Way 1

++++++++++++++++

class Program

{

    static void Main(string[] args)

    {

        string tenantId = "72f988bf-xxxx-xxxx-xxxx-2d7cd011db47";  // find how to get in in attached doc

        string appId = "d06989c6-xxx-xxxx-xxxx-595be40c08a2"; // find how to get in in attached doc

        string secretKey = "G7####7uAS/oo###+a/eFjmP#######X8wVk4="; // find how to get in in attached doc

        string functionappHttpFunctionURL = "<https://examplededicatedfunctionapp.azurewebsites.net>"; //its basically function app url

        var context = new AuthenticationContext("<https://login.microsoftonline.com/>" + tenantId + "/oauth2/token?response\_type=code+id\_token");

        ClientCredential clientCredential = new ClientCredential(appId, secretKey);

        var tokenResponse = context.AcquireTokenAsync(appId , clientCredential).Result;

        var accessToken = tokenResponse.AccessToken;

        using (var client = new HttpClient())

        {

            client.DefaultRequestHeaders.Add("Authorization", "Bearer " + accessToken); //adding header to request

            client.BaseAddress = new Uri(functionappHttpFunctionURL);

            string urlParameters = "";

            HttpResponseMessage response = client.GetAsync(urlParameters).Result;

        }

    }

}

Way 2

++++++++++++++++

1. Native app is registered in Azure AD – if not follow the steps in <https://docs.microsoft.com/en-us/azure/app-service/app-service-mobile-how-to-configure-active-directory-authentication#optional-configure-a-native-client-application>;
2. From the case screenshots I see that the Authentication is enabled for the Function and that correct;
3. Token request code will something like this:

string tenantid = "*[tenant id]*";

string resourceId = "*[tenant id]*";

string clientId = "*[tenant id]*";

Uri redirectUri =  new Uri("[*https://nativeapp.com*](https://nativeapp.com)");

var authenticationContext = new AuthenticationContext(String.Format("[https://login.microsoftonline.com/{0}/](https://login.microsoftonline.com/%7b0%7d/)", tenantid), TokenCache.DefaultShared);

       var authenticationResult = authenticationContext.AcquireTokenAsync(

             resourceId,

             clientId,

             redirectUri,

             new PlatformParameters(PromptBehavior.Always));

1. The request to the Azure Function should look like this:

HttpClient client = new HttpClient();

      client.BaseAddress = new Uri("[*https://function.azurewebsites.net/api/*](https://function.azurewebsites.net/api/)");

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

       client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authenticationResult.AccessToken);

       HttpResponseMessage response = await client.PostAsync("function1?code=gS2943…D6h4SLfg==", httpContent);

The overall steps for the configuration that we’re following are in the next link:

<https://docs.microsoft.com/en-us/azure/app-service/app-service-mobile-how-to-configure-active-directory-authentication>

Way 3

++++++++++++++++

1. In the App config file,



1. In the program.cs file,

        static void Main(string[] args)

        {

            string url = ConfigurationManager.AppSettings["ApiURL"];

            TokenResponse accessToken = new TokenResponse();

            PostAsync(url, accessToken)

            .Wait();

        }

        //This method will perform POST operation

        public static async Task PostAsync(string url, TokenResponse accessToken)

        {

            accessToken = await GetADAccessToken();

            using (var httpClient = new HttpClient())

            {

                httpClient.BaseAddress = new Uri("<https://management.azure.com/>");

                httpClient.DefaultRequestHeaders.Clear();

                httpClient.DefaultRequestHeaders.Authorization =

                   new System.Net.Http.Headers.AuthenticationHeaderValue

                   (accessToken.Token\_Type, accessToken.Access\_Token);

                var response = await httpClient.PostAsync(url, null);

            }

        }

        //This method will help to fetch the AccessToken to authenticate API header

        public static async Task<TokenResponse> GetADAccessToken(bool forApi = true)

        {

            string tenantID = ConfigurationManager.AppSettings["TenantId"];

            string authContextURL = "<https://login.microsoftonline.com/>" + tenantID;

            string clientId = ConfigurationManager.AppSettings["ApplicationID"];

            string clientSecret = ConfigurationManager.AppSettings["ApplicationKey"];

            var authenticationContext = new AuthenticationContext(authContextURL);

            var credential = new ClientCredential(clientId: clientId, clientSecret: clientSecret);

            var result = await authenticationContext.AcquireTokenAsync(resource: "<https://management.azure.com/>", clientCredential: credential);

            if (result == null)

            {

                throw new InvalidOperationException("Failed to obtain the Azure Active Directive token");

            }

            var response = new TokenResponse() { Access\_Token = result.AccessToken, Token\_Type = result.AccessTokenType };

            return response;

        }

Way 4

++++++++++++++++

[FunctionName("PipeFunc")]

public static async Task<HttpResponseMessage> Run([HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)]HttpRequestMessage req, TraceWriter log)

{

log.Info("C# HTTP trigger function processed a request.");

// parse query parameter

string name = req.GetQueryNameValuePairs()

.FirstOrDefault(q => string.Compare(q.Key, "name", true) == 0)

.Value;

// Get request body

dynamic data = await req.Content.ReadAsAsync<object>();

// Set name to query string or body data

name = name ?? data?.name;

// Set name to query string or body data

/\* refer : https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-create-service-principal-portal#create-an-azure-active-directory-application

