# Use Azure Logic Apps to Receive Notifications

## **Problem:**

I have multiple Apps registered under App Registration in Azure Active Directory with Application Client Secret configured. This Application Client ID and Application Client Secret are used by my apps to prove its identity when requesting a token for authenticated users with Azure Active Directory to access other resources/applications on Azure. For example, I use these Application Client ID and Application Client Secret to get my app settings (or app secrets) from Azure Key Vault and to access Web APIs etc. The Application Client ID and Application Client Secret are provided in the app settings of my app which I load them while running and get access to resources as mentioned above.

But the downside of the Application Client Secret is that the expiration can be set no more than 2 years (24 months). Whenever the Application Client Secret expires, we must generate a new secret with some expiration and replace the old one in app settings to the new one.

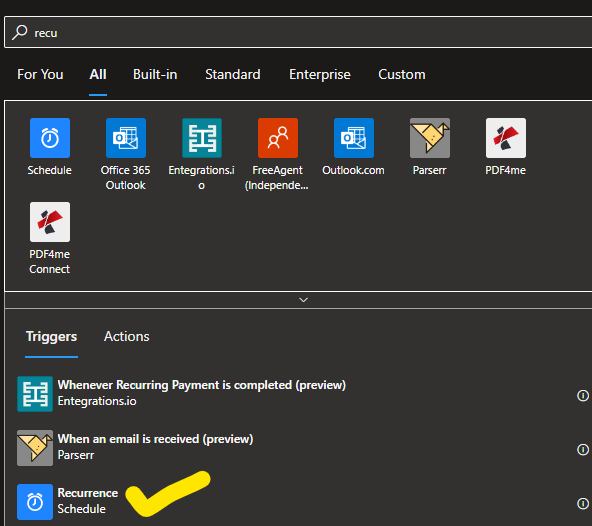
In some cases, we might have apps that are used for long time and more than 2 years. When the Application Client Secret expires on Azure, we don’t receive any notification beforehand about this expiration date which makes the apps fail to authenticate without prior notification to regenerate (imagine you have apps that your customers use, and it breaks suddenly authenticating itself.)

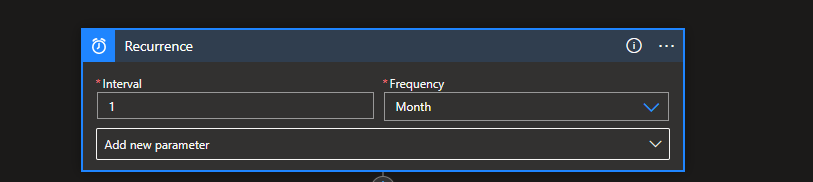
Hence, there is a need to have some sort of process that keeps checking the expiration dates of all the Application Client Secrets of all the apps registered under Azure Active Directory and notify the app owner about upcoming expiration dates of Client Secrets providing the owner an opportunity to regenerate the secret and update their apps.

## **Solution:**

For this case, I used Azure Logic Apps to verify the Application Client Secrets expiration dates and send email alerts to app owner.

1. Create Logic App on Azure. Search for Logic Apps on Portal -> Logic Apps -> Add. I used Plan type as “Consumption” in this case.
2. Open Logic App Designer -> Blank App
3. The first thing I added is Add Recurrence. I set the frequency to repeat the process once per month. The logic app should perform recurring verification with some frequency to repeat the process as Client Secrets change (new ones added, old ones updated etc.)





## **Approach**:

We use Microsoft Graph API, to get all the information about Apps that are registered under Azure App Directory.

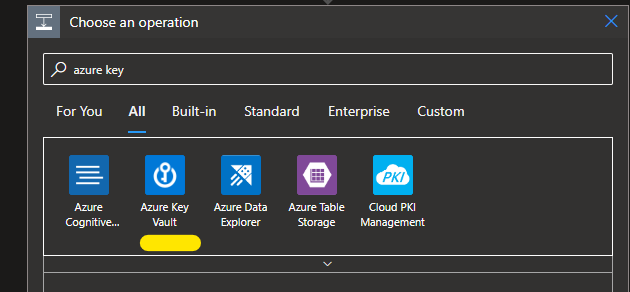
1. Send inputs to authorize our request to Graph API,
2. Get output (registered app details) into variables,
3. Check expiration date for each app in the output from Graph API and send email notification if the expiration date is soon.

