Programming Authentication with Azure Active Directory

Lab Time: 60 minutes

Lab Folder: c:\Student\Modules\07_AzureActiveDirectory\Labs

Lab Overview: In this lab you will configure your Azure Active Directory (Azure AD) tenant to provide authentication services to a custom application. You will do this by creating a new Azure application using the Azure portal.

Exercise 1: Register a New Application with Azure Active Directory

In this exercise, you will register a new application with Azure AD and you will configure the application's required permissions to access the Microsoft Graph API.
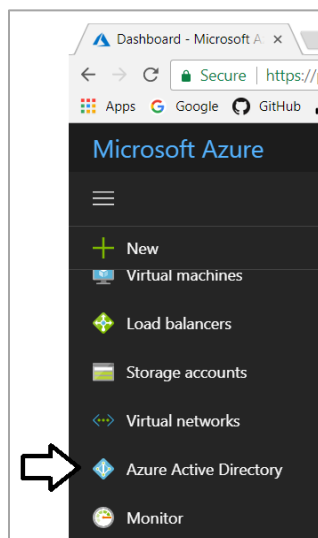
1. Log into the Azure Portal

   a) In the browser, navigate to the Azure portal at https://portal.azure.com.

   b) When you are prompted to log in, provide the credentials to log in with your Office 365 user account name.

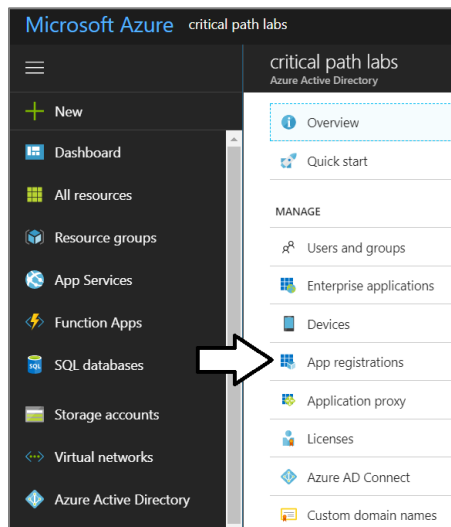   c) If you are prompted to start a tour of Microsoft Azure, click **Maybe later**.
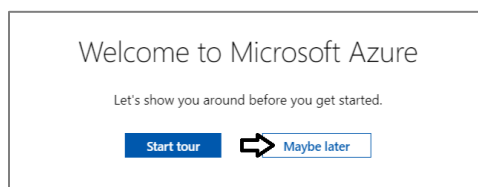
   d) Once you are log into the Azure portal, check the email address in the login menu in the upper right to make sure you are logged in the Azure portal with the correct identity.
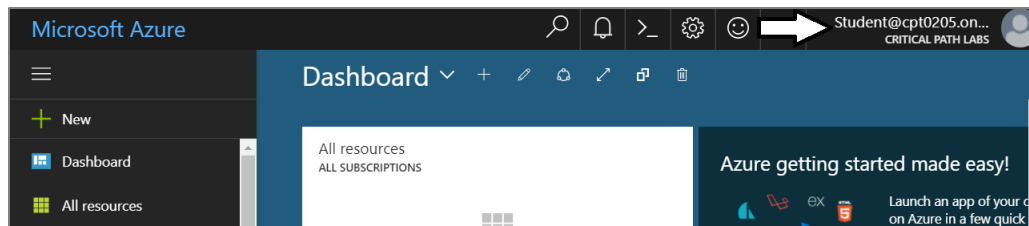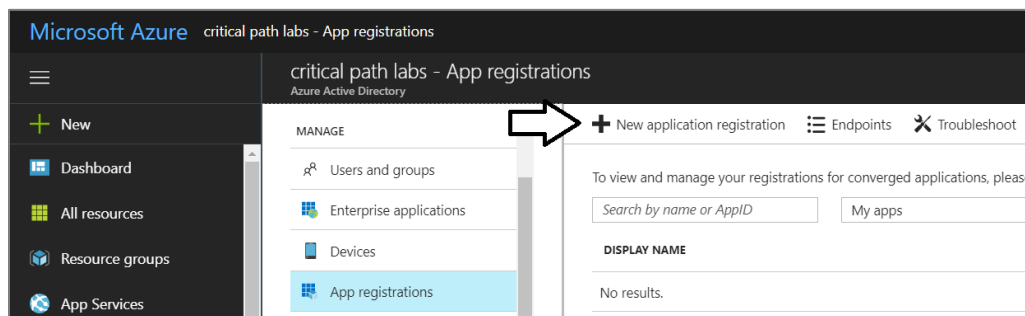

2. Register a new Azure application.

   a) In the left navigation, scroll down and click on the link for **Azure Active Directory**.

MSD365: Modern SharePoint and Office 365 Development
Module 07 Lab: Programming Authentication with Azure Active Directory

Live Lab Version: Sep 12, 2018

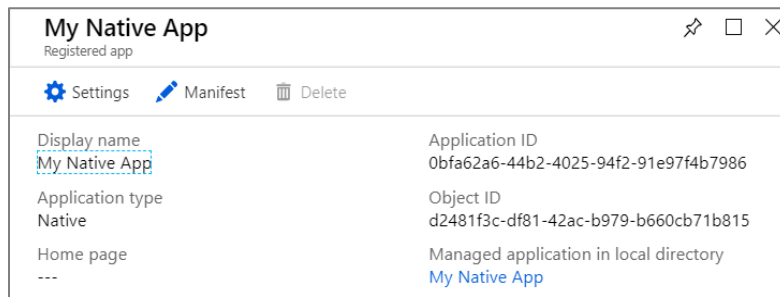b) Click the link for **App registration**.

c) Click **New application registration**.

d) In the Create dialog…
   i) Add a **Name** of **My Native App**.
   ii) Set the **Application type** to **Native**.
   iii) Set the **Redirect URI** to **https://localhost/app1234**.
   iv) Click the **Create** button to create the new application.

MSD365: Modern SharePoint and Office 365 Development
Module 07 Lab: Programming Authentication with Azure Active Directory

Live Lab Version: Sep 12, 2018

e) You should now see a view which displays the details of the new application.

