



Microsoft Power Platform

Dev in a day

Lab 04 Azure Function/ May 2022

Table of Contents

Lab Scenario	1
Exercise 1 – Create Azure Function.....	1
Task 1: Install Azure tools extension	1
Task 2: Create function	2
Exercise 2 - Function implementation	4
Task 1: Implement function	4
Exercise 3 – Publish to Azure	9
Task 1: Publish.....	9
Task 2: Register Connector Client app	16
Exercise 3 – Create Connector	17
Task 1: Create connector	17
Task 2: Test connector	21
Exercise 4 – Use Function from Canvas App.....	22
Task 1: Use function.....	22
Task 2: Test application.....	25

Lab Scenario

Working as part of the PrioritZ fusion team you will be configuring a custom connector for a new API you build using Azure Functions. The team has decided to move the logic when a user creates a new “ask” to the Azure Function API. This will keep the Power App formula simple and allow more complex logic to be added in the future. In this lab you will create the function, use the Dataverse API, secure the API with Azure AD, configure a custom connector to use the API, and change the Power App to use the connector.

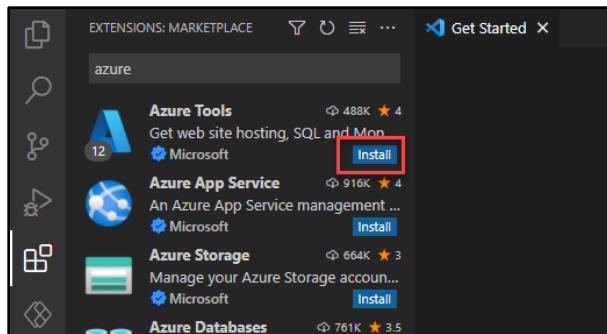
Note: This lab requires an Azure subscription (or trial) in the same tenant as your Dataverse environment.

Exercise 1 – Create Azure Function

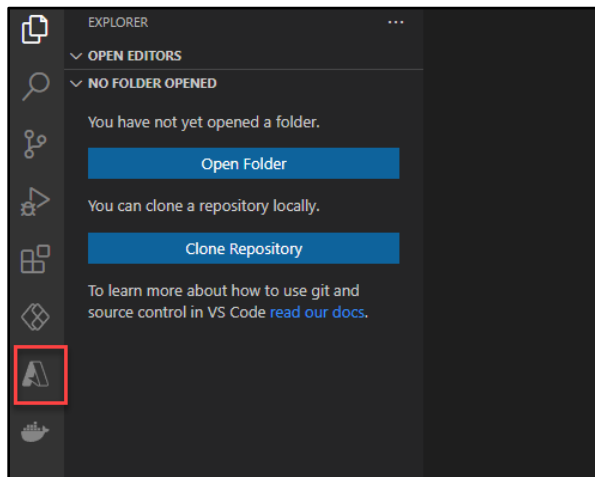
In this exercise, you install Azure tools extension for Visual Studio Code and create the function

Task 1: Install Azure tools extension

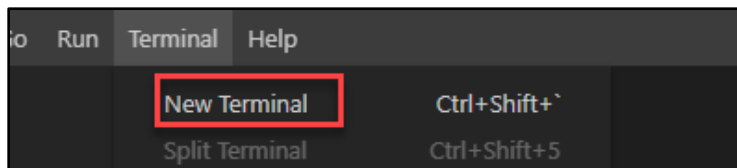
1. Start Visual Studio Code.
2. Select the **Extensions** tab.
3. Search for Azure and click **Install Azure Tools**.



4. Wait for the installation to complete.
5. You should now see the new Azure Tools extension you added.

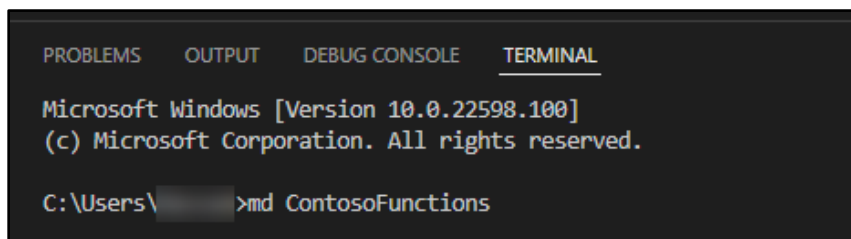


6. Click **Terminal** and select **New Terminal**.



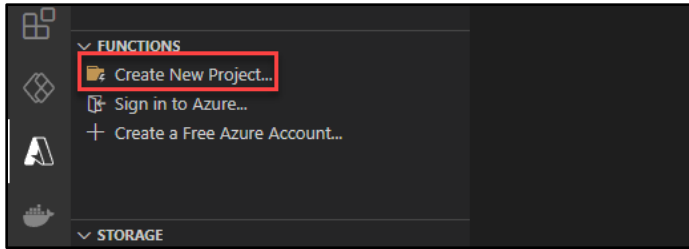
7. Go to the terminal and run the command below to create new folder.

```
md ContosoFunctions
```

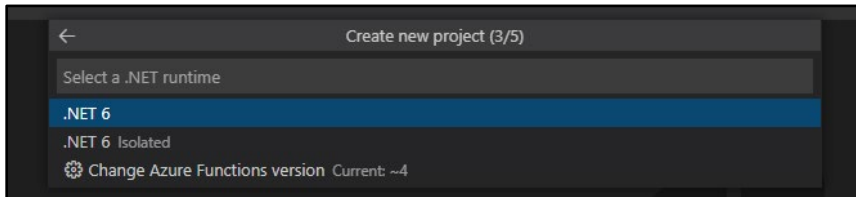


Task 2: Create function

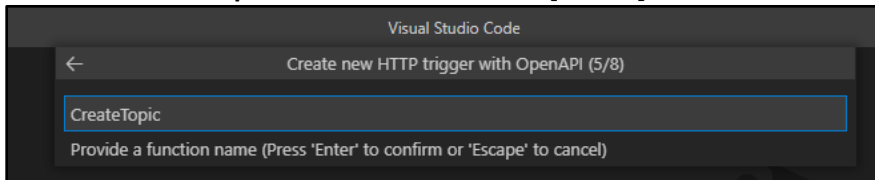
1. Select **Azure Tool**, go to the **Functions** section, and click **Create New Project**.



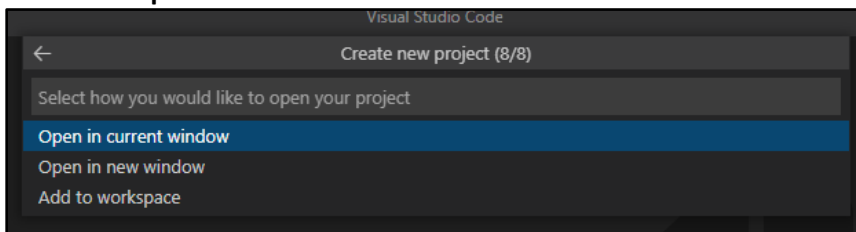
2. Select the **ContosoFunctions** folder you created and click **Select**.
3. Select **C#** for language.
4. Select **.NET 6** for .NET runtime.



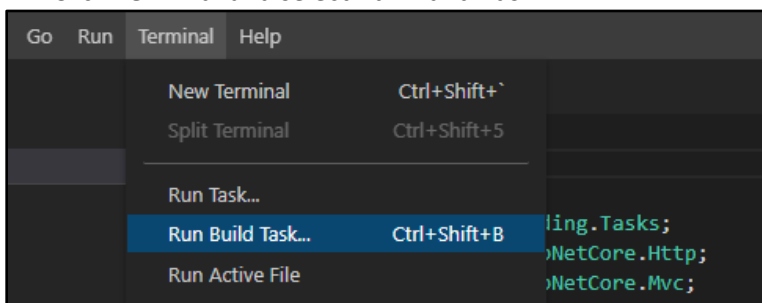
5. Select **HTTP trigger with OpenAPI** for template.
6. Enter **CreateTopic** for function name and **[ENTER]**.



7. Enter **Contoso.PrioritZ** for namespace and **[ENTER]**
8. Select **Anonymous** for AccessRights. Later we will protect the function using Azure AD.
9. Select **Open in current window**.



10. Your function should open in Visual Studio Code.
11. Click **Terminal** and select **Run Build Task**.



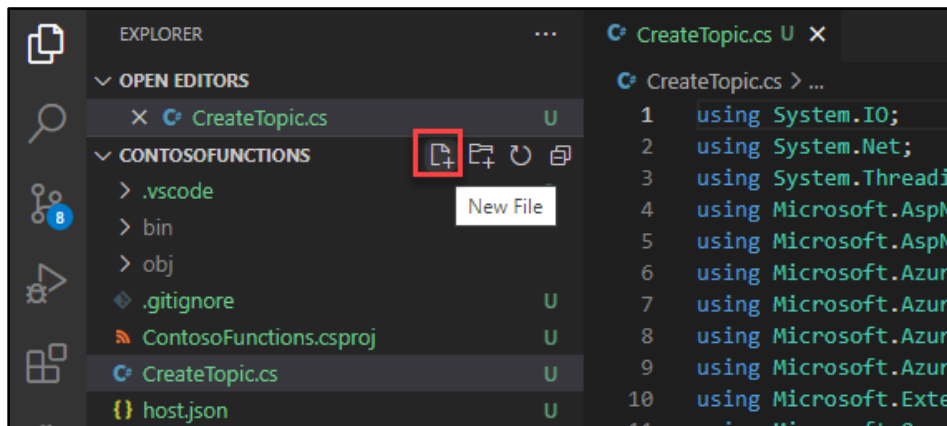
12. The build should succeed.
13. Go to Terminal and press any key close it.

