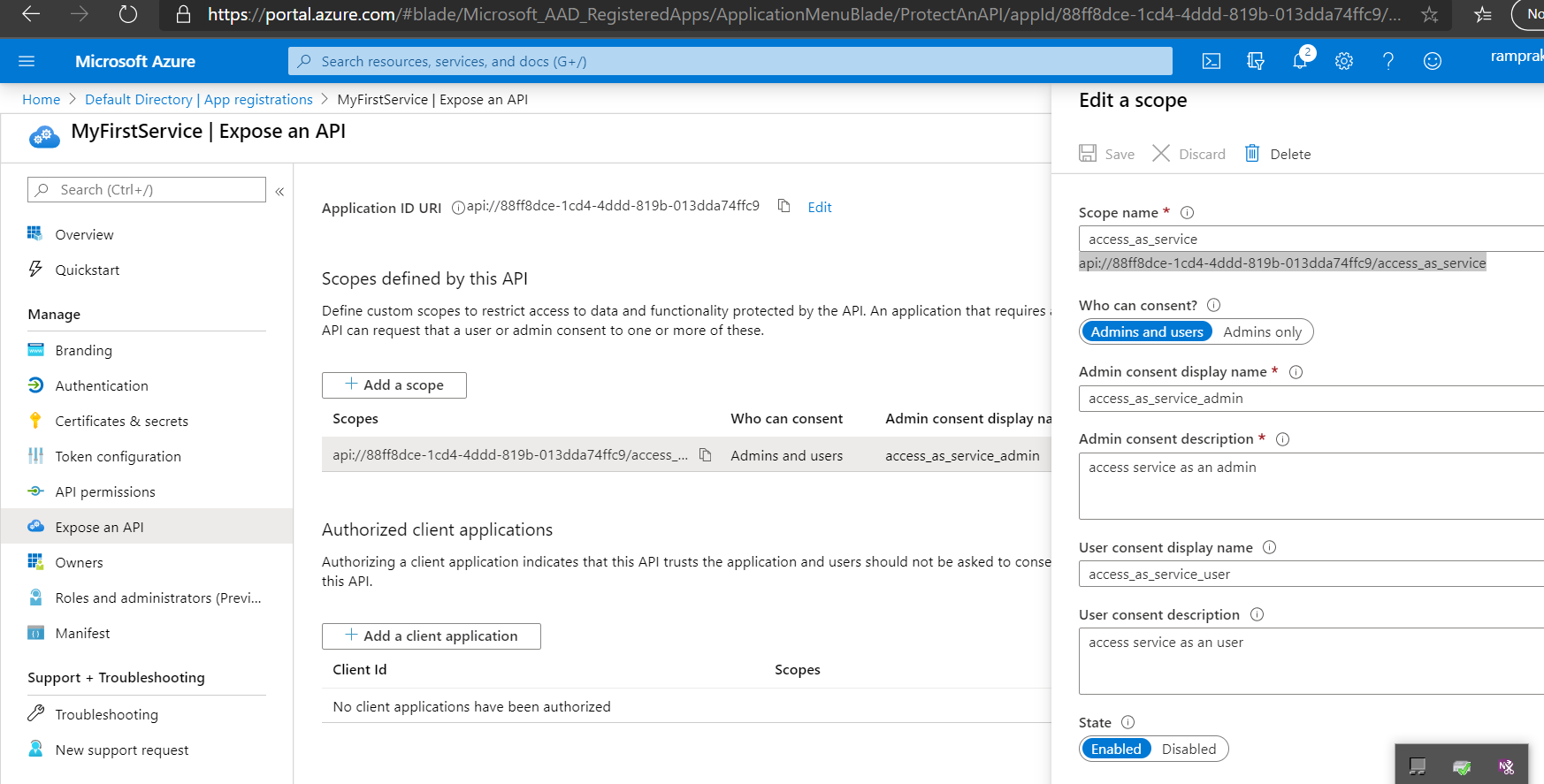
Service - WebApp Client in Azure scenario

TO CONFIGURE THE SERVICE APPLICATION

Step 1: To create a new app on portal.azure.com, SIGN WITH YOUR Microsoft user id, register a new application for your service Web API application (menu – Azure Active Directory -> App registrations -> New Registration

