

Developing with the Power BI Service API

Setup Time: 40 minutes

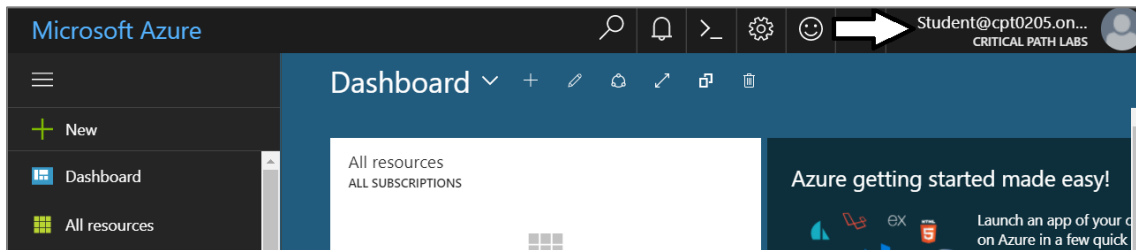
Lab Folder: C:\Student\Modules\03_PowerBIServiceAPI\Lab

Overview: In this lab, you will begin by creating a new Azure AD application that allows you to call the Power BI Service API. After that, you will use Visual Studio to create a new C# console application that programs using the Power BI SDK.

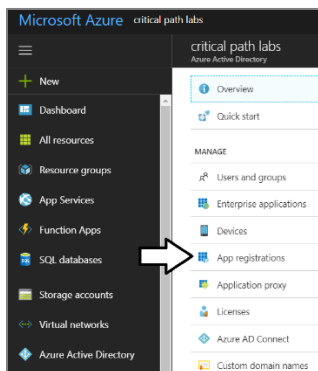
Exercise 1: Register a New Application with Azure Active Directory

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

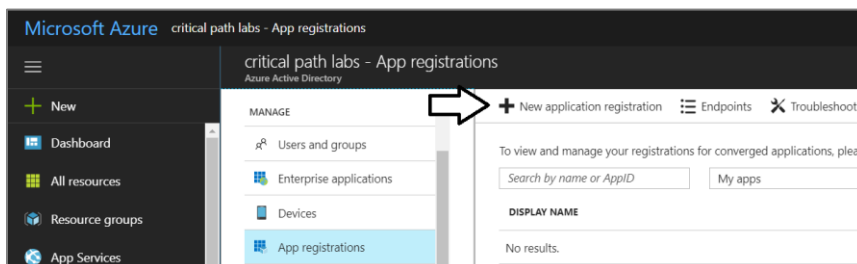
1. Log into the Azure Portal
 - a) In the browser, navigate to the Azure portal at <https://portal.azure.com>.
 - b) When you are prompted to log in, provide the credentials to log in with your Office 365 user account name.
 - c) Once you are logged into the Azure portal, check the email address in the login menu in the upper right to make sure you are logged in the Azure portal with the correct identity.



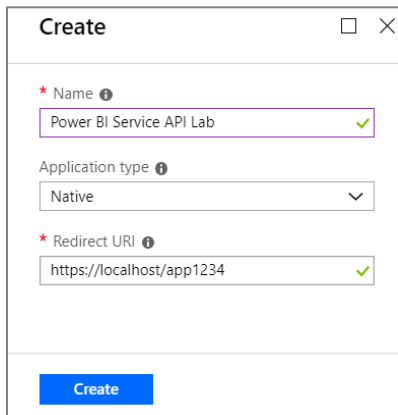
2. Register a new Azure AD application.
 - a) In the left navigation, scroll down and click on the link for **Azure Active Directory**.
 - b) Click the link for **App registration**.



