Developing with the Power BI Service API

Setup Time: 60 minutes

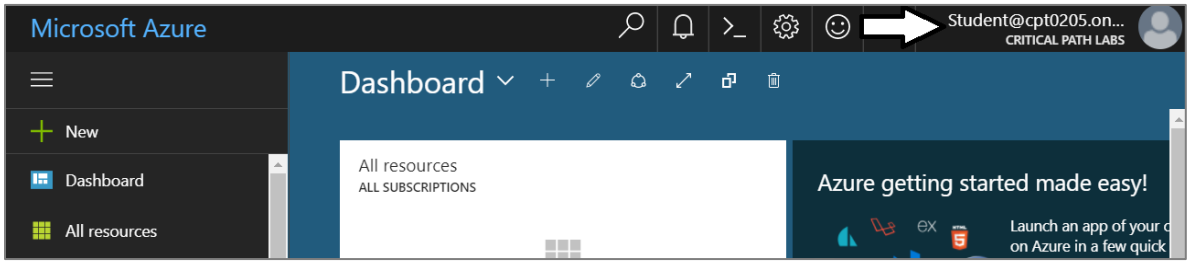
Lab Folder: C:\Student\Modules\07_PowerBiServiceApi\Lab

Overview: In this lab, you will log into an Azure AD user account that has been created for you in an Azure AD tenant shared by all students which has a domain name of **powerbimvps.onmicrosoft.com**. Once you have logged into the Power BI service and started your 60 Power BI Pro trial, you will be able to upload PBIX files into your personal workspace and you can begin designing dashboard and reports in the browser. The lab will also step you through downloading and installing Power BI Desktop as well as publishing a Power BI Desktop project to the Power BI service. You create a new app workspace and populate it with a dataset, a report and a dashboard. In the final exercises, you will program against the Power BI Service API.

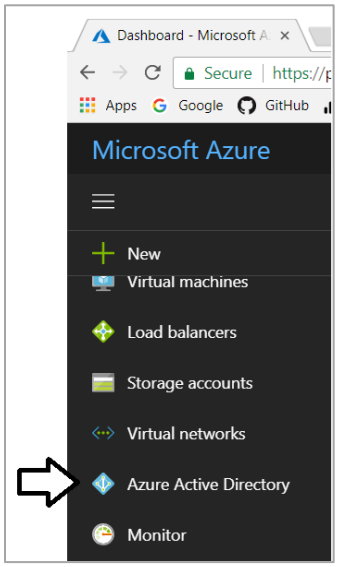
Exercise 1: Register a Native Client Application using the Azure Portal

In this exercise, you will register a new native client application with Azure AD and you will configure the application's required permissions to access the Power BI Service API.

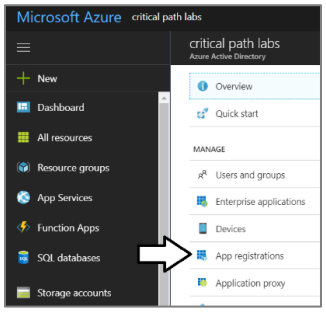
1. Log into the Azure Portal
   a) In the browser, navigate to the Azure portal at https://portal.azure.com.
   b) When you are prompted to log in, provide the credentials to log in with your Office 365 user account name.
   c) If you are prompted to start a tour of Microsoft Azure, click **Maybe later**.
   d) Once you are logged 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.



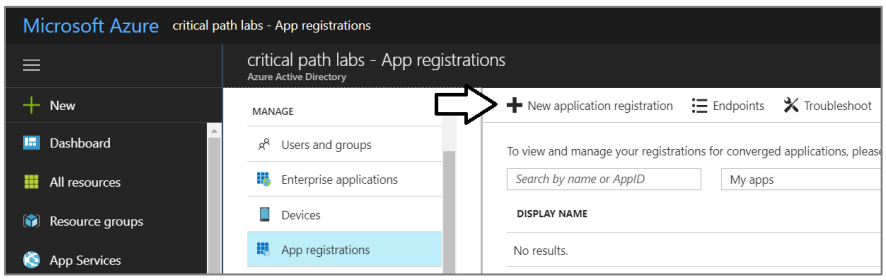
2. Register a new Azure AD application.
   a) In the left navigation, scroll down and click on the link for **Azure Active Directory**.



