# Getting Started with the Power BI API – Querying the Power BI REST API Directly (with C#)

**blog.jpries.com**/2020/01/03/getting-started-with-the-power-bi-api-querying-the-power-bi-rest-api-directly-with-c

jpries

January 4, 2020

In my previous post I explored the idea of accessing the Power BI API via PowerShell for the purpose of reading report inventory and other object and configuration information and then loading that exported data into SQL Server tables to query to answer various questions about an organization's configuration.

In this post, I'm going to do the same thing, but taken to the next level (and going a bit overboard) by accessing the Power BI REST API directly with a C# application I wrote.

This post is targeted toward those already comfortable writing basic programs (C#, Python, or anything capable of making a web request and parsing a JSON response). If this isn't you, there is no harm in sticking with the PowerShell cmdlets, they're very capable and fast to get started with. For anyone else, let's dive in!

For a language to use, I picked C#, with the goal of creating a basic command line based console application, as I was already pretty familiar with C# as well as how to make a web request, receive the response, and deserialize the JSON response into usable objects. My goal was to create a console application that I could run, which would then query the API for a full inventory of Power BI assets for an organization, then store the results in easy-to-query SQL server tables.

The first challenge I encountered (and even now still was probably the biggest challenge of this project) was authentication. With PowerShell, authentication was easy — run the "Login-PowerBI" (aka "Connect-PowerBIServiceAccount") cmdlet, receive a graphical authentication prompt, and then the session is authenticated. With my C# application, I had a bit more work to do…

**Authentication**

From my research, it looked like I had two solid options for authenticating to the Power BI API:

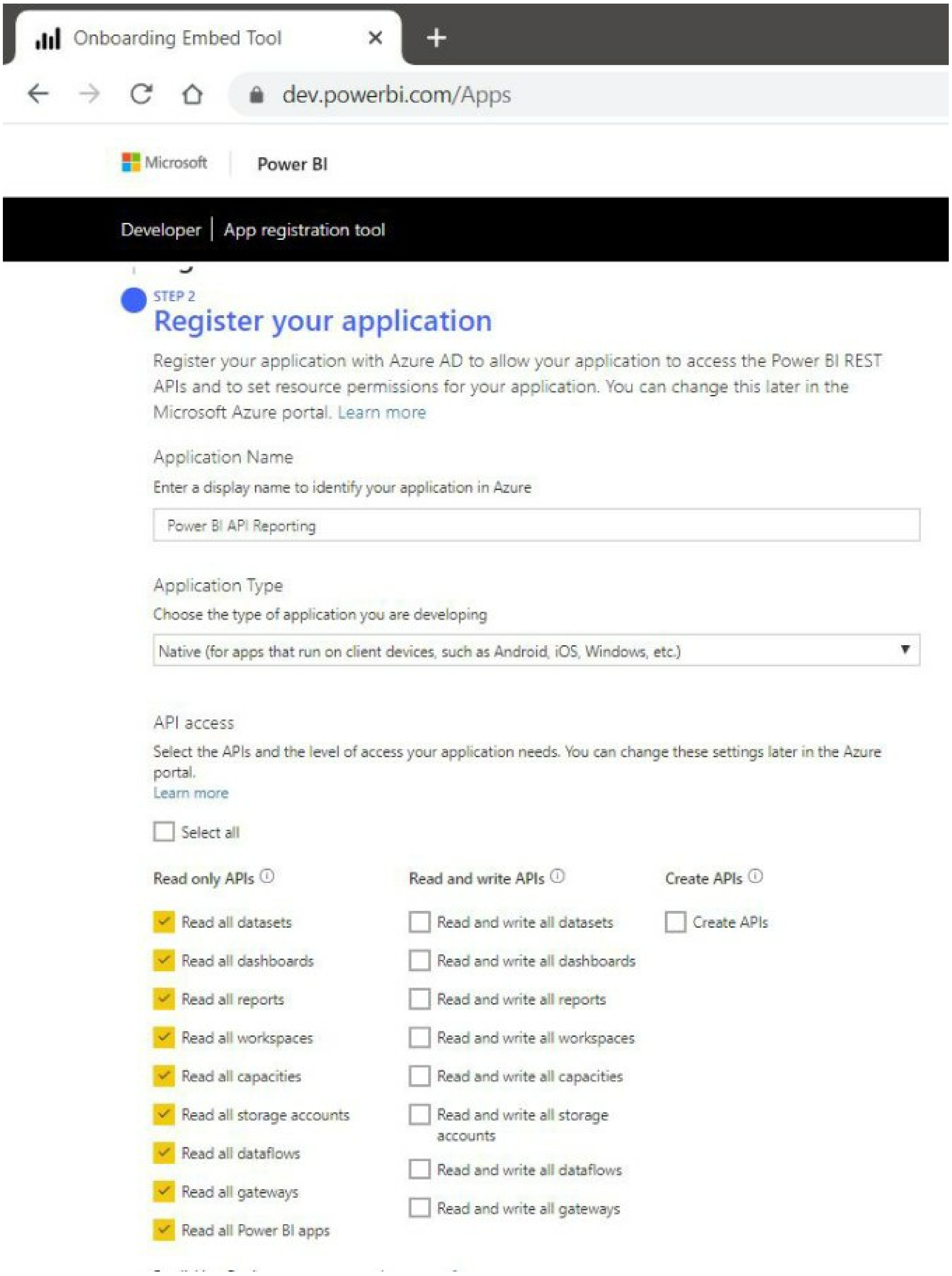
1. Use the PowerShell cmdlets — This is the first thing I tried, as it was pretty simple to do. This consisted of launching two PowerShell commands. The first performs the authentication and the second exports the token to a string which can then be included in all subsequent commands:
   - Connect-PowerBIServiceAccount
   - Get-PowerBIAccessToken -AsString
2. Authenticate to Azure Active Directory (ADAL) with an interactive login or credentials stored in the application. This performs the authentication and then provides an authentication token string which can then be provided with all subsequent API requests.

Option 2 was my preferred option as my goal with this exercise was to not have to rely on the PowerShell cmdlets. But figuring out exactly how to do it was easier said than done. It took me quite a bit of trial and error to fit the pieces together — especially the nuances between doing a standard interactive login vs. saving credentials in the application and sending them without a login prompt. In figuring all of this out, Microsoft posted a very helpful sample application in the EmbedReport.aspx.cs file and the Utils.cs file. Additionally, between this post from Benjamin Perkins, this sample from Ravi Mandal, this post on saved credentials, and finally this Microsoft sample, I was able to learn how to use the AcquireTokenAsync function with Azure Active Directory which ultimately was the method I used.

Fortunately, you don't have to go through all of that. I've pieced all those good nuggets of information together in my included sample application. Instead you can spend your time on the more interesting part — the data.

Some important notes on configuring authentication:

1. While the sample code shows how to put the pieces together, it is still necessary to register your application in Azure Active Directory to be able to perform an authentication. I followed the steps in this article, which refers to this link, to create a registration for my program and grant it access to read Power BI data for my organization (note, an account to authenticate is still needed). It is important to register as a **Native** app and make a note of the Application ID that is generated. You can always go back and review the settings for the app via this link.
2. Make sure to note the Application ID. Technically you also need the Authority URL, Redirect URL, and Resource URL — though these are known values not unique to the application.
3. For this method, your application will require the Microsoft Identity Model Client library, which can be installed in Visual Studio via Nuget.



Onboarding Embed Tool

dev.powerbi.com/Apps

Microsoft | Power BI

Developer | App registration tool

STEP 2

Register your application

Register your application with Azure AD to allow your application to access the Power BI REST APIs and to set resource permissions for your application. You can change this later in the Microsoft Azure portal. Learn more

Application Name

Enter a display name to identify your application in Azure

Power BI API Reporting

Application Type

Choose the type of application you are developing

Native (for apps that run on client devices, such as Android, iOS, Windows, etc.)