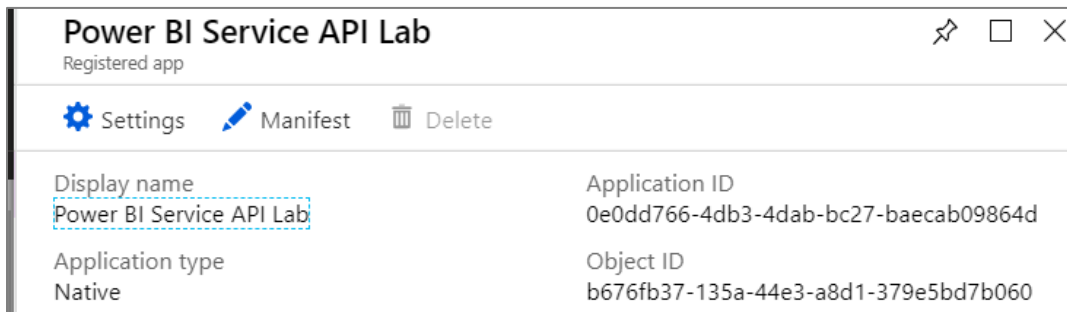
- c) Click **New application registration**.



- d) In the Create blade, enter the following information.
- Add a **Name** of **Power BI Service API Lab**.
 - Set the **Application type** to **Native**.
 - Set the **Redirect URI** to <https://localhost/app1234>.
 - Click the **Create** button to create the new application.

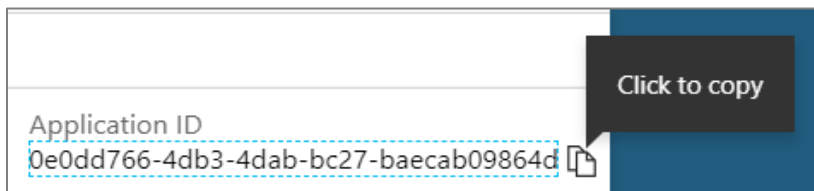


- e) Once you have created the new application you should see the new application summary view.

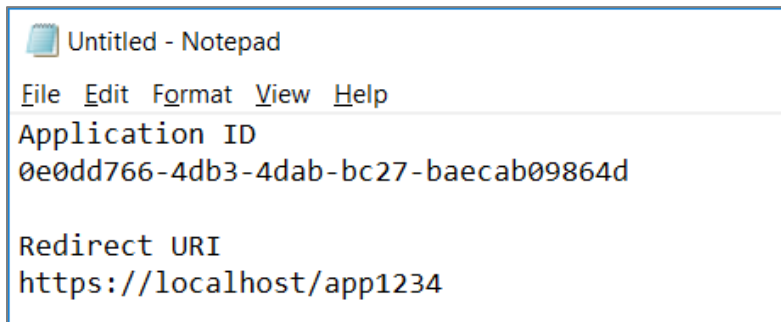


Power BI Service API Lab	
Registered app	
Settings Manifest Delete	
Display name	Application ID
Power BI Service API Lab	0e0dd766-4db3-4dab-bc27-baecab09864d
Application type	Object ID
Native	b676fb37-135a-44e3-a8d1-379e5bd7b060

- f) Copy the Application ID to the Windows clipboard.



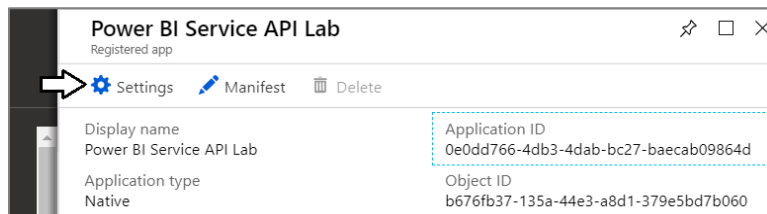
- g) Launch Notepad and paste the Application ID into a new text file. Also add the value of the Redirect URI.



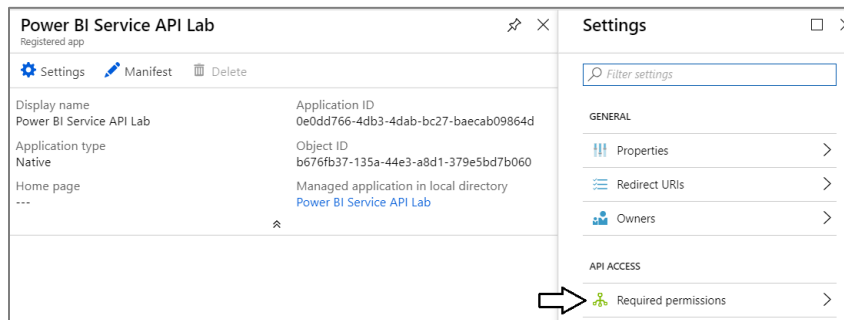
```
Untitled - Notepad
File Edit Format View Help
Application ID
0e0dd766-4db3-4dab-bc27-baecab09864d

Redirect URI
https://localhost/app1234
```

