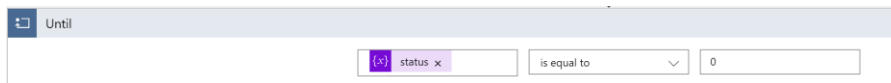
<https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-create-variables-store-values#initialize-variable>

A form titled "Initialize variable" with a purple icon of a variable on the left. It contains three fields: "Name" with the value "status", "Type" with a dropdown menu showing "Integer", and "Value" with the value "0".

Add an 'Until' Loop

In this loop's condition we will select the variable initialized before('status') configuring: is equal to 0. When the action succeeds the loop will run once, otherwise will run more than once depending on the settings in the loop and expecting that some re-execution succeeds.

<https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-control-flow-loops#until-loop>

A horizontal bar with a blue icon of a loop on the left, the word "Until" in the center, and a configuration area on the right. The configuration area contains a variable selector showing "status", a dropdown menu showing "is equal to", and a text box with the value "0".

In this loop you can change limit options, you can select an execution limit and to time out limit. In the example below the loop will only run a max of 5 times and would timeout if it runs for 5 minutes

Until

Edit in advanced mode

Change limits ^

Count

5

Timeout

PT5M

Add a Scope

The next step is to add a scope inside the loop, this will be used to capture the error of the action added within the scope

<https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-control-flow-run-steps-group-scopes#add-steps-to-scope>

Scope

Add an Action in the scope

Add the action you want the workflow re-execute when it fails.

Scope

HTTP

Add an action

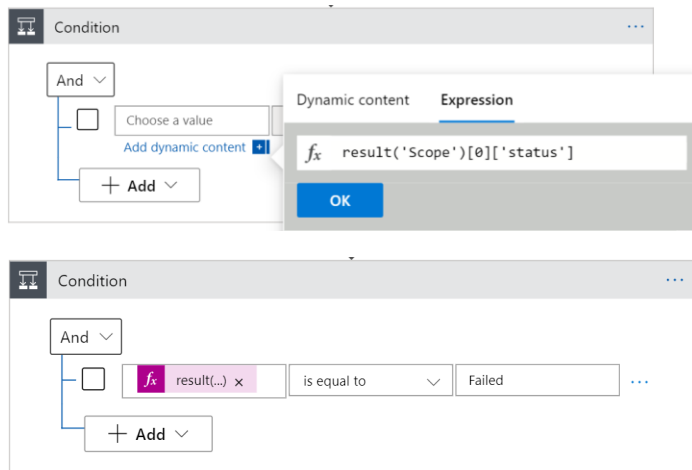
Add a Condition step

After the Scope you can add a condition.

<https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-control-flow-conditional-statement>

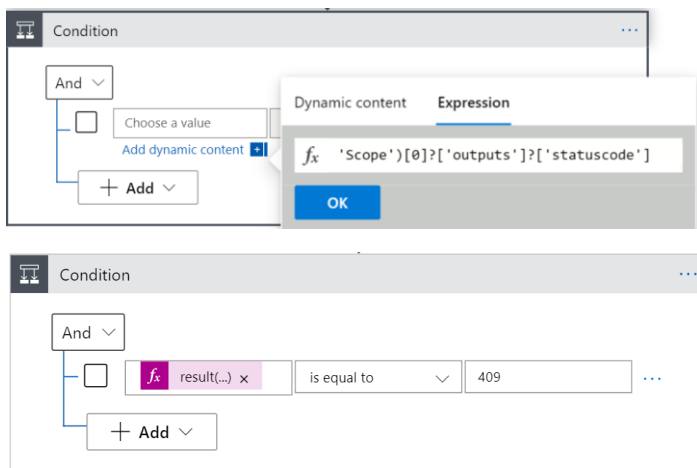
The condition can be the following:

result('Scope')[0]['status']	is equal to	Failed
------------------------------	-------------	--------



Or the condition can check status code. In the example below the action will re-execute only when it fails with code 409

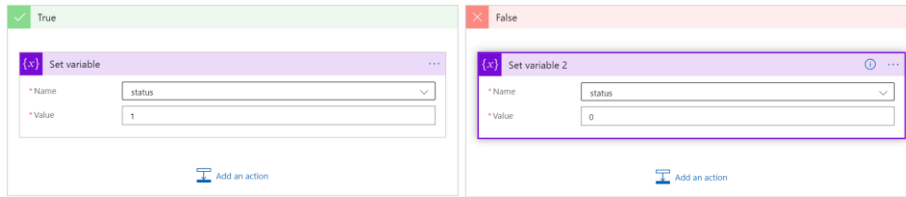
<code>result('Scope')[0]?['outputs']?['statuscode']</code>	is equal to	409
--	-------------	-----



If the action within the scope fails, the result will be True which means the action needs to re-execute. If the action within the scope succeeds, the result will be False which means the loop should break to continue to run subsequent actions in the workflow.

Set the variable

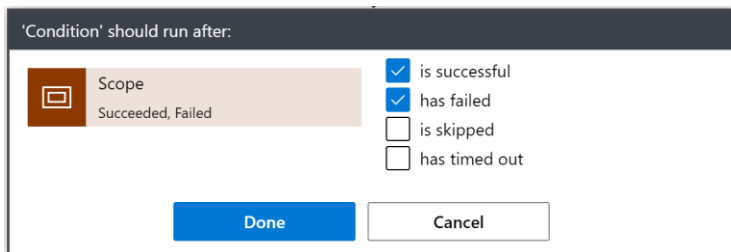
In the true section we set the variable 'status' to 1 so the loop can run again, and in the false section we set the variable 'status' to 0 to break the loop and the workflow run subsequent actions.



Configure run after in the condition step

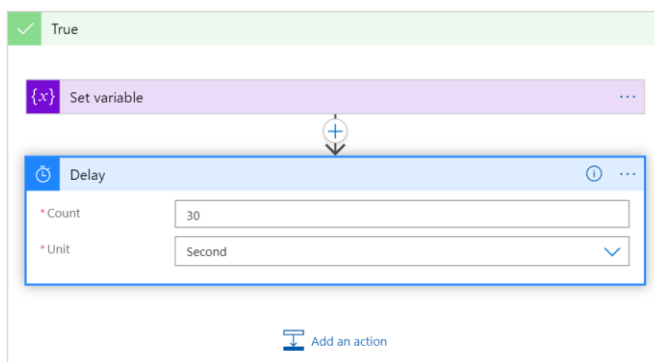
To allow the workflow run keeps running after the action fails, we need to configure the run after in the condition step.

<https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-exception-handling#catch-and-handle-failures-by-changing-run-after-behavior>



Add a delay if needed

Additionally, in the True section you could add a waiting interval using a delay step so the workflow can wait a certain period of time before executing the action again.



After completing all the steps described you can test your Logic App.