3. Copy the GUID for the Application ID.
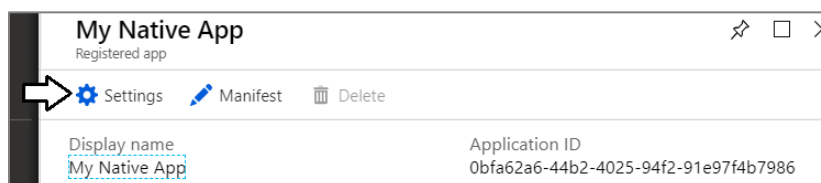
a) Copy the Application ID to the Windows clipboard.

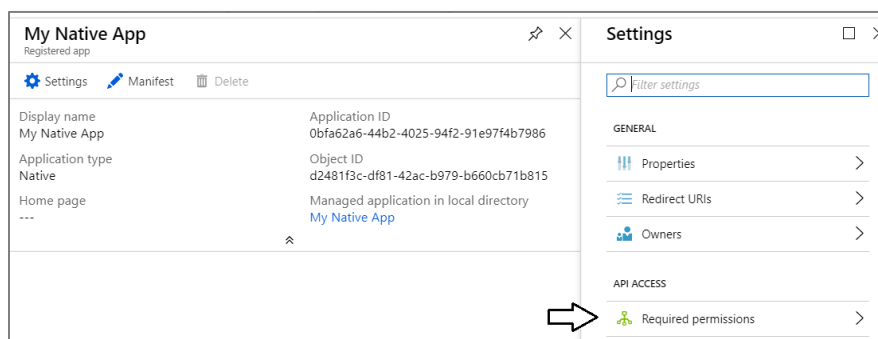Remember that the **Application ID** is also commonly called the **Client ID**.

b) Launch Notepad and paste the **Application ID** into a new document.

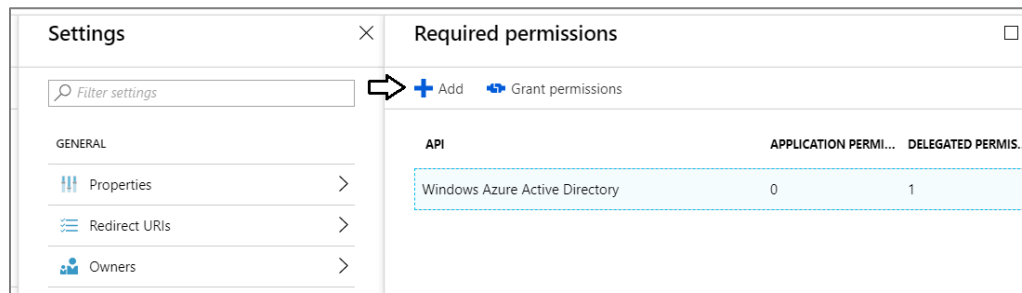c) Also add the **Redirect URI** value of **https://localhost/App1234**.

d) Click on the **Settings** link to configure application settings,

e) Click **Required permissions**.

MSD365: Modern SharePoint and Office 365 Development
Module 07 Lab: Programming Authentication with Azure Active Directory

Live Lab Version: Sep 12, 2018

f) Click the **Add** button on the **Required permissions** blade.
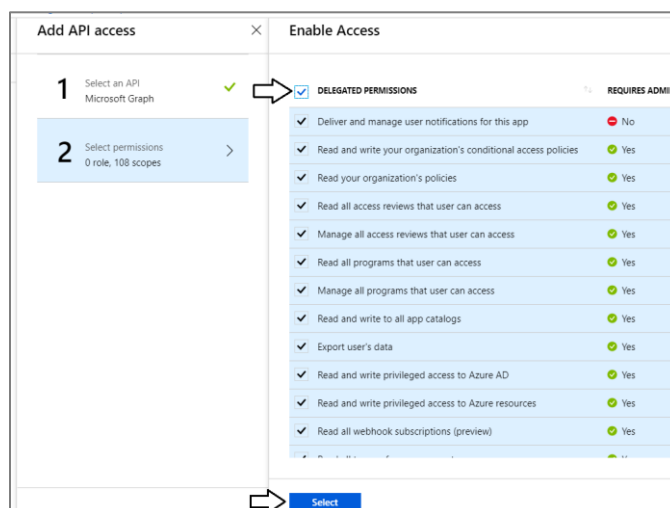

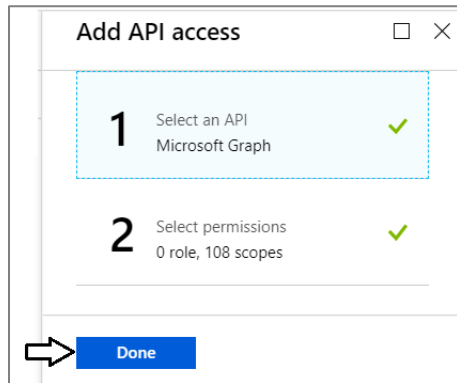g) Click the **Select an API** option in the **Add API access** blade.
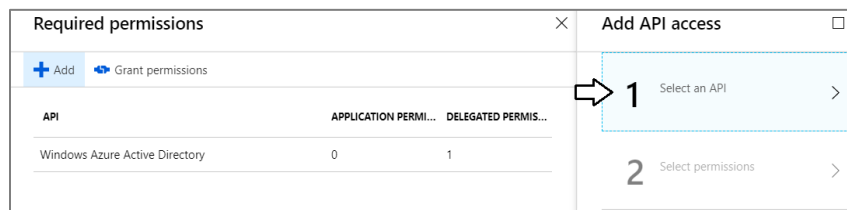
h) In the **Select an API** blade, click **Microsoft Graph**.
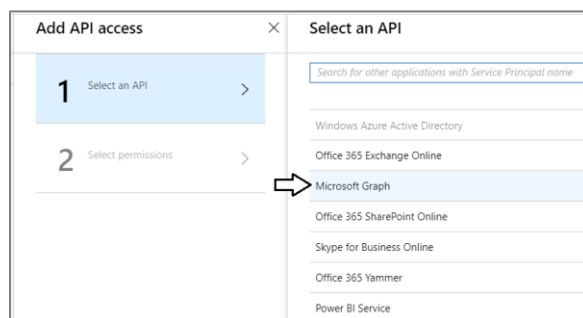
i) In the **Enable Access** blade, click the top checkbox for **DELEGATED PERMISSIONS** to select all the permissions.

j) Once you have selected all the permissions, click the **Select** button at the bottom of the blade.

