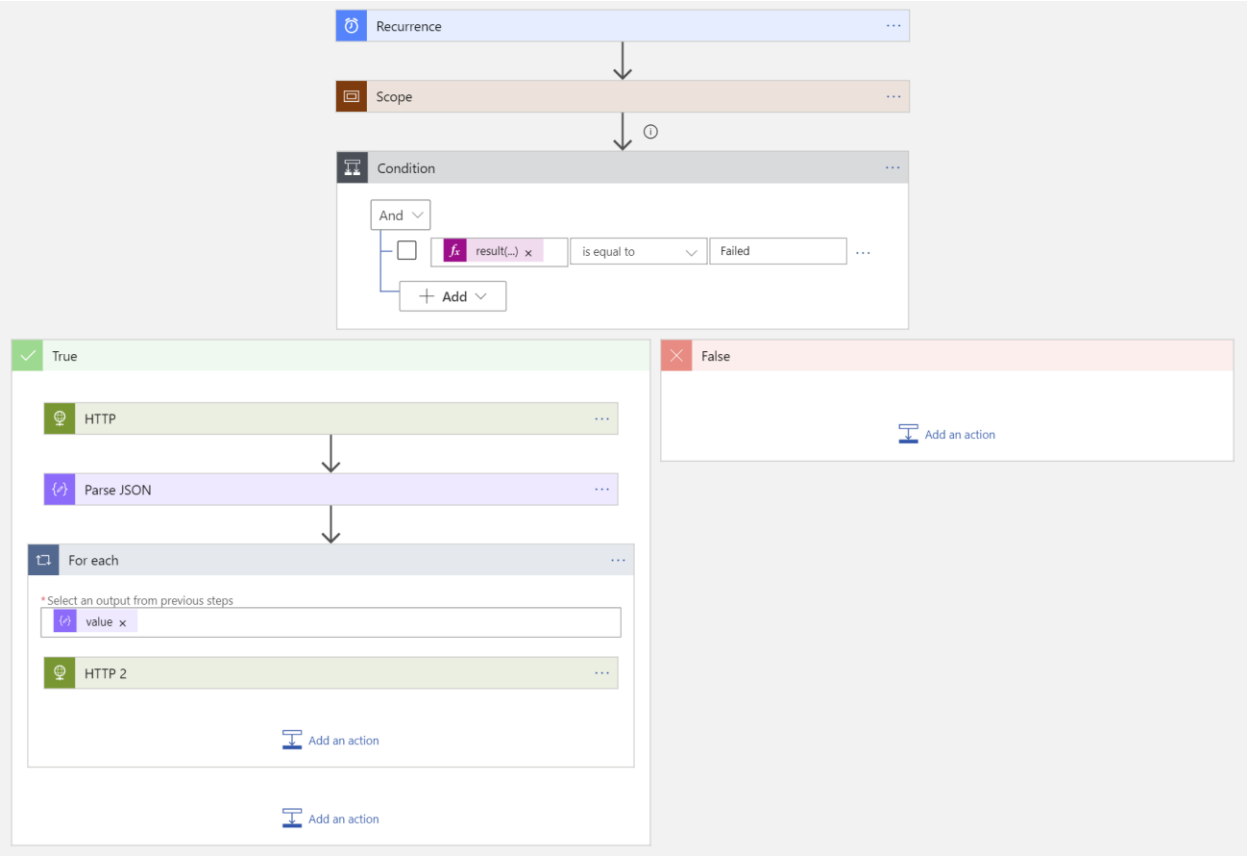
Logic Apps | Resubmit RunId automatically when an action fails

The solution described in this document is to resubmit a workflow run when a specific action within the workflow fails. This is done using Rest API.

Caution: If the action in scope is failing without any way to automatically succeed while resubmitting the workflow run, this can create an endless loop where the Logic App workflow will be executed over and over again until the action in scope succeeds, the Logic App workflow is cancelled or the Logic App is disabled.

Solution design Overview



Steps to follow and related documentation

Add A trigger

Every workflow includes 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

Add a Scope

The next step is adding a scope, this will be used to handle exception 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

Add an Action within the scope

Once you have added the Scope you can add the monitored action within the scope, it can be any action that you want the workflow automatically checks if it fails to resubmit the current workflow run.

Add a Condition step

After the Scope step you can add a condition step which is part of the control built-in connector

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

The validation will be the following:

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

If the action within the scope fails, the result will be True which means the workflow will go to the true section where it resubmits the workflow run. If the action within the scope succeeds, the result will be false which means the workflow will go to the False section where it does not resubmit the workflow run.

Configure run after in the condition step

By default, a workflow run fails when any action within the workflow fails, hence, to allow the workflow run keeps running after an action fails we need to configure the run after in the next step where the failure occurs, in this case we expect that the action we added in the scope may fail, so 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

How to call Rest API from Logic Apps

To call Rest API we can use HTTP action

https://docs.microsoft.com/en-us/azure/connectors/connectors-native-http#add-an-http-action

For this example, we will resubmit a RunId by using Rest API (Workflow Trigger Histories – Resubmit)

https://docs.microsoft.com/en-us/rest/api/logic/workflowtriggerhistories/resubmit

POST
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Logic/workflows/{workflowName}/triggers/{triggerName}/histories/{historyName}/resubmit?api-version=2016-06-01

To successfully execute the Rest API above there are two consideration we need to take care of:

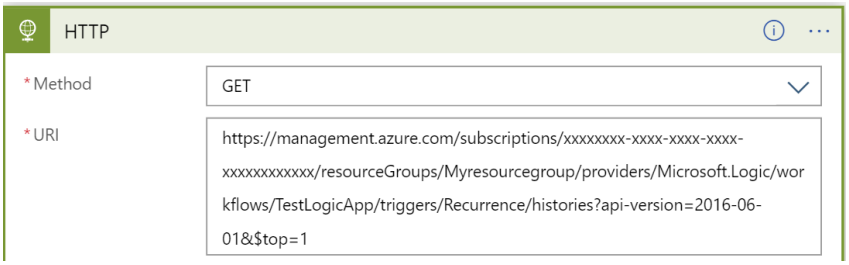
- Workflow Trigger Histories – Resubmit URL needs a history name which is a RunId, so we first need to list the RunId required
  https://docs.microsoft.com/en-us/rest/api/logic/workflowtriggerhistories/list
  GET
  https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Logic/workflows/{workflowName}/triggers/{triggerName}/histories?api-version=2016-06-01&$top={$top}&$filter={$filter}

- Resubmit Rest API requires authentication, to allow this we can enabled manages identity
  https://docs.microsoft.com/en-us/azure/logic-apps/create-managed-service-identity

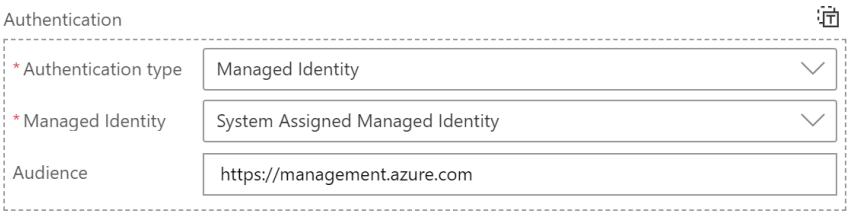
Add HTTP action to List RunId

After adding the http action, select GET in the method field, then modify the sample URL in the REST API link below according to your Logic App resource ID, your trigger name and ”&$top=1” at the end of the URL to list only the latest RunId (which is the running RunId), then add that URL to the URI field.

Example:



Then set up the authentication parameter, example



https://docs.microsoft.com/en-us/azure/connectors/connectors-native-http#add-an-http-action

https://docs.microsoft.com/en-us/rest/api/logic/workflowtriggerhistories/list

Add a Parse JSON

https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-perform-data-operations#parse-json-action

The content you can use is the http body output that you can select from the dynamic content

For the schema you can use the following:

```
{
    "properties": {
        "nextLink": {
            "type": "string"
        },
        "value": {
            "items": {
                "properties": {
                    "id": {
                        "type": "string"
                    },
                    "name": {
                        "type": "string"
                    },
                    "properties": {
```

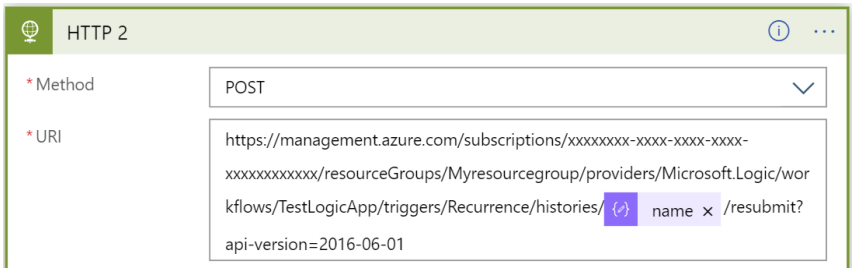
```
            "properties": {
                "code": {
                    "type": "string"
                },
                "correlation": {
                    "properties": {
                        "clientTrackingId": {
                            "type": "string"
                        }
                    },
                    "type": "object"
                },
                "endTime": {
                    "type": "string"
                },
                "fired": {
                    "type": "boolean"
                },
                "run": {
                    "properties": {
                        "id": {
                            "type": "string"
                        },
                        "name": {
                            "type": "string"
                        },
                        "type": {
                            "type": "string"
                        }
                    },
                    "type": "object"
                },
                "scheduledTime": {
                    "type": "string"
                },
                "startTime": {
                    "type": "string"
                },
                "status": {
                    "type": "string"
                }
            },
            "type": "object"
        },
        "type": {
            "type": "string"
        }
    },
```

```
                "required": [
                    "properties",
                    "id",
                    "name",
                    "type"
                ],
                "type": "object"
            },
            "type": "array"
        }
    },
    "type": "object"
}
```

Add HTTP action to resubmit the RunId

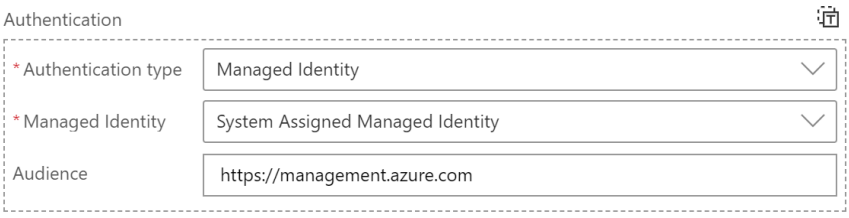
After adding the http action, select POST in the method field, then modify the sample URL in the REST API link below according to your Logic App resource ID and for the historyname you can select the "name" property from the Parse_JSON output.

Example:



Note: selecting the "name" property in the URI field will automatically add a foreach loop, this happens because that property is withing the value array which has only 1 item because we are only listing the latest RunId in the previous http action.

Then set up the authentication parameter, example:



https://docs.microsoft.com/en-us/azure/connectors/connectors-native-http#add-an-http-action

https://docs.microsoft.com/en-us/rest/api/logic/workflowtriggerhistories/resubmit

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