\* https://www.netiq.com/communities/cool-solutions/creating-application-client-id-client-secret-microsoft-azure-new-portal/

\* http://blogs.ultima.com/azure-data-factory

\* https://docs.microsoft.com/en-us/azure/data-factory/quickstart-create-data-factory-dot-net

\*

\*

\*

\*

\*

\* /\*/

log.Info($"C# Timer trigger function executed at: {DateTime.Now}");

var activeDirectoryEndpoint = "https://login.windows.net/";

var resourceManagerEndpoint = "https://management.azure.com/";

var windowsManagementUri = "https://management.core.windows.net/";

var subscriptionId = "622XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1fa1";

var activeDirectoryTenantId = "72f98622XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1fa1b47";

var clientId = "8504622XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1fa1df4"; //AAD app's application id

var clientSecret = "rnMiYQTjlH217JpK/NoEIzKolKFUTm0xKAKXxlKoxh0="; // AAD aaps application key

var resourceGroupName = "AzurePraXXXXXXXXXXXXXXXXXXXXXXXXXXXXX-RouceGroup";

var dataFactoryName = "AzDXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";

var pipelineName = "ANXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";

var authenticationContext = new AuthenticationContext(activeDirectoryEndpoint + activeDirectoryTenantId);

var credential = new ClientCredential(clientId: clientId, clientSecret: clientSecret);

var result = authenticationContext.AcquireTokenAsync(resource: windowsManagementUri, clientCredential: credential).Result;

if (result == null) throw new InvalidOperationException("Invalid Operation");

var token = result.AccessToken;

var aadTokenCredentials = new TokenCloudCredentials(subscriptionId, token);

var resourceManagerUri = new Uri(resourceManagerEndpoint);

var client = new DataFactoryManagementClient(aadTokenCredentials, resourceManagerUri);

try

{

var slice = DateTime.Now.AddDays(-1);

var pl = new Pipeline();

pl.Name = pipelineName;

pl.Properties = new PipelineProperties();

pl.Properties.Start = DateTime.Parse($"{slice.Date:yyyy-MM-dd}T00:00:00Z");

pl.Properties.End = DateTime.Parse($"{slice.Date:yyyy-MM-dd}T23:59:59Z");

pl.Properties.IsPaused = false;

pl.Properties.Activities = new List<Activity>();

client.Pipelines.CreateOrUpdate(resourceGroupName, dataFactoryName, new PipelineCreateOrUpdateParameters()

{

Pipeline = pl

});

var df=client.DataFactories.List(resourceGroupName);

var pl2 = client.Pipelines.Get(resourceGroupName, dataFactoryName, pipelineName);

}

catch (Exception e)

{

log.Info(e.Message);

}

// Set name to query string or body data

log.Info("C# HTTP trigger function end a request 1.");

return name == null

? req.CreateResponse(HttpStatusCode.BadRequest, "Please pass a name on the query string or in the request body")

: req.CreateResponse(HttpStatusCode.OK, "Hello " + name);

log.Info("C# HTTP trigger function end a request 2.");

}

Way 5

++++++++++++++++

public static void Run(TimerInfo myTimer, TraceWriter log)

{

log.Info($"C# Timer trigger function executed at: {DateTime.Now}");

var activeDirectoryEndpoint = "https://login.windows.net/";

var resourceManagerEndpoint = "https://management.azure.com/";

var windowsManagementUri = "https://management.core.windows.net/";

var subscriptionId = "91020b2b-XXXXXXXXXXX-1dfd2ecd7c3e";

var activeDirectoryTenantId = "37be05f1- XXXXXXXXXXX -f5fc7e3956c8";

var clientId = "13f3179b- XXXXXXXXXXX ";

var clientSecret = "hDbEnD1Pot XXXXXXXXXXX 6QOi XXXXXXXXXXX l76a2Y6ngSKglBc=";

var resourceGroupName = "PV- XXXXXXXXXXX -GRP";

var dataFactoryName = "PCT XXXXXXXXXXX aFactory";

var pipelineName = "ANP\_ XXXXXXXXXXX \_ABLB\_TO\_SQLDB";

var authenticationContext = new AuthenticationContext(activeDirectoryEndpoint + activeDirectoryTenantId);

var credential = new ClientCredential(clientId: clientId, clientSecret: clientSecret);

var result = authenticationContext.AcquireTokenAsync(resource: windowsManagementUri, clientCredential: credential).Result;

if (result == null) throw new InvalidOperationException("Invalid Operation");

var token = result.AccessToken;

var aadTokenCredentials = new TokenCloudCredentials(subscriptionId, token);

var resourceManagerUri = new Uri(resourceManagerEndpoint);

var client = new DataFactoryManagementClient(aadTokenCredentials, resourceManagerUri);

try

{

var slice = DateTime.Now.AddDays(-1);

var pl = client.Pipelines.Get(resourceGroupName, dataFactoryName, pipelineName);

pl.Pipeline.Properties.Start = DateTime.Parse($"{slice.Date:yyyy-MM-dd}T00:00:00Z");

pl.Pipeline.Properties.End = DateTime.Parse($"{slice.Date:yyyy-MM-dd}T23:59:59Z");

pl.Pipeline.Properties.IsPaused = false;

client.Pipelines.CreateOrUpdate(resourceGroupName, dataFactoryName, new PipelineCreateOrUpdateParameters()

{

Pipeline = pl.Pipeline

});

}

catch (Exception e)

{

log.Info(e.Message);

}

}