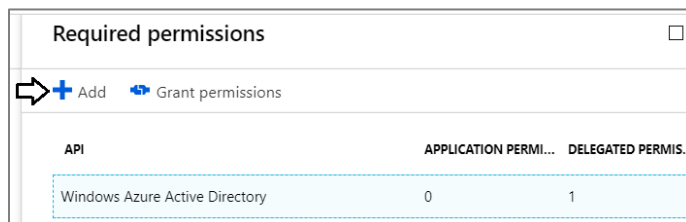
- h) Click on the **Settings** link to display the **Settings** blade.



- i) In the **Settings** blade, click **Required permissions** link.

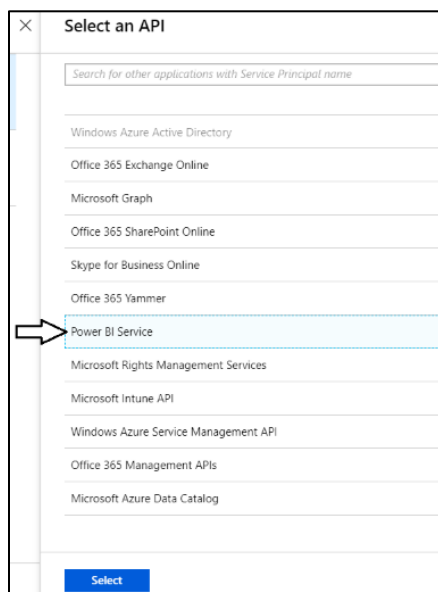


- j) In the **Required permissions** blade, click the **Add** button on.

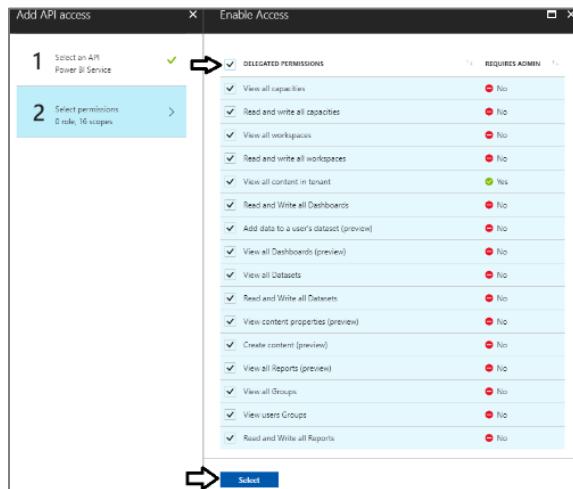


- k) Click the **Select an API** option in the **Add API** access blade.

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



- m) In the **Enable Access** blade, click the top checkbox for **DELEGATED PERMISSIONS** to select all the permissions.
- n) Once you have selected all the permissions, click the **Select** button at the bottom of the blade.



- o) Click the **Done** button at the bottom of the **Add API Access** blade.
- p) At this point, you should be able to verify that the Power BI Service has been added to the **Required permissions** list.

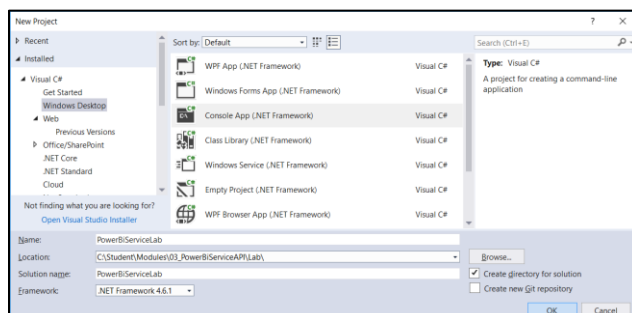
Required permissions			<input type="checkbox"/>
+ Add Grant permissions			
API	APPLICATION PERMIS...	DELEGATED PERMIS...	
Windows Azure Active Directory	0	1	
Power BI Service	0	19	

You are now done registering your application with Azure AD.