Step 2: Define a scope for the created web API application by using the menu Expose an API. I have created a scope api://88ff8dce-1cd4-4ddd-819b-013dda74ffc9/access\_as\_service as follows



Step 3: Add the following code in Startup.cs, to secure API with Azure token service (add Microsoft.Web.Identity reference in VS. At the time of writing, the project Microsoft.Web.Identity is not available in nuget but available as a separate project in Githhub <https://github.com/Azure-Samples/active-directory-aspnetcore-webapp-openidconnect-v2/tree/master/Microsoft.Identity.Web>)

public void ConfigureServices(IServiceCollection services)

{

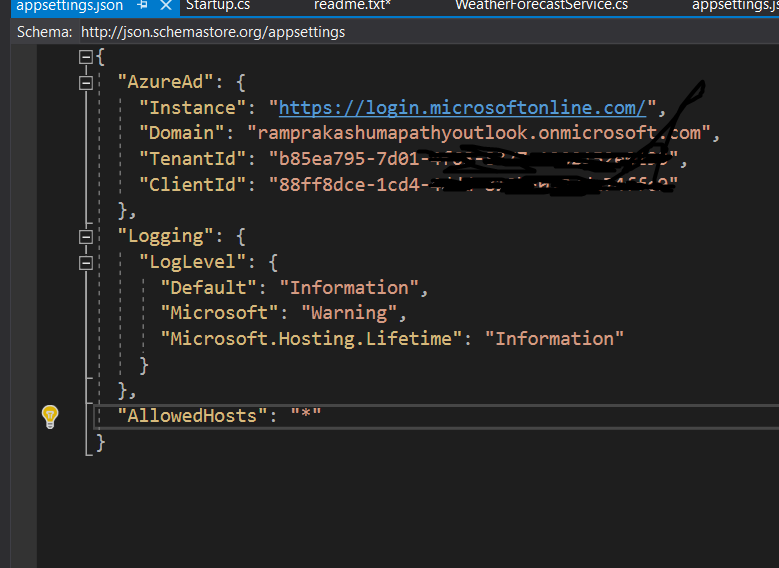
services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddProtectedWebApi("AzureAd", Configuration, options => Configuration.Bind("AzureAD", options));

services.AddControllers();

}

The following are the appsetings keys



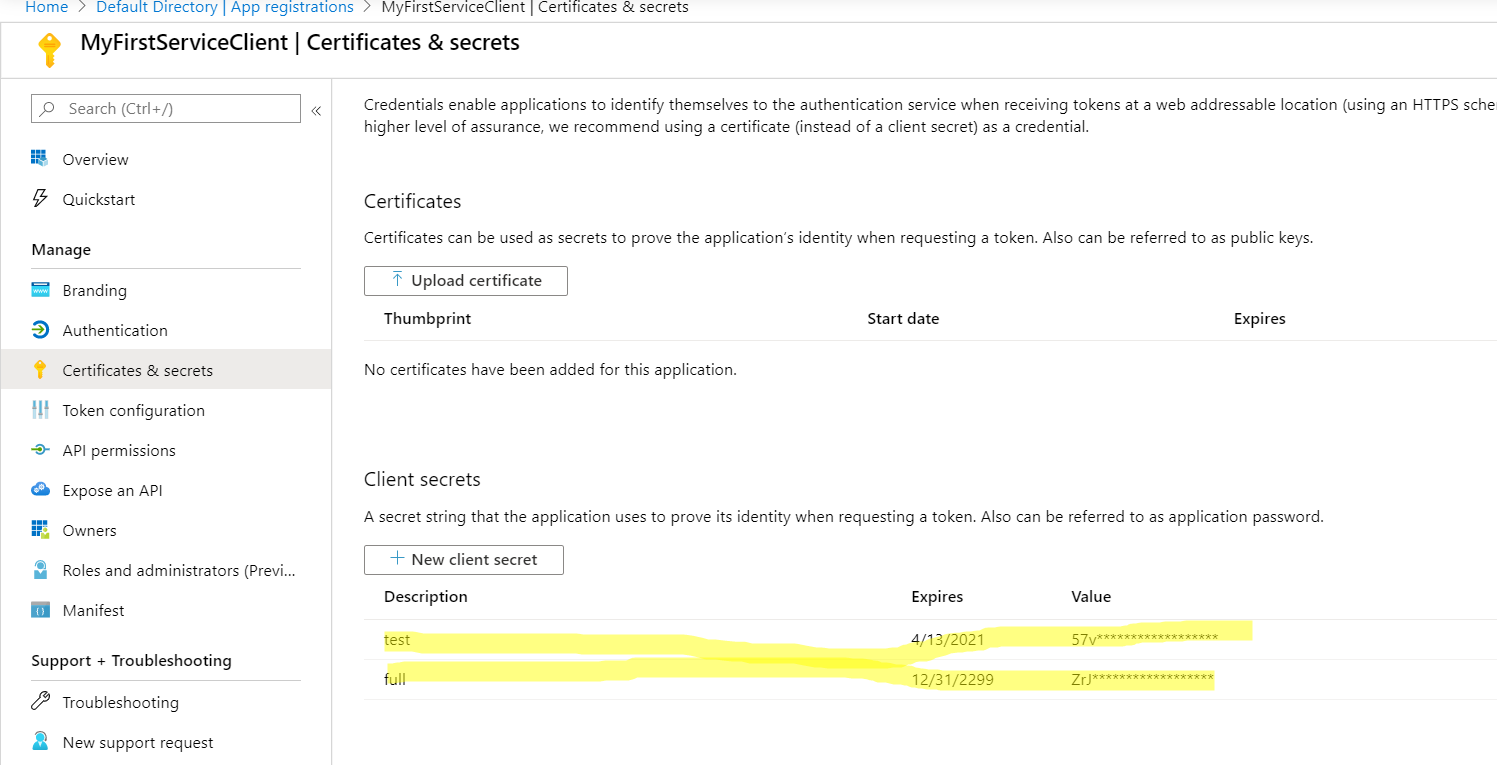
At this point the Web API is ready for use.