Exercise 2 - Function implementation

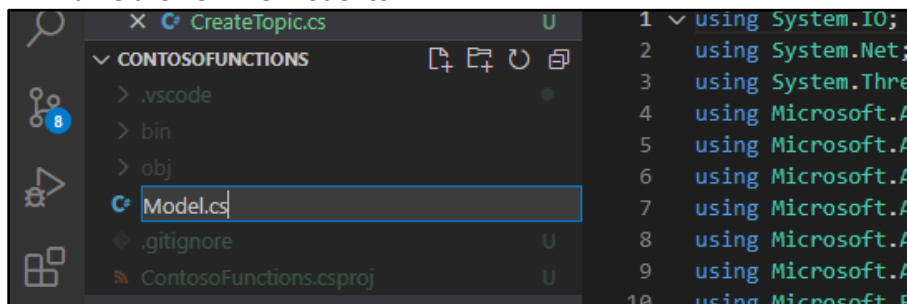
In this exercise, you will implement the function.

Task 1: Implement function

1. Click **New file**.



2. Name the new file **Model.cs**



3. Open the new **Model.cs** file and paste the code below. This will define the data that will be sent from the Power App.

```
using System;
using Microsoft.Azure.WebJobs.Extensions.OpenApi.Core.Attributes;
using Microsoft.OpenApi.Models;
namespace Contoso.PrioritZ
{
    public class TopicItemModel
    {
        public string Choice { get; set; }
        public string Photo { get; set; }
    }
    public class TopicModel
```

```

{
    [OpenApiProperty(Nullable = false, Description = "This is a topic")]
    public string Topic { get; set; }
    public string Details { get; set; }
    public DateTime RespondBy { get; set; }
    public string MyNotes { get; set; }
    public string Photo { get; set; }
    public TopicItemModel[] Choices {get;set;}
}
}

```

4. Open the **CreateTopic** file.
5. Locate the Run method attributes and replace them with the attributes below. This provides user friendly names when we create a connector to use the API.

```

[FunctionName("CreateTopic")]
[OpenApiOperation(operationId: "CreateTopic", tags: new[] { "name" }, Summary =
"Create Topic", Description = "Create Topic", Visibility =
OpenApiVisibilityType.Important)]
[OpenApiSecurity("function_key", SecuritySchemeType.ApiKey, Name = "code", In =
OpenApiSecurityLocationType.Query)]
[OpenApiResponseWithBody(statusCode: HttpStatusCode.OK, contentType:
"application/json", bodyType: typeof(Guid), Description = "The Guid response")]
[OpenApiRequestBody(contentType: "application/json", bodyType:
typeof(TopicModel))]

```

```

[FunctionName("CreateTopic")]
[OpenApiOperation(operationId: "CreateTopic", tags: new[] { "name" }, Summary = "Create Topic", Description = "Create Topic", Visibility = OpenApiVisibilityType.Important)]
[OpenApiSecurity("function_key", SecuritySchemeType.ApiKey, Name = "code", In = OpenApiSecurityLocationType.Query)]
[OpenApiResponseWithBody(statusCode: HttpStatusCode.OK, contentType: "application/json", bodyType: typeof(Guid), Description = "The Guid response")]
[OpenApiRequestBody(contentType: "application/json", bodyType: typeof(TopicModel))]
0 references
public async Task<IActionResult> Run(

```

6. Remove **get** from the Run method. You should only have post.

```

[OpenApiResponseWithBody(statusCode: HttpStatusCode.OK, contentType: "application/json", bodyType: typeof(Guid), Description = "The Guid response")]
[OpenApiRequestBody(contentType: "application/json", bodyType: typeof(TopicModel))]
0 references
public async Task<IActionResult> Run(
    [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route = null)] HttpRequest req)
{

```

7. Go to the **Terminal** and add **Power Platform Dataverse Client** package.

```
dotnet add package Microsoft.PowerPlatform.Dataverse.Client
```

```
Microsoft Windows [Version 10.0.22598.100]
(c) Microsoft Corporation. All rights reserved.

C:\Users\...\ContosoFunctions>dotnet add package Microsoft.PowerPlatform.Dataverse.Client
```

8. Wait for the package to be added.
9. Add **Azure Identity** package.

```
dotnet add package Azure.Identity
```

10. Wait for the package to be added.
11. Open the **CreateTopic** file and add the using statements below.

```
using System;
using Microsoft.Identity.Client;
using Azure.Core;
using Azure.Identity;
using Microsoft.PowerPlatform.Dataverse.Client;
using Microsoft.Azure.WebJobs.Extensions.OpenApi.Core.Enums;
using Microsoft.Xrm.Sdk;
```

12. Add the below method after the Run method. This method will use the token passed from the calling app to get a new token that will allow the function to use the Dataverse API on behalf of the calling user.

```
public static async Task<string> GetAccessTokenAsync(HttpRequest req,string
resourceUri)
{
    //Get the calling user token from the request to use as
    UserAssertion
    var headers = req.Headers;
    var token = string.Empty;
    if (headers.TryGetValue("Authorization", out var authHeader))
    {
        if (authHeader[0].StartsWith("Bearer "))
        {
            token = authHeader[0].Substring(7, authHeader[0].Length -
7);
        }
    }

    string[] scopes = new[] { $"{resourceUri}/.default" };
}
```

```

        string clientSecret =
Environment.GetEnvironmentVariable("ClientSecret");
        string clientId = Environment.GetEnvironmentVariable("ClientID");
        string tenantId = Environment.GetEnvironmentVariable("TenantID");

        var app = ConfidentialClientApplicationBuilder.Create(clientId)
            .WithClientSecret(clientSecret)
            .WithAuthority($"https://login.microsoftonline.com/{tenantId}")
            .Build();

        //Get On Behalf Of Token for calling user
        UserAssertion userAssertion = new UserAssertion(token);
        var result = await app.AcquireTokenOnBehalfOf(scopes,
userAssertion).ExecuteAsync();

        return result.AccessToken;
    }

```

13. Replace the code inside the **Run** method with code below. This will get an instance of the Dataverse API and use the GetAccessToken function we just defined.

```

_logger.LogInformation("Starting Create Topic");

var serviceClient = new ServiceClient(
    instanceUrl: new Uri(Environment.GetEnvironmentVariable("DataverseUrl")),
    tokenProviderFunction: async uri => { return await
GetAccessTokenAsync(req, Environment.GetEnvironmentVariable("DataverseUrl"));
},
    useUniqueInstance: true,
    logger: _logger);

if (!serviceClient.IsReady)
{
    throw new Exception("Authentication Failed!");
}

```

14. Add the following code after the if statement of the **Run** method to reserialize the request. This will get us the data passed from the caller.

```

string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
var data = JsonConvert.DeserializeObject<TopicModel>(requestBody);

```


15. Add the code below that will create the row to the **Run** method. This code creates the rows in Dataverse and is where we might add more logic in the future.

```
var ask = new Entity("contoso_prioritztopic");
ask["contoso_topic"] = data.Topic;
ask["contoso_details"] = data.Details;
ask["contoso_mynotes"] = data.MyNotes;
ask["contoso_respondby"] = data.RespondBy.Date;
if (data.Photo != null)
{
    // Remove unnecessary double quotes,
    // Remove everything before the first comma (embedded stuff)
    ask["contoso_photo"] =
Convert.FromBase64String(data.Photo.Trim('"').Split(',')[1]);
}
var topicId = await serviceClient.CreateAsync(ask);

foreach (var choice in data.Choices)
{
    var item = new Entity("contoso_prioritztopicitem");
    item["contoso_choice"] = choice.Choice;
    item["contoso_prioritztopic"] = new
EntityReference("contoso_prioritztopic", topicId);
    if (choice.Photo != null)
    {
        item["contoso_photo"] =
Convert.FromBase64String(choice.Photo.Trim('"').Split(',')[1]);
    }

    var choiceId = await serviceClient.CreateAsync(item);
}
```

16. Return the topic id as JSON (required by Power Apps). Add the code below to the **Run** method.

```
return new OkObjectResult(topicId);
```

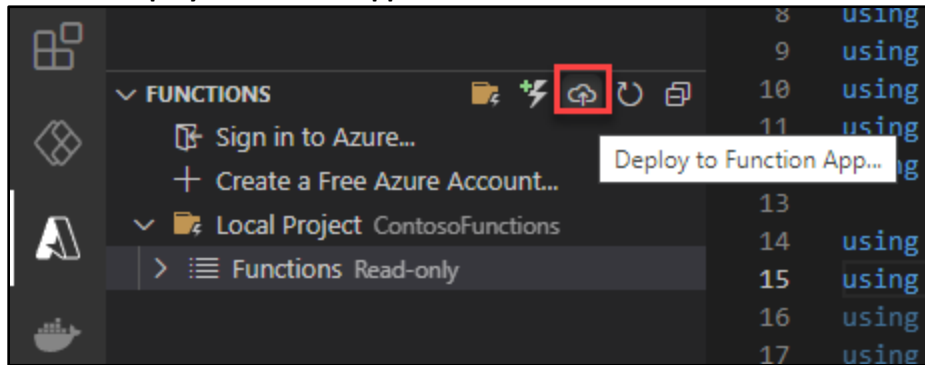
17. Click **Terminal** and select **Run Build Task**.
18. The run should succeed. Press any key to stop.