API access

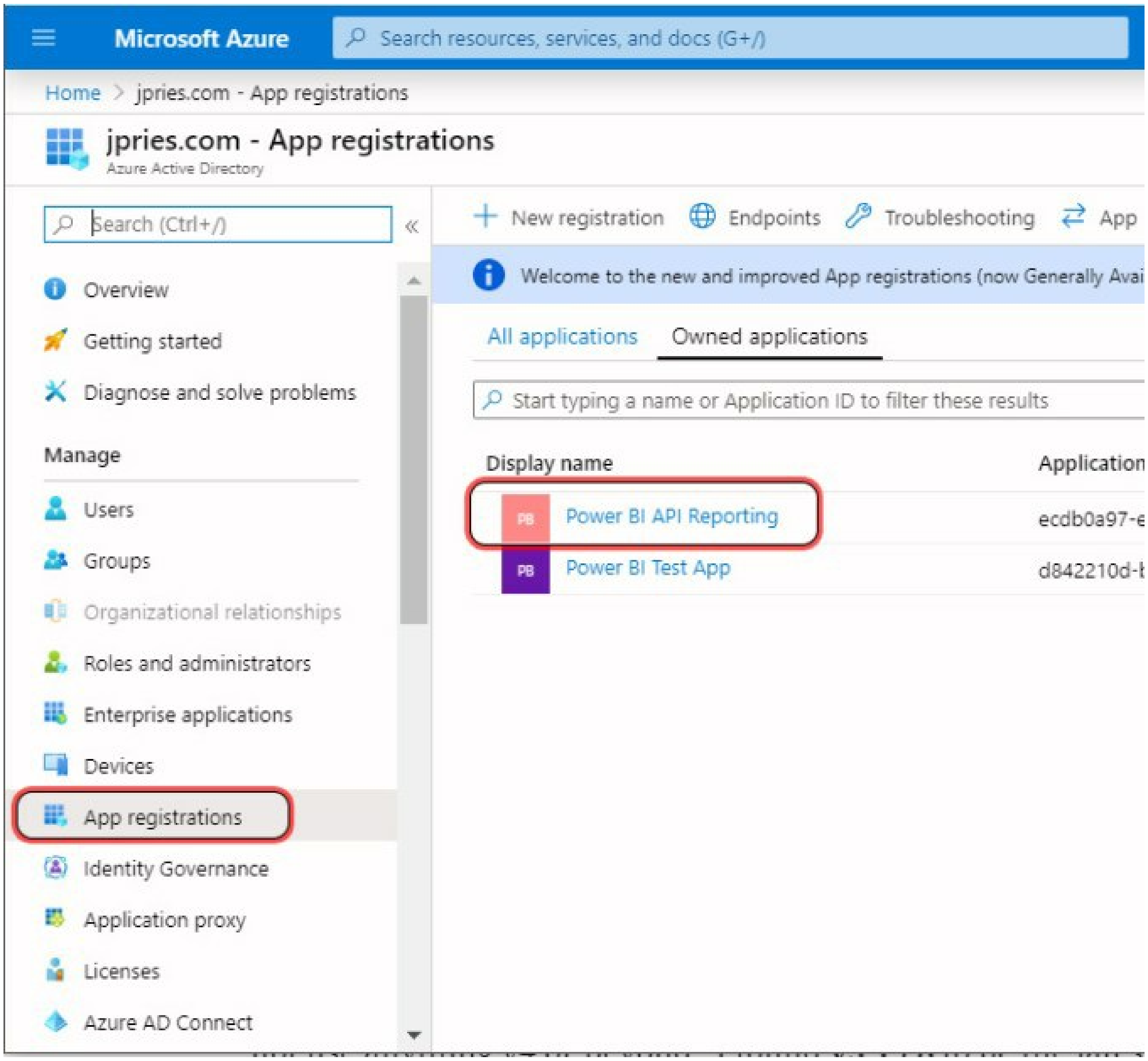
Select the APIs and the level of access your application needs. You can change these settings later in the Azure portal.
Learn more

☐ Select all

| Read only APIs | Read and write APIs | Create APIs |
|---|---|---|
| ✓ Read all datasets | ☐ Read and write all datasets | ☐ Create APIs |
| ✓ Read all dashboards | ☐ Read and write all dashboards | |
| ✓ Read all reports | ☐ Read and write all reports | |
| ✓ Read all workspaces | ☐ Read and write all workspaces | |
| ✓ Read all capacities | ☐ Read and write all capacities | |
| ✓ Read all storage accounts | ☐ Read and write all storage accounts | |
| ✓ Read all dataflows | ☐ Read and write all dataflows | |
| ✓ Read all gateways | ☐ Read and write all gateways | |
| ✓ Read all Power BI apps | | |

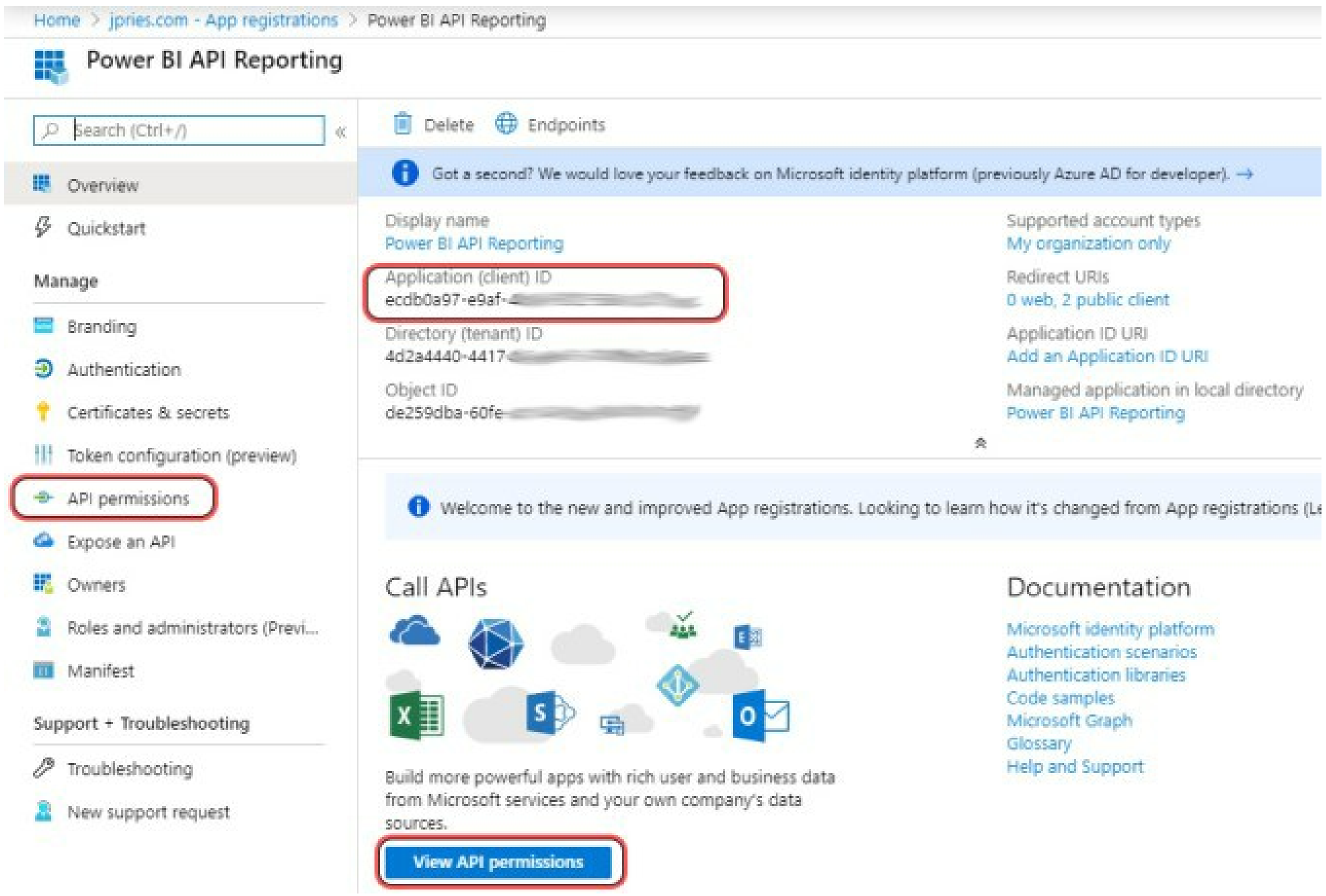
The dev.powerbi.com/apps app registration wizard

You can always revisit these settings (or create a new registration) later by visiting https://portal.azure.com then by selecting Azure Active Directory and going to the "App Registrations" section, finding your registration, then clicking on the App registrations button and viewing the API permissions.



| Display name | Application |
|---|---|
| PB Power BI API Reporting | ecdb0a97-e |
| PB Power BI Test App | d842210d-t |

