Microsoft

Microsoft Business Applications Summit

June 10–11, 2019 | Atlanta, GA

Microsoft

Developing with Power BI Embedding

Ted Pattison
Developer, Instructor and MVP

Microsoft Business
Applications Summit

Please Take Our Pre-Session Survey

From your laptop, navigate to this URL

- https://aka.ms/DevIADSurvey

From your mobile device, scan this image

Who Is Ted Pattison?

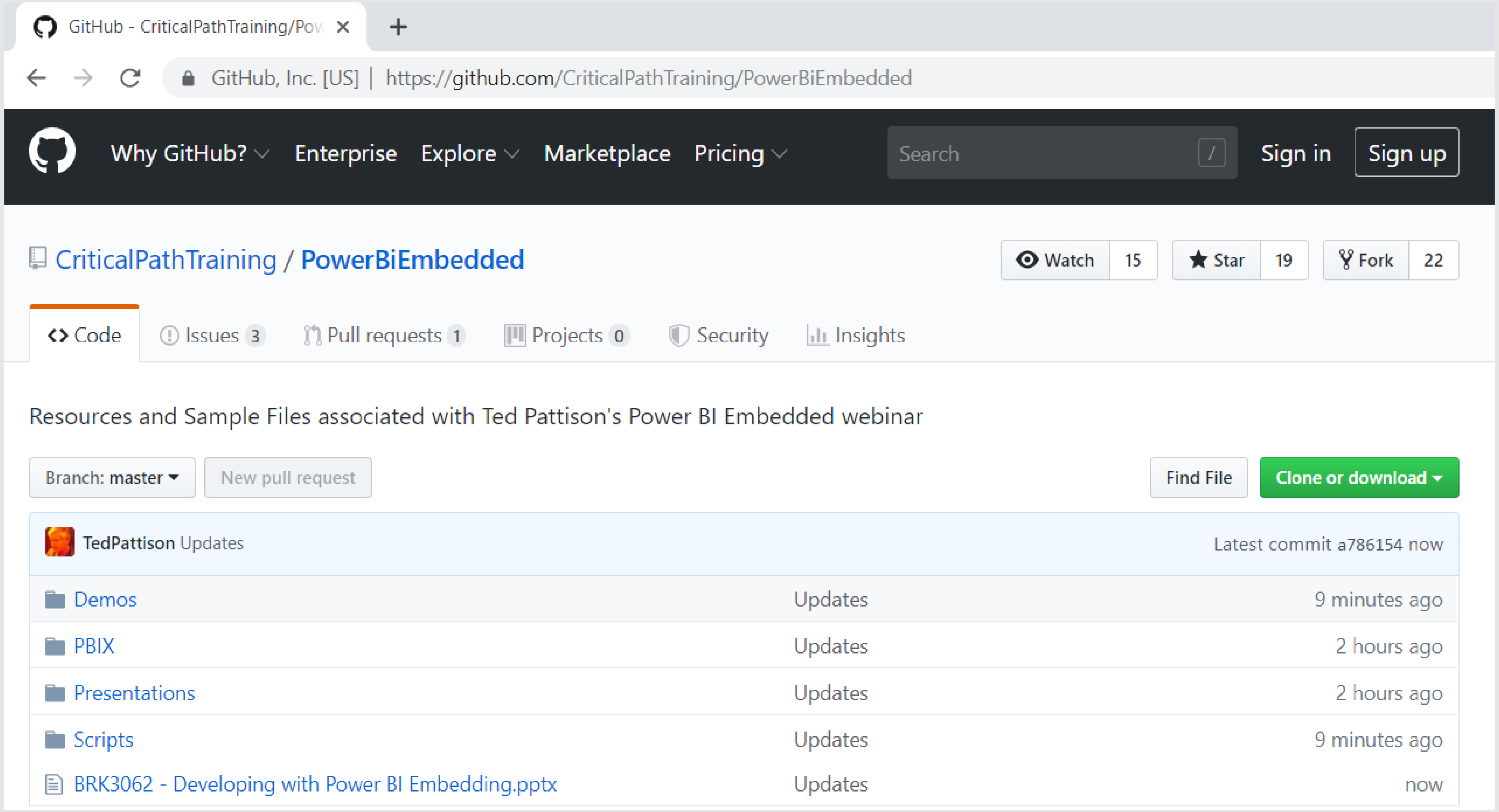
ted.pattison@criticalpathtraining.com

- Developer, Instructor and Author
- 14-time MVP – Power BI, PowerApps, Flow and SharePoint
- Leader of Tampa PowerApps-Flow User Group
- Owner of **Critical Path Training**
- Ted teaches these courses
  - Power BI Certification Bootcamp
  - Power BI Developer in a Day
  - Power BI Developer Bootcamp
  - Building Business Solutions with PowerApps and Flow
- More info at https://www.CriticalPathTraining.com

