Contents

[**Create an application in Azure Portal** 1](#_Toc64303177)

[**New Registration** 1](#_Toc64303178)

[**Generate Secret Key** 3](#_Toc64303179)

[**API Permissions** 4](#_Toc64303180)

[**Add Application in Dynamics 365 FO** 5](#_Toc64303181)

[**Creating visual studio project for consuming OData service** 6](#_Toc64303182)

[**Creating Authentication Utility** 9](#_Toc64303183)

[**Creating the console application** 15](#_Toc64303184)

# **Create an application in Azure Portal**

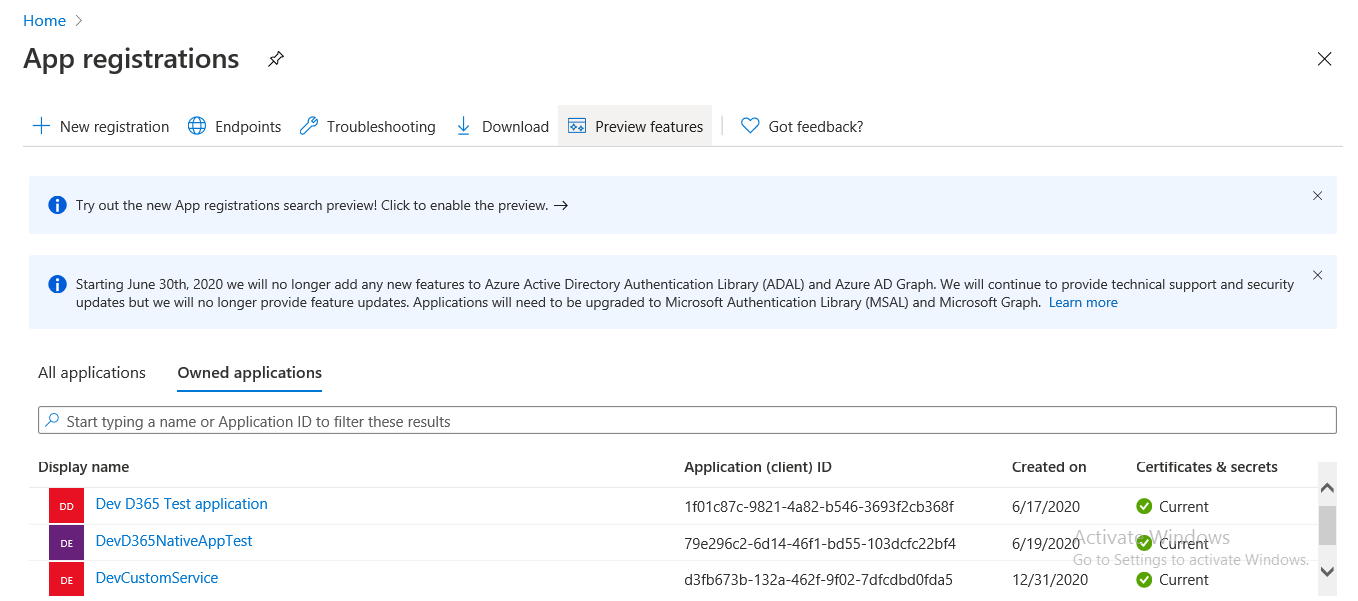
Ref Links

<http://d365technext.blogspot.com/2018/06/azure-app-registration.html>

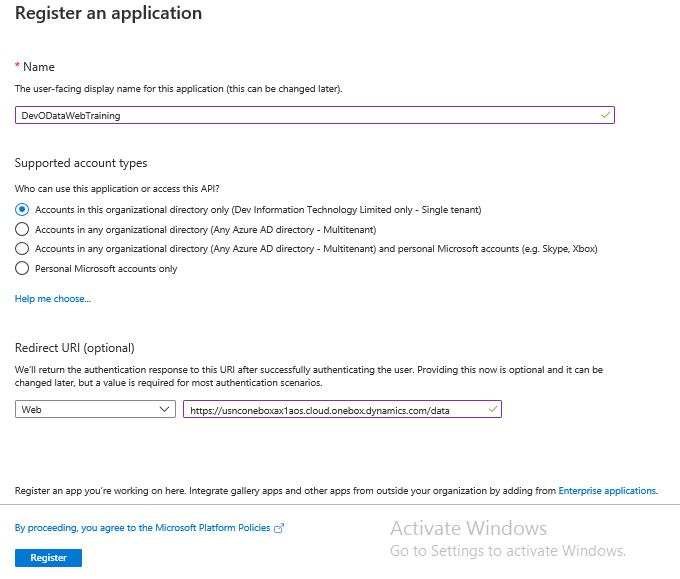
<https://www.d365ug.com/blogs/nandita-nityanandam/2018/11/20/third-party-integration-in-d365fo-using-odata>

## **New Registration**

Login on <https://portal.azure.com/> using your credentials. Select Azure Active Directory >> App Registration.



