Contents

[**Create custom service** 1](#_Toc60312557)

[**Create Dynamics Model and Project in visual studio** 1](#_Toc60312558)

[**Create Service class** 1](#_Toc60312559)

[**Create Service** 2](#_Toc60312560)

[**Create Service Group** 3](#_Toc60312561)

[**Verify service on D365 FO URL** 4](#_Toc60312562)

[**Register New Application on Azure Portal** 5](#_Toc60312563)

[**New Registration** 5](#_Toc60312564)

[**Generate Secret Key** 6](#_Toc60312565)

[**API Permissions** 6](#_Toc60312566)

[**Add Application in Dynamics AX FO** 7](#_Toc60312567)

[**Consume AX custom service in Third Party Application** 8](#_Toc60312568)

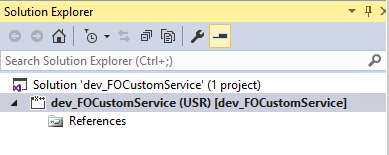
[**AuthenticationUtility Application** 8](#_Toc60312569)

[**Thirdparty Application** 12](#_Toc60312570)

# **Create custom service**

Ref Link: <https://community.dynamics.com/365/financeandoperations/b/365foandaxtechnicalworld/posts/custom-web-service-in-few-steps-d365fo#:~:text=Today%2C%20I%20will%20explain%20how,write%20a%20method%20get%20foo.&text=Step%2D3%20Class%20association%20with%20service.&text=Step%2D4%20Add%20method%20of,those%20you%20want%20to%20expose>

## **Create Dynamics Model and Project in visual studio**



## **Create Service class**

Add class “IntegrationServiceClass” with method below:

**Method returning static Value:**

public str getTest()

{

return "Custom Service has been called Successfully.";

}

**Method returning customer Detail:**

public str getCustomer(str strDataAreaId, str strAccountNum)

{

CustTable custTable;

changecompany(strDataAreaId)

{

select \* from custTable where custTable.AccountNum == strAccountNum;

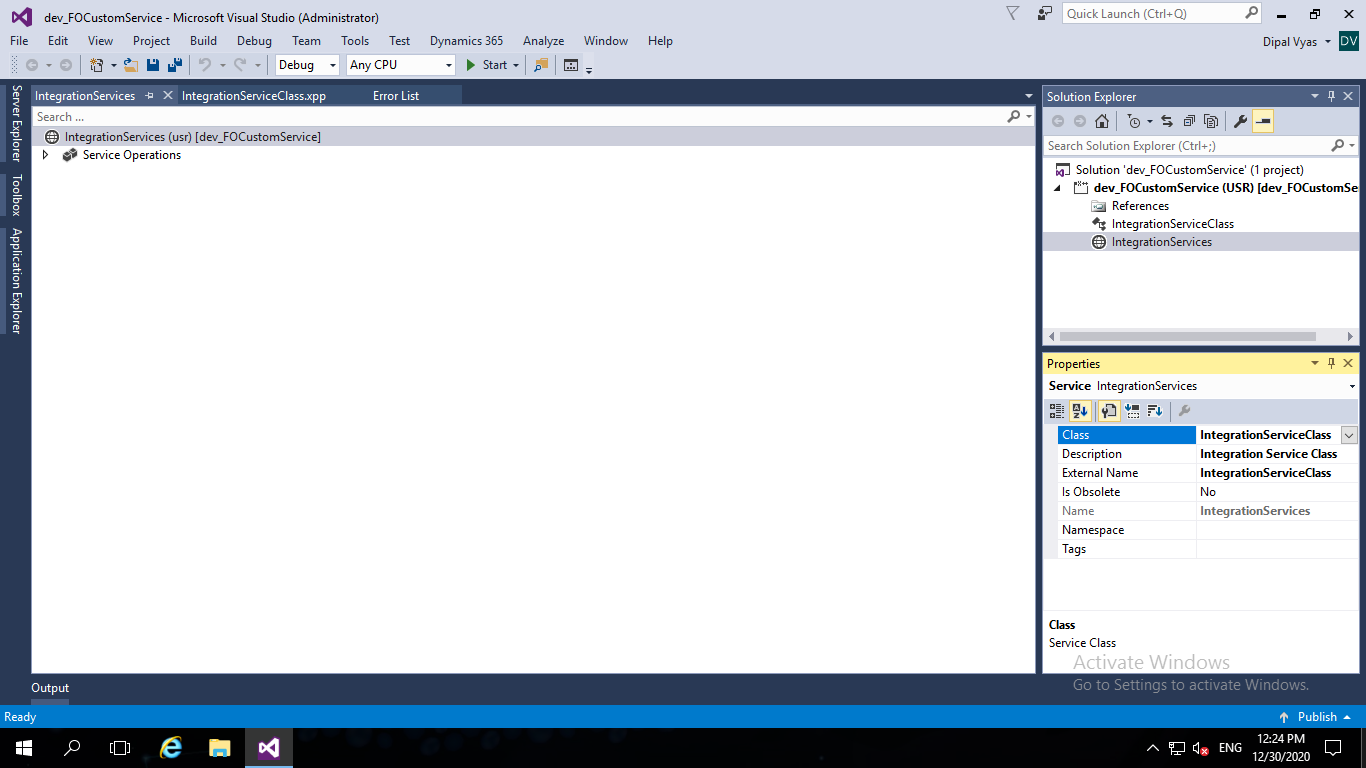
}

return custTable.AccountNum + " - " + custTable.name();

