

## Lab 15. (Approval Advanced) Implementing the Flow Controller Pattern

**Learning objective:** Create state machines and have Flows that can run beyond the 30 days limitation by using the Flow Controller pattern.

**Duration:** 30 minutes.

**Scenario:** When a user submits a new expense in a SharePoint list, the expense will have to be approved by a first user (line manager); if this user doesn't react on time, another user (big boss) will react. If the big boss doesn't react on time, the system will ask the big boss to react again and again. The Flow should work even if the whole process takes more than 30 days (current Flow limitation).

### Tasks:

This lab illustrates an implementation of the Flow Controller pattern with the HTTP action. This requires a premium license, but an implementation with a flow started from a message stored into a SharePoint list will be available soon and has been tested years ago.

This controller pattern Flow provides more flexibility in the Flow architecture design; we will apply this concept to have Flows that can run beyond the 30 days limitation (Flow run and approval), but also to implement state machines even though the current Flow designer doesn't support state machines yet.

In this lab, we will create 3 Flows:

- The launcher Flow
- The controller Flow
- The generic approval Flow

Setup

1. Create a SharePoint list named "**Expenses**" with 3 columns: the default column (**title**), a column **Amount** (type number) and a column named **Status** (choice with the values **In progress**, **Rejected**, **Accepted**).

### Launcher Flow implementation

2. Create a Flow named **Approval launcher** that will start when a new expense is submitted in this list.
3. The trigger of this Flow is **When an item is created** (SharePoint Connector)
4. Add a **compose** action, call it **const controller URL** and set its value to "todo". We will update it afterward.
5. Add an **HTTP action**, name it **HTTP - Call Controller**
  - a. Set the method POST
  - b. Set the URI to the Output of **const controller URL**
  - c. In the Headers set **content-type** to **application/Json**
  - d. Set **Body** like as described below:

You can notice that ID is not surrounded by double quotes because it is an integer.

When an item is created

const controller URL

Inputs \*

todo

HTTP - call controller

Method \*

POST

URI \*

Outputs

Headers

content-type	application/json
Enter key	Enter value

Queries

Enter key	Enter value
-----------	-------------

Body

```
{
  "itemid": ID,
  "requester": "Created By Email",
  "from": "launcher",
  "response": ""
}
```

Cookie

Enter HTTP cookie

Show advanced options

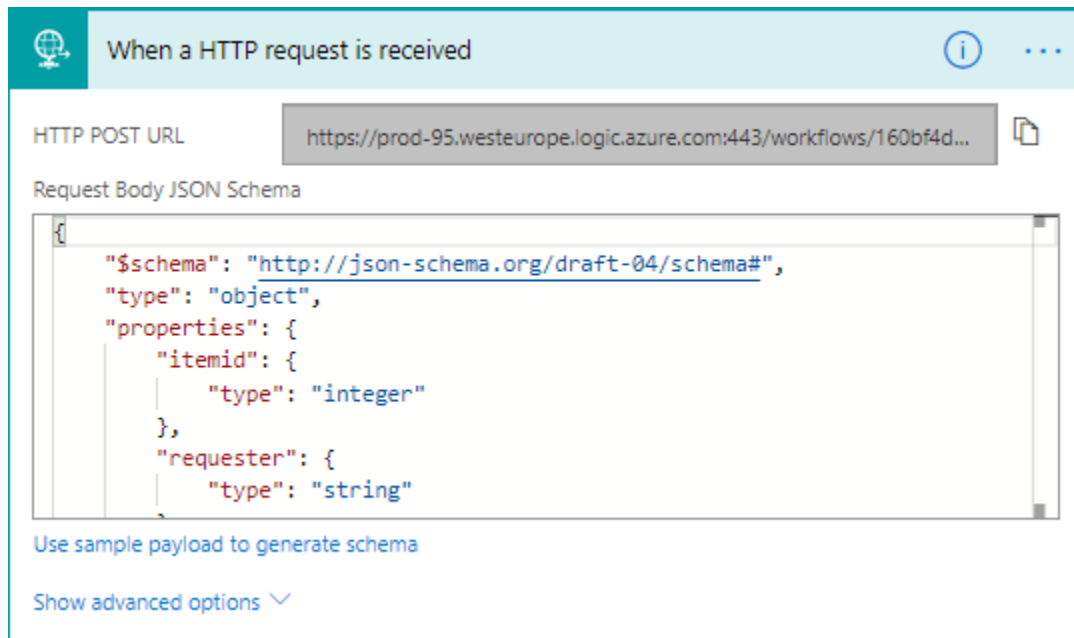
The Flow controller will be implemented in the last place. Let us focus on the Approval logic.

### Approval generic Stage implementation

1. Create a new Flow called **Approval generic stage** that starts with the trigger **When an HTTP request is received**.
2. In the Request body of the trigger, copy and paste the following JSON schema:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "itemid": {
      "type": "integer"
    }
  },
}
```