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



c) Click **New application registration**.



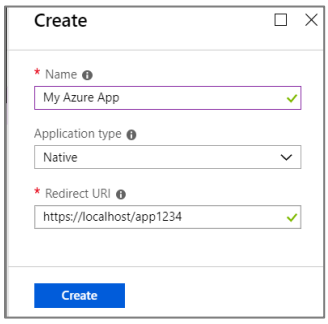
d) In the **Create** blade, enter the following information.

i) Add a **Name** of **My Azure App**.

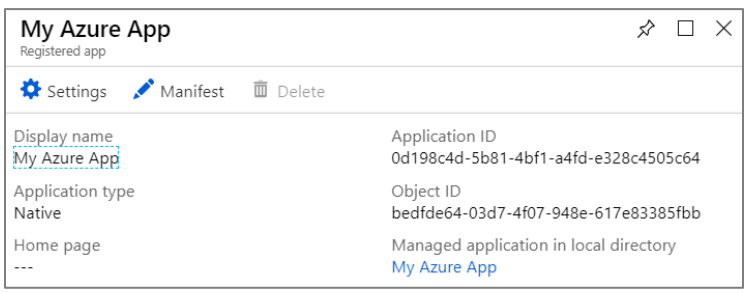
ii) Set the **Application type** to **Native**.

iii) Set the **Redirect URI** to https://localhost/app1234.

iv) Click the **Create** button to create the new application.

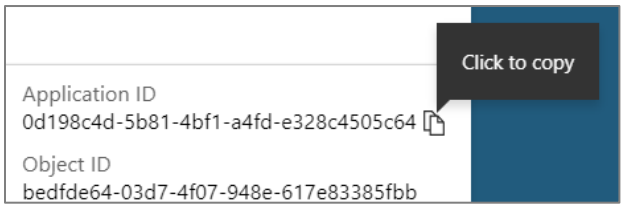


e) You should now see the summary page for the new Azure AD application.

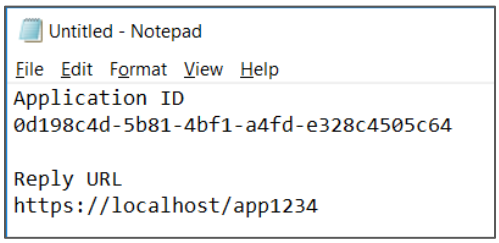


Azure AD will always generate a new GUID for the application ID any time your create a new Azure AD application.

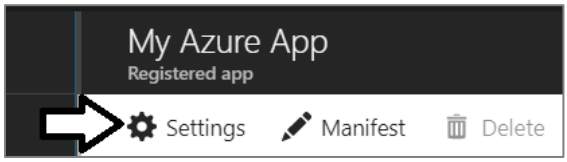
3. Copy the GUID for the Application ID.
   a) Copy the Application ID to the Windows clipboard.



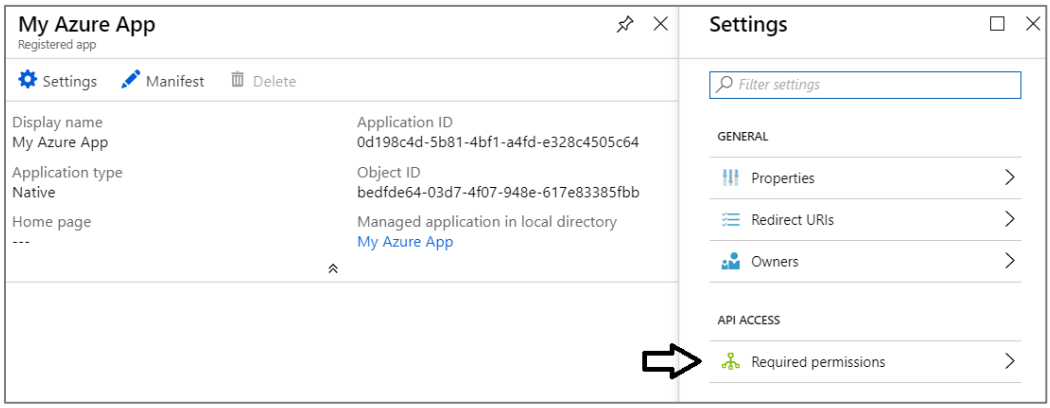
   b) Launch Notepad and paste the Application ID into a new document. Also add the value of the Redirect URI.