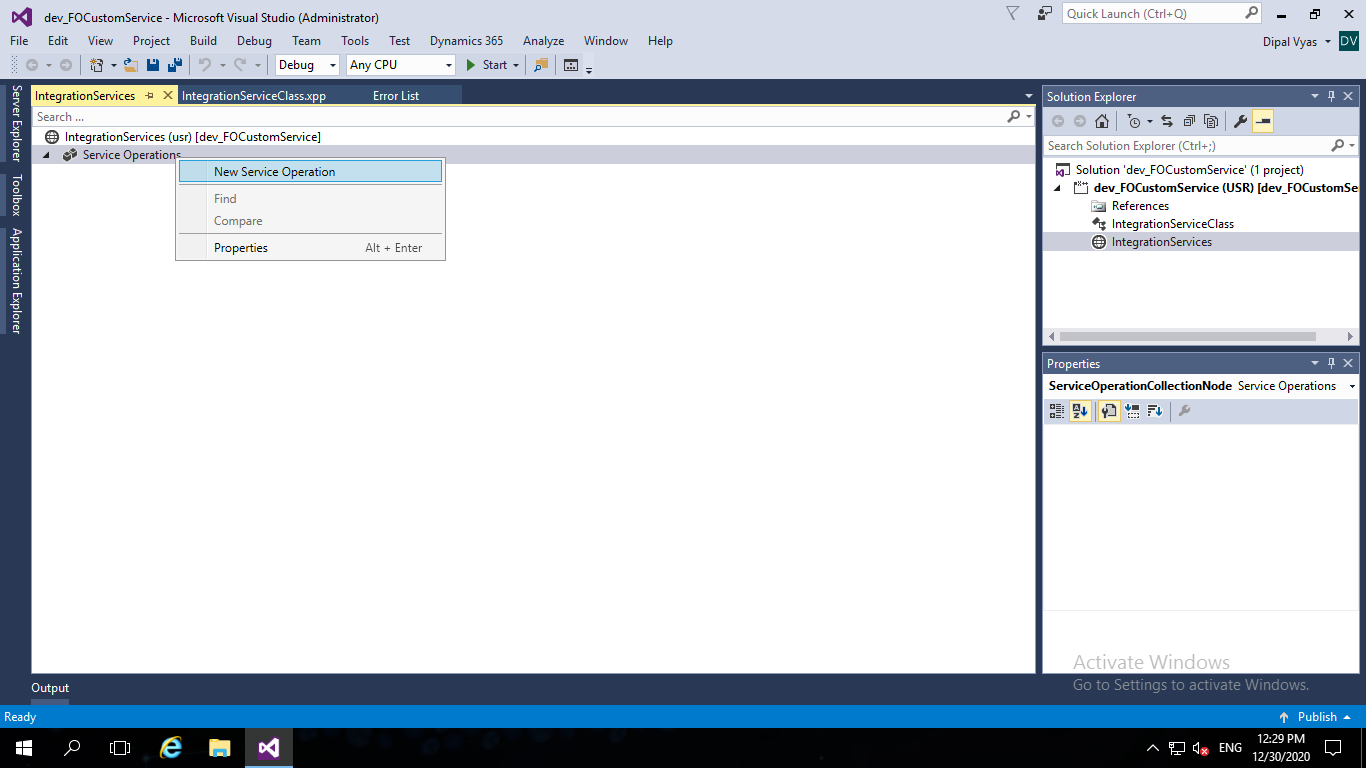
}

## **Create Service**

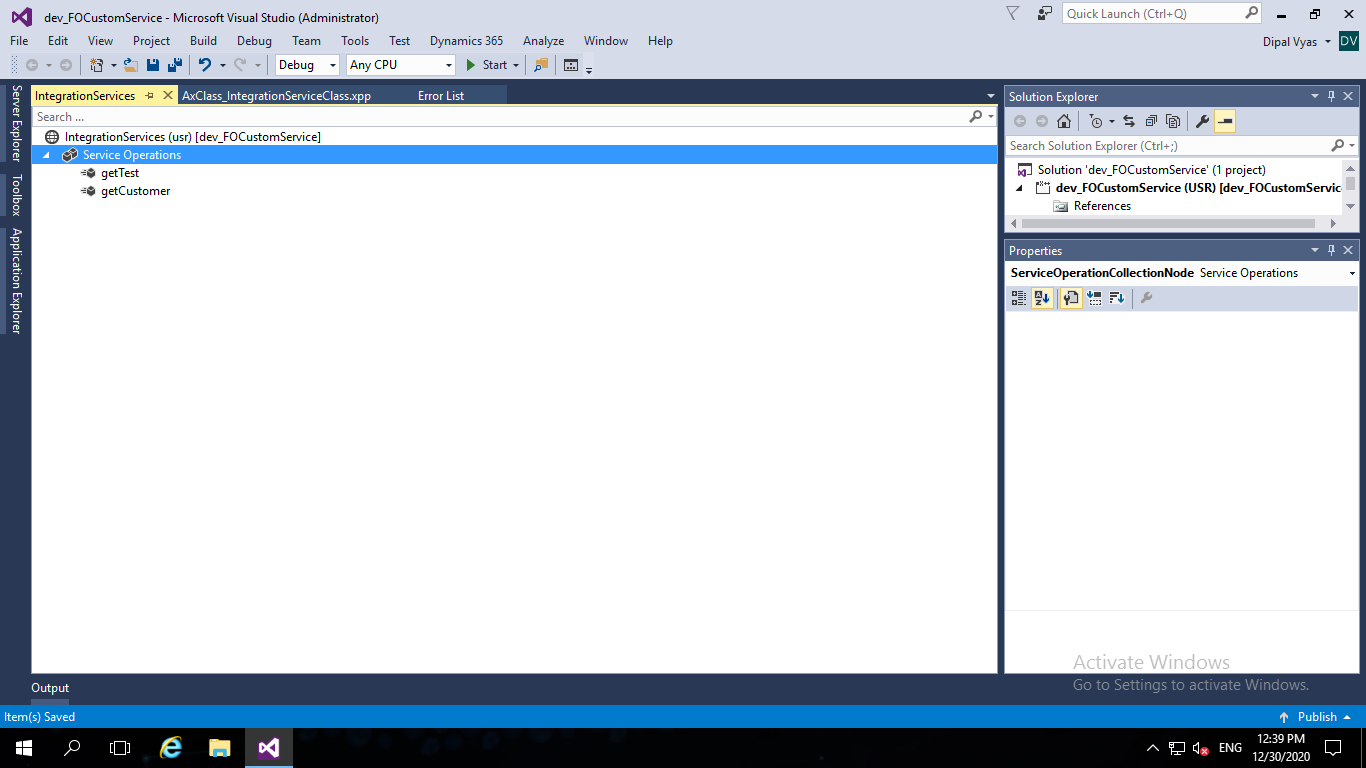
Right click on solution explorer and add service named “IntegrationServices”. Set Properties as shown in the screen below:



Right click on Service Operations and add New Service Operations for the methods created in Service class.

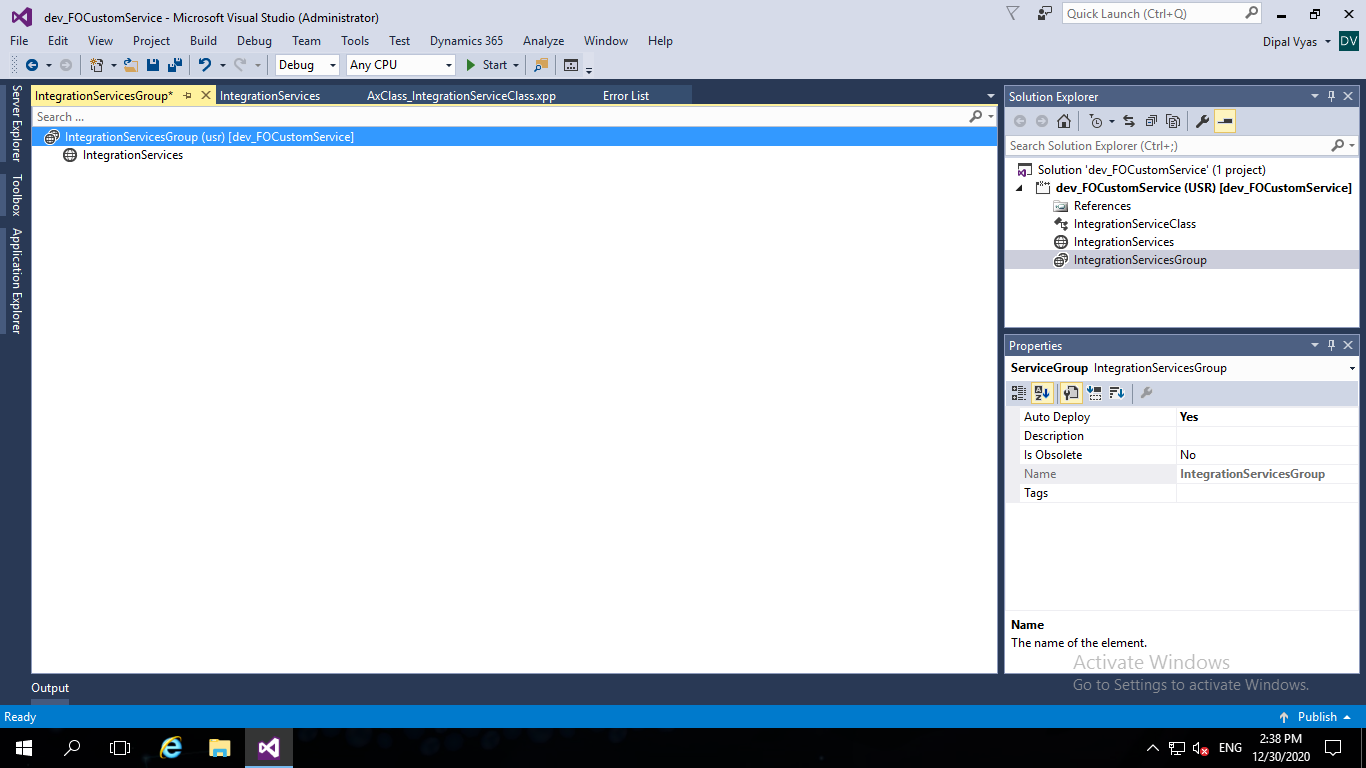


Set Name and Method for the created service operations.



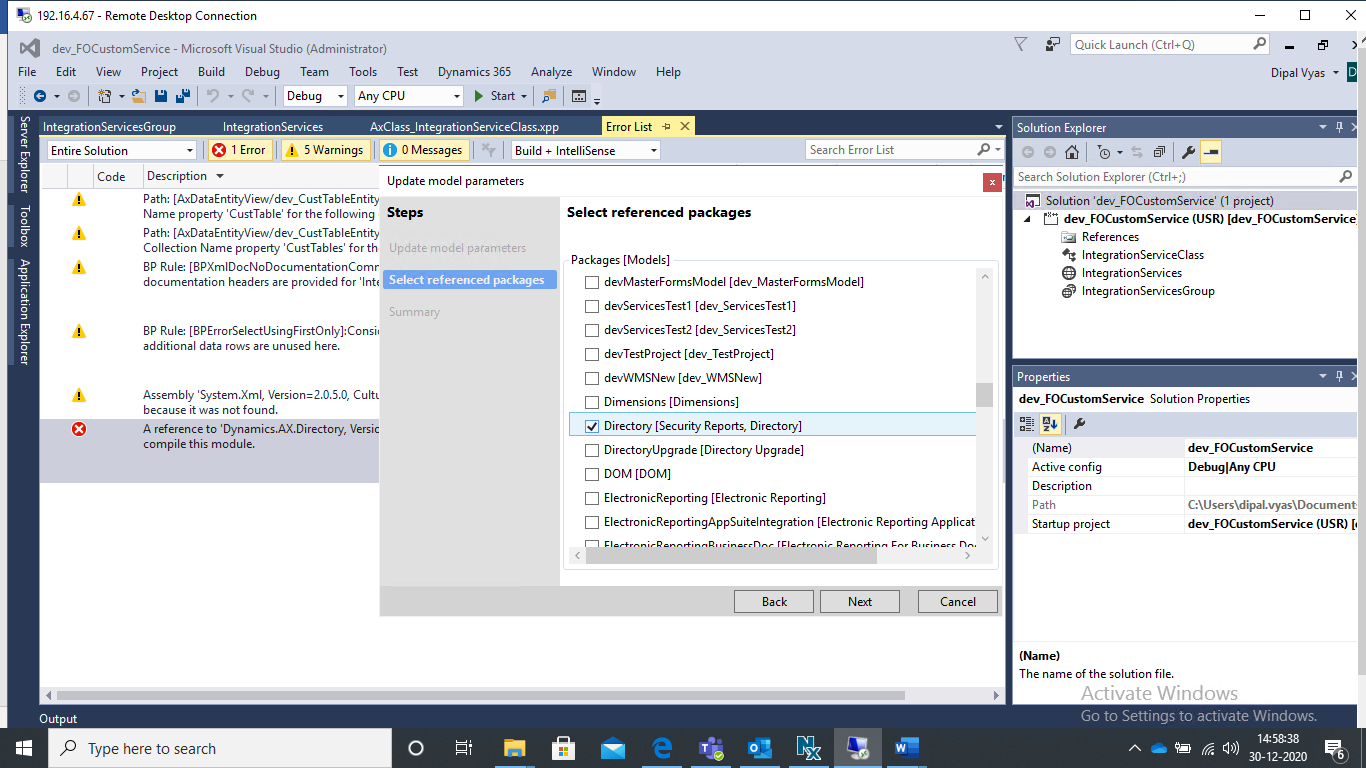
## **Create Service Group**

Add Service Group named “IntegrationServicesGroup” and add new service named “IntegrationServices”. Set Auto Deploy to “Yes”.