```
"requester": {
  "type": "string"
},
"approver": {
  "type": "string"
},
"stagename": {
  "type": "string"
}
},
"required": [
  "requester",
  "itemid",
  "approver",
  "stagename"
]
}
```



3. Add a **Compose** action named **const controller URL**. We still have to generate the controller; in the meantime, store a string "todo" in this action.
4. Add a SharePoint **Get item** action to retrieve you expense details; the id we must be grabbed from the trigger "itemid" parameter:

The screenshot displays a Microsoft Power Automate flow with the following steps:

- When a HTTP request is received** (Trigger)
- const controller url** (Variable)
- Get item** (Action)

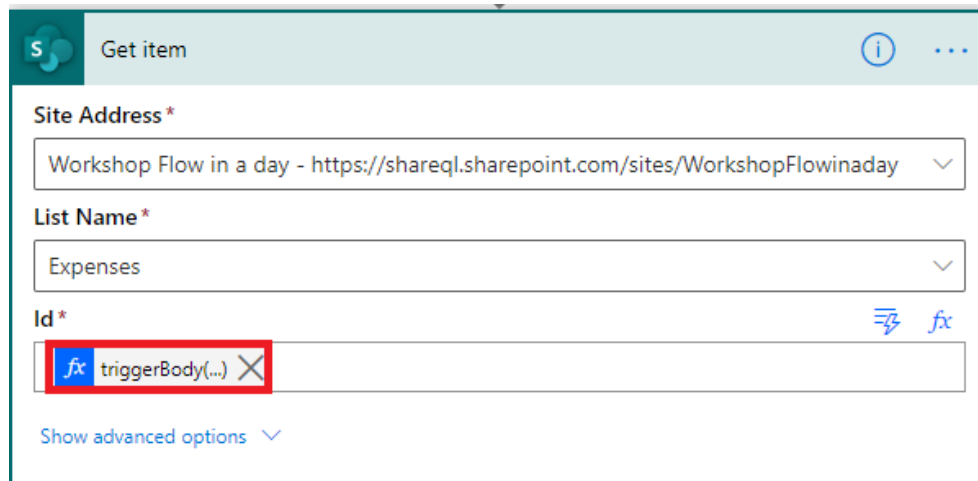
The **Get item** step is expanded, showing the following configuration:

- Site Address \***: Workshop Flow in a day - <https://shareql.sharepoint.com/sites/WorkshopFlowinaday>
- List Name \***: Expenses
- Id \***: `triggerBody()['itemid']`

The **Expression** pane shows the formula `triggerBody()['itemid']`. Below it, the **Outputs** pane lists the following outputs:

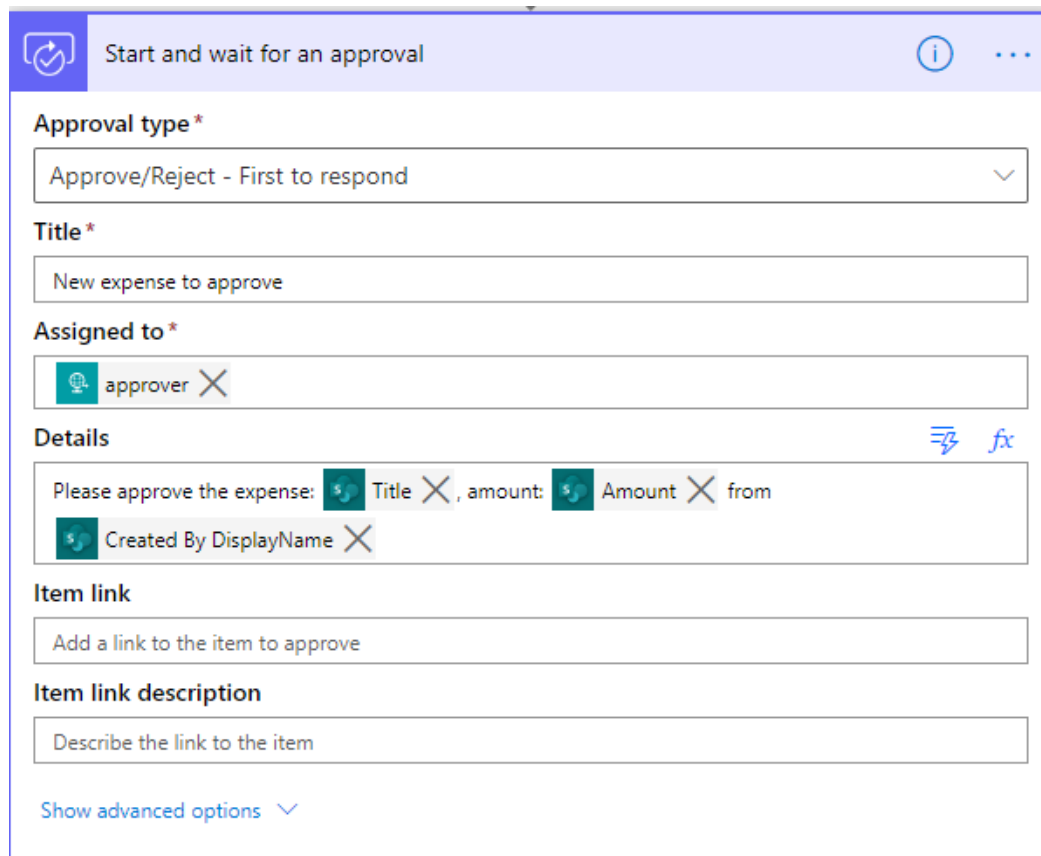
- const controller url**
- Outputs**: Sample data not available
- When a HTTP request is received**
- itemid**: Sample data not available

A red box highlights the expression `triggerBody()['itemid']` in the Expression pane. Another red box highlights the **itemid** output in the Outputs pane, with a red line connecting the two.



The screenshot shows the 'Get item' action configuration window. It has a title bar with a green 'S' icon and the text 'Get item'. Below the title bar, there are three input fields: 'Site Address \*' with a dropdown menu showing 'Workshop Flow in a day - https://shareql.sharepoint.com/sites/WorkshopFlowinaday', 'List Name \*' with a dropdown menu showing 'Expenses', and 'Id \*' with a text input field containing 'triggerBody(...)'. To the right of the 'Id \*' field are two icons: a blue 'fx' icon and a blue 'fx' icon. Below the input fields is a link that says 'Show advanced options' with a downward arrow.

5. Add a **Start and wait for an approval** action and ask the line manager to approve:



The screenshot shows the 'Start and wait for an approval' action configuration window. It has a title bar with a blue icon and the text 'Start and wait for an approval'. Below the title bar, there are several input fields: 'Approval type \*' with a dropdown menu showing 'Approve/Reject - First to respond', 'Title \*' with a text input field containing 'New expense to approve', 'Assigned to \*' with a dropdown menu showing 'approver', 'Details' with a text input field containing 'Please approve