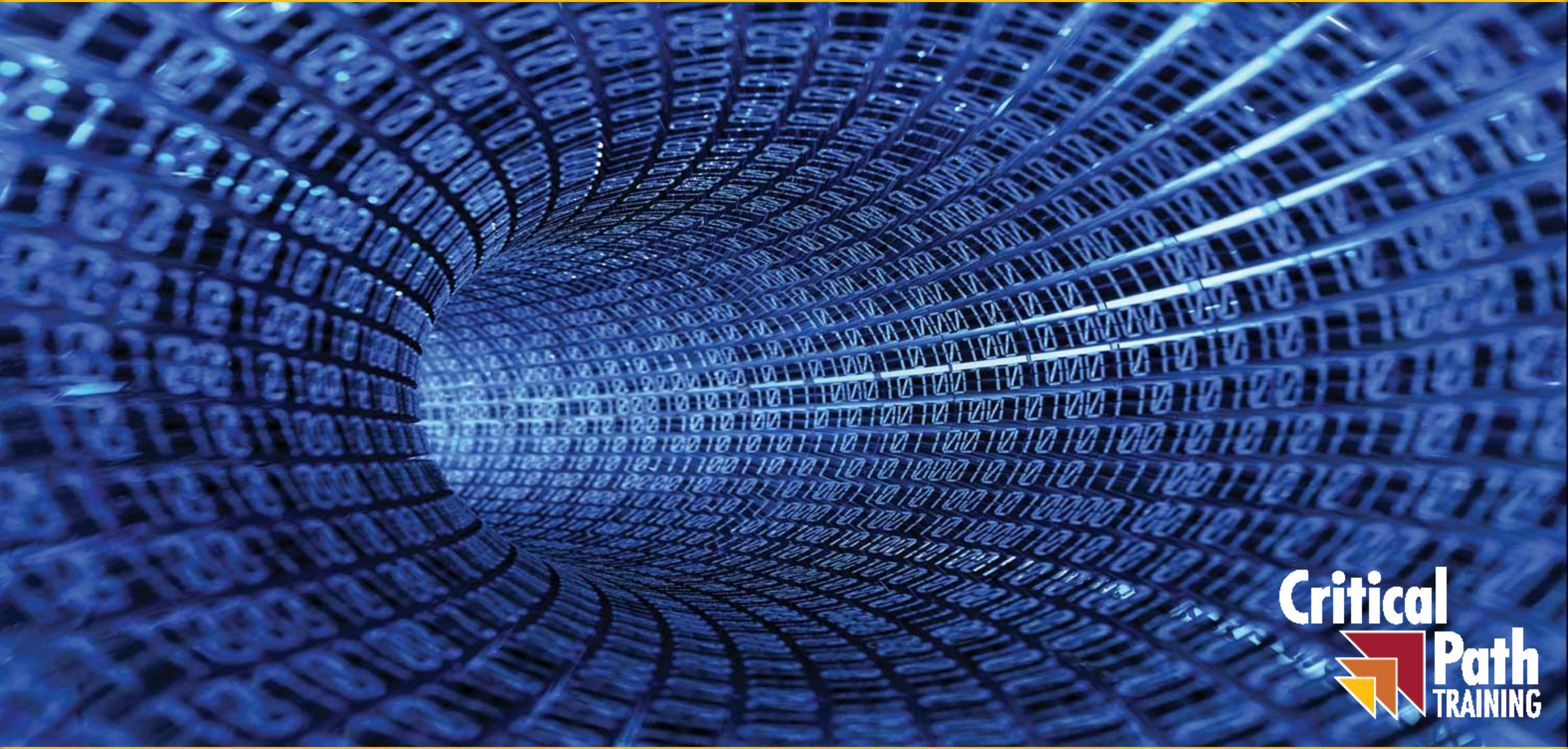
Developing with the Power BI Service API



Agenda

- ➢ Power BI Service API Overview
- Registering Applications with Azure AD
- Security Programming with Azure AD
- Importing PBIX Project Files using Code
- Programming Datasets and Data Sources

The Power BI Service API

- What is Power BI Service API?
  - API built on OAuth2, OpenID Connect, REST and ODATA
  - Allows developers to program datasets, reports and dashboards
  - Also known as **Power BI REST API** & ***Power BI API***

- What can you do with the Power BI Service API?
  - Import PBIX files
  - Perform operations on datasets and data sources
  - Embed Power BI reports and dashboards tiles in web pages
  - Create and populate streaming datasets for real-time dashboards

What Operations Are Supported in v1.0?

- Workspace Operations
  - Get Groups
- Dataset Operations
  - Get Datasets
  - Create Dataset
- Table Operations
  - Get Tables
  - Alter Table Schema
- Table Row Operations
  - Add Rows
  - Delete Rows

More Operations

- Report Operations
  - Get Reports
- Dashboard Operations
  - Get Dashboards
  - Get Dashboard Tiles
- Import (PBIX) Operations
  - Create Import
  - Get Imports
  - Get Import by GUID
  - Get Import by File Path

Getting Started

- What you need to get started?
  - User account in an Azure AD tenancy
  - Pro license for Power BI
  - Application registration in Azure AD
  - Visual Studio

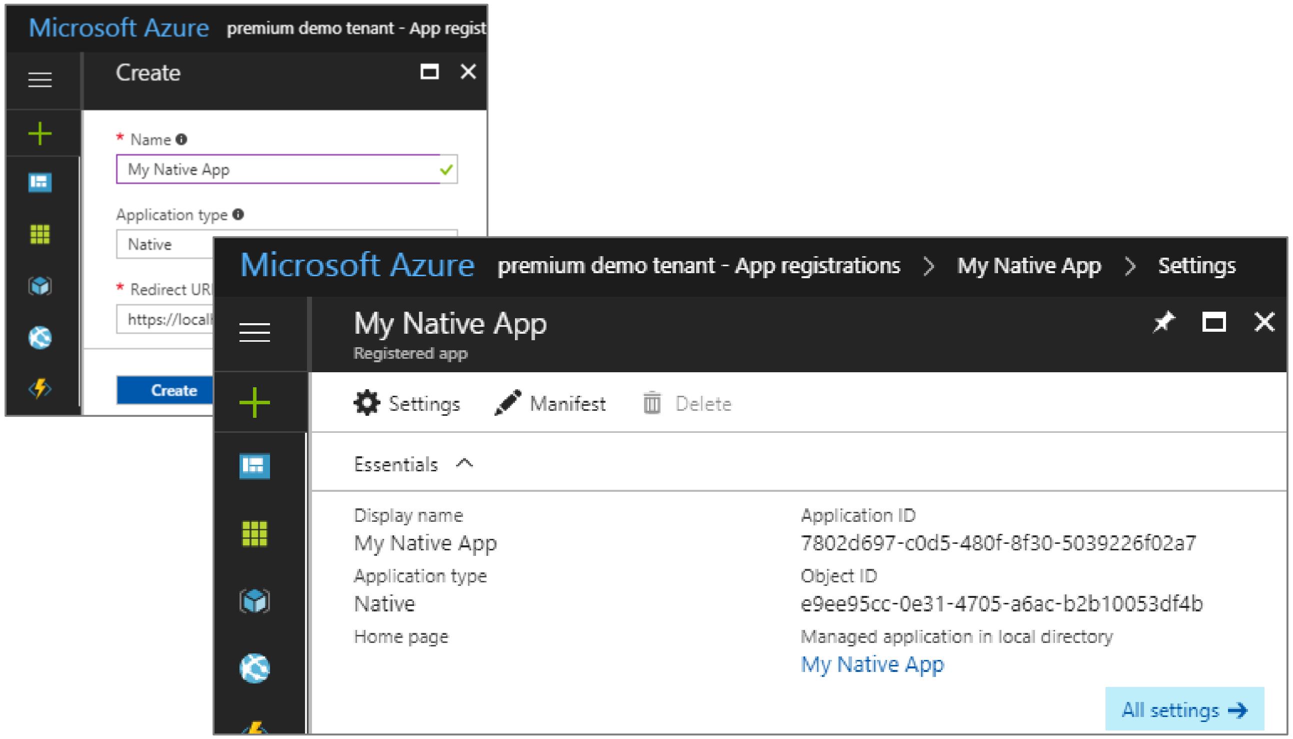
Agenda

- ✓ Power BI Service API Overview
- ➢ Registering Applications with Azure AD
- Security Programming with Azure AD
- Importing PBIX Project Files using Code
- Programming Datasets and Data Sources

Tenants and Organizational Accounts

- Azure AD used to authenticate users and apps
  - PBI licenses are assigned to Azure AD user accounts
  - Organization owns a tenant (i.e. directory)
  - AAD tenant contains user accounts and groups
  - AAD tenant contains set of registered applications
- You must register your application with Azure AD
  - Requirement of calling to Power BI service API
  - Applications registered as Web app or Native app
  - Registered applications are assigned GUID for client ID
  - Application is configured with permissions

Creating an Azure AD Application



Microsoft Azure premium demo tenant - App regist

Create

* Name: My Native App

Application type: Native

* Redirect URI: https://locall

Create

Microsoft Azure premium demo tenant - App registrations > My Native App > Settings

My Native App
Registered app

Settings | Manifest | Delete

Essentials

Display name
My Native App

Application type
Native

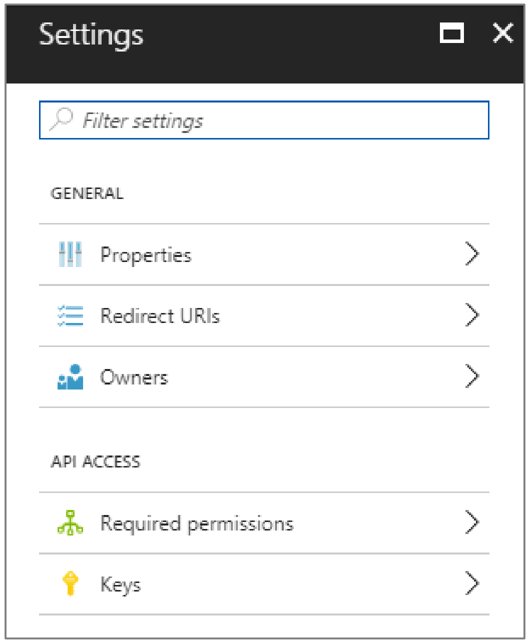
Home page

Application ID
7802d697-c0d5-480f-8f30-5039226f02a7

Object ID
e9ee95cc-0e31-4705-a6ac-b2b10053df4b

Managed application in local directory
My Native App

All settings →



Settings

Filter settings

GENERAL

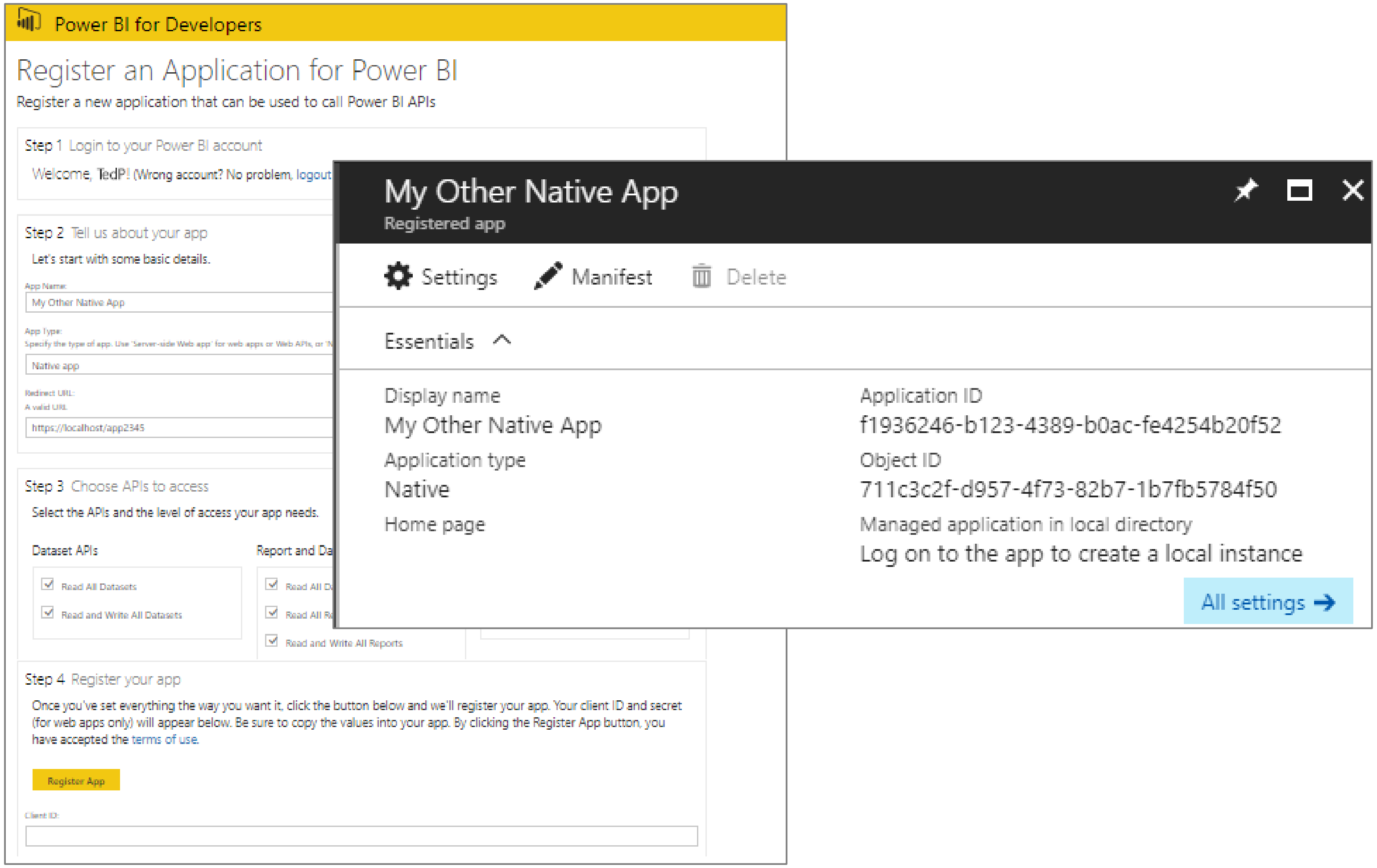
- Properties
- Redirect URIs
- Owners

API ACCESS

- Required permissions
- Keys

Power BI App Registration Page

- https://app.powerbi.com/apps



Power BI for Developers

Register an Application for Power BI

Register a new application that can be used to call Power BI APIs

Step 1 Login to your Power BI account

Welcome, TedP! (Wrong account? No problem, logout

Step 2 Tell us about your app

Let's start with some basic details.

App Name:
My Other Native App

App Type:
Specify the type of app. Use 'Server-side Web app' for web apps or Web APIs, or 'N
Native app

Redirect URL:
A valid URL
https://localhost/app2345

Step 3 Choose APIs to access

Select the APIs and the level of access your app needs.

Dataset APIs
- ✓ Read All Datasets
- ✓ Read and Write All Datasets

Report and Da
- ✓ Read All D
- ✓ Read All R
- ✓ Read and Write All Reports

Step 4 Register your app

Once you've set everything the way you want it, click the button below and we'll register your app. Your client ID and secret (for web apps only) will appear below. Be sure to copy the values into your app. By clicking the Register App button, you have accepted the terms of use.

Register App

Client ID:

My Other Native App
Registered app

Settings Manifest Delete

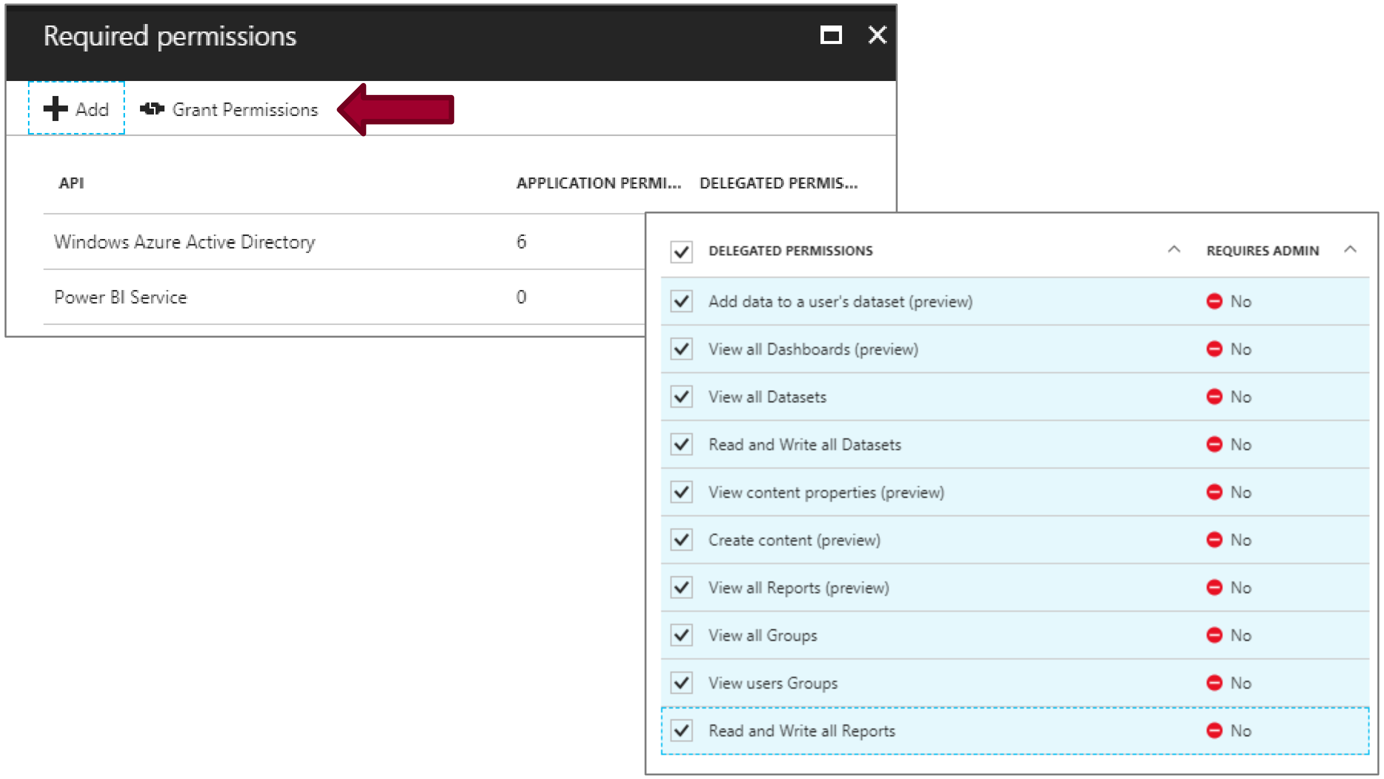
Essentials

Display name	Application ID
My Other Native App	f1936246-b123-4389-b0ac-fe4254b20f52
Application type	Object ID
Native	711c3c2f-d957-4f73-82b7-1b7fb5784f50
Home page	Managed application in local directory
	Log on to the app to create a local instance

All settings →

Application Permissions

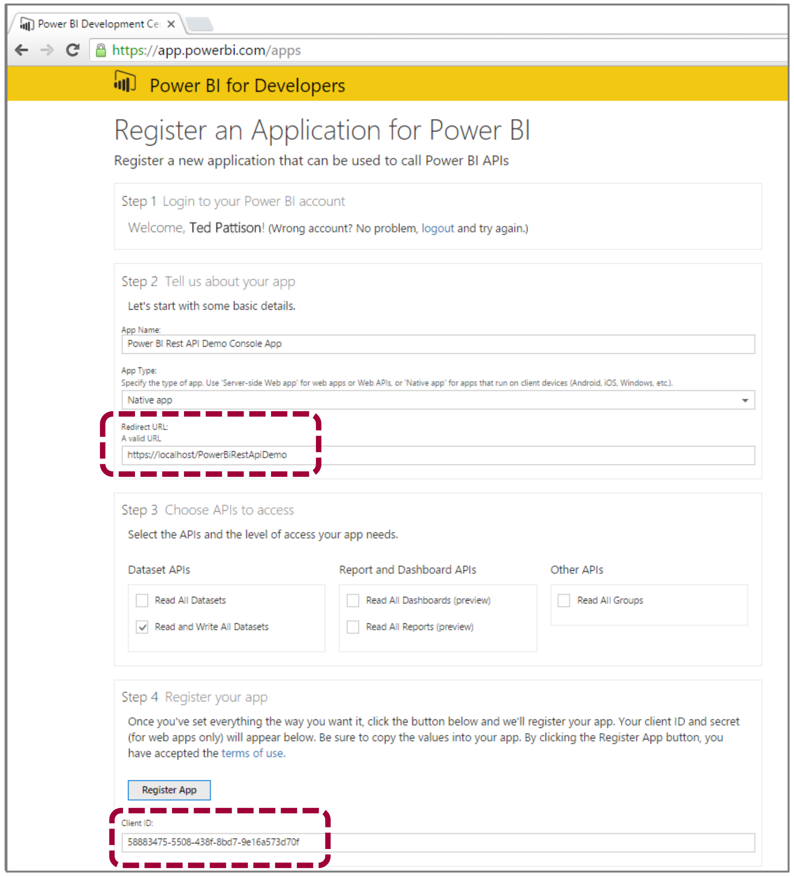
- Applications can be granted permissions to other applications
  - Application permissions are app-only permissions
  - Delegated permissions are (app + user) permissions
  - Delegated permissions requires 1-time consent from user


Required permissions

API	APPLICATION PERMI...	DELEGATED PERMIS...
Windows Azure Active Directory	6	
Power BI Service	0	

✓ DELEGATED PERMISSIONS	REQUIRES ADMIN
✓ Add data to a user's dataset (preview)	No
✓ View all Dashboards (preview)	No
✓ View all Datasets	No
✓ Read and Write all Datasets	No
✓ View content properties (preview)	No
✓ Create content (preview)	No
✓ View all Reports (preview)	No
✓ View all Groups	No
✓ View users Groups	No
✓ Read and Write all Reports	No

Power BI App Registration



Register an Application for Power BI

Register a new application that can be used to call Power BI APIs

Step 1 Login to your Power BI account

Welcome, Ted Pattison! (Wrong account? No problem, logout and try again.)

Step 2 Tell us about your app

Let's start with some basic details.

App Name:
Power BI Rest API Demo Console App

App Type:
Specify the type of app. Use 'Server-side Web app' for web apps or Web APIs, or 'Native app' for apps that run on client devices (Android, iOS, Windows, etc.).
Native app

Redirect URL:
A valid URL
https://localhost/PowerBiRestApiDemo

Step 3 Choose APIs to access

Select the APIs and the level of access your app needs.

Dataset APIs	Report and Dashboard APIs	Other APIs
☐ Read All Datasets	☐ Read All Dashboards (preview)	☐ Read All Groups
☑ Read and Write All Datasets	☐ Read All Reports (preview)	

Step 4 Register your app

Once you've set everything the way you want it, click the button below and we'll register your app. Your client ID and secret (for web apps only) will appear below. Be sure to copy the values into your app. By clicking the Register App button, you have accepted the terms of use.

Register App

Client ID:
58883475-5508-438f-8bd7-9e16a573d70f

Agenda

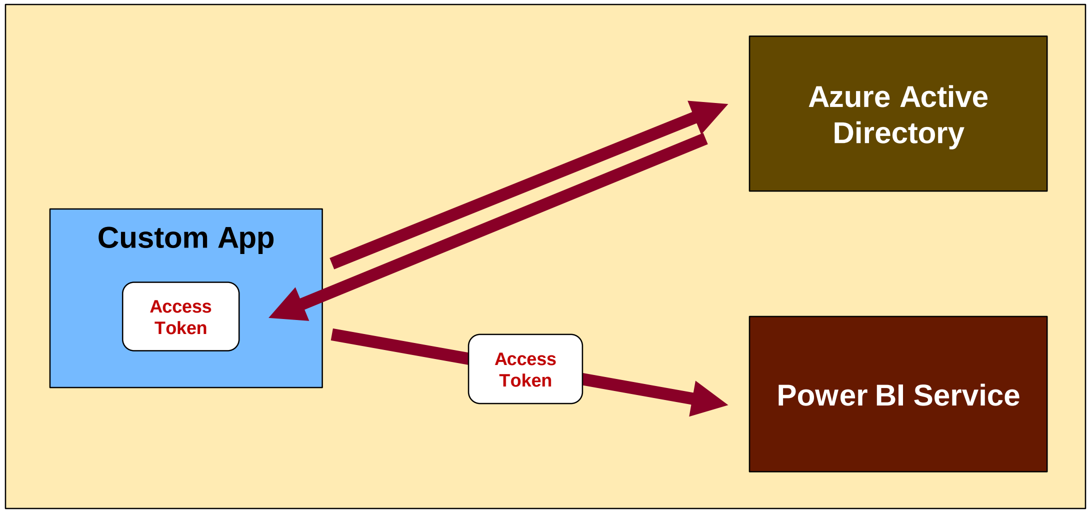
- ✓ Power BI Service API Overview
- ✓ Registering Applications with Azure AD
- ➢ Security Programming with Azure AD
- Importing PBIX Project Files using Code
- Programming Datasets and Data Sources

Authentication Flows

- Client Credentials Grant Flow *(confidential client)*
  - Authentication based on SSL certificate with public-private key pair
  - Used to obtain access token when using app-only permissions
- Authorization Code Grant Flow *(confidential client)*
  - Client first obtains authorization code then access token
  - Server-side application code never sees user's password
- Implicit Grant Flow *(public client)*
  - Used in SPAs built with JavaScript and AngularJS
  - Application obtains access token w/o acquiring authorization code
- User Credentials Flow *(public client)*
  - Used in Native clients to obtain access code
  - Requires passing user name and password

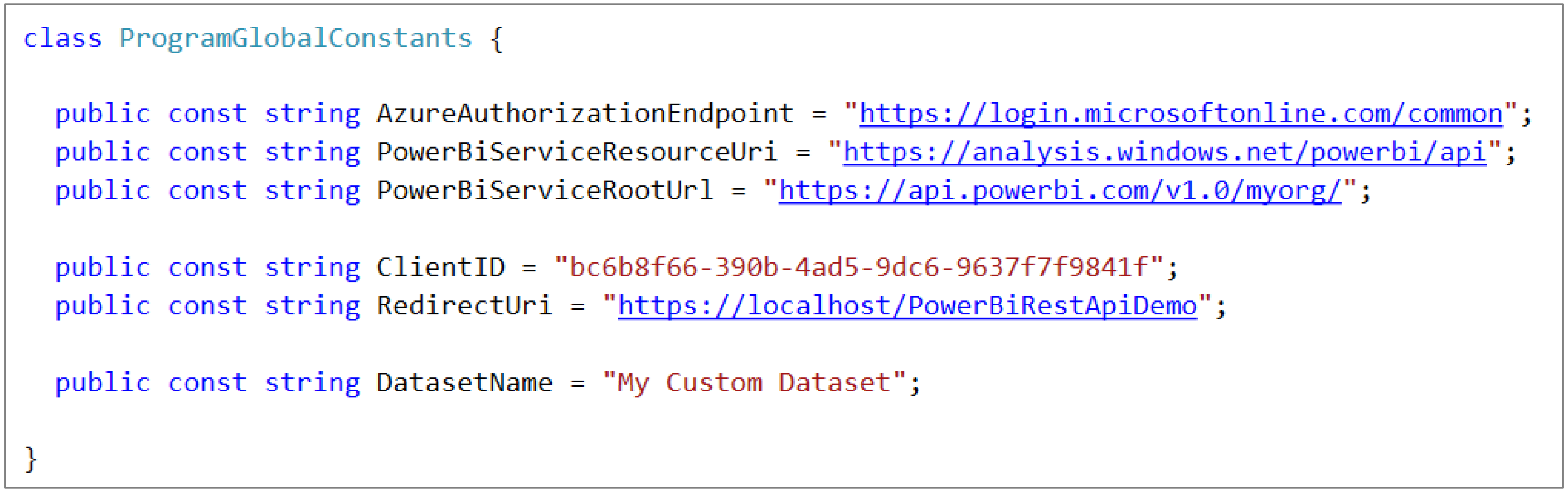
Authenticating with Azure AD

- User must be authenticated against Azure AD
  - User authentication used to obtain access token
  - Can be accomplished with the Azure AD Authentication Library
  - Access token pass to Power BI Service API in call REST calls



Important Application Constants

- Power BI Service API requires Client ID and several URLs
  - These values are tracked as application constants



```
class ProgramGlobalConstants {

  public const string AzureAuthorizationEndpoint = "https://login.microsoftonline.com/common";
  public const string PowerBiServiceResourceUri = "https://analysis.windows.net/powerbi/api";
  public const string PowerBiServiceRootUrl = "https://api.powerbi.com/v1.0/myorg/";

  public const string ClientID = "bc6b8f66-390b-4ad5-9dc6-9637f7f9841f";
  public const string RedirectUri = "https://localhost/PowerBiRestApiDemo";

  public const string DatasetName = "My Custom Dataset";

}
```

Using ADAL to Retrieve an Access Token

```
class PowerBiWorkspaceManager {

  #region "Authentication Details"

  protected string AccessToken = string.Empty;

  protected void GetAccessToken() {

1   // create new authentication context
    var authenticationContext = new AuthenticationContext(ProgramGlobalConstants.AzureAuthorizationEndpoint);

2   // use authentication context to trigger user sign-in and return access token
    var userAuthnResult = authenticationContext.AcquireToken(ProgramGlobalConstants.PowerBiServiceResourceUri,
                                                             ProgramGlobalConstants.ClientID,
                                                             new Uri(ProgramGlobalConstants.RedirectUri),
                                                             PromptBehavior.Auto);
3   // cache access token in AccessToken field
    AccessToken = userAuthnResult.AccessToken;

  }

  #endregion
```

Agenda

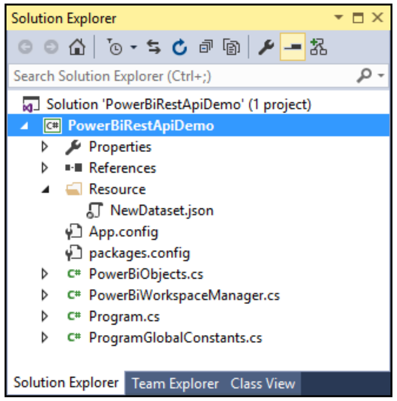
- ✓ Power BI Service API Overview
- ✓ Registering Applications with Azure AD
- ✓ Security Programming with Azure AD
- ➢ Importing PBIX Project Files using Code
- Programming Datasets and Data Sources

Agenda

- ✓ Power BI Service API Overview
- ✓ Registering Applications with Azure AD
- ✓ Security Programming with Azure AD
- ✓ Importing PBIX Project Files using Code
- ➢ Programming Datasets and Data Sources

The PowerBiRestApiDemo Sample Project

- Console application project in Visual Studio 2015



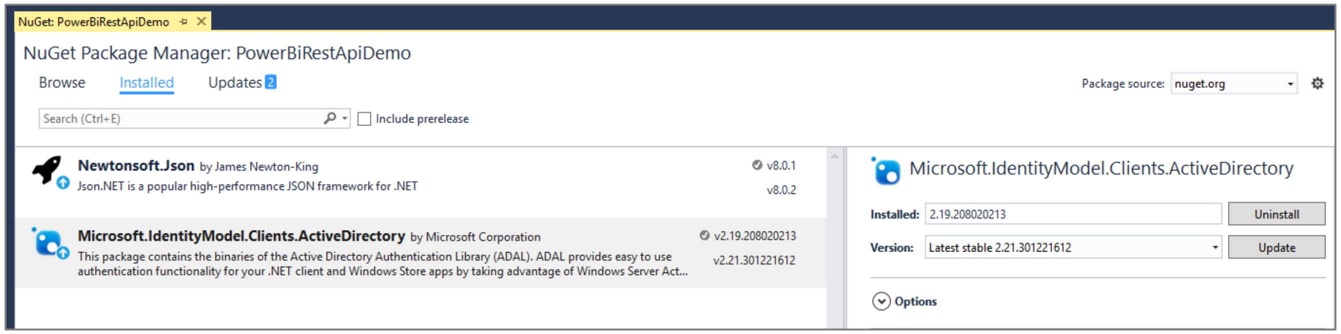
Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'PowerBiRestApiDemo' (1 project)
  - PowerBiRestApiDemo
    - Properties
    - References
    - Resource
      - NewDataset.json
    - App.config
    - packages.config
    - PowerBiObjects.cs
    - PowerBiWorkspaceManager.cs
    - Program.cs
    - ProgramGlobalConstants.cs

Solution Explorer | Team Explorer | Class View

- PowerBiRestApiDemo project contains two NuGet packages
  - Newtonsoft.Json used to convert JSON data to and from and C#
  - Azure ADAL used to get access tokens from Azure Active Directory



NuGet: PowerBiRestApiDemo

NuGet Package Manager: PowerBiRestApiDemo

Browse | Installed | Updates 2

Package source: nuget.org

Search (Ctrl+E) | Include prerelease

Newtonsoft.Json by James Newton-King — v8.0.1 / v8.0.2
Json.NET is a popular high-performance JSON framework for .NET

Microsoft.IdentityModel.Clients.ActiveDirectory by Microsoft Corporation — v2.19.208020213 / v2.21.301221612
This package contains the binaries of the Active Directory Authentication Library (ADAL). ADAL provides easy to use authentication functionality for your .NET client and Windows Store apps by taking advantage of Windows Server Act...

Microsoft.IdentityModel.Clients.ActiveDirectory

Installed: 2.19.208020213 | Uninstall

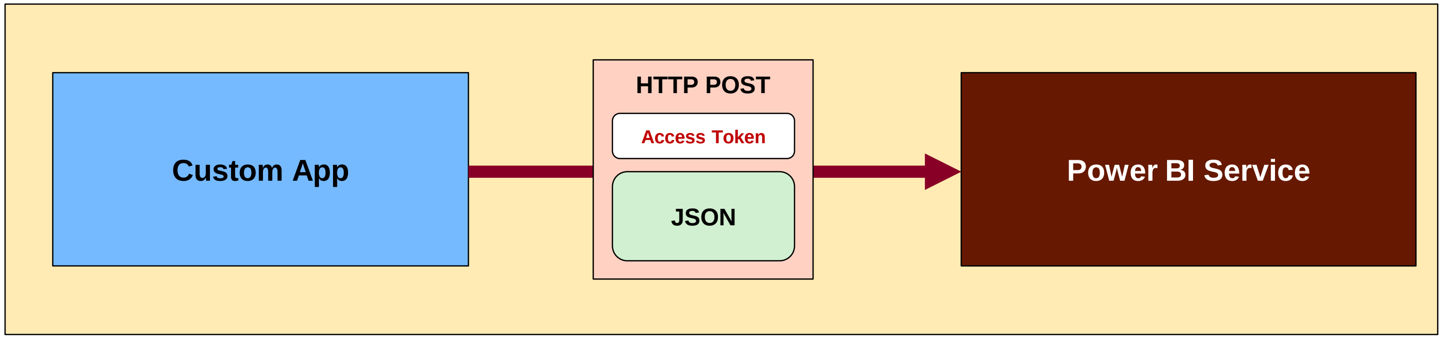
Version: Latest stable 2.21.301221612 | Update