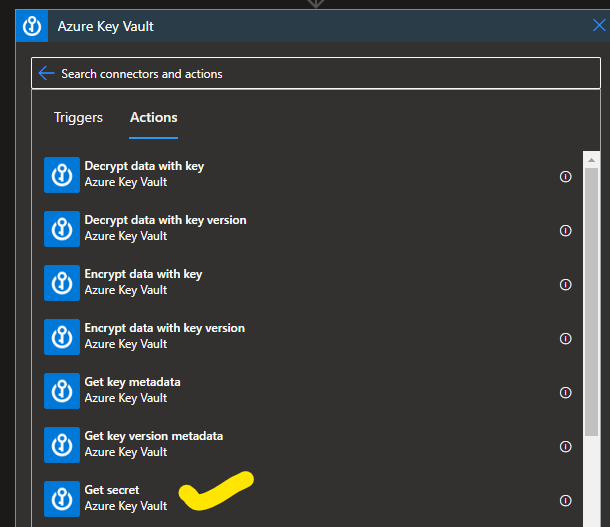
### **Inputs**:

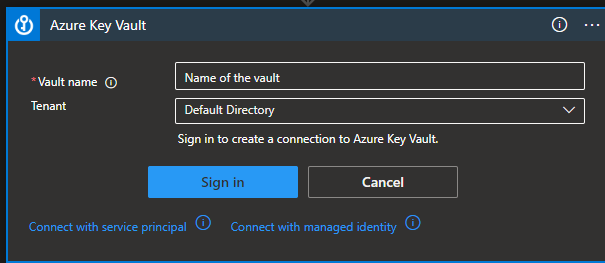
We need Tenant ID, and Application ID and Application Secret of this logic app. For that, we need to register the logic app under Azure Active Directory. (Don’t worry, the expiration date of client secret of this app will also be considered by the logic app to verify and notify user.)

To use these values in my logic app, I created a Key Vault with 3 secrets (Tenant Id, client ID and Client secret). We can always use static values in logic app but having them set in key vault helps for easy updates of any future changes to these secrets instead of updating the whole logic app.

Create next Step, choose Action as Azure Key Vault

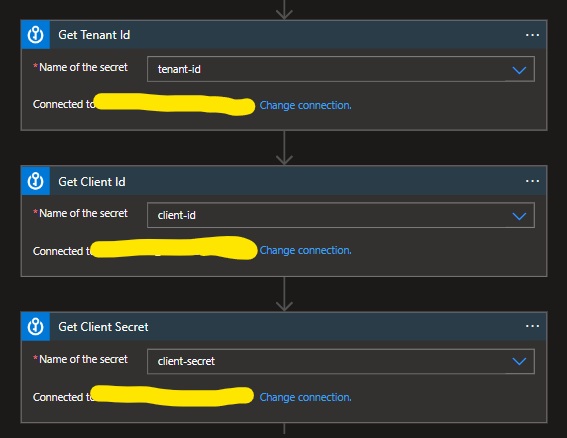




Once you select “Get Secret” Click on Change Connection -> Add New -> Add name of the vault we created above (with 3 secrets) -> Sign In.  


Once done, select Name of the secret from the list.

Repeat this process, by adding action for each secret to load the 2 secrets and it looks like below

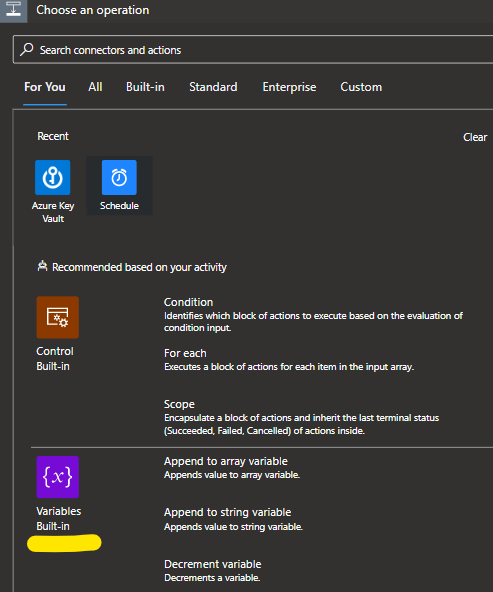


### **Outputs**:

Initialize variables to store required output values from Graph API to be used in our logic.

We need 3 outputs from Graph API – appId, display name, password credentials. (you can create more variables if you want to send more information in the email).

Create Action by selecting “Variables Built-in” -> Initialize variable

Graphical user interface, text, application

Description automatically generated

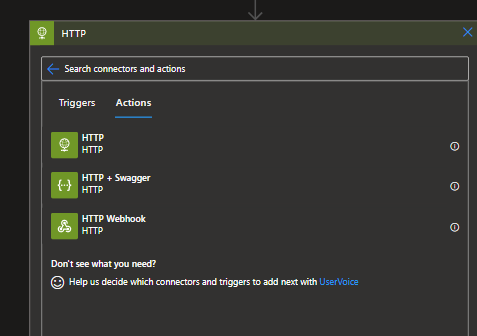
Graphical user interface

Description automatically generated

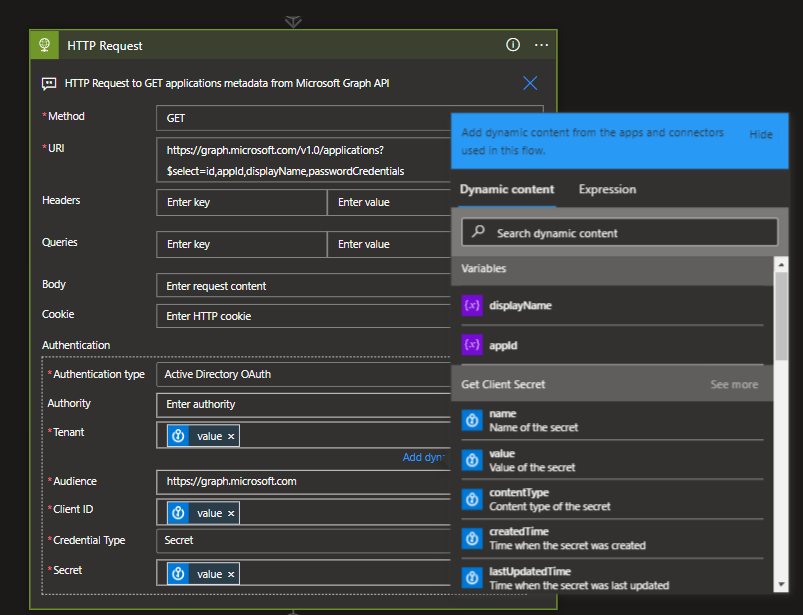
### **Logic**

Request to Graph API:

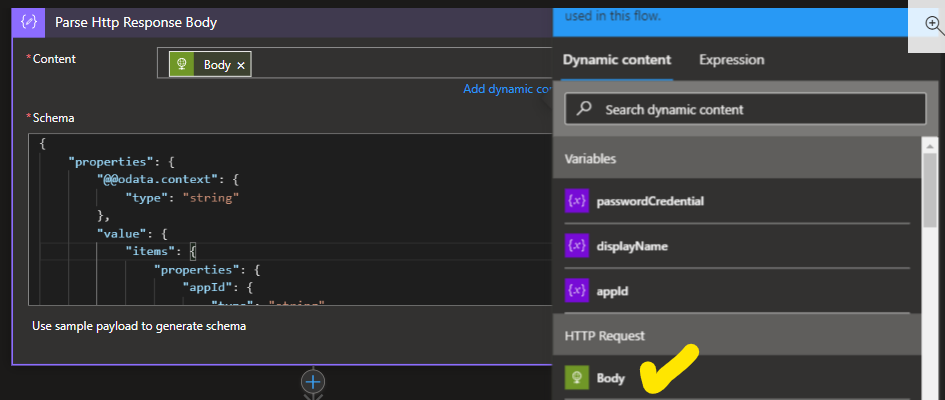
Now create HTTP Request to Graph API. Create Action and Select “HTTP”.