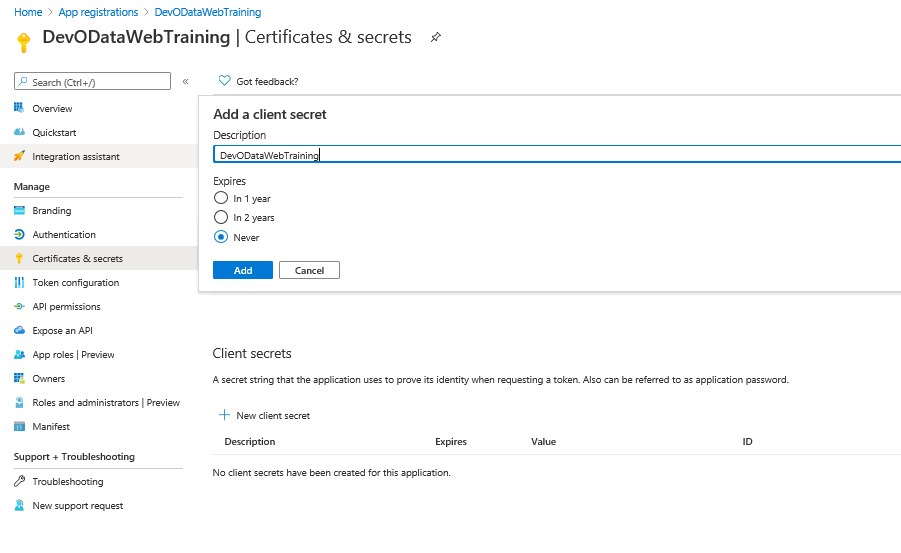
Click on **“New Registration”.** Enter details as shown in the screen below and register the application.



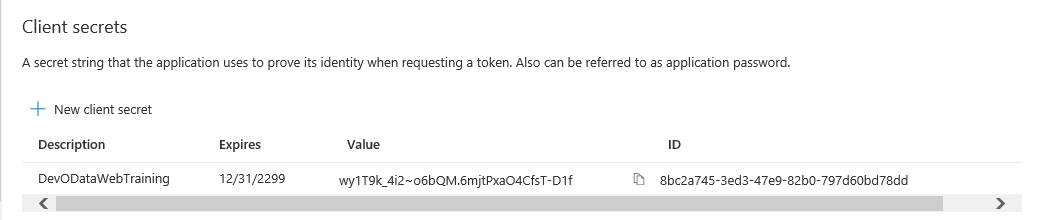
## **Generate Secret Key**

To create secret key, select certificates & secrets. Click on New client secret.

Enter details as shown in the screen below and Add the client secret.



Copy the client secret value

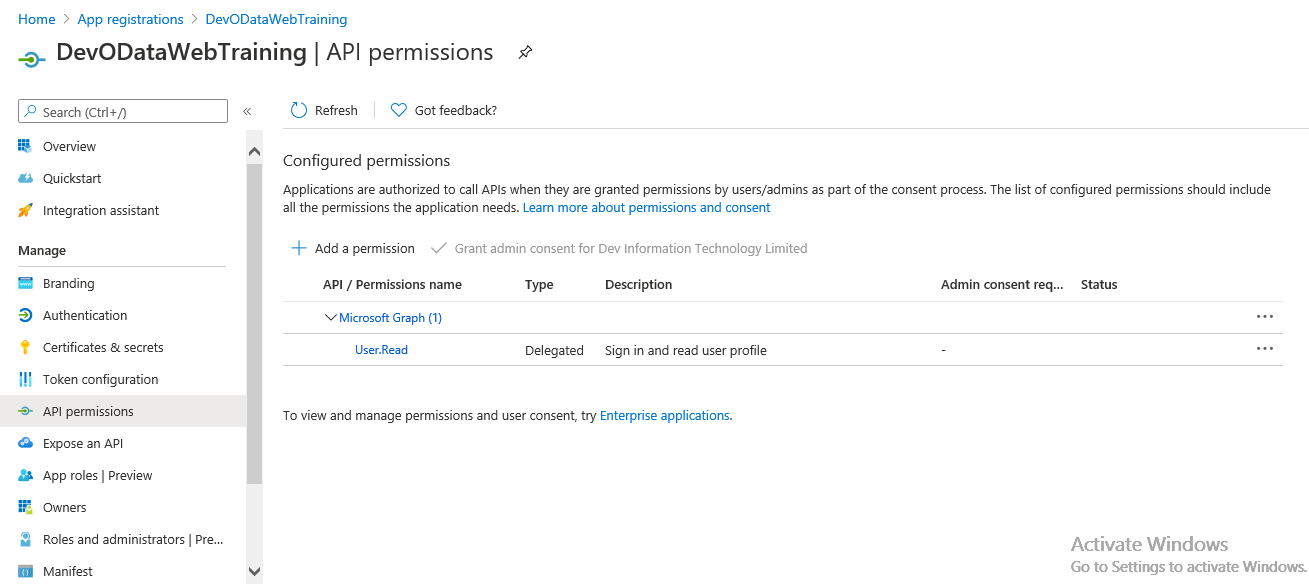


Secret key value: wy1T9k\_4i2~o6bQM.6mjtPxaO4CfsT-D1f

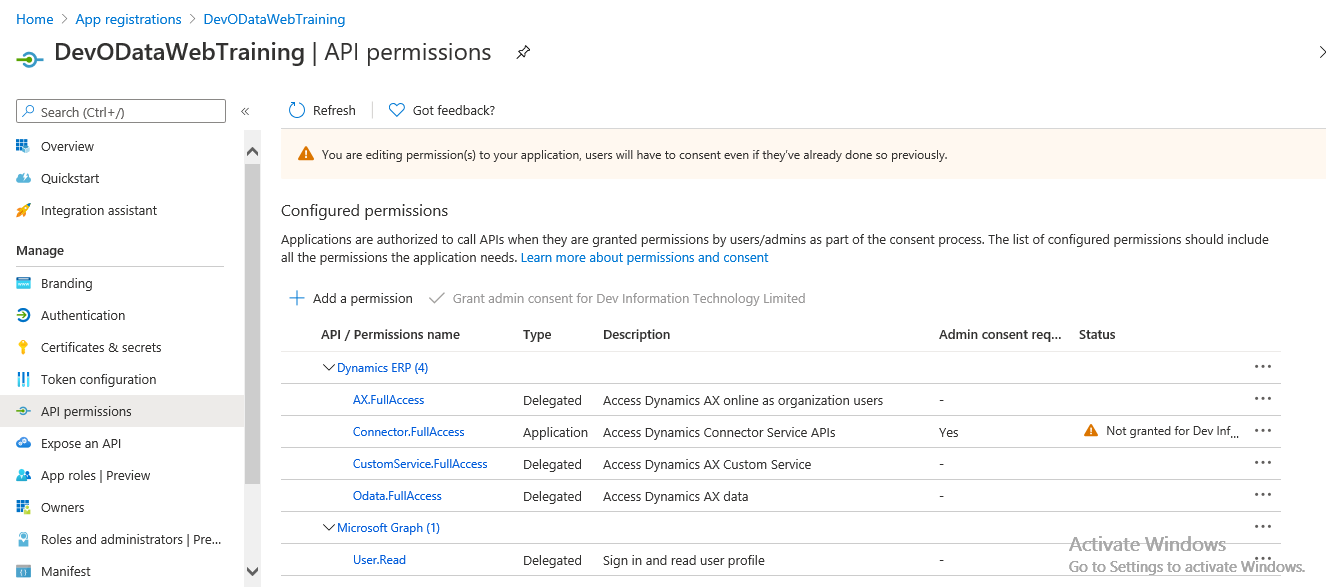
This value will not be visible again so do not miss to copy this.