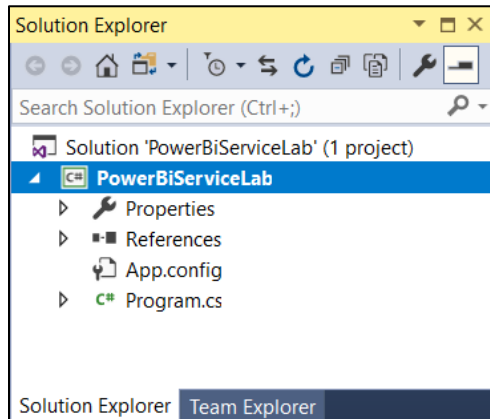
Exercise 2: Call the Power BI Service API using Direct REST Calls

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

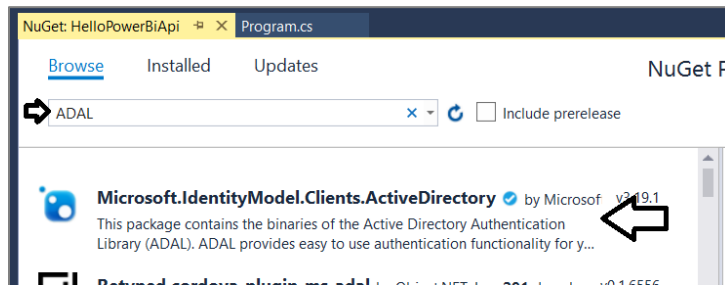
- 1. Create a new C# Console application in Visual Studio.
 - a) Launch Visual Studio.
 - b) Create a new project by running the **File > New Project** command.
 - c) Select a project type of Console App from the Visual C# project templates.
 - d) Give the project a Name of **PowerBiServiceLab** and
 - e) Give the project a Location of **C:\Student\Modules\03_PowerBiServiceAPILab**.
 - f) Click OK to create the new project.



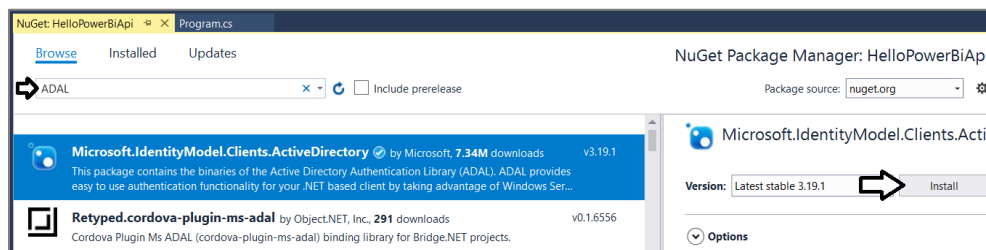
- g) You should now have a new project named **PowerBiServiceLab**.



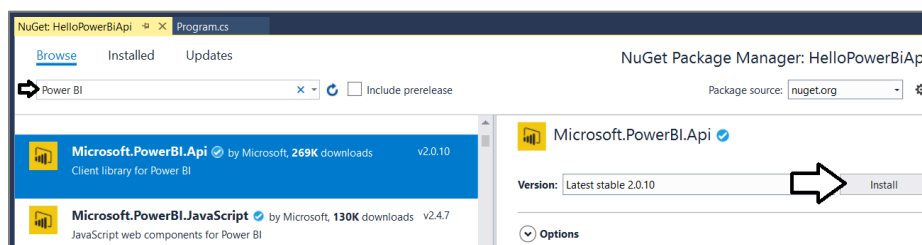
2. Add the NuGet packages to the project required to program the Power BI Service API using the Power BI SDK.
- a) Right-click the top-level node for the **PowerBiServiceLab** project and select **Manage NuGet Packages....**
 - b) Click the Browse tab and type **ADAL** into the search box.
 - c) Locate the package **Microsoft.IdentityModel.Clients.ActiveDirectory**. This is the Active Directly Authentication library.



- d) Select and install **Microsoft.IdentityModel.Clients.ActiveDirectory**.