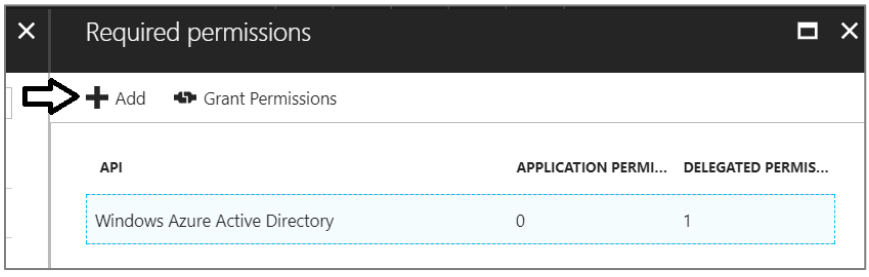
   c) Click on the **Settings** link to configure application settings,



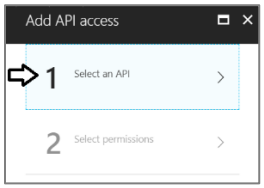
   d) In the **Settings** blade, click **Required permissions**.



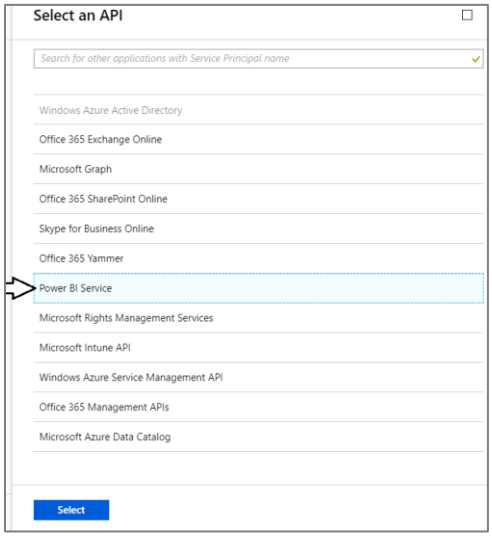
   e) Click the **Add** button on the **Required permissions** blade.



f) Click the **Select an API** option in the **Add API** access blade.

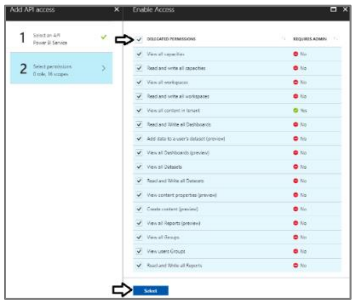


g) In the **Select an API** blade, click **Power BI Service**.



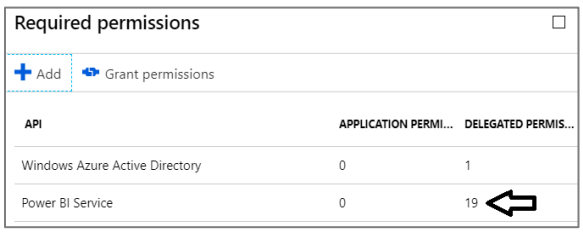
h) In the **Enable Access** blade, click the top checkbox for **DELEGATED PERMISSIONS** to select all the permissions.

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



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

k) At this point, you should be able to verify that the Power BI Service has been added to the **Required permissions** list.