## **API Permissions**

To add permissions, click on API Permissions >> Add a permission



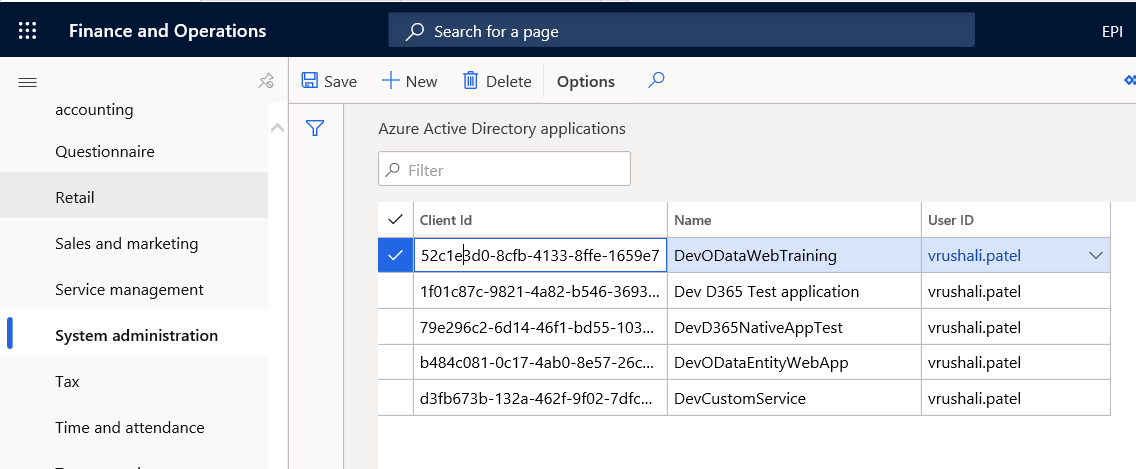
Add a permission >> Microsoft APIs >> Dynamics ERP. Add all delegated and application permissions, click on Add permission. Permission will be added as shown in the screen below.



# **Add Application in Dynamics 365 FO**

System Administration >> Azure Active Directory applications >> New

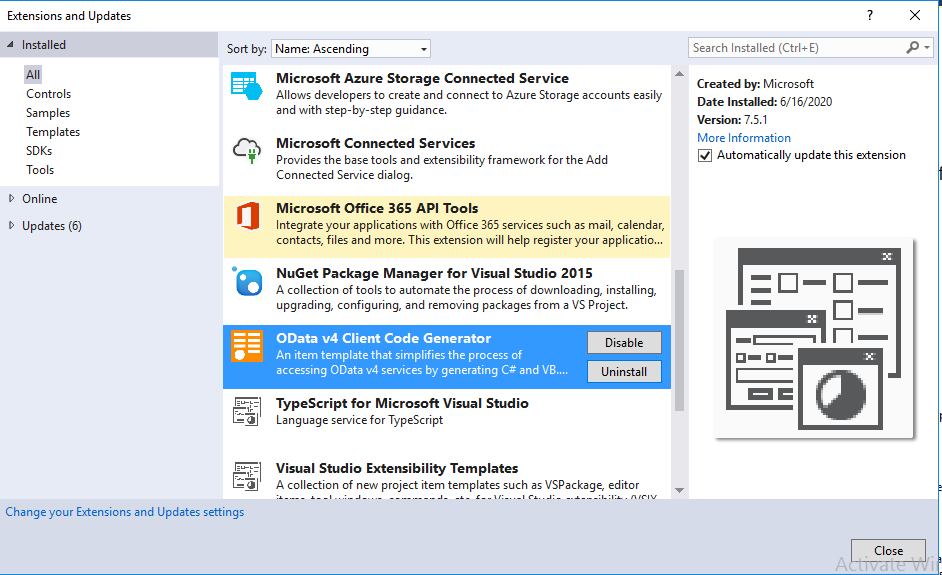
Enter Azure application client id in Application Id, application name and user id.