Exercise 3 – Publish to Azure

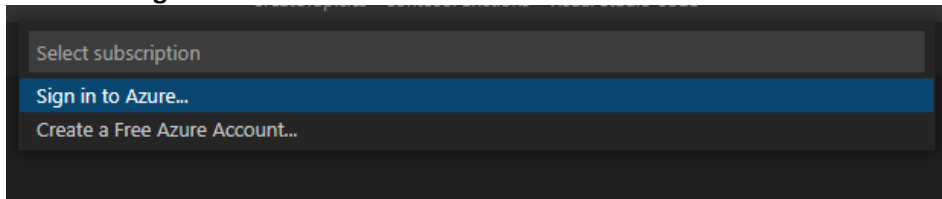
In this exercise, you will deploy the function to Azure.

Task 1: Publish

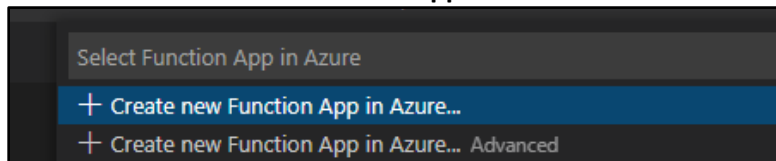
1. Select **Azure Tools**.
2. Click **Deploy to Function App**.



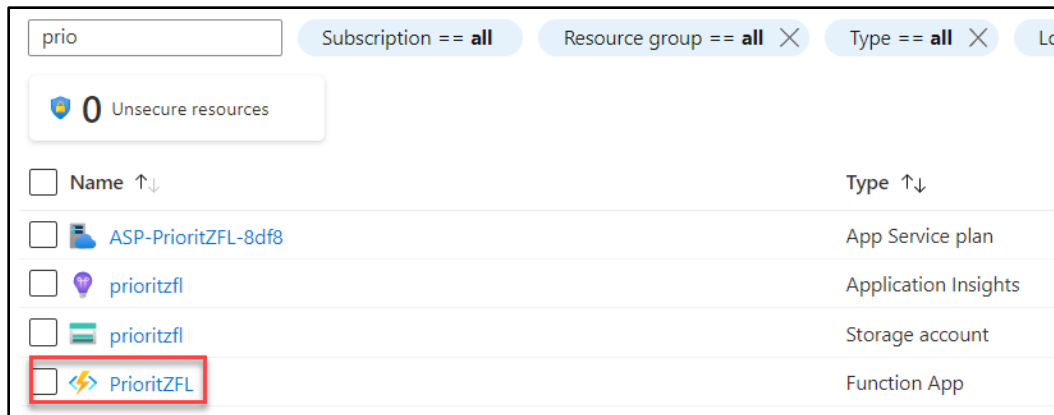
3. Select **Sign in to Azure**.



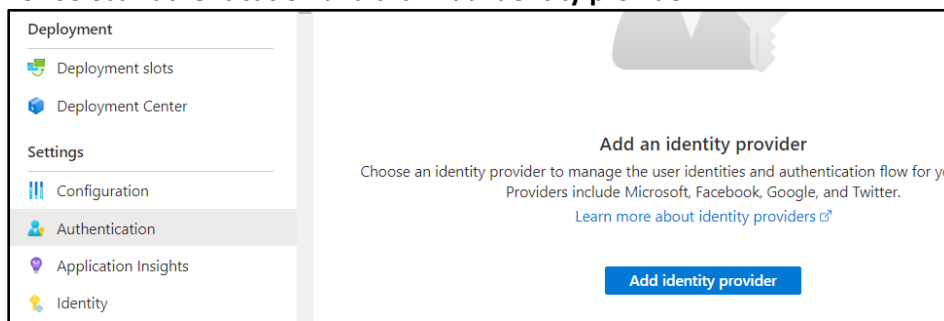
4. Sign into an Azure account that has a subscription or create a trial.
5. Close the sign in browser window.
6. Select **+ Create new Function App**.



7. Enter **PrioritZFuncFL** for function app name and [ENTER]. Replace FL with your initials.
8. Select **.NET 6**.
9. Select your location.
10. Wait for the function to be deployed.
11. Navigate to <https://portal.azure.com/>
12. Select **All resources**, search for the function app you deployed and click to open it.

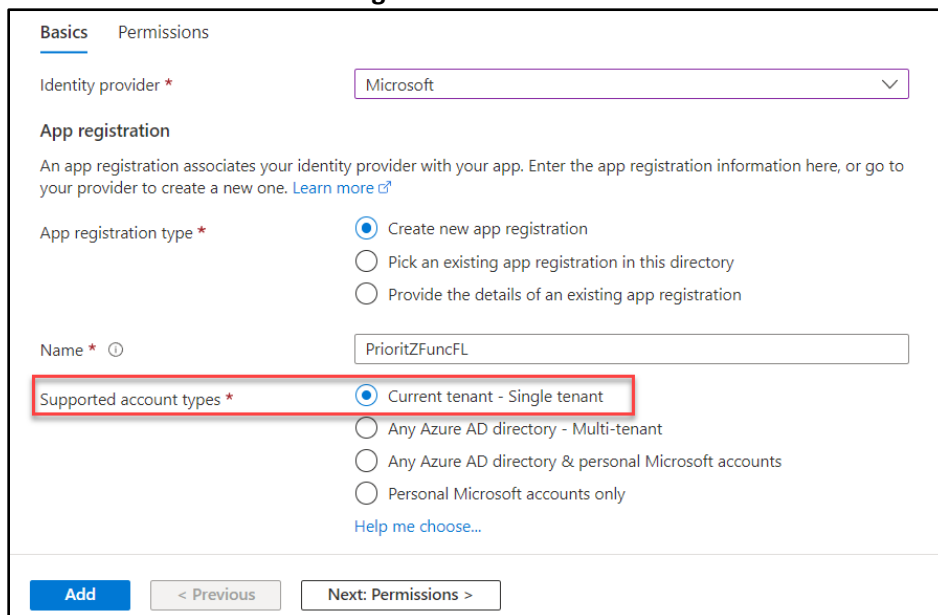


13. Select **Authentication** and click **Add identity provider**.



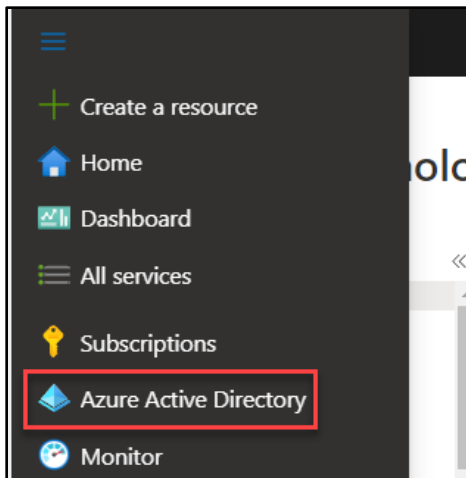
14. Select **Microsoft** for Identity provider,

15. Select **Current tenant - Single tenant** and click **Add**.



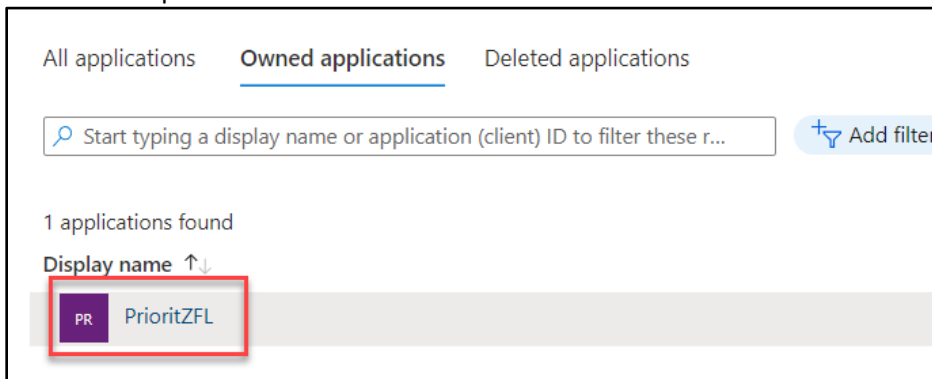
16. Go to the **Portal menu** page of Azure portal.

17. Select **Azure Active Directory**.

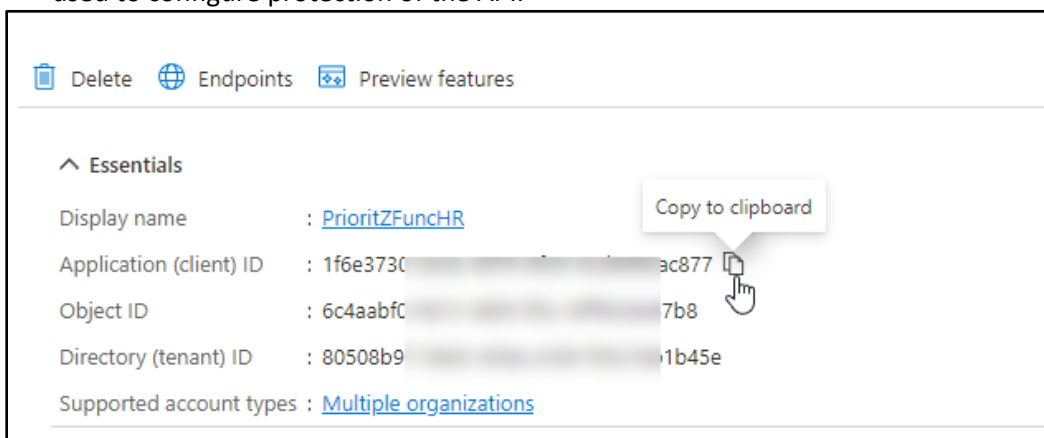


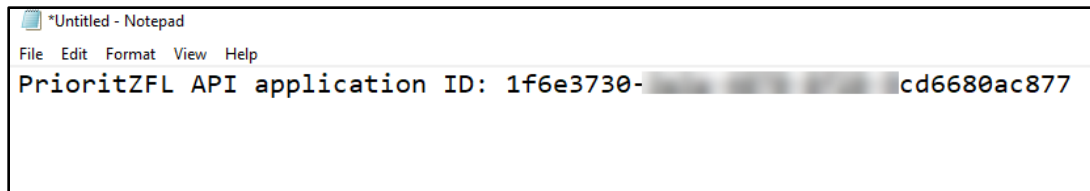
18. Select **App registrations**.

19. Click to open the **PrioritZFL**.

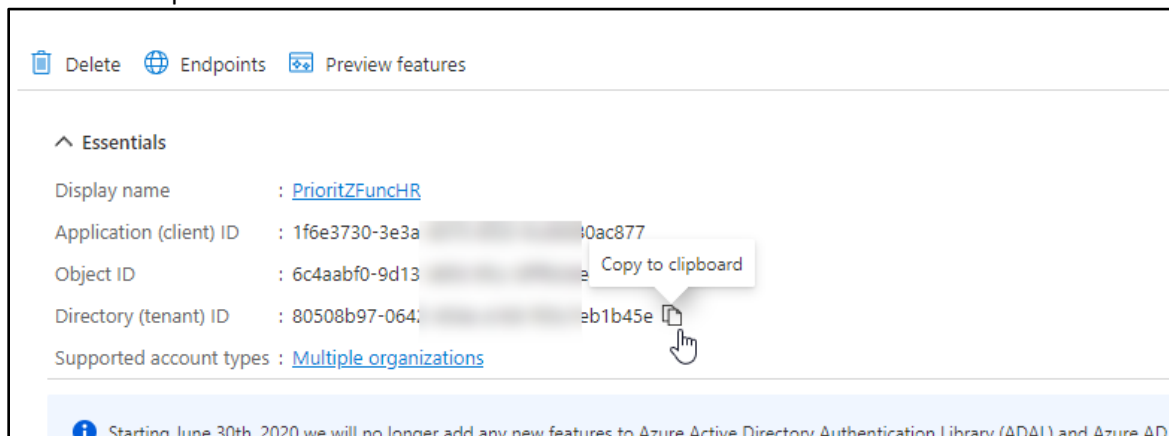


20. Copy the **Application (client) ID** of the **PrioritZFL** application registration and keep it on a notepad as **PrioritZFL API application ID**. You will need this id in future steps. This ID will be used to configure protection of the API.



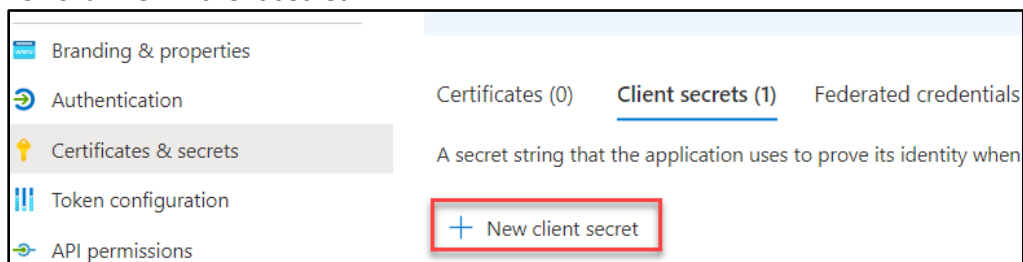


21. Copy the **Directory (tenant) ID** and keep it on a notepad as **Tenant ID**. You will need this id in future steps.

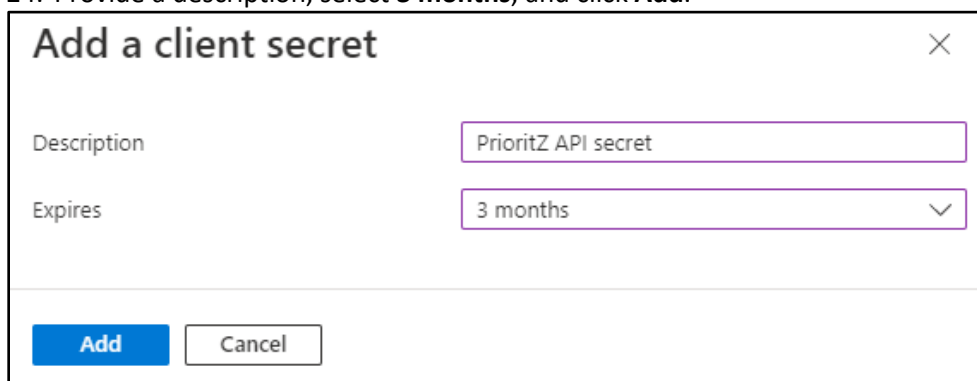


22. Select **Certificates & secrets**.

23. Click **New + client secret**.



24. Provide a description, select **3 months**, and click **Add**.

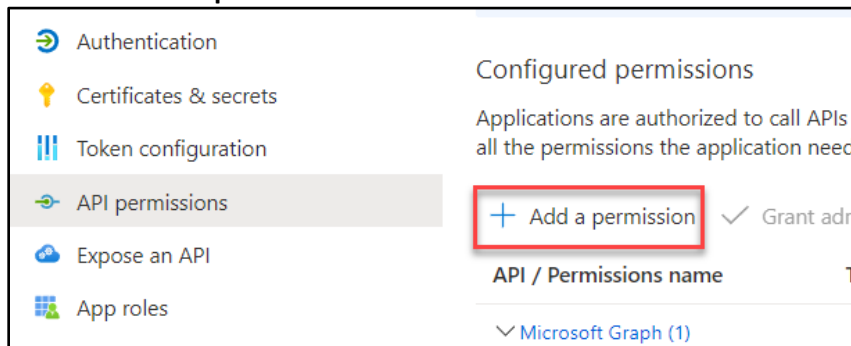


25. Copy the **Value** and keep it in a notepad as **PrioritZFL API Secret**. You need this value in future steps.

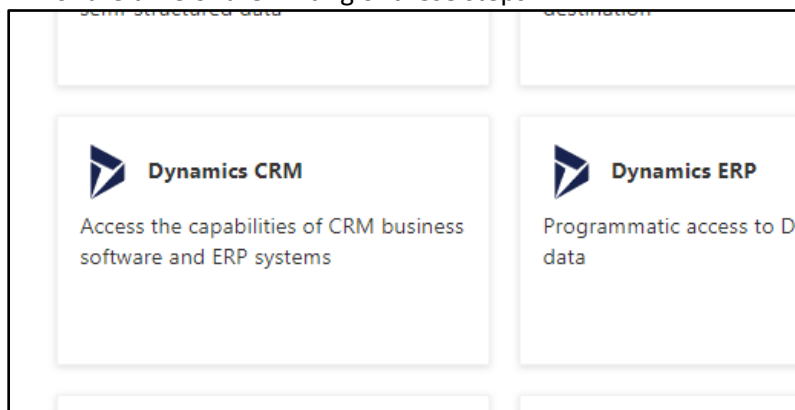
+ New client secret			
Description	Expires	Value	Copy to clipboard et ID
Prioritz API secret	7/20/2022	4B88Q~fF96yuCkD3c16czmkGwV.exCER...	92593e21-07a2-4d7b-9d92-7bb5461cd6...
Generated by App Service	4/20/2032	1Lg*****	9f360c3f-8bfe-4256-9389-18c059333c6a

26. Select **API permissions**.

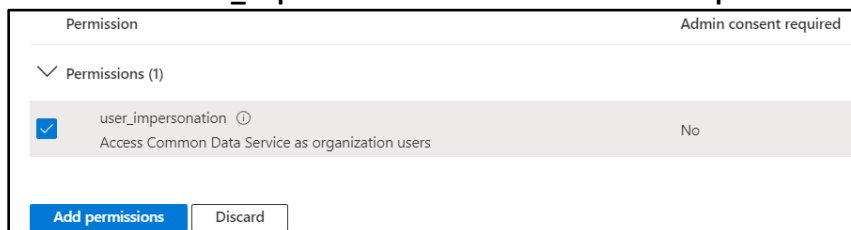
27. Click + **Add a permission**.