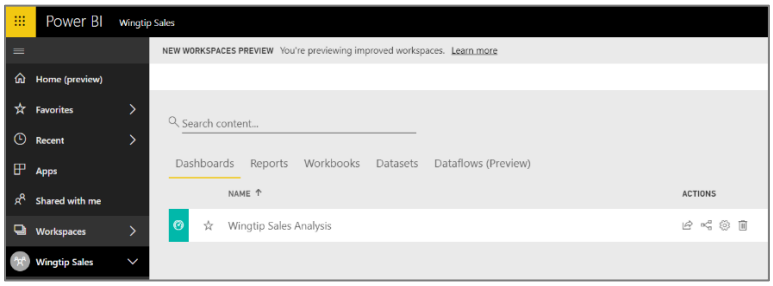


You are now done registering your application with Azure AD.

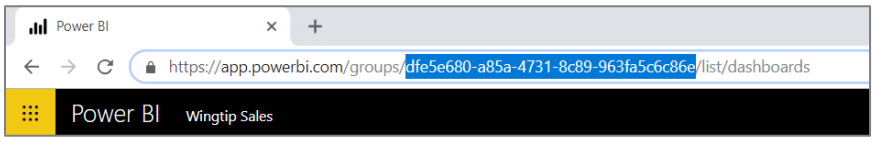
Exercise 2: Call the Power BI Service API from a C# Console Application

In this exercise, you will create a simple C# Console application to call into the Power BI Service API.

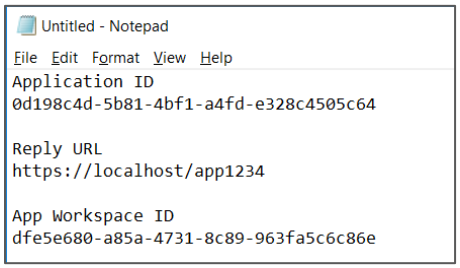
1. Determine the app workspace ID for the **Wingtip Sales** app workspace you created in earlier lab exercise.
   a) Return to the browser and navigate back to the **Wingtip Sales** app workspace in the Power BI portal.



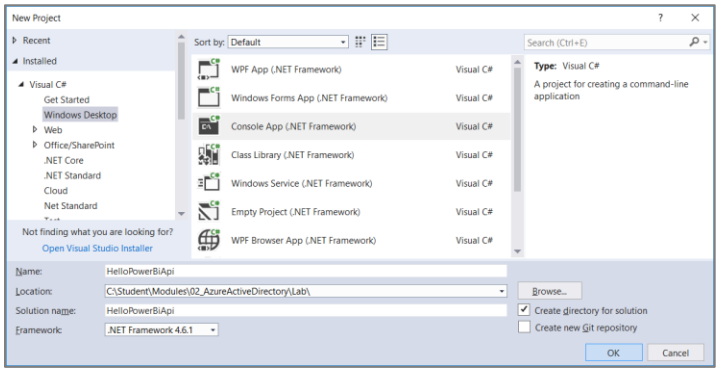
   b) In the browser address bar, select and copy the app workspace ID that appears just after **groups/** in the URL



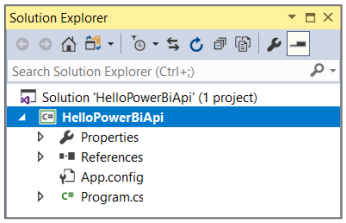
   c) Copy the app workspace ID to the text file you created in notepad as shown in the following screenshot.



2. Create a new C# Console application in Visual Studio.
   a) Launch Visual Studio.
   b) Create a new project by running the **File > New Project** command.
   c) Navigate to **Installed > Windows Desktop > Visual C#** project templates and select a project type of **Console App**.
   d) Set the **Location** to **C:\Student\Modules\02_AzureActiveDirectory\Lab**.
   e) Give the project a name of **HelloPowerBiApi** and click OK.

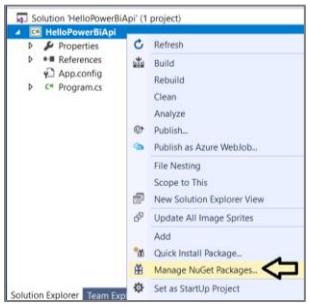


f) You should now have a new project named **HelloPowerBiApi**.