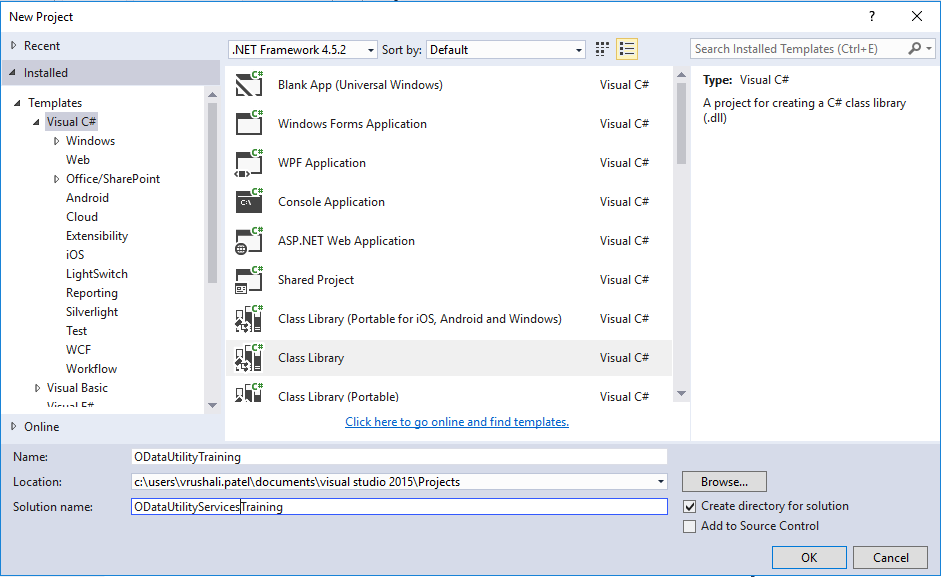
# **Creating visual studio project for consuming OData service**

**Creating Odata Utility**

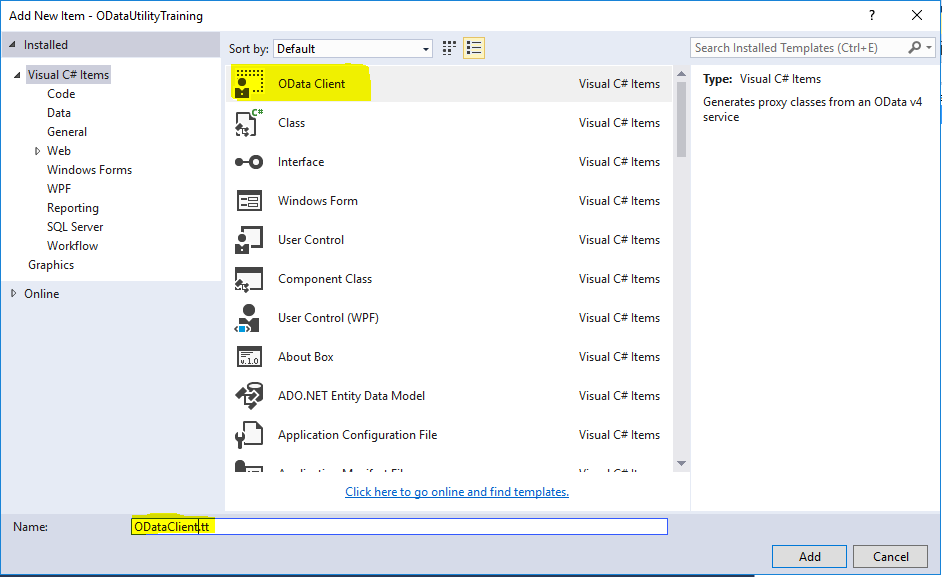
1. Open Visual Studio. Go to Tools >> Extension and Updates and find “OData v4 Client Code Generator”. If this is not installed then install it from the same window.



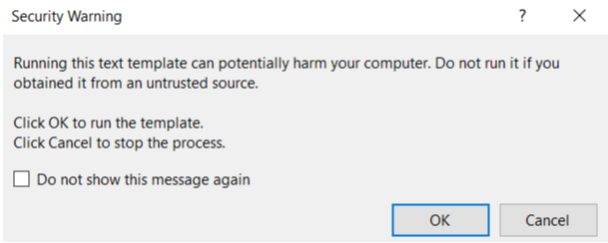
1. Create a new C# project of type Class Library with the name “ODataUtilityTraining”. Add this project to a new solution with the name “ODataServicesTraining”.



1. Delete the default class (class1) that is created.
2. Right click on project and click “New item”. In the Add New Item dialog, search for “OData Client”, which will be the class type. This will generate Proxy classes from the OData v4 service. The name of the class will be ODataClient.tt



1. When you get the following message box, click on Cancel.



1. Open ODataClient.tt file and find the code **public const string MetadataDocumentUri = “”;** and replace “” by your <D365FO URL>/data/$metadata like as below:

**public const string MetadataDocumentUri = "https://usnconeboxax1aos.cloud.onebox.dynamics.com/data/$metadata";**

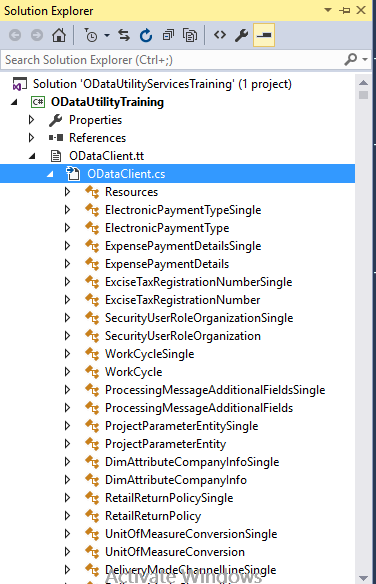
1. Next, go to the code **public const string TempFilePath = “”;** and add here any temppath as below.

**public const string TempFilePath = @"C:\DumpPathTraining\Edmx.xml";**

This path will be used to navigate to the “Edmx” xml file generated while processing the tt file.

Save the file and you will get the same message box (as step #5) again. Click on OK.