MSD365: Modern SharePoint and Office 365 Development
Module 07 Lab: Programming Authentication with Azure Active Directory

Live Lab Version: Sep 12, 2018

k) Click the **Done** button at the bottom of the **Add API Access** blade.

l) At this point, you should be able to verify that the Microsoft Graph has been added to the **Required permissions** list.
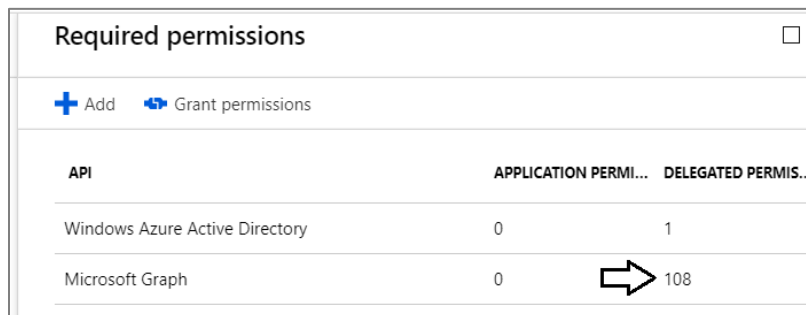
You can see that the Microsoft Graph API currently support 108 different delegated permissions. That's a pretty large number!

You are now done registering your application with Azure AD.

Exercise 2: Call the Microsoft Graph API using Direct REST Calls

In this exercise, you will create a simple C# Console application to call into the Microsoft Graph API.

1. Create a new C# Console application in Visual Studio.
   a) Launch Visual Studio 2017.
   b) Create a new project by running the **File > New Project** command.
   c) Select a project type of Console App from the Visual C# project templates.
   d) Give the project a name of **MicrosoftGraphNativeApp** and click **OK**.

MSD365: Modern SharePoint and Office 365 Development
Module 07 Lab: Programming Authentication with Azure Active Directory

Live Lab Version: Sep 12, 2018
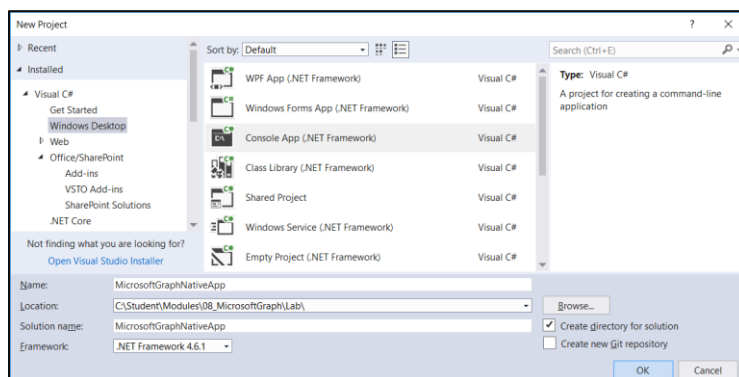
e) You should now have a new project named **MicrosoftGraphNativeApp**.
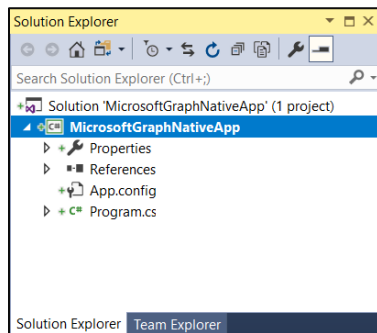
2. Add a NuGet package to the project for the Azure Active Directory Authentication library.

   a) Right-click the top-level node for the **MicrosoftGraphNativeApp** project and select **Manage NuGet Packages…**.
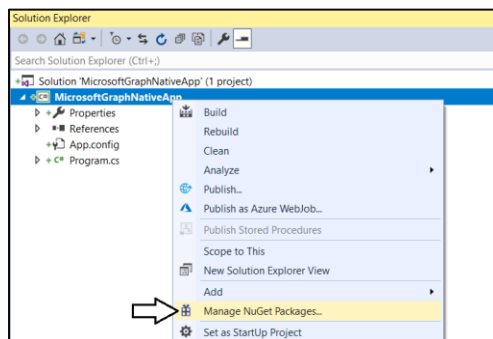
   b) Click the **Browse** tab and type **ADAL** into the search box.

   c) Locate the package **Microsoft.IdentityModel.Clients.ActiveDirectory** which is the Active Directly Authentication library.
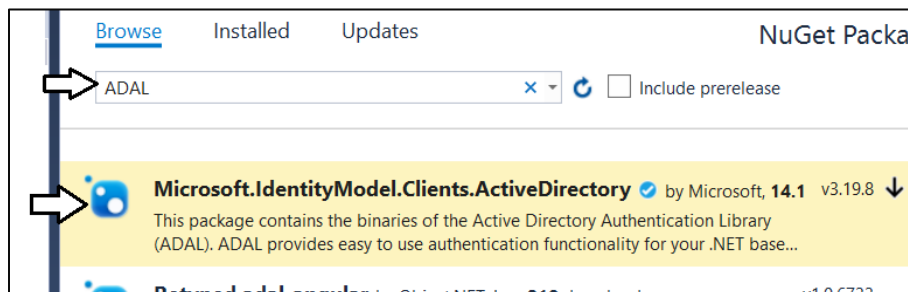
   d) Select and install **Microsoft.IdentityModel.Clients.ActiveDirectory**.
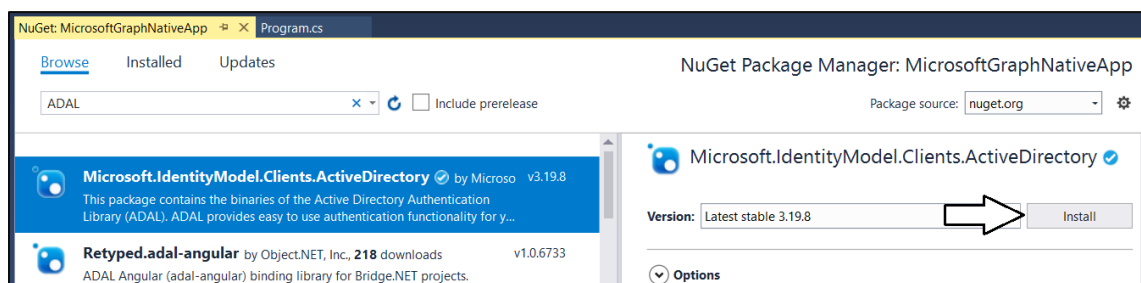
   e) When prompted about the licensing agreement, click **I Agree**.