Options

Executing Power BI Service API Calls

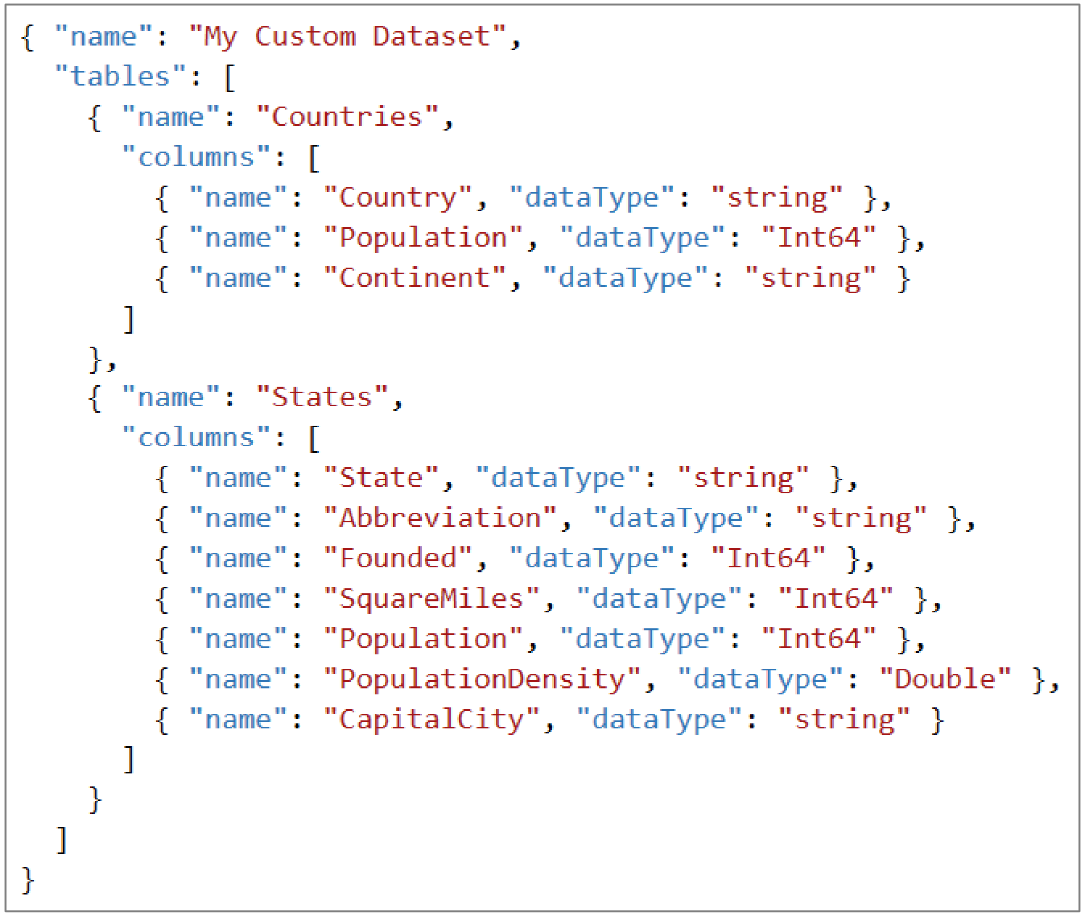
- Generic helper methods designed to execute HTTP operations

```
private void ExecutePostRequest(string restUri, string postBody) {
1 // prepare REST call
  HttpContent body = new StringContent(postBody);
  body.Headers.ContentType = new MediaTypeWithQualityHeaderValue("application/json");
  HttpClient client = new HttpClient();
  client.DefaultRequestHeaders.Add("Accept", "application/json");
  client.DefaultRequestHeaders.Add("Authorization", "Bearer " + AccessToken);

2 // execute REST call
  HttpResponseMessage response = client.PostAsync(restUri, body).Result;
}
```



Using JSON to Create a Custom Dataset

- Dataset created using JSON-formatted Table Schema



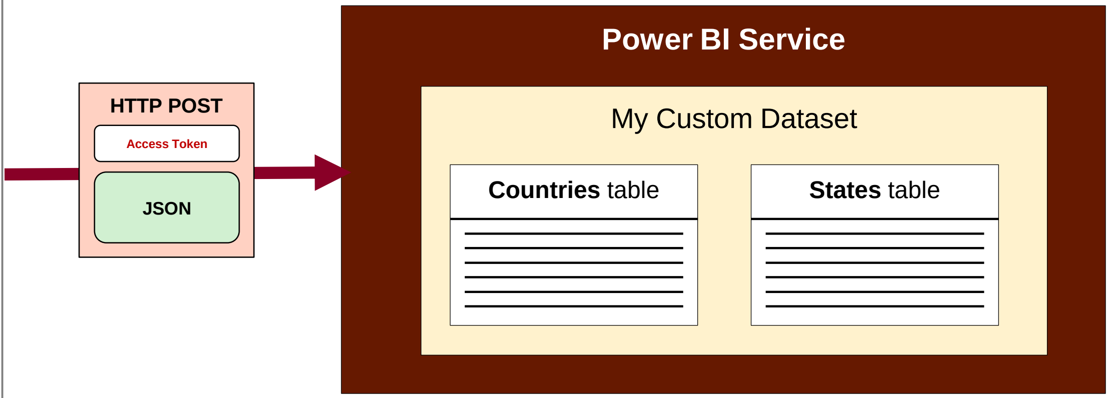
```json
{ "name": "My Custom Dataset",
  "tables": [
    { "name": "Countries",
      "columns": [
        { "name": "Country", "dataType": "string" },
        { "name": "Population", "dataType": "Int64" },
        { "name": "Continent", "dataType": "string" }
      ]
    },
    { "name": "States",
      "columns": [
        { "name": "State", "dataType": "string" },
        { "name": "Abbreviation", "dataType": "string" },
        { "name": "Founded", "dataType": "Int64" },
        { "name": "SquareMiles", "dataType": "Int64" },
        { "name": "Population", "dataType": "Int64" },
        { "name": "PopulationDensity", "dataType": "Double" },
        { "name": "CapitalCity", "dataType": "string" }
      ]
    }
  ]
}
```

Creating a Custom Dataset

- Dataset created by executing HTTP POST operation
  - One-time operation done as application begins running

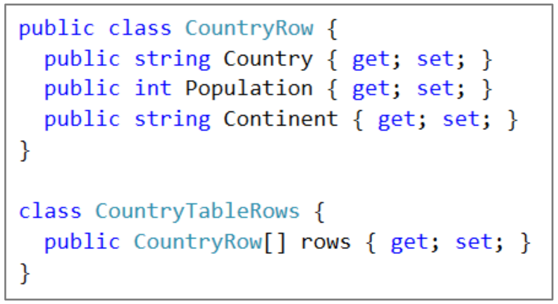
```
// prepare call to create new dataset
1 string restUrlDatasets = ProgramGlobalConstants.PowerBiServiceRootUrl + "datasets";
  string jsonNewDataset = Properties.Resources.NewDataset_json;
// execute REST call to create new dataset
2 string json = ExecutePostRequest(restUrlDatasets, jsonNewDataset);
// retrieve Guid to track dataset ID
  Dataset dataset = JsonConvert.DeserializeObject<Dataset>(json);
3 CustomDatasetId = dataset.id;
```

```
{ "name": "My Custom Dataset",
  "tables": [
    { "name": "Countries",
      "columns": [
        { "name": "Country", "dataType": "string" },
        { "name": "Population", "dataType": "Int64" },
        { "name": "Continent", "dataType": "string" }
      ]
    },
    { "name": "States",
      "columns": [
        { "name": "State", "dataType": "string" },
        { "name": "Abbreviation", "dataType": "string" },
        { "name": "Founded", "dataType": "Int64" },
        { "name": "SquareMiles", "dataType": "Int64" },
        { "name": "Population", "dataType": "Int64" },
        { "name": "PopulationDensity", "dataType": "Double" },
        { "name": "CapitalCity", "dataType": "string" }
      ]
    }
  ]
}
```