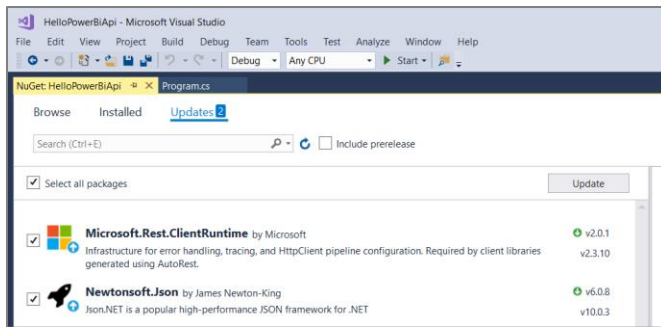


- e) When prompted about the licensing agreement, click **I Agree**.
- f) Search for Power BI and then find and install the **Microsoft.PowerBI.Api**.



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

3. Update all NuGet packages.
 - a) Navigate to the **Update** tab and update any packages that have updates available.



- b) Close the window for the NuGet Package Manager.
4. Add the starter C# code to **program.cs**.
 - a) Using Windows Explorer, locate the file named **ProgramStarter.cs.txt** located in the Student folder at the following path.

C:\Student\Modules\03_PowerBIServiceAPI\StarterFiles\ProgramStarter.cs.txt
 - b) Open the file named **ProgramStarter.cs.txt** in Notepad and copy its contents into the Window clipboard.
 - c) Return to the **PowerBIServiceLab** 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 Microsoft.IdentityModel.Clients.ActiveDirectory;
using Microsoft.PowerBI.Api.V2;
using Microsoft.Rest;

class Program {

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

    // enter the correct configuration values for your environment
    static string appworkspaceId = "";
    static string clientId = "";
    static string redirectUrl = "https://localhost/app1234";

    static string GetAccessToken() { ... }

    static PowerBIClient GetPowerBIClient() { ... }

    static void Main() { ... }

    static void DisplayPersonalWorkspaceAssets() { ... }

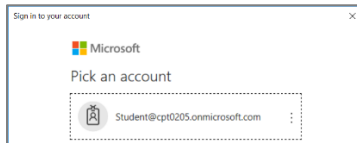
}
```

5. Update the code with your app workspace ID, the Azure AD application ID and Redirect URI.
 - a) Locate the section of the code with the static properties named **appWorkspaceId**, **clientId** and **redirectUrl**.
 - b) Replace these values with the values you copied into Notepad earlier.

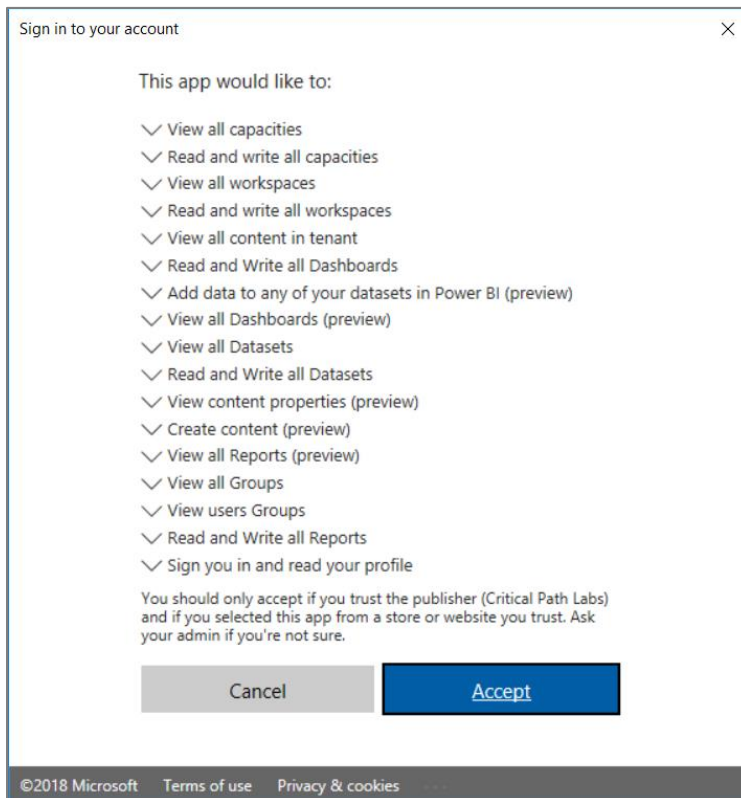
```
// enter the correct configuration values for your environment
static string appworkspaceId = "dfe5e680-a85a-4731-8c89-963fa5c6c86e";
static string clientId = "0e0dd766-4db3-4dab-bc27-baecab09864d";
static string redirectUrl = "https://localhost/app1234";
```

- 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 prompted to sign in, log in using your Office 365 user account credentials.



- c) When prompted with the **Required permissions** dialog, click **Accept**.



- d) The application should run and call into the Power BI Service API to retrieve data about the contents of the app workspace.