MSD365: Modern SharePoint and Office 365 Development
Module 07 Lab: Programming Authentication with Azure Active Directory

Live Lab Version: Sep 12, 2018

3. Add a second NuGet package for **Newtonsoft.Json**.

    a) Click the **Browse** tab and type **Newtonsoft** into the search box.

    b) Locate and install the package **Newtonsoft.Json**.
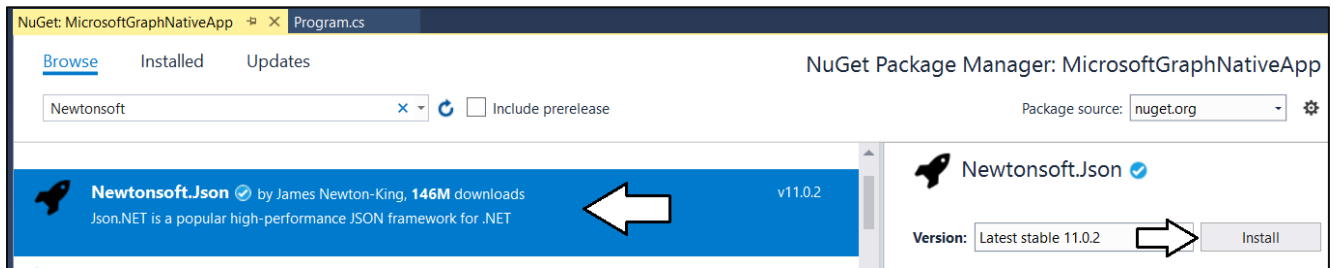
    c) You should now have the following two NuGet packages installed.
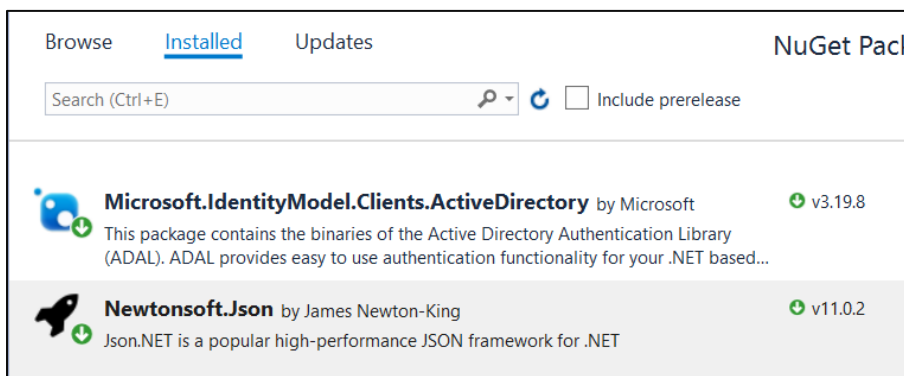
    d) Close the window for the Nuget Package Manager.

4. Add the **TokenManager** class to the **MicrosoftGraphNativeApp** project

    a) In the root folder of the project, create a new folder named **Models**.

    b) Add a C# source file into the **Models** folder named **TokenManager.cs**.

MSD365: Modern SharePoint and Office 365 Development
Module 07 Lab: Programming Authentication with Azure Active Directory

Live Lab Version: Sep 12, 2018
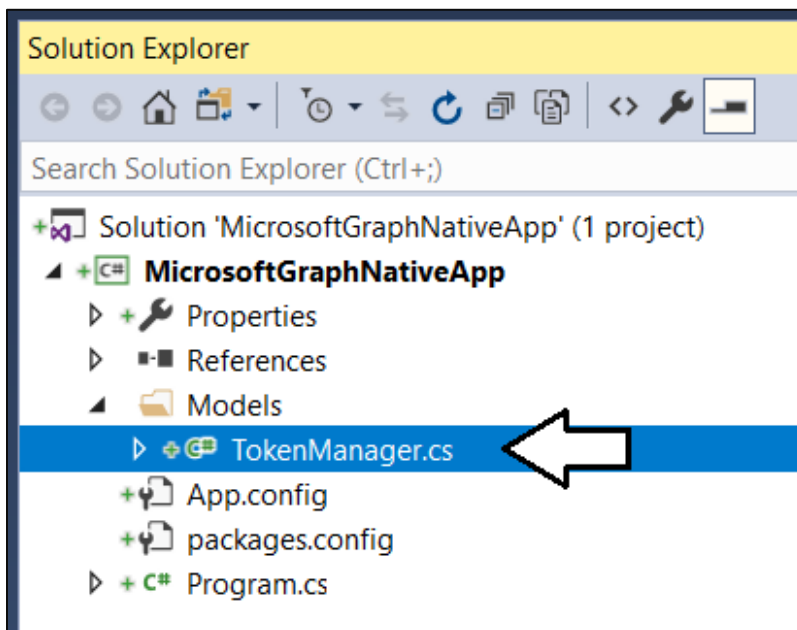
c) Add the following code to **TokenManager.cs**.

```csharp
using System;
using Microsoft.IdentityModel.Clients.ActiveDirectory;

namespace MicrosoftGraphNativeApp.Models {

  class TokenManager {

    const string authority = "https://login.microsoftonline.com/common";
    const string resourceMicrosoftGraphAPI = "https://graph.microsoft.com";

    const string clientId = "ADD_YOUR_APPLICATION_ID_HERE";
    const string replyUrl = "https://localhost/app1234";
    readonly static Uri replyUri = new Uri(replyUrl);

    public static string GetAccessToken() {

      // create authentication context
      AuthenticationContext authContext = new AuthenticationContext(authority);

      // obtain access token
      var authResult =
        authContext.AcquireTokenAsync(
            resourceMicrosoftGraphAPI,
            clientId,
            replyUri,
            new PlatformParameters(PromptBehavior.Auto));

      // return access token to caller
      return authResult.Result.AccessToken;
    }
  }
}
```