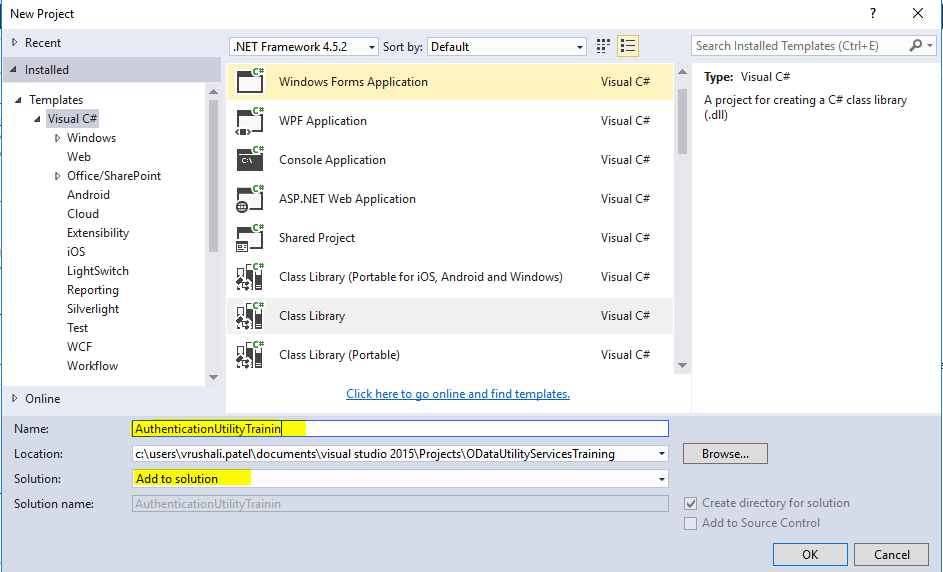
After the process is completed, verify if the .cs file generated is proper. Expand the .tt file and then the .cs file. You will see all the classes generated from the .tt file.



1. Right click on the project and click on Build.

# **Creating Authentication Utility**

1. Create a new C# project of type Class Library with the name “AuthenticationUtilityTrainin”. Add this project to the existing “OdataUtilityServicesTraining” solution.

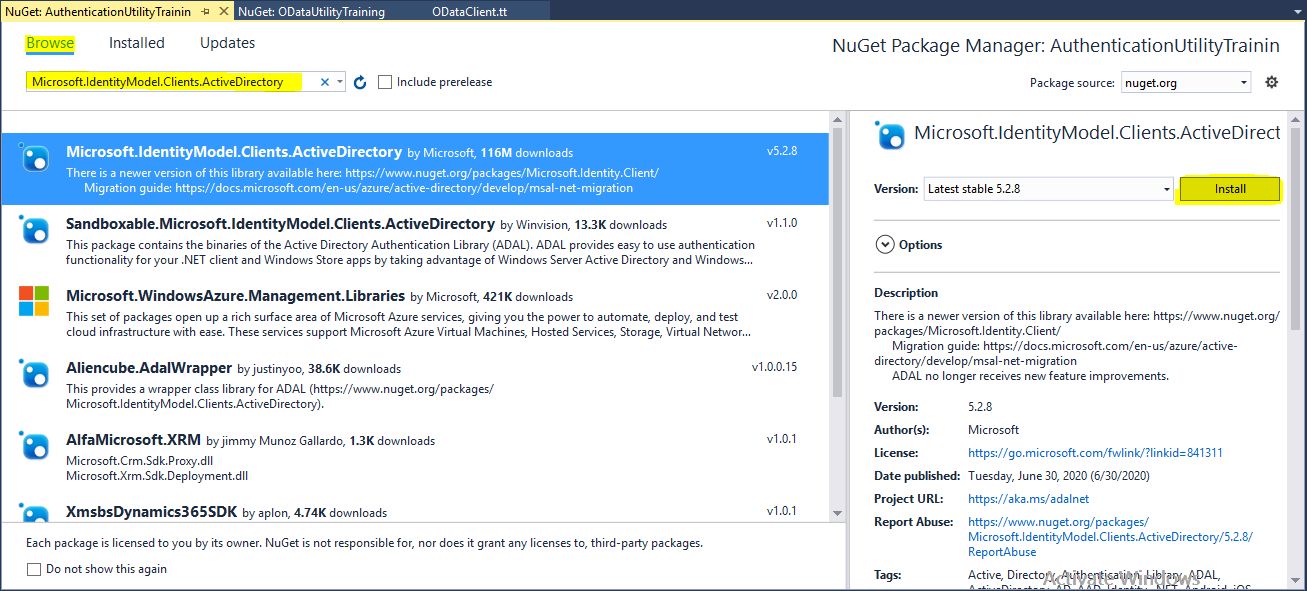


1. Install the following NuGet Packages:

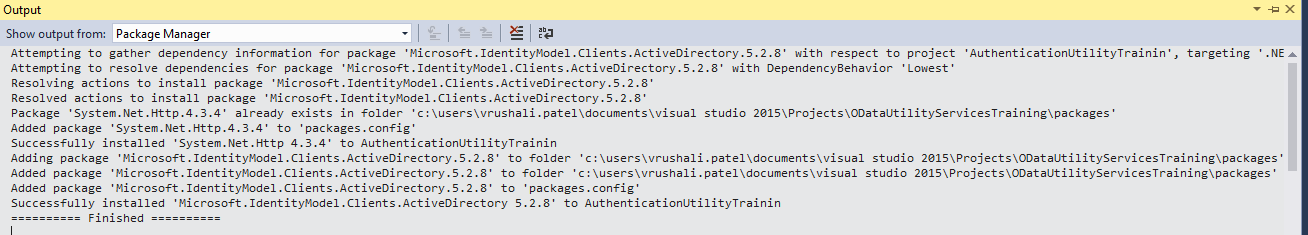
* Microsoft.IdentityModel.Clients.ActiveDirectory
* Newtonsoft.Json
* Microsoft.Odata.Client(this will automatically install a few more dependent packages)

Right click on the Project “AuthenticationUtilityTrainin” and select the option “Manage Nuget Packages”. So that below window will open.