the expense: Title, amount: Amount from Created By DisplayName', 'Item link' with a text input field containing 'Add a link to the item to approve', and 'Item link description' with a text input field containing 'Describe the link to the item'. To the right of the 'Details' field are two icons: a blue 'fx' icon and a blue 'fx' icon. Below the input fields is a link that says 'Show advanced options' with a downward arrow.

There are now 2 options: the approver reacts on time, or he does not.

- a) If he reacts on time, the process is completed.
  - b) If he does not react, the approval will be escalated to the big boss.
6. Select the **Start and wait for an approval** action setting and set the timeout to PT2M (2 minutes):

Settings for 'Start and wait for an approval'

Secure Inputs (Preview)  
Secure inputs of the operation.  
Secure Inputs ☐ Off

Secure Outputs (Preview)  
Secure outputs of the operation and references of output properties.  
Secure Outputs ☐ Off

Timeout  
Limit the maximum duration an asynchronous pattern may take. Note: this does not alter the request timeout of a single request.  
Duration ⓘ PT2M

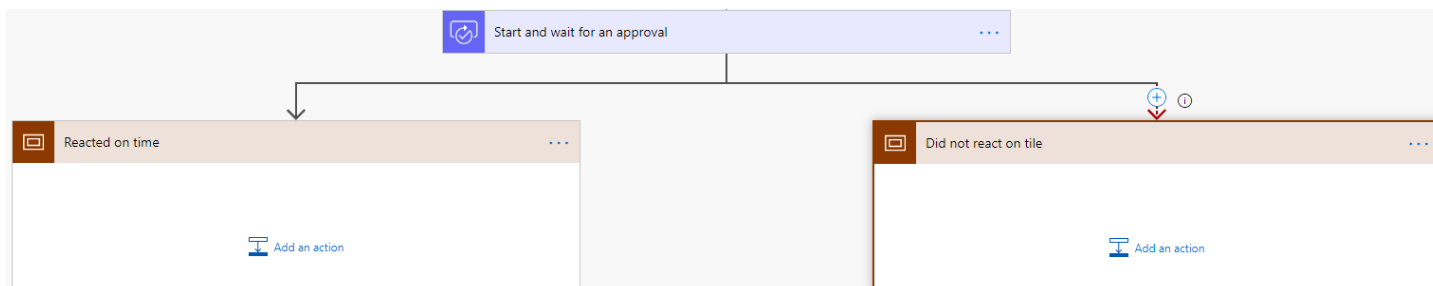
Retry Policy  
A retry policy applies to intermittent failures, characterized as HTTP status codes 408, 429, and 5xx, in addition to any connectivity exceptions. The default is an exponential interval policy set to retry 4 times.  
Type Default

Tracked Properties

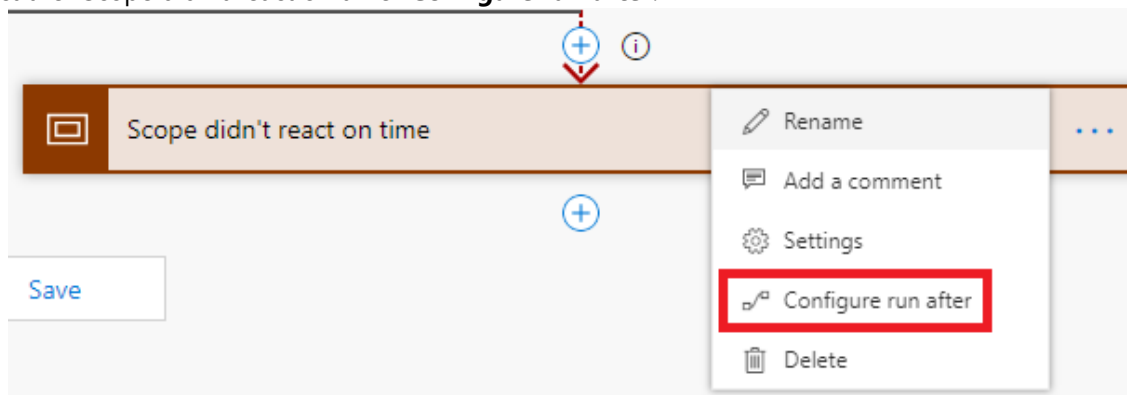
	Properties

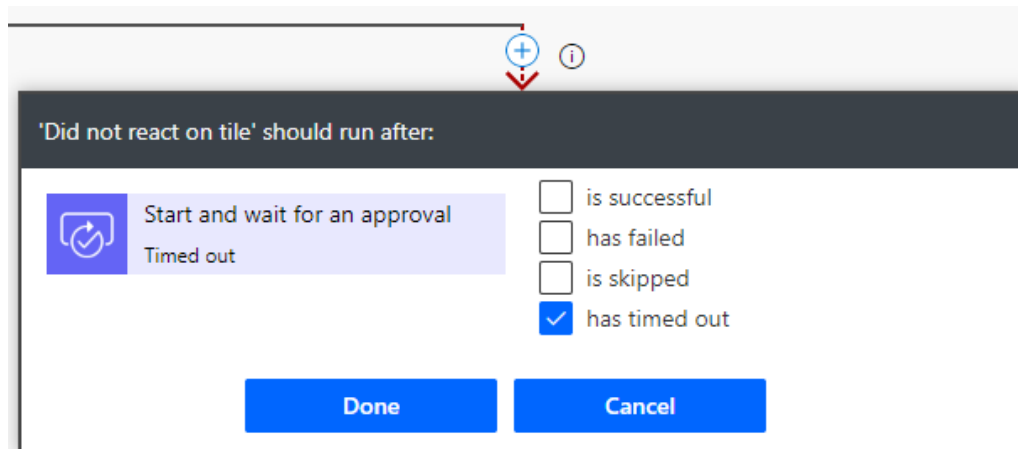
Done Cancel

7. Add a parallel branch with 2 scopes and rename them accordingly:



8. Select the "Scope didn't react on time" **Configure run after**:



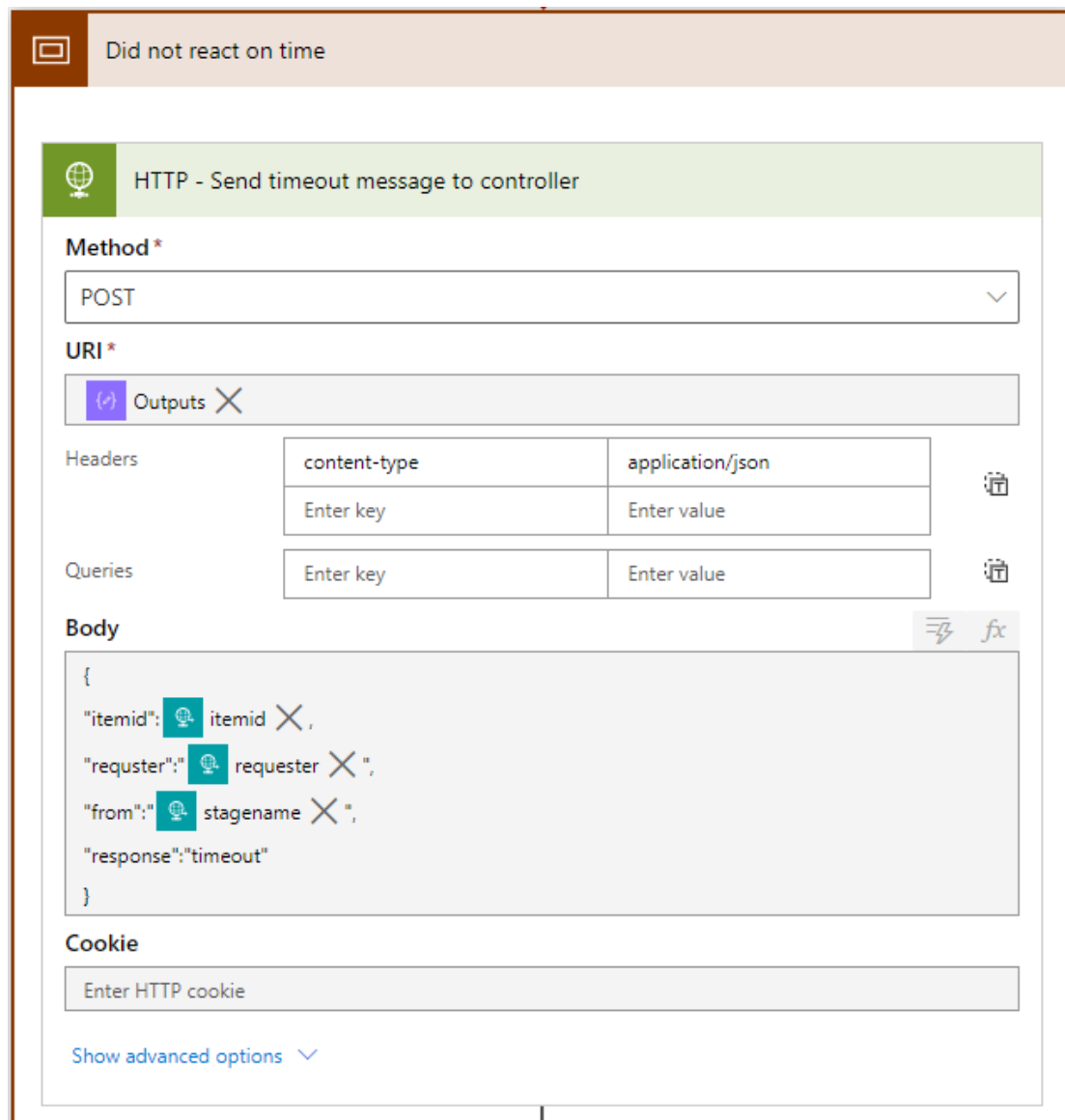


A modal dialog box titled "'Did not react on tile' should run after:". It contains a list of conditions with checkboxes. The condition "has timed out" is selected with a blue checkmark. At the bottom are "Done" and "Cancel" buttons.

Condition	Selected
is successful	<input type="checkbox"/>
has failed	<input type="checkbox"/>
is skipped	<input type="checkbox"/>
has timed out	<input checked="" type="checkbox"/>

9. The check **has timed out**.

10. In the same scope, add an **HTTP** action and set its settings as follows:



The configuration panel for the "HTTP - Send timeout message to controller" action. It includes fields for Method (POST), URI (Outputs), Headers, Queries, Body (JSON), and Cookie.

**Method \***  
POST

**URI \***  
Outputs

**Headers**

content-type	application/json
Enter key	Enter value

**Queries**

Enter key	Enter value
-----------	-------------

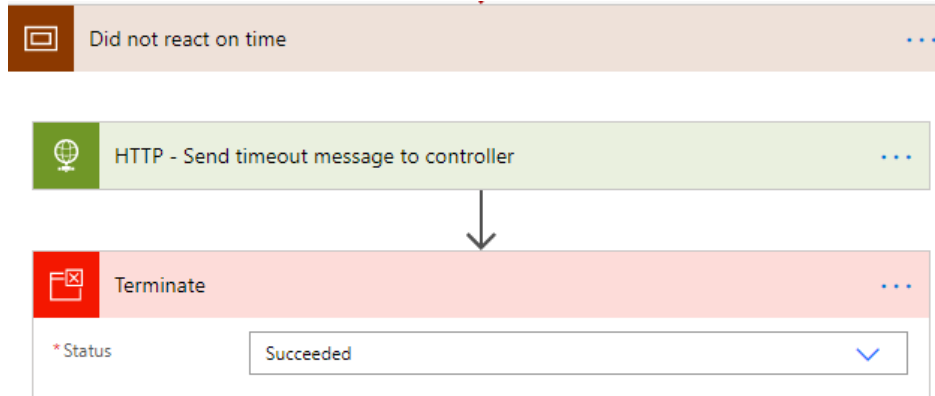
**Body**