28. Select **Dynamics CRM**. Dynamics CRM is Dataverse, the Azure portal just hasn't been updated as of the time of the writing of these steps.



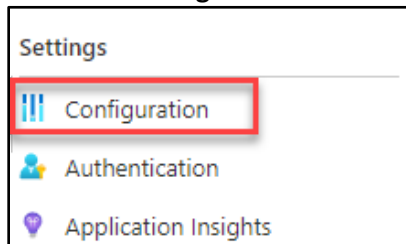
29. Check the **user_impersonation** checkbox and click **Add permission**.



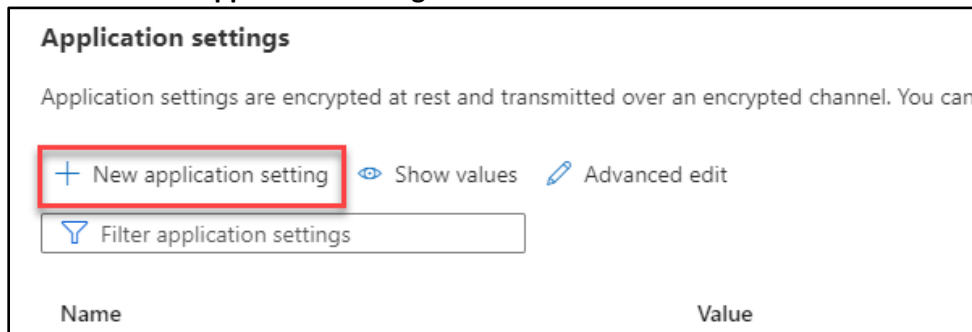
30. Go back to **Home** and open the **PrioritzFL** function app.

Recent resources	
Name	Type
 PrioritZFL	Function App

31. Select **Configuration**.



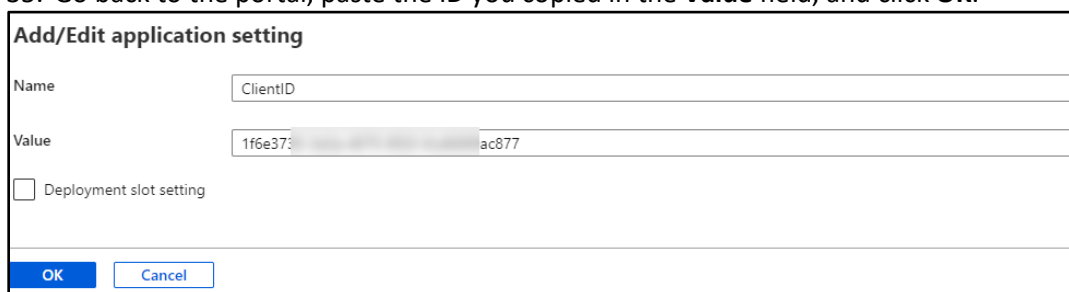
32. Click **+ New application setting**.


 A screenshot of the 'Application settings' page. At the top, it says 'Application settings are encrypted at rest and transmitted over an encrypted channel. You can'. Below this, there is a button '+ New application setting' which is highlighted with a red rectangular box. To its right are links for 'Show values' and 'Advanced edit'. Below the button is a search bar labeled 'Filter application settings'. At the bottom, there are two columns: 'Name' and 'Value'.

33. Enter **ClientID** for Name.

34. Go to your notepad and copy the **PrioritZFL API application ID**.

35. Go back to the portal, paste the ID you copied in the **Value** field, and click **OK**.


 A screenshot of the 'Add/Edit application setting' dialog box. It has two input fields: 'Name' with the value 'ClientID' and 'Value' with a long alphanumeric string '1f6e373...ac877'. Below these fields is a checkbox labeled 'Deployment slot setting' which is unchecked. At the bottom are two buttons: 'OK' and 'Cancel'.

36. Click **+ New application setting** again.

37. Enter **ClientSecret** for Name.

38. Go to your notepad and copy the **PrioritZFL API Secret**.

39. Go back to the portal, paste the secret you copied in the **Value** field, and click **OK**.

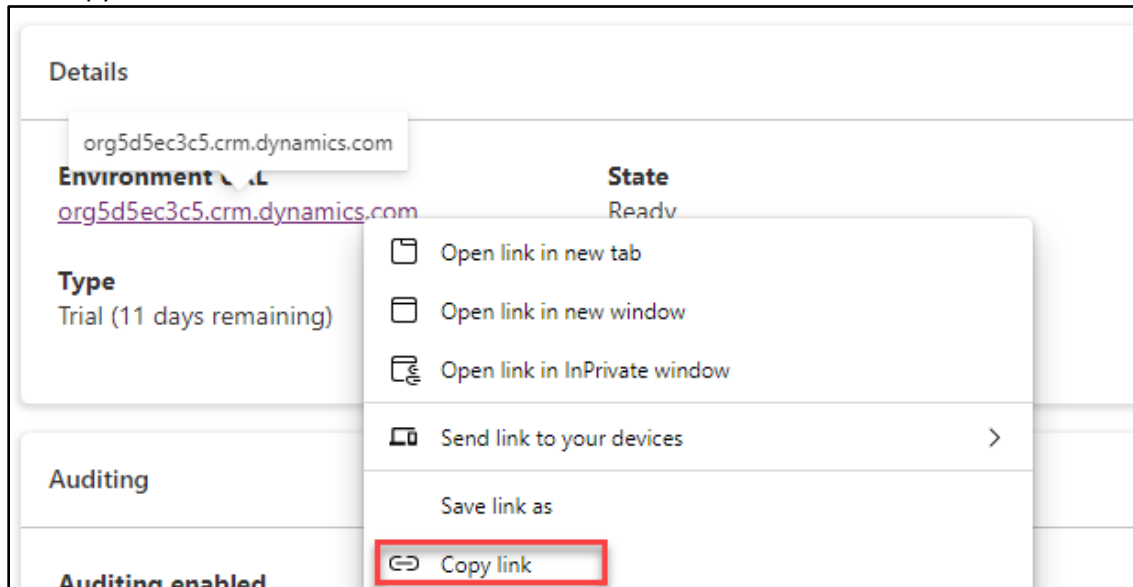
40. Click **+ New application setting** again.

41. Enter **TenantID** for Name.

42. Go to your notepad and copy the **Tenant ID**.

43. Go back to the portal, paste the Tenant ID you copied in the **Value** field, and click **OK**.

44. Click + **New application setting** one more time.
45. Enter **DataverseURL** for Name.
46. Start a new browser window or tab and navigate to [Power Platform admin center](#) and select **Environments**.
47. Click to open the Dev environment you are using for this lab.
48. Copy the **Environment URL**.



49. Go back to the portal, paste the URL you copied in the **Value** field, and click **OK**.
50. You should see the four application settings you added. Click **Save**.

Name	Value	Source
APPINSIGHTS_INSTRUMENTATIONKEY	Hidden value. Click to show value	App Service Config
AzureWebJobsStorage	Hidden value. Click to show value	App Service Config
ClientID	Hidden value. Click to show value	App Service Config
ClientSecret	Hidden value. Click to show value	App Service Config
DataverseURL	Hidden value. Click to show value	App Service Config
FUNCTIONS_EXTENSION_VERSION	Hidden value. Click to show value	App Service Config
FUNCTIONS_WORKER_RUNTIME	Hidden value. Click to show value	App Service Config
MICROSOFT_PROVIDER_AUTHENTICATION_SECRET	Hidden value. Click to show value	App Service Config
TenantID	Hidden value. Click to show value	App Service Config
WEBSITE_CONTENTAZUREFILECONNECTIONSTRING	Hidden value. Click to show value	App Service Config