In the “Installed” tab, verify that package is installed or not. If it is not installed then go to “Browse” tab and write the package name in “Search” window as shown in below screen.

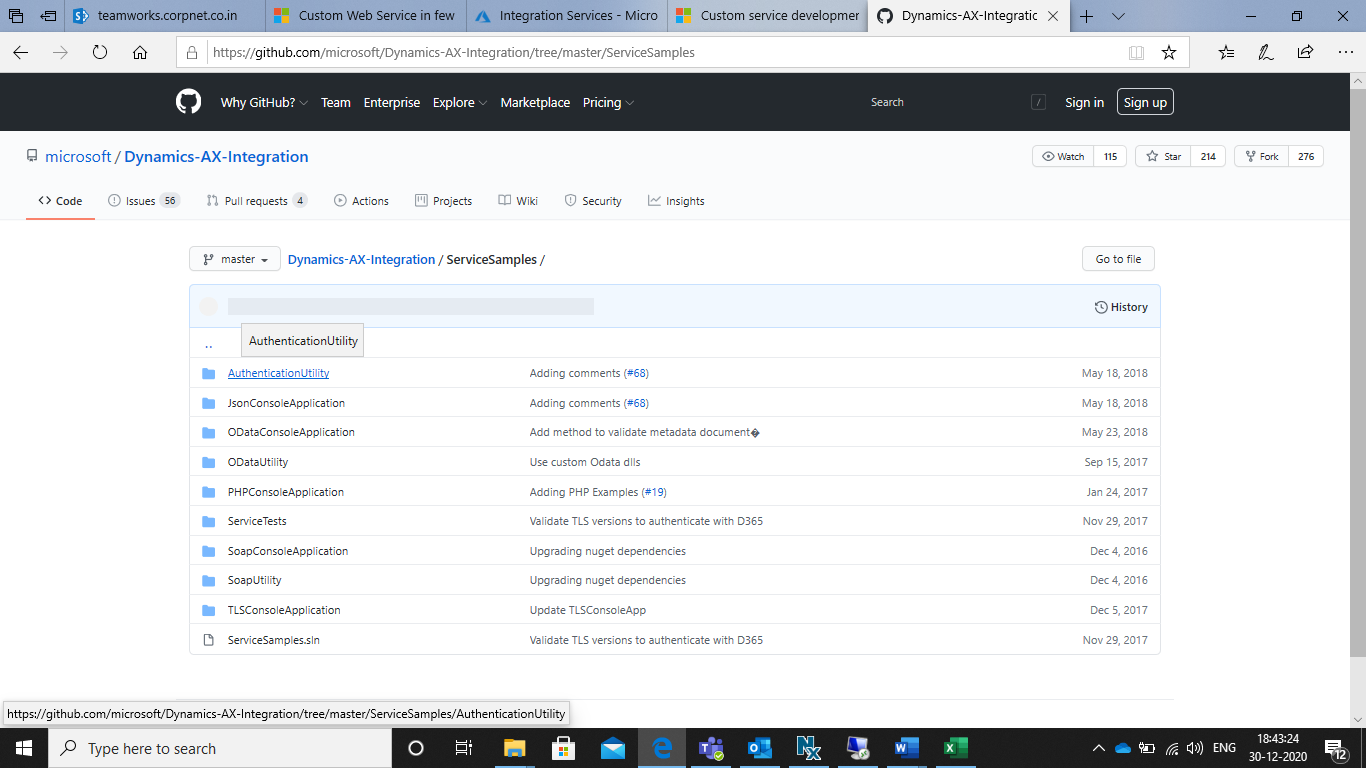


Click on “Install” button to install the package. After the package has been installed successfully, below screen will open.



1. Follow the same steps to install other required packages also.
2. Delete the default class (class1) that is created.
3. Right click on the project and click on New Item. In the Add New Item dialog, select Code → Class and create a new class with the name ClientConfiguration.cs.

You can copy the code in this classe from **Microsoft Dynamics AX Integration GitHub repository**.



Below is the code of clientConfiguration, I have updated for my project.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AuthenticationUtilityTrainin

{

public partial class ClientConfiguration

{

public static ClientConfiguration Default { get { return ClientConfiguration.OneBox; } }

public static ClientConfiguration OneBox = new ClientConfiguration()

{

// You only need to populate this section if you are logging on via a native app. For Service to Service scenarios in which you e.g. use a service principal you don't need that.

UriString = "https://usnconeboxax1aos.cloud.onebox.dynamics.com/",

UserName = "vrushali.patel@corpnet.co.in",

// Insert the correct password here for the actual test.

Password = "",

// You need this only if you logon via service principal using a client secret. See: https://docs.microsoft.com/en-us/dynamics365/unified-operations/dev-itpro/data-entities/services-home-page to get more data on how to populate those fields.

// You can find that under AAD in the azure portal

ActiveDirectoryResource = "https://usnconeboxax1aos.cloud.onebox.dynamics.com", // Don't have a trailing "/". Note: Some of the sample code handles that issue.

// You can find the TenantID from the application registered on Azure Portal

ActiveDirectoryTenant = "https://login.microsoftonline.com/7eb749a1-a2ec-4260-bb87-c77c7a0a0b7a", // Some samples: https://login.windows.net/yourtenant.onmicrosoft.com, https://login.windows.net/microsoft.com

ActiveDirectoryClientAppId = "52c1e3d0-8cfb-4133-8ffe-1659e7a5cf76",

// Insert here the application secret when authenticate with AAD by the application

ActiveDirectoryClientAppSecret = "wy1T9k\_4i2~o6bQM.6mjtPxaO4CfsT-D1f",

// Change TLS version of HTTP request from the client here

// Ex: TLSVersion = "1.2"

// Leave it empty if want to use the default version

TLSVersion = "",

};

public string TLSVersion { get; set; }

public string UriString { get; set; }

public string UserName { get; set; }

public string Password { get; set; }

public string ActiveDirectoryResource { get; set; }

public String ActiveDirectoryTenant { get; set; }

public String ActiveDirectoryClientAppId { get; set; }

public string ActiveDirectoryClientAppSecret { get; set; }

}

}

1. Right click on the project and click on New Item. In the Add New Item dialog, select Code → Class and create a new class with the name OAuthHelper.cs.

Write below code:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.IdentityModel.Clients.ActiveDirectory;

namespace AuthenticationUtilityTrainin

{

public class OAuthHelper

{

/// <summary>

/// The header to use for OAuth authentication.

/// </summary>

public const string OAuthHeader = "Authorization";

/// <summary>

/// Retrieves an authentication header from the service.

/// </summary>

/// <returns>The authentication header for the Web API call.</returns>

public static string GetAuthenticationHeader(bool useWebAppAuthentication = false)

{

string aadTenant = ClientConfiguration.Default.ActiveDirectoryTenant;

string aadClientAppId = ClientConfiguration.Default.ActiveDirectoryClientAppId;

string aadClientAppSecret = ClientConfiguration.Default.ActiveDirectoryClientAppSecret;

string aadResource = ClientConfiguration.Default.ActiveDirectoryResource;

AuthenticationContext authenticationContext = new AuthenticationContext(aadTenant, false);

AuthenticationResult authenticationResult;

if (useWebAppAuthentication)

{

if (string.IsNullOrEmpty(aadClientAppSecret))

{

Console.WriteLine("Please fill AAD application secret in ClientConfiguration if you choose authentication by the application.");

throw new Exception("Failed OAuth by empty application secret.");

}

try

{

// OAuth through application by application id and application secret.

var creadential = new ClientCredential(aadClientAppId, aadClientAppSecret);

authenticationResult = authenticationContext.AcquireTokenAsync(aadResource, creadential).Result;

}

catch (Exception ex)

{

Console.WriteLine(string.Format("Failed to authenticate with AAD by application with exception {0} and the stack trace {1}", ex.ToString(), ex.StackTrace));

throw new Exception("Failed to authenticate with AAD by application.");

}

}

else

{

// OAuth through username and password.

string username = ClientConfiguration.Default.UserName;

string password = ClientConfiguration.Default.Password;

if (string.IsNullOrEmpty(password))

{

Console.WriteLine("Please fill user password in ClientConfiguration if you choose authentication by the credential.");

throw new Exception("Failed OAuth by empty password.");

}

try

{

// Get token object

var userCredential = new UserPasswordCredential(username, password); ;

authenticationResult = authenticationContext.AcquireTokenAsync(aadResource, aadClientAppId, userCredential).Result;

}

catch (Exception ex)

{

Console.WriteLine(string.Format("Failed to authenticate with AAD by the credential with exception {0} and the stack trace {1}", ex.ToString(), ex.StackTrace));

throw new Exception("Failed to authenticate with AAD by the credential.");

}

}

// Create and get JWT token

return authenticationResult.CreateAuthorizationHeader();

}

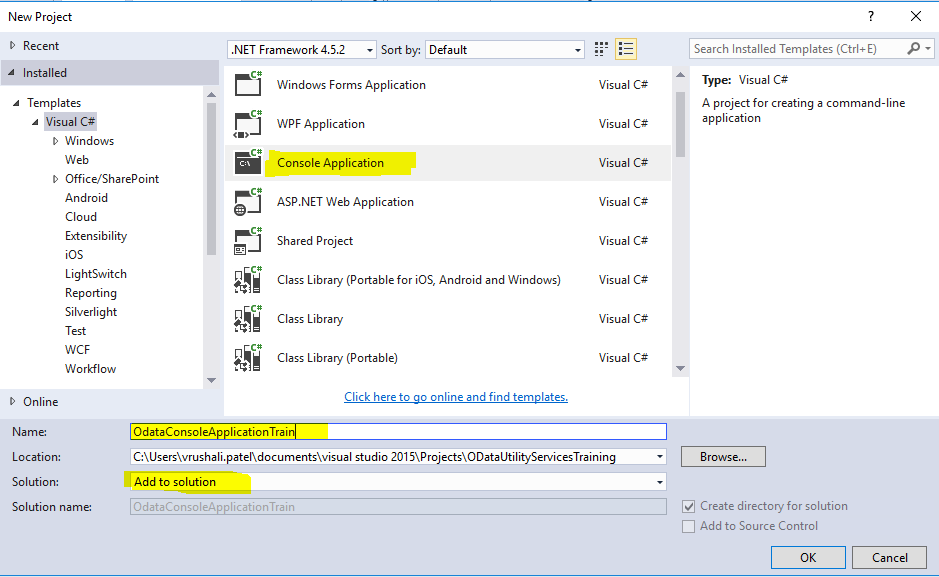
}

}

1. Right click on the project and click on Build.

# **Creating the console application**

1. Create a new C# project of type Console Application with the name “OdataConsoleApplicationTrain”. Add this project to the existing “OdataUtilityServicesTraining” solution.