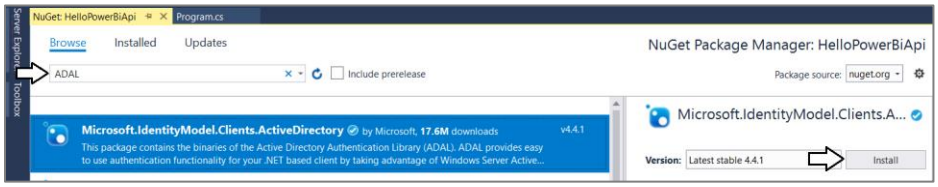
3. Install the NuGet packages for the Azure Active Directory Authentication Library and the Newtonsoft JSON converter.

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



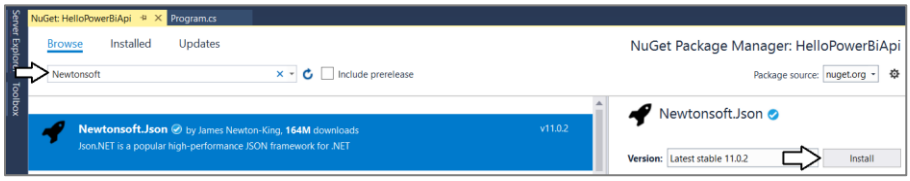
b) Click the **Browse** tab and type **ADAL** into the search box.

c) Select and install the Active Directly Authentication library package named **Microsoft.IdentityModel.Clients.ActiveDirectory**.

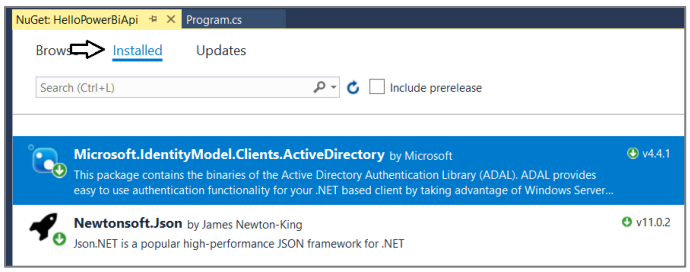


d) When prompted about the licensing agreement, click **I Agree**.

e) Search for **Newtonsoft** and then select and install the **Newtonsoft.Json** package.



f) At this point, you should have two packages installed in your project as shown in the following screenshot.



g) Close the window for the Nuget Package Manager.

4. Add starter code to the project by copying and pasting from a pre-provided text file.
   a) Using Windows Explorer, locate the file named **PowerBiStarter.cs.txt** located in the **Student** folder at the following path.

   ```
   C:\Student\Modules\02_AzureActiveDirectory\Lab\PowerBiStarter.cs.txt
   ```

   b) Open the file named **PowerBiStarter.cs.txt** in Notepad and copy its contents into the Window clipboard.
   c) Return to the **HelloPowerBiApi** project in Visual Studio.
   d) Open the source file named **program.cs**.
   e) Delete all the code inside **program.cs** and replace it with the content you copied into the Windows clipboard.
   f) You should now have the basic code for a simple C# console application which access the Power BI Service API.

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Net.Http;
using Newtonsoft.Json;
using Microsoft.IdentityModel.Clients.ActiveDirectory;

namespace HelloPowerBiServiceApi {

  class Program {

    const string aadAuthorizationEndpoint = "https://login.windows.net/common";
    const string resourceUriPowerBi = "https://analysis.windows.net/powerbi/api";

    static readonly Uri redirectUri = new Uri("https://localhost/app1234");

    const string clientId = "PASTE_YOUR_AZURE_APPLICATION_ID_HERE";
    const string appWorkspaceId = "PASTE_YOUR_POWER_BI_APP_WORKSPACE_ID_HERE";

    static string GetAccessToken() ...

    static string GetAccessTokenUnattended() ...

    static string ExecuteGetRequest(string restUrl) ...

    static void Main() ...

  }

  public class Report ...

  public class ReportCollection ...