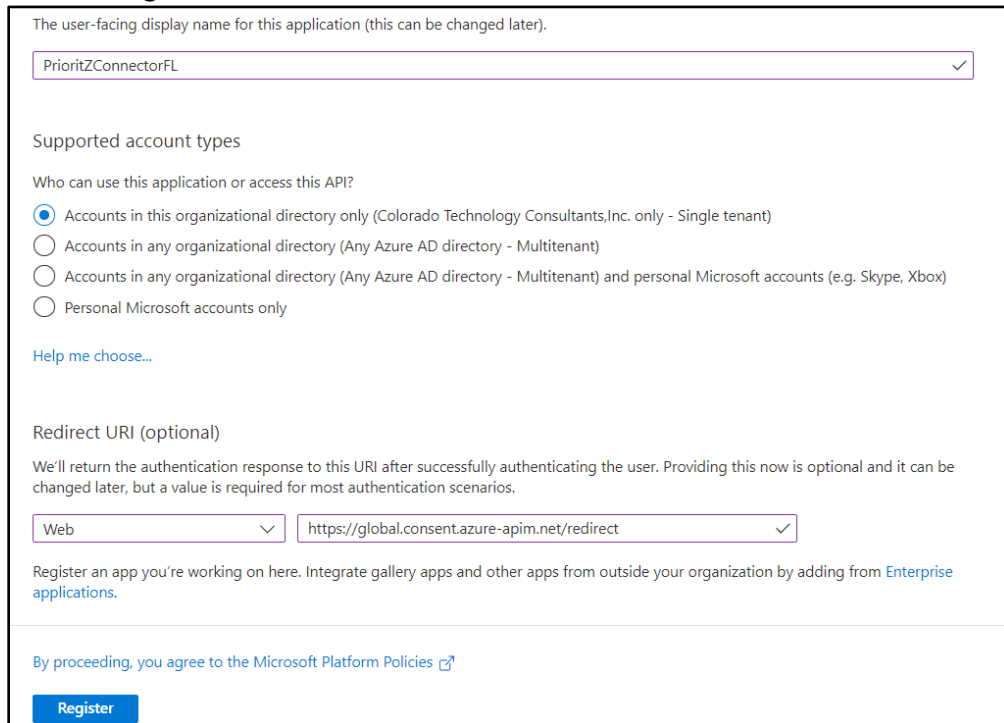
51. Click **Continue**.
52. Navigate to https://login.microsoftonline.com/{tenant id}/adminconsent?client_id={api app id}
Replace {tenant id} and {api app id} with tenant id and PrioritZFL API application ID from your

notepad. If you are not a tenant administrator, you will need to work with your trainer/administrator to consent.

53. Click **Accept**.

Task 2: Register Connector Client app

1. Click on the **Portal menu** and select **Azure Active Directory**.
2. Select **App registrations** and click **+ New registration**. This application registration will be used for the connector to access the protected API.
3. Enter **PrioritZConnector<Initials>** for Name, select **Accounts in this organizational directory only**, select **Web** for Redirect URI, enter <https://global.consent.azure-apim.net/redirect> and click **Register**.



The user-facing display name for this application (this can be changed later).

PrioritZConnectorFL ✓

Supported account types

Who can use this application or access this API?

☒ Accounts in this organizational directory only (Colorado Technology Consultants, Inc. only - Single tenant)

☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant)

☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

☐ Personal Microsoft accounts only

[Help me choose...](#)

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

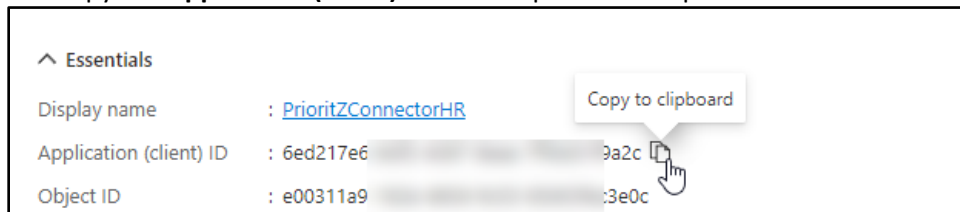
Web ✓ <https://global.consent.azure-apim.net/redirect> ✓

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from [Enterprise applications](#).

By proceeding, you agree to the [Microsoft Platform Policies](#)

Register

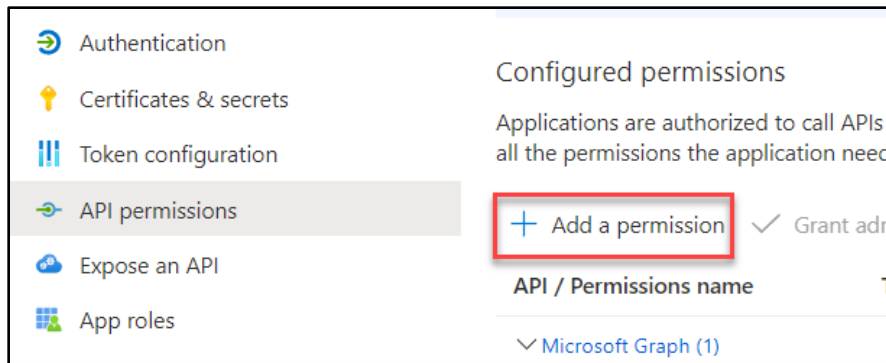
4. Copy the **Application (client) ID** and keep it in a notepad as **PrioritZFL Connector application ID**.



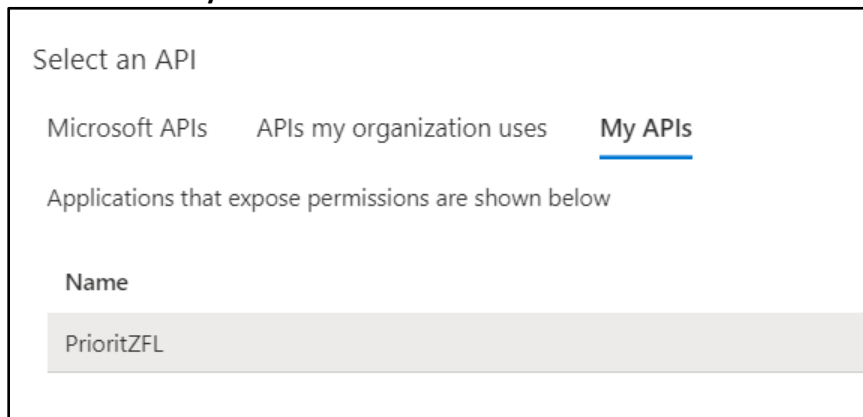
^ Essentials

Display name	:	PrioritZConnectorHR	
Application (client) ID	:	6ed217e6-9a2c	
Object ID	:	e00311a9-3e0c	

5. Select **Certificates & secrets** and click **+ New client secret**.
6. Provide a description, select **3 months**, and click **Add**.
7. Copy the secret **Value** and keep it on a notepad as **PrioritZFL Connector Secret**.
8. Select **API permissions** and click **+ Add a permission**.



9. Select the **My APIs** tab and select **PrioritZFL**.



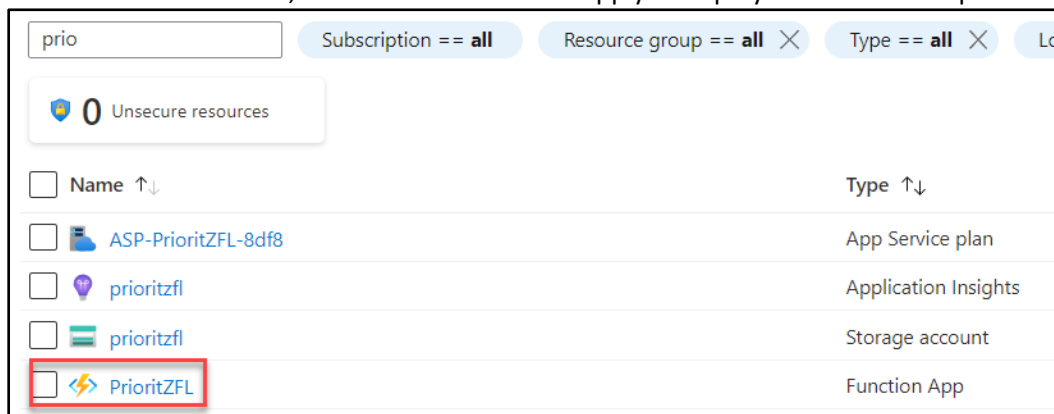
10. Check the **user_impersonation** checkbox and click **Add permission**.

Exercise 3 – Create Connector

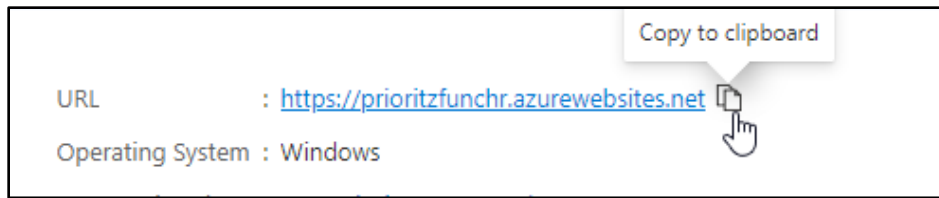
In this exercise, you will create a new custom connector.

Task 1: Create connector

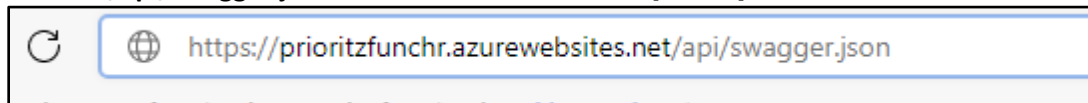
1. Navigate to <https://portal.azure.com/>
2. Select **All resources**, search for the function app you deployed and click to open it.



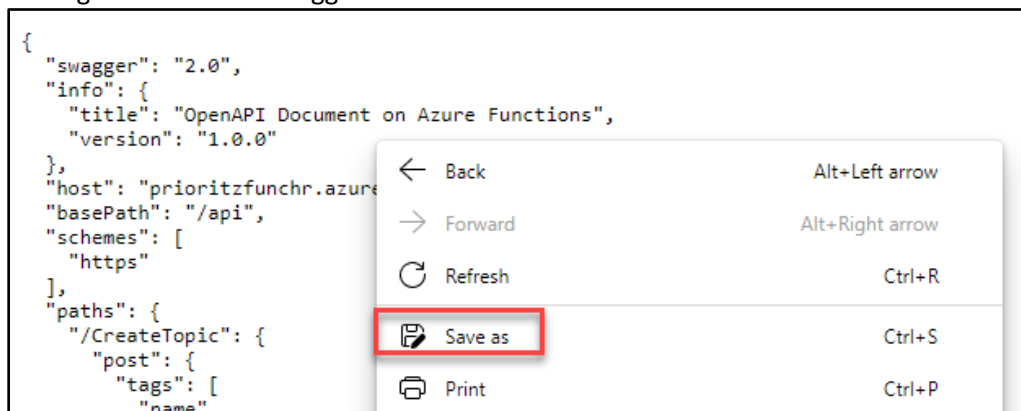
3. Copy the **URL**.



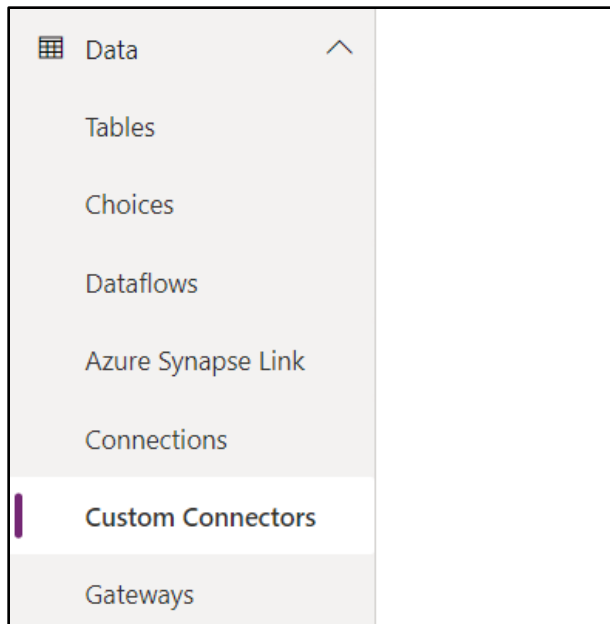
4. Open a new browser tab or window and paste the URL you copied.
5. Add **/api/swagger.json** to the end of the URL and [ENTER]



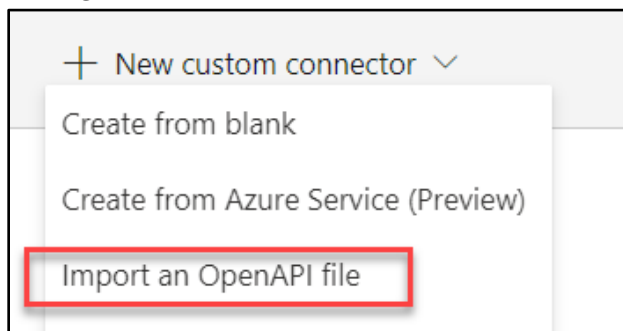
6. Click **Accept** if prompted for permissions.
7. Right click on the swagger and select **Save as**.



8. Save the file on your local machine.
9. Navigate to [Power Apps maker portal](#) and make sure you have the correct Dev environment selected.
10. Expand **Data** and select **Custom Connectors**.



11. Click on the chevron button next to the New custom connector and select **Import an OpenAPI file**.



12. Enter **PrioritZ Connector** for name and click **Import**.

A screenshot of the 'Import an OpenAPI file' dialog box. It has a title bar 'Import an OpenAPI file'. Below it, there's a 'Connector name' field with the text 'PrioritZ Connector'. Below that, there's an 'Import an OpenAPI file' field with the text 'OpenAPI file'. To the right of this field is a purple 'Import' button. At the bottom, there are two buttons: 'Continue' and 'Cancel'.

13. Select the swagger file you saved and click **Open**.

14. Click **Continue**.

15. Provide Description and advance to **Security**.

Description

PrioritZ Azure function

☐ Connect via on-premises data gateway [Learn more](#)

Scheme *

☒ HTTPS ☐ HTTP

Host *

prioritzfunchr.azurewebsites.net

Base URL

/api

Security →

16. Select **OAuth 2.0** for Authentication type.
17. Select **Azure Active Directory** Identity provider.
18. Copy the **PrioritZFL Connector application ID** from your notepad and paste it in the **Client id** field.
19. Copy the **PrioritZFL Connector Secret** your notepad and paste it in the **Client secret** field.
20. Copy the **Tenat ID** from your notepad and replace common with it in the **Tenant ID** field.
21. Copy the **PrioritZ API application ID** from your notepad and paste it in the **Resource URL** field.
22. Enter **true** for Enable on-behalf-of login.
23. Click **Create connector**.

4. Code (Preview) > 5. Test ☐ Swagger Editor ☒ Create connector

OAuth 2.0

Identity Provider

Azure Active Directory

Client id *

0b8 0c

Client secret *

.....

Login URL

https://login.windows.net

Tenant ID

8050 ic7eb1b45e

Resource URL *

295 62014

Enable on-behalf-of login

true

Scope

Scope

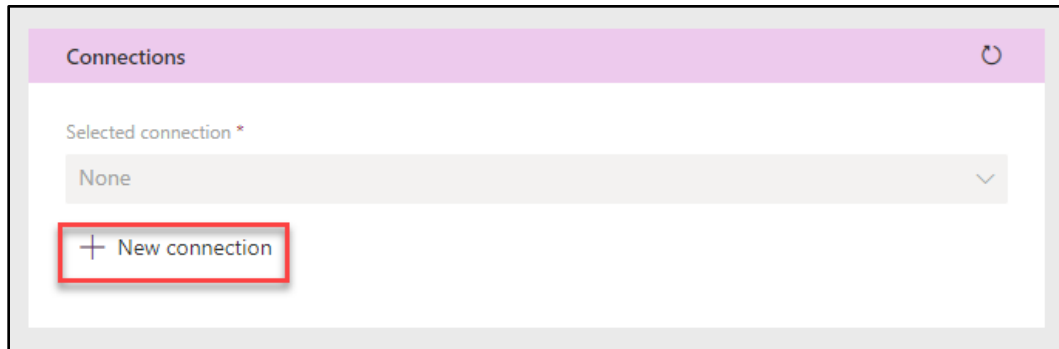
Redirect URL

Save the custom connector to generate the redirect URL

Edit

Task 2: Test connector

1. Select the **Test** tab.
2. Click **+ New connection**.



3. Click **Create**.
4. Provide your credentials.
5. Click **Accept**.
6. Select **Custom connectors** and click **Edit** on the **PrioritZ connector**.



7. Select the **Test** tab.
8. Make sure the connection you created is selected.
9. Turn on **Raw Body**.
10. Provide the JSON below and click **Test operation**.

```
{
  "topic": "Test Topic",
  "details": "From Azure Function",
  "respondBy": "2022-11-01",
  "myNotes": "It worked",
  "choices": [
    {
      "choice": "Choice 1"
    }
  ]
}
```

CreateTopic

Raw Body

☒ On

```

{
  "topic": "Test Topic",
  "details": "From Azure Function",
  "respondBy": "2022-11-01",
  "myNotes": "It worked",
  "choices": [
    {
      "choice": "Choice 1"
    }
  ]
}

```

Test operation

11. The operation test should succeed, and the response should look like the image below.

Request

Response

Status

(200)

Headers

```

{
  "content-encoding": "gzip",
  "content-type": "application/json; charset=utf-8",
  "date": "Thu, 21 Apr 2022 18:23:49 GMT",
  "request-context": "appld=cid-v1:ad12b022-f7df-44f7-8d83-21f7060ed5b0",
  "strict-transport-security": "max-age=31536000; includeSubDomains",

```

Body

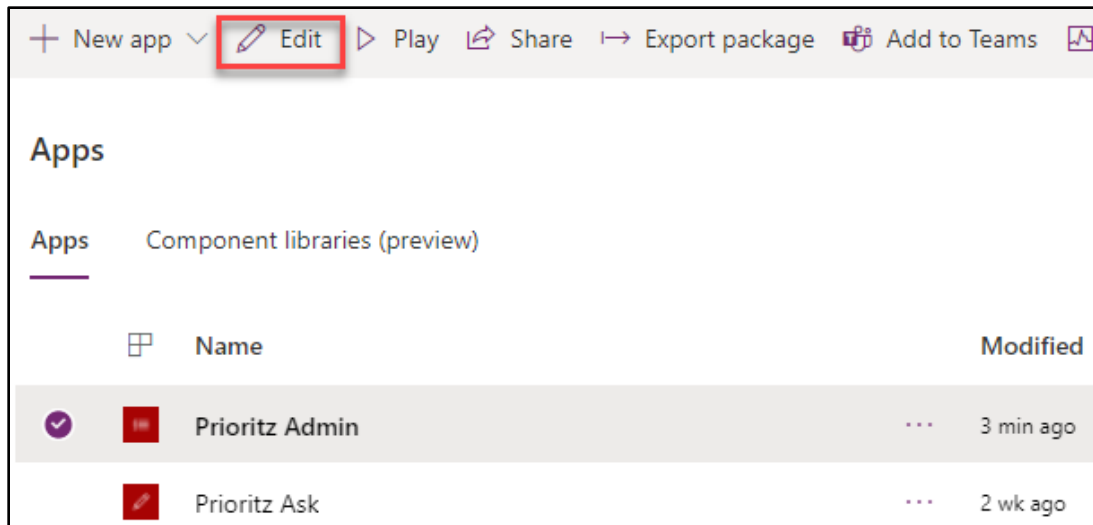
"9093c92c-a0c1-ec11-983e-000d3a591cea"