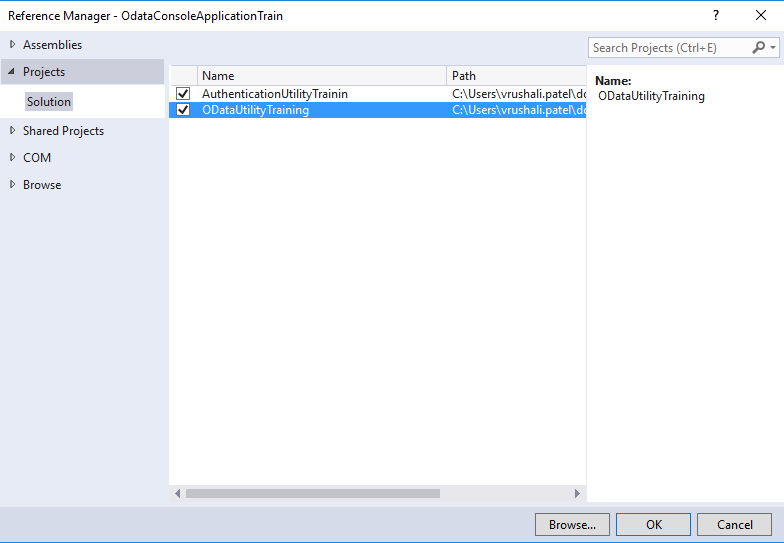
1. Install the following NuGet Packages as shown in above steps:

* Microsoft.IdentityModel.Clients.ActiveDirectory
* Newtonsoft.Json
* Microsoft.OData.Client (this will automatically install few more dependent packages)

1. Install the following reference from the solution:

Right click on the project. Add >> Reference

* ODataUtilityTraining
* AuthenticationUtilityTrainin



1. Open the program.cs class generated in the console application and write below code in this:

using AuthenticationUtilityTrainin;

using Microsoft.OData.Client;

using ODataUtilityTraining.Microsoft.Dynamics.DataEntities;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OdataConsoleApplicationTrain

{

class Program

{

private static string ODataEntityPath = ClientConfiguration.Default.UriString;

static string ODataServiceEndpoint = ODataEntityPath + "data";

private static Uri oDataUri = new Uri(ODataServiceEndpoint, UriKind.Absolute);

private static Resources context = new Resources(oDataUri);

static void Main(string[] args)

{

Console.WriteLine("Set HTTP header...");

context.SendingRequest2 += new EventHandler<SendingRequest2EventArgs>(delegate (object sender, SendingRequest2EventArgs e)

{

var authenticationHeader = OAuthHelper.GetAuthenticationHeader(true);

e.RequestMessage.SetHeader(OAuthHelper.OAuthHeader, authenticationHeader);

});

Console.WriteLine("Creating new customer using Data Entity OData...");

FleetCustomer myCustomer = new FleetCustomer();

DataServiceCollection<FleetCustomer> customersCollection = new DataServiceCollection<FleetCustomer>(context);

customersCollection.Add(myCustomer);

myCustomer.FirstName = "DevOdata Customer Web Test";

myCustomer.LastName = "LastName Web Test";

myCustomer.DriverLicense = "License 1014 Test";

DataServiceResponse response = null;

try

{

response = context.SaveChanges(SaveChangesOptions.PostOnlySetProperties | SaveChangesOptions.BatchWithSingleChangeset);

Console.WriteLine("created ok");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message + ex.InnerException);

}

Console.ReadLine();

GetAllCustomers();

Console.ReadLine();

}

private static void GetAllCustomers()

{

//Using the DataServiceContext class, we can simply perform a LINQ query to retrieve all Customers from the Customer entity.

//This internally will issue web request by using the URI that represents OData protocol

//get all customers

var query = from customer in context.FleetCustomers

select customer;

Console.ForegroundColor = ConsoleColor.Cyan;

//Display the http request Uri that was sent to fetch all customers

Console.WriteLine("Request Uri: " + query.ToString());

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine();

//After retrieving all the customers, iterate through the collection and print each one.

foreach (FleetCustomer c in query)

{

Console.WriteLine("Customer.Driver License: " + c.DriverLicense);

Console.WriteLine("First Name: " + c.FirstName);

Console.WriteLine("Last Name: " + c.LastName);

Console.WriteLine("++++++++");

//Console.ReadLine();

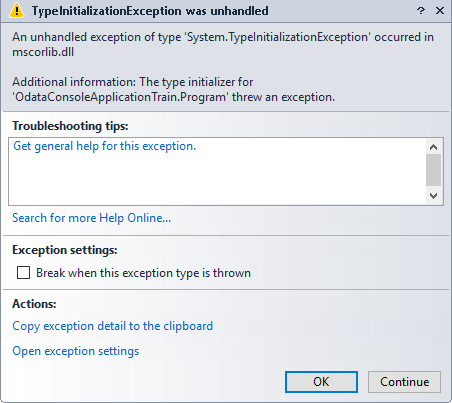
}

}

}

}

1. Right click on the project and click on “Build”.
2. Right click on “ODataConsoleApplicationTrain” and click on “Set as StartUp Project”.
3. Build the project and click on “Execute”.
4. If you face below error, then follow the suggested steps as below:



1. Open the Edmx file from the path which we have defined in the “OdataClient.tt” file in Visual Studio.
2. Replace ‘’ (double quote) by ‘ (Single quote) and then save the file.

Reference Link:

https://community.dynamics.com/365/financeandoperations/f/dynamics-365-for-finance-and-operations-forum/293972/odata-exception-the-type-initializer-for-generatededmmodel-threw-an-exception#:~:text=9%3A41%20AM-,OData%20Exception%20%3A%20The%20type%20initializer,'GeneratedEdmModel'%20threw%20an%20exception.&text=Root%20cause%20of%20the%20issue,%22%20in%20ODataClient1.tt%20file.&text=Change%20the%20file%20path%20in%20auto%20generated%20OdataClient

1. Build the solution. Then after run the project. You will see the success message as below:

