

# Microsoft Power BI

# Power BI (Title)

MVP Name / Presenter Name

Title -

#MVP -

---

## Agenda

- 
- Why .NET Core
    - Authenticating with Microsoft.Identity.Web
    - Calling the Power BI Service API
    - Adding Client-side TypeScript Support
    - Creating a View Model for App Workspaces
    - Working with Multi-resource Embed Tokens

# .NET Core

- Decouple from Windows and IIS
- Create applications and web APIs in cross-platform manner
- Break up System.Web.dll monolith
- Applications only load the services they need
- Available .NET Core Hosts
  - Kestrel, IIS, HTTP.sys, Nginx, Apache and Docker

# ASP.NET Core MVC

- **Support for building web UI and web APIs**
  - MVC pattern helps make your web APIs and web apps testable.
  - Razor Pages provide simple page-based programming model
  - Razor markup provides syntax for Razor Pages and MVC views.
  - Tag Helpers enable server-side code to create/render HTML elements
  - Web APIs have built-in support for content negotiation (e.g. json vs xml)
  - Model binding maps HTTP requests to parameterized action methods
  - Model validation automatically performed on client-side and server-side

# Essential .NET Core Topics

- Understanding Dependency injection is import
  - Services are registered on application start up
  - Services injected into classes using parameterized constructors

```
appsettings.json*  +  X
{
  "MySettings": {
    "FavoriteColor": "Red, no Blue"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

```
namespace UserOwnsData {
    public class Startup {

        public IConfiguration Configuration { get; }
        public string FavoriteColor;

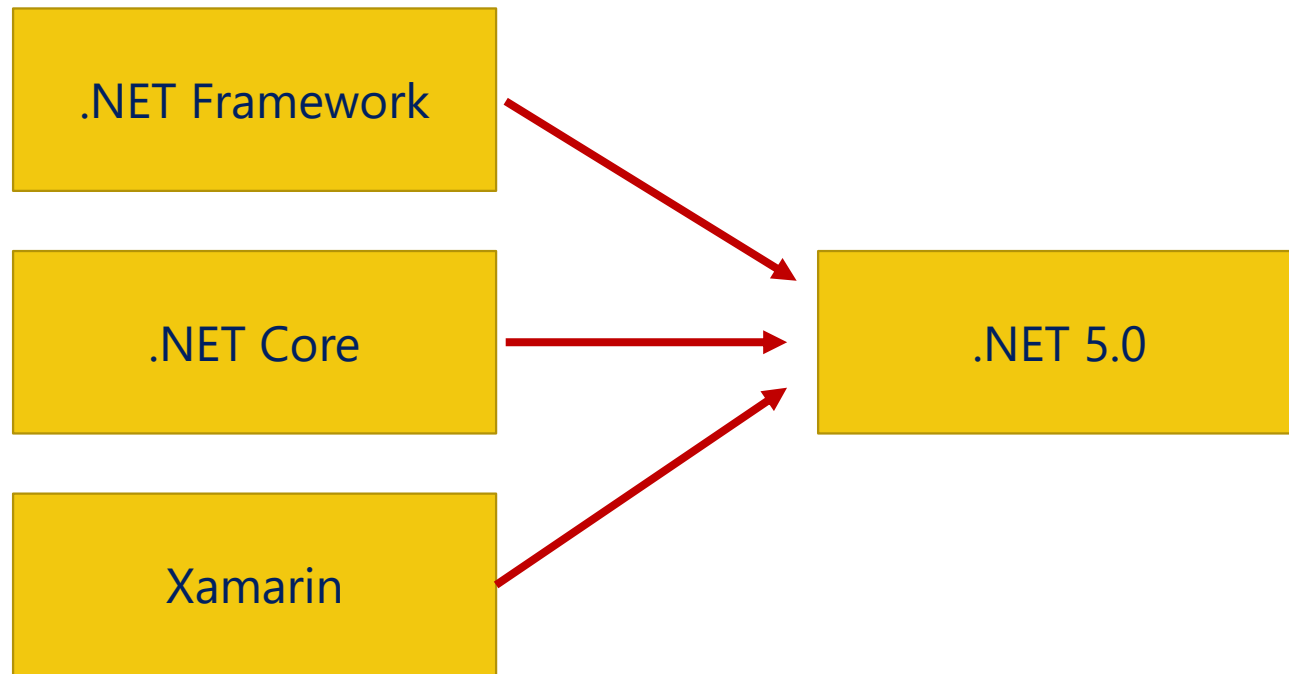
        // pass Configuration service to constructor using dependency injection
        public Startup(IConfiguration configuration) {

            // make Configuration service available to all methods in this class
            Configuration = configuration;

            // use Configuration service to retrieve app setting
            this.FavoriteColor = configuration["MySettings:FavoriteColor"];
        }
    }
}
```

# .NET 5.0

- What is .NET 5?
  - Merging of separate platforms into a single platform



# Install .NET Core SDK


<https://dotnet.microsoft.com/download>

Windows

Linux

macOS

Docker



## .NET Core

### .NET Core 3.1

.NET Core is a cross-platform version of .NET for building websites, services, and console apps.

Run Apps ⓘ


Download .NET Core Runtime

Build Apps ⓘ

Download .NET Core SDK

Advanced ⓘ

All .NET Core downloads...



## .NET Framework

### .NET Framework 4.8

.NET Framework is a Windows-only version of .NET for building any type of app that runs on Windows.

Run Apps ⓘ

Download .NET Framework Runtime

Build Apps ⓘ

Download .NET Framework Dev Pack

Advanced ⓘ

All .NET Framework downloads...



# .NET Core CLI

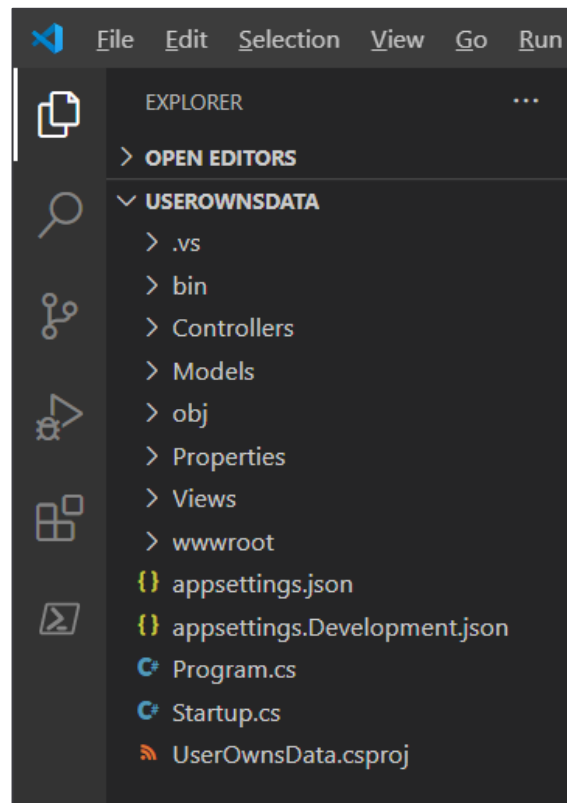
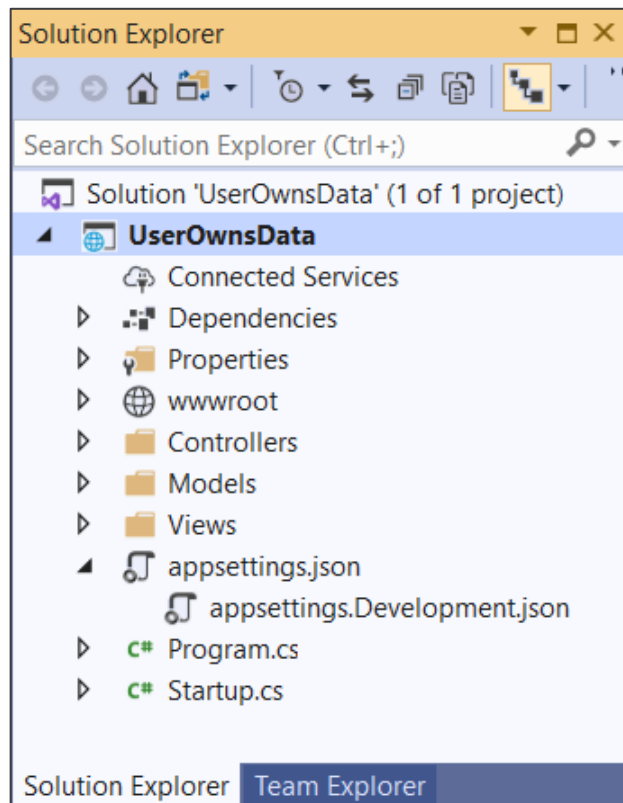
- The .NET Core command-line interface (CLI)
  - Cross-platform toolchain for creating, debugging and publishing applications
  - Create new applications using **dotnet new** command
  - Add NuGet packages using **dotnet add package** command

CreateNetCoreProject.ps1 X

```
dotnet new mvc --auth SingleOrg --framework netcoreapp3.1  
  
dotnet remove package Microsoft.AspNetCore.Authentication.AzureAD.UI  
  
dotnet add package Microsoft.Identity.Web -v 0.2.0-preview  
dotnet add package Microsoft.Identity.Web.UI -v 0.2.0-preview  
dotnet add package Microsoft.PowerBi.Api
```

# Visual Studio 2019 versus Visual Studio Code

- Should you use Visual Studio 2019 versus Visual Studio Code?
  - Yes, either one can be used to develop for .NET Core 3.1 and .NET 5



# Creating Azure AD Application with PowerShell

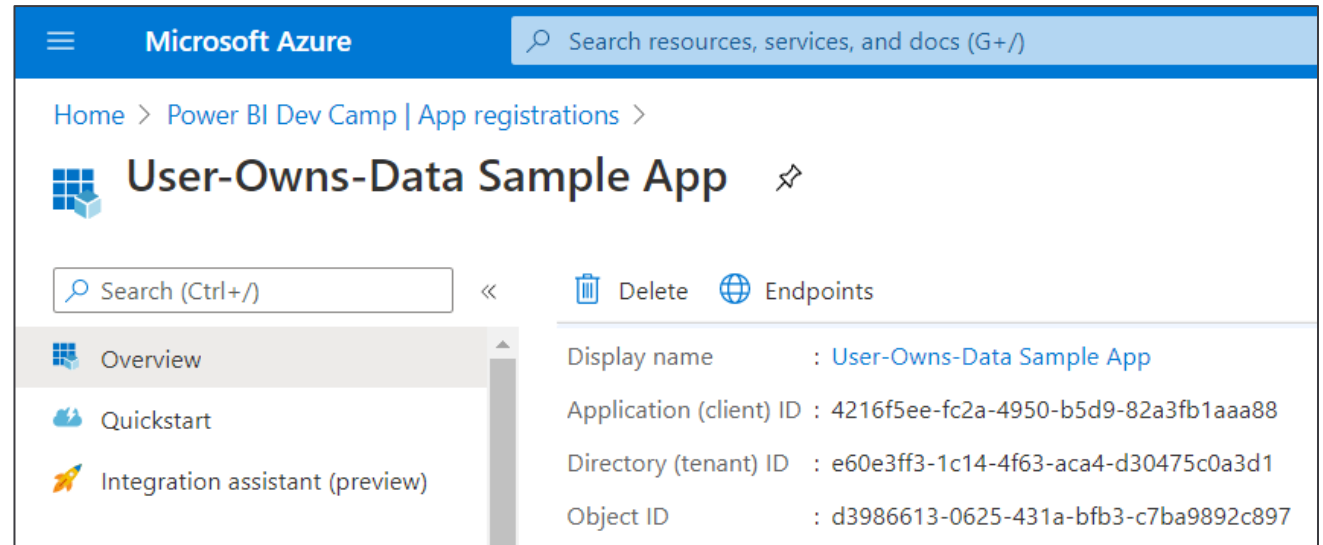
```
$authResult = Connect-AzureAD
$userAccountId = $authResult.Account.Id
$user = Get-AzureADUser -ObjectId $userAccountId

$appDisplayName = "User-Owns-Data Sample App"
$replyUrl = "https://localhost:44300/signin-oidc"

# create app secret
$newGuid = New-Guid
$appSecret = ([System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes(($newGuid))))+"="
$startDate = Get-Date
$passwordCredential = New-Object -TypeName Microsoft.Open.AzureAD.Model.PasswordCredential
$passwordCredential.StartDate = $startDate
$passwordCredential.EndDate = $startDate.AddYears(1)
$passwordCredential.KeyId = $newGuid
$passwordCredential.Value = $appSecret

# create Azure AD Application
$saadApplication = New-AzureADApplication `
    -DisplayName $appDisplayName `
    -PublicClient $false `
    -AvailableToOtherTenants $false `
    -ReplyUrls @($replyUrl) `
    -Homepage $replyUrl `
    -PasswordCredentials $passwordCredential

# assign current user as owner
$appId = $saadApplication.AppId
Add-AzureADApplicationOwner -ObjectId $saadApplication.ObjectId -RefObjectId $user.ObjectId
```





The screenshot shows the Microsoft Azure portal interface. At the top, there's a search bar and navigation links. The main heading is 'User-Owns-Data Sample App'. Below this, there's a search bar and a list of tabs: Overview, Quickstart, and Integration assistant (preview). The 'Overview' tab is selected, showing the app's details: Display name (User-Owns-Data Sample App), Application (client) ID (4216f5ee-fc2a-4950-b5d9-82a3fb1aaa88), Directory (tenant) ID (e60e3ff3-1c14-4f63-aca4-d30475c0a3d1), and Object ID (d3986613-0625-431a-bfb3-c7ba9892c897).

## Client secrets

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

[+ New client secret](#)

Description	Expires	Value	
App Secret	7/25/2021	952eBr.pO4-AZ03huF~f4NP_dDMJzmyGuu	 

---

## Agenda

- 
- ✓ Why .NET Core?
  - Authenticating with Microsoft.Identity.Web
    - Calling the Power BI Service API
    - Adding Client-side TypeScript Support
    - Creating a View Model for App Workspaces
    - Working with Multi-resource Embed Tokens

# Introducing Microsoft.Identity.Web

- **What is Microsoft.Identity.Web**
  - Set of components and classes to assist developers
  - Used to perform authentication in Web applications and Web APIs
- **When to use**
  - In Web application and Web APIs built on .NET Core 3.1 and .NET 5
  - Currently in preview – release schedule synced with .NET 5 (November 2020)

# Sample Code and Tutorial

- Stored in a GitHub repository for easy download
  - <https://github.com/TedPattison/PowerBIEmbedding>

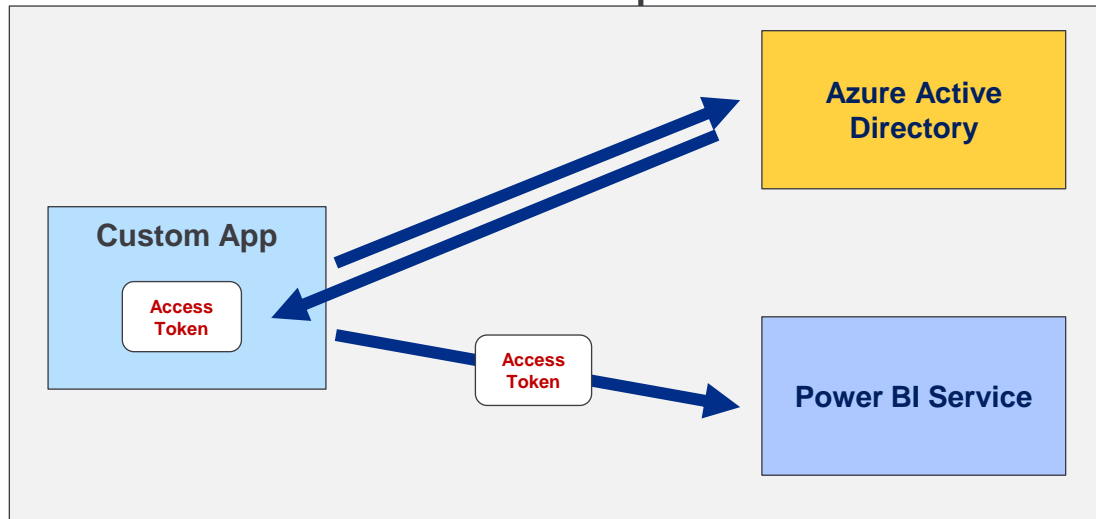
The screenshot shows the GitHub repository page for TedPattison/PowerBIEmbedding. The repository is in the 'master' branch. The file list includes:

File Name	Commit Type	Time Ago
Authentication patterns for Power BI embedding.pdf	Updated	3 minutes ago
05-Developer Onickstart into Power BI Embedding.pdf	Updated	2 minutes ago
01-Developing with Power BI Embedding.pdf	Updated	2 minutes ago
.gitignore	Initial commit	last month
2chqrz	Updated	55 days ago
pbix	Updated	2 minutes ago
rap	Updated	last month
Demos	Updated	13 minutes ago

Below the file list, there are buttons for 'Branch: master', 'New pull request', 'Find file', and 'Clone or download'. A summary bar shows 9 commits, 1 branch, 0 packages, 0 releases, 1 contributor, and MIT license. The repository description is 'Developer resources for learning to develop software with Power BI embedding.' At the bottom, there are buttons for 'Watch', 'Star' (0), and 'Fork' (0).

# Authenticating with Azure AD

- Custom applications must authenticate with Azure AD
  - Your code implements and authentication flow to obtain access token
  - Access token must be passed when calling Power BI Service API



- Microsoft supports two endpoints for programming authentication
  - Azure AD V1 endpoint (released to GA over 8 years ago)
  - Azure AD V2 endpoint (released to GA in May 2019)

# Microsoft Authentication Libraries

- XXX




**Microsoft.IdentityModel.Clients.ActiveDirectory**  by Microsoft, **30M** downloads

v5.0.5

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...



**Microsoft.Identity.Client** by Microsoft

 v4.0.0

This package contains the binaries of the Microsoft Authentication Library for .NET (MSAL.NET). MSAL.NET makes it easy to obtain tokens from the Microsoft identity platform for developers (formally Az...



Prerelease

**Microsoft.Identity.Web** by Microsoft

v0.2.1-preview

This package enables ASP.NET Core Web apps and Web APIs to use the Microsoft identity platform (formerly Azure AD v2.0).



Prerelease

**Microsoft.Identity.Web.UI** by Microsoft

v0.2.1-preview

This package enables UI for ASP.NET Core Web apps that use Microsoft.Identity.Web.



---

## Agenda

- 
- ✓ Why .NET Core?
  - ✓ Authenticating with Microsoft.Identity.Web
  - Calling the Power BI Service API
    - Adding Client-side TypeScript Support
    - Creating a View Model for App Workspaces
    - Working with Multi-resource Embed Tokens

# Power BI Service API Scopes

- Azure AD V2 endpoint requires passing scopes
  - Scopes define permissions required in access token
  - Scopes defined as resource + permission

```
https://analysis.windows.net/powerbi/api/ + Report.ReadWrite.All  
static string[] scopesDefault = new string[] {  
    "https://analysis.windows.net/powerbi/api/.default"  
};  
  
static string[] scopesReadWorkspaceAssets = new string[] {  
    "https://analysis.windows.net/powerbi/api/Dashboard.Read.All",  
    "https://analysis.windows.net/powerbi/api/Dataset.Read.All",  
    "https://analysis.windows.net/powerbi/api/Report.Read.All"  
};  
  
static string[] scopesReadUserApps = new string[] {  
    "https://analysis.windows.net/powerbi/api/App.Read.All"  
};  
  
static string[] scopesManageWorkspaceAssets = new string[] {  
    "https://analysis.windows.net/powerbi/api/Content.Create",  
    "https://analysis.windows.net/powerbi/api/Dashboard.ReadWrite.All",  
    "https://analysis.windows.net/powerbi/api/Dataset.ReadWrite.All",  
    "https://analysis.windows.net/powerbi/api/Group.Read.All",  
    "https://analysis.windows.net/powerbi/api/Report.ReadWrite.All",  
    "https://analysis.windows.net/powerbi/api/Workspace.ReadWrite.All"  
};
```

---

## Agenda

- 
- ✓ Why .NET Core?
  - ✓ Authenticating with Microsoft.Identity.Web
  - ✓ Calling the Power BI Service API
  - Adding Client-side TypeScript Support
    - Creating a View Model for App Workspaces
    - Working with Multi-resource Embed Tokens

---

## Agenda

- 
- ✓ Why .NET Core?
  - ✓ Authenticating with Microsoft.Identity.Web
  - ✓ Calling the Power BI Service API
  - ✓ Adding Client-side TypeScript Support
  - Creating a View Model for App Workspaces
    - Working with Multi-resource Embed Tokens

---

## Agenda

- 
- ✓ Why .NET Core?
  - ✓ Authenticating with Microsoft.Identity.Web
  - ✓ Calling the Power BI Service API
  - ✓ Adding Client-side TypeScript Support
  - ✓ Creating a View Model for App Workspaces
  - Working with Multi-resource Embed Tokens

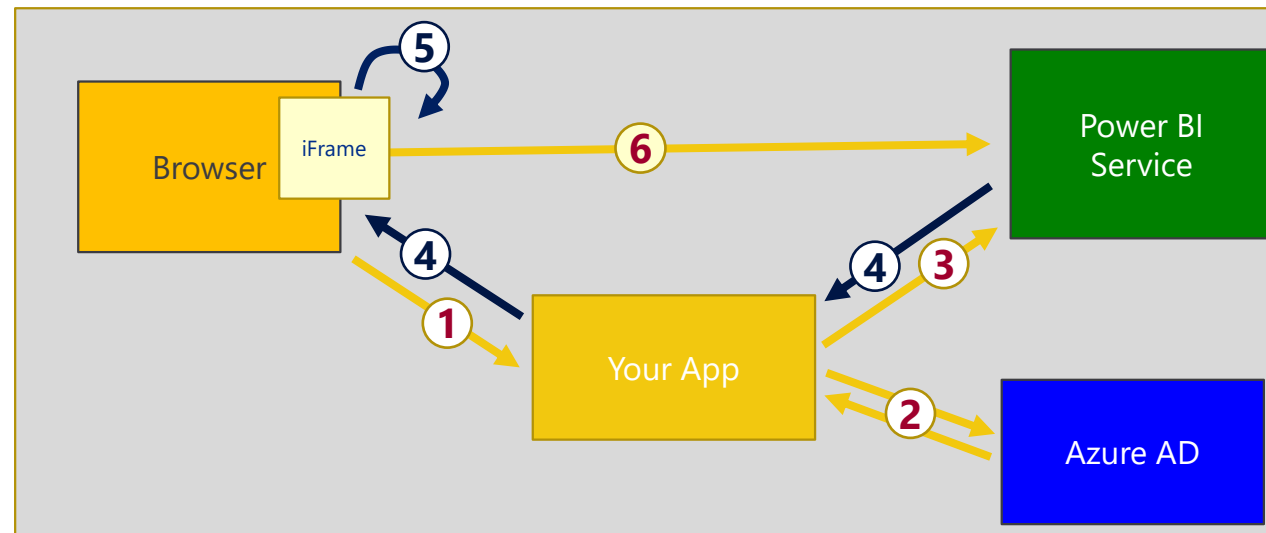
---

## Summary

- ✓ Why .NET Core?
- ✓ Authenticating with Microsoft.Identity.Web
- ✓ Calling the Power BI Service API
- ✓ Adding Client-side TypeScript Support
- ✓ Creating a View Model for App Workspaces
- ✓ Working with Multi-resource Embed Tokens

# Power BI Embedding – The Big Picture

- User launches your app using a browser
- App authenticates with Azure Active Directory and obtains access token
- App uses access token to call to Power BI Service API
- App retrieves data for embedded resource and passes it to browser.
- Client-side code uses Power BI JavaScript API to create embedded resource
- Embedded resource session created between browser and Power BI service



# Choosing the Correct Embedding Model

## User-Owns-Data Embedding

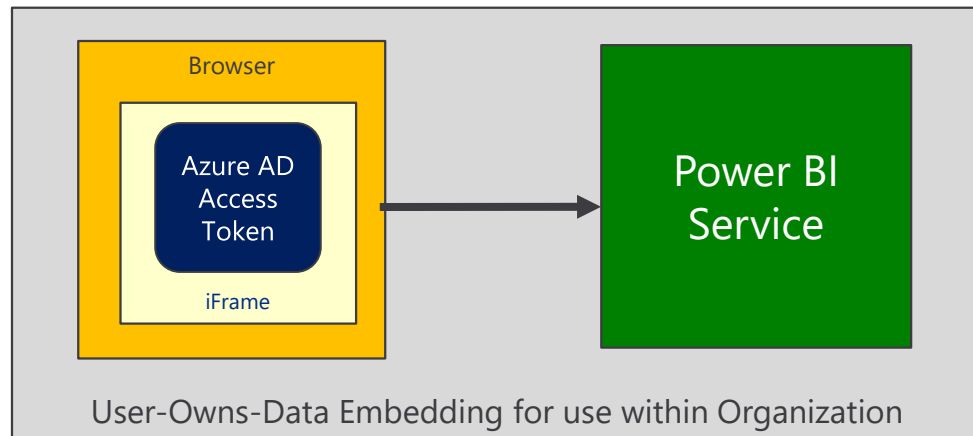
All users require a Power BI license

Useful in corporate environments

App authenticates as current user

Your code runs with user's permissions

User's access token passed to browser



## App-Owns-Data Embedding

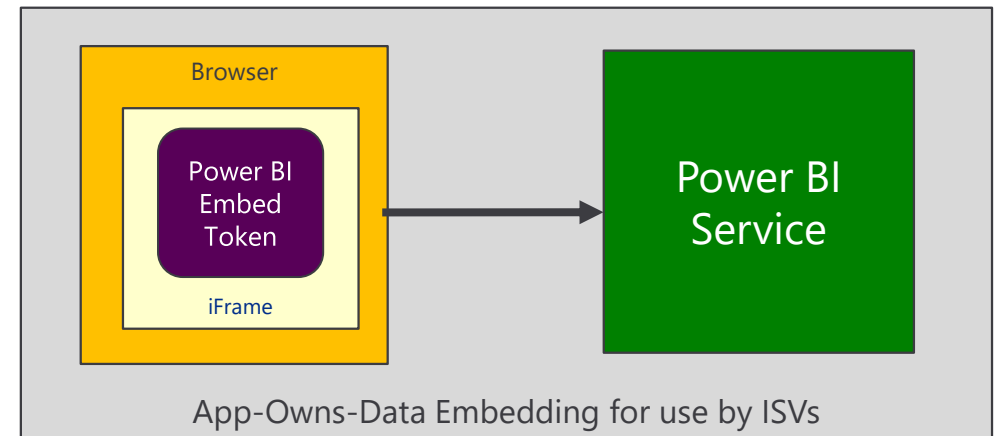
No users require Power BI license

Useful in commercial applications

App authenticates with app-only identity

Your code runs with admin permissions

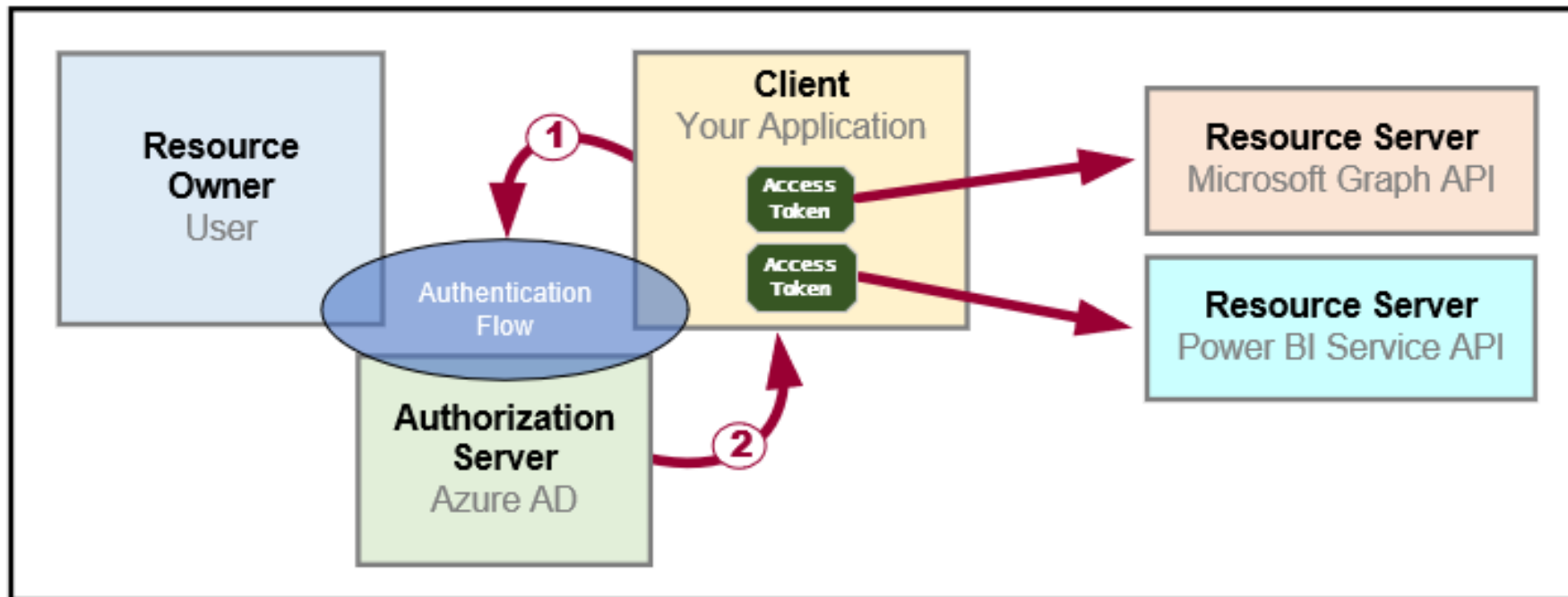
Embed token passed to browser





# OAuth 2.0 Fundamentals

- Client application calls to resource server on behalf of a user
  - Client application implements **authentication flow** to acquire access token
  - Access token contains permission grants for client to call resource server
  - Client passes access token in Authorization request header when calling to resource server
- on has proper



# Access Tokens

- **Access token is a bearer token**
  - Access token can be used by anyone who bears it (e.g. stolen)
  - Access tokens should always be passed over HTTPS using the Authorization header
  - Access token expires after an hour
- **There are two types of access token**
  - User tokens
  - App-only tokens

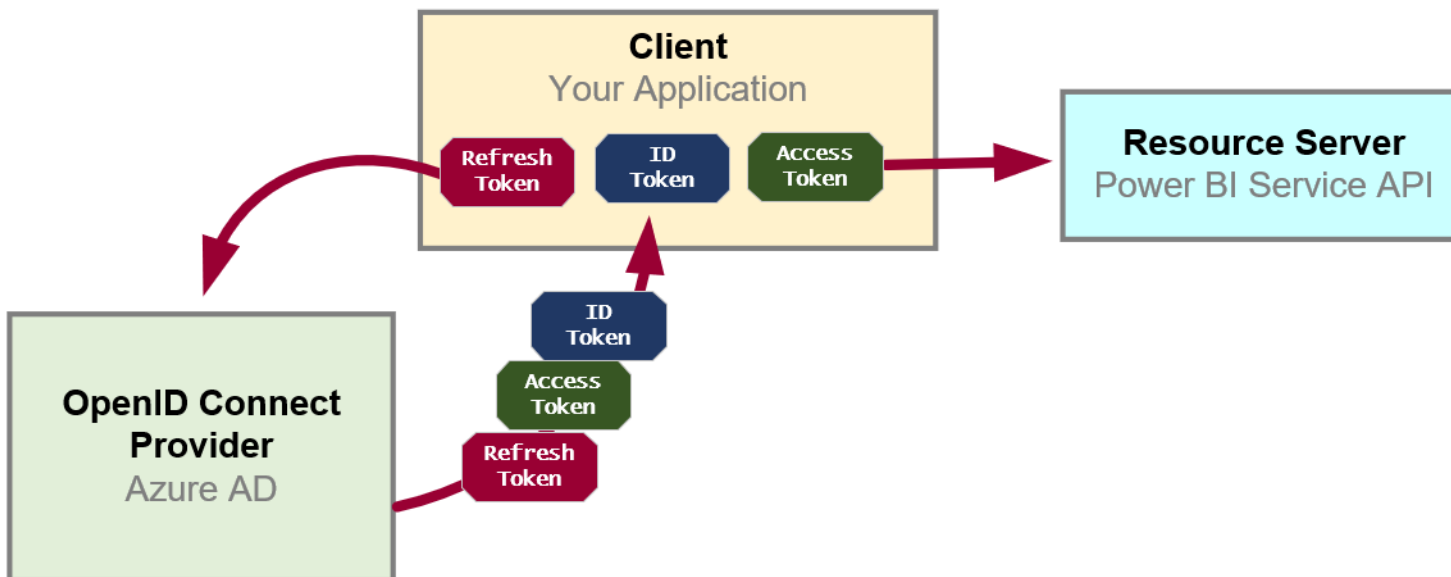
```
{
  "aud": "https://analysis.windows.net/powerbi/api",
  "iss": "https://sts.windows.net/16f9b31b-f2df-4c69-9c0d-7855c7c7f546",
  "iat": 1587579545,
  "nbf": 1587579545,
  "exp": 1587583445,
  "acct": 0,
  "acr": "1",
  "aio": "ATQAY/8PAAAAuKi4ZmLpFIj0Q/wiQakzBrSxxh72y7un",
  "amr": [ "pwd" ],
  "appid": "02aebffb-4d99-4541-b13e-c584f758b207",
  "appidacr": "1",
  "family_name": "Pattison",
  "given_name": "Ted",
  "ipaddr": "47.200.121.31",
  "name": "Ted Pattison",
  "oid": "bacec5c6-c519-49a9-bebe-22ead20fb6b7",
  "puid": "10032000AFDB80c5",
  "scp": "Dashboard.Read.All Dataflow.Read.All Dataset.Read.All",
  "sub": "hqc9o0FJmH9q9gzkoMOSGWhd18YILV6VP9LWxAHGRU",
  "tid": "16f9b31b-f2df-4c69-9c0d-7855c7c7f546",
  "unique_name": "tedp@pbi0413.onmicrosoft.com",
  "upn": "tedp@pbi0413.onmicrosoft.com",
  "uti": "z1TX1kJGpEuEF79UPcUOAA",
  "ver": "1.0",
  "wids": [ "62e90394-69f5-4237-9190-012177145e10" ]
}
```

# Refresh Tokens

- **Access tokens expiration after an hour**
  - How do you get new access tokens without requiring the user to sign in?
- **Refresh tokens used to manage access token expiration**
  - Authorization server passes refresh tokens to client application along with access token
  - Refresh token has lifetime of 14 days by default (90 days max)
  - Refresh token acts as a credential used to obtain new access token without user interaction
  - Refresh tokens often cached in browser storage or in backend database

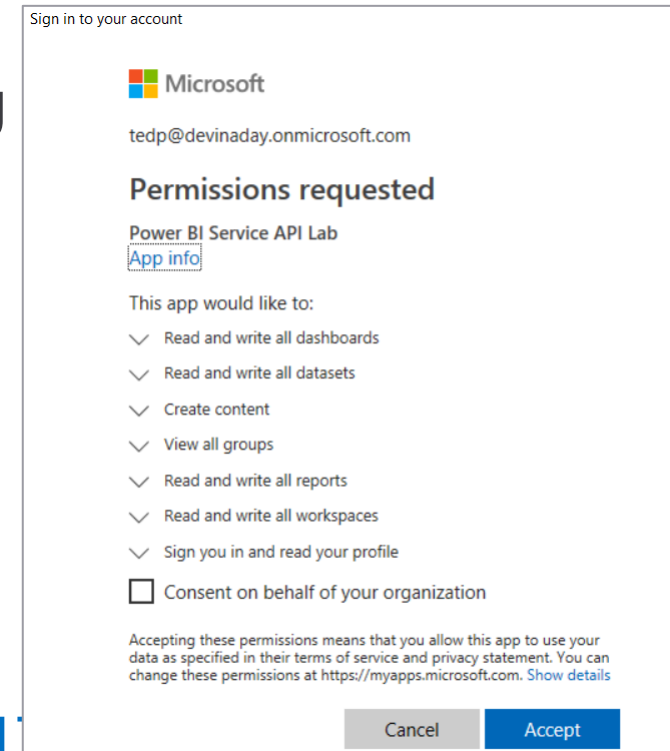
# Open ID Connect and ID Tokens

- **OAuth 2.0 has shortcomings with authentication & identity**
  - It does not provide client with means to validate access tokens
  - Lack of validation makes client vulnerable to token forgery attacks
- **Open ID Connect is standard which extends OAuth 2.0**
  - OpenID Connect provider passes ID token in addition to OAuth 2.0 tokens
  - OpenID Connect provider provides client with keys for token validation



# Delegated Permissions and Scopes

- **Client application requires delegated permissions**
  - Client application needs delegated permissions to make API calls on behalf of user
  - But first, user must consent to delegated permissions requested by client application
  - Client application indicates what permissions it needs using
- **Each delegated permission has ID known as scope**
  - Scope name usually begins with name of hosting resource
  - Resource ID for the Power BI Service API  
`https://analysis.windows.net/powerbi/api`
  - Examples of scopes supported by the Power BI Service API  
`https://analysis.windows.net/powerbi/api/Dashboard.Read.All`  
`https://analysis.windows.net/powerbi/api/Dataset.Read.All`



# Public Clients versus Confidential Web Clients

- **OAuth 2.0 defines two different types of client applications**
  - Public clients
  - Confidential web clients
- **Public clients used for desktop and native applications**
  - Cannot keep a secret - entire application is deployed to client device
  - Does not run from verifiable endpoint on the Internet
- **Confidential web clients used for web applications and service**
  - Web application can track application secret on server-side with code or configuration
  - Runs from verifiable endpoint on the Internet

# OAuth 2.0 Client Registration

- Client must be registered with authorization server
  - Authorization server tracks each client application with unique Client ID
  - Client can be configured with Reply URLs (aka redirect URI)
  - Reply URL used to transmit security tokens to clients
  - Client registration can track other attributes (e.g. credentials & default

Authorization Server				
Azure AD				
Registered Applications				
Name	App ID	Permissions	Reply URL	Credentials
App1	guid1	...	none	none
App2	guid2	...	...	secret key
App3	guid3	...	...	X.509 Certificate

# Authentication Flows

- **Public/native clients**
  - Interactive Flow  
Used in public clients to obtain access token interactively
  - User Password Credential Flow  
Used in Native clients to obtain access code  
Requires passing user name and password across network
- **Confidential Web clients**
  - Client Credentials Flow  
Authentication based on password or certificate held by application  
Used to obtain app-only access tokens
  - Implicit Flow  
Used in SPAs built with JavaScript and AngularJS  
Application obtains access token w/o acquiring authorization code
  - Authorization Code Flow  
Client first obtains authorization code sent back to browser  
Client then obtains access token in server-to-server call



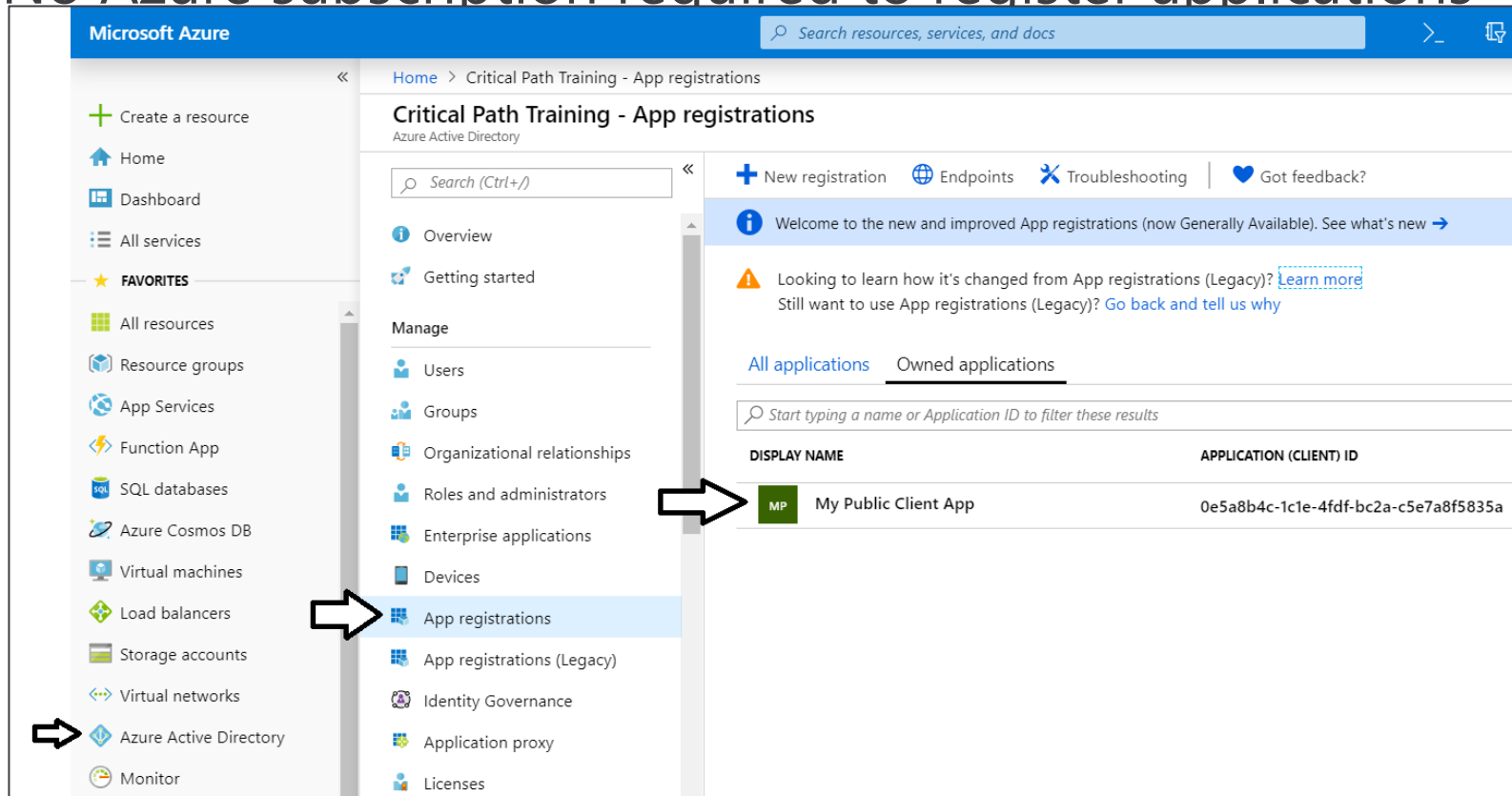
---

## Agenda

- 
- ✓ OAuth 2.0 and OpenID Connect
  - Microsoft Identity Platform 2.0
  - Developing with the Power BI .NET SDK
  - Developing with the App-Owns-Data Model
  - Developing Single Page Applications (SPAs)
  - Developing Secure Web Applications

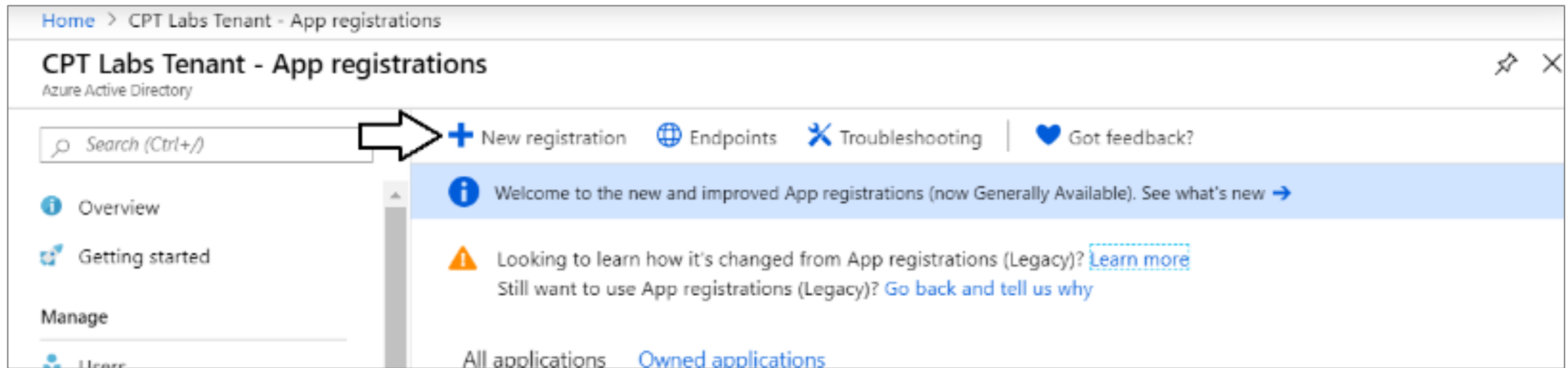
# The Azure Portal

- Azure portal allows you to register Azure AD applications
  - Azure Portal accessible at <https://portal.azure.com>
  - No Azure subscription required to register applications



# Azure AD Applications

- Creating applications required for AAU authentication
  - Applications are created as Public client applications or Web Applications
  - Application identified using GUID known as application ID
  - Application ID often referred to as client ID or app ID



# Azure AD Application Types

- **Public client (mobile and desktop)**
  - Used to register public/native clients
- **Web**
  - Used to register confidential web clients

## Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Web



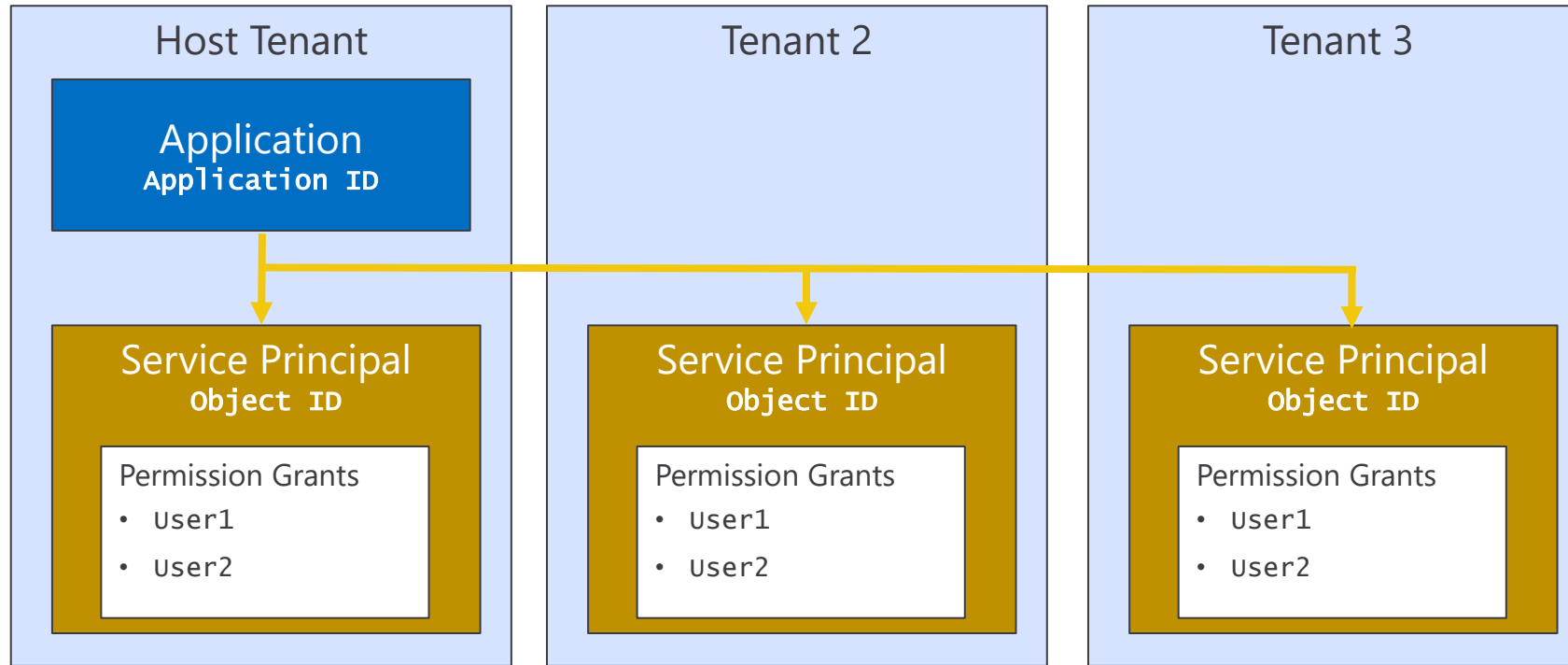
e.g. `https://myapp.com/auth`

Public client (mobile & desktop)

Web

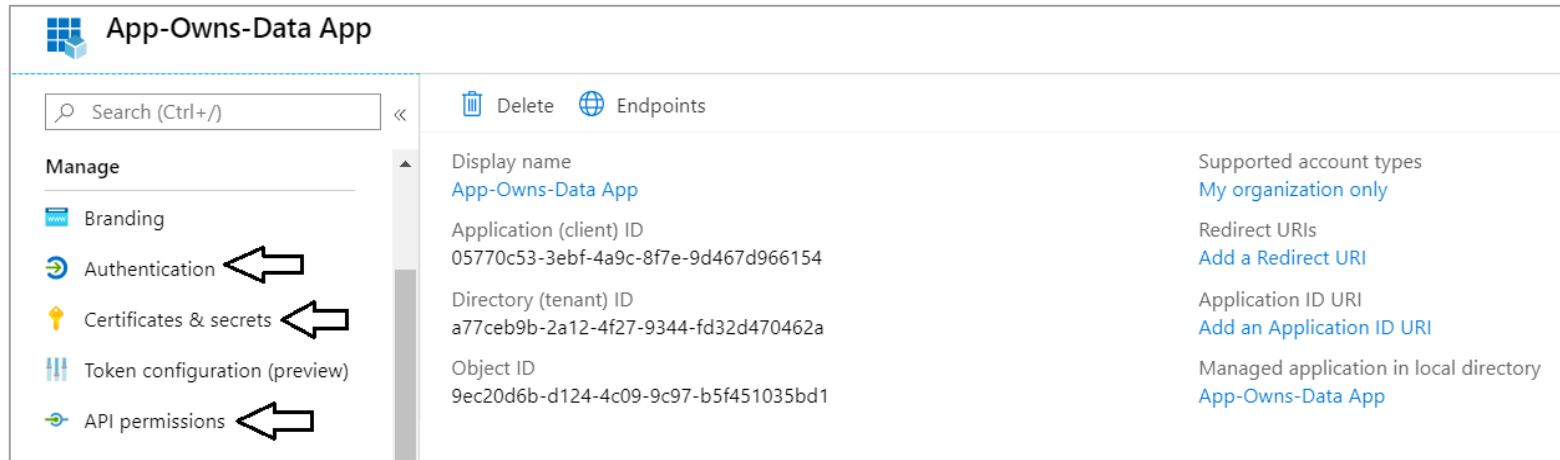
# Azure AD Applications versus Service Principals

- Azure AD creates service principal(s) for each application
  - Service principal created once per tenant
  - Service principal acts as first-class AAD security principal

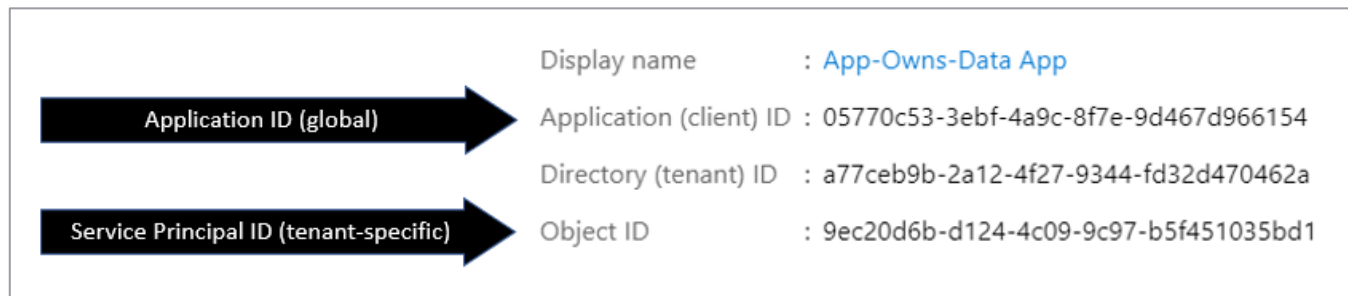


# Configuring Azure AD Applications

- Azure Portal provides UI to create and configure applications



- New application generated with Application ID and Object ID



# Configuring Required Permissions

- Application configured with permissions
  - Default permissions allows user authentication – but that's it
  - To use APIs, you can assign permissions to the application
  - This was required when using Azure AD v1 endpoint and ADAL
  - This is now optional when using Azure AD v2 endpoint and MSAL

Power BI Service API Lab - API permissions

«

Overview

Quickstart

Manage

Branding

Authentication

Certificates & secrets

API permissions

Expose an API

### API permissions

Applications are authorized to use APIs by requesting permissions. These permissions show up during the consent process where users are given the opportunity to grant/deny access.

[+ Add a permission](#)

API / PERMISSIONS NAME	TYPE	DESCRIPTION	ADMIN CONSENT REQUIRED
▼ Microsoft Graph (1)			
User.Read	Delegated	Sign in and read user profile	-

These are the permissions that this application requests statically. You may also request user consent-able permissions dynamically through code. [See best practices for requesting permissions](#)

# Registering Azure AD Apps with PowerShell