TO CONFIGURE THE WEB APPLICATION TO CONSUME WEB SERVICE

Step 1: To create a new app on portal.azure.com, SIGN WITH YOUR Microsoft user id, register a new application for your web application (menu – Azure Active Directory -> App registrations -> New Registration

Step 2: Since this a user interactive web application, your web application needs to be authenticated in azure portal with signin page. When authentication is complete, the azure portal needs to be redirected to your application. So, the redirect url and logout url information needs to register after authentication. (menu – Azure Active Directory -> Your registered application -> Authentication)

Step3: Register your application secret to prove its identity when request a token for the service to call. This is done by New Client Secret button (menu – Azure Active Directory -> Your registered application -> Certificates & secrets). Copy the secret key value once generated because it is available only once.



Step 4: Add the following code in Startup.cs, to secure web app and call the web API with Azure authenticated token. The first highlighted part of the code down below enables the web application authentication with Azure and the second one protects the web api and configures the web application to call the web api with azure generated token. Remember to present scope (2nd parameter in AddWebAppCallProtectedWebApi) we have created in step 2 of the Web API service.

public void ConfigureServices(IServiceCollection services)

{

services.AddAuthentication(OpenIdConnectDefaults.AuthenticationScheme)

.AddSignIn("AzureAd", Configuration, options => Configuration.Bind("AzureAd", options));

services.AddWebAppCallsProtectedWebApi(Configuration, new string[] { Configuration["WebAPIs:MyFirstService.Scope"] })

.AddInMemoryTokenCaches();

services.AddDistributedSqlServerCache(options =>

{

options.ConnectionString = Configuration.GetConnectionString("Ecommerce");

options.SchemaName = "common";

options.TableName = "TokenCache";

});

services.AddControllersWithViews(options =>

{

var policy = new AuthorizationPolicyBuilder()

.RequireAuthenticatedUser()

.Build();