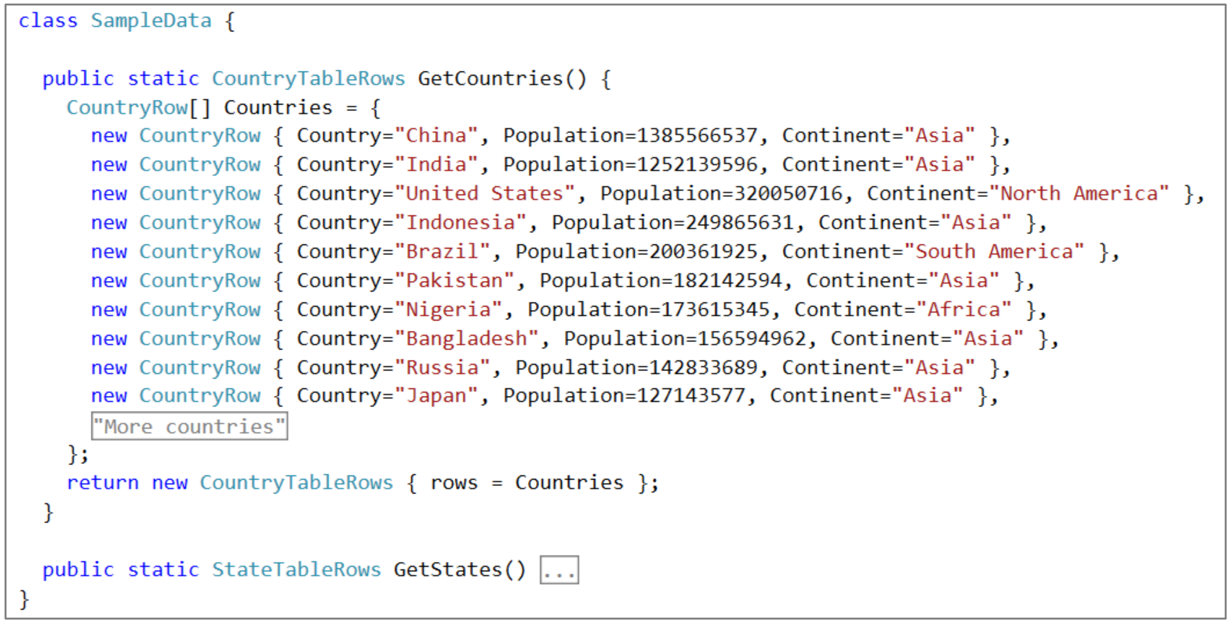


Designing C# Classes to Convert to JSON

- C# class can be created to facilitate reading & writing JSON
  - Newtonsoft.Json package contains classes for performing conversions



```
public class CountryRow {
  public string Country { get; set; }
  public int Population { get; set; }
  public string Continent { get; set; }
}

class CountryTableRows {
  public CountryRow[] rows { get; set; }
}
```



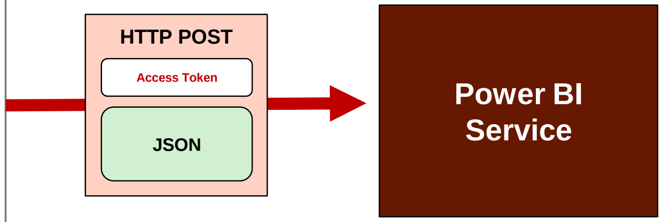
```
class SampleData {

  public static CountryTableRows GetCountries() {
    CountryRow[] Countries = {
      new CountryRow { Country="China", Population=1385566537, Continent="Asia" },
      new CountryRow { Country="India", Population=1252139596, Continent="Asia" },
      new CountryRow { Country="United States", Population=320050716, Continent="North America" },
      new CountryRow { Country="Indonesia", Population=249865631, Continent="Asia" },
      new CountryRow { Country="Brazil", Population=200361925, Continent="South America" },
      new CountryRow { Country="Pakistan", Population=182142594, Continent="Asia" },
      new CountryRow { Country="Nigeria", Population=173615345, Continent="Africa" },
      new CountryRow { Country="Bangladesh", Population=156594962, Continent="Asia" },
      new CountryRow { Country="Russia", Population=142833689, Continent="Asia" },
      new CountryRow { Country="Japan", Population=127143577, Continent="Asia" },
      "More countries"
    };
    return new CountryTableRows { rows = Countries };
  }

  public static StateTableRows GetStates() ...
}
```

Adding Rows to a Table in a Dataset

- Executing POST to add rows to Countries table

```
public void AddCountryRows() {
  string restUrlDatasets = ProgramGlobalConstants.PowerBiServiceRootUrl + "datasets/";
  CountryTableRows countryRows = SampleData.GetCountries();
1 string jsonCountryRows = JsonConvert.SerializeObject(countryRows);
  string restUrlCountryTableRows = string.Format("{0}/{1}/tables/Countries/rows", restUrlDatasets, CustomDatasetId);
2 string json = ExecutePostRequest(restUrlCountryTableRows, jsonCountryRows);
}
```

```
{ "rows": [
    { "Country":"China", "Population":1385566537, "Continent":"Asia" },
    { "Country":"India", "Population":1252139596, "Continent":"Asia" },
    { "Country":"United States", "Population":320050716, "Continent":"North America" },
    { "Country":"Indonesia", "Population":249865631, "Continent":"Asia"},
    { "Country":"Brazil", "Population":200361925, "Continent":"South America"},
    { "Country":"Pakistan", "Population":182142594, "Continent":"Asia"},
    { "Country":"Nigeria", "Population":173615345, "Continent":"Africa"},
    { "Country":"Bangladesh", "Population":156594962, "Continent":"Asia"},
    { "Country":"Russia", "Population":142833689, "Continent":"Asia"},
    { "Country":"Japan", "Population":127143577, "Continent":"Asia"}
  ]
}
```



Countries Table

Country	Population	Continent
China	1385566537	Asia
India	1252139596	Asia
United States	320050716	North America
Indonesia	249865631	Asia
Brazil	200361925	South America
Pakistan	182142594	Asia
Nigeria	173615345	Africa
Bangladesh	156594962	Asia
Russia	142833689	Asia
Japan	127143577	Asia

Summary

- ✓ Power BI Service API Overview
- ✓ Registering Applications with Azure AD
- ✓ Security Programming with Azure AD
- ✓ Importing PBIX Project Files using Code
- ✓ Programming Datasets and Data Sources