```
{
  "itemid": itemid,
  "requester": requester,
  "from": stagename,
  "response": "timeout"
}
```

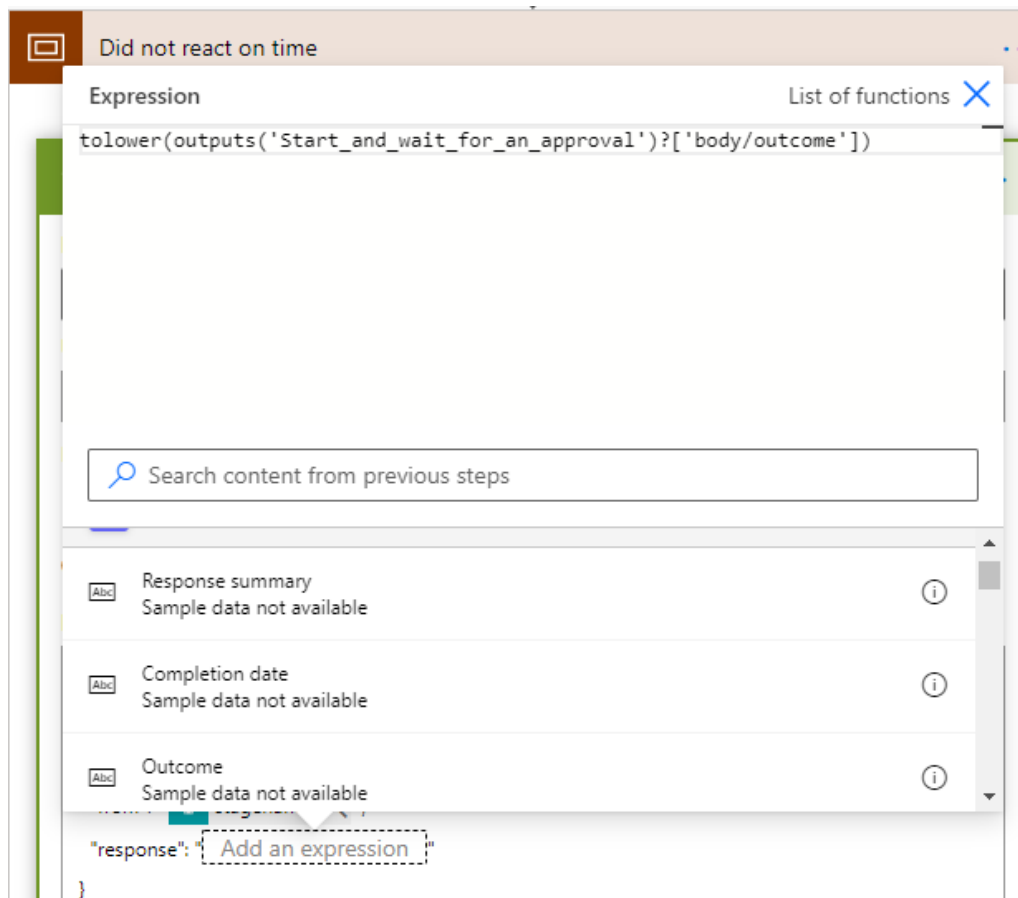
**Cookie**  
Enter HTTP cookie

[Show advanced options](#)


11. Add a **Terminate** action with a Succeeded status:




12. In the other scope, add an **HTTP** action (or copy the one you just created), but this time the “response” value should be grabbed from the Approval outcome:







 Did not react on time

 HTTP - Send timeout message to controller



Method \*

POST


URI \*

 Outputs 









Headers

content-type	application/json		
Enter key	Enter value		

Queries


Enter key	Enter value		
-----------	-------------	--	---

Body

```
{
  "itemid":  itemid ,
  "requester": " requester ",
  "from": " stagename ",
  "response": " tolower(...) "
```

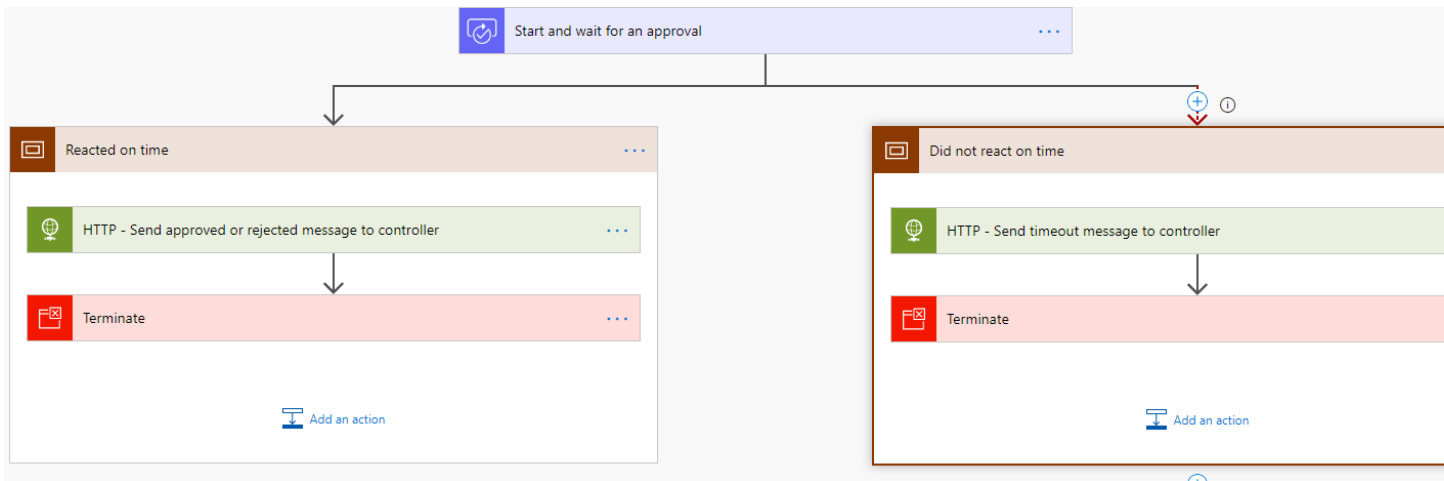
Cookie

Enter HTTP cookie

Show advanced options 

13. Add a Terminate action just after the HTTP action.

14. The 2 branches should look like this:



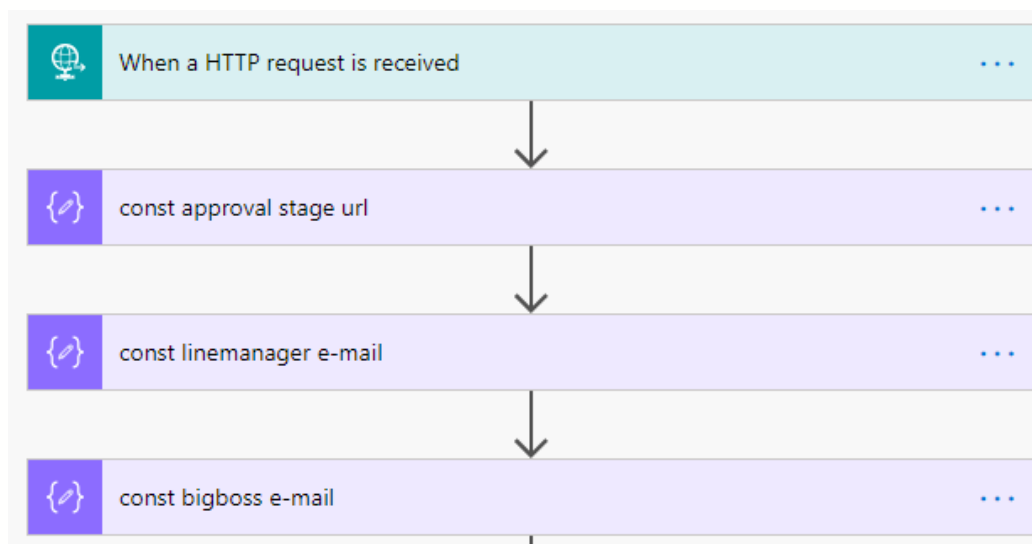
15. Save the Flow.

## Approval Controller Flow implementation

1. We will now create the Flow controller Flow. Create a Flow called **Approval Controller**.
2. The trigger of this Flow must be **When HTTP Request is received**
3. Copy and paste the following schema into the request trigger JSON schema:

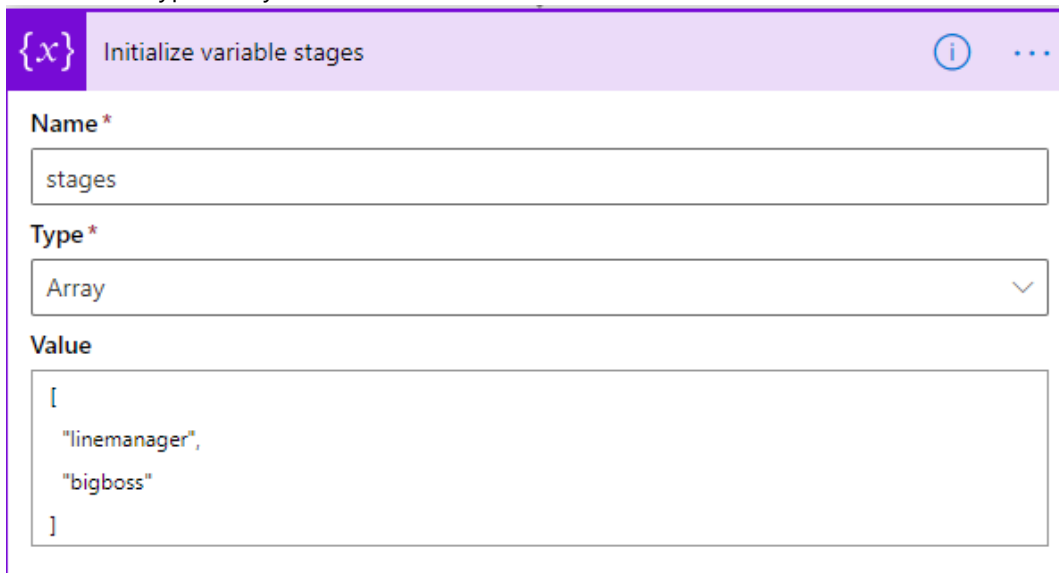
```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "itemid": {
      "type": "integer"
    },
    "requester": {
      "type": "string"
    },
    "from": {
      "type": "string"
    },
    "response": {
      "type": "string"
    }
  },
  "required": [
    "requester",
    "itemid",
    "from"
  ]
}
```

4. Add 3 **Compose** actions and name them as illustrated below:



5. Store the e-mails addresses of the **linemanager** and **bigboss** in the 2 compose actions

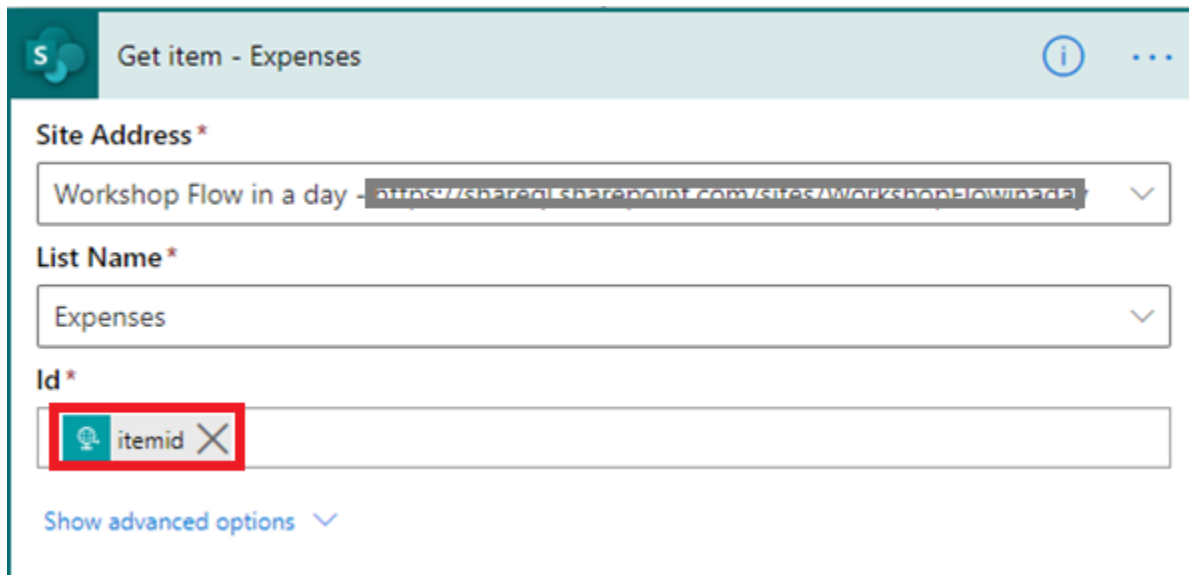
6. Add "todo" in the **const approval stage URL**.
7. Define a variable of type **Array**:



The screenshot shows the 'Initialize variable stages' action in Microsoft Power Automate. The action is represented by a purple header bar with a variable icon and the text 'Initialize variable stages'. Below the header, there are three input fields: 'Name \*' with the value 'stages', 'Type \*' with a dropdown menu set to 'Array', and 'Value' with a text area containing a JSON array: 