}
```

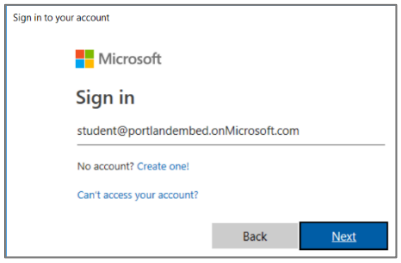
5. Update the code to include your Azure AD application ID and your Power BI app workspace ID.
   a) Locate the section of the code with the static properties named **clientId** and **appWorkspaceId**.

```
const string clientId = "PASTE_YOUR_AZURE_APPLICATION_ID_HERE";
const string appWorkspaceId = "PASTE_YOUR_POWER_BI_APP_WORKSPACE_ID_HERE";
```

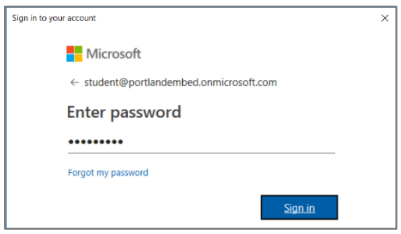
   b) Replace these values with the values you copied into Notepad earlier.

```
const string clientId = "0d198c4d-5b81-4bf1-a4fd-e328c4505c64";
const string appWorkspaceId = "dfe5e680-a85a-4731-8c89-963fa5c6c86e";
```

   c) Save your changes to **program.cs**.
6. Run the application to call to the Power BI Service API.
   a) Press the **{F5}** key to begin a debugging session.
   b) When promoted to sign in, enter the name of your Office 365 user account and click **Next**.



   c) Enter your password and click **Sign in**.

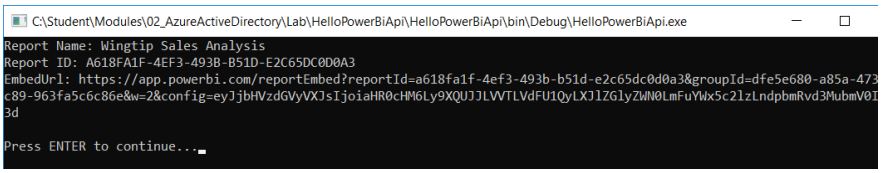


At this point, you have not consented to allow the application to call the Power BI Service API on your behalf. Therefore, Azure AD will prompt you with **the Permissions requested** dialog which will allow you to grant the application the permissions it needs to run.

d) When prompted with the **Permissions requested** dialog, review the listed permissions and then click **Accept**.

In the simple scenario in this lab, your application will not require all these permissions. However, you have now seen all the possible permissions you can request for the Power BI Service API in the **Permissions requested** dialog.

e) The application should now run and execute a request into the Power BI Service API to retrieve data the reports in the Wingtip Sales workspace. You should see output in the Console window about the **Wingtip Sales Analysis** report.



Congratulations. You have now successfully called the Power BI Service API.

7. Modify the C# console application to acquire an access token using the User Password Credential flow.
   a) Inside the **Program** class in **Program.cs**, locate the method named **GetAccessTokenUnattended**.

```
static string GetAccessTokenUnattended() {
  string userName = "REPLACE_WITH_MASTER_USER_ACCOUNT_NAME";
  string userPassword = "REPLACE_WITH_MASTER_USER_ACCOUNT_PASSWORD";
  var authContext = new AuthenticationContext(aadAuthorizationEndpoint);
  var userPasswordCredential = new UserPasswordCredential(userName, userPassword);
  AuthenticationResult result =
    authContext.AcquireTokenAsync(resourceUriPowerBi, clientId, userPasswordCredential).Result;
  return result.AccessToken;
}
```

   b) Update the variables named **username** and **userPassword** to initialize them with your Office 365 user account credentials.

```
static string GetAccessTokenUnattended() {
  string userName = "student@portlandembed.onMicrosoft.com";
  string userPassword = "pass@word1";
  var authContext = new AuthenticationContext(aadAuthorizationEndpoint);
  var userPasswordCredential = new UserPasswordCredential(userName, userPassword);
  AuthenticationResult result =
    authContext.AcquireTokenAsync(resourceUriPowerBi, clientId, userPasswordCredential).Result;
  return result.AccessToken;
}
```

   c) Inspect the method named **ExecuteGetRequest** and locate the call to **GetAccessToken**.