In Azure Active Directory at https://portal.azure.com, click on App Registrations then click on your registration or create a New Registration



Home > jpries.com - App registrations > Power BI API Reporting

Power BI API Reporting

Display name: Power BI API Reporting

Application (client) ID: ecdb0a97-e9af-[redacted]

Directory (tenant) ID: 4d2a4440-4417[redacted]

Object ID: de259dba-60fe[redacted]

Supported account types: My organization only

Redirect URIs: 0 web, 2 public client

Application ID URI: Add an Application ID URI

Managed application in local directory: Power BI API Reporting

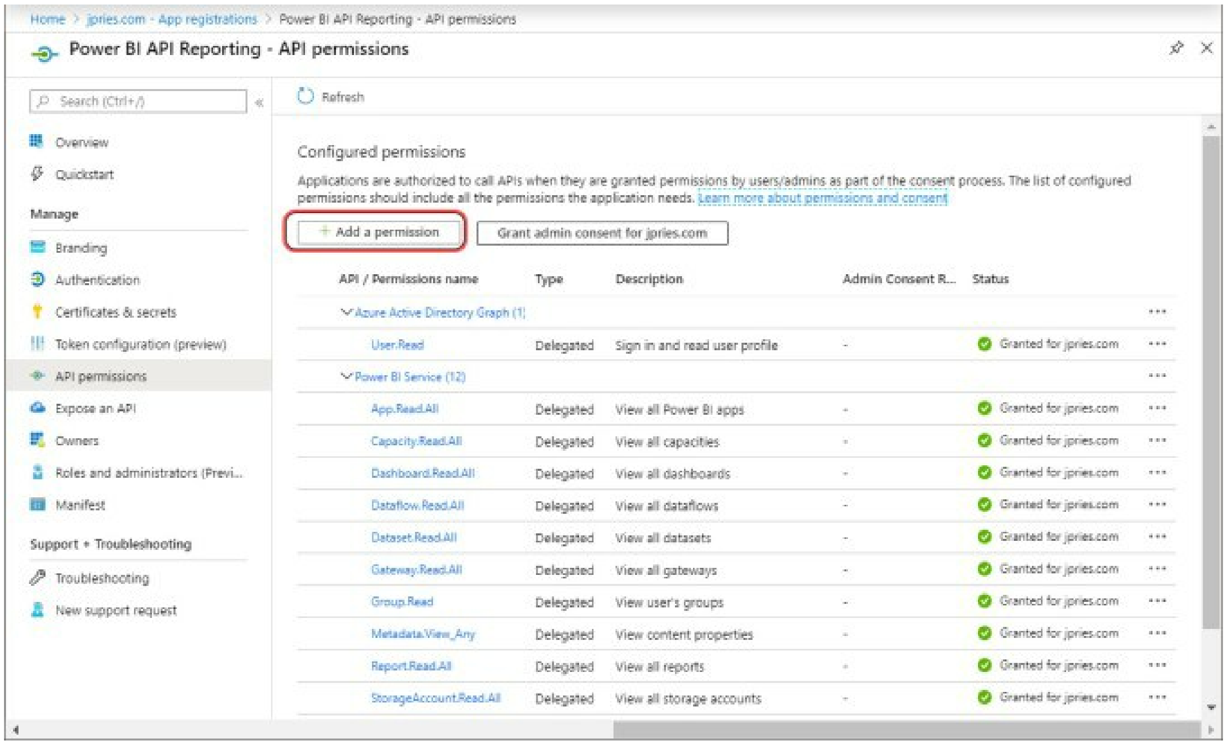
Got a second? We would love your feedback on Microsoft identity platform (previously Azure AD for developer). →

Welcome to the new and improved App registrations. Looking to learn how it's changed from App registrations (Le

Call APIs

Build more powerful apps with rich user and business data from Microsoft services and your own company's data sources.

Within your registration, click on View API permissions either on the left menu or bottom button. Also note that your Application Client ID used in your application is visible at the top of the page.



Home > jpries.com - App registrations > Power BI API Reporting - API permissions

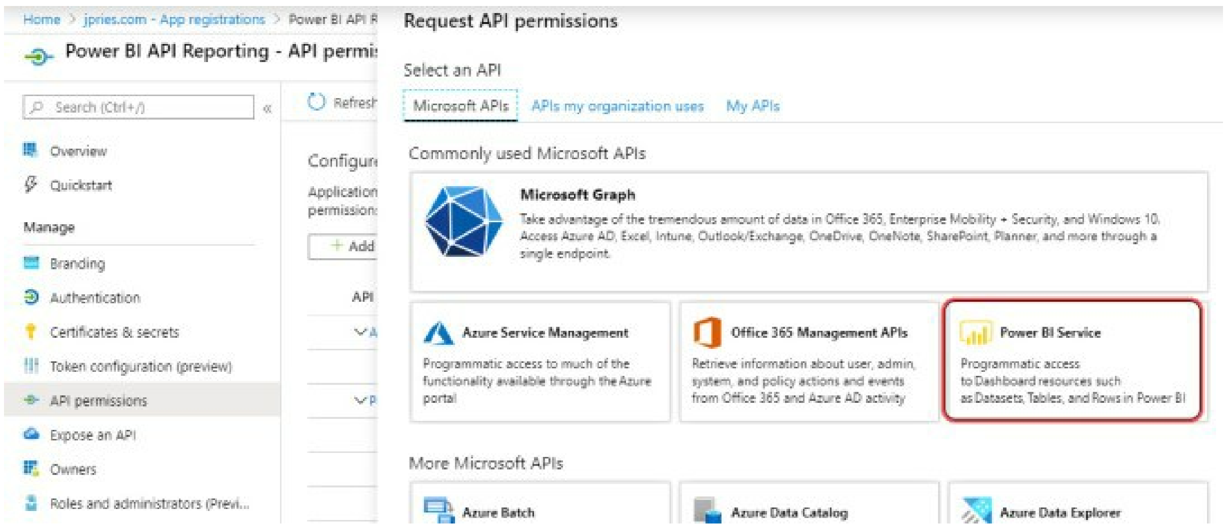
Power BI API Reporting - API permissions

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. Learn more about permissions and consent

| API / Permissions name | Type | Description | Admin Consent R... | Status |
|---|---|---|---|---|
| Azure Active Directory Graph (1) | | | | |
| User.Read | Delegated | Sign in and read user profile | - | ✓ Granted for jpries.com |
| Power BI Service (12) | | | | |
| App.Read.All | Delegated | View all Power BI apps | - | ✓ Granted for jpries.com |
| Capacity.Read.All | Delegated | View all capacities | - | ✓ Granted for jpries.com |
| Dashboard.Read.All | Delegated | View all dashboards | - | ✓ Granted for jpries.com |
| Dataflow.Read.All | Delegated | View all dataflows | - | ✓ Granted for jpries.com |
| Dataset.Read.All | Delegated | View all datasets | - | ✓ Granted for jpries.com |
| Gateway.Read.All | Delegated | View all gateways | - | ✓ Granted for jpries.com |
| Group.Read | Delegated | View user's groups | - | ✓ Granted for jpries.com |
| Metadata.View_Any | Delegated | View content properties | - | ✓ Granted for jpries.com |
| Report.Read.All | Delegated | View all reports | - | ✓ Granted for jpries.com |
| StorageAccount.Read.All | Delegated | View all storage accounts | - | ✓ Granted for jpries.com |

View the permissions on this page and add additional ones with the Add a permission button. Note that permissions starting with Tenant. will require an O365 admin to visit this page to click on the "Grant admin consent" button.


Home > jpries.com - App registrations > Power BI API R

Power BI API Reporting - API permi

Search (Ctrl+/)

Overview

Quickstart

Manage

Branding

Authentication

Certificates & secrets

Token configuration (preview)

API permissions

Expose an API

Owners

Roles and administrators (Previ...

Refresh

Configure

Application permission

+ Add

API

Request API permissions

Select an API

Microsoft APIs | APIs my organization uses | My APIs

Commonly used Microsoft APIs

**Microsoft Graph**
Take advantage of the tremendous amount of data in Office 365, Enterprise Mobility + Security, and Windows 10. Access Azure AD, Excel, Intune, Outlook/Exchange, OneDrive, OneNote, SharePoint, Planner, and more through a single endpoint.

**Azure Service Management**
Programmatic access to much of the functionality available through the Azure portal

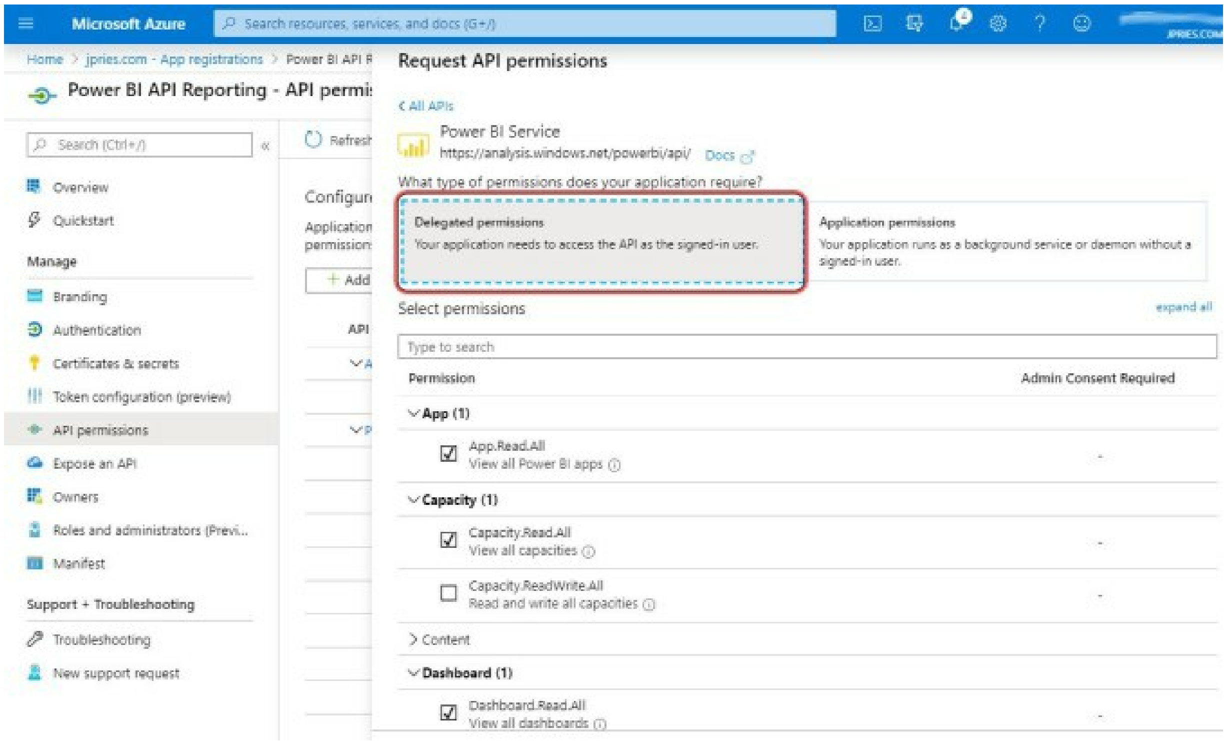
**Office 365 Management APIs**
Retrieve information about user, admin, system, and policy actions and events from Office 365 and Azure AD activity

**Power BI Service**
Programmatic access to Dashboard resources such as Datasets, Tables, and Rows in Power BI

More Microsoft APIs

Azure Batch | Azure Data Catalog | Azure Data Explorer

After clicking on Add a Permission, select Power BI Service.


Microsoft Azure | Search resources, services, and docs (G+/)

Home > jpries.com - App registrations > Power BI API R

Power BI API Reporting - API permi

Search (Ctrl+/)

Overview

Quickstart

Manage

Branding

Authentication

Certificates & secrets

Token configuration (preview)

API permissions

Expose an API

Owners

Roles and administrators (Previ...

Manifest

Support + Troubleshooting

Troubleshooting

New support request

Refresh

Configure

Application permission

+ Add

API

Request API permissions

< All APIs

Power BI Service
https://analysis.windows.net/powerbi/api/ Docs

What type of permissions does your application require?

**Delegated permissions**
Your application needs to access the API as the signed-in user.

**Application permissions**
Your application runs as a background service or daemon without a signed-in user.

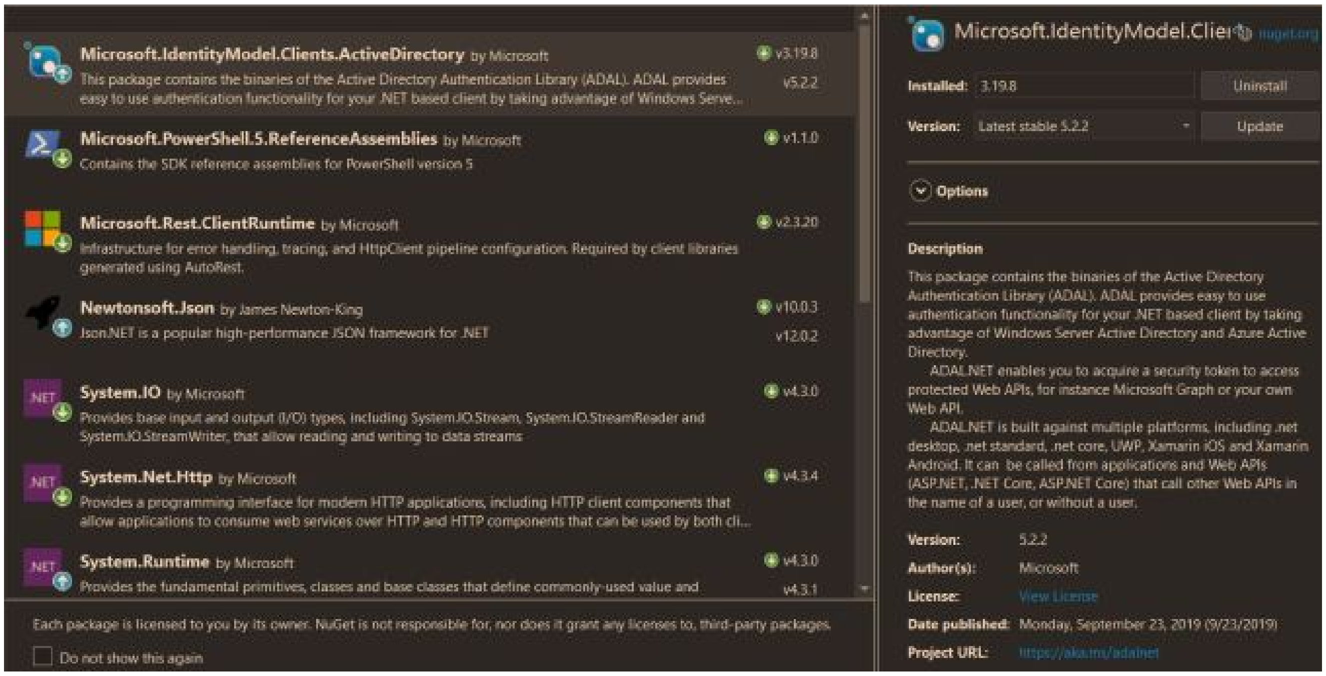
Select permissions | expand all

Type to search

| Permission | Admin Consent Required |
|---|---|
| **App (1)** | |
| ☑ App.Read.All<br>View all Power BI apps | - |
| **Capacity (1)** | |
| ☑ Capacity.Read.All<br>View all capacities | - |
| ☐ Capacity.ReadWrite.All<br>Read and write all capacities | - |
| > Content | |
| **Dashboard (1)** | |
| ☑ Dashboard.Read.All<br>View all dashboards | - |

*Ensure that you select "Delegated permissions" (the option to use when the application connects with user credentials). Select the desired permissions and press Add permissions. Note that Tenant.Read.All and Tenant.Write.All permissions will require Azure AD / O365 admin consent before becoming effective. Also note that you need at least one Power BI permission assigned to your application to be able to authenticate as a Power BI application.*

I encountered an interesting quirk while troubleshooting authentication issues. The current version of the Microsoft Identity Model Client library (using the https://login.windows.net/common/oauth2/authorize AuthorityURL) to access the Power BI API doesn't work. Instead, it is very important to not use anything v4 or beyond. I found **v3.19.8** to be the latest v3 implementation and that worked great.



**Microsoft.IdentityModel.Clients.ActiveDirectory** by Microsoft v3.19.8 v5.2.2
This package contains the binaries of the Active Directory Authentication Library (ADAL). ADAL provides easy to use authentication functionality for your .NET based client by taking advantage of Windows Serve...

**Microsoft.PowerShell.5.ReferenceAssemblies** by Microsoft v1.1.0
Contains the SDK reference assemblies for PowerShell version 5

**Microsoft.Rest.ClientRuntime** by Microsoft v2.3.20
Infrastructure for error handling, tracing, and HttpClient pipeline configuration. Required by client libraries generated using AutoRest.

**Newtonsoft.Json** by James Newton-King v10.0.3 v12.0.2
Json.NET is a popular high-performance JSON framework for .NET

**System.IO** by Microsoft v4.3.0
Provides base input and output (I/O) types, including System.IO.Stream, System.IO.StreamReader and System.IO.StreamWriter, that allow reading and writing to data streams

**System.Net.Http** by Microsoft v4.3.4
Provides a programming interface for modern HTTP applications, including HTTP client components that allow applications to consume web services over HTTP and HTTP components that can be used by both cli...

**System.Runtime** by Microsoft v4.3.0 v4.3.1
Provides the fundamental primitives, classes and base classes that define commonly-used value and

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

☐ Do not show this again

Microsoft.IdentityModel.Clie nuget.org

**Installed:** 3.19.8 Uninstall

**Version:** Latest stable 5.2.2 Update

**Options**

**Description**

This package contains the binaries of the Active Directory Authentication Library (ADAL). ADAL provides easy to use authentication functionality for your .NET based client by taking advantage of Windows Server Active Directory and Azure Active Directory.
ADAL.NET enables you to acquire a security token to access protected Web APIs, for instance Microsoft Graph or your own Web API.
ADAL.NET is built against multiple platforms, including .net desktop, .net standard, .net core, UWP, Xamarin iOS and Xamarin Android. It can be called from applications and Web APIs (ASP.NET, .NET Core, ASP.NET Core) that call other Web APIs in the name of a user, or without a user.

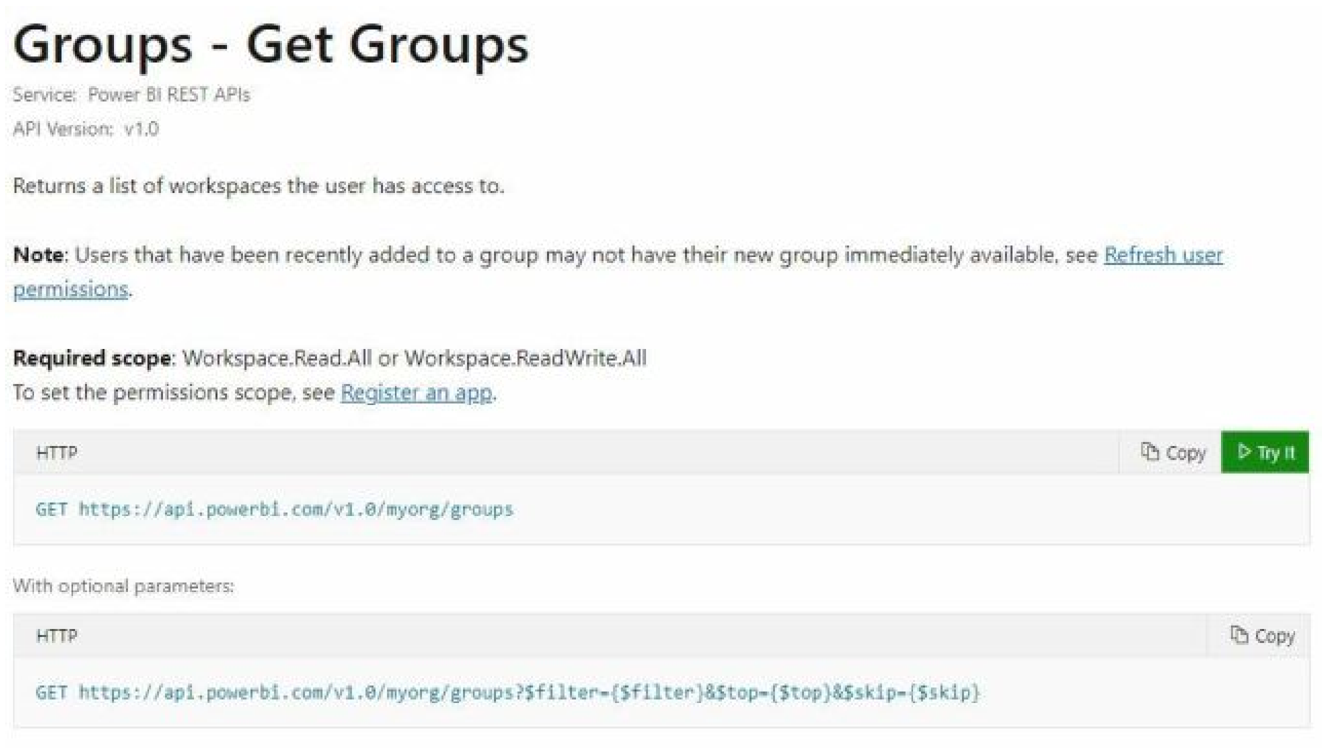
| | |
|---|---|
| **Version:** | 5.2.2 |
| **Author(s):** | Microsoft |
| **License:** | View License |
| **Date published:** | Monday, September 23, 2019 (9/23/2019) |
| **Project URL:** | https://aka.ms/adalnet |

## Data

Thankfully, after wrangling with authentication and getting a valid authentication token, handling the data is relatively straightforward.

The Power BI API Reference is well documented and ever expanding. The reference breaks the available commands into groups with a number of read and write functions typically in each group. For the purposes of this article, I've only used the few read operations — primarily for collecting inventory information about my environment. When getting started, I recommend starting with commands that start with the "Get" prefix.

To get information about Workspaces, for instance, that can be found in the "Groups" group of the reference (this is the only one that isn't named intuitively — reflecting a name change of the functions along the way.) The "Get Groups" function will return information for Workspaces in the Power BI Service for which you have access.


Groups - Get Groups

Service: Power BI REST APIs

API Version: v1.0

Returns a list of workspaces the user has access to.

**Note**: Users that have been recently added to a group may not have their new group immediately available, see Refresh user permissions.

**Required scope**: Workspace.Read.All or Workspace.ReadWrite.All
To set the permissions scope, see Register an app.

```
GET https://api.powerbi.com/v1.0/myorg/groups
```

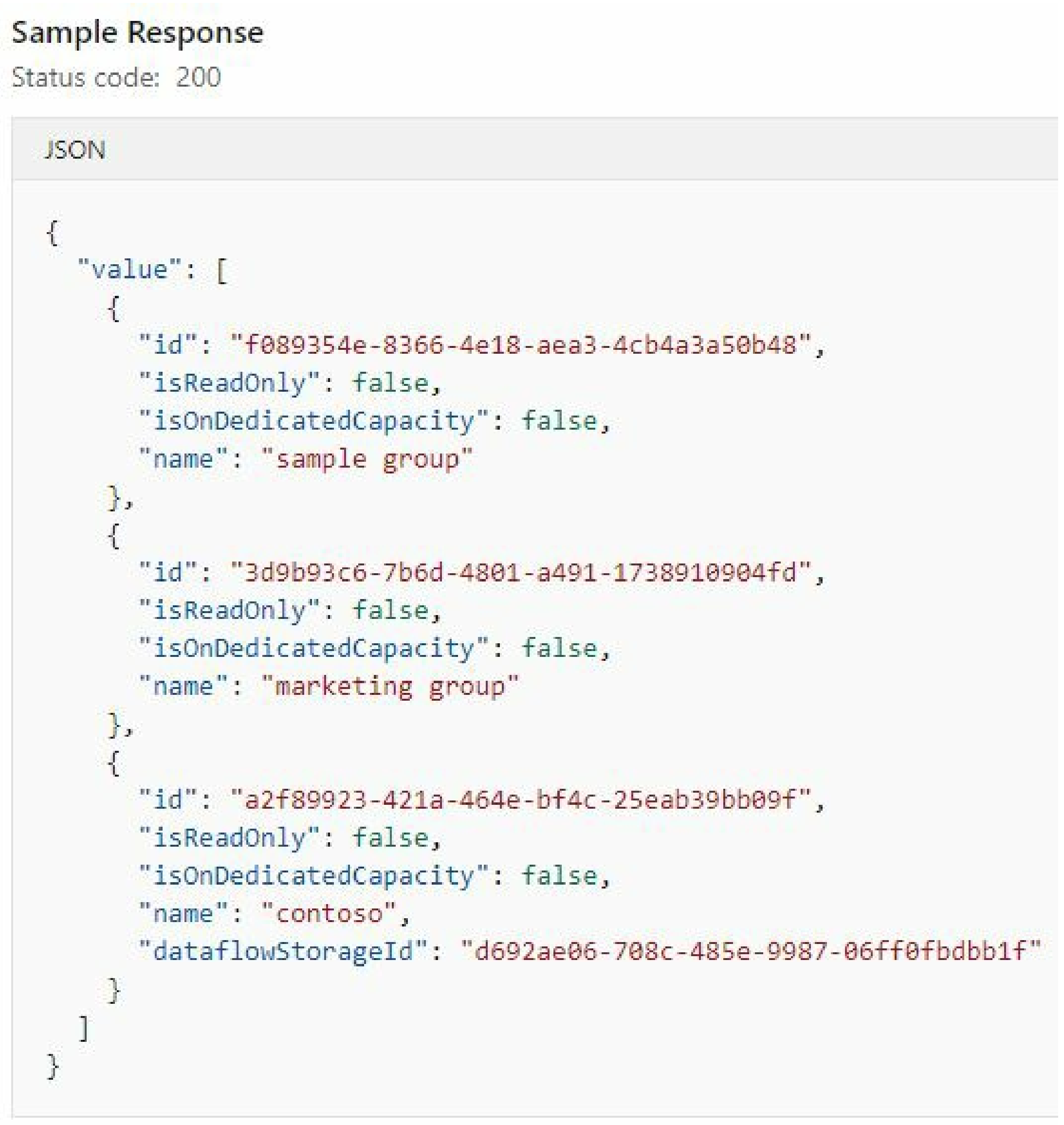
With optional parameters:

```
GET https://api.powerbi.com/v1.0/myorg/groups?$filter={$filter}&$top={$top}&$skip={$skip}
```

For each command, the method for submitting the request to the service is specified (in this case, and most, it’s the GET method) as well as the URI to send the request to. Many commands will also allow you to specify parameters in the request — for instance if you needed to get all of the datasets for a particular workspace (group), the workspace unique identifier would be a parameter.

Note on the top of each API page, it lists what application permissions are needed for the function. Most permissions can be assigned by a user. To use any function in the Admin section, the Tenant.Read.All permission is needed, which will require an Office 365 administrator to approve the permission grant.

A successful request will return a JSON response which can then be de-serialized using your method of choice back into usable data.


Sample Response

Status code: 200

JSON

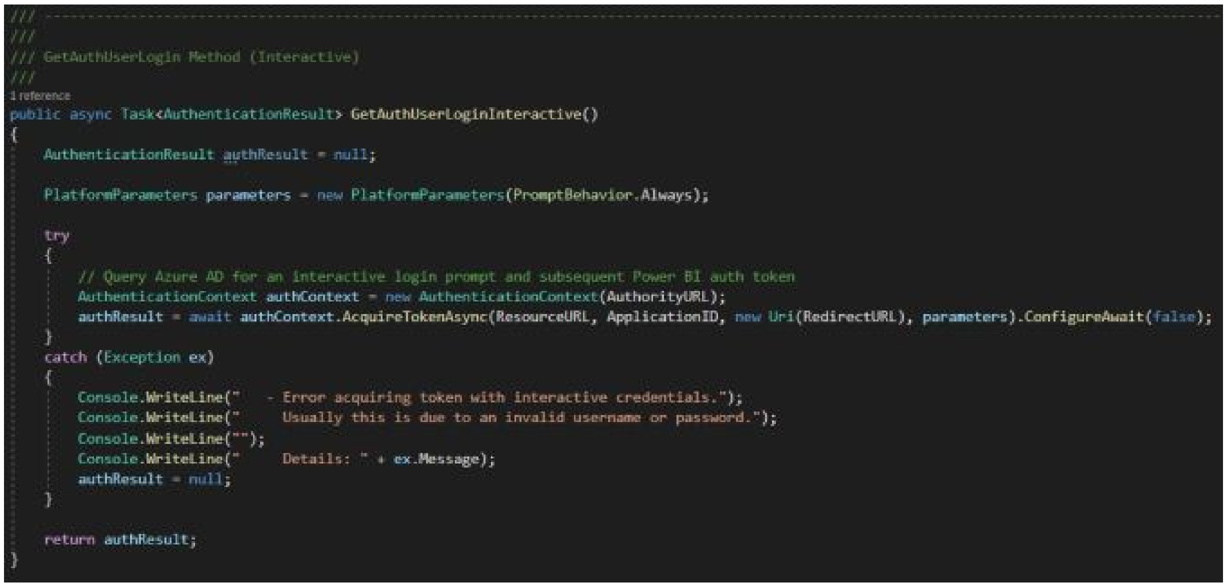
```
{
  "value": [
    {
      "id": "f089354e-8366-4e18-aea3-4cb4a3a50b48",
      "isReadOnly": false,
      "isOnDedicatedCapacity": false,
      "name": "sample group"
    },
    {
      "id": "3d9b93c6-7b6d-4801-a491-1738910904fd",
      "isReadOnly": false,
      "isOnDedicatedCapacity": false,
      "name": "marketing group"
    },
    {
      "id": "a2f89923-421a-464e-bf4c-25eab39bb09f",
      "isReadOnly": false,
      "isOnDedicatedCapacity": false,
      "name": "contoso",
      "dataflowStorageId": "d692ae06-708c-485e-9987-06ff0fbdbb1f"
    }
  ]
}
```

In my sample C# console application, I make use of the HttpClient class, which is built into the .NET framework, to send the authentication request, receive the response, and then send a data request and receive the response. Additionally, I make use of the JavaScriptSerializer class (also built into the .NET framework) to de-serialize the JSON data response received into individual objects I can output to the screen.

**Sample Application**

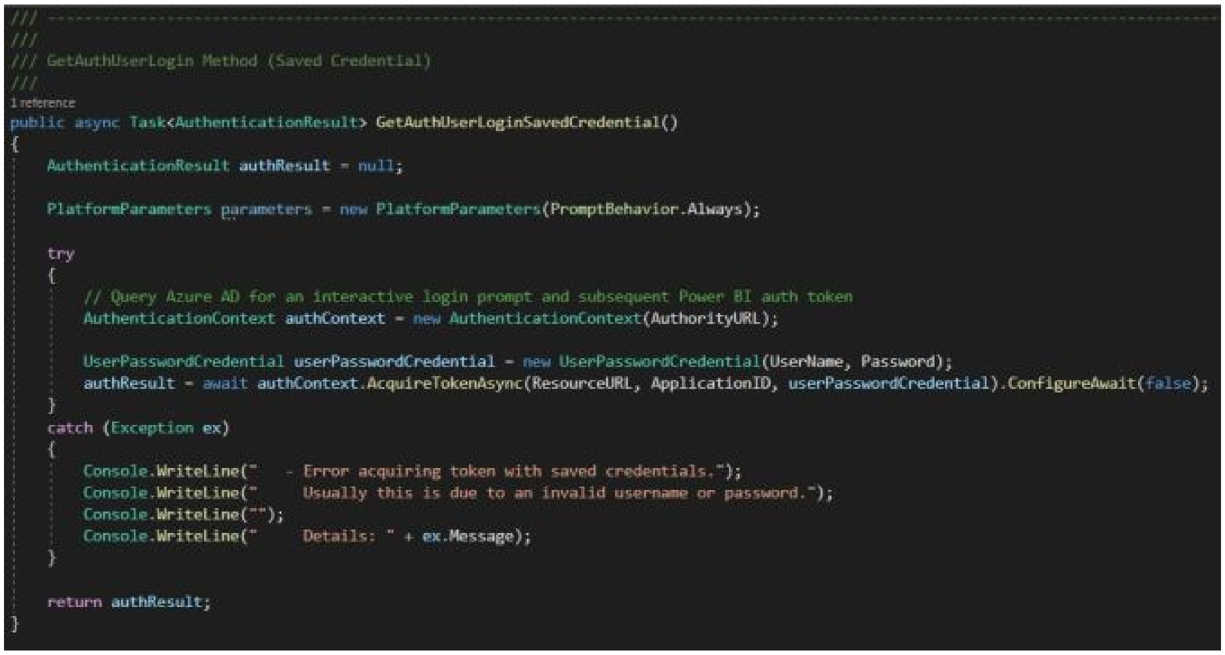
To tie things altogether, I developed a very basic C# .NET console application, the Visual Studio Solution for which can be found here.

The application is fairly minimal with all of its configuration information stored in constants at the top of the main class source file, PowerBIDataExportSample.cs to make everything as simple as possible. Aside from querying Azure Active Directory and Power BI, the program does not read or write any other information, except for displaying it to the console. As such, it has minimal dependencies.



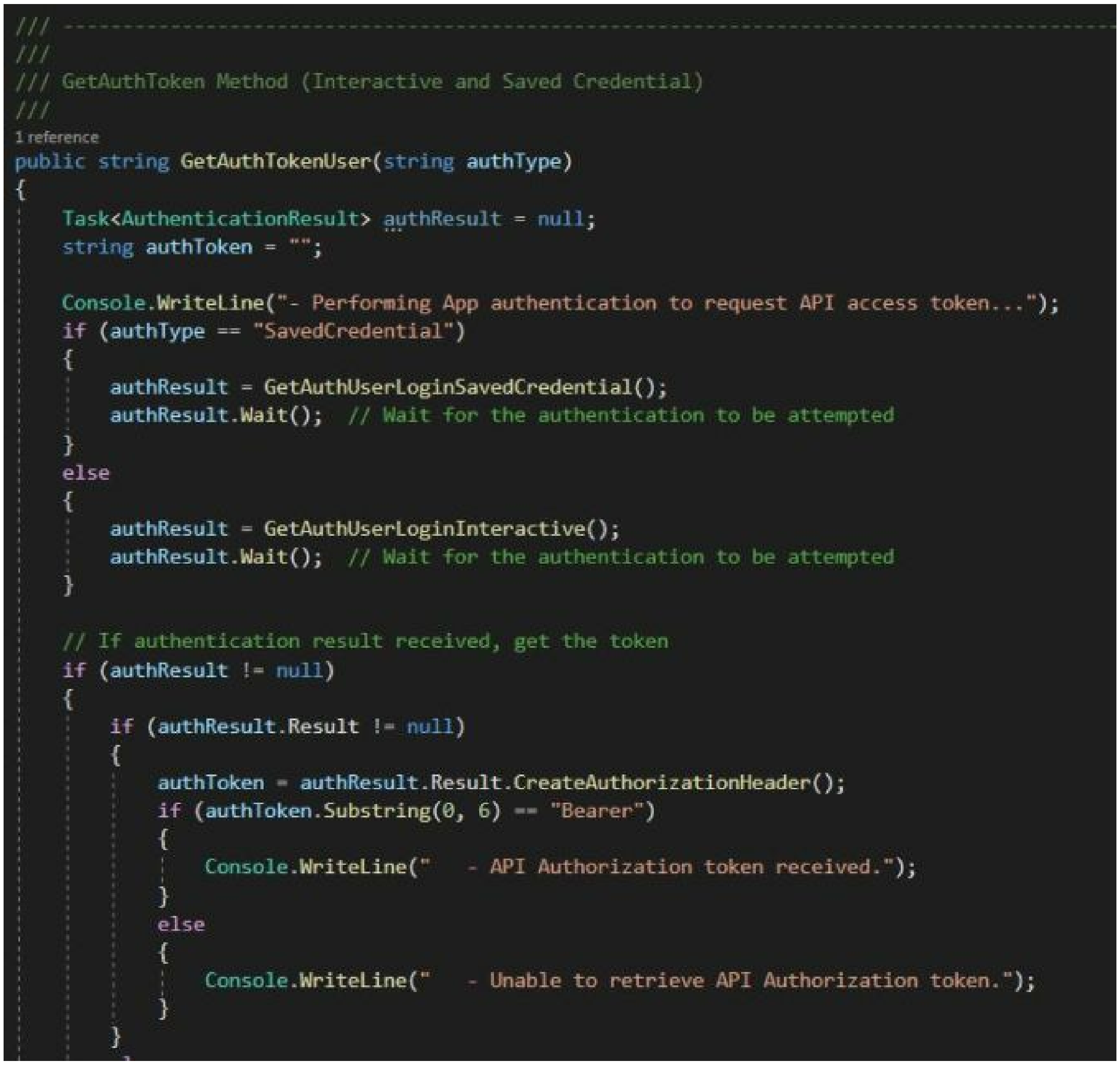
```
/// -----------------------------------------------------------------------------------------------------------------
///
/// GetAuthUserLogin Method (Interactive)
///
1 reference
public async Task<AuthenticationResult> GetAuthUserLoginInteractive()
{
    AuthenticationResult authResult = null;

    PlatformParameters parameters = new PlatformParameters(PromptBehavior.Always);

    try
    {
        // Query Azure AD for an interactive login prompt and subsequent Power BI auth token
        AuthenticationContext authContext = new AuthenticationContext(AuthorityURL);
        authResult = await authContext.AcquireTokenAsync(ResourceURL, ApplicationID, new Uri(RedirectURL), parameters).ConfigureAwait(false);
    }
    catch (Exception ex)
    {
        Console.WriteLine("   - Error acquiring token with interactive credentials.");
        Console.WriteLine("     Usually this is due to an invalid username or password.");
        Console.WriteLine("");
        Console.WriteLine("     Details: " + ex.Message);
        authResult = null;
    }

    return authResult;
}
```

Sample code snippet to perform an interactive authentication to Azure Active Directory for Power BI.



```
/// -----------------------------------------------------------------------------------------------------------------
///
/// GetAuthUserLogin Method (Saved Credential)
///
1 reference
public async Task<AuthenticationResult> GetAuthUserLoginSavedCredential()
{
    AuthenticationResult authResult = null;

    PlatformParameters parameters = new PlatformParameters(PromptBehavior.Always);

    try
    {
        // Query Azure AD for an interactive login prompt and subsequent Power BI auth token
        AuthenticationContext authContext = new AuthenticationContext(AuthorityURL);

        UserPasswordCredential userPasswordCredential = new UserPasswordCredential(UserName, Password);
        authResult = await authContext.AcquireTokenAsync(ResourceURL, ApplicationID, userPasswordCredential).ConfigureAwait(false);
    }
    catch (Exception ex)
    {
        Console.WriteLine("   - Error acquiring token with saved credentials.");
        Console.WriteLine("     Usually this is due to an invalid username or password.");
        Console.WriteLine("");
        Console.WriteLine("     Details: " + ex.Message);
    }

    return authResult;
}
```

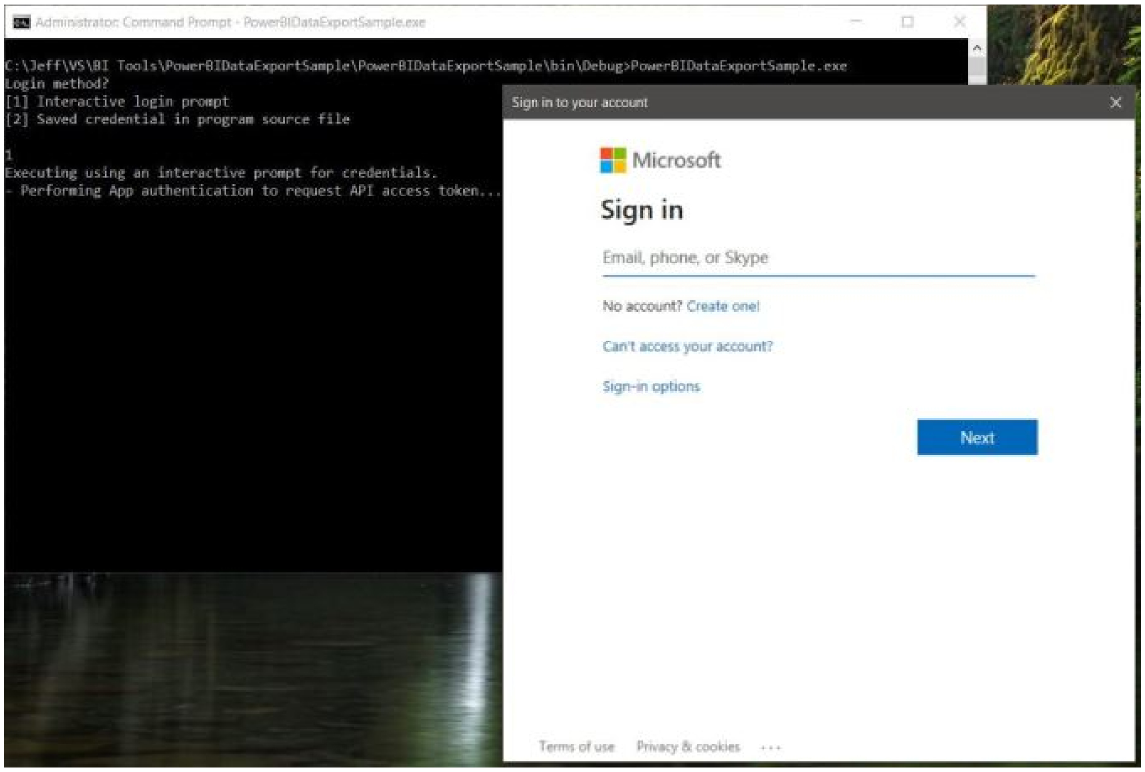
*Sample code snippet to perform a saved credential authentication to Azure Active Directory for Power BI.*



```
/// -------------------------------------------------------------------------------
///
/// GetAuthToken Method (Interactive and Saved Credential)
///
1 reference
public string GetAuthTokenUser(string authType)
{
    Task<AuthenticationResult> authResult = null;
    string authToken = "";

    Console.WriteLine("- Performing App authentication to request API access token...");
    if (authType == "SavedCredential")
    {
        authResult = GetAuthUserLoginSavedCredential();
        authResult.Wait();  // Wait for the authentication to be attempted
    }
    else
    {
        authResult = GetAuthUserLoginInteractive();
        authResult.Wait();  // Wait for the authentication to be attempted
    }

    // If authentication result received, get the token
    if (authResult != null)
    {
        if (authResult.Result != null)
        {
            authToken = authResult.Result.CreateAuthorizationHeader();
            if (authToken.Substring(0, 6) == "Bearer")
            {
                Console.WriteLine("   - API Authorization token received.");
            }
            else
            {
                Console.WriteLine("   - Unable to retrieve API Authorization token.");
            }
        }
```

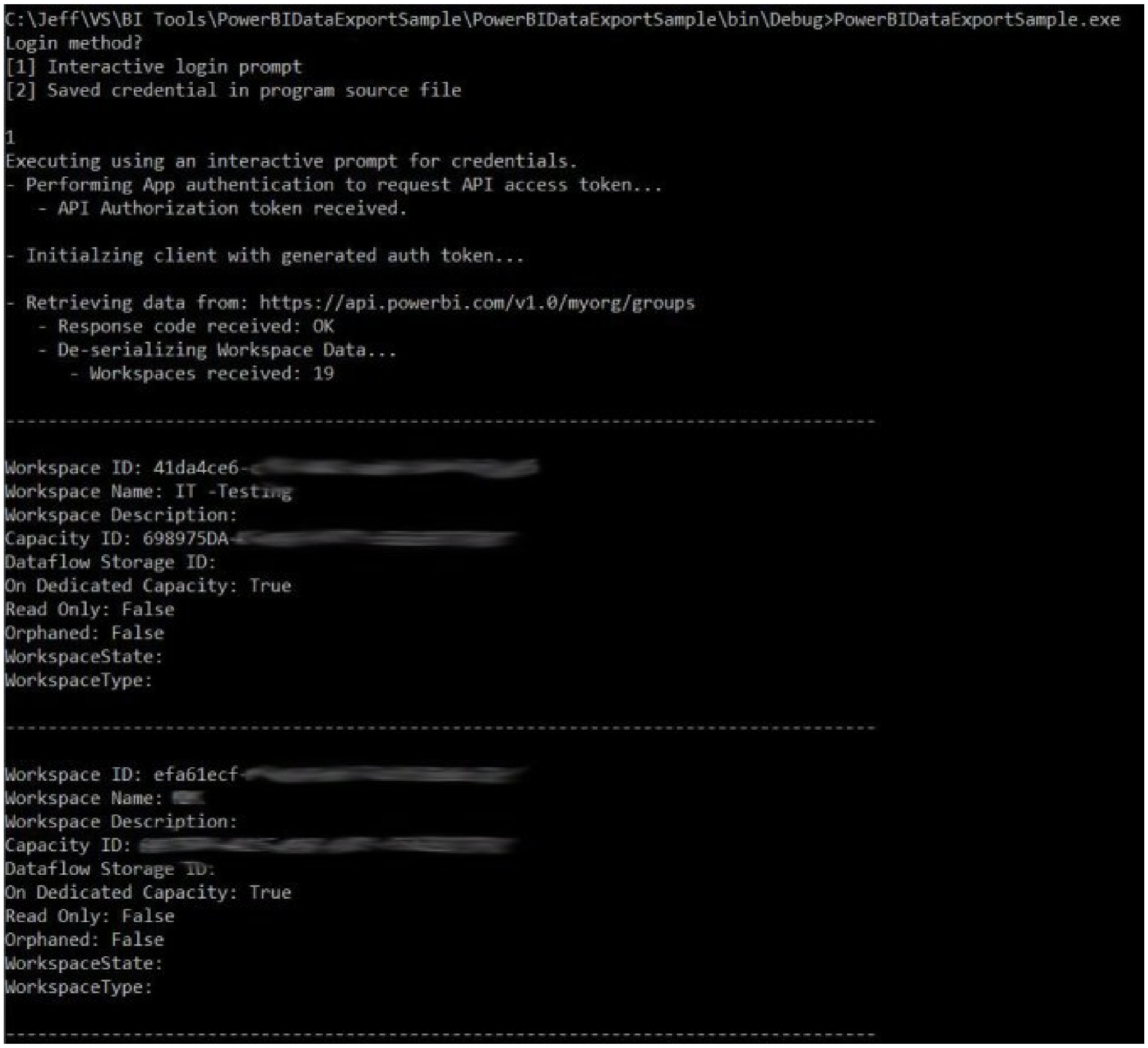
*Sample code snippet to call either the Interactive or Saved Credential method, wait for the response, and then request the "bearer token" which is then used in subsequent API requests.*

When run, the application prompts for whether you'd like to perform an interactive authentication (recommended) or use the values which were stored in the source file when compiled. Once authentication credentials have been entered (which are sent directly to Azure Active Directory) the program sends the "Get Workspaces" command to the Power BI REST API, receives the result JSON file, de-serializes it into individual pieces of data, and outputs the data to the screen.



```
Administrator: Command Prompt - PowerBIDataExportSample.exe

C:\Jeff\VS\BI Tools\PowerBIDataExportSample\PowerBIDataExportSample\bin\Debug>PowerBIDataExportSample.exe
Login method?
[1] Interactive login prompt
[2] Saved credential in program source file

1
Executing using an interactive prompt for credentials.
- Performing App authentication to request API access token...
```

Interactive login prompt when called from my sample Power BI console application.



```
C:\Jeff\VS\BI Tools\PowerBIDataExportSample\PowerBIDataExportSample\bin\Debug>PowerBIDataExportSample.exe
Login method?
[1] Interactive login prompt
[2] Saved credential in program source file

1
Executing using an interactive prompt for credentials.
- Performing App authentication to request API access token...
   - API Authorization token received.

- Initialzing client with generated auth token...

- Retrieving data from: https://api.powerbi.com/v1.0/myorg/groups
   - Response code received: OK
   - De-serializing Workspace Data...
      - Workspaces received: 19

--------------------------------------------------------------------------------

Workspace ID: 41da4ce6-[redacted]
Workspace Name: IT -Testing
Workspace Description:
Capacity ID: 698975DA-[redacted]
Dataflow Storage ID:
On Dedicated Capacity: True
Read Only: False
Orphaned: False
WorkspaceState:
WorkspaceType:

--------------------------------------------------------------------------------

Workspace ID: efa61ecf-[redacted]
Workspace Name: [redacted]
Workspace Description:
Capacity ID: [redacted]
Dataflow Storage ID:
On Dedicated Capacity: True
Read Only: False
Orphaned: False
WorkspaceState:
WorkspaceType:

--------------------------------------------------------------------------------
```

Sample Power BI console application outputting workspace data from the Power BI Rest API.

Utilizing the sample above, its possible to expand much further into many of the other areas of the Power BI Service (Datasets, Data Sources, Apps, Gateways, and more) and build out a robust inventory collection system. See my GitHub for more information on this in the future.

**Resources**

**Posts in Series**