```
C:\Windows\system32\cmd.exe
Listing assets in app workspace: dfe5e680-a85a-4731-8c89-963fa5c6c86e

Datasets:
- Wingtip Sales Analysis [8524226b-d91e-463f-b8c2-17427af29659]

Reports:
- Wingtip Sales Analysis [A618FA1F-4EF3-493B-B51D-E2C65DC0D0A3]

Dashboards:
- Wingtip Sales Analysis [f0c94cca-cfb3-4f38-afec-f3af729ee5f3]

Press any key to continue . . .
```

Since you will be running this program quite a few times as you write more code, it will make development less tedious if you modify the **GetAccessToken** method so it can run in an unattended fashion without requiring you to sign in interactively.

7. Modify the **GetAccessToken** method to acquire access tokens using the User Password Credential flow.

- a) The following code listing shows the current implementation of the **GetAccessToken** method.

```
static string GetAccessToken() {  
    // create new authentication context  
    var authenticationContext = new AuthenticationContext(aadAuthorizationEndpoint);  
  
    // use authentication context to trigger user sign-in and return access token  
    var promptBehavior = new PlatformParameters(PromptBehavior.SelectAccount);  
    var userAuthnResult = authenticationContext.AcquireTokenAsync(resourceUriPowerBi,  
                                                                clientId,  
                                                                new Uri(redirectUrl),  
                                                                promptBehavior).Result;  
  
    // return access token to caller  
    return userAuthnResult.AccessToken;  
}
```

- b) Replace the code in **GetAccessToken** with the following code which implements the User Password Credentials flow.

```
static string GetAccessToken() {  
    // create new authentication context  
    var authenticationContext = new AuthenticationContext(aadAuthorizationEndpoint);  
  
    // use authentication context to sign-in using User Password Credentials flow  
    string masterUserAccount = "ACCOUNT_NAME_OF_MASTER_USER";  
    string masterUserPassword = "PASSWORD_OF_MASTER_USER";  
    UserPasswordCredential creds = new UserPasswordCredential(masterUserAccount, masterUserPassword);  
  
    var userAuthnResult =  
        authenticationContext.AcquireTokenAsync(resourceUriPowerBi, clientId, creds).Result;  
  
    // return access token to caller  
    return userAuthnResult.AccessToken;  
}
```

- c) Update the variables **masterUserAccount** and **masterUserPassword** with the credentials for your Office 365 account.

```
// use authentication context to sign-in using User Password Credentials flow  
string masterUserAccount = "student@portlandembed.onmicrosoft.com";  
string masterUserPassword = "pass@word1";  
UserPasswordCredential creds = new UserPasswordCredential(masterUserAccount, masterUserPassword);
```

- d) Save your changes to **program.cs**.

8. Run the application to call to the Power BI Service API.

- a) Press the **{F5}** key to begin a debugging session.
b) The program should run as it did before but it should no longer require you to interactively enter a user name and password.

Exercise 3: Write C# Code to Create an App Workspace and Upload a PBIX Project File

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

1. Create a new C# Console application in Visual Studio.

- a) x

```
static string CreateAppWorkspace(string Name) {  
    PowerBIClient pbiclient = GetPowerBIClient();  
    // create new app workspace  
    GroupCreationRequest request = new GroupCreationRequest(Name);  
    Group aws = pbiclient.Groups.CreateGroup(request);  
    // return app workspace ID  
    return aws.Id;  
}
```

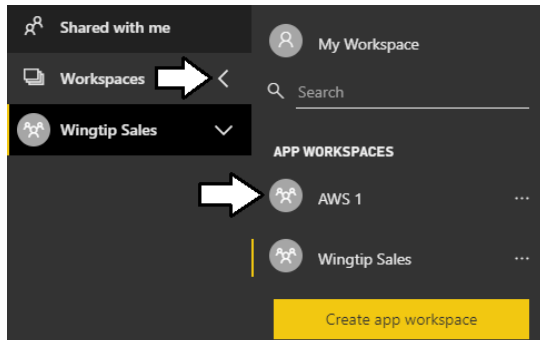