```
static string ExecuteGetRequest(string restUrl) {
  HttpClient client = new HttpClient();
  HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, restUrl);
  request.Headers.Add("Authorization", "Bearer " + GetAccessToken());
  request.Headers.Add("Accept", "application/json;odata.metadata=minimal");
  HttpResponseMessage response = client.SendAsync(request).Result;
  if (response.StatusCode != HttpStatusCode.OK) {
    throw new ApplicationException("Error occured calling the Power BI Servide API");
  }
  return response.Content.ReadAsStringAsync().Result;
}
```

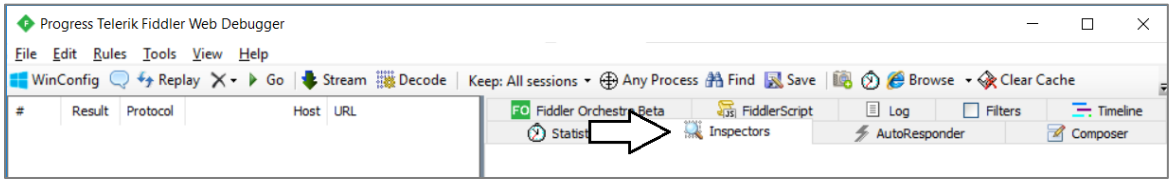
   d) Replace the call to the call to **GetAccessToken** with a call to **GetAccessTokenUnattended**.