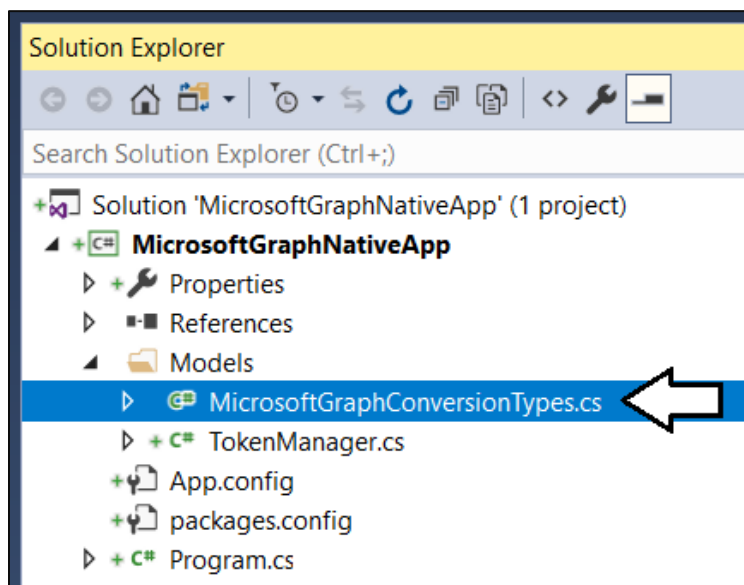d) Locate the **const** named **clientId** and replace **ADD_YOUR_APPLICATION_ID_HERE** with the GUID for your application ID.

```csharp
const string clientId = "0bfa62a6-44b2-4025-94f2-91e97f4b7986";
```

e) Save your changes ad close **TokenManager.cs**.

5. Add C# types to assist with JSON to C# conversion.

   a) Into the **Models** folder, add a new C# source file named **MicrosoftGraphConversionTypes.cs**.

MSD365: Modern SharePoint and Office 365 Development
Module 07 Lab: Programming Authentication with Azure Active Directory

Live Lab Version: Sep 12, 2018

b) Add the following code to **MicrosoftGraphConversionTypes.cs**.

```csharp
using System.Collections.Generic;

namespace MicrosoftGraphNativeApp.Models {

  public class Office365User {
    public string id { get; set; }
    public string displayName { get; set; }
    public string givenName { get; set; }
    public object jobTitle { get; set; }
    public string mail { get; set; }
    public object mobilePhone { get; set; }
    public object officeLocation { get; set; }
    public string preferredLanguage { get; set; }
    public string surname { get; set; }
    public string userPrincipalName { get; set; }
  }

  public class Office365OrganizationCollection {
    public List<Office365Organization> value { get; set; }
  }

  public class Office365Organization {
    public string id { get; set; }
    public List<Office365AssignedPlan> assignedPlans { get; set; }
    public List<object> businessPhones { get; set; }
    public object city { get; set; }
    public object country { get; set; }
    public string countryLetterCode { get; set; }
    public string displayName { get; set; }
    public List<object> marketingNotificationEmails { get; set; }
    public object onPremisesLastSyncDateTime { get; set; }
    public object onPremisesSyncEnabled { get; set; }
    public object postalCode { get; set; }
    public string preferredLanguage { get; set; }
    public List<Office365ProvisionedPlan> provisionedPlans { get; set; }
    public List<object> securityComplianceNotificationMails { get; set; }
    public List<object> securityComplianceNotificationPhones { get; set; }
    public object state { get; set; }
    public object street { get; set; }
    public List<string> technicalNotificationMails { get; set; }
    public List<Office365VerifiedDomain> verifiedDomains { get; set; }
  }

  public class Office365AssignedPlan {
    public string assignedDateTime { get; set; }
    public string capabilityStatus { get; set; }
    public string service { get; set; }
    public string servicePlanId { get; set; }
  }

  public class Office365ProvisionedPlan {
    public string capabilityStatus { get; set; }
    public string provisioningStatus { get; set; }
    public string service { get; set; }
  }

  public class Office365VerifiedDomain {
    public string capabilities { get; set; }
    public bool isDefault { get; set; }
    public bool isInitial { get; set; }
    public string name { get; set; }
    public string type { get; set; }
  }

}
```

c) Save your changes ad close **MicrosoftGraphConversionTypes.cs**.

MSD365: Modern SharePoint and Office 365 Development
Module 07 Lab: Programming Authentication with Azure Active Directory

Live Lab Version: Sep 12, 2018

6. Add code to **program.cs**.

a) Open **program.cs** in an editor window.

b) Delete all the existing code in **program.cs** and replace it with the following code.

```
using System;
using System.Linq;
using System.Net;
using System.Net.Http;
using Newtonsoft.Json;
using MicrosoftGraphNativeApp.Models;

namespace MicrosoftGraphNativeApp {

  class Program {

    static void Main() {

      // call to Microsoft Graph using REST API
      DisplayCurrentUserInfo_REST();
      DisplayOrganizationInfo_REST();

      Console.WriteLine();
      Console.WriteLine("Press ENTER to end program");
      Console.ReadLine();
    }

    // URLs used when making direct REST calls to Microsoft Graph API
    const string rootMicrosoftGraphAPI = "https://graph.microsoft.com/v1.0/";
    const string restUrlCurrentUser = "https://graph.microsoft.com/v1.0/me/";
    const string restUrlCurrentOrganization = "https://graph.microsoft.com/v1.0/organization/";

    static void DisplayCurrentUserInfo_REST() {
    }

    static void DisplayOrganizationInfo_REST() {
    }

  }
}
```