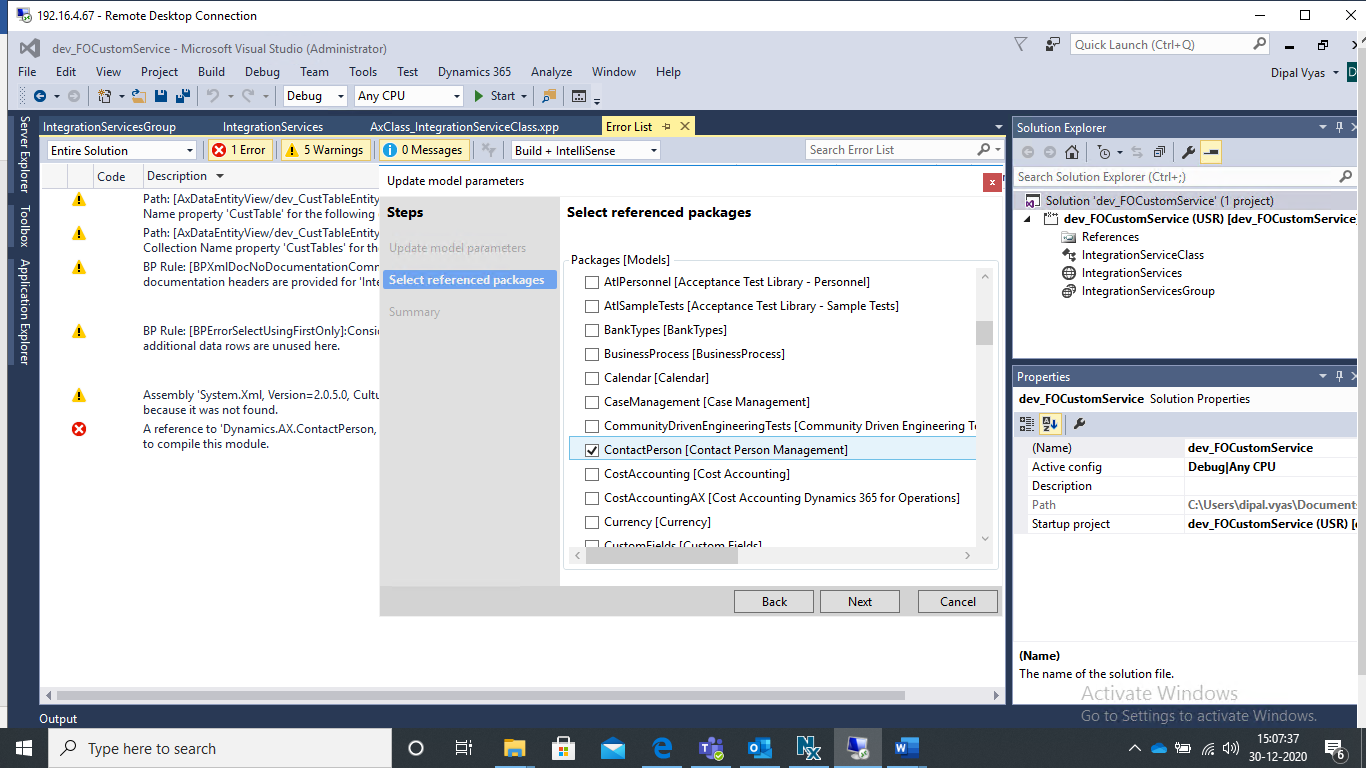


Build the project.

Update model and add “Directory” package to solve error of “Dynamics.AX.Directory”.



Update model and add “ContactPerson” Package to solve error of “Dynamics.AX.ContactPerson”.



Build the project. On building the project, service will be automatically deployed.

## **Verify service on D365 FO URL**

**Service URL**  
https://usnconeboxax1aos.cloud.onebox.dynamics.com/api/services/**[ServiceGroupName]/[ServiceName]/[MethodName]**

Verify service on D365 FO URL

[https://usnconeboxax1aos.cloud.onebox.dynamics.com/api/services/**IntegrationServicesGroup/IntegrationServices/getTest**](https://usnconeboxax1aos.cloud.onebox.dynamics.com/api/services/IntegrationServicesGroup/IntegrationServices/getTest)

# **Register New Application on Azure Portal**

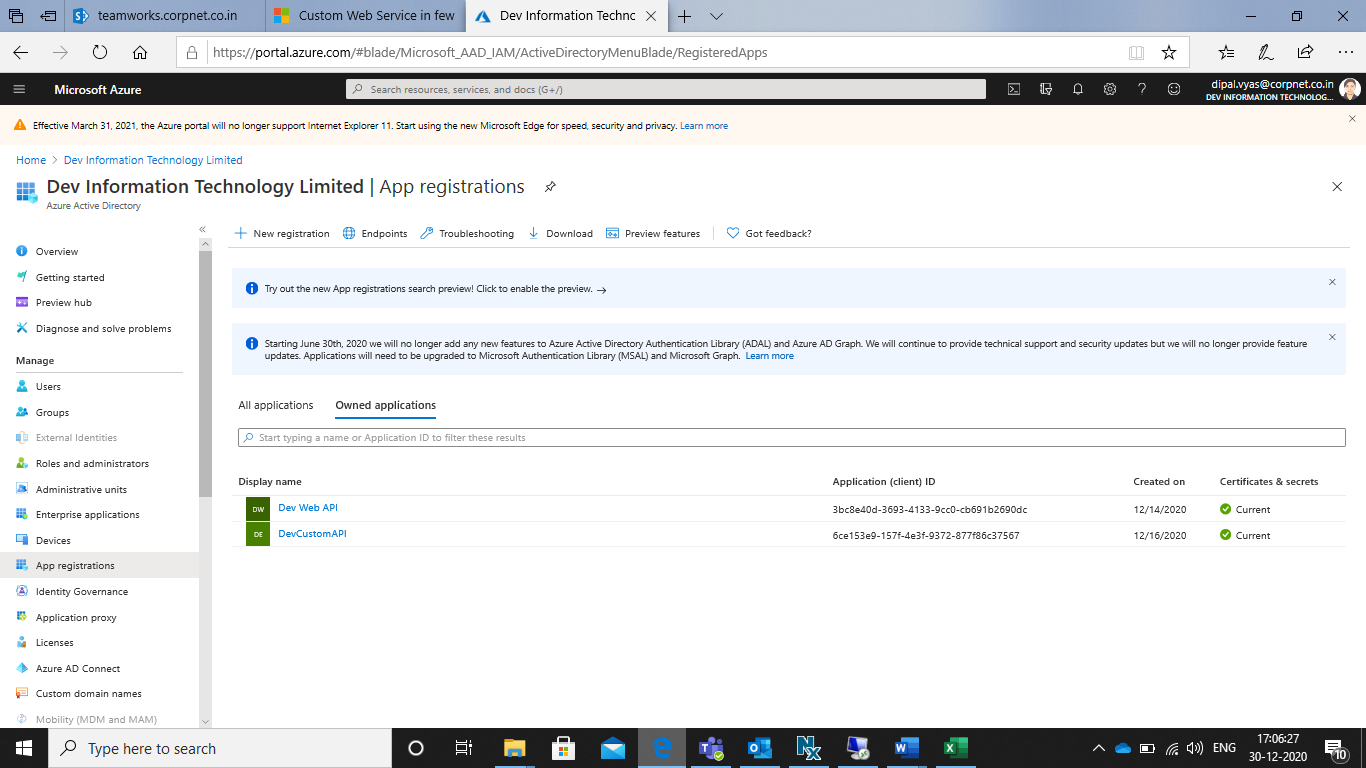
Ref Links

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

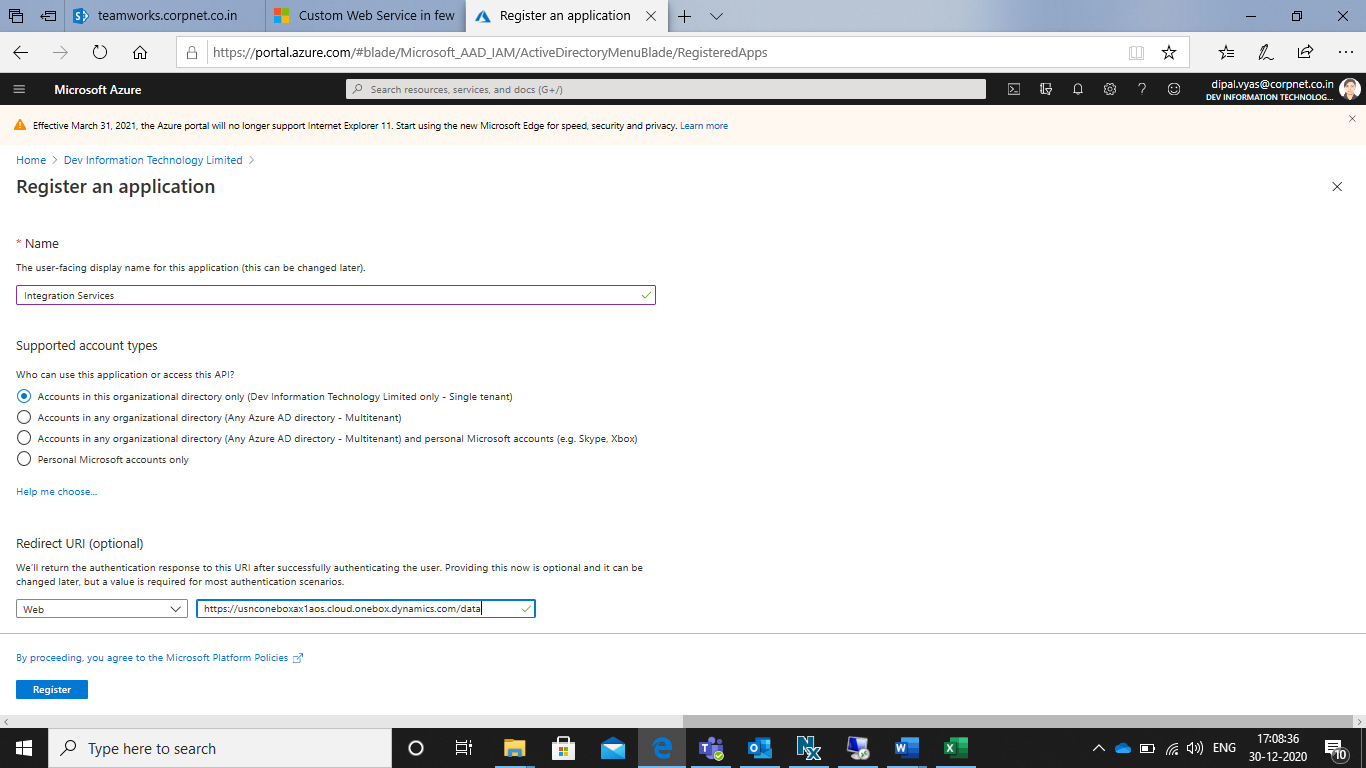
<https://www.d365ug.com/blogs/nandita-nityanandam/2018/11/20/third-party-integration-in-d365fo-using-odata>