```
$authResult = Connect-AzureAD

# display name for new public client app
$appDisplayName = "My Power BI Service App"

# get user account ID for logged in user
$user = Get-AzureADUser -ObjectId $authResult.Account.Id

# get tenant name of logged in user
$tenantName = $authResult.TenantDomain

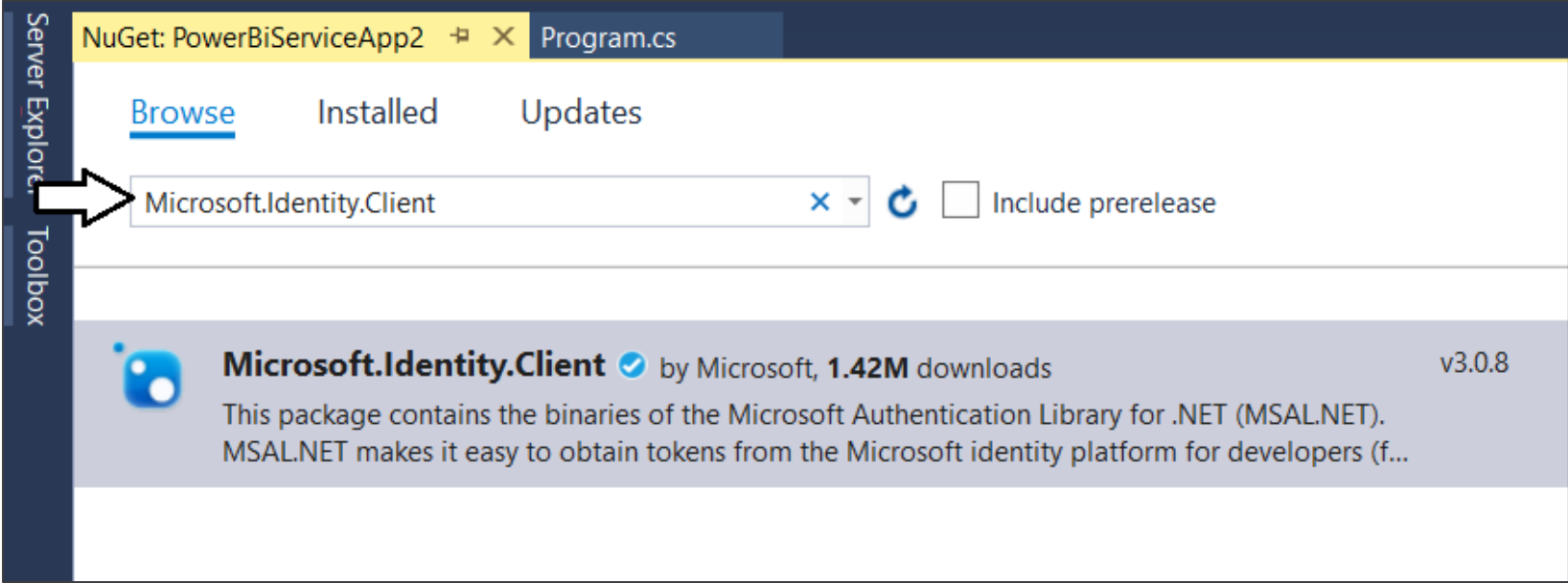
# create Azure AD Application
$replyUrl = "https://localhost/app1234"
$aadApplication = New-AzureADApplication `
    -DisplayName $appDisplayName `
    -PublicClient $true `
    -AvailableToOtherTenants $false `
    -ReplyUrls @($replyUrl)

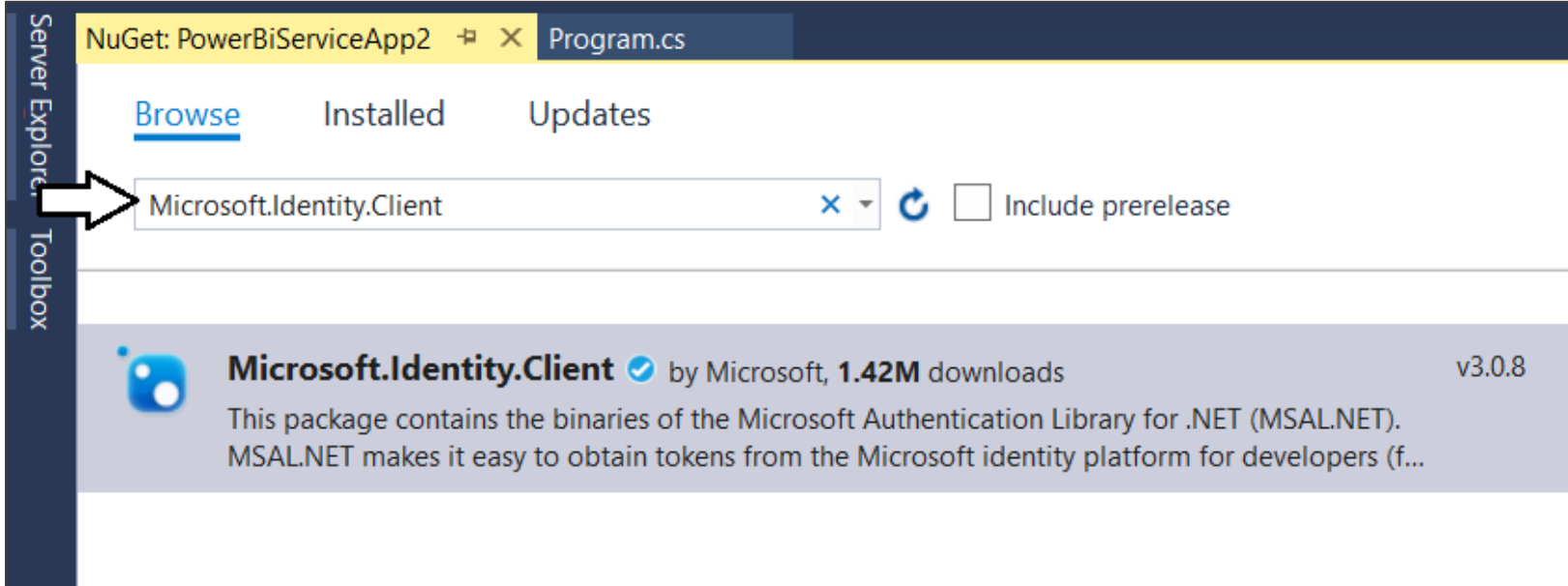
# create service principal for application
$appId = $aadApplication.AppId
$serviceServicePrincipal = New-AzureADServicePrincipal -AppId $appId

# assign current user as application owner
Add-AzureADApplicationOwner -ObjectId $aadApplication.ObjectId -RefObjectId $user.ObjectId
```



# Microsoft Authentication Library (MSAL .NET)

- Developing with the Microsoft Authentication Library for .NET
  - Provides assistance implementing authentication flows with Azure AD v2 Endpoint
  - Added to project as `Microsoft.Identity.Client` NuGet package
  - Provides different builder classes for *public clients* vs *confidential web clients*
-  on acquisition



# Interactive Access Token Acquisition

- Flow implemented using **PublicClientApplication** object
  - Created using **PublicClientApplicationBuilder** object
  - Requires passing redirect URI that matches redirect URI that is registered
  - You can control prompting behavior

```
static string GetAccessTokenInteractive(string[] scopes) {  
    var appPublic = PublicClientApplicationBuilder.Create(clientId)  
        .WithAuthority(tenantCommonAuthority)  
        .WithRedirectUri(redirectUri)  
        .Build();  
  
    var authResult = appPublic.AcquireTokenInteractive(scopes)  
        .WithPrompt(Prompt.SelectAccount)  
        .ExecuteAsync().Result;  
  
    return authResult.AccessToken;  
}
```

# User Credential Password Flow

- MSAL supports user credential password flow
  - Authentication performed by passing user password across the network
  - Microsoft recommends against using this flow in production scenarios

```
static string GetAccessTokenWithUserPassword(string[] scopes) {  
    var appPublic = PublicClientApplicationBuilder.Create(clientId)  
        .WithAuthority(tenantCommonAuthority)  
        .Build();  
  
    string username = "chuckster@devinaday2019.onMicrosoft.com";  
    string userPassword = "myCAT$rightLEG";  
    SecureString userPasswordSecure = new SecureString();  
    foreach (char c in userPassword) {  
        userPasswordSecure.AppendChar(c);  
    }  
  
    var authResult = appPublic.AcquireTokenByUsernamePassword(scopes, username, userPasswordSecure)  
        .ExecuteAsync().Result;  
  
    return authResult.AccessToken;  
}
```

- Create a Token Cache Helper class

```
// create new public client application
var appPublic = PublicClientApplicationBuilder.Create(applicationId)
    .WithAuthority(tenantCommonAuthority)
    .WithRedirectUri(redirectUri)
    .Build();

// connect application to token cache
TokenCacheHelper.EnableSerialization(appPublic.UserTokenCache);
```

[illegible]

# Acquiring Access Token from the Token Cache

```
public static string GetAccessToken(string[] scopes) {  
    // create new public client application  
    var appPublic = PublicClientApplicationBuilder.Create(applicationId)  
        .WithAuthority(tenantCommonAuthority)  
        .WithRedirectUri(redirectUri)  
        .Build();  
  
    // connect application to token cache  
    TokenCacheHelper.EnableSerialization(appPublic.UserTokenCache);  
  
    AuthenticationResult authResult;  
    try {  
        // try to acquire token from token cache  
        var user = appPublic.GetAccountsAsync().Result.FirstOrDefault();  
        authResult = appPublic.AcquireTokenSilent(scopes, user).ExecuteAsync().Result;  
    }  
    catch {  
        try {  
            // try to acquire token with non-interactive User Password Credential Flow  
            SecureString userPasswordSecure = new System.Security.SecureString();  
            foreach (char c in userPassword) {  
                userPasswordSecure.AppendChar(c);  
            }  
            authResult = appPublic.AcquireTokenByUsernamePassword(scopes, userName, userPasswordSecure).ExecuteAsync().Result;  
        }  
        catch {  
            // try to acquire token with interactive flow  
            authResult = appPublic.AcquireTokenInteractive(scopes).ExecuteAsync().Result;  
        }  
    }  
  
    // return access token to caller  
    return authResult.AccessToken;  
}
```

---

## Agenda

- 
- ✓ OAuth 2.0 and OpenID Connect
  - ✓ Microsoft Identity Platform 2.0
  - Developing with the Power BI .NET SDK
    - Developing with the App-Owns-Data Model
    - Developing Single Page Applications (SPAs)
    - Developing Secure Web Applications

# What Is the Power BI Service API?

- **What is the Power BI Service API?**
  - API built on OAuth2, OpenID Connect, REST and ODATA
  - API secured by Azure Active Directory (AAD)
  - API to program with workspaces, datasets, reports & dashboards
  - API also often called "Power BI REST API"
- **What can you do with the Power BI Service API?**
  - Publish PBIX project files
  - Update connection details and datasource credentials
  - Create workspaces and clone content across workspaces
  - Embed Power BI reports and dashboards tiles in web pages
  - Create streaming datasets in order to build real-time dashboards

# Calling the Power BI Service API

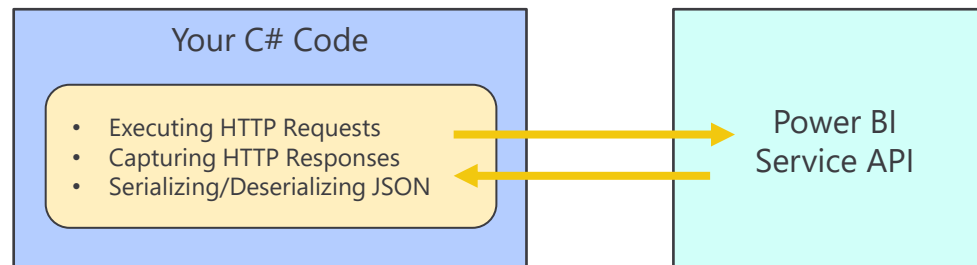
Direct REST calls without using the Power BI .NET SDK

```
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 occurred calling the Power BI Service API");
    }
    return response.Content.ReadAsStringAsync().Result;
}

static void Main() {
    // get report data from app workspace
    string restUrl = "https://api.powerbi.com/v1.0/myorg/groups/" + appWorkspaceId + "/reports/";
    var json = ExecuteGetRequest(restUrl);
    ReportCollection reports = JsonConvert.DeserializeObject<ReportCollection>(json);
    foreach (Report report in reports.value) {
        Console.WriteLine("Report Name: " + report.name);
        Console.WriteLine();
    }
}
```

```
public class Report {
    public string id { get; set; }
    public string name { get; set; }
    public string webUrl { get; set; }
    public string embedUrl { get; set; }
    public bool isOwnedByMe { get; set; }
    public string datasetId { get; set; }
}

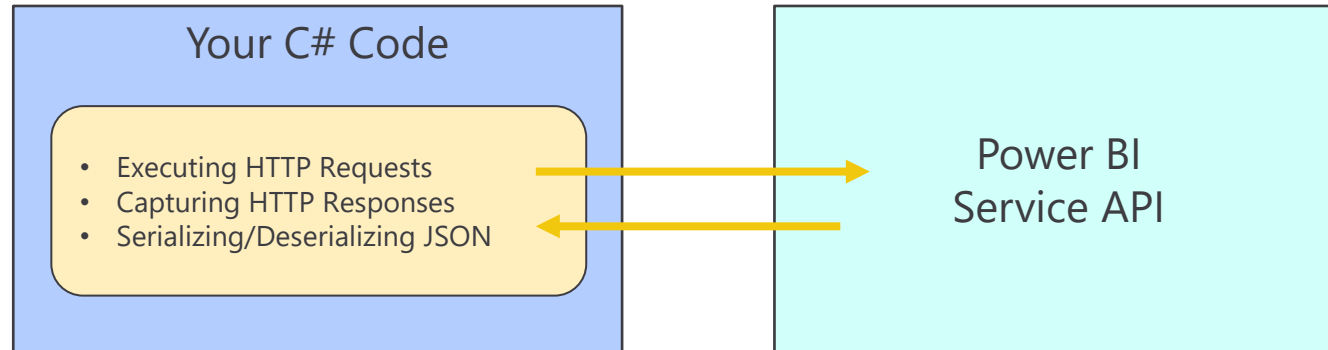
public class ReportCollection {
    public List<Report> value { get; set; }
}
```



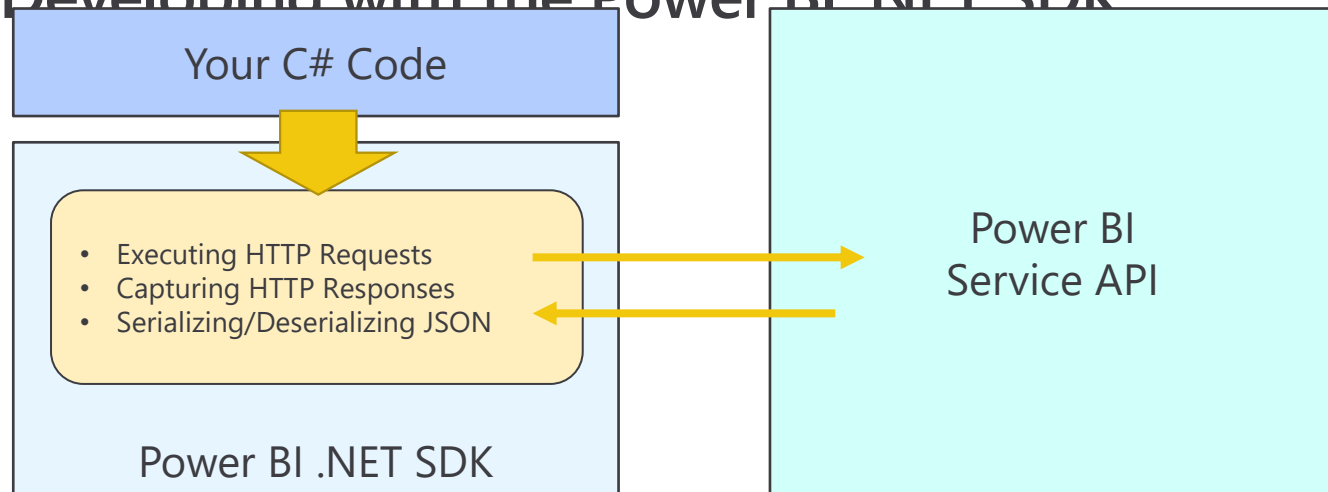


# Power BI .NET SDK

- Developing without the Power BI .NET SDK



- Developing with the Power BI .NET SDK



# Migrating to v3 of the Power BI .NET SDK

- You must move to SDK v3 to take advantage of latest API features
  - Automated server-side generation of PDF file from Power BI report
  - Advanced generation of embed tokens



- Beware of breaking changes when moving from v2.x to 3.x
  - Namespace *Microsoft.PowerBI.Api.v2* renamed to *Microsoft.PowerBI.Api*
  - Namespace *Microsoft.PowerBI.Api.Models.v2* renamed to *Microsoft.PowerBI.Api.Models*
  - *Parameters for Power BI resource IDs now based on GUIDs instead of strings*

# Initializing an Instance of PowerBIClient

- PowerBIClient object serves as top-level object
  - Used to execute calls against Power BI Service
  - Initialized with function to retrieve AAD access token

```
static string GetAccessToken() ...

static PowerBIClient GetPowerBiClient() {
    var tokenCredentials = new TokenCredentials(GetAccessToken(), "Bearer");
    return new PowerBIClient(new Uri(urlPowerBiRestApiRoot), tokenCredentials);
}

static void Main() {
    PowerBIClient pbiClient = GetPowerBiClient();
    var reports = pbiClient.Reports.GetReports().Value;
    foreach (var report in reports) {
        Console.WriteLine(report.Name);
    }
}
```

# User APIs versus Admin APIs

- **Power BI User APIs (e.g. GetGroupsAsync)**
  - provides users with access to personal workspace
  - provides users with access to app workspaces
  - provides service principal (SP) with access to app workspaces
- **Power BI Admin APIs (e.g. GetGroupsAsAdminAsync)**
  - provides users with tenant-level access to all workspaces
  - does not currently support app-only authentication

# Calling into the Power BI Admin API

## Admin API exposed using AsAdmin methods

Example `pbiClient.Groups.GetGroupsAsAdmin(top: 100).Value;`

Makes it possible to access every workspace in current tenant

Requires access token for user who is tenant or Power BI admin

```
static void DisplayAllWorkspacesInTenant() {  
    string[] scopesTenantAdmin = new string[] {  
        "https://analysis.windows.net/powerbi/api/Tenant.ReadWrite.All", // requires admin  
    };  
  
    string AccessToken = GetAccessTokenInteractive(scopesKitchenSink);  
  
    var pbiClient = new PowerBIClient(new Uri(urlPowerBiRestApiRoot),  
                                     new TokenCredentials(AccessToken, "Bearer"));  
  
    Console.WriteLine("Display All Workspaces in Tenant using GetGroupsAsAdmin:");  
    var workspaces = pbiClient.Groups.GetGroupsAsAdmin(top: 100).Value;  
  
    foreach (var workspace in workspaces) {  
        Console.WriteLine("- " + workspace.Type + ": " + workspace.Name + " [" + workspace.Id + "] ");  
    }  
    Console.WriteLine();  
}
```

token

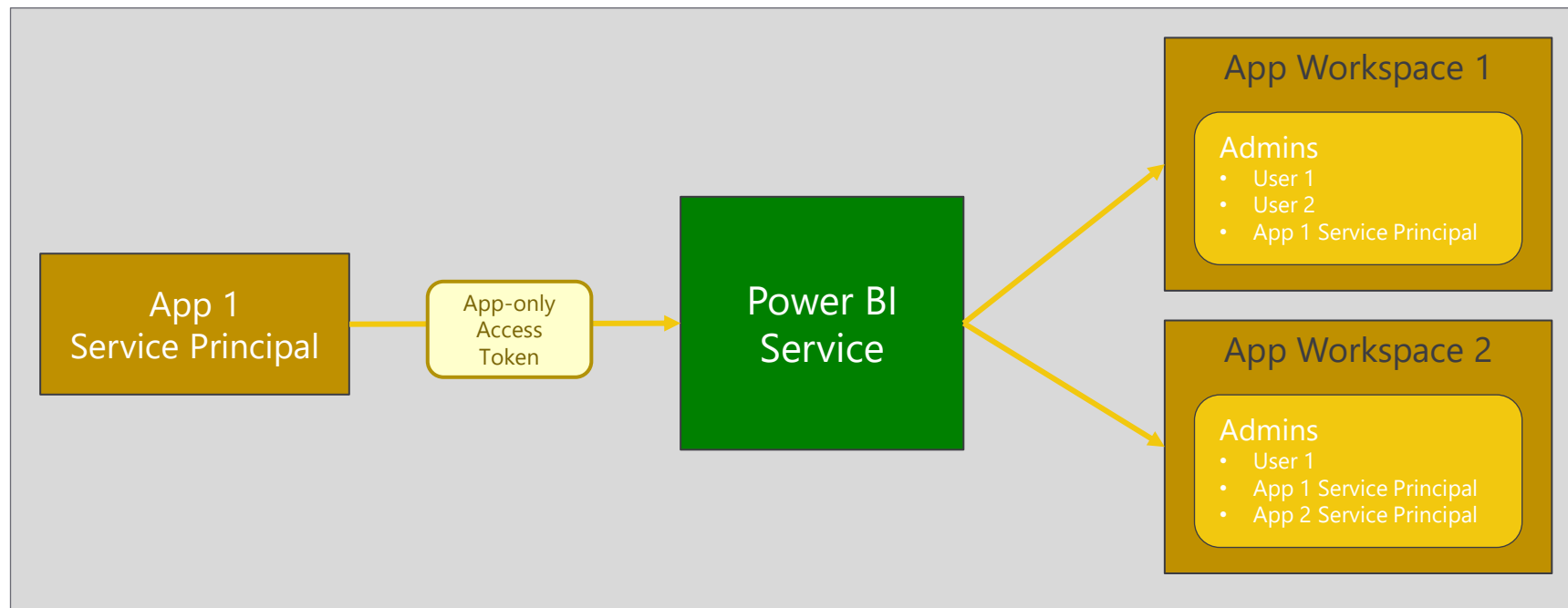
---

## Agenda

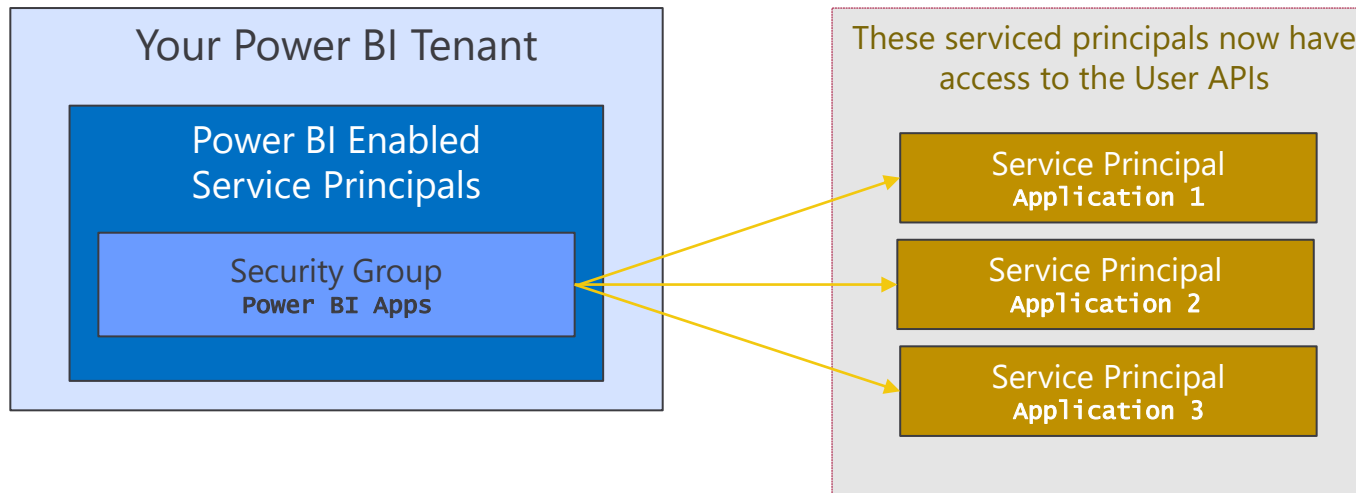
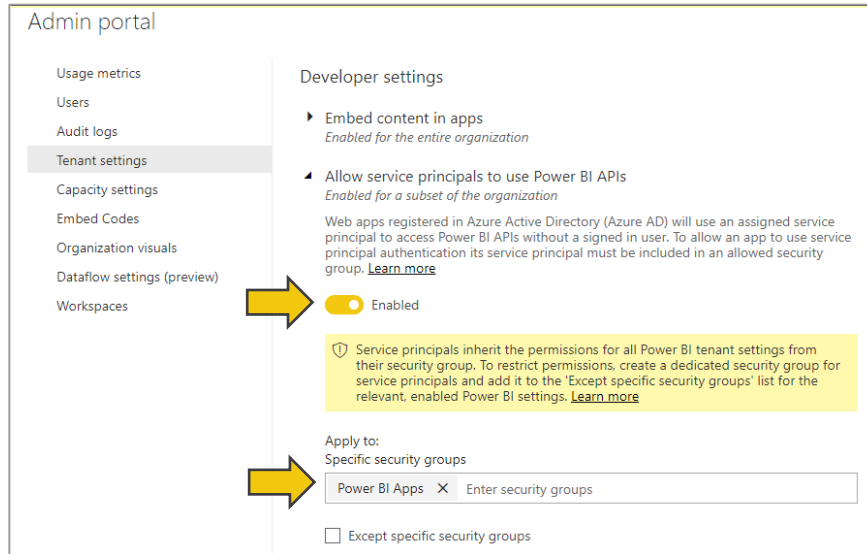
- 
- ✓ OAuth 2.0 and OpenID Connect
  - ✓ Microsoft Identity Platform 2.0
  - ✓ Developing with the Power BI .NET SDK
  - Developing with the App-Owns-Data Model
    - Developing Single Page Applications (SPAs)
    - Developing Secure Web Applications

# App-only Access Control

- Service Principal used to configure access control
  - Requires the use of v2 app workspaces
  - Service principal must be added to app workspaces as admin or member
  - Access control NOT based on Azure AD permissions



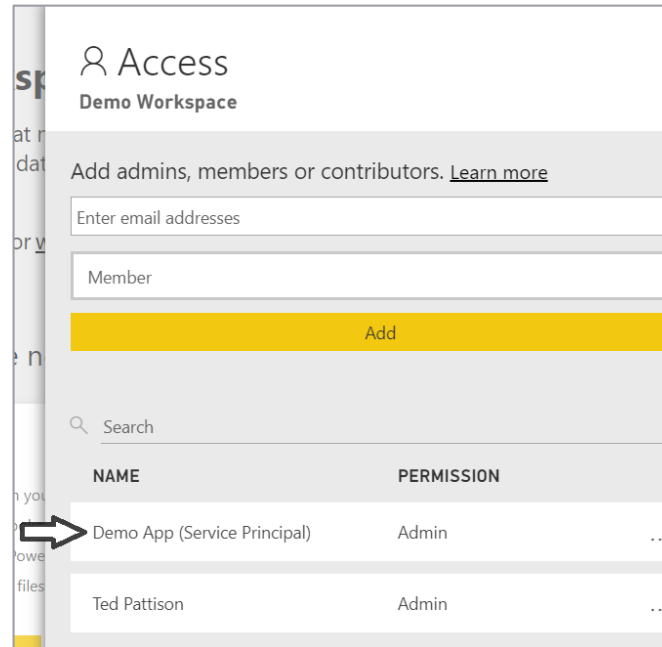
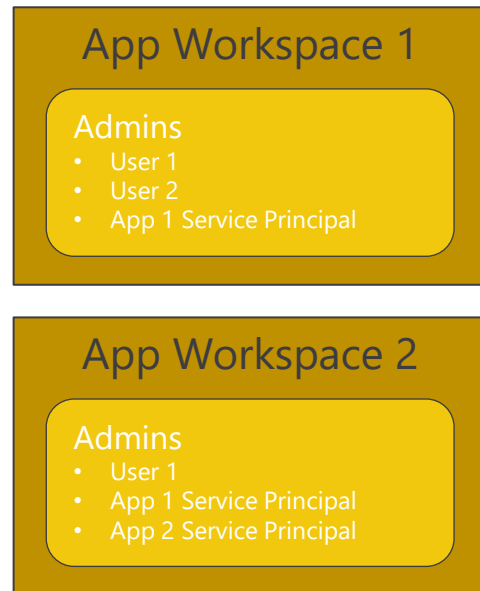
# Tenant Setup





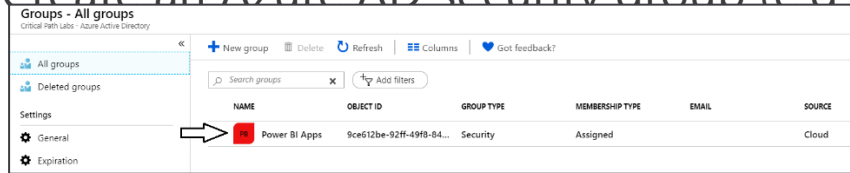
# App-only Access with PBI Service API

- Service Principal added to workspace as admin
  - Only works with v2 app workspaces
  - Provides full workspace access to service principal

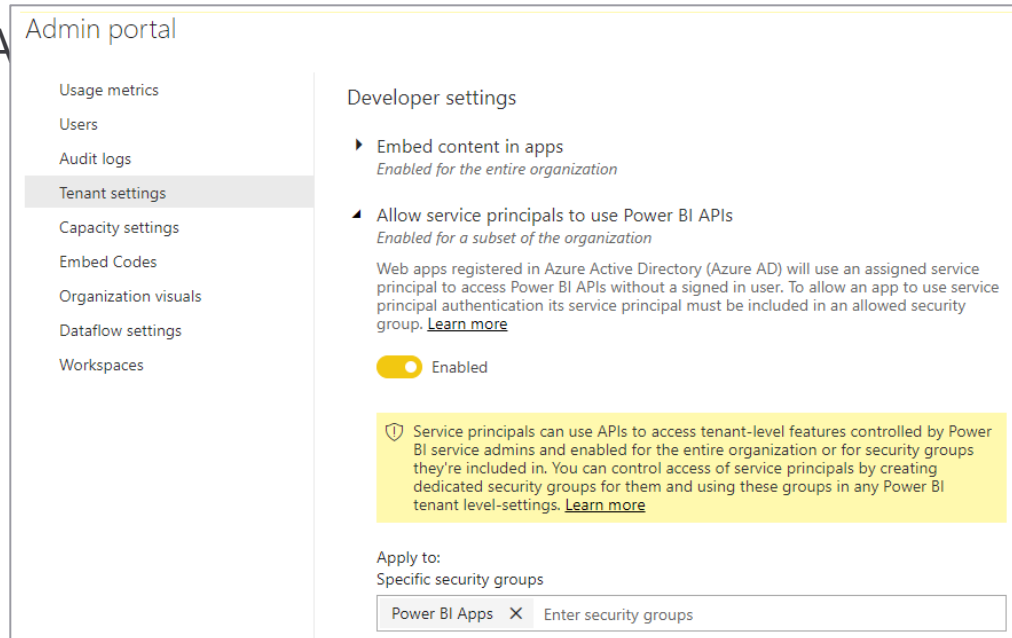


# Setting Up for App-Owns-Data – Part 1

- Enable Service Principal Access to Power BI Service API
- Create an Azure AD security group (e.g. Power BI Apps)




- Admin portal *use Power BI APIs*



# Setting Up for App-Owns-Data – Part 2

- Create a confidential client in your Azure AD tenant

Display name	: <a href="#">App-Owns-Data App</a>	Supported account types	: <a href="#">My organization only</a>
Application (client) ID	: af5d13b2-daa3-4b14-ac20-3ee338220630	Redirect URIs	: <a href="#">Add a Redirect URI</a>
Directory (tenant) ID	: 17b8d777-a84a-4b90-b4a3-559ee5cbf07e	Managed application in ...	: <a href="#">App-Owns-Data App</a>
Object ID	: 8f1f9327-f5cc-46b5-babf-6e821a5df079		

-  **CONFIDENTIAL CLIENT**

Manage

Branding

Authentication

Certificates & secrets


API permissions

Expose an API

Redirect URIs

The URIs that we will accept as destinations when returning authentication responses (tokens) after successfully authenticating users. Also referred to as reply URLs.  
[Learn more about adding support for web, mobile and desktop clients](#)

TYPE	REDIRECT URI
Web	e.g. <a href="https://myapp.com/auth">https://myapp.com/auth</a>

-  **SECRET**

Manage

Branding

Authentication

Certificates & secrets

API permissions


Expose an API

Client secrets

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

DESCRIPTION	EXPIRES	VALUE
No description	6/3/2020	Hidden

-  **PERMISSIONS**

Manage

Branding

Authentication

Certificates & secrets

API permissions

Expose an API

Owners

API permissions

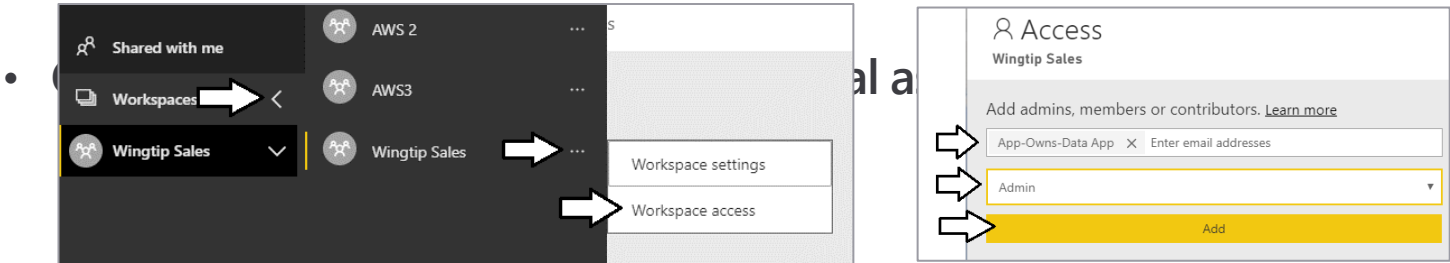
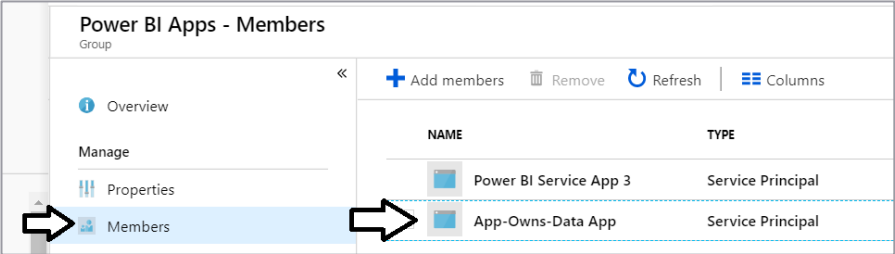
Applications are authorized to use APIs by requesting permissions. These permissions show up during the consent process where users are given the opportunity to grant/deny access.

+ Add a permission

API / PERMISSIONS NAME	TYPE	DESCRIPTION	ADMIN CONSENT REQUIRED
No permissions added			

# Setting Up for App-Owns-Data – Part 3

- Add application's service principal in Power BI Apps security group



NAME	PERMISSION	
App-Owns-Data App (Service Principal)	Admin	...
Power BI Service App 3 (Service Princip...	Admin	...
Ted Pattison	Admin	...

- Set the workspace admin

# Client Credentials Flow

- Client credentials flow used to obtain app-only token
  - Requires passing app secret (e.g. app password or certificate)
  - Requires passing tenant-specific endpoint

```
const string clientId = "e6a54dc4-7345-495d-b029-88c6349b62d2";
const string clientSecret = "M2MwODBhOTEtOWUyYi00NWQ1LWJmMTQzMjM1ZTAzMzZjOTMx=";
const string tenantName = "devinaday2019.onmicrosoft.com";

// endpoint for tenant-specific authority
const string tenantSpecificAuthority = "https://login.microsoftonline.com/" + tenantName;

static string GetAppOnlyAccessToken() {
    var appConfidential = ConfidentialClientApplicationBuilder.Create(clientId)
        .WithClientSecret(clientSecret)
        .WithAuthority(tenantSpecificAuthority)
        .Build();

    string[] scopesDefault = new string[] { "https://analysis.windows.net/powerbi/api/.default" };

    var authResult = appConfidential.AcquireTokenForClient(scopesDefault).ExecuteAsync().Result;

    return authResult.AccessToken;
}
```

# Generating Embed Tokens

- You can embed reports using an AAD token, but...
  - You might want embed resource using more restricted tokens
  - You might want stay within the bounds of Power BI licensing terms
- You generate embed tokens with the Power BI Service API
  - Each embed token created for one specific resource
  - Embed token provides restrictions on whether user can view or edit
  - Embed token can only be generated in dedicated capacity (semi-enforced)
  - Embed token can be generated to support row-level security (RLS)

```
Report report = reports.Where(r => r.Id == reportId).First();  
var generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "edit");  
var token = client.Reports.GenerateTokenInGroupAsync(appWorkspaceId,  
                                                    report.Id,  
                                                    generateTokenRequestParameters).Result;
```

# Getting the Data for Report Embedding

```
public static async Task<ReportEmbeddingData> GetReportEmbeddingData() {  
    PowerBIClient pbiClient = GetPowerBiClient();  
  
    var report = await pbiClient.Reports.GetReportInGroupAsync(workspaceId, reportId);  
    var embedUrl = report.EmbedUrl;  
    var reportName = report.Name;  
  
    GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "edit");  
    string embedToken =  
        (await pbiClient.Reports.GenerateTokenInGroupAsync(workspaceId,  
                                                            report.Id,  
                                                            generateTokenRequestParameters)).Token;  
  
    return new ReportEmbeddingData {  
        reportId = reportId,  
        reportName = reportName,  
        embedUrl = embedUrl,  
        accessToken = embedToken  
    };  
}
```

# Getting the Data for Dashboard Embedding

```
public static async Task<DashboardEmbeddingData> GetDashboardEmbeddingData() {  
    PowerBIClient pbiClient = GetPowerBiClient();  
  
    var dashboard = await pbiClient.Dashboards.GetDashboardInGroupAsync(workspaceId, dashboardId);  
    var embedUrl = dashboard.EmbedUrl;  
    var dashboardDisplayName = dashboard.DisplayName;  
  
    GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "view");  
  
    string embedToken =  
        (await pbiClient.Dashboards.GenerateTokenInGroupAsync(workspaceId,  
                                                                dashboardId,  
                                                                generateTokenRequestParameters)).Token;  
  
    return new DashboardEmbeddingData {  
        dashboardId = dashboardId,  
        dashboardName = dashboardDisplayName,  
        embedUrl = embedUrl,  
        accessToken = embedToken  
    };  
}
```



# Getting Data for New Report Embedding

```
public static NewReportEmbeddingData GetNewReportEmbeddingData() {  
    string embedUrl = "https://app.powerbi.com/reportEmbed?groupId=" + workspaceId;  
    PowerBIClient pbiclient = GetPowerBiClient();  
    GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "create",  
                                                                                      datasetId: datasetId);  
    string embedToken = pbiclient.Reports.GenerateTokenForCreateInGroup(workspaceId,  
                                                                           generateTokenRequestParameters).Token;  
    return new NewReportEmbeddingData {  
        workspaceId = workspaceId,  
        datasetId = datasetId,  
        embedUrl = embedUrl,  
        accessToken = embedToken  
    };  
}
```

---

## Agenda

- 
- ✓ OAuth 2.0 and OpenID Connect
  - ✓ Microsoft Identity Platform 2.0
  - ✓ Developing with the Power BI .NET SDK
  - ✓ Developing with the App-Owns-Data Model
  - Developing Single Page Applications (SPAs)
  - Developing Secure Web Applications

# Understanding Implicit Flow

- **Used in single page applications (SPAs)**
  - All code and configuration loaded into client-side browser
  - Application runs as web application at verifiable URL on the Internet
  - SPA must be running at URL registered as a redirect URL
  - Application isn't really confidential - it cannot keep any secret
- **Implicit flow involves passing access token directly to browser**
  - Passing access token to browser isn't ideal - but it's tolerated when developing SPAs
- **MSAL provides a version of JavaScript/TypeScript Developers**
  - There are node.js packages and NuGet packages for msal.js

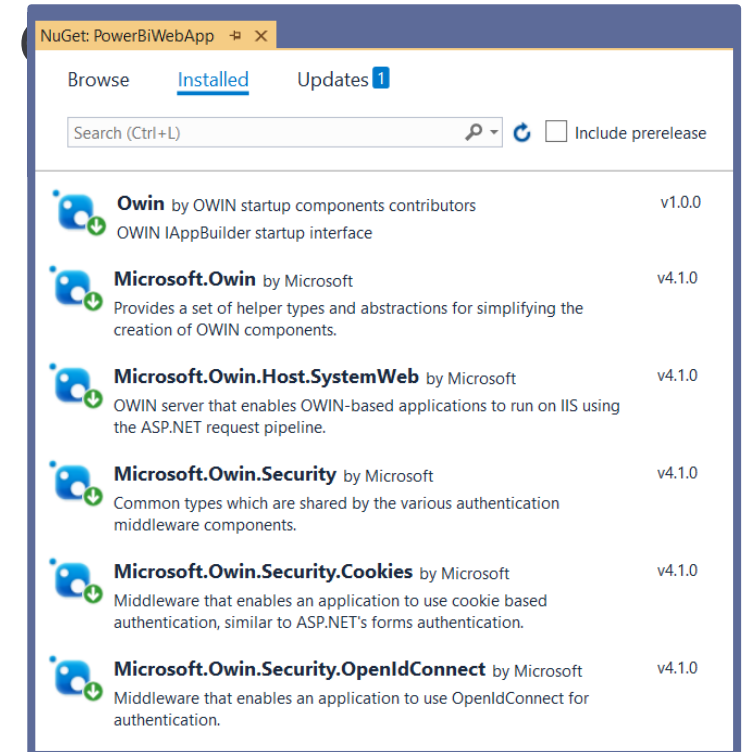
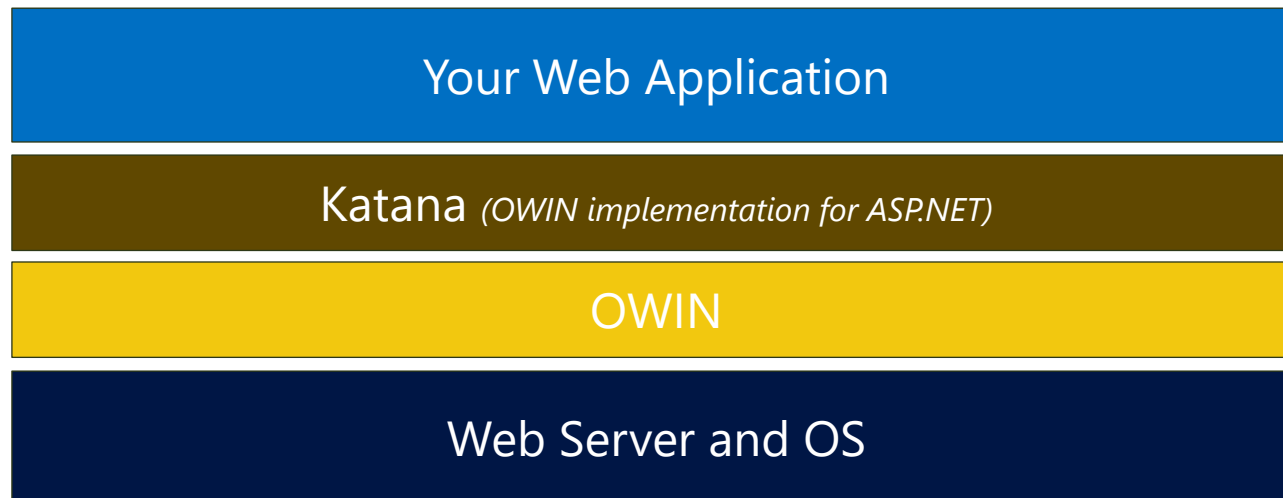
---

## Agenda

- ✓ OAuth 2.0 and OpenID Connect
- ✓ Microsoft Identity Platform 2.0
- ✓ Developing with the Power BI .NET SDK
- ✓ Developing with the App-Owns-Data Model
- ✓ Developing Single Page Applications (SPAs)
- Developing Secure Web Applications

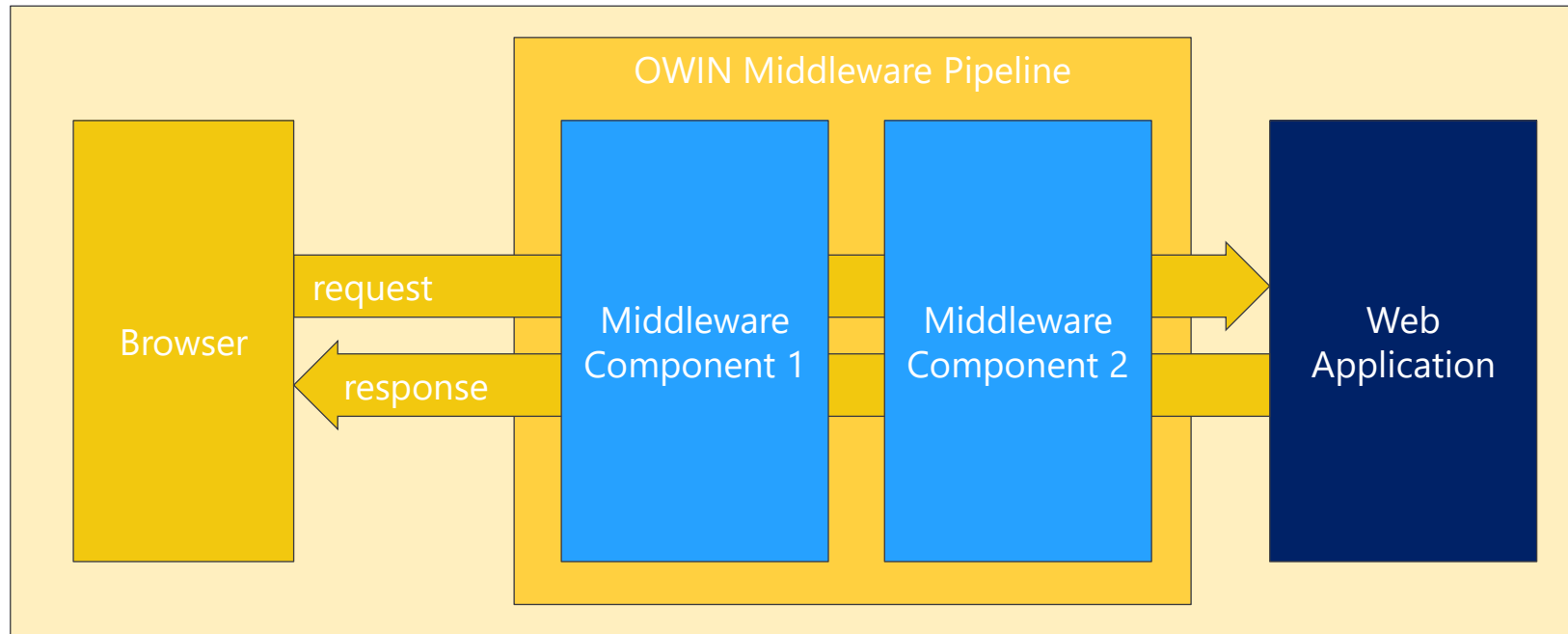
# Open Web Interfaces for NET (OWIN)

- OWIN interfaces decouple web server from application
  - OWIN serves to decouple .NET applications from Windows and IIS
  - OWIN promotes the development of smaller modules (middleware)
- Microsoft's Implementation known as Katana
  - Makes it possible to use OWIN with ASP.NET and ASP.NET Core
  - Microsoft provides OWIN-based security middleware



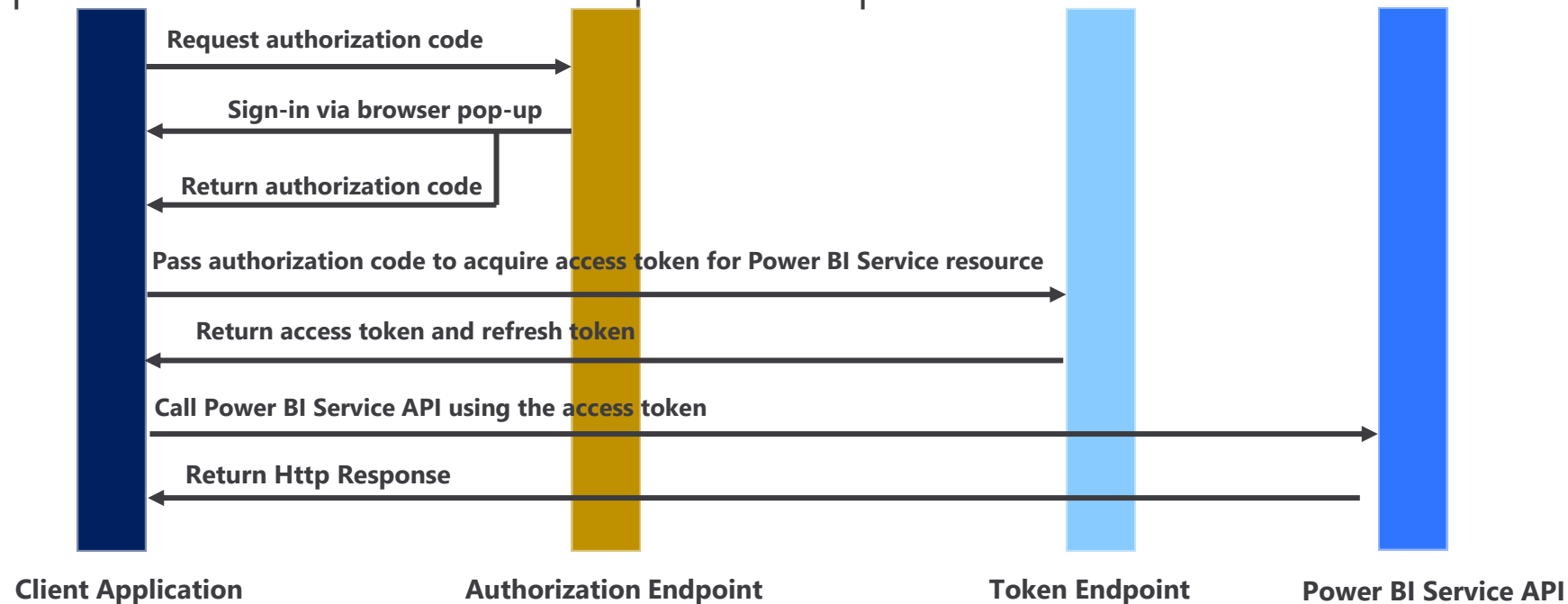
# OWIN Middleware Modules

- OWIN create pipeline of middleware components
  - Middleware components added to pipeline on application startup
  - Middleware components pre-process and post process requests
  - Middleware components commonly used to set up authentication



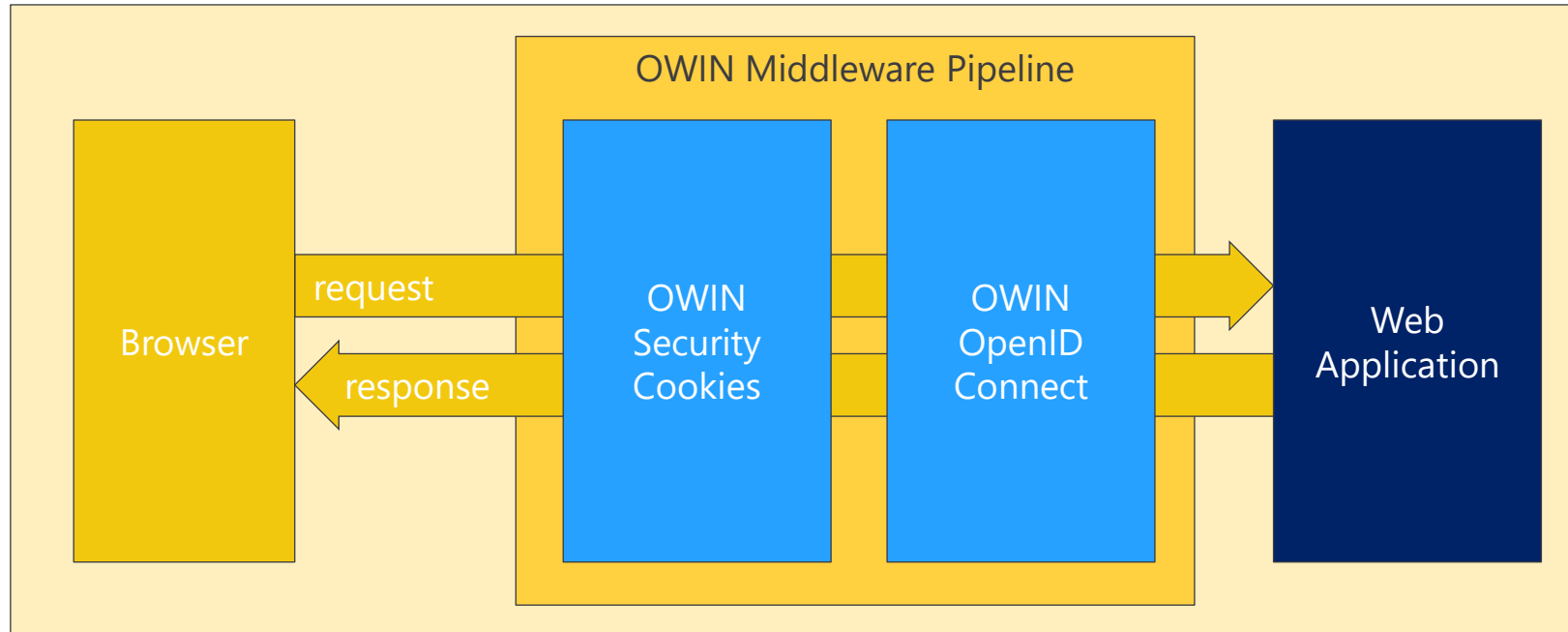
# Authorization Code Grant Flow

- Sequence of Requests in Authorization Code Grant Flow
  - Application redirects to AAD authorization endpoint
  - User prompted to sign in using Windows logon page
  - User prompted to consent to permissions (first access)
  - AAD redirects to application with authorization code
  - Application calls to AAD token endpoint to acquire access token



# OWIN OpenID Connect Module

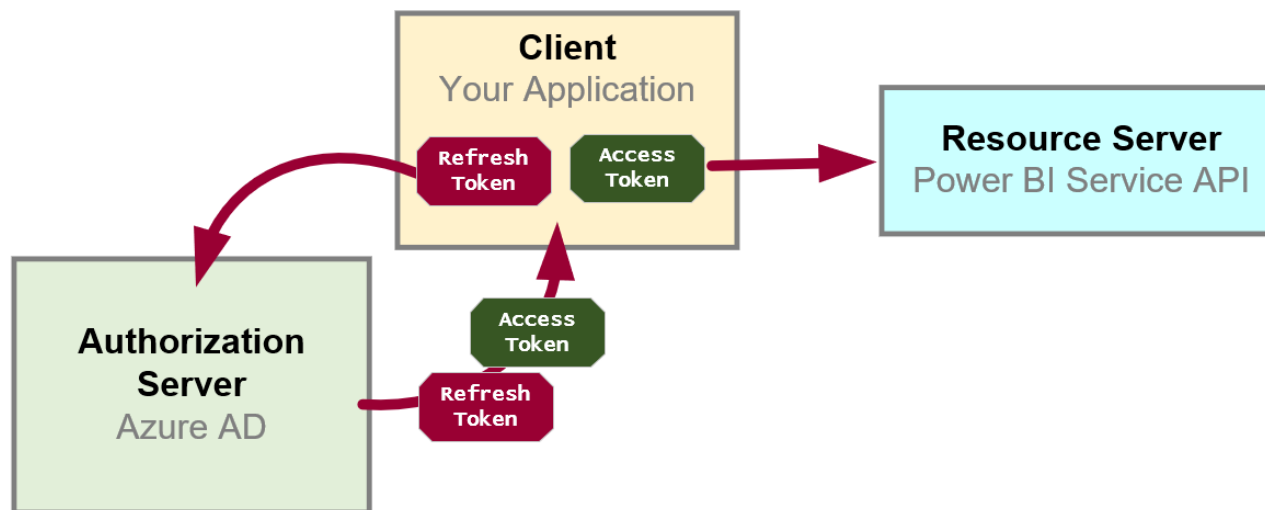
- OpenID Connect module used to implement Authorization Code Flow
  - Redirects browsers to authorization endpoint
  - Provides notification when receiving authorization code callback





# Token Caching and Refresh Tokens

- OAuth 2.0 provide solution for access token expiration
  - Access tokens have default lifetime of 60 minutes
  - Authorization server passes refresh token along with access token
  - Refresh token used as a credential to redeem new access token
  - Refresh token default lifetime is 14 days (max 90 days)
  - Refresh tokens often persistent in database or browser storage
  - MSAL offers built-in support to manage token caching



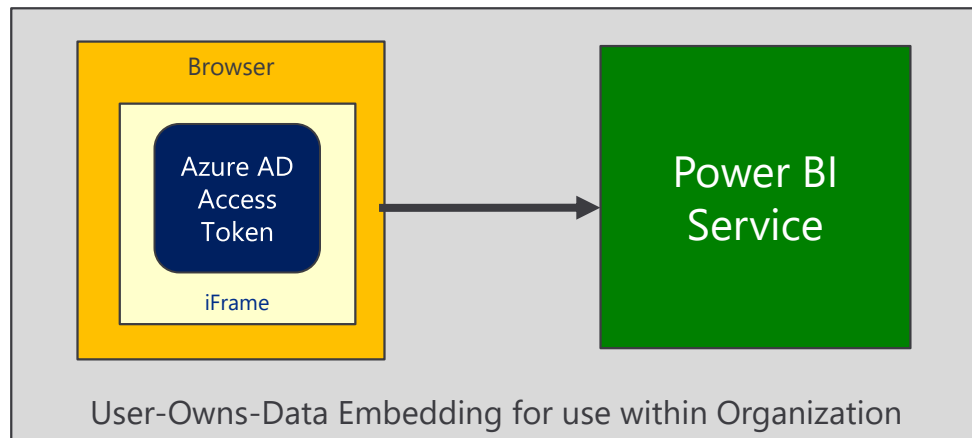
# Summarization of Authentication Flows

- **Public clients**
  - Interactive Flow
  - User Password Credential Flow
- **Confidential web clients**
  - Client Credentials Flow
  - Implicit Flow
  - Authorization Code Flow

# Summarizing Power BI Authentication Patterns

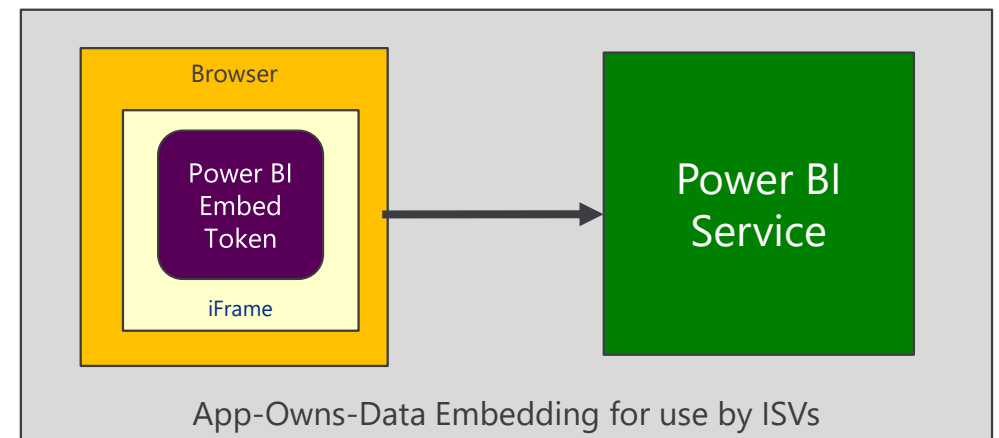
## User-Owns-Data Embedding

App authenticates as current user  
Use implicit flow for SPAs  
Use auth code for better security  
User's access token passed to browser  
Row-level security (RLS) works as expected



## App-Owns-Data Embedding

App authenticates with app-only identity  
Use client credentials flow  
Service principle must be workspace member  
Embed token passed to browser  
*EffectiveIdentity* used with RLS



---

## Summary

- 
- ✓ OAuth 2.0 and OpenID Connect
  - ✓ Microsoft Identity Platform 2.0
  - ✓ Developing with the Power BI .NET SDK
  - ✓ Developing with the App-Owns-Data Model
  - ✓ Developing Single Page Applications (SPAs)
  - ✓ Developing Secure Web Applications

# Questions