Exercise 4 – Use Function from Canvas App

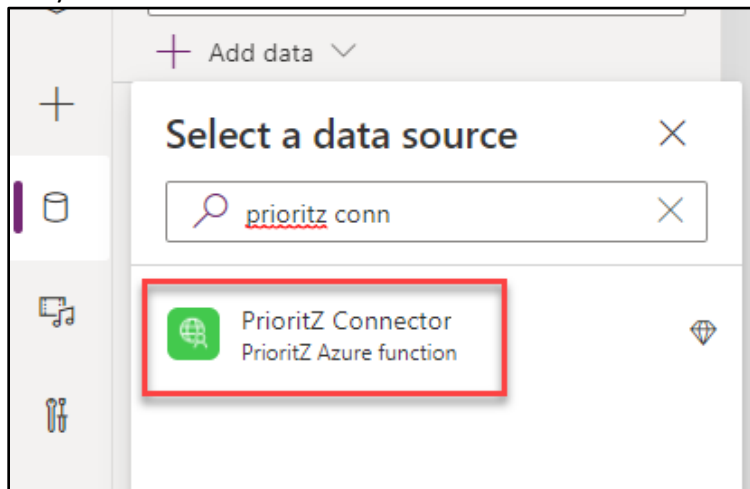
In this exercise, you will use then Azure function you created via the custom connector from the PrioritZ Admin canvas application.

Task 1: Use function

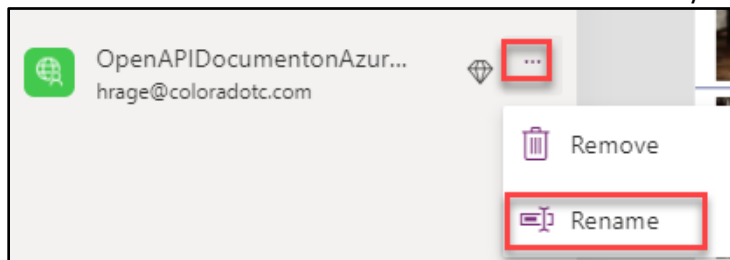
1. Navigate to [Power Apps maker portal](#) and make sure you are in correct environment.
2. Select Apps, select the **PrioritZ Admin** application and click **Edit**.



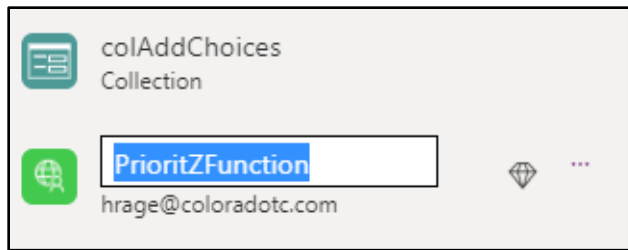
3. Select **Data**, click **+ Add data**, search for prioritz connector, and select the **Prioritz Connector** you created.



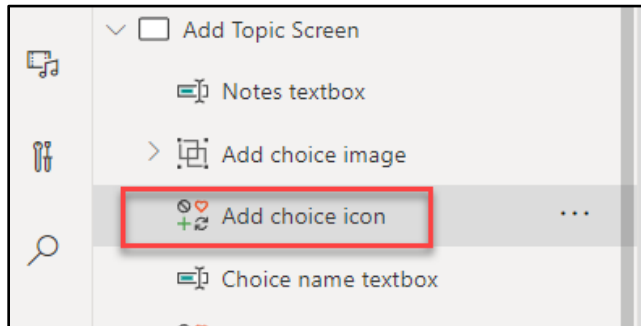
4. Add the connector by clicking again.
5. Click on the **... More actions** button of the connector you just added and select **Rename**.



6. Rename the connector **PrioritZFunction**.



7. Select the **Tree view** and expand the **Add Topic Screen**.
8. Select the **Add choice icon**.



9. Replace the **OnSelect** formula of the **Add choice icon** with the formula below. This adjusts the column names to match the API and encodes the photos.

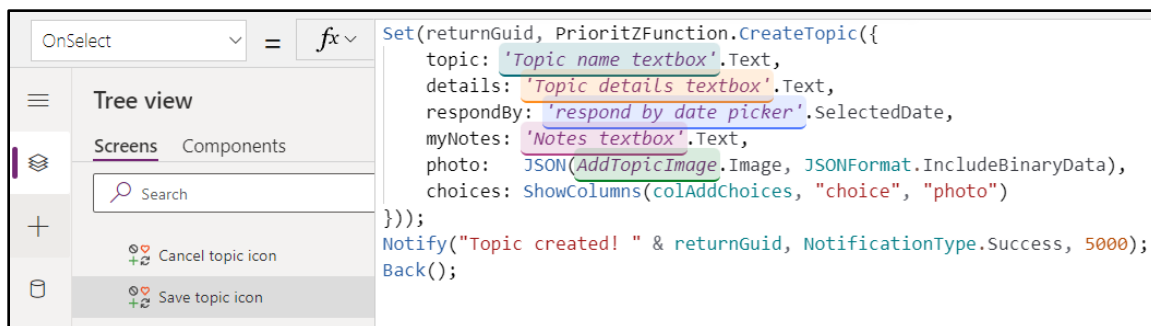
```
Collect(
    colAddChoices,
    {
        choice: 'Choice name textbox'.Text,
        photoRaw: UploadedImage1.Image,
        photo: JSON(
            UploadedImage1.Image,
            JSONFormat.IncludeBinaryData
        )
    }
);
Reset('Choice name textbox');
Reset(AddMediaButton2)
```



10. Select **Save topic icon**.

11. Replace the **OnSelect** formula of the **Save topic icon** with the formula below. This changes to have the API create the “ask”.

```
Set(returnGuid, PrioritZFunction.CreateTopic({
    topic: 'Topic name textbox'.Text,
    details: 'Topic details textbox'.Text,
    respondBy: 'respond by date picker'.SelectedDate,
    myNotes: 'Notes textbox'.Text,
    photo: JSON(AddTopicImage.Image, JSONFormat.IncludeBinaryData),
    choices: ShowColumns(colAddChoices, "choice", "photo")
}));
Notify("Topic created! " & returnGuid, NotificationType.Success, 5000);
Back();
```



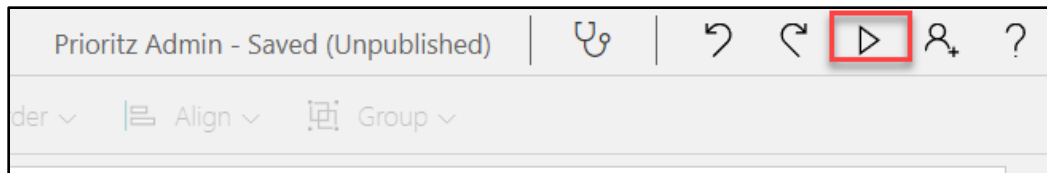
12. Click **File** and select **Save**.

13. Click on the ← back button.

14. Do not navigate away from this page.

Task 2: Test application

1. Select the **Home Screen** and click **Preview the app**.



2. Click on the + add button.
3. Enter **Function Test** for Topic, **Testing the function** for Details. **Note for testing the function** for Note, select a date for Response by, and click **add a picture**.

 A screenshot of the Prioritz Admin form. The header shows the "Contoso Coffee" logo and "Prioritz Admin". The form has several input fields: "Function Test" (Topic), "Testing the function" (Details), "Note for testing the function" (Note), and "Respond By" (Response By) with a date picker set to "4/30/2022". There is a "Choice" dropdown menu on the right. At the bottom, there is a button labeled "Tap or click to add a picture" which is highlighted with a red box.

4. Select any small image from your local machine.
5. Enter **Test choice one** for Choice and click **add a picture**.
6. Select any small image from your machine and click +.

 A screenshot of the Prioritz Admin form. The "Choice" dropdown menu is set to "Test choice one". Below the dropdown, there is a "Change Picture" button. To the right of the dropdown, there is a plus sign icon (+) which is highlighted with a red box.

7. Enter **Test choice two** for Choice and click **add a picture**.
8. Select any image from your machine and click +.
9. Click **Save**.

Contoso Coffee PrioritZ Admin

Function Test

Testing the function

Note for testing the function

Respond By

4/30/2022

Change Picture

Choice

Tap or click to add a picture

Test choice one

Test choice two

10. The new topic should be saved, and you should be navigated back to the main screen.
11. Locate the new topic you created and open it.

Contoso Coffee PrioritZ Admin

Function Test

4/29/2022 6:00 PM

12. You should see the two choices you added to topic.

Contoso Coffee PrioritZ Admin

Function Test

Testing the function

Test choice one

Test choice two