```
static string ExecuteGetRequest(string restUrl) {
  HttpClient client = new HttpClient();
  HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, restUrl);
  request.Headers.Add("Authorization", "Bearer " + GetAccessTokenUnattended());
  request.Headers.Add("Accept", "application/json;odata.metadata=minimal");
  HttpResponseMessage response = client.SendAsync(request).Result;
  if (response.StatusCode != HttpStatusCode.OK) {
    throw new ApplicationException("Error occured calling the Power BI Servide API");
  }
  return response.Content.ReadAsStringAsync().Result;
}
```

   e) Rerun the program by pressing the **{F5}** key.

The program should run now successfully with requiring an interactive login.

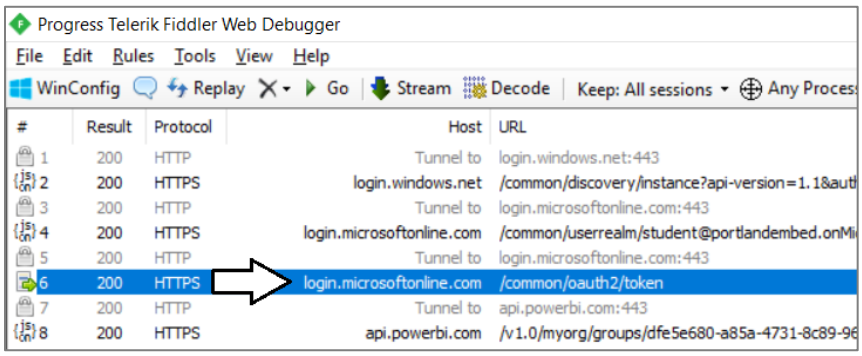
8. Inspect calls from your application to the Power BI Service API using the Fiddler utility.
   a) Launch Fiddler.
   b) Click on the **Inspectors** tab on the left so you can see details of the request and response for each HTTP request.



c) Return to Visual Studio and run the **HelloPowerBiApi** application in the Visual Studio debugger by pressing the **{F5}** key.

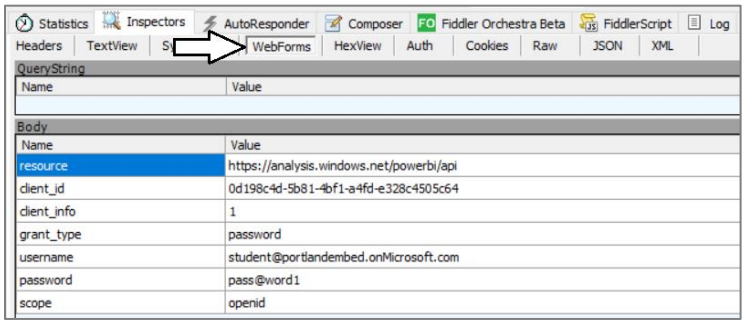
d) Once the application has run, return to Fiddler and examine the HTTP requests.

e) Look through the HTTP requests in Fiddler and locate the call to **https://login.microsoftonline.com/common/oauth2/token**.



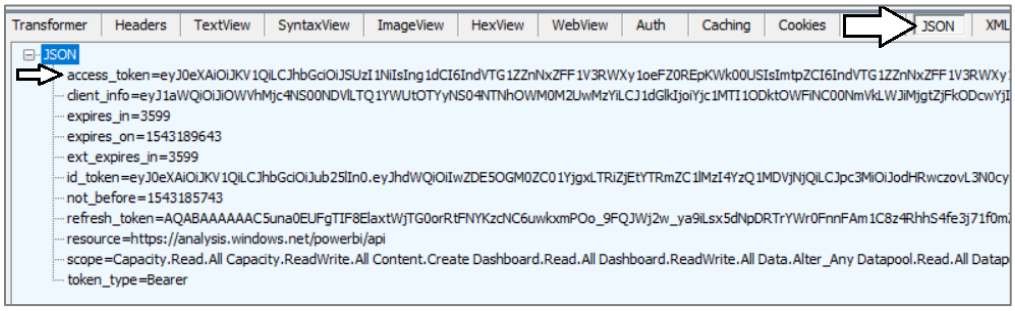
f) On the right in the request details pane, click the **WebForms** tab so you can see the form variables passed in the request.

g) You should be able to see form variables for **resource**, **client_id**, **grant_type**, **username**, **password** and **scope**.



h) On the right in the response details pane, click the **JSON** tab to see the results formated as JSON.

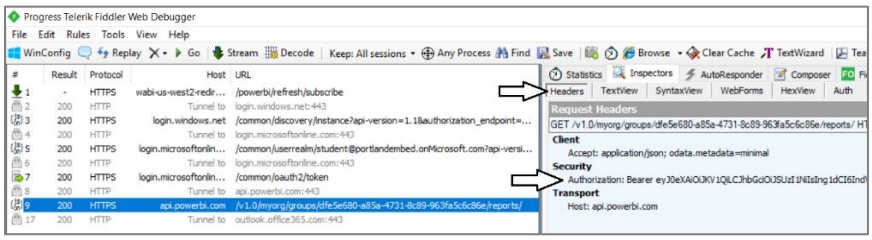
i) You should be able to see the JSON response which includes a value for the **access_token**.



j) In the request list on the left, locate the call to the Power BI Service which has a **Host** value of **app.powerbi.com**.

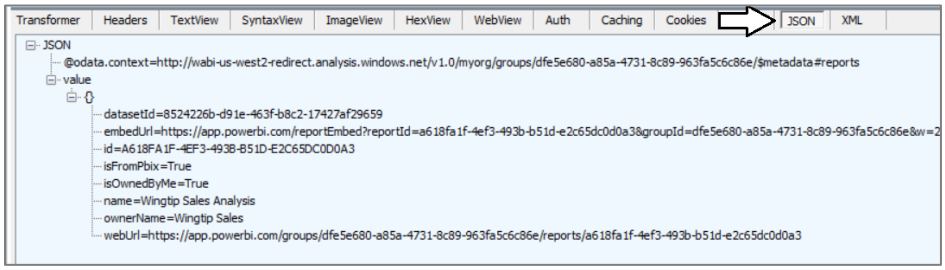
k) On the top right in the request details pane, click the **Headers** tab so you can see the request headers.

l) You should be able to see the **Authorization** header which contains the text **"Bearer "** and then the access token.



m) On the bottom right in the response details pane, click the **JSON** tab so you can see the response in a JSON format.

n) Examine the response and view the values included for each report.



If you have never used Fiddler before, it's a great tool for debugging problems with your authentication code.

9. Close the browser, return to Visual Studio and terminate the current debugging session.