## **New Registration**

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



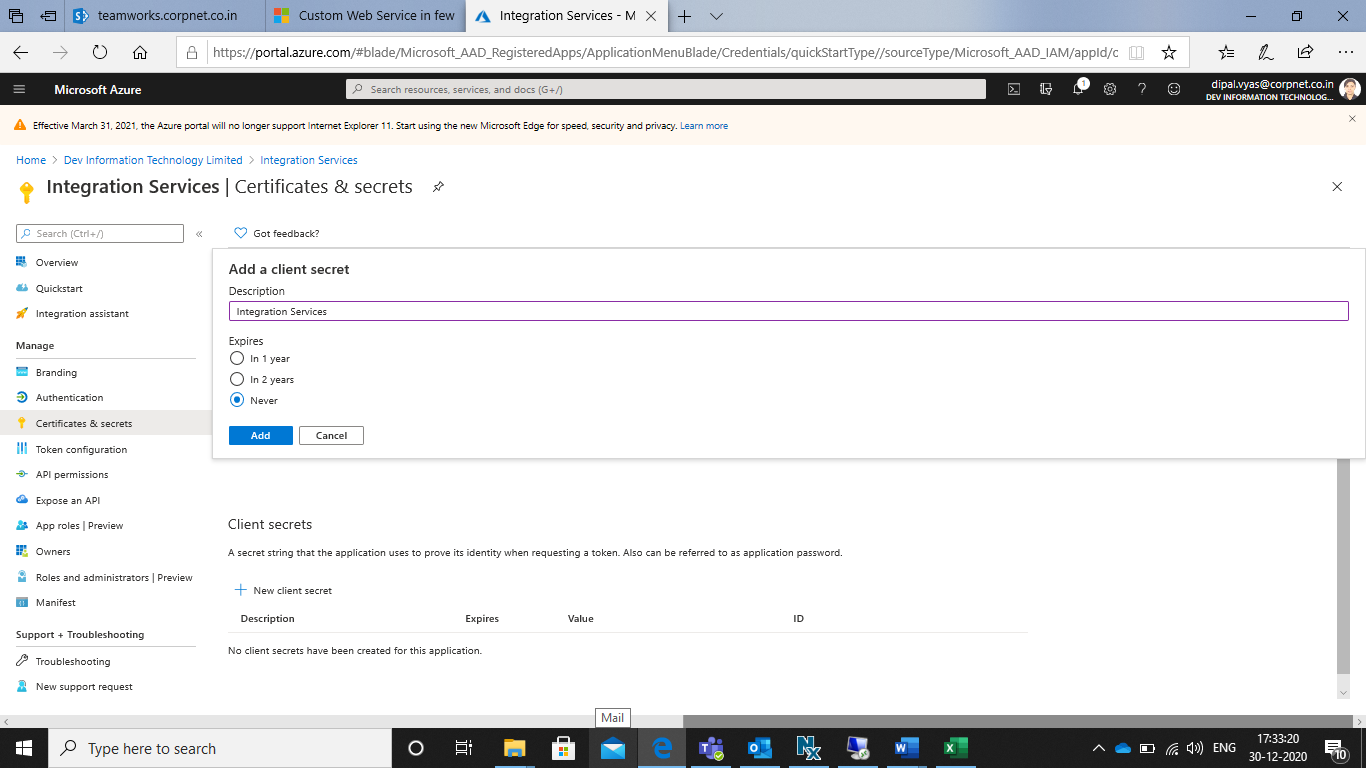
Click on **“New Registration”.** Enter details as shown in the screen below and register the application.



