Microsoft

Microsoft Business Applications Summit

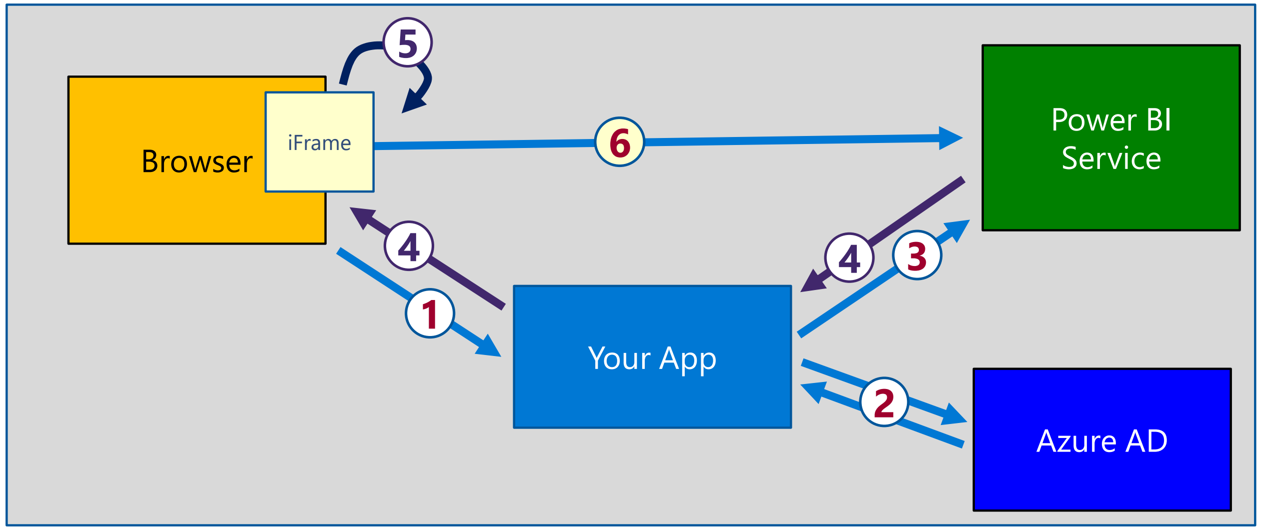
Microsoft

Authentication patterns for Power BI embedding

Ted Pattison
Principle Program Manager
Power BI Customer Advisory Team (CAT)

Power BI Embedding – The Big Picture

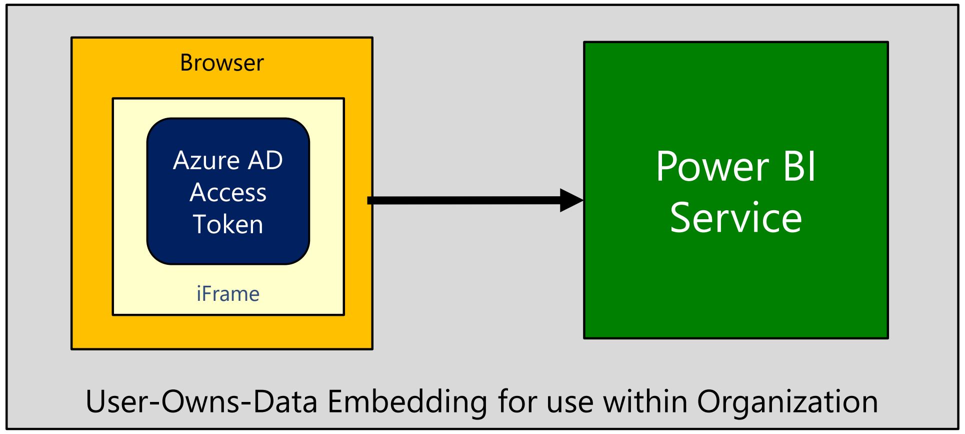
- User launches your app using a browser
- App authenticates with Azure Active Directory and obtains access token
- App uses access token to call to Power BI Service API
- App retrieves data for embedded resource and passes it to browser.
- Client-side code uses Power BI JavaScript API to create embedded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

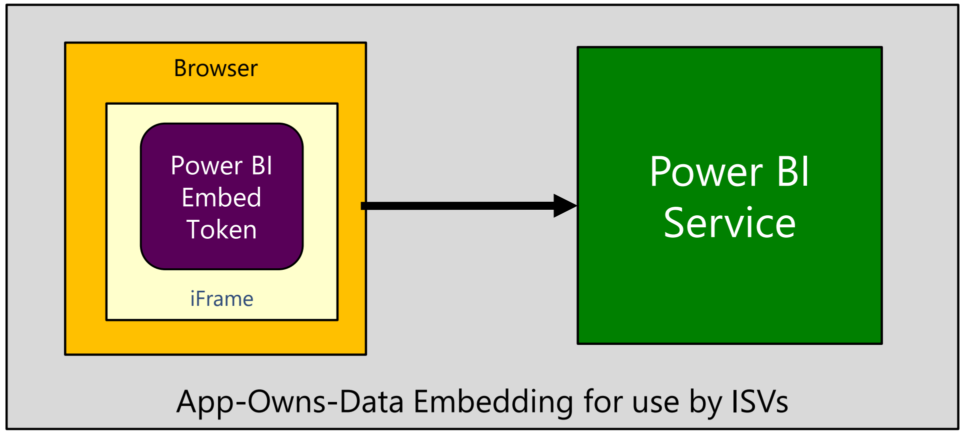


Choosing the Correct Embedding Model

- 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



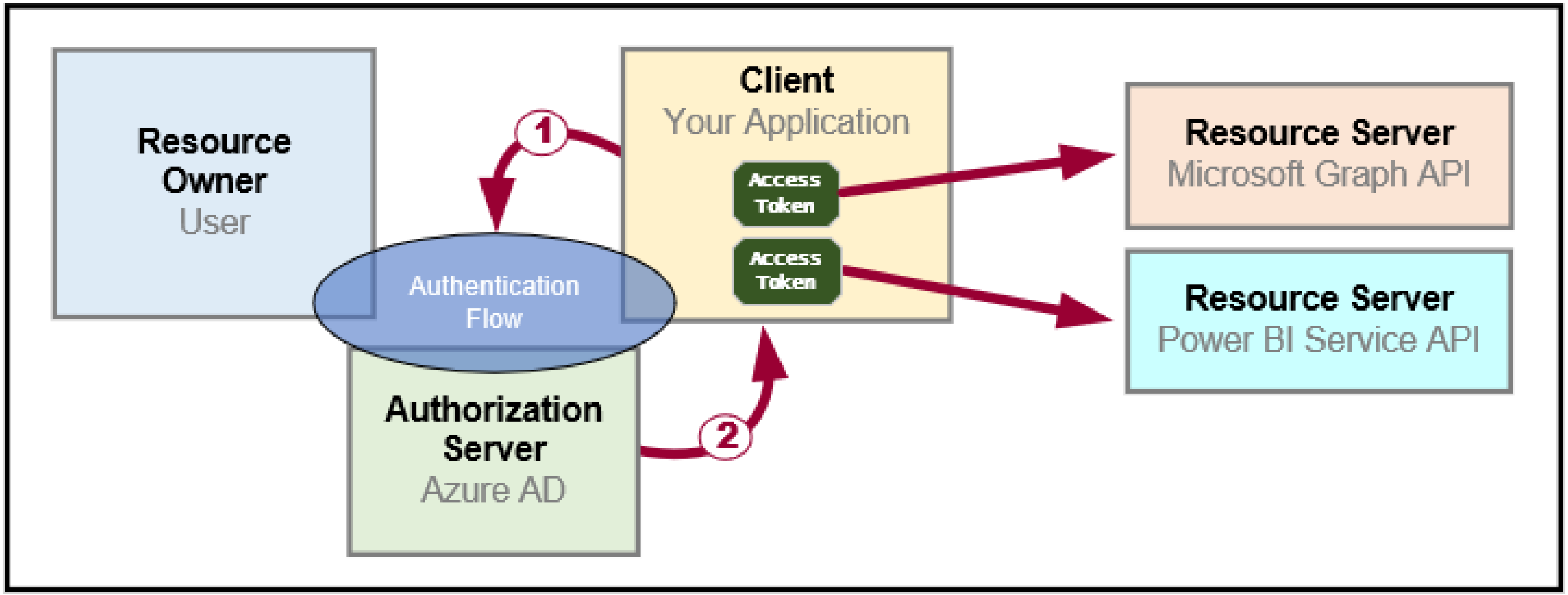


Agenda

- ➢ 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

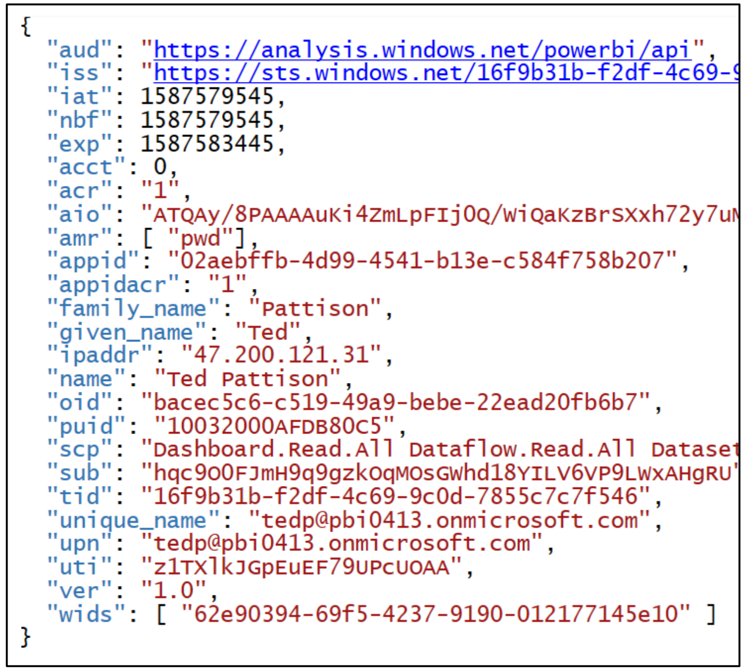
OAuth 2.0 Fundamentals

- 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
  - Resource server inspects access token to ensure client application has proper permissions



Access Tokens

- Access token is a bearer token
  - Access token can be used by anyone who bears it (e.g. steals it)
  - Access tokens should always be passed over HTTPS using TLS
  - 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-4c69-9
  "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
  "sub": "hqc9O0FJmH9q9gzkOqMOsGWhd18YILV6VP9LWxAHgRU"
  "tid": "16f9b31b-f2df-4c69-9c0d-7855c7c7f546",
  "unique_name": "tedp@pbi0413.onmicrosoft.com",
  "upn": "tedp@pbi0413.onmicrosoft.com",
  "uti": "z1TXlkJGpEuEF79UPcUOAA",
  "ver": "1.0",
  "wids": [ "62e90394-69f5-4237-9190-012177145e10" ]
}
```

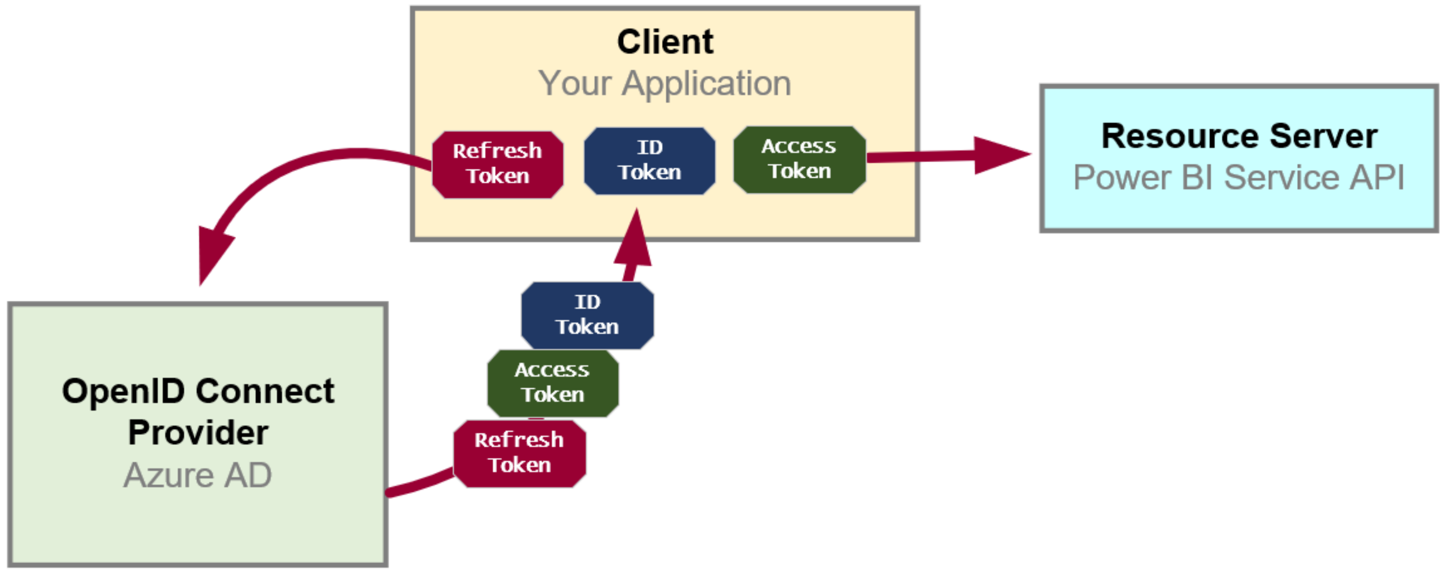
Refresh Tokens

- 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

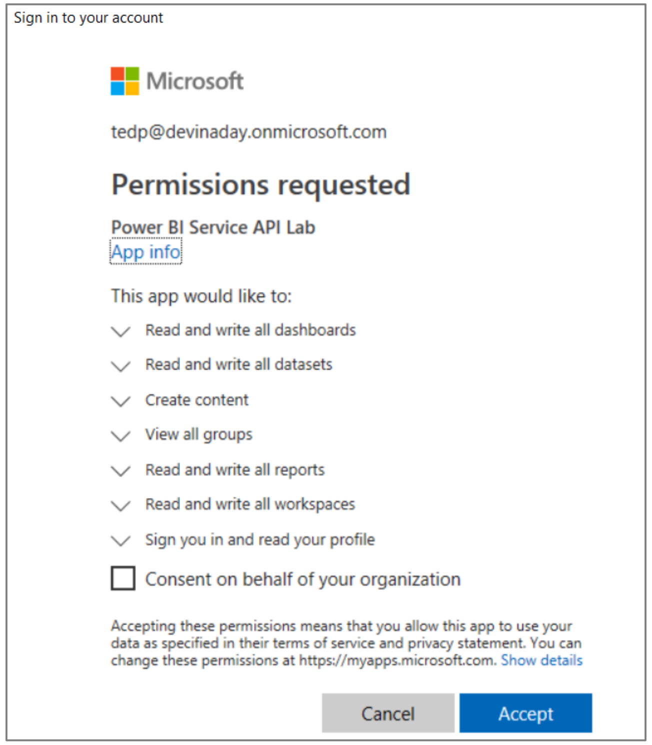
- OAuth 2.0 has shortcomings with authentication & identity
  - It does not provide client with means to validate access tokens
  - Lack of validation makes client vulnerable to token forgery attacks
- Open ID Connect is standard which extends OAuth 2.0
  - OpenID Connect provider passes ID token in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

- 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
    - https://analysis.windows.net/powerbi/api/Report.Read.All


Sign in to your account

Microsoft

tedp@devinaday.onmicrosoft.com

Permissions requested

Power BI Service API Lab

App info

This app would like to:

- Read and write all dashboards
- Read and write all datasets
- Create content
- View all groups
- Read and write all reports
- Read and write all workspaces
- Sign you in and read your profile

☐ Consent on behalf of your organization

Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. You can change these permissions at https://myapps.microsoft.com. Show details

Public Clients versus Confidential Web Clients

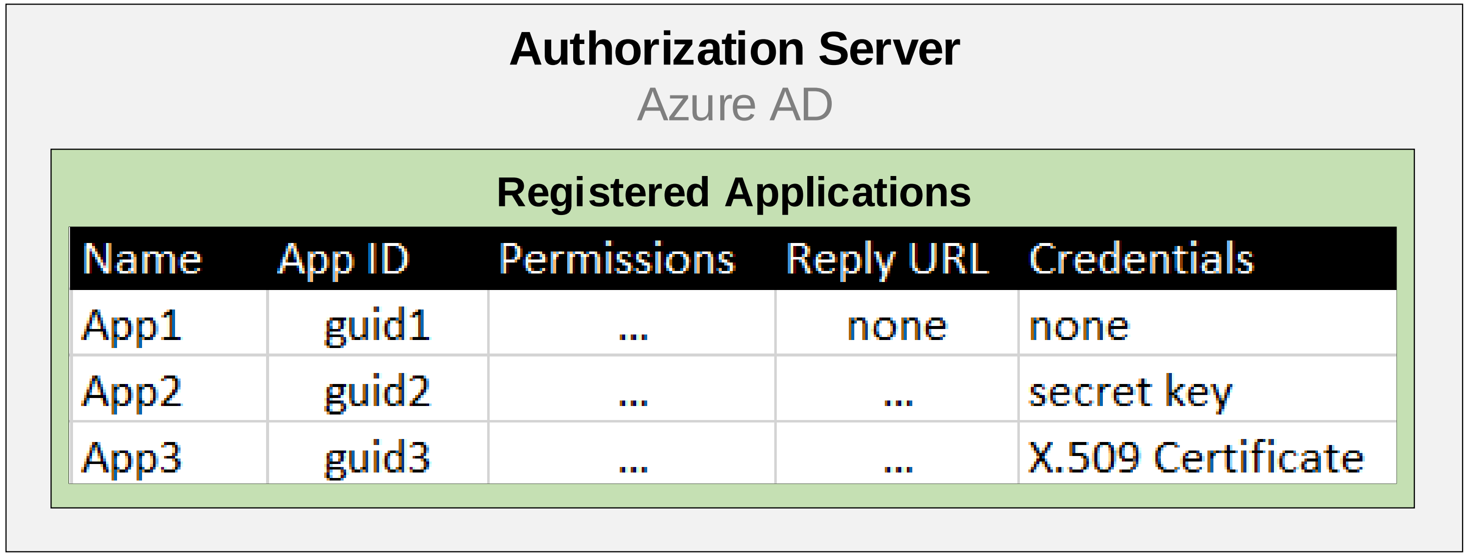
- OAuth 2.0 defines two different types of client applications
  - Public clients
  - Confidential web clients

- 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 permissions)



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

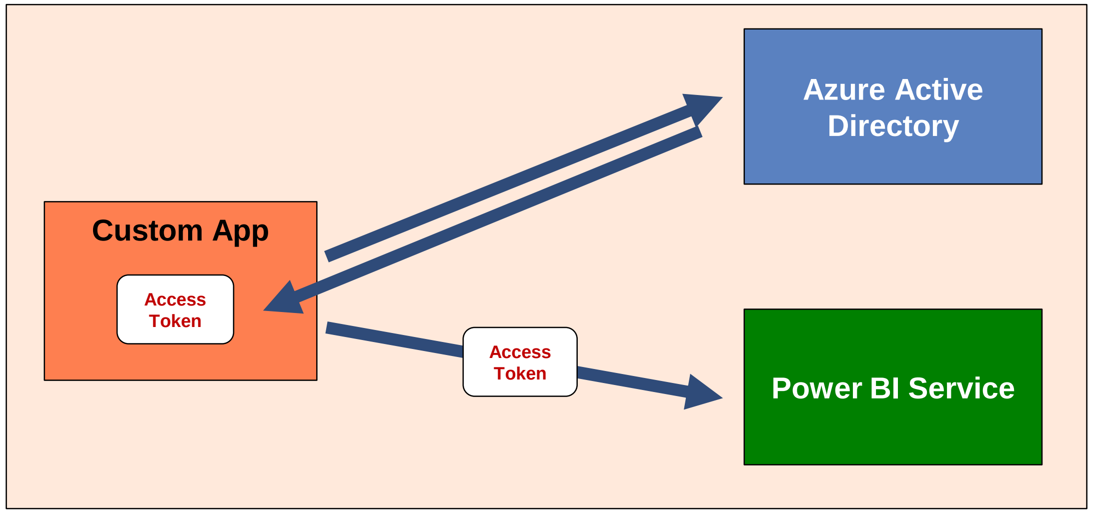
- Public/native clients
  - Interactive Flow
    - Used in public clients to obtain access token interactively
  - User Password Credential Flow
    - Used in Native clients to obtain access code
    - Requires passing user name and password across network
- Confidential Web clients
  - Client Credentials 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

Authenticating with Azure AD

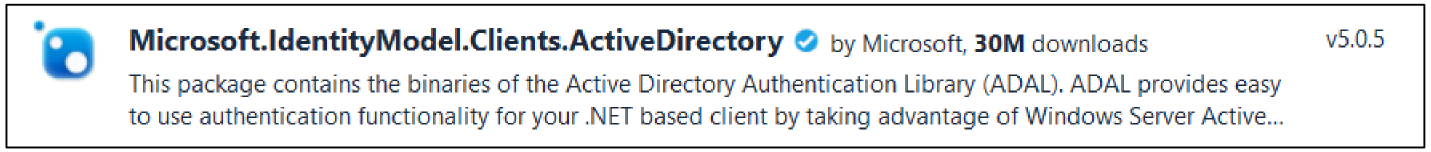
- Custom applications must authenticate with Azure AD
  - Your code implements and authentication flow to obtain access token
  - Access token must be passed when calling Power BI Service API



- 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)

Azure AD Endpoints and Libraries

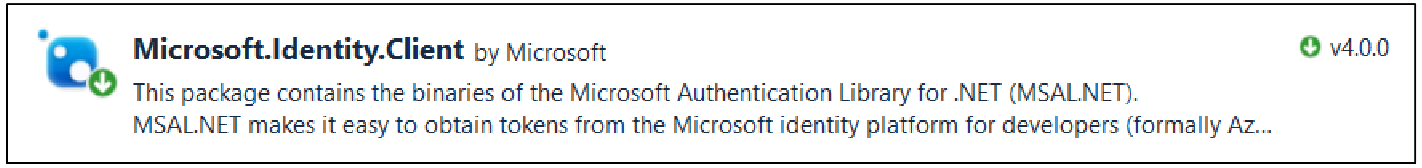
- Authenticating with the Azure AD V1 Endpoint
  - Heavily used over the last 8 years
  - Accessed through Azure AD Authentication Library (ADAL)



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...

- Authenticating with the Azure AD V2 Endpoint
  - Moved from preview to GA in May 2019
  - Accessed through Microsoft Authentication Library (MSAL)



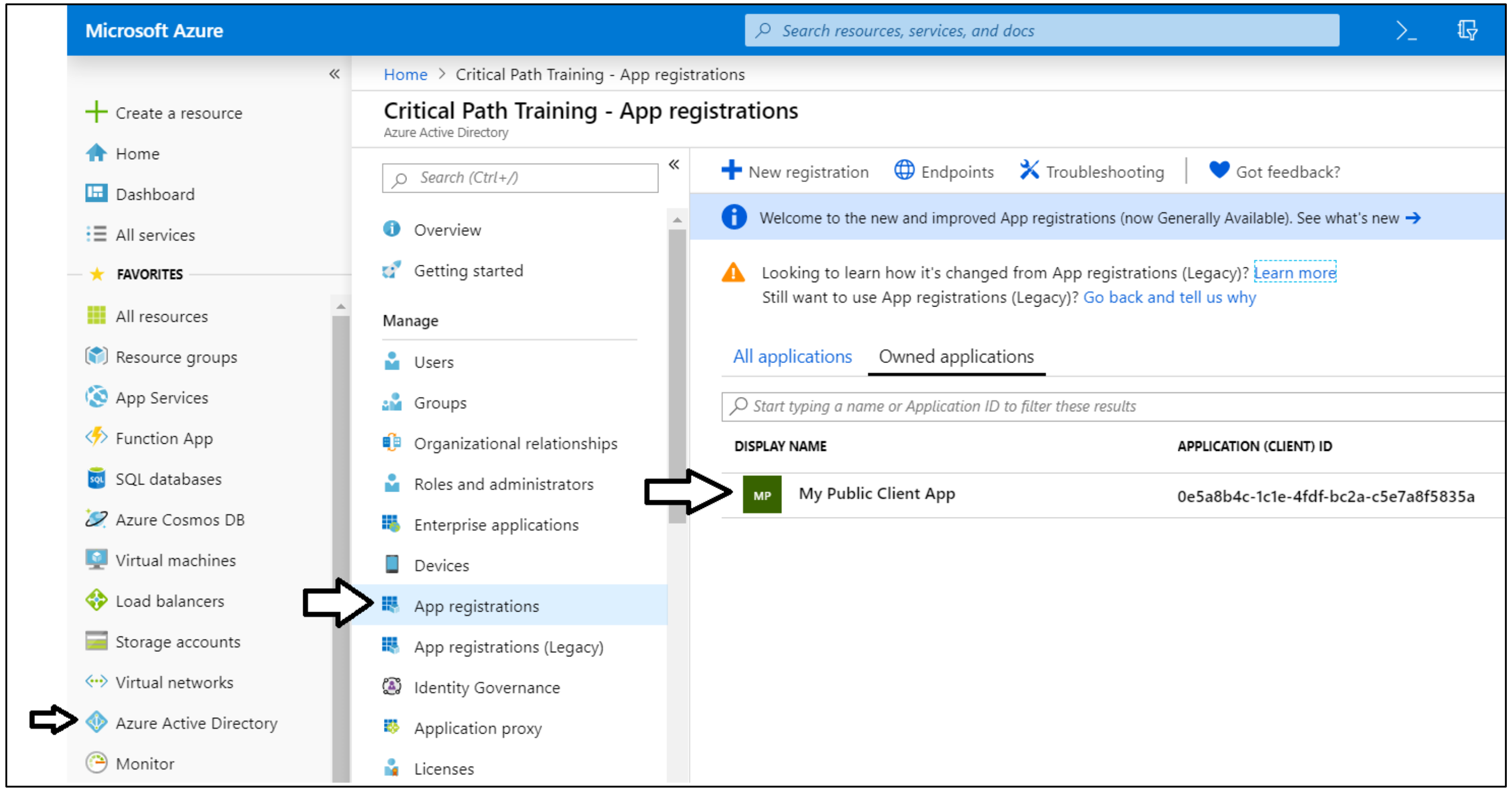
Microsoft.Identity.Client by Microsoft v4.0.0

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...

- Why move to the Azure AD V2 Endpoint?
  - Dynamic incremental consent
  - New features and authentication flows (e.g. device code flow)

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



Microsoft Azure

Search resources, services, and docs

Home > Critical Path Training - App registrations

Critical Path Training - App registrations
Azure Active Directory

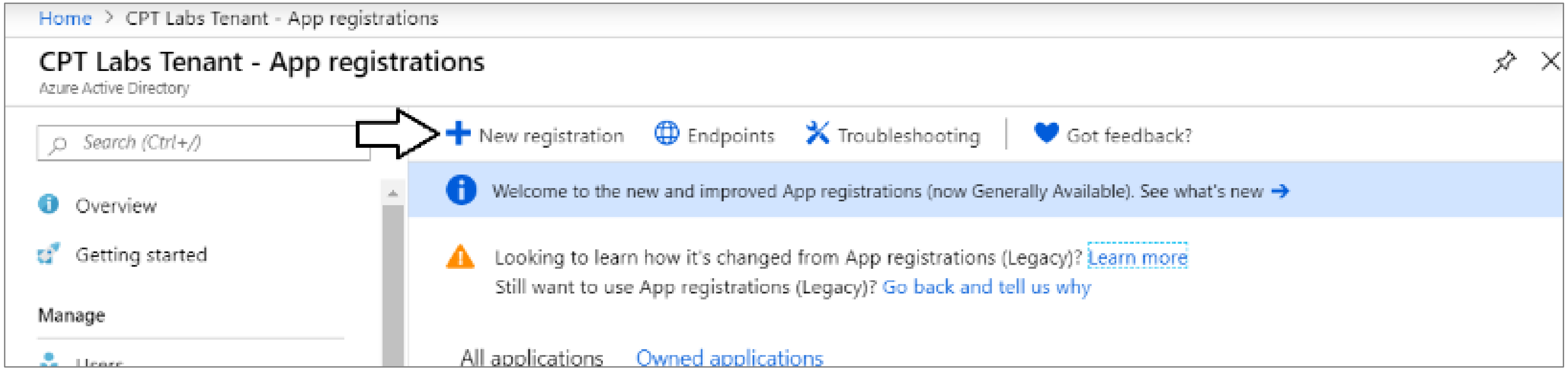
Welcome to the new and improved App registrations (now Generally Available). See what's new →

Looking to learn how it's changed from App registrations (Legacy)? Learn more
Still want to use App registrations (Legacy)? Go back and tell us why

DISPLAY NAME	APPLICATION (CLIENT) ID
MP My Public Client App	0e5a8b4c-1c1e-4fdf-bc2a-c5e7a8f5835a

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



Home > CPT Labs Tenant - App registrations

CPT Labs Tenant - App registrations

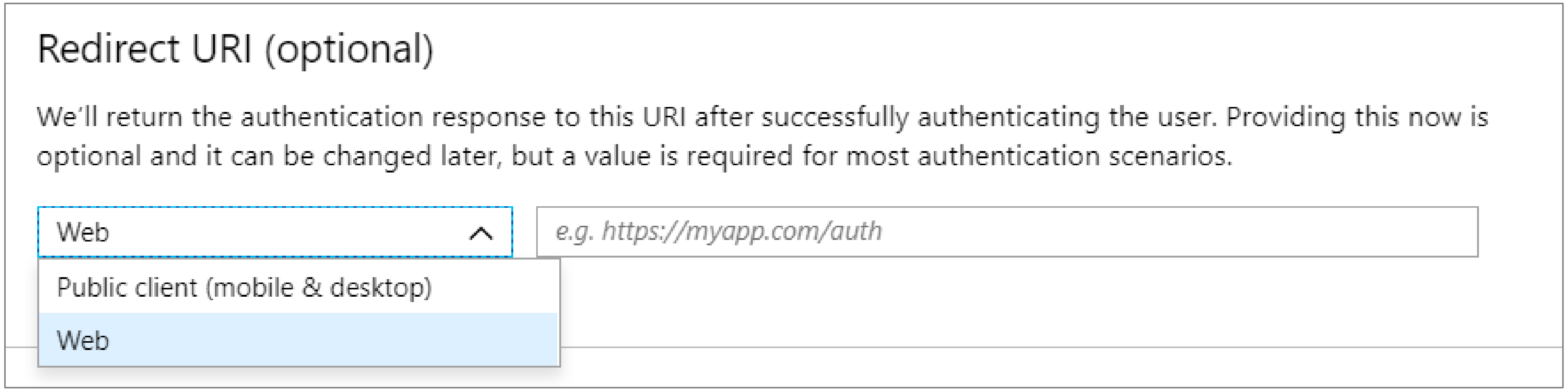
Azure Active Directory

Welcome to the new and improved App registrations (now Generally Available). See what's new →

Looking to learn how it's changed from App registrations (Legacy)? Learn more
Still want to use App registrations (Legacy)? Go back and tell us why

Azure AD Application Types

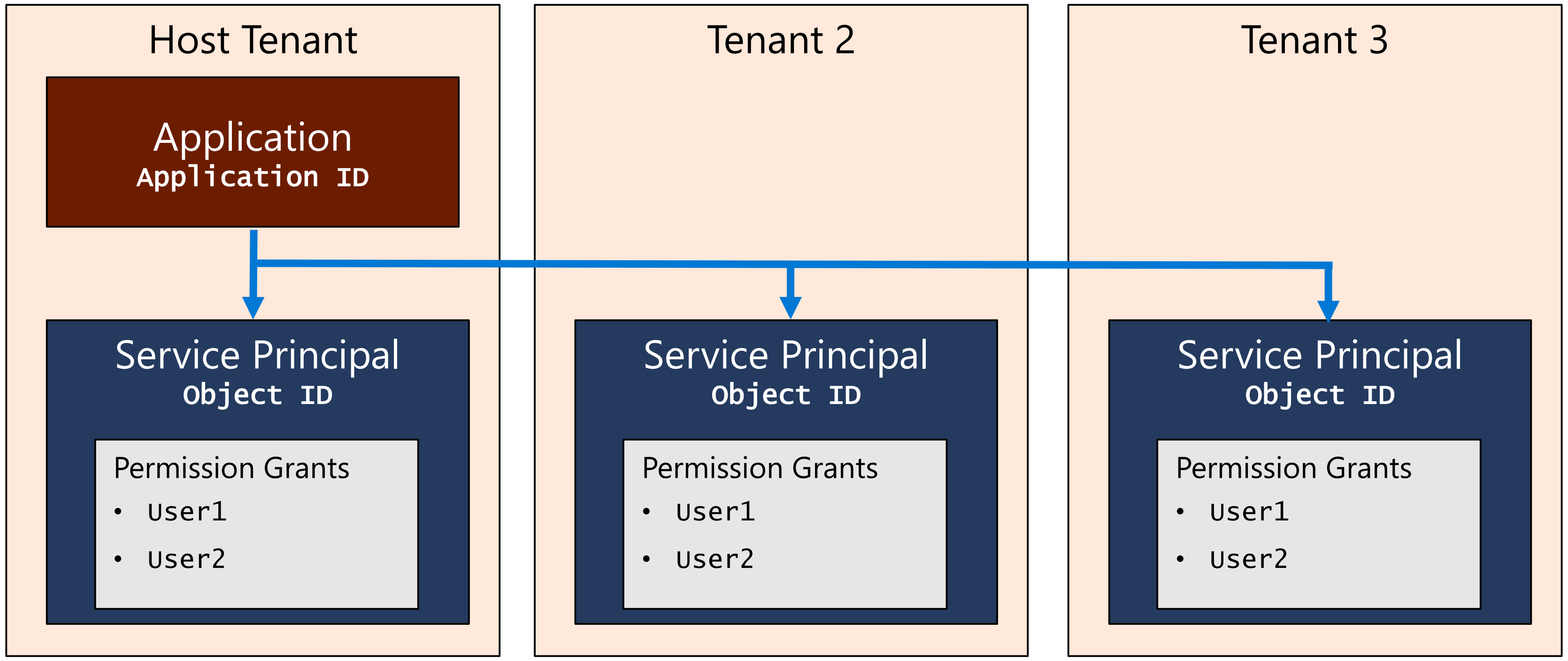
- Public client (mobile and desktop)
  - Used to register public/native clients
- Web
  - Used to register confidential web clients


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.

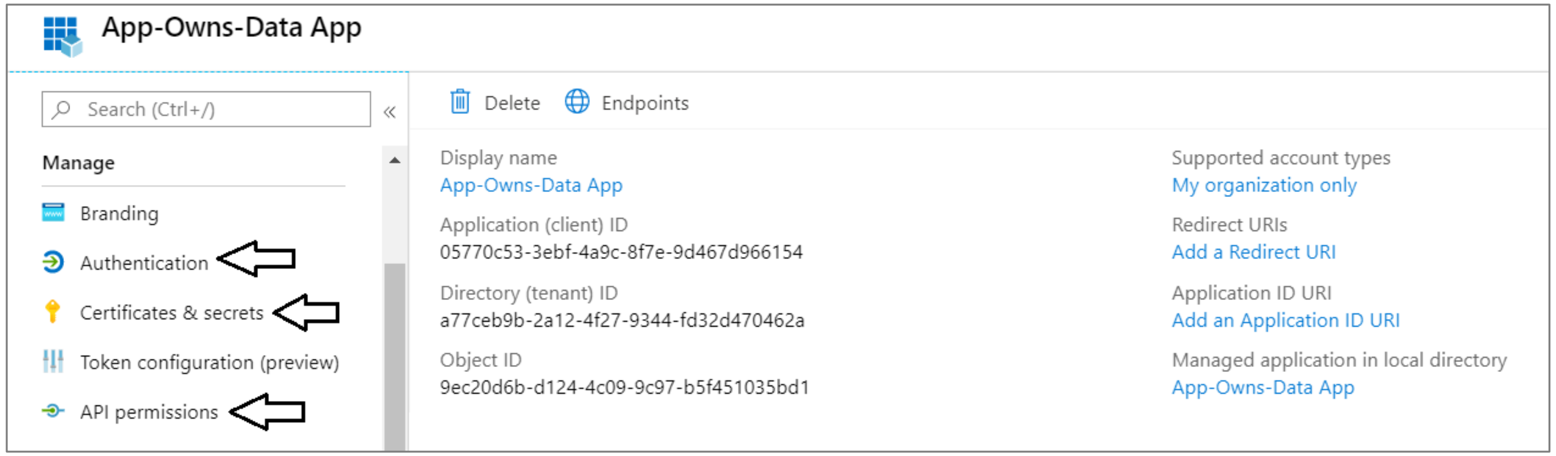
Azure AD Applications versus Service Principals

- 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



App-Owns-Data App

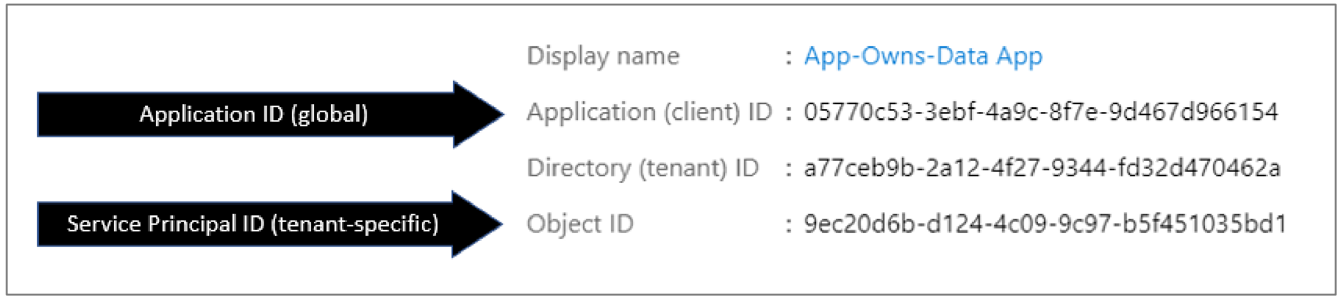
Search (Ctrl+/)

Manage
- Branding
- Authentication
- Certificates & secrets
- Token configuration (preview)
- API permissions

Delete | Endpoints

Display name	Supported account types
App-Owns-Data App	My organization only
Application (client) ID	Redirect URIs
05770c53-3ebf-4a9c-8f7e-9d467d966154	Add a Redirect URI
Directory (tenant) ID	Application ID URI
a77ceb9b-2a12-4f27-9344-fd32d470462a	Add an Application ID URI
Object ID	Managed application in local directory
9ec20d6b-d124-4c09-9c97-b5f451035bd1	App-Owns-Data App

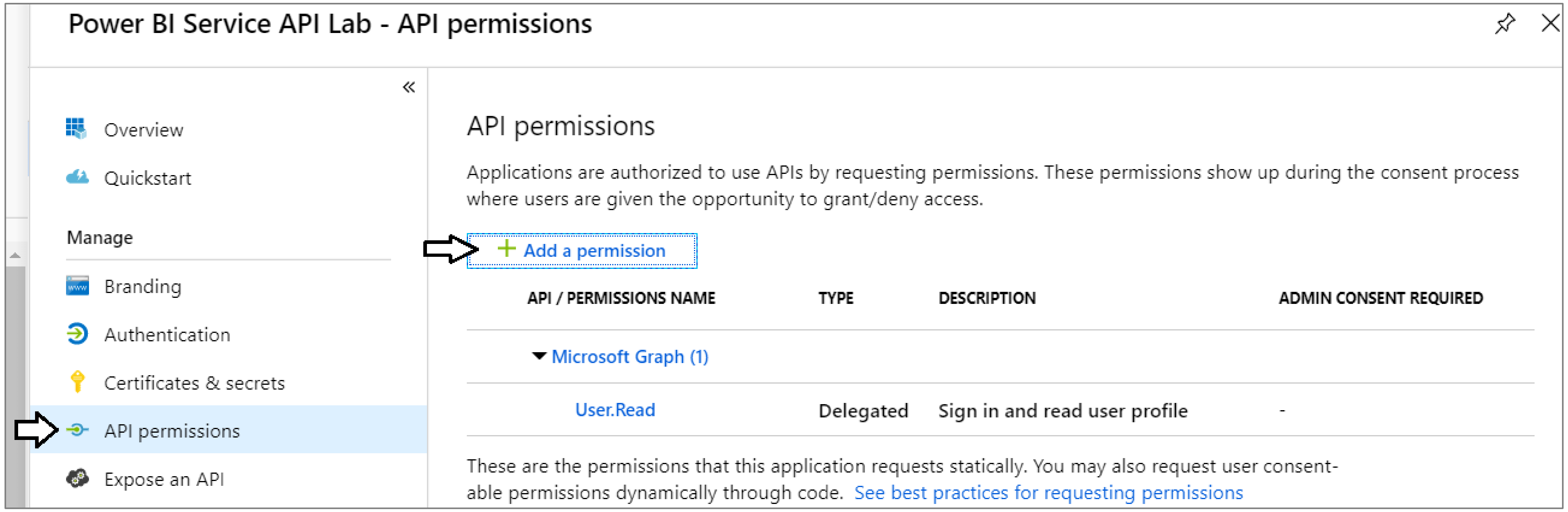
- New application generated with Application ID and Object ID



Display name	:	App-Owns-Data App
Application (client) ID	:	05770c53-3ebf-4a9c-8f7e-9d467d966154
Directory (tenant) ID	:	a77ceb9b-2a12-4f27-9344-fd32d470462a
Object ID	:	9ec20d6b-d124-4c09-9c97-b5f451035bd1

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

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.

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

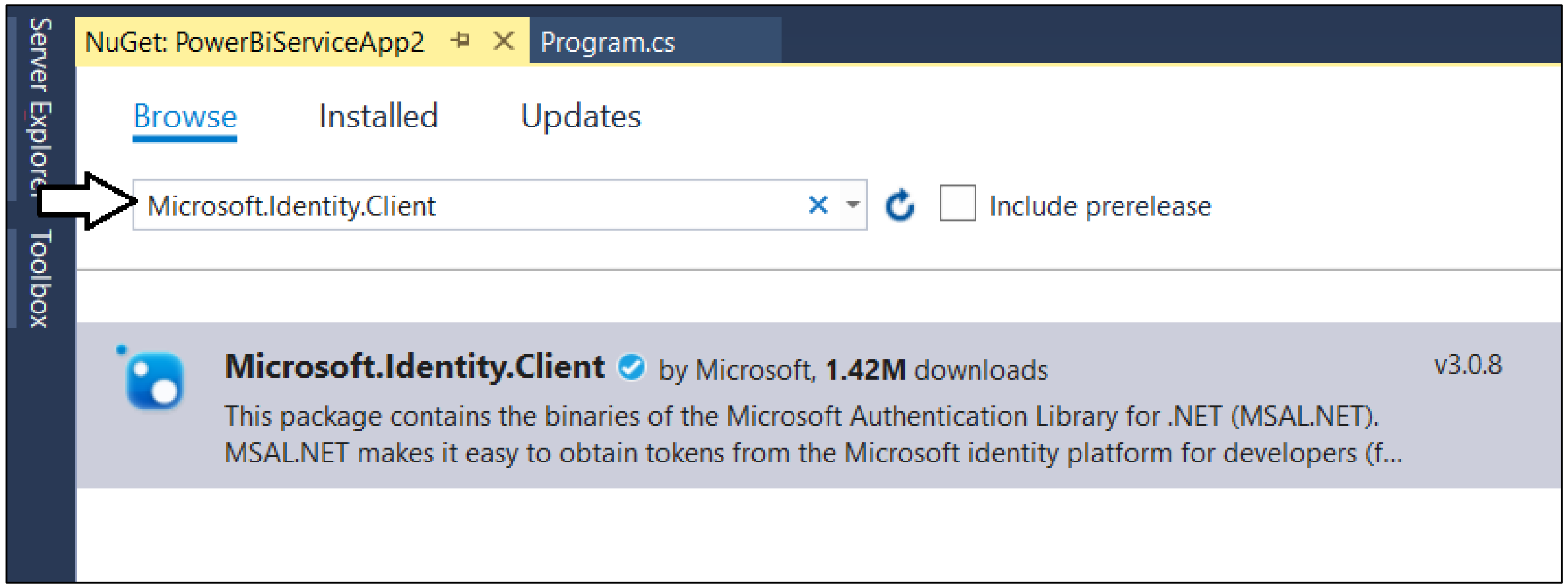
These are the permissions that this application requests statically. You may also request user consent-able permissions dynamically through code. See best practices for requesting permissions

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*
  - Provides built-in support for token caching and silent token acquisition



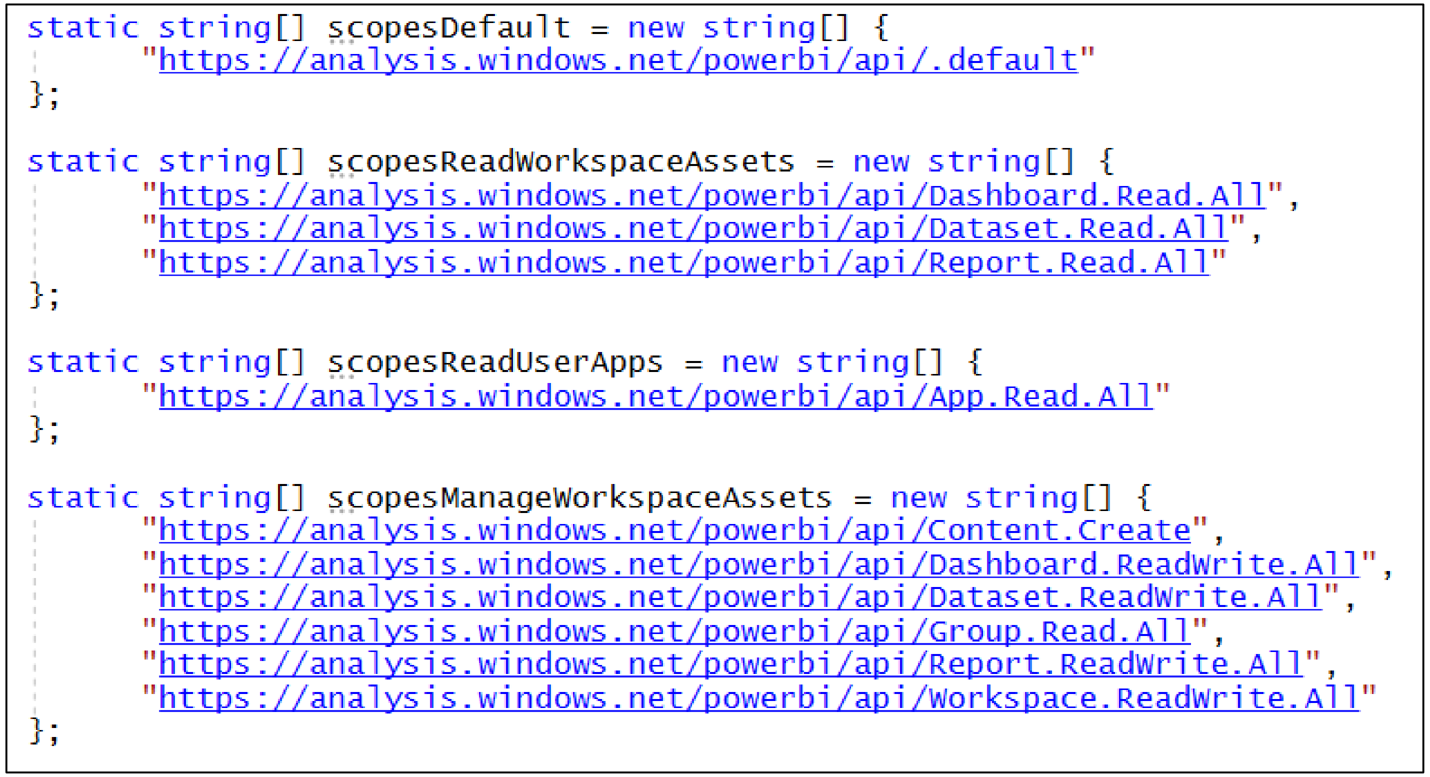
Microsoft.Identity.Client

Microsoft.Identity.Client by Microsoft, **1.42M** downloads v3.0.8

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 (f...

Power BI Service API Scopes

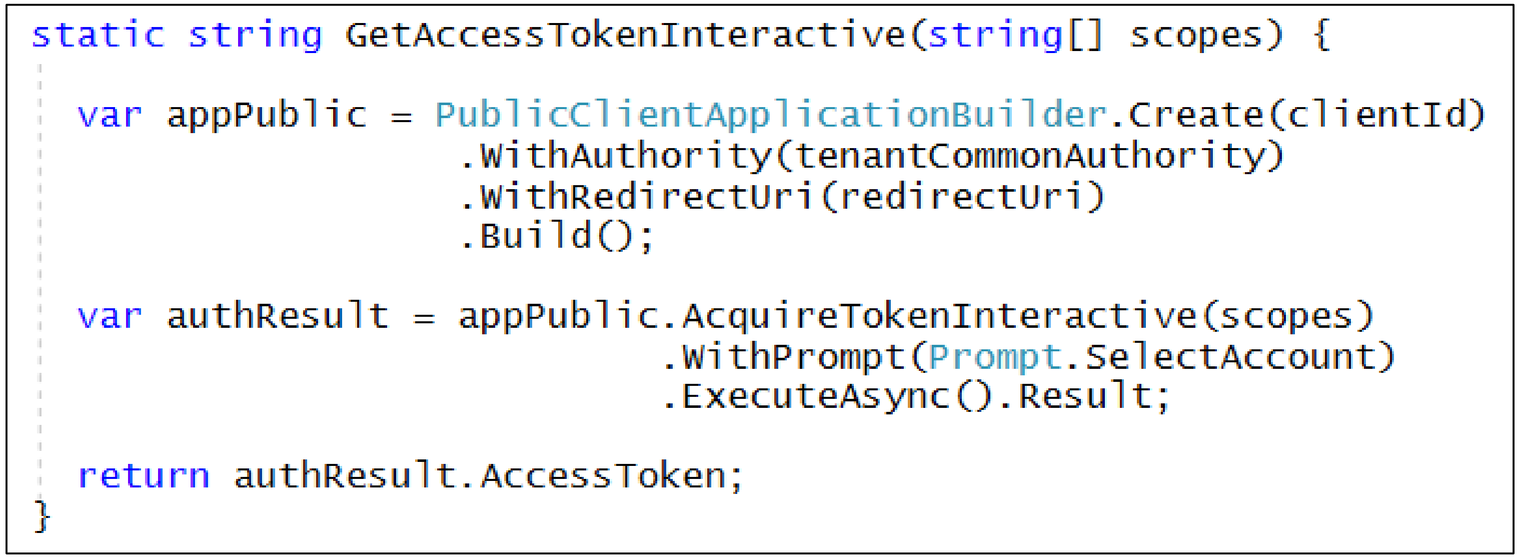
- Azure AD V2 endpoint requires passing scopes
  - Scopes define permissions required in access token
  - Scopes defined as resource + permission
    `https://analysis.windows.net/powerbi/api/` + `Report.ReadWrite.All`



```
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"
};
```

Interactive Access Token Acquisition

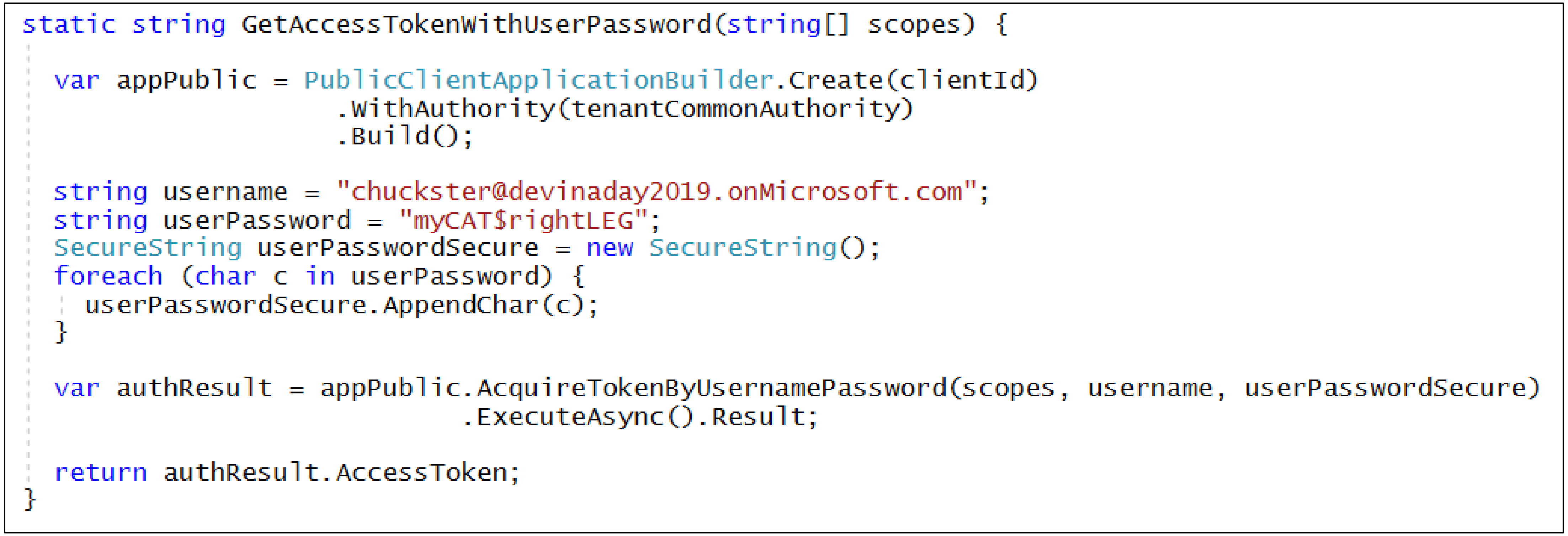
- Flow implemented using PublicClientApplication object
  - Created using PublicClientApplicationBuilder object
  - Requires passing redirect URI that matches redirect URI that is registered
  - You can control prompting behavior



```
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;
}
```

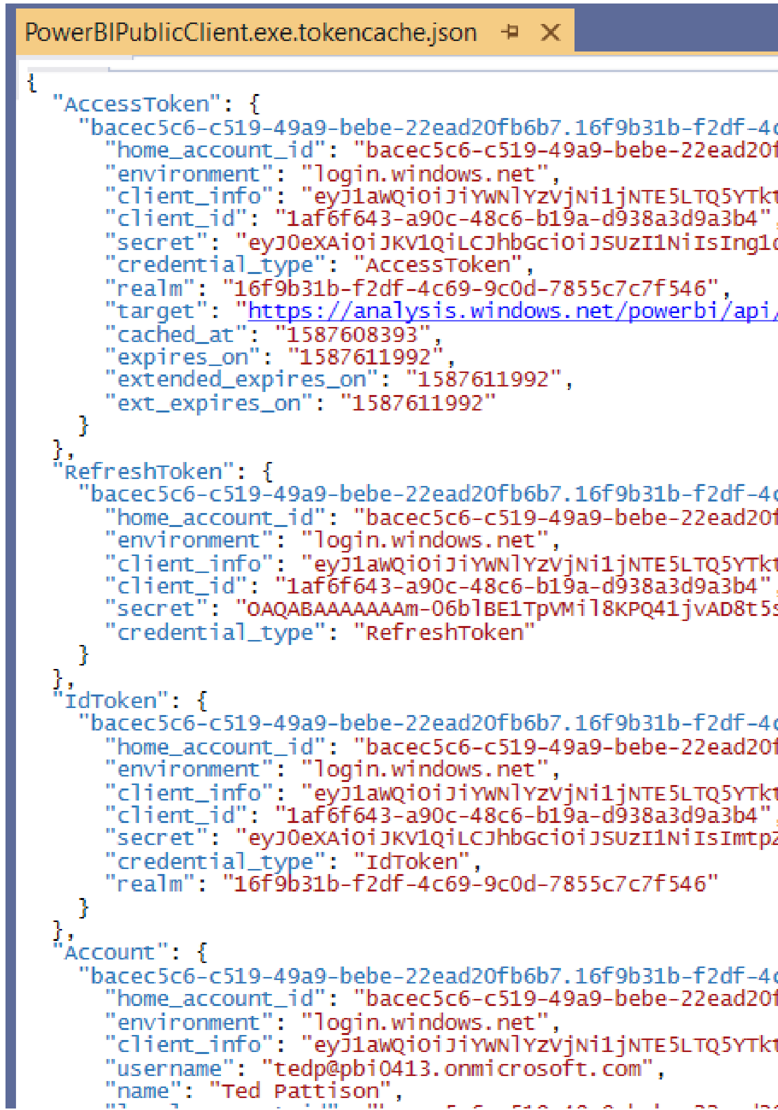
Token Caching for Public Client with MSAL

- Create a Token Cache Helper class

```
static class TokenCacheHelper {

  private static readonly string CacheFilePath = Assembly.GetExecutingAssembly().Location + ".tokencache.json";
  private static readonly object FileLock = new object();

  public static void EnableSerialization(ITokenCache tokenCache) {
    tokenCache.SetBeforeAccess(BeforeAccessNotification);
    tokenCache.SetAfterAccess(AfterAccessNotification);
  }

  private static void BeforeAccessNotification(TokenCacheNotificationArgs args) {
    lock (FileLock) {
      // repopulate token cache from persisted store
      args.TokenCache.DeserializeMsalV3(File.Exists(CacheFilePath) ? File.ReadAllBytes(CacheFilePath) : null);
    }
  }

  private static void AfterAccessNotification(TokenCacheNotificationArgs args) {
    // if the access operation resulted in a cache update
    if (args.HasStateChanged) {
      lock (FileLock) {
        // write token cache changes to persistent store
        File.WriteAllBytes(CacheFilePath, args.TokenCache.SerializeMsalV3());
      }
    }
  }
}
```

- Enable token cache for application

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

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



```
{
  "AccessToken": {
    "bacec5c6-c519-49a9-bebe-22ead20fb6b7.16f9b31b-f2df-4c
      "home_account_id": "bacec5c6-c519-49a9-bebe-22ead20f
      "environment": "login.windows.net",
      "client_info": "eyJ1aWQiOiJiYWNlYzVjNi1jNTE5LTQ5YTkt
      "client_id": "1af6f643-a90c-48c6-b19a-d938a3d9a3b4",
      "secret": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1c
      "credential_type": "AccessToken",
      "realm": "16f9b31b-f2df-4c69-9c0d-7855c7c7f546",
      "target": "https://analysis.windows.net/powerbi/api/
      "cached_at": "1587608393",
      "expires_on": "1587611992",
      "extended_expires_on": "1587611992",
      "ext_expires_on": "1587611992"
    }
  },
  "RefreshToken": {
    "bacec5c6-c519-49a9-bebe-22ead20fb6b7.16f9b31b-f2df-4c
      "home_account_id": "bacec5c6-c519-49a9-bebe-22ead20f
      "environment": "login.windows.net",
      "client_info": "eyJ1aWQiOiJiYWNlYzVjNi1jNTE5LTQ5YTkt
      "client_id": "1af6f643-a90c-48c6-b19a-d938a3d9a3b4",
      "secret": "OAQABAAAAAAAm-06blBE1TpVMil8KPQ41jvAD8t5s
      "credential_type": "RefreshToken"
    }
  },
  "IdToken": {
    "bacec5c6-c519-49a9-bebe-22ead20fb6b7.16f9b31b-f2df-4c
      "home_account_id": "bacec5c6-c519-49a9-bebe-22ead20f
      "environment": "login.windows.net",
      "client_info": "eyJ1aWQiOiJiYWNlYzVjNi1jNTE5LTQ5YTkt
      "client_id": "1af6f643-a90c-48c6-b19a-d938a3d9a3b4",
      "secret": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZ
      "credential_type": "IdToken",
      "realm": "16f9b31b-f2df-4c69-9c0d-7855c7c7f546"
    }
  },
  "Account": {
    "bacec5c6-c519-49a9-bebe-22ead20fb6b7.16f9b31b-f2df-4c
      "home_account_id": "bacec5c6-c519-49a9-bebe-22ead20f
      "environment": "login.windows.net",
      "client_info": "eyJ1aWQiOiJiYWNlYzVjNi1jNTE5LTQ5YTkt
      "username": "tedp@pbi0413.onmicrosoft.com",
      "name": "Ted Pattison",
```

Acquiring Access Token from the Token Cache

```csharp
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”

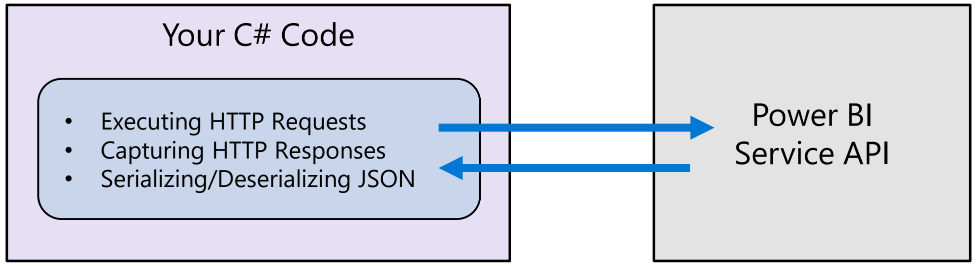
- What can you do with the Power BI Service API?
  - Publish PBIX project files
  - Update connection details and datasource credentials
  - Create workspaces and clone content across workspaces
  - Embed Power BI reports and dashboards tiles in web pages
  - Create streaming datasets in order to build real-time dashboards

Calling the Power BI Service API

Direct REST calls without using the Power BI .NET SDK

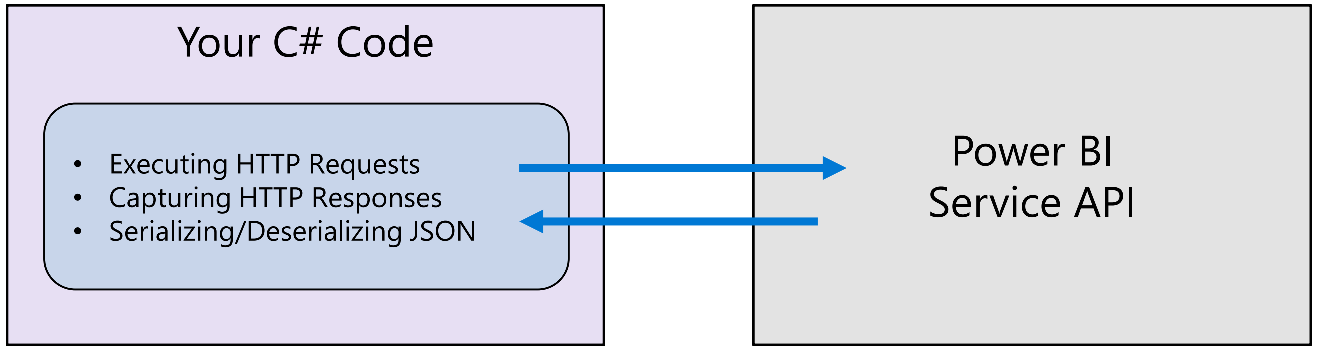
```
static string ExecuteGetRequest(string restUrl) {
  HttpClient client = new HttpClient();
  HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, restUrl);
  request.Headers.Add("Authorization", "Bearer " + GetAccessToken());
  request.Headers.Add("Accept", "application/json;odata.metadata=minimal");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

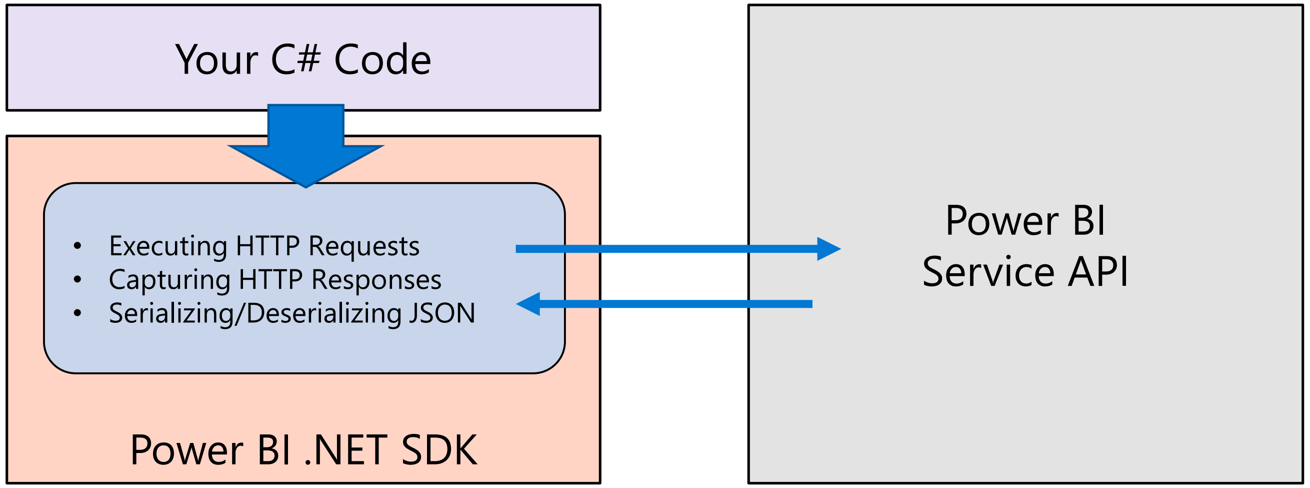

Your C# Code
• Executing HTTP Requests
• Capturing HTTP Responses
• Serializing/Deserializing JSON
Power BI Service API

Power BI .NET SDK

- Developing without the Power BI .NET SDK

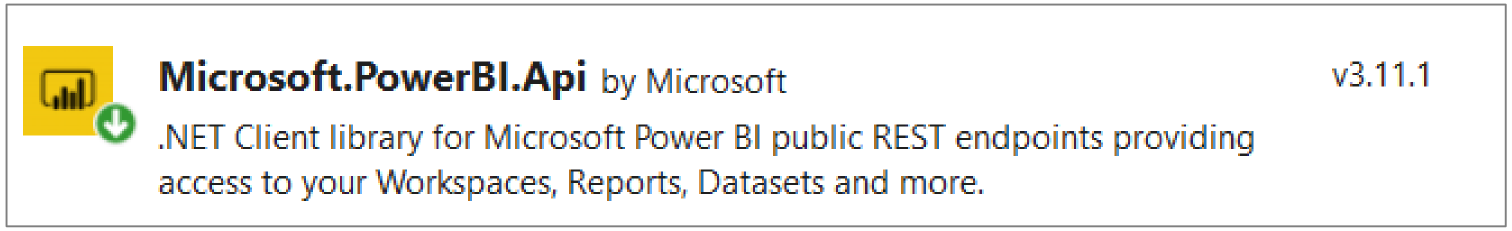

Your C# Code
• Executing HTTP Requests
• Capturing HTTP Responses
• Serializing/Deserializing JSON
Power BI Service API

- Developing with the Power BI .NET SDK


Your C# Code
• Executing HTTP Requests
• Capturing HTTP Responses
• Serializing/Deserializing JSON
Power BI .NET SDK
Power BI Service API

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



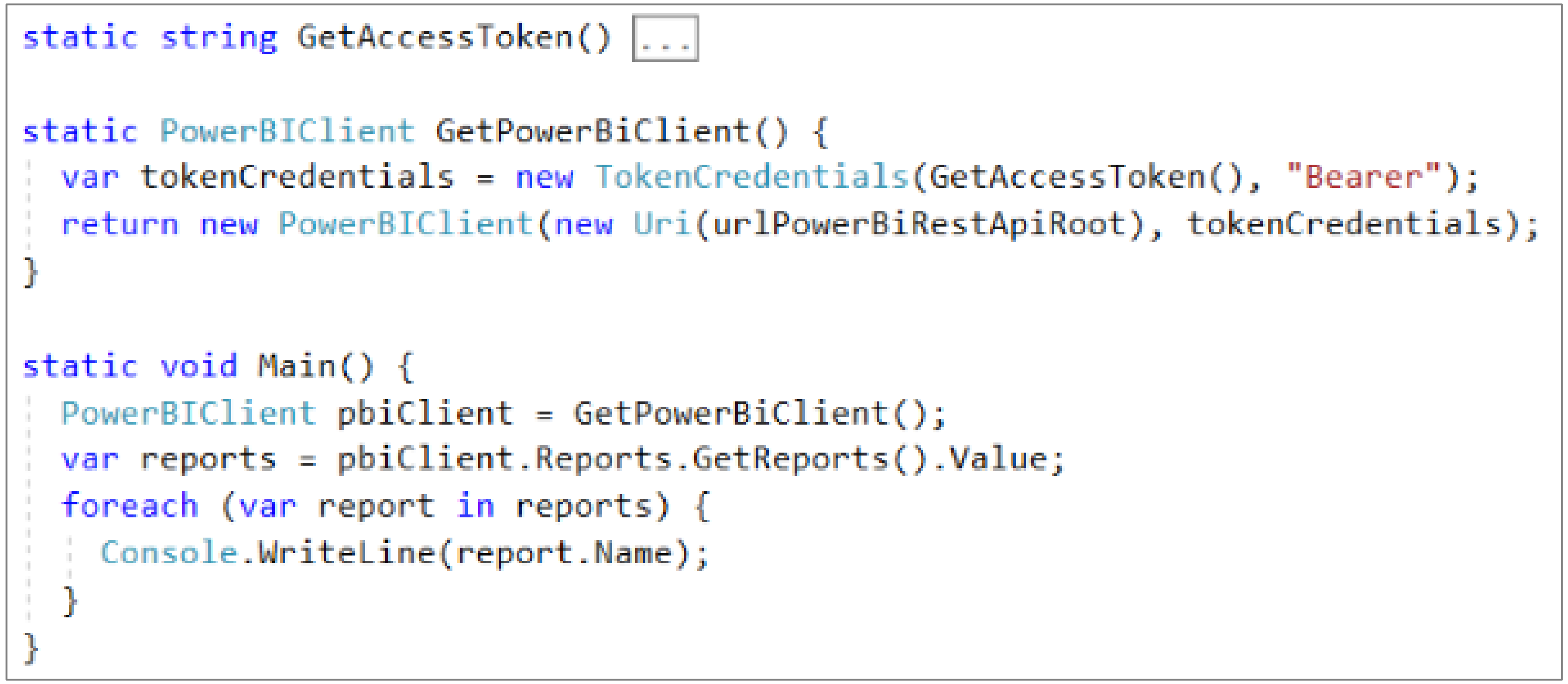
Microsoft.PowerBI.Api by Microsoft v3.11.1

.NET Client library for Microsoft Power BI public REST endpoints providing access to your Workspaces, Reports, Datasets and more.

- 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

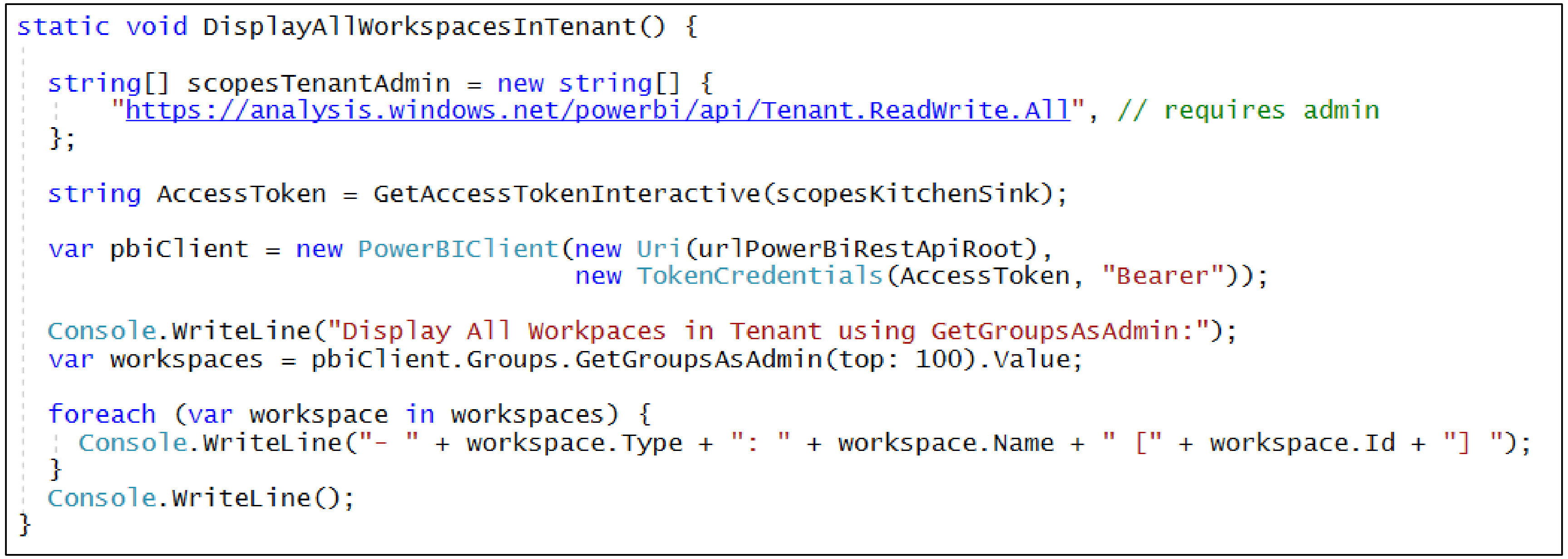
- Power BI User APIs (e.g. GetGroupsAsync)
  - provides users with access to personal workspace
  - provides users with access to app workspaces
  - provides service principal (SP) with access to app workspaces

- 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
- Not supported when calling as service principal with app-only token



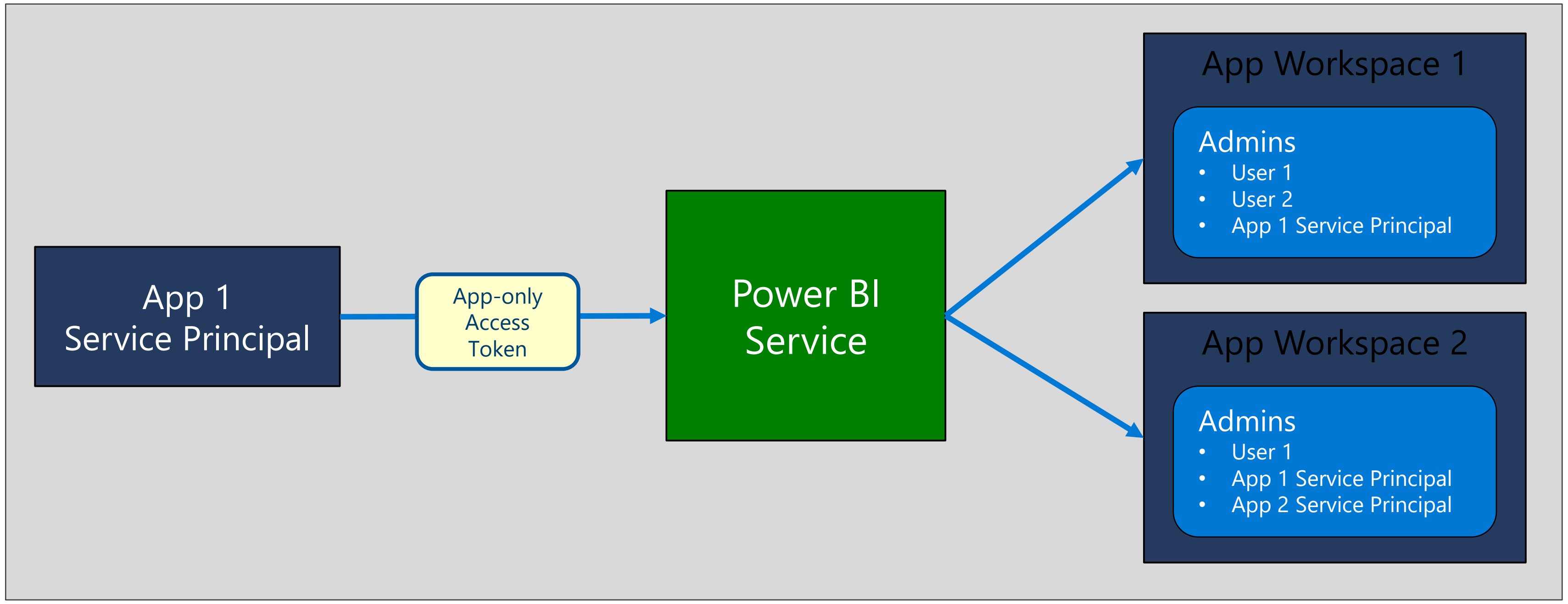
```
static void DisplayAllWorkspacesInTenant() {

  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 Workpaces 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();
}
```

Agenda

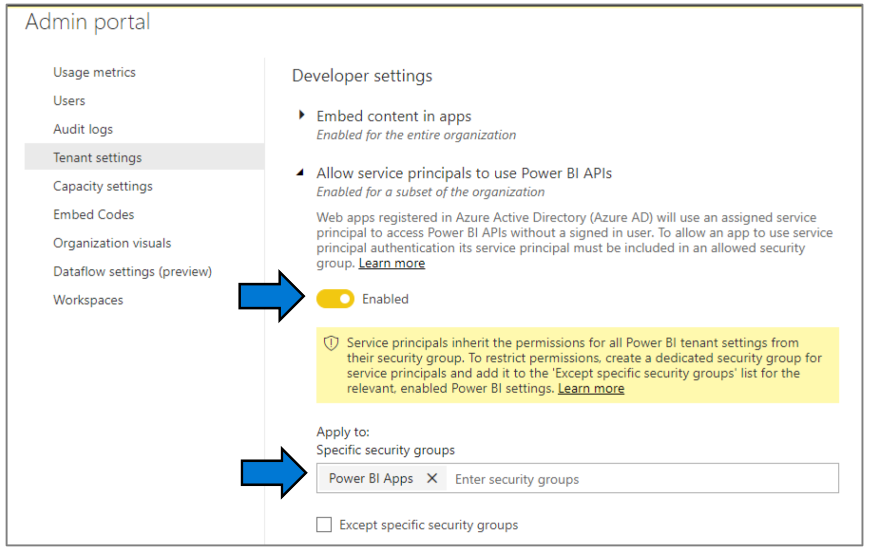
- ✓ OAuth 2.0 and OpenID Connect
- ✓ Microsoft Identity Platform 2.0
- ✓ Developing with the Power BI .NET SDK
- ➢ Developing with the App-Owns-Data Model
- • Developing Single Page Applications (SPAs)
- • Developing Secure Web Applications

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 <u>NOT</u> based on Azure AD permissions



Tenant Setup



Admin portal

Usage metrics
Users
Audit logs
Tenant settings
Capacity settings
Embed Codes
Organization visuals
Dataflow settings (preview)
Workspaces

Developer settings

Embed content in apps
Enabled for the entire organization

Allow service principals to use Power BI APIs
Enabled for a subset of the organization

Web apps registered in Azure Active Directory (Azure AD) will use an assigned service principal to access Power BI APIs without a signed in user. To allow an app to use service principal authentication its service principal must be included in an allowed security group. Learn more

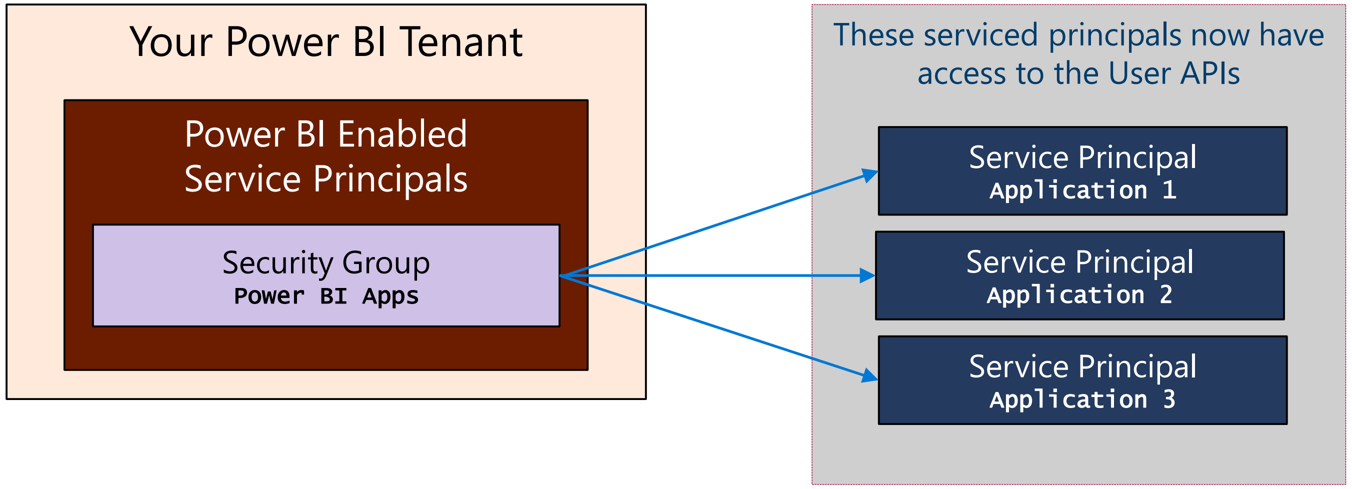
Enabled

Service principals inherit the permissions for all Power BI tenant settings from their security group. To restrict permissions, create a dedicated security group for service principals and add it to the 'Except specific security groups' list for the relevant, enabled Power BI settings. Learn more

Apply to:
Specific security groups

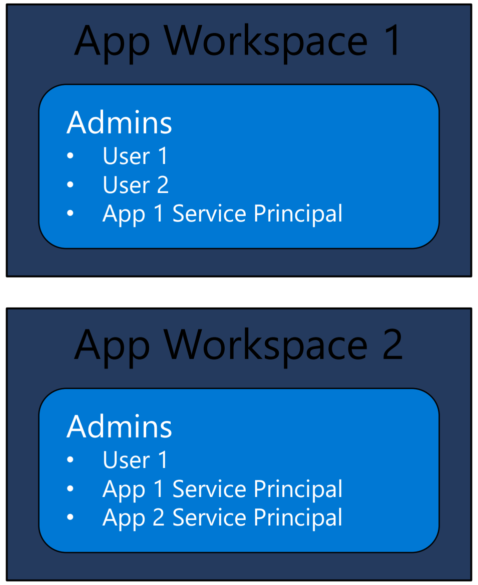
Power BI Apps ✕ Enter security groups

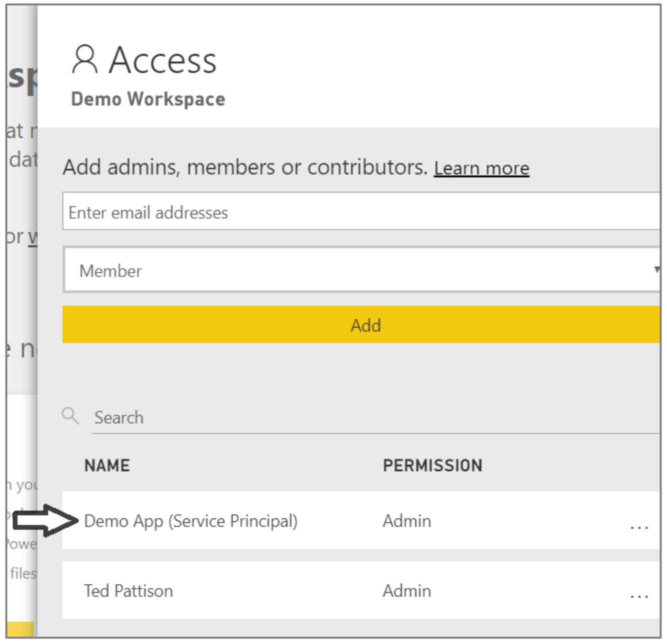
☐ Except specific security groups


Your Power BI Tenant
Power BI Enabled Service Principals
Security Group
Power BI Apps
These serviced principals now have access to the User APIs
Service Principal Application 1
Service Principal Application 2
Service Principal Application 3

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



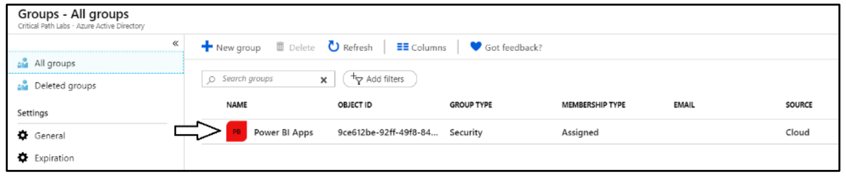


Add admins, members or contributors. Learn more

NAME	PERMISSION
Demo App (Service Principal)	Admin
Ted Pattison	Admin

Setting Up for App-Owns-Data – Part 1

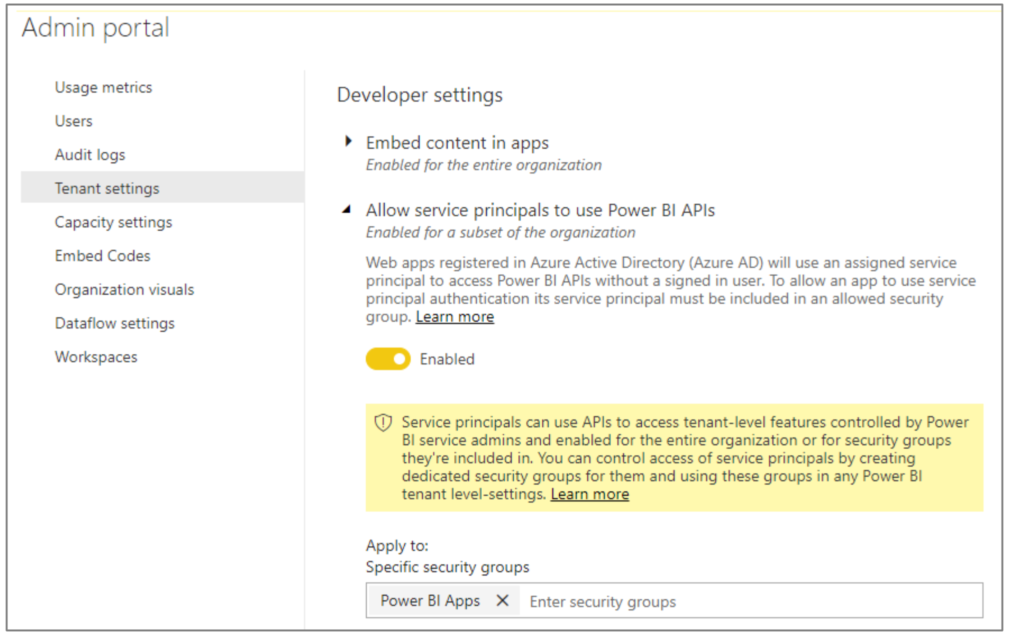
- Enable Service Principal Access to Power BI Service API
  - Create an Azure AD security group (e.g. Power BI Apps)



Groups - All groups
Critical Path Labs - Azure Active Directory

NAME	OBJECT ID	GROUP TYPE	MEMBERSHIP TYPE	EMAIL	SOURCE
PB Power BI Apps	9ce612be-92ff-49f8-84...	Security	Assigned		Cloud

  - Add group to *Power BI Allow service principals to use Power BI APIs*



Admin portal

Developer settings

Embed content in apps
Enabled for the entire organization

Allow service principals to use Power BI APIs
Enabled for a subset of the organization

Web apps registered in Azure Active Directory (Azure AD) will use an assigned service principal to access Power BI APIs without a signed in user. To allow an app to use service principal authentication its service principal must be included in an allowed security group. Learn more

Enabled

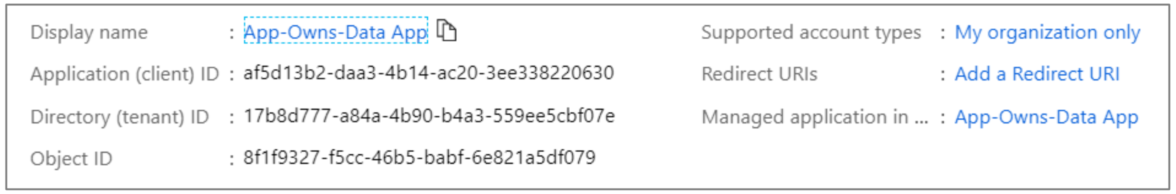
Service principals can use APIs to access tenant-level features controlled by Power BI service admins and enabled for the entire organization or for security groups they're included in. You can control access of service principals by creating dedicated security groups for them and using these groups in any Power BI tenant level-settings. Learn more

Apply to:
Specific security groups

Power BI Apps ✕ Enter security groups

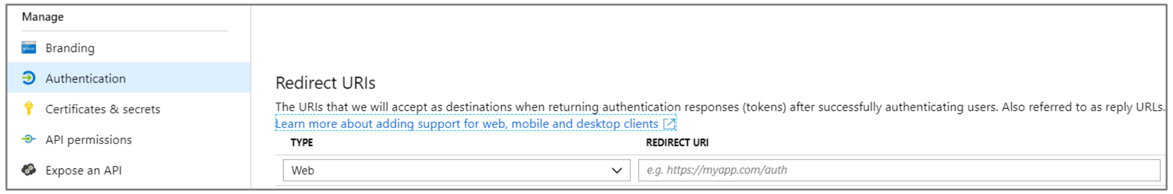
Setting Up for App-Owns-Data – Part 2

- Create a confidential client in your Azure AD tenant



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

- Configured as `TYPE=Web` and no need for a redirect URL

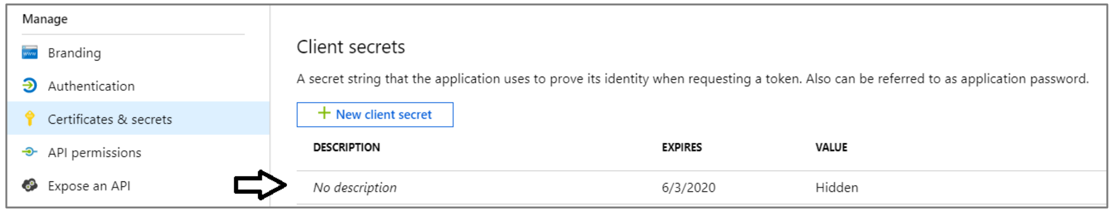


Redirect URIs

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. https://myapp.com/auth

- Add a client secret or a client certificate

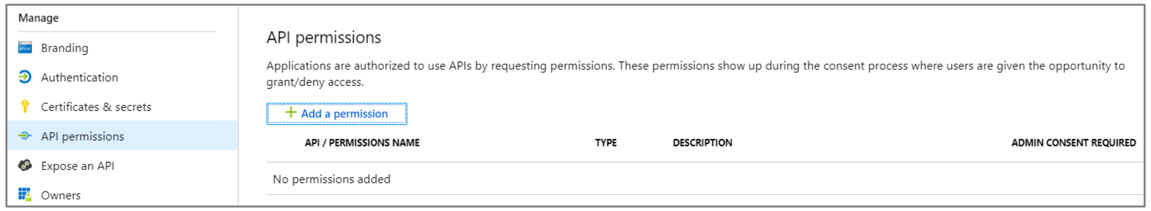


Client secrets

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

DESCRIPTION	EXPIRES	VALUE
No description	6/3/2020	Hidden

- No need to configure any permissions



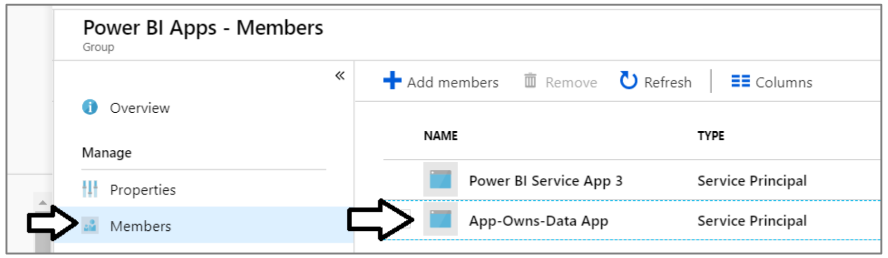
API permissions

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.

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

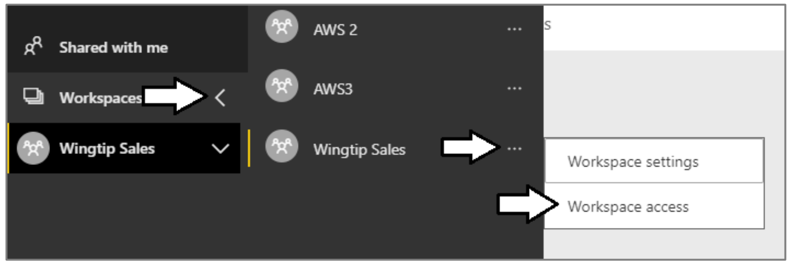
Setting Up for App-Owns-Data – Part 3

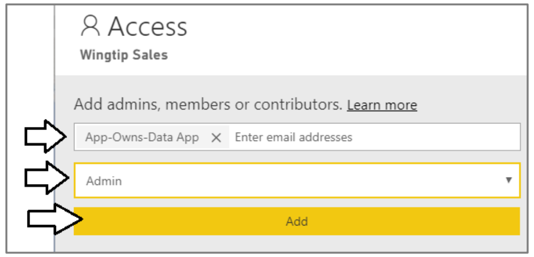
- Add application's service principal in **Power BI Apps** security group



NAME	TYPE
Power BI Service App 3	Service Principal
App-Owns-Data App	Service Principal

- Configure application's service principal as workspace admin




Access
Wingtip Sales

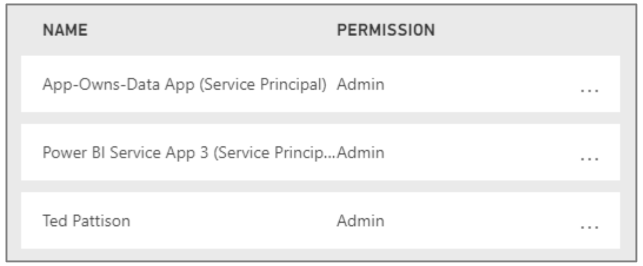
Add admins, members or contributors. Learn more

App-Owns-Data App ✕ Enter email addresses

Admin

Add

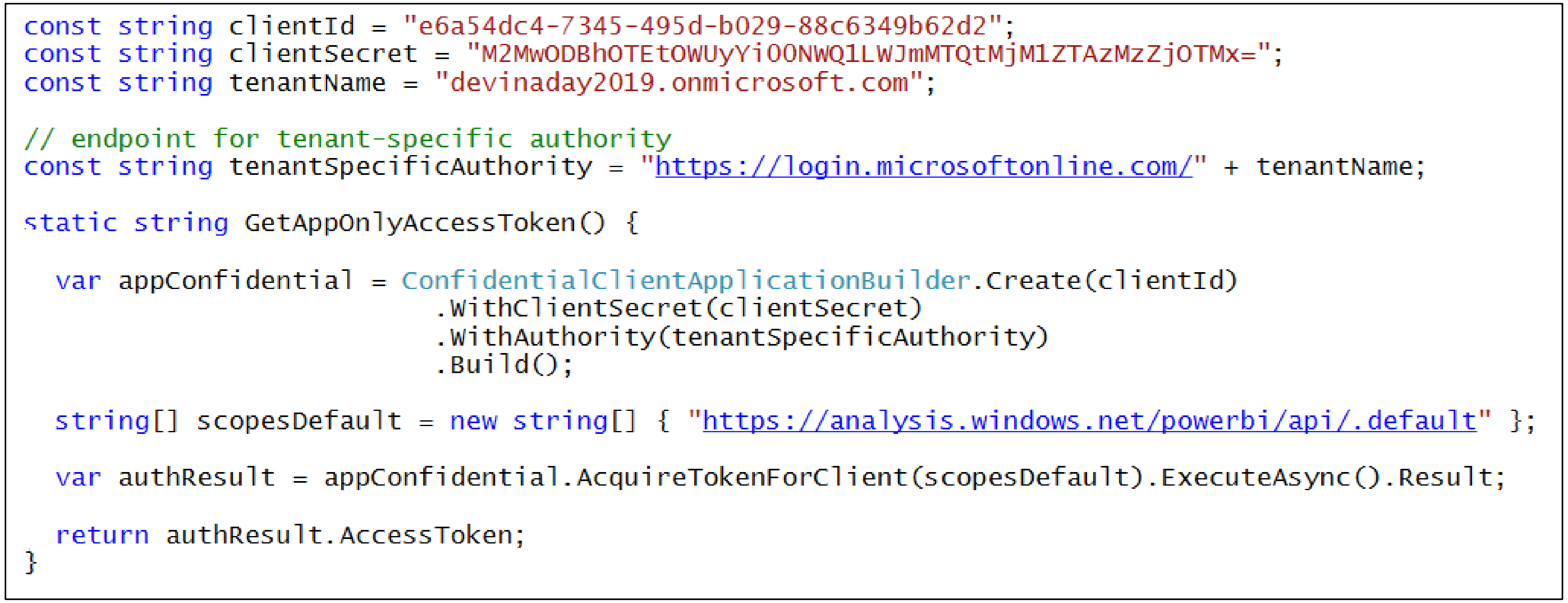
- Service principal should now be workspace admin



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

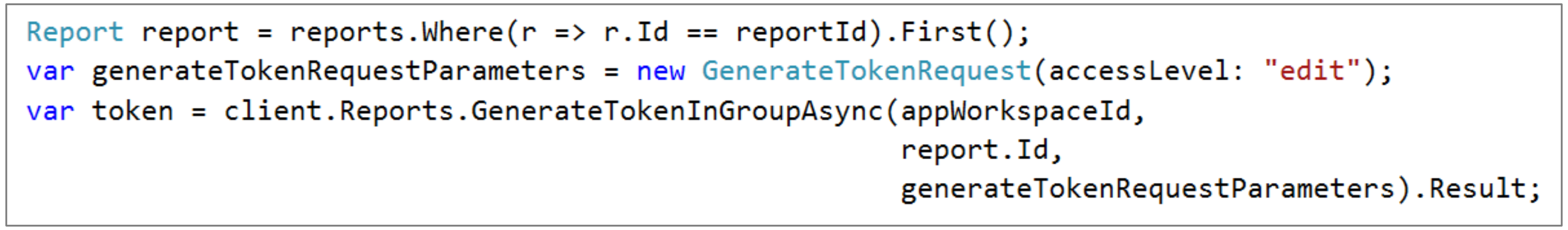


```
const string clientId = "e6a54dc4-7345-495d-b029-88c6349b62d2";
const string clientSecret = "M2MwODBhOTEtOWUyYi00NWQ1LWJmMTQtMjM1ZTAzMzZjOTMx=";
const string tenantName = "devinaday2019.onmicrosoft.com";

// 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

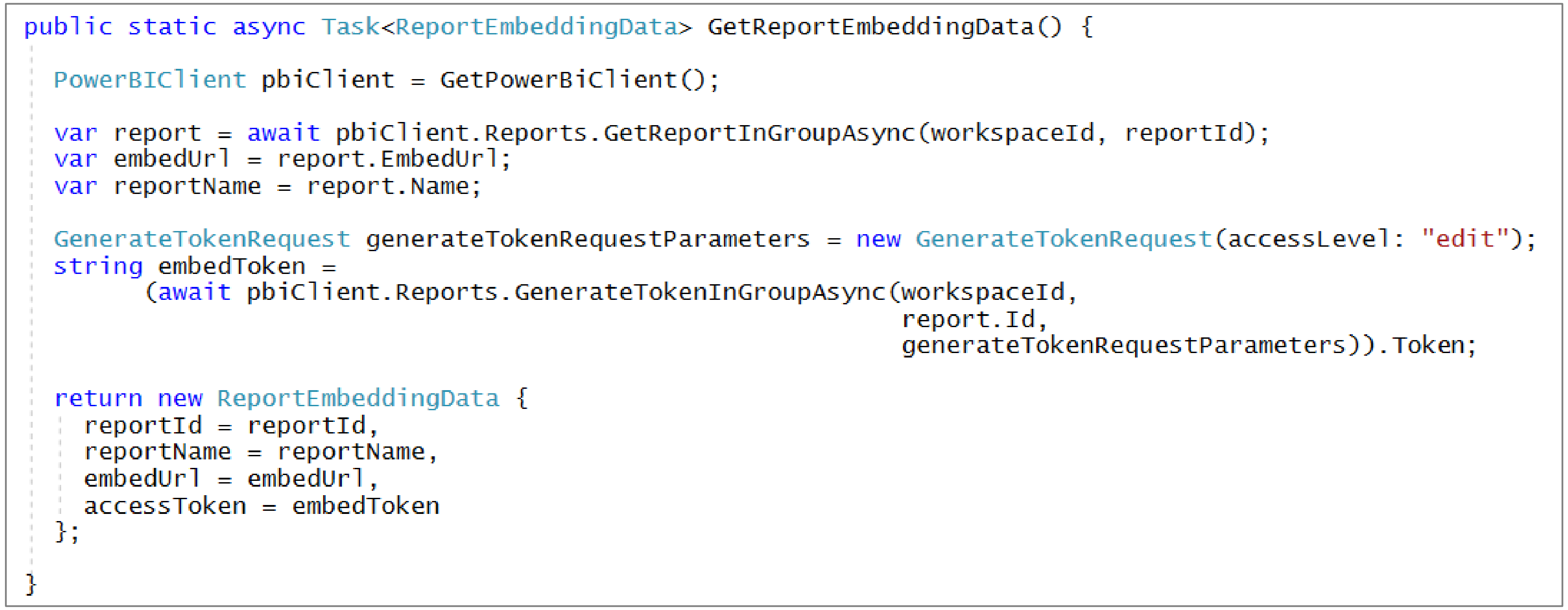
- You can embed reports using an AAD token, but…
  - You might want embed resource using more restricted tokens
  - You might want stay within the bounds of Power BI licensing terms

- 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



```csharp
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

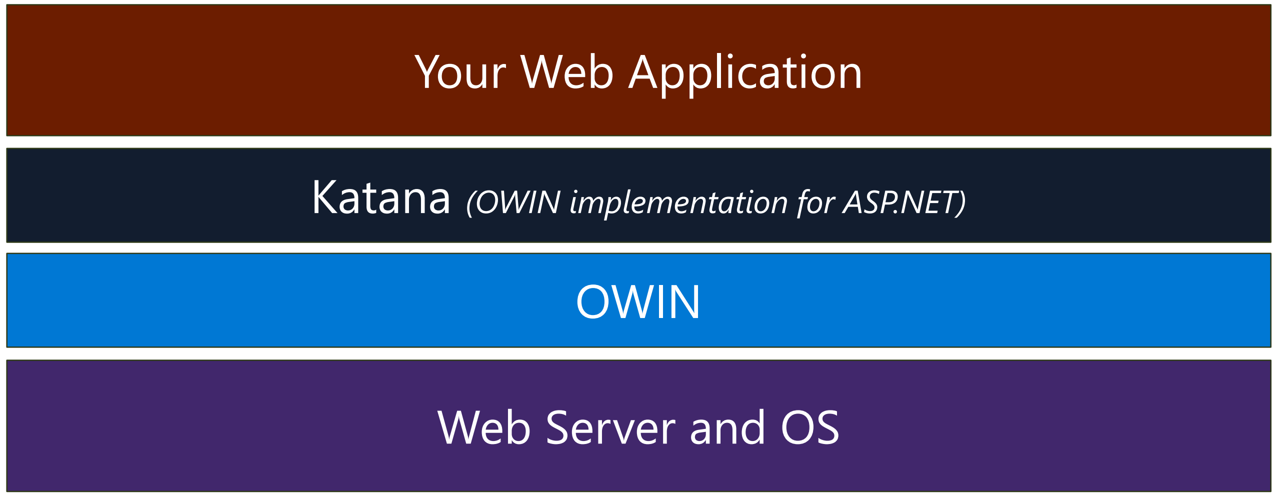
- Used in single page applications (SPAs)
  - All code and configuration loaded into client-side browser
  - Application runs as web application at verifiable URL on the Internet
  - SPA must be running at URL registered as a redirect URL
  - Application isn't really confidential - it cannot keep any 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

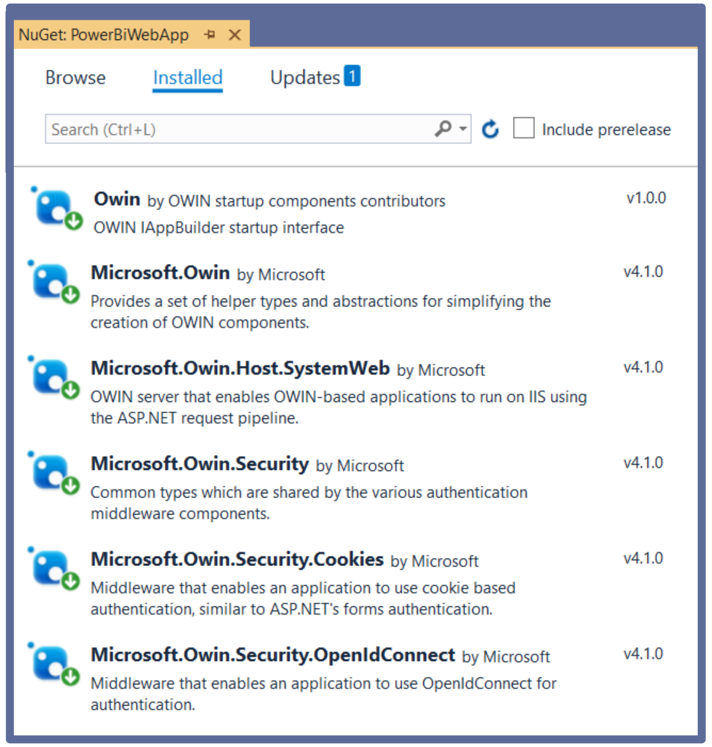
Agenda

- ✓ 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 Core
  - Microsoft provides OWIN-based security middleware





Owin by OWIN startup components contributors v1.0.0
OWIN IAppBuilder startup interface

Microsoft.Owin by Microsoft v4.1.0
Provides a set of helper types and abstractions for simplifying the creation of OWIN components.

Microsoft.Owin.Host.SystemWeb by Microsoft v4.1.0
OWIN server that enables OWIN-based applications to run on IIS using the ASP.NET request pipeline.

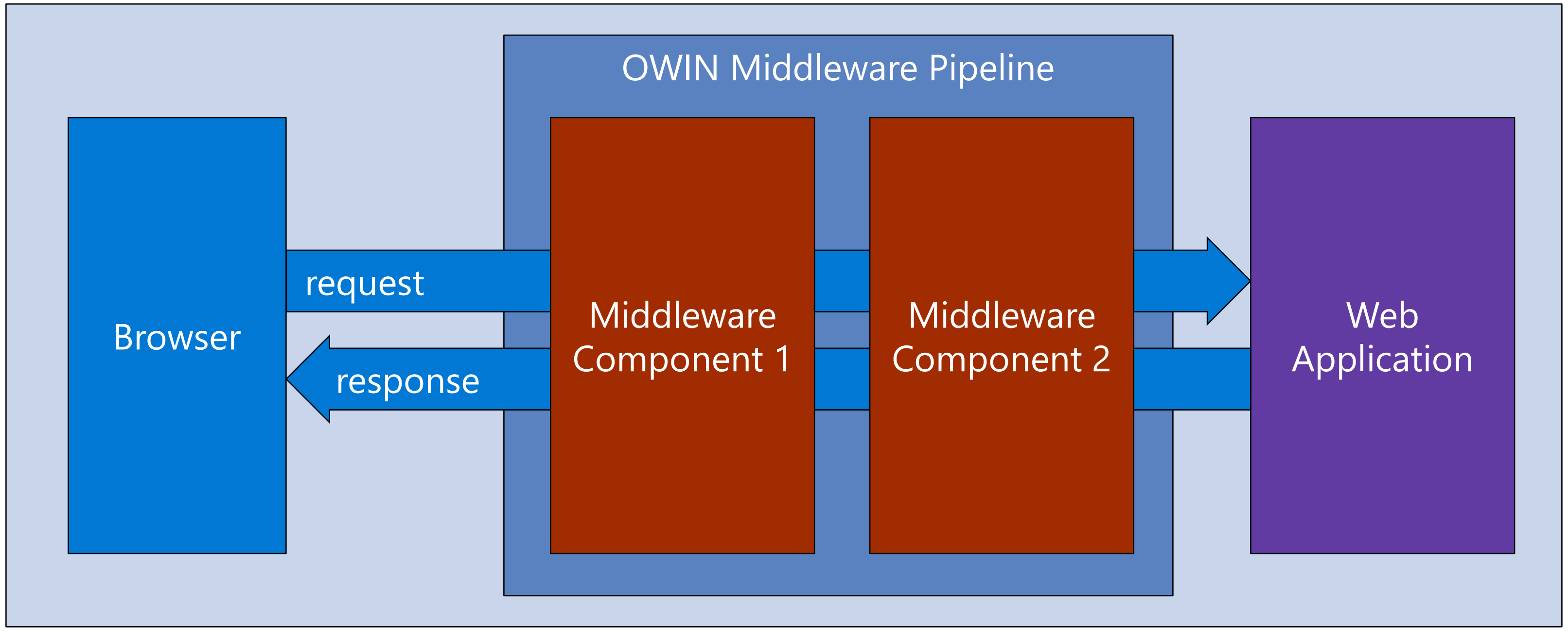
Microsoft.Owin.Security by Microsoft v4.1.0
Common types which are shared by the various authentication middleware components.

Microsoft.Owin.Security.Cookies by Microsoft v4.1.0
Middleware that enables an application to use cookie based authentication, similar to ASP.NET's forms authentication.

Microsoft.Owin.Security.OpenIdConnect by Microsoft v4.1.0
Middleware that enables an application to use OpenIdConnect for authentication.

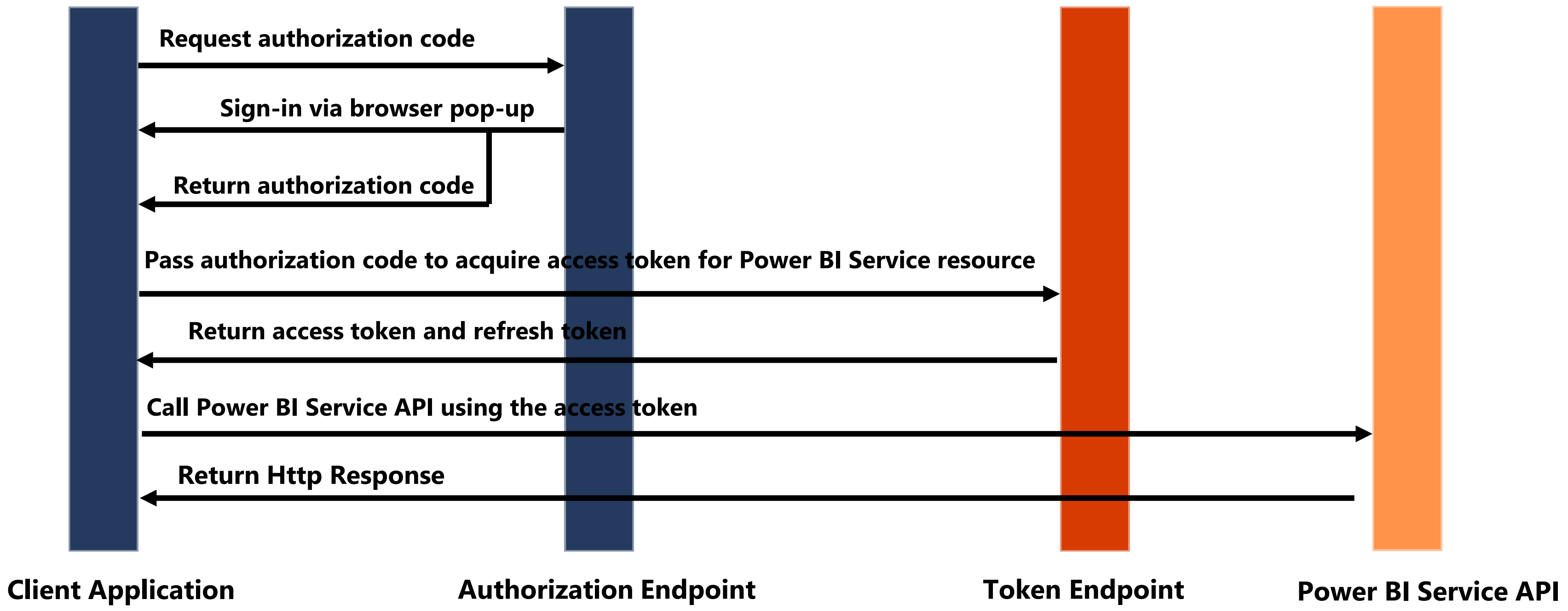
OWIN Middleware Modules

- 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



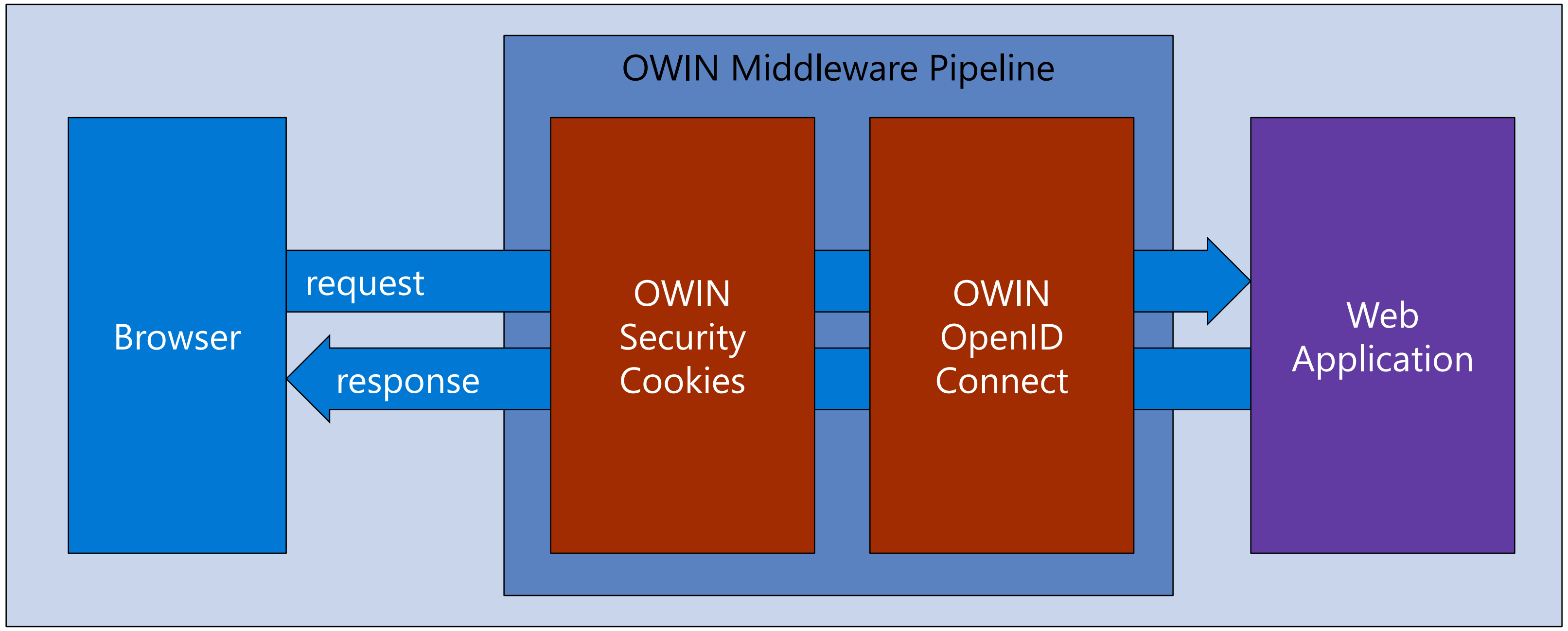
Authorization Code Grant Flow

- 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


Request authorization code
Sign-in via browser pop-up
Return authorization code
Pass authorization code to acquire access token for Power BI Service resource
Return access token and refresh token
Call Power BI Service API using the access token
Return Http Response
Client Application
Authorization Endpoint
Token Endpoint
Power BI Service API

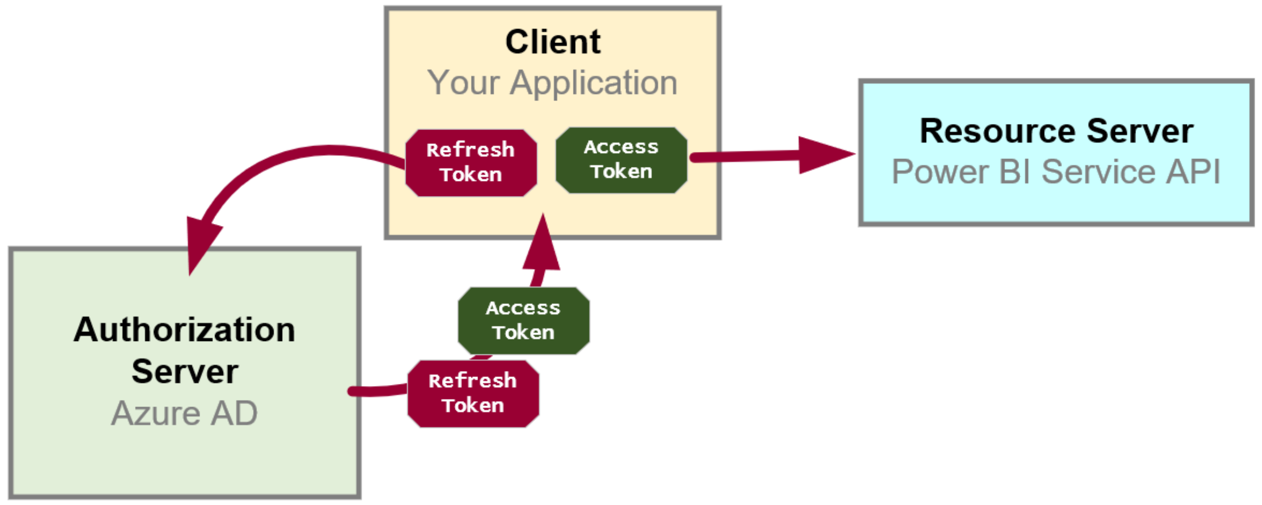
OWIN OpenID Connect Module

- 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



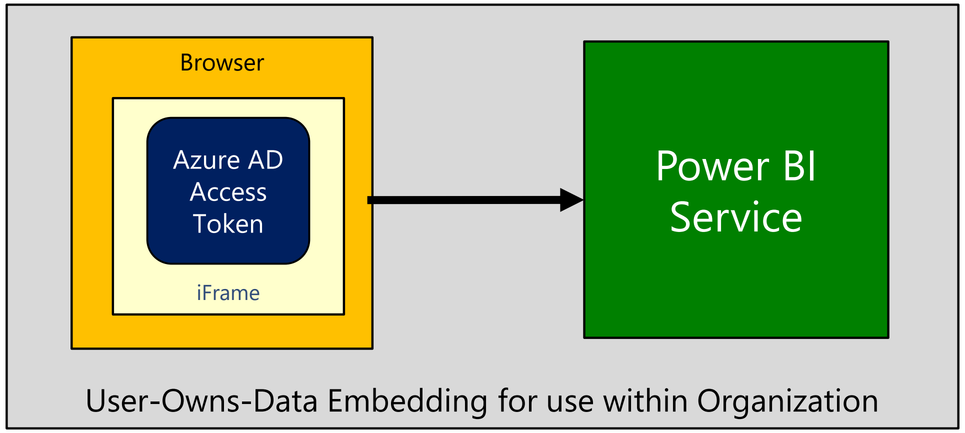
Summarization of Authentication Flows

- Public clients
  - Interactive Flow
  - User Password Credential Flow

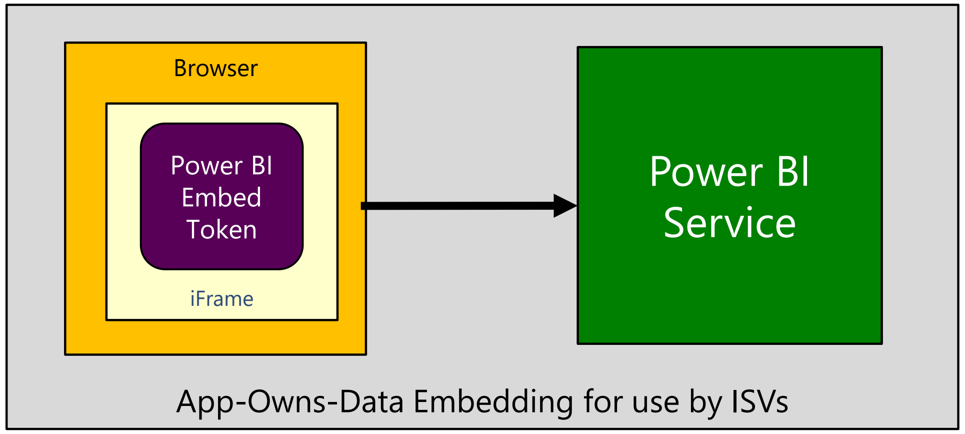
- Confidential web clients
  - Client Credentials Flow
  - Implicit Flow
  - Authorization Code Flow

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