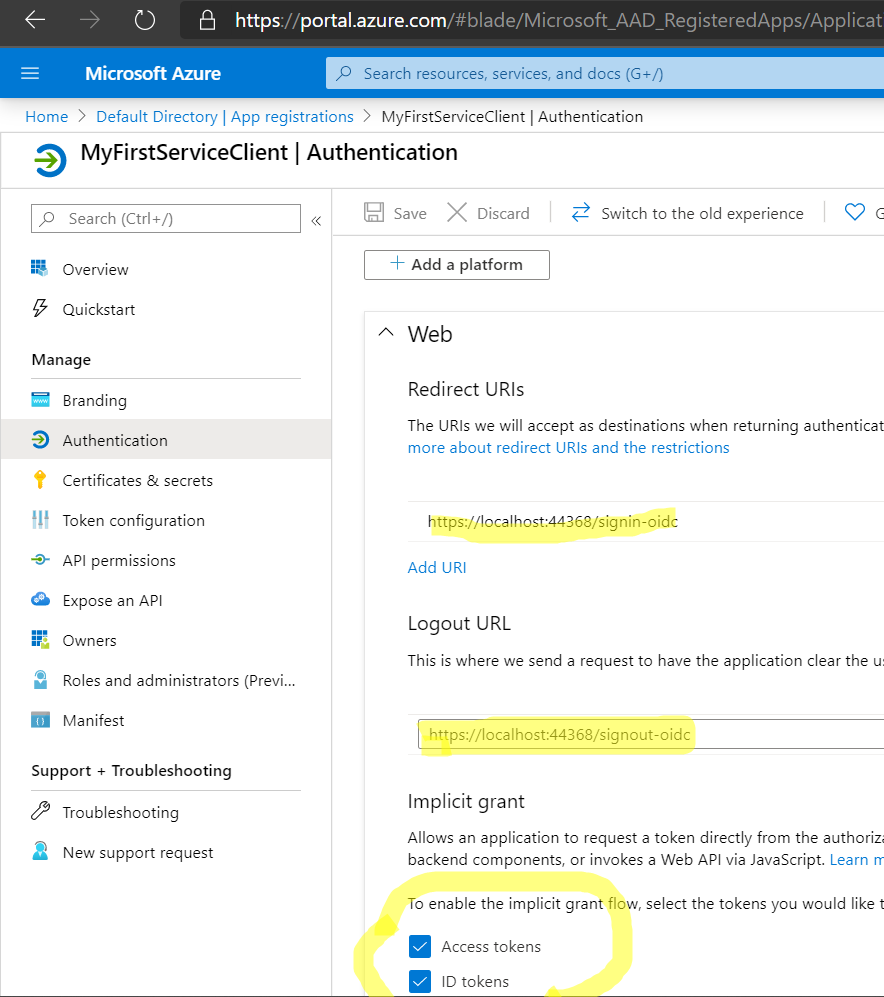
options.Filters.Add(new AuthorizeFilter(policy));

});

services.AddRazorPages();

services.AddHttpClient<IWeatherForcastService, WeatherForecastService>();

}



The following are appsettings keys

{

"AzureAd": {

"Instance": "https://login.microsoftonline.com/",

"Domain": "ramprakashumapathyoutlook.onmicrosoft.com",

"TenantId": "b85ea795-7d01-4f68-8377-1302152e0193",

"ClientId": "ecd87ad1-14e3-4558-b054-def7bd06d622",

"CallbackPath": "/signin-oidc",

"ClientSecret": "ZrJfk9Zr7Oy=3c@NJk8-\_7Rsv7ce@2gZ"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"Ecommerce": "Data Source=(LocalDb)\\MSSQLLocalDB;Database=Ecommerce;Trusted\_Connection=True;"

},

"WebAPIs": {

"MyFirstService.Scope": "api://88ff8dce-1cd4-4ddd-819b-013dda74ffc9/access\_as\_service",

"MyFirstService.BaseAddress": "https://localhost:44310"

}

}

Step 5: When call the web api, the following code prepares and appends the bearer token on http header.

public async Task<IEnumerable<WeatherForecast>> GetAsync()

{

await PrepareAuthenticatedClient();

var response = await \_httpClient.GetAsync($"{ \_TodoListBaseAddress}/WeatherForecast");

if (response.StatusCode == HttpStatusCode.OK)

{

var content = await response.Content.ReadAsStringAsync();

IEnumerable<WeatherForecast> results = JsonSerializer.Deserialize<IEnumerable<WeatherForecast>>(content);

return results;

}

throw new HttpRequestException($"Invalid status code in the HttpResponseMessage: {response.StatusCode}.");

}

private async Task PrepareAuthenticatedClient()

{

var accessToken = await \_tokenAcquisition.GetAccessTokenForUserAsync(new[] { \_TodoListScope });

\_httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", accessToken);

\_httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

}