CriticalPath TRAINING

Get the Slides and Sample Code

https://github.com/CriticalPathTraining/PowerBiEmbedded



CriticalPathTraining / PowerBiEmbedded

Resources and Sample Files associated with Ted Pattison's Power BI Embedded webinar

TedPattison Updates — Latest commit a786154 now

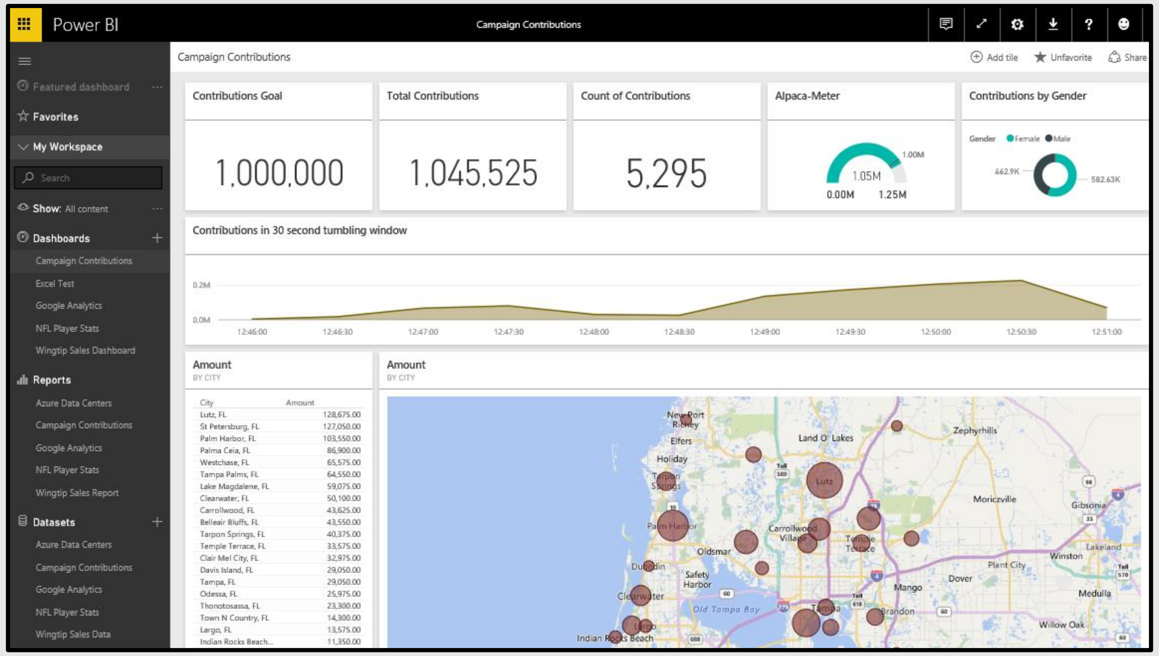
Demos	Updates	9 minutes ago
PBIX	Updates	2 hours ago
Presentations	Updates	2 hours ago
Scripts	Updates	9 minutes ago
BRK3062 - Developing with Power BI Embedding.pptx	Updates	now

Agenda

- Power BI Embedding Fundamentals
- Authentication with Azure AD
- Programming the Power BI Service API
- App-only Authentication
- Single Page Applications (SPAs) with React.js
- Programming the Power BI JavaScript API

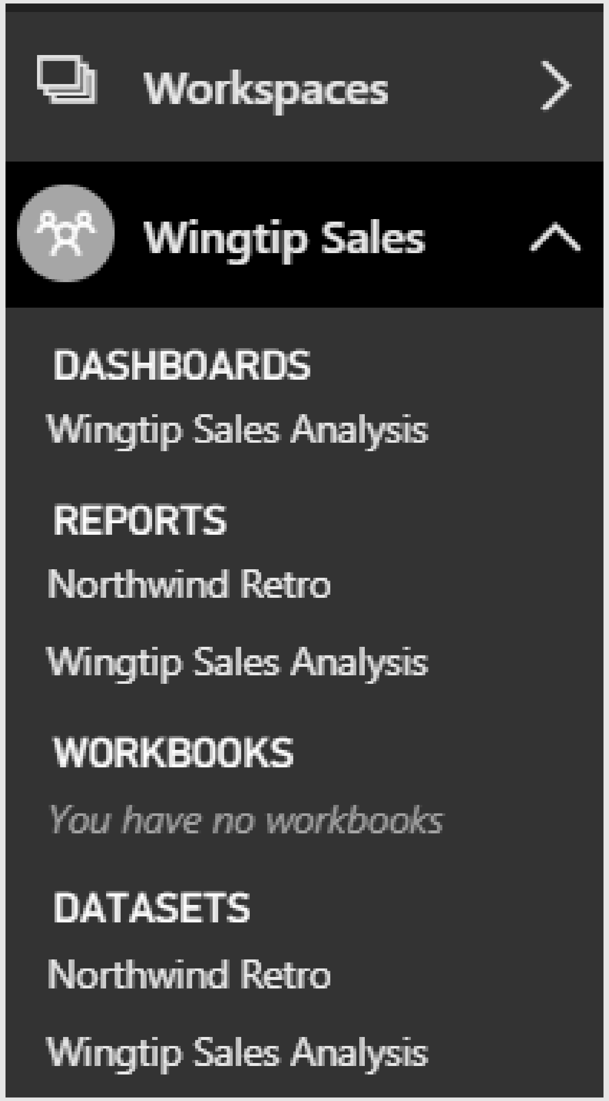
The Power BI Service – Who Is It For?

- Provides SaaS service used by web and mobile users
  - Power BI portal accessible to browsers at https://app.powerbi.com
  - Power BI mobile accessible to users on mobile phones & devices
- Provides PaaS service used by software developers
  - Power BI Service API accessible at https://api.powerbi.com



City	Amount
Lutz, FL	128,675.00
St Petersburg, FL	127,050.00
Palm Harbor, FL	103,550.00
Palma Ceia, FL	86,900.00
Westchase, FL	65,575.00
Tampa Palms, FL	64,550.00
Lake Magdalene, FL	59,075.00
Clearwater, FL	50,100.00
Carrollwood, FL	43,625.00
Belleair Bluffs, FL	43,550.00
Tarpon Springs, FL	40,375.00
Temple Terrace, FL	33,575.00
Clair Mel City, FL	32,975.00
Davis Island, FL	29,050.00
Tampa, FL	29,050.00
Odessa, FL	25,975.00
Thonotosassa, FL	23,300.00
Town N Country, FL	14,300.00
Largo, FL	13,575.00
Indian Rocks Beach...	11,350.00

Central Power BI Concepts


Workspaces

Wingtip Sales

DASHBOARDS

Wingtip Sales Analysis

REPORTS

Northwind Retro

Wingtip Sales Analysis

WORKBOOKS

You have no workbooks

DATASETS

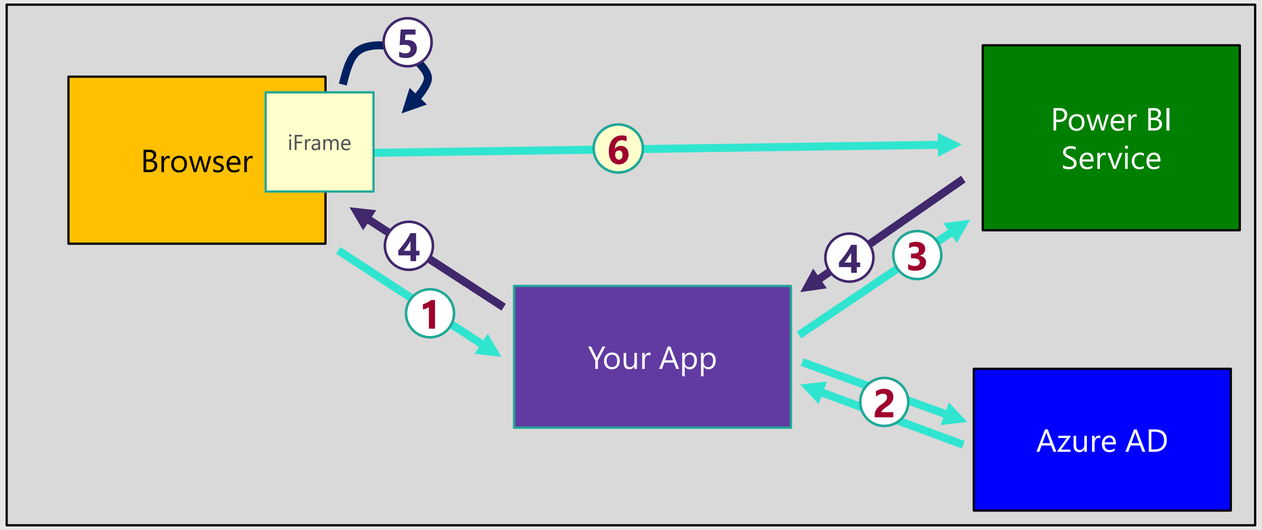
Northwind Retro

Wingtip Sales Analysis

- Workspace
  - Secure container for publishing content
  - Every licensed user gets a personal workspace
  - App workspaces created for custom solutions
- Dashboard
  - Consolidated view into reports and datasets
  - Custom solution entry point for mobile users
- Report
  - Collection of pages with tables & visualizations
  - Provides interactive control of filtering
- Dataset
  - Data model containing one or more tables
  - Can be very simple or very complex

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



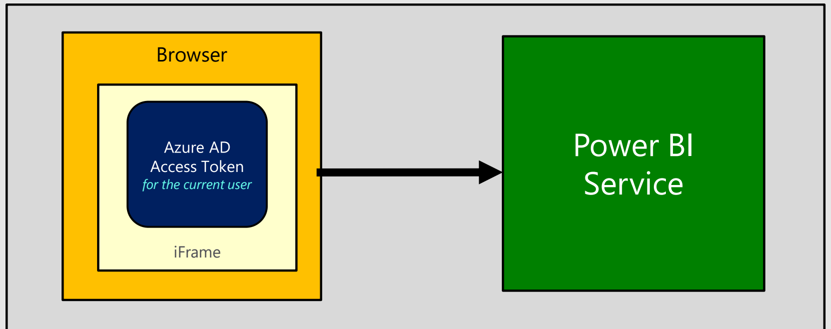
First Party Embedding vs Third Party Embedding

First Party Embedding

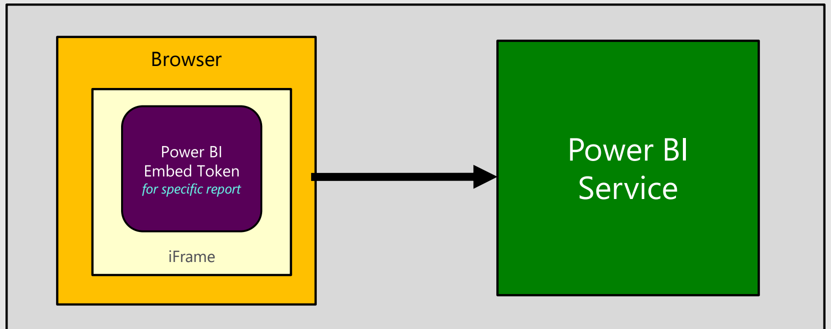
- Known as **User-Owns-Data** Model
- All users require a Power BI license
- Useful in corporate environments
- App authenticates as current user
- Your code runs with user's permissions
- User's access token passed to browser

Third Party Embedding

- Known as **App-Owns-Data** Model
- No users require Power BI license
- Useful for commercial applications
- App authenticates with app-only identity
- Your code runs with admin permissions
- Embed token passed to browser



First Party Embedding for use within Organization



Third Party Embedding for use by ISVs

Embeddable Resources

- Reports
- Dashboards
- Dashboard Tiles
- New Reports
- Q&A Experience
- Visuals in custom layout

DEMO

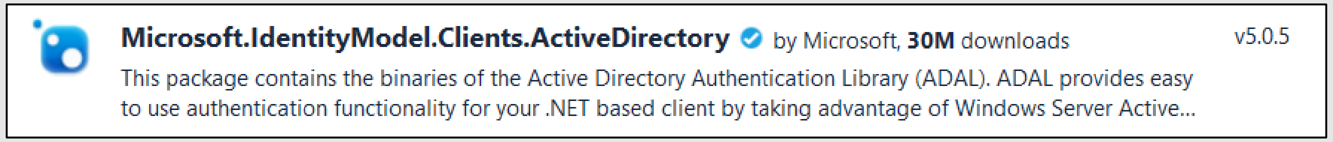
The Power BI Embedded from the User Perpsective

Agenda

- ✓ Power BI Embedding Fundamentals
- ➢ Authentication with Azure AD
- Programming the Power BI Service API
- App-only Authentication
- Single Page Applications (SPAs) with React.js
- Programming the Power BI JavaScript API

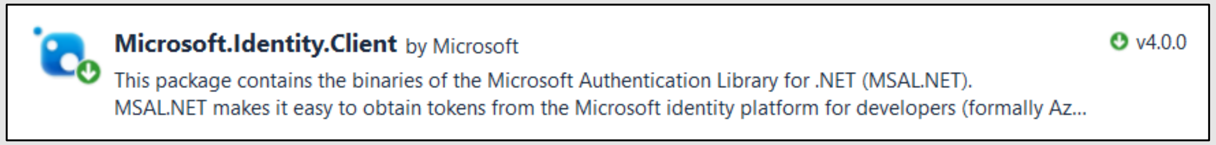
Azure AD Endpoints and Libraries

- Authenticating with the Azure AD V1 Endpoint
  - Heavily used over the last 5-6 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...

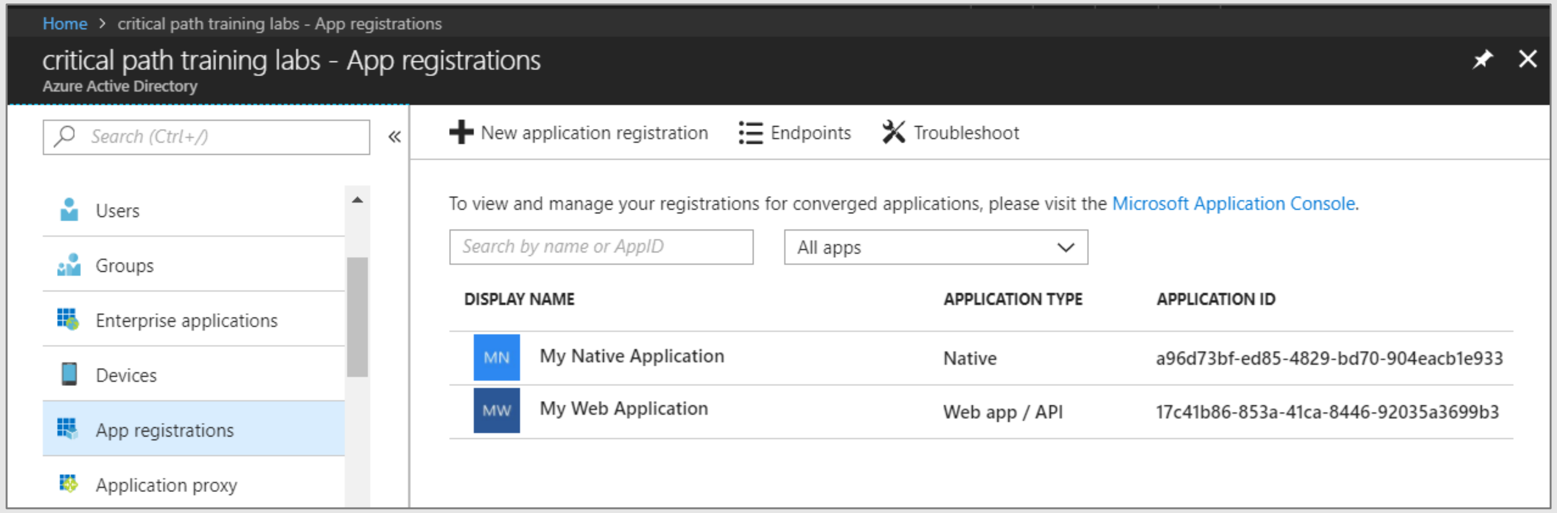
- Why move to the Azure AD V2 Endpoint?
  - Dynamic Incremental consent
  - New authentication flows (e.g. device code flow)

Authentication Flows

- **User Password Credential Flow** *(public client)*
  - Used in Native clients to obtain access code
  - Requires passing user name and password across network
- **Device Code Flow** *(public client)*
  - New style of authentication introduced with Azure AD v2 Endpoint
- **Client Credentials Flow** *(confidential client)*
  - Authentication based on password or certificate held by application
  - Used to obtain app-only access tokens
- **Authorization Code Flow** *(confidential client)*
  - Client first obtains authorization code sent back to browser
  - Client then obtains access token in server-to-server call
- **Implicit Flow** *(public client)*
  - Used in SPAs built with JavaScript and AngularJS
  - Application obtains access token w/o acquiring authorization code

Azure AD Applications

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

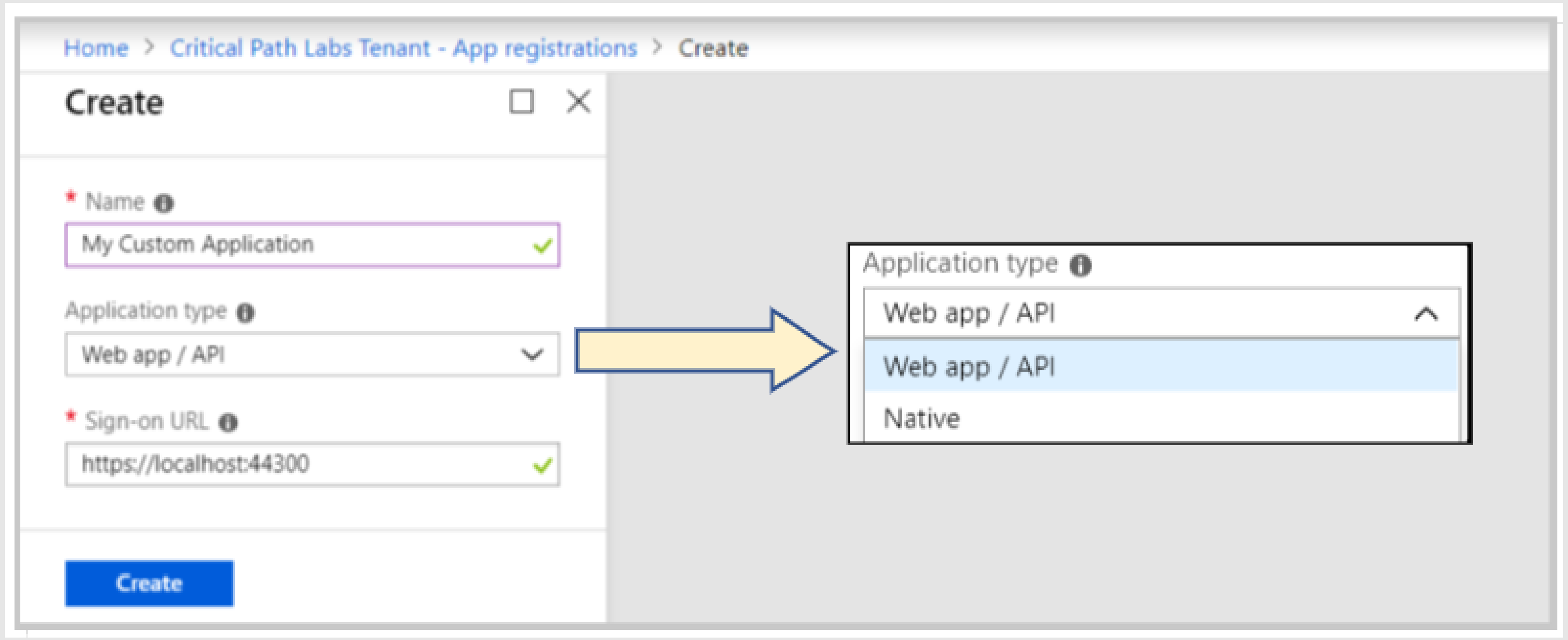


To view and manage your registrations for converged applications, please visit the Microsoft Application Console.

DISPLAY NAME	APPLICATION TYPE	APPLICATION ID
MN My Native Application	Native	a96d73bf-ed85-4829-bd70-904eacb1e933
MW My Web Application	Web app / API	17c41b86-853a-41ca-8446-92035a3699b3

Application Types

- Azure AD Application Types
  - Native clients
  - Web app / API client



Home > Critical Path