```
}
```

b) Modify Main

```
static void Main() {  
    //DisplayPersonalWorkspaceAssets();  
    CreateAppWorkspace("AWS 1");  
}
```

c) x



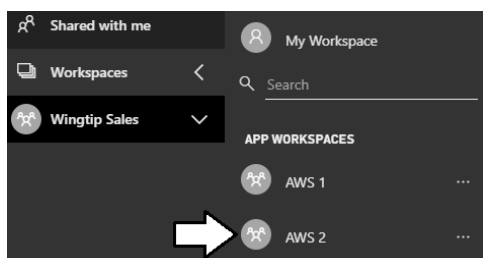
d) x

```
static void PublishPBIX(string appworkspaceId, string PbixFilePath, string ImportName) {  
    Console.WriteLine("Publishing " + PbixFilePath);  
    PowerBIClient pbiclient = GetPowerBIClient();  
    FileStream stream = new FileStream(PbixFilePath, FileMode.Open, FileAccess.Read);  
    var import = pbiclient.Imports.PostImportWithFileInGroup(appworkspaceId, stream, ImportName);  
    Console.WriteLine("Publishing process completed");  
}
```

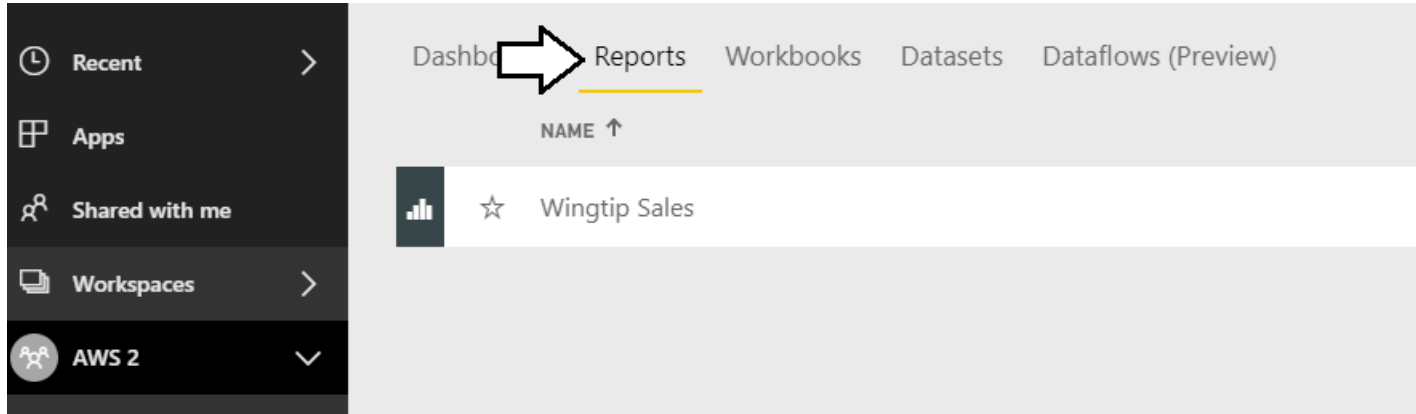
e) x

```
static void Main() {  
    //DisplayPersonalWorkspaceAssets();  
    //CreateAppWorkspace("AWS 1");  
  
    string appworkspaceId = CreateAppWorkspace("AWS 2");  
  
    string pbixPath = @"C:\Student\PBIX\wingtip Sales Analysis.pbix";  
    string importName = "wingtip Sales";  
    PublishPBIX(appworkspaceId, pbixPath, importName);  
}
```

f) x



g) x



h) x

Exercise 4: Write C# Code to Copy Power BI Content Across Workspaces

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

1. Create a new C# Console application in Visual Studio.
- 2.

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