c) Replace the **DisplayCurrentUserInfo_REST** method with the following implementation.

```
static void DisplayCurrentUserInfo_REST() {

  HttpClient client = new HttpClient();
  HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, restUrlCurrentUser);
  request.Headers.Add("Authorization", "Bearer " + TokenManager.GetAccessToken());
  request.Headers.Add("Accept", "application/json");

  HttpResponseMessage response = client.SendAsync(request).Result;

  if (response.StatusCode != HttpStatusCode.OK) {
    throw new ApplicationException("Error!!!!!");
  }

  string jsonResult = response.Content.ReadAsStringAsync().Result;
  Office365User user = JsonConvert.DeserializeObject<Office365User>(jsonResult);

  Console.WriteLine("Current user info obtained with REST API");
  Console.WriteLine("-----------------------------------------");
  Console.WriteLine("ID: " + user.id);
  Console.WriteLine("User Principal Name: " + user.userPrincipalName);
  Console.WriteLine("Display Name: " + user.displayName);
  Console.WriteLine("First Name: " + user.givenName);
  Console.WriteLine("Last Name: " + user.surname);
  Console.WriteLine("Mail: " + user.mail);
  Console.WriteLine();
  Console.WriteLine();
}
```
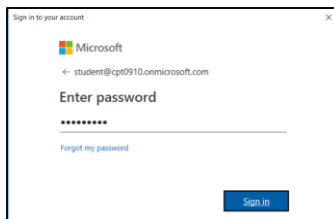
MSD365: Modern SharePoint and Office 365 Development
Module 07 Lab: Programming Authentication with Azure Active Directory
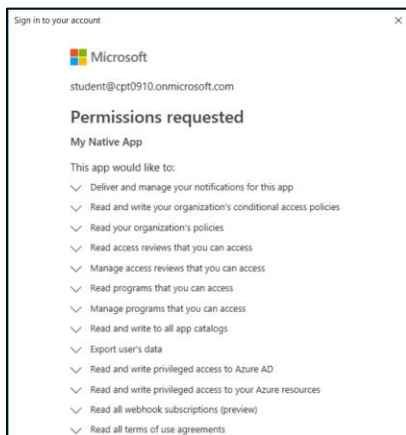
Live Lab Version: Sep 12, 2018

7. Replace the **DisplayOrganizationInfo_REST** method with the following implementation.

```
static void DisplayOrganizationInfo_REST() {

  HttpClient client = new HttpClient();
  HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, restUrlCurrentOrganization);
  request.Headers.Add("Authorization", "Bearer " + TokenManager.GetAccessToken());
  request.Headers.Add("Accept", "application/json");

  HttpResponseMessage response = client.SendAsync(request).Result;

  if (response.StatusCode != HttpStatusCode.OK) {
    throw new ApplicationException("Error!!!!!");
  }

  string jsonResult = response.Content.ReadAsStringAsync().Result;
  Office365OrganizationCollection orgs =
      JsonConvert.DeserializeObject<Office365OrganizationCollection>(jsonResult);
  Office365Organization org = orgs.value.FirstOrDefault<Office365Organization>();

  Console.WriteLine("Organization info obtained with REST API");
  Console.WriteLine("-------------------------------------------");
  Console.WriteLine("ID: " + org.id);
  Console.WriteLine("Display Name: " + org.displayName);
  Console.WriteLine("Tenant Domain: " +
                    org.verifiedDomains.FirstOrDefault<Office365VerifiedDomain>().name);
  Console.WriteLine("Country Letter Code: " + org.countryLetterCode);
  Console.WriteLine("Technical Email: " + org.technicalNotificationMails.FirstOrDefault<string>());
  Console.WriteLine();
  Console.WriteLine();
}
```
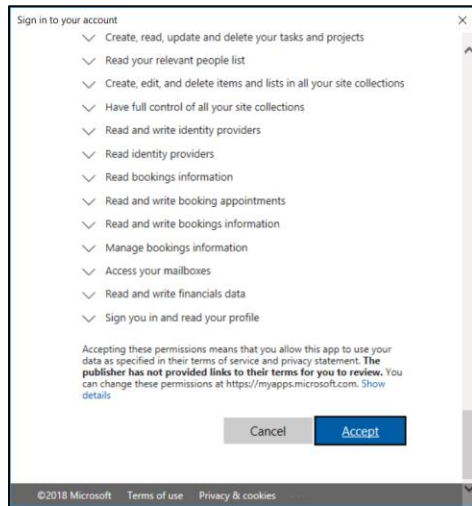
   a) Save your changes to **program.cs**.

8. Run the application to call to the Microsoft Graph API.

   a) Press the {F5} key to begin a debugging session.

   b) When promoted to sign in, log in using your Office 365 account and credentials.

   c) When prompted with the **Permissions requested** dialog, inspect all the properties that are listed.

MSD365: Modern SharePoint and Office 365 Development
Module 07 Lab: Programming Authentication with Azure Active Directory

Live Lab Version: Sep 12, 2018

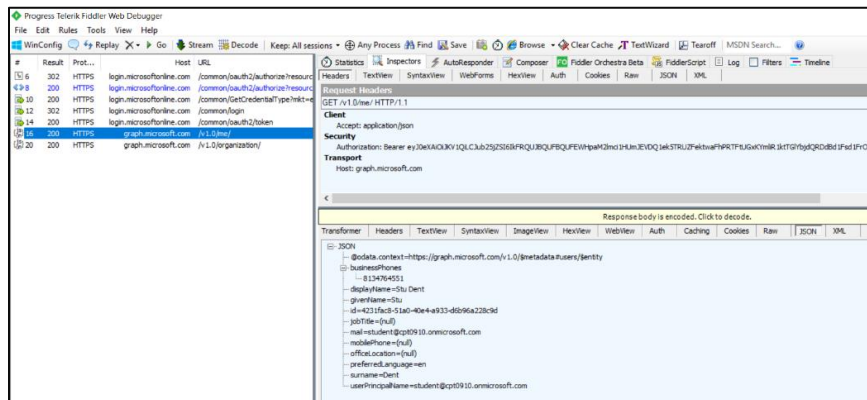d) Scroll to the bottom of the permission list in the dialog and click **Accept**.

e) The application should now continue to run.

f) The application should call into the Microsoft Graph API and retrieve data which id displayed to the console window.

9. Run the program again and monitor it with Fiddler

   a) Open Fiddler and run the program a second time.

   b) Monitor the calls into Azure AD and into the Microsoft Graph API

Congratulations. You have now successfully called into the Microsoft Graph API.

MSD365: Modern SharePoint and Office 365 Development
Module 07 Lab: Programming Authentication with Azure Active Directory

Live Lab Version: Sep 12, 2018

Exercise 3: Calling to the Microsoft Graph using the .NET SDK

In this exercise, you will continue working with the C# Console application to call into the Microsoft Graph API using the SDK.