Add following configuration in Http Request. Select Authentication values from previous defined Input variables. (Select “value of the secret” for each input variable)



Parse the HTTP Response – We need to add this step to parse the response from HTTP Request.



Schema: (Just to check)

{

    "properties": {

        "@@odata.context": {

            "type": "string"

        },

        "value": {

            "items": {

                "properties": {

                    "appId": {

                        "type": "string"

                    },

                    "displayName": {

                        "type": "string"

                    },

                    "id": {

                        "type": "string"

                    },

                    "passwordCredentials": {

                        "items": {

                            "properties": {

                                "customKeyIdentifier": {},

                                "displayName": {},

                                "endDateTime": {

                                    "type": "string"

                                },

                                "hint": {},

                                "keyId": {

                                    "type": "string"

                                },

                                "secretText": {},

                                "startDateTime": {

                                    "type": "string"

                                }

                            },

                            "required": [

                                "customKeyIdentifier",

                                "displayName",

                                "endDateTime",

                                "hint",

                                "keyId",

                                "secretText",

                                "startDateTime"

                            ],

                            "type": "object"

                        },

                        "type": "array"

                    }

                },

                "required": [

                    "id",

                    "appId",

                    "displayName",

                    "passwordCredentials"

                ],

                "type": "object"

            },

            "type": "array"

        }

    },

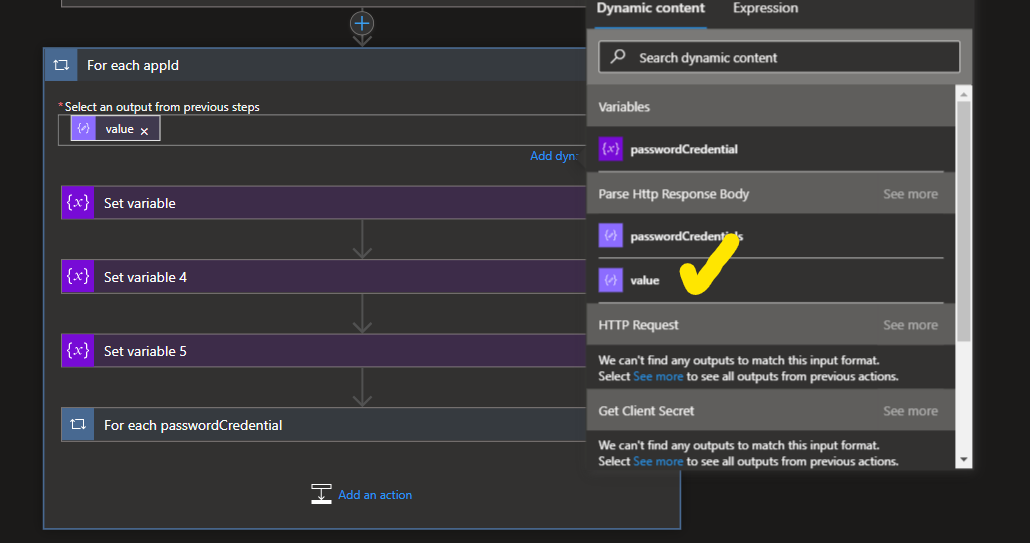
    "type": "object"

}

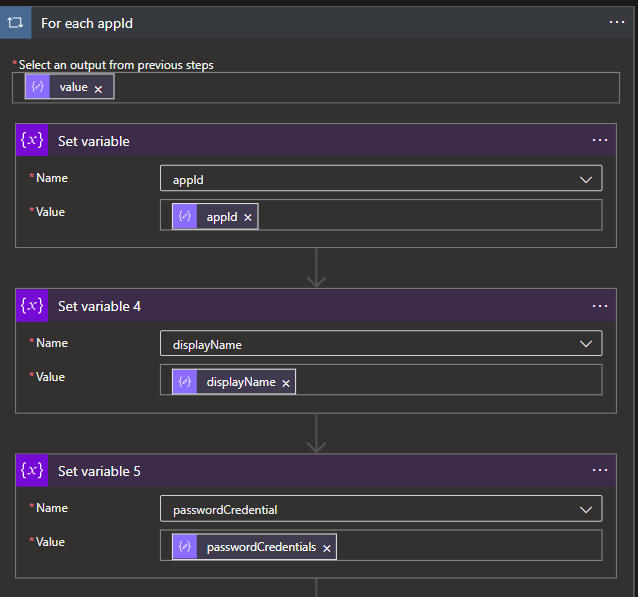
The next step is to add our logic.