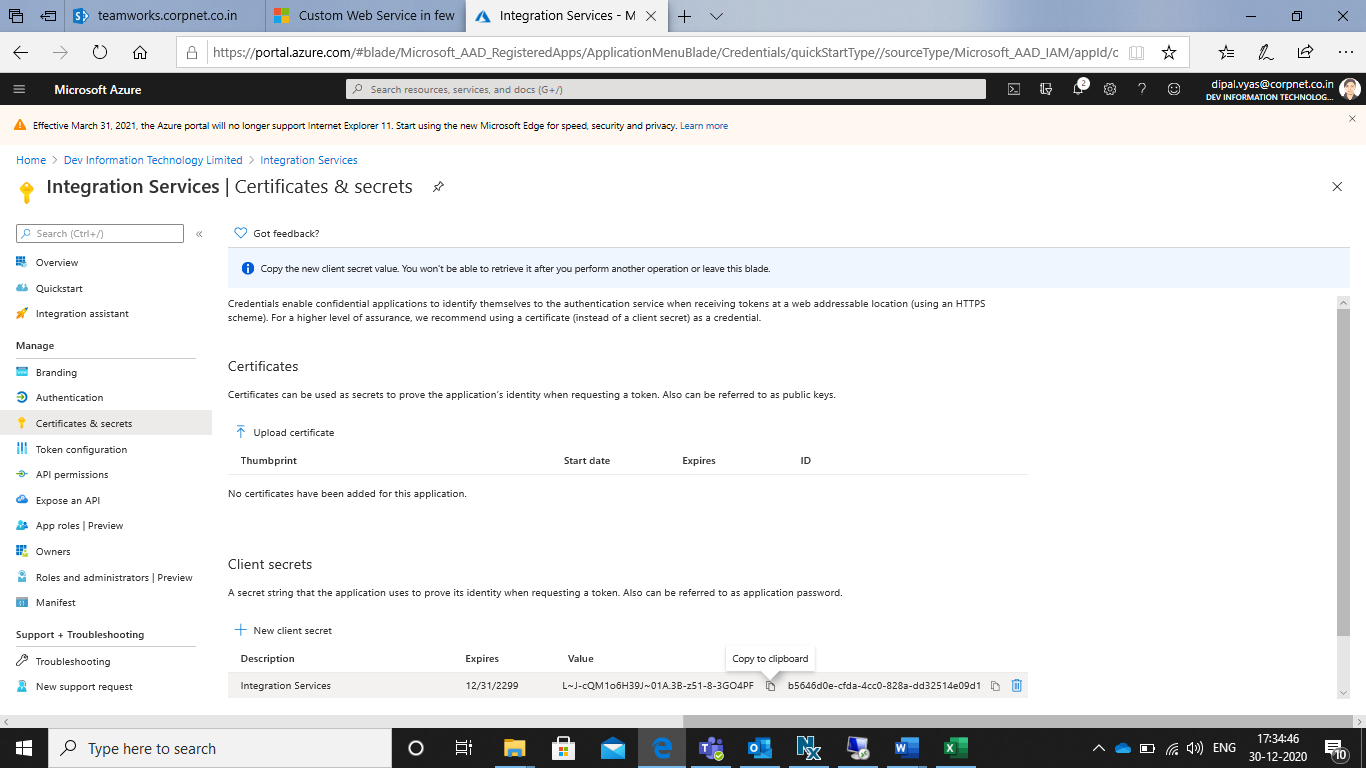
## **Generate Secret Key**

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

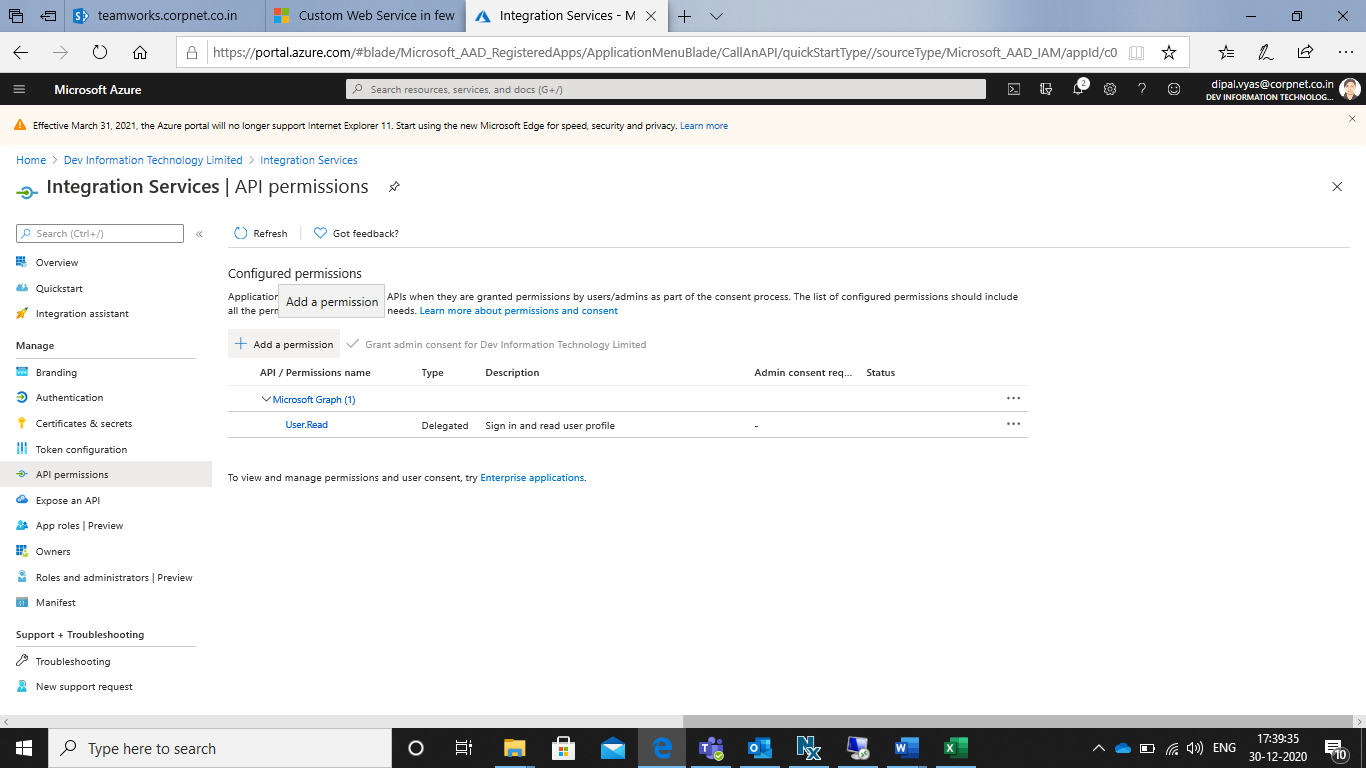


Secret key value: L~J-cQM1o6H39J~01A.3B-z51-8-3GO4PF

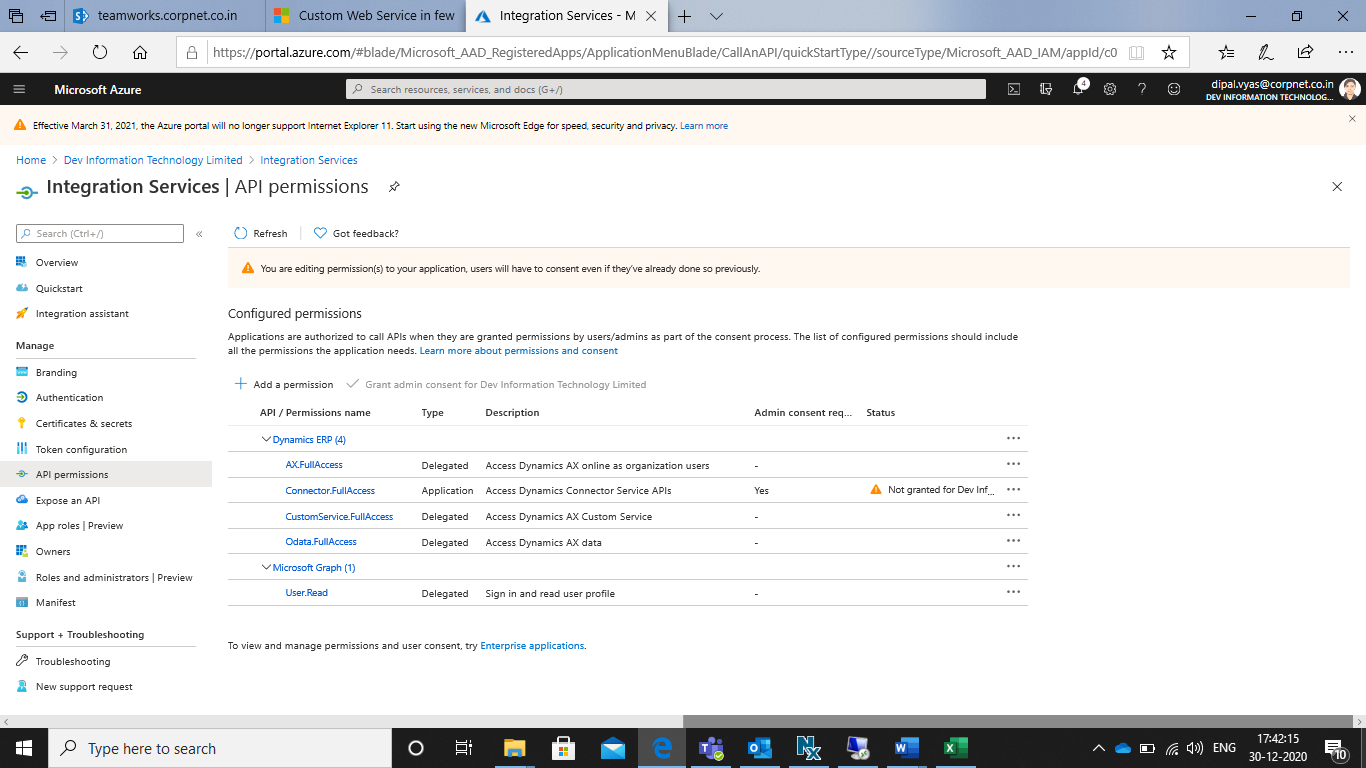
This value will not be visible again so do not miss to copy this.

## **API Permissions**

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



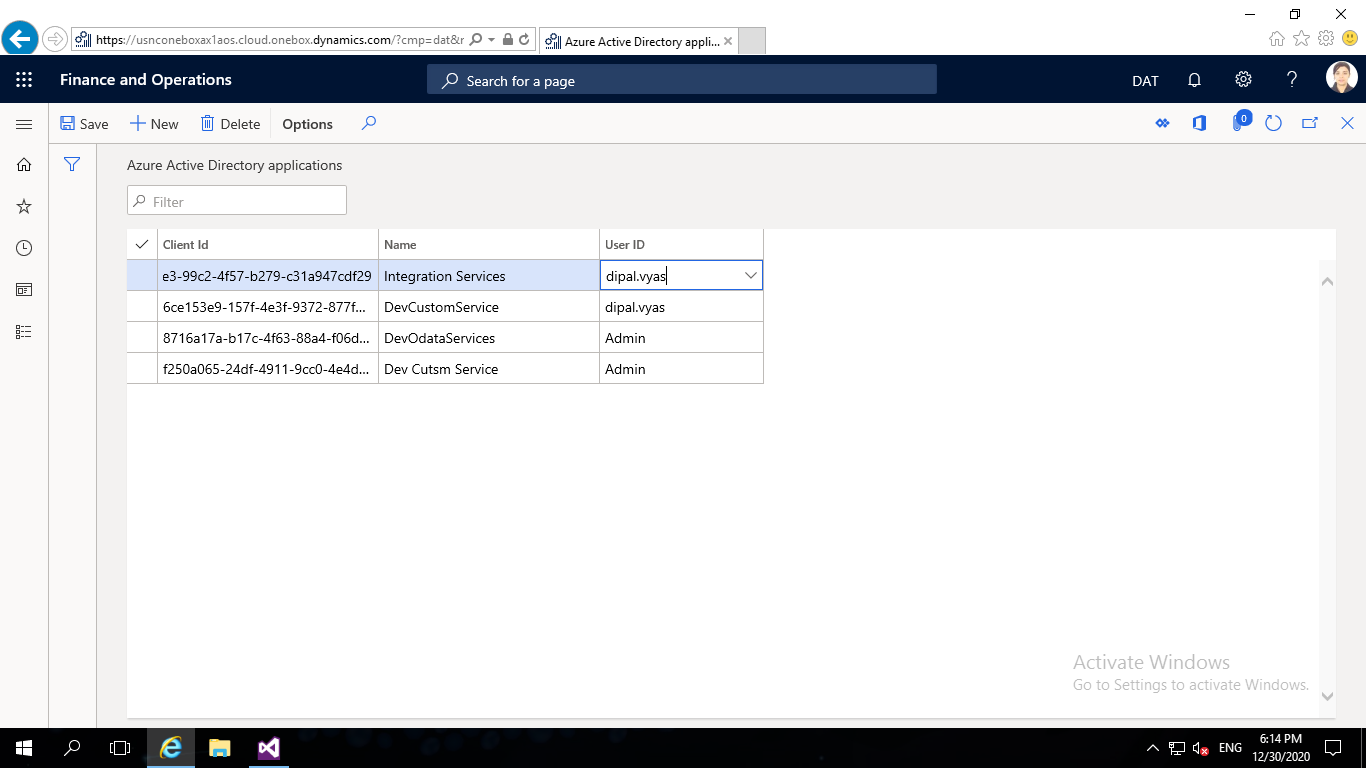
Add a permission >> Microsoft APIs >> Dynamics ERP. Add all delegated and application permissions, click on Add permission. Permission will be added as shown in the screen below.



# **Add Application in Dynamics AX FO**

System Administration >> Azure Active Directory applications >> New

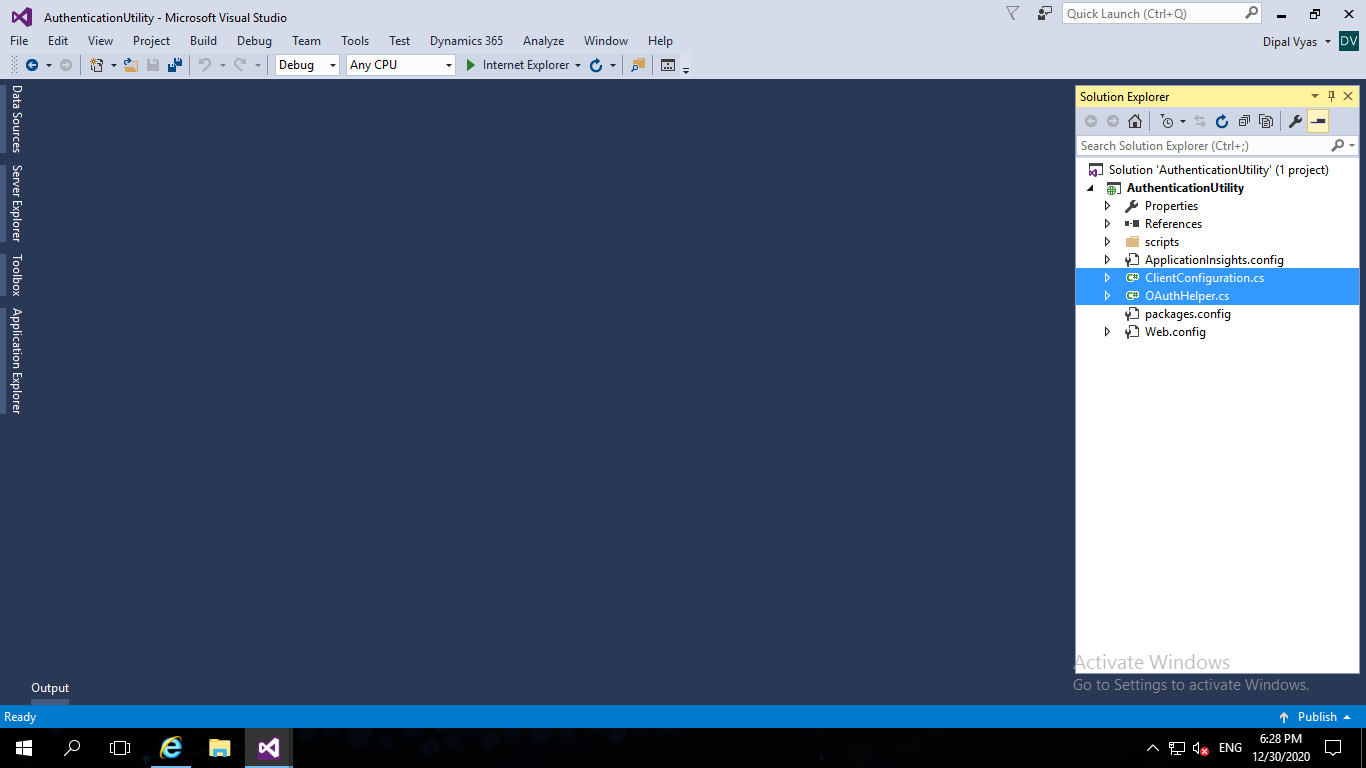
Enter Azure application client id in Application Id, application name and user id.



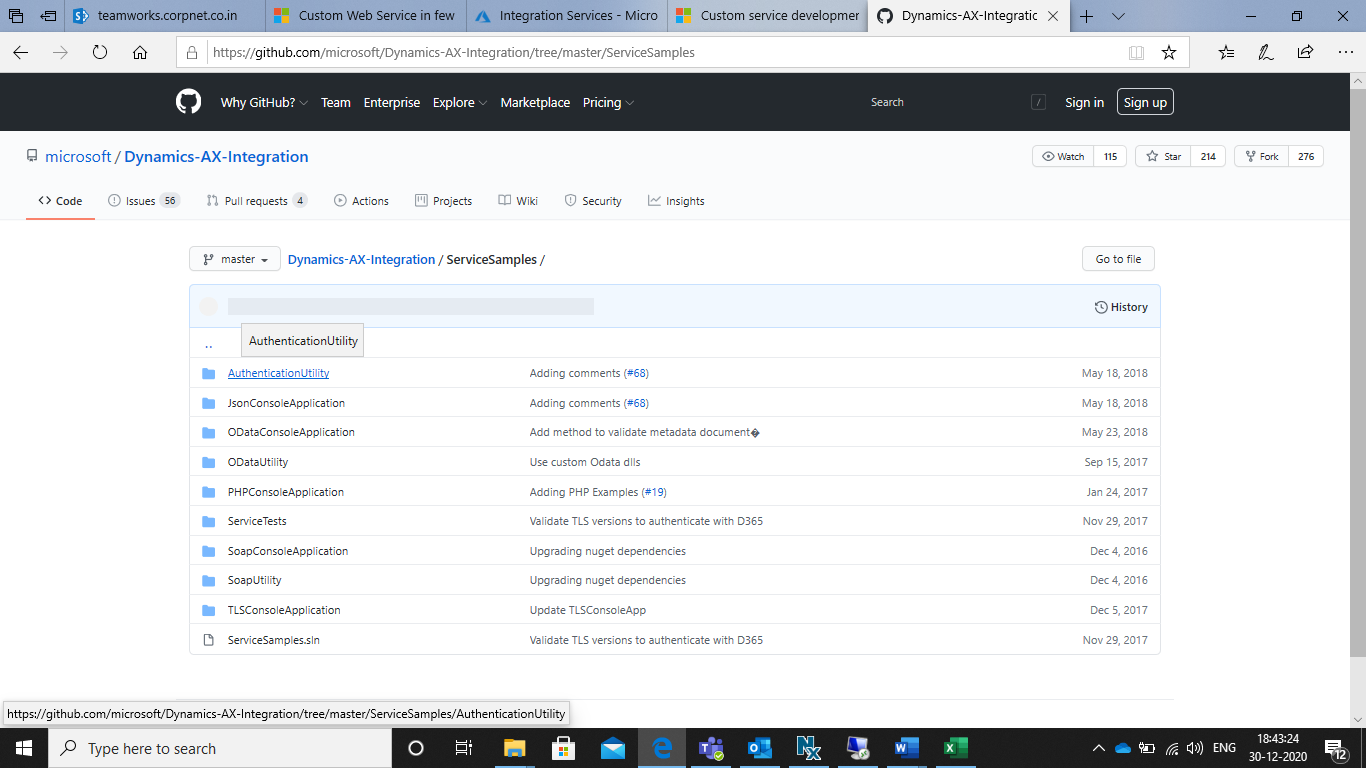
# **Consume AX custom service in Third Party Application**

## **AuthenticationUtility Application**

Create .net application for AuthenticationUtility. Add two classes “ClientConfiguration.cs” and “OAuthHelper.cs”.



You can copy the code in these two classes from **Microsoft Dynamics AX Integration GitHub repository**.



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

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AuthenticationUtility

{

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 = "dipal.vyas@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 = "c07b97e3-99c2-4f57-b279-c31a947cdf29",

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

ActiveDirectoryClientAppSecret = "L~J-cQM1o6H39J~01A.3B-z51-8-3GO4PF",

// 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; }

}

}

Add “OAuthHelper.cs” class.

Before adding this class add directory for Microsoft.IdentityModel.Clients.ActiveDirectory from Tools >> NuGet Package Manger >> Package Manager Console with below command.

Install-Package Microsoft.IdentityModel.Clients.ActiveDirectory -Version 5.2.8

Add OAuthHelper code as below and build the application.

using Microsoft.IdentityModel.Clients.ActiveDirectory;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AuthenticationUtility

{

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();

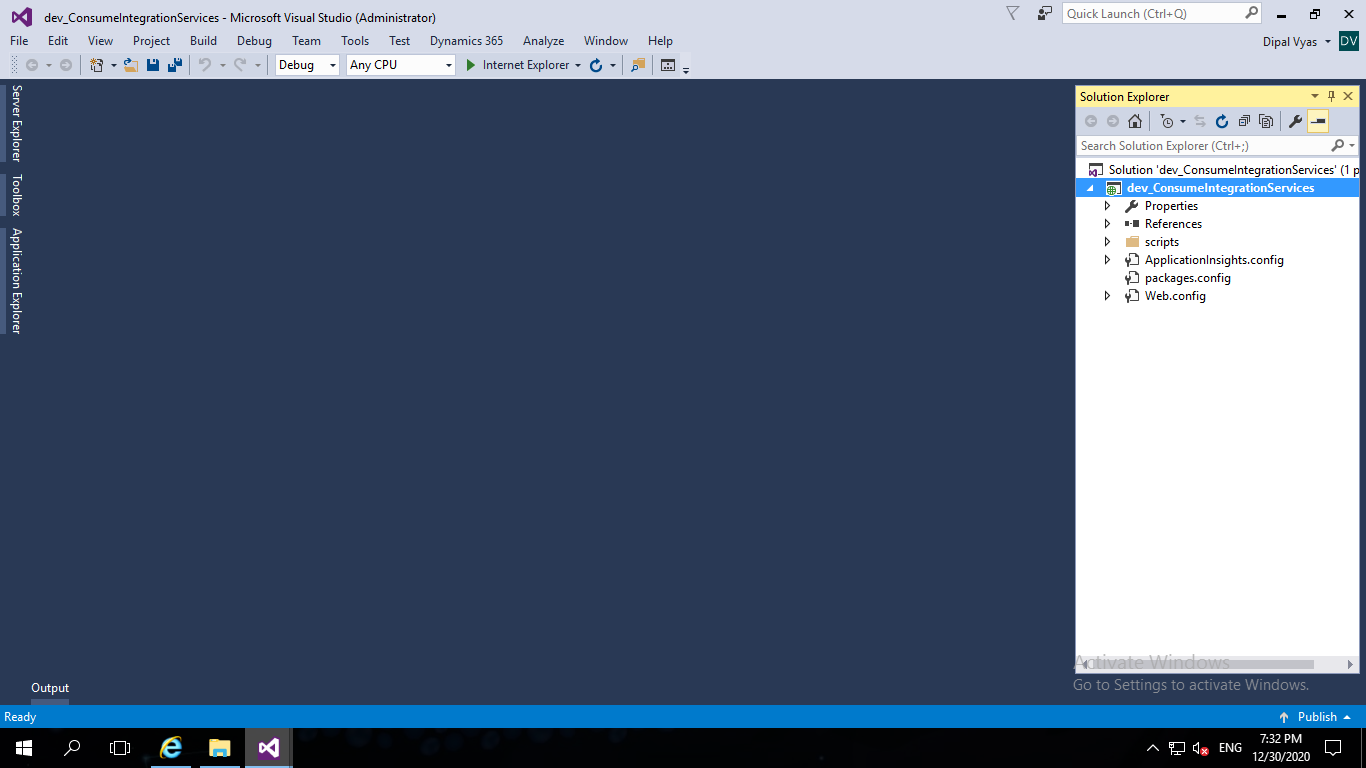
}

}

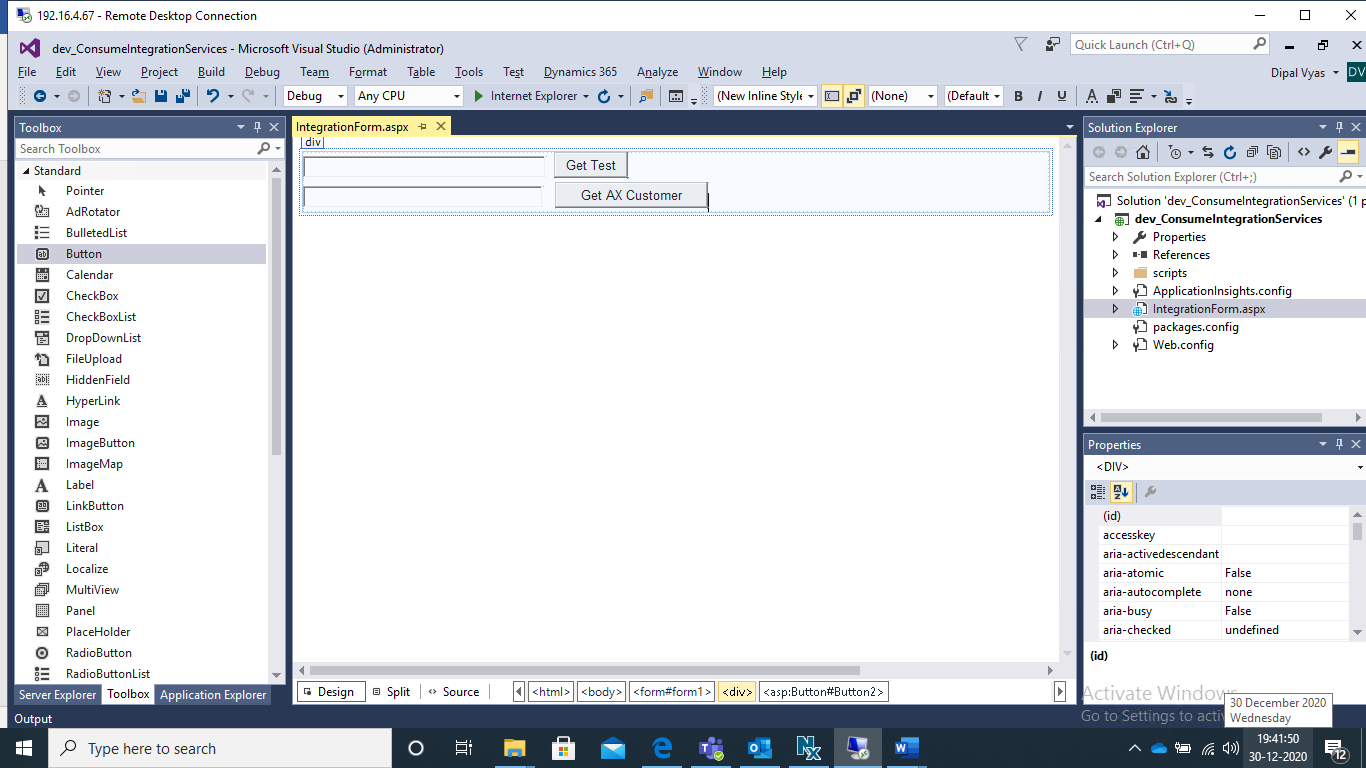
}

## **Thirdparty Application**

Create new .net application “dev\_ConsumeIntegrationServices”. Add reference of AuthenticationUtility.dll in this application.

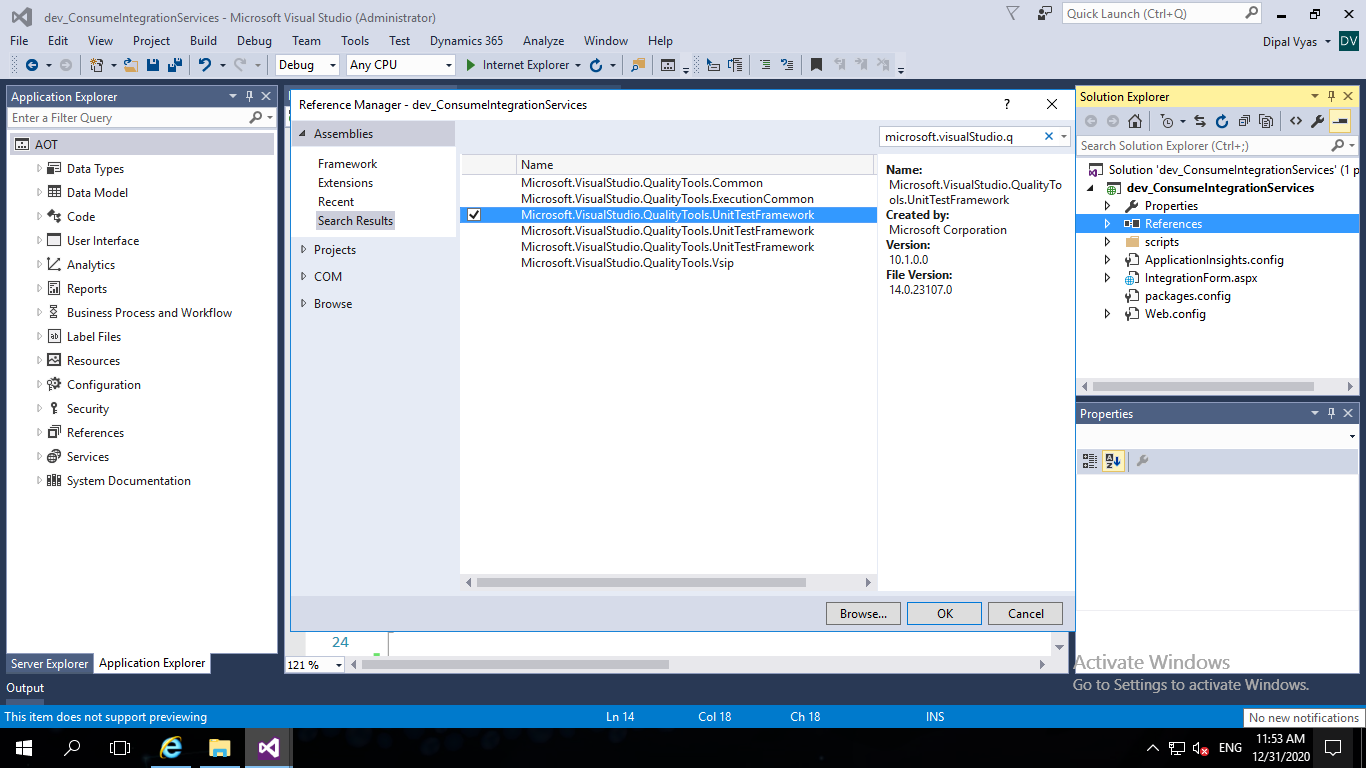


Add Web Form with TextBox and Button to access service of AX as shown below:



Before applying the code, add references shown below:

Microsoft.VisualStudio.QualityTools.UnitTestFramework



To use Newtonsoft.Json library, install package Install-Package Newtonsoft.Json -Version 12.0.3 from Tools >> NuGet Package Manager >> Package Manager Console

Below is the code to access services of AX in the .net application:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using AuthenticationUtility;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

//using SoapUtility.UserSessionServiceReference;

using System;

using System.Text;

using System.IO;

using System.Net;

namespace dev\_ConsumeIntegrationServices

{

public partial class IntegrationForm : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

#region Method - without Parameters

protected void btnGetTest\_Click(object sender, EventArgs e)

{

string GetTest = ClientConfiguration.Default.UriString + "api/services/IntegrationServicesGroup/IntegrationServices/getTest";

var request = HttpWebRequest.Create(GetTest);

request.Headers[OAuthHelper.OAuthHeader] = OAuthHelper.GetAuthenticationHeader(true);

request.Method = "POST";

request.ContentLength = 0;

using (var response = (HttpWebResponse)request.GetResponse())

{

using (Stream responseStream = response.GetResponseStream())

{

using (StreamReader streamReader = new StreamReader(responseStream))

{

string responseString = streamReader.ReadToEnd();

Assert.AreEqual(HttpStatusCode.OK, response.StatusCode);

Assert.IsFalse(string.IsNullOrEmpty(responseString));

//Console.WriteLine(responseString);

TextBox1.Text = responseString;

}

}

}

}

#endregion

#region Get Customer - Method with Parameters

protected void btnGetCustomer\_Click(object sender, EventArgs e)

{

string getCustomer = ClientConfiguration.Default.UriString + "api/services/IntegrationServicesGroup/IntegrationServices/getCustomer";

var request = HttpWebRequest.Create(getCustomer);

request.Headers[OAuthHelper.OAuthHeader] = OAuthHelper.GetAuthenticationHeader(true);

request.Method = "POST";

// Pass parameters in JsonSerialize Object start

var requestContract = new

{

strDataAreaId = "USMF",

strAccountNum = "DE-001"

};

var requestContractString = JsonConvert.SerializeObject(requestContract);

using (var stream = request.GetRequestStream())

{

using (var writer = new StreamWriter(stream))

{

writer.Write(requestContractString);

}

}

// Pass parameters in JsonSerialize Object end

using (var response = (HttpWebResponse)request.GetResponse())

{

using (Stream responseStream = response.GetResponseStream())

{

using (StreamReader streamReader = new StreamReader(responseStream))

{

string responseString = streamReader.ReadToEnd();

Assert.AreEqual(HttpStatusCode.OK, response.StatusCode);

Assert.IsFalse(string.IsNullOrEmpty(responseString));

TextBox2.Text = responseString;

//Console.WriteLine(responseString);

}

}

}

}

}

#endregion

}

# **AX custom service with User credentials**

Two types of Azure application registration 1) Web API 2) Native Application

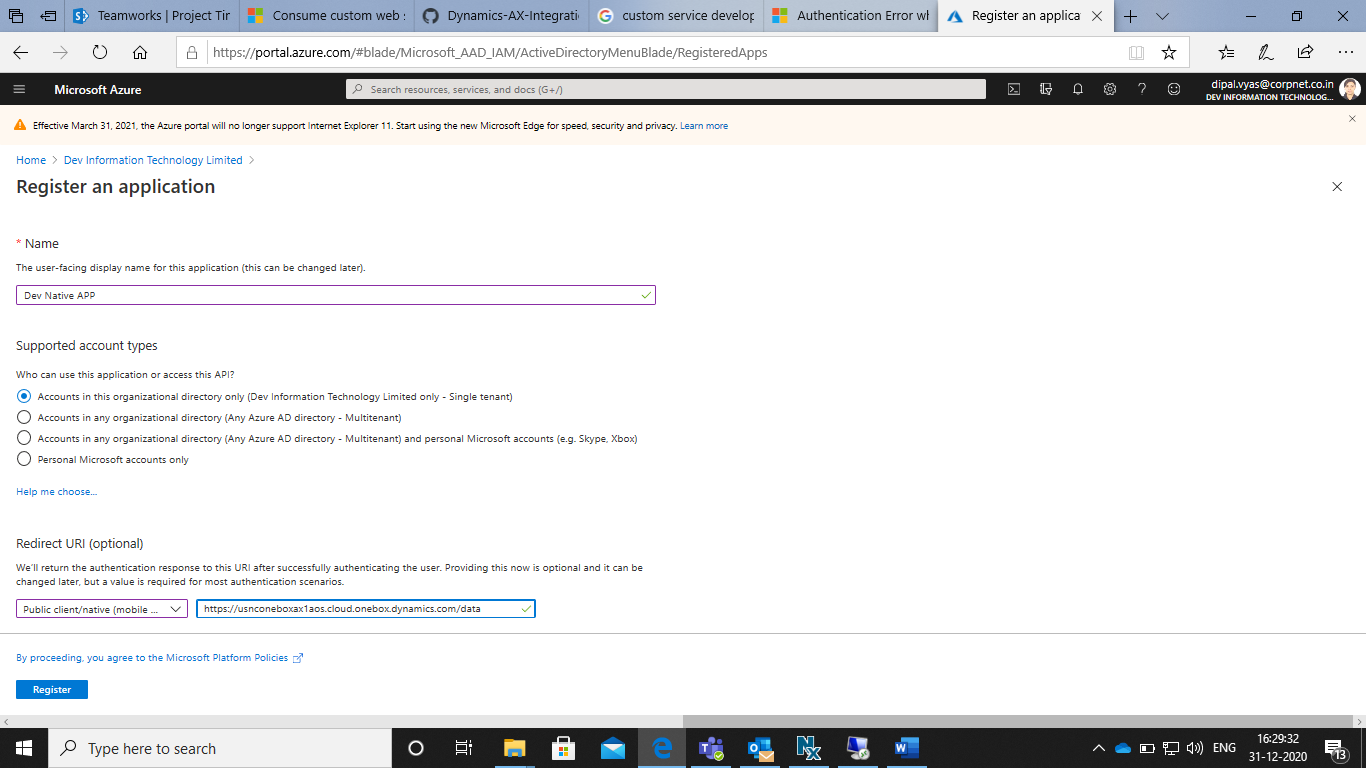
The Web API works with authentication token and Native Application works with user credentials.

In Third party application, passing “true” in parameter value is required to call the service with authentication token.

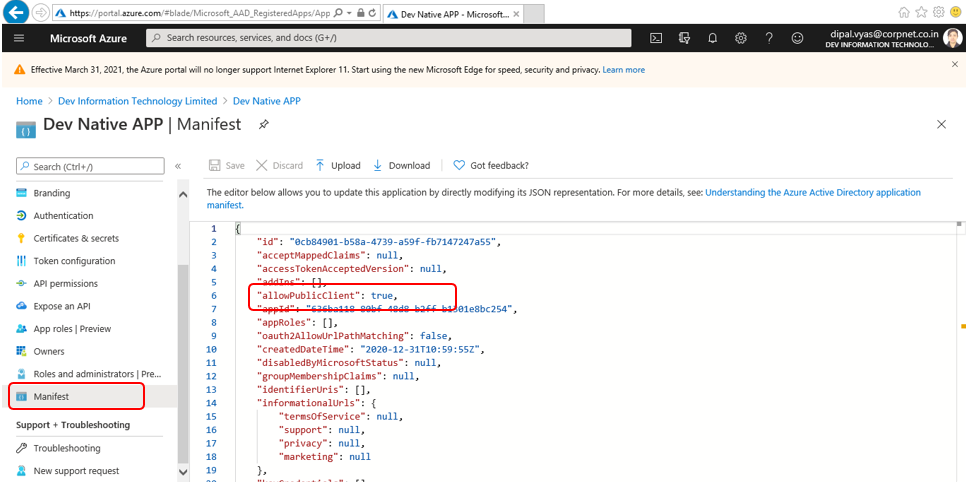
request.Headers[OAuthHelper.OAuthHeader] = OAuthHelper.GetAuthenticationHeader(true);

For Native Application, register an Azure application. Add API permissions and generate secret key:

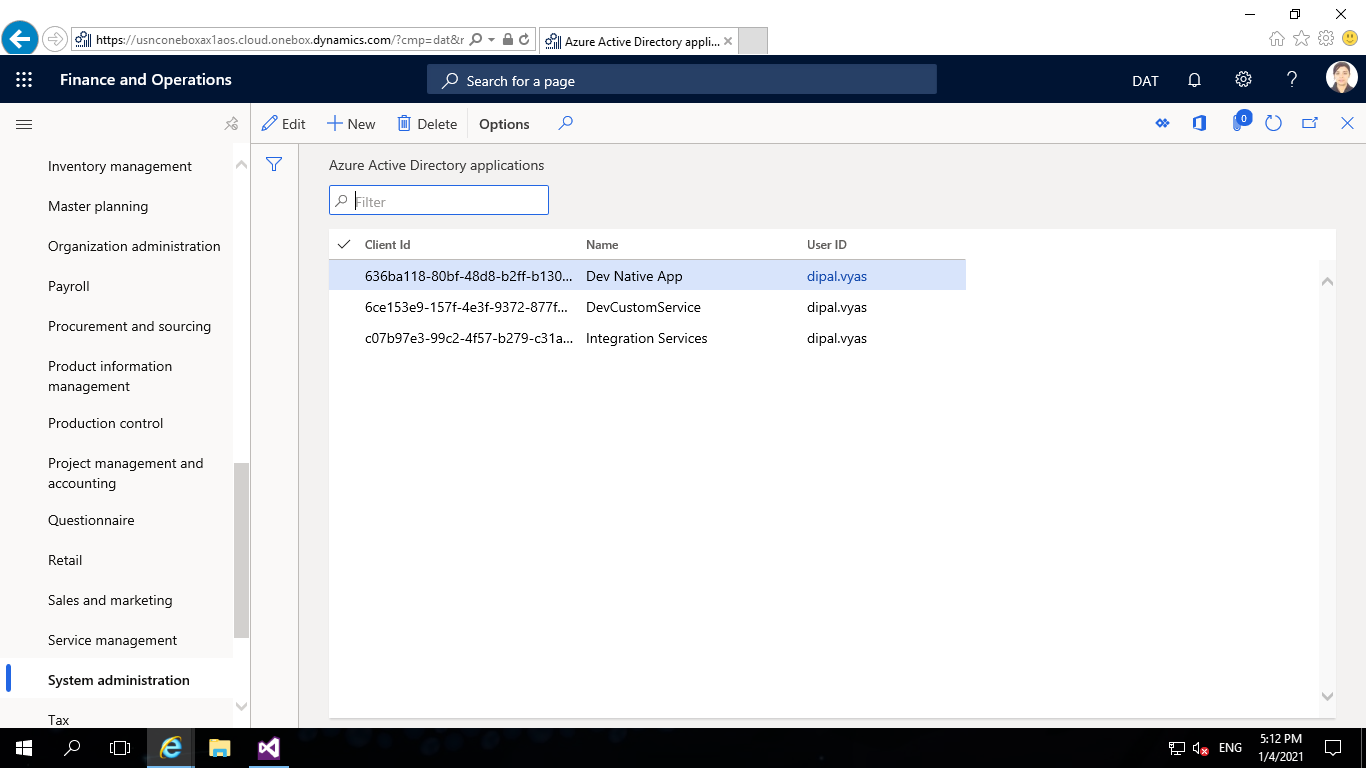
Secret Key Value: FQMg\_o.\_1g-MHB9.hT5Ubp5~Ct01SDt31d



To call the application with user credentials, set allowPublicClient to “true” in Azure Native Application >> Manifest



Add native application client in Azure Active Directory Application in D365 FO.



In Third party application,

* Set Client application id, secret value as below. Setting secret value is not mandatory.

ClientConfiguration.Default.ActiveDirectoryClientAppId = <Application Id >;

ClientConfiguration.Default.ActiveDirectoryClientAppSecret = <Secret Value>;

* Do not pass any value in below line. It will by default consider as false and call the service through user credentials. Debug the code and review.

request.Headers[OAuthHelper.OAuthHeader] = OAuthHelper.GetAuthenticationHeader( );