1. Make sure you still have the **MicrosoftGraphNativeApp** project open in Visual Studio 2017.

2. Add a new NuGet package for the Microsoft Graph .NET SDK.

   a) Right-click the top-level node for the **MicrosoftGraphNativeApp** project and select **Manage NuGet Packages…**.

   b) Click the **Browse** tab and type **Microsoft.Graph** into the search box.

   c) Locate and install the package **Microsoft.Graph** which is the Microsoft Graph .NET SDK.
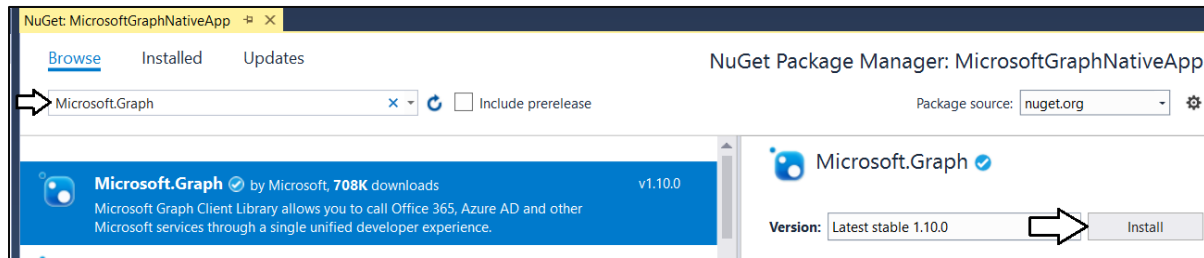
   d) After installing the **Microsoft.Graph** NuGet package inspect the list of installed packages.
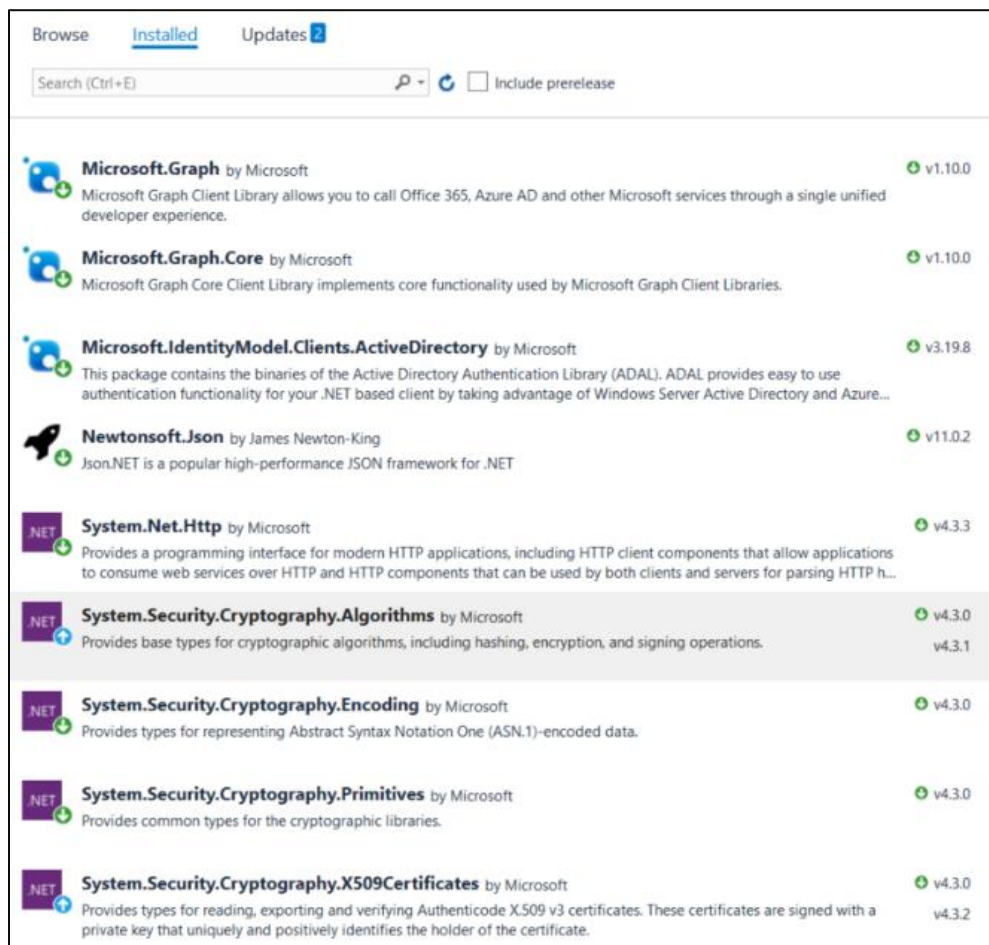
When you add the NuGet package from **Microsoft.Graph**, there are several dependent packages that are added as well.

MSD365: Modern SharePoint and Office 365 Development
Module 07 Lab: Programming Authentication with Azure Active Directory

Live Lab Version: Sep 12, 2018

3. Add the **MyAuthProvider** class.

   a) Into the **Models** folder, add a new C# source file named **MyAuthProvider.cs**.
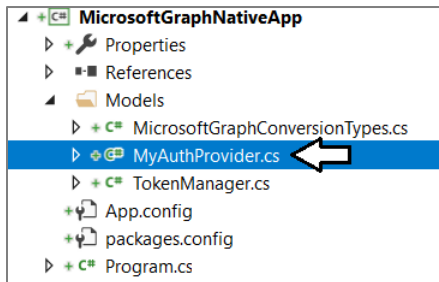
   b) Add the following code to **MyAuthProvider.cs**.

```csharp
using System;
using System.Net.Http;
using System.Threading.Tasks;
using Microsoft.IdentityModel.Clients.ActiveDirectory;
using Microsoft.Graph;

namespace MicrosoftGraphNativeApp.Models {

  class MyAuthProvider : IAuthenticationProvider {

    const string authority = "https://login.microsoftonline.com/common";
    const string resourceMicrosoftGraphAPI = "https://graph.microsoft.com";

    const string clientId = "ADD_YOUR_APPLICATION_ID_HERE";
    const string replyUrl = "https://localhost/app1234";
    readonly static Uri replyUri = new Uri(replyUrl);

    public async Task AuthenticateRequestAsync(HttpRequestMessage request) {

      // Use ADAL to obtain access token - ADAL performs tokn caching behind the scenes
      AuthenticationContext authContext = new AuthenticationContext(authority);
      AuthenticationResult authResult =
        await authContext.AcquireTokenAsync(
              resourceMicrosoftGraphAPI,
              clientId,
              replyUri,
              new PlatformParameters(PromptBehavior.Auto));

      // Insert access token into authorization header for each outbound request
      request.Headers.Add("Authorization", "Bearer " + authResult.AccessToken);
    }
  }

}
```

   c) Locate the **const** named **clientId** and replace **ADD_YOUR_APPLICATION_ID_HERE** with the GUID for your application ID.

```csharp
const string clientId = "0bfa62a6-44b2-4025-94f2-91e97f4b7986";
```

   d) Save your changes ad close **MyAuthProvider.cs**.