```
[  
  "linemanager",  
  "bigboss"  
]
```

8. Add a SharePoint **Get item** action:



The screenshot shows the 'Get item - Expenses' action in Microsoft Power Automate. The action is represented by a teal header bar with a SharePoint icon and the text 'Get item - Expenses'. Below the header, there are three input fields: 'Site Address \*' with a dropdown menu showing 'Workshop Flow in a day - https://github.com/Expenses/WorkshopFlowinaday', 'List Name \*' with a dropdown menu set to 'Expenses', and 'Id \*' with a text area containing 'itemid'. A red box highlights the 'itemid' text. At the bottom, there is a link 'Show advanced options' with a dropdown arrow.

9. Add an **update item** action to change the status of the current expense and rename it **Update expense status**:

Update item - Expenses

Site Address \*

Workshop Flow in a day - [redacted]

List Name \*

Expenses

Id \*

itemid

Title \*

Title

Amount

Status

Being evaluated

Show advanced options

10. After the Update expense status, add a **Switch** action and check the **from** value:

Switch

\*On

from

11. Rename the switch **Check where the call comes from**.

12. In the switch, we will analyze 3 scenarios:

The **from** comes from the **linemanager** approval, from the **bigboss** approval or it is empty:

Check where the call comes from

\*On

from

Linemanager

bigboss

Default

If **from** is empty, we will go into the **Default branch** where we will start the first approval (in this case, line manager approval):

13. Add an **HTTP** request in the default branch:

Default

If no case contains a matching value

The screenshot shows the configuration for an HTTP action titled "HTTP - ask line Manager to approve". The "Method" is set to "POST". The "URI" field contains a dynamic content icon and the text "Outputs". The "Headers" section has a table with one row: "content-type" with the value "application/json". Below it is a row with "Enter key" and "Enter value". The "Queries" section has a row with "Enter key" and "Enter value". The "Body" section contains a JSON object with four properties: "itemid" (dynamic content icon), "requester" (dynamic content icon), "approver" (dynamic content icon and "Outputs"), and "stagename" (fx icon and "variables(...)").

Headers	
content-type	application/json
Enter key	Enter value

Queries	
Enter key	Enter value

```
{
  "itemid": "itemid",
  "requester": "requester",
  "approver": "Outputs",
  "stagename": "variables(...)"
}
```

- In the URI: define the output value of the Compose **const approval stage URL**.
- In the **approver**, define the output of the Compose **const linemanager email**.
- In the stagename, type the expression `variables('stages')[0]`


14. Here is what we must do if the value of from is **"bigboss"**: (**Big Boss Branch**)

We must check if the response is timeout; if that is the case, then we need to call Big Boss again (new HTTP action):


bigboss

\* Equals

bigboss




Check if Big Boss did timeout


 response x


is equal to v

timeout

+ Add v

 If yes

 HTTP - ask Big Boss to approve again ...

 If no

15. The implementation of the HTTP action **ask Big Boss to approve again** is the following:

If yes

HTTP - ask Big Boss to approve again

Method \*

POST

URI \*

Outputs

Headers

content-type	application/json
Enter key	Enter value

Queries

Enter key	Enter value
-----------	-------------

Body

```
{
  "itemid":  itemid,
  "requester": " requester",
  "approver": " Outputs",
  "stagename": " variables(...)"
}
```

Cookie

Enter HTTP cookie


Show advanced options


Where the "stagename" value is the expression: `variables('stages')[1]`

- Let's implement the **LineManager branch**. If the message is timed-out, we must call the big boss again. It is pretty much what we have implemented in the big boss branch:

\* Equals

linemanager



**Check if Linemanager did timeout**





 response x

is equal to

timeout

+ Add v


**If yes**


**HTTP - Ask Big Boss to approve**



**Method \***  
 POST

**URI \***  
 Outputs X

**Headers**

content-type	application/json	X
Enter key	Enter value	

**Queries**

Enter key	Enter value	
-----------	-------------	--

**Body**


```

{
  "itemid": itemid X,
  "requester": requester X,
  "approver": Outputs X,
  "stagename": fx variables(...) X
}

```

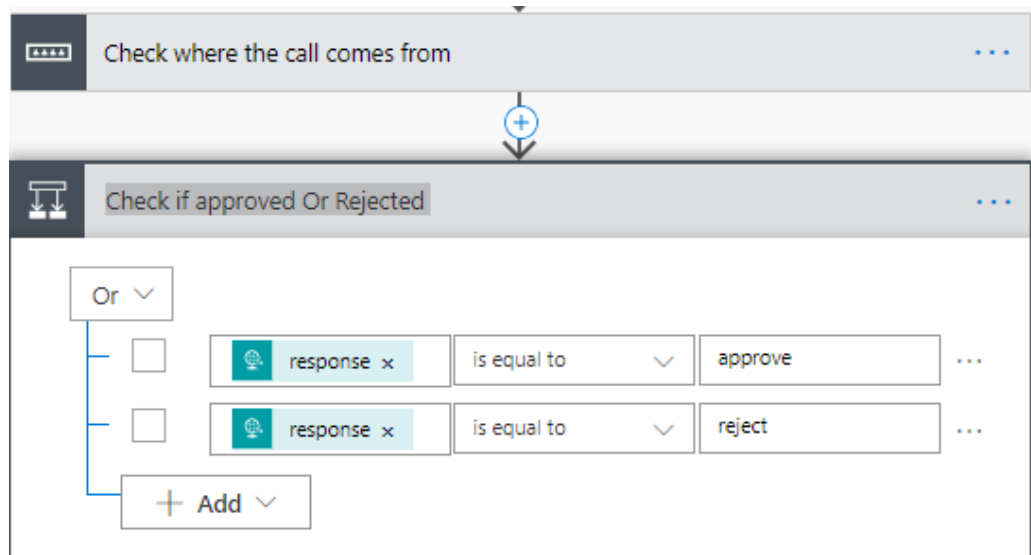
**Cookie**  
 Enter HTTP cookie

[Show advanced options](#) v


**If no**

- After the switch, we test if the response was "approve" or "reject": add a condition and name it **Check if approved Or Rejected**





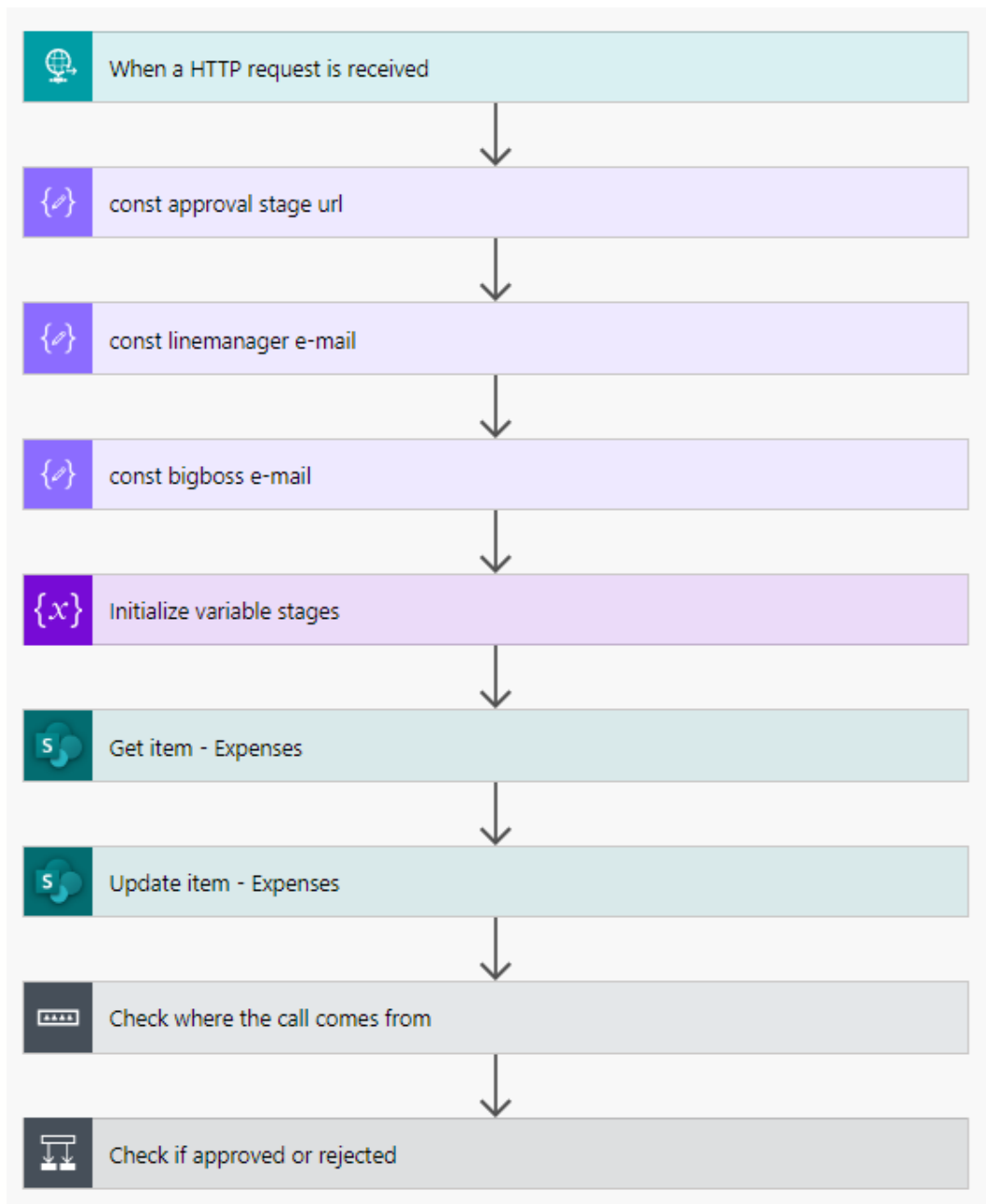
18. In the left branch of the condition update the SharePoint item status:

The screenshot displays two steps from a Microsoft Power Automate flow. The first step, 'Check if approved or rejected', is a conditional step with an 'Or' logic. It contains two conditions: 'response' is equal to 'approved' and 'response' is equal to 'rejected'. Below these conditions is an '+ Add' button. The second step, 'Update item - approval status', is an update step for a SharePoint list. It includes fields for 'Site Address' (set to 'Workshop Flow in a day - https://shareql.sharepoint.com/sites/WorkshopFlowinaday'), 'List Name' (set to 'Expenses'), 'Id' (set to 'itemid'), 'Title' (set to 'Title'), 'Amount' (empty), and 'Status' (set to an 'if(...)' expression). A 'Show advanced options' link is visible at the bottom of the step.

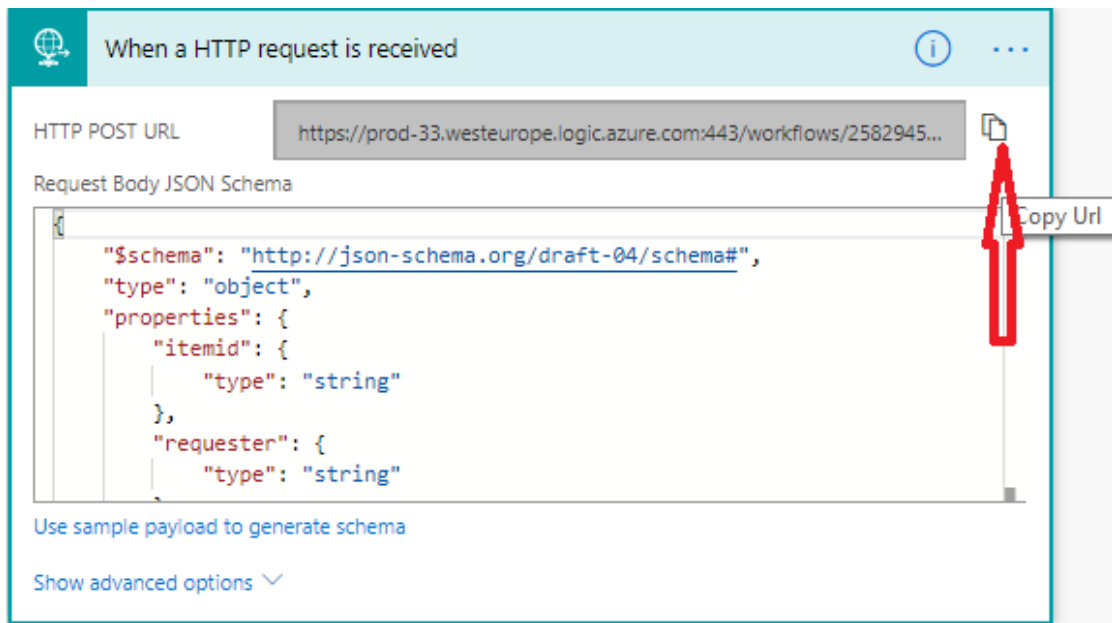
The expression that updates the Status Value is the following:

```
if(equals(triggerBody()?['response'], 'approved'), 'Approved', 'Rejected')
```

Your Flow should look like this:



19. Save the Flow and open it again the generate the associated public URL in the request trigger (and copy the URL):



20. Open the Generic Stage Flow and paste this URL in the **Compose const Controller URL**.
21. Copy the Generic Stage public URL and paste it in the controller Flow (in the Compose const approval stage URL).
22. Test you Flow by adding a new expense in the SharePoint list.