Add for each loop to go over each app information we get from HTTP Response.

1. Select “value” from “Parse Http Response Body” step.



Set the following Output variables with values from http Response Body:



Now add another For-each loop inside this For-Each loop to go over Password Credentials Array (which will have expiration date/endDateTime ).

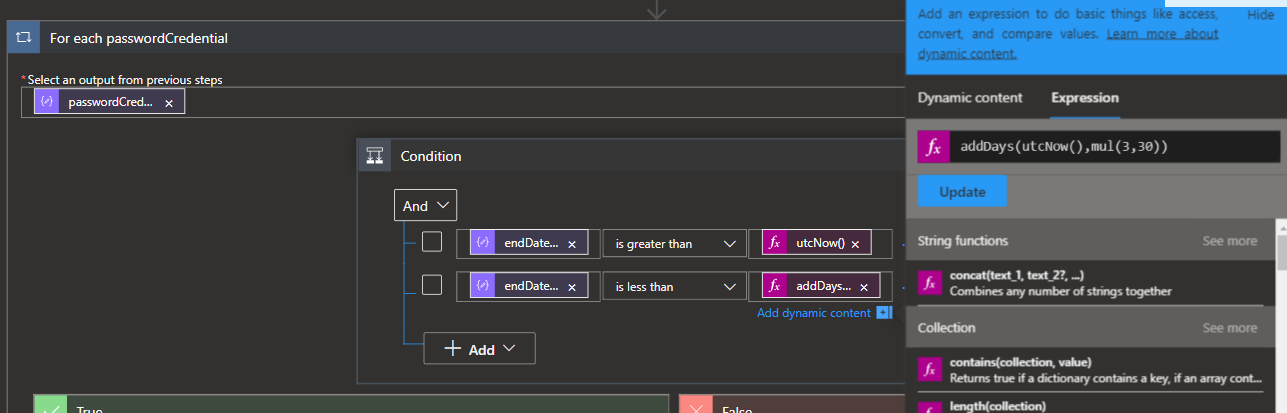
In my case, Condition:

*if (endDateTime > today and endDateTime (expires) within next 3 months)*

*Send email*

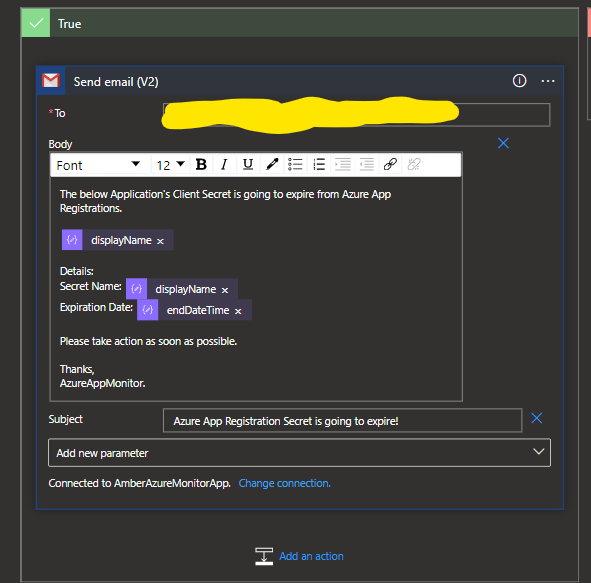
*Else*

*Do Nothing.*



In the above you can see expressions for functions to verify dates.

Send email if condition satisfies.



That’s it, you will get email alerts if your app’s Client Secret is going to expire soon.