4. Update **program.cs** to call the Microsoft Graph API using the SDK.

   a) Inside **program.cs** underneath the existing **using** statements, add a new **using** statement for **Microsoft.Graph**.

```csharp
using System;
using System.Linq;
using System.Net;
using System.Net.Http;
using Newtonsoft.Json;
using MicrosoftGraphNativeApp.Models;
using Microsoft.Graph;
```

b) Place you cursor at the bottom of the **Program** class to make some extra space to add more code..

```
namespace MicrosoftGraphNativeClient {

  class Program {
    static void Main() ...

    // URLs used when making direct REST calls to Microsoft Graph API
    const string rootMicrosoftGraphAPI = "https://graph.microsoft.com/v1.0/";
    const string restUrlCurrentUser = "https://graph.microsoft.com/v1.0/me/";
    const string restUrlCurrentOrganization = "https://graph.microsoft.com/v1.0/organiz

    static void DisplayCurrentUserInfo_REST() ...

    static void DisplayOrganizationInfo_REST() ...



  }
}
```

c) Paste the following code into the bottom of the **Program** class where you create the extra pace.

```
// primary entry point object for .NET client library for the Microsoft Graph API
static GraphServiceClient graphServiceClient = new GraphServiceClient(new MyAuthProvider());

static void DisplayCurrentUserInfo_DotNet() {
}

static void DisplayOrganizationInfo_DotNet() {
}
```

d) Replace the **DisplayCurrentUserInfo_DotNet** method with the following code.

```
static void DisplayCurrentUserInfo_DotNet() {

    // call across Internet and wait for response
    var user = graphServiceClient.Me.Request().GetAsync().Result;

    Console.WriteLine("Current user info obtained with .NET Client");
    Console.WriteLine("------------------------------------------");
    Console.WriteLine("Display Name: " + user.DisplayName);
    Console.WriteLine("First Name: " + user.GivenName);
    Console.WriteLine("Last Name: " + user.Surname);
    Console.WriteLine("User Principal Name: " + user.UserPrincipalName);
    Console.WriteLine();
    Console.WriteLine();
}
```

e) Replace the **DisplayOrganizationInfo_DotNet** method with the following code.

```
static void DisplayOrganizationInfo_DotNet() {

  // call across Internet and wait for response
  var org = graphServiceClient
            .Organization
            .Request()
            .GetAsync()
            .Result.FirstOrDefault<Organization>();

  Console.WriteLine("Current organizational info obtained with .NET Client");
  Console.WriteLine("----------------------------------------------");
  Console.WriteLine("ID: " + org.Id);
  Console.WriteLine("Display Name: " + org.DisplayName);
  Console.WriteLine("Tenant Domain: " + org.VerifiedDomains.FirstOrDefault<VerifiedDomain>().Name);
  Console.WriteLine("Country Letter Code: " + org.CountryLetterCode);
  Console.WriteLine("Technical Email: " + org.TechnicalNotificationMails.FirstOrDefault());
  Console.WriteLine();
  Console.WriteLine();
}
```

f) Add the two lines in the **Main** method to call **DisplayCurrentUserInfo_DotNet** and **DisplayOrganizationInfo_DotNet**.
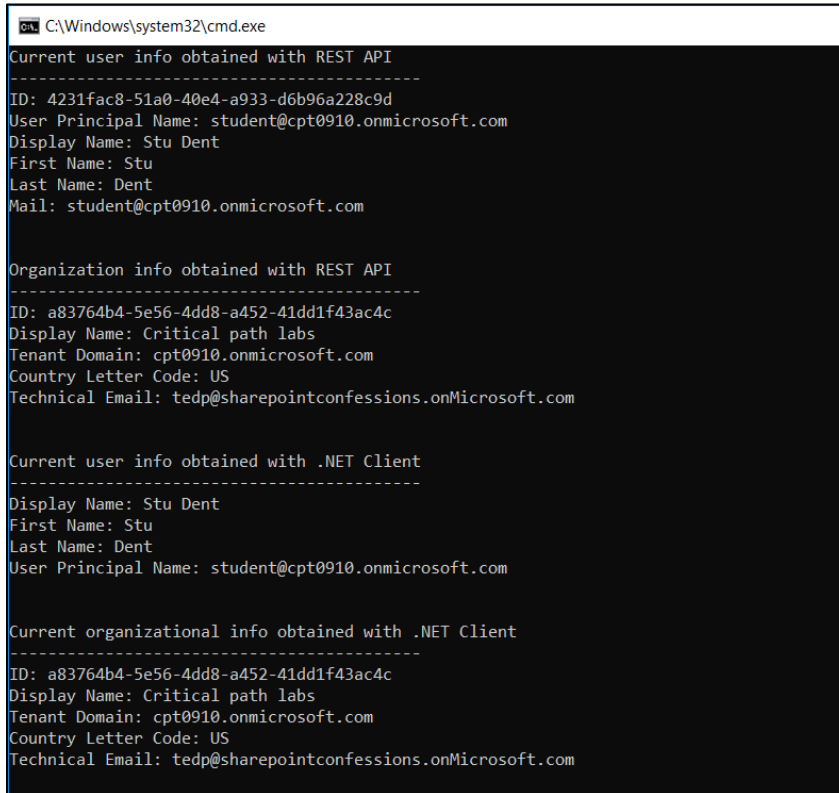
```
static void Main() {

    // call to Microsoft Graph using REST API
    DisplayCurrentUserInfo_REST();
    DisplayOrganizationInfo_REST();

    // call to Microsoft Graph using .NET client
    DisplayCurrentUserInfo_DotNet();
    DisplayOrganizationInfo_DotNet();

    Console.WriteLine();
    Console.WriteLine("Press ENTER to end program");
    Console.ReadLine();

}
```

5. Test your work by running the application.

a) Press the {F5} key to begin a debugging session.

b) When promoted to sign in, log in using your Office 365 account and credentials.

c) You should see that the Microsoft Graph API calls made with the .NET SDK are executing successfully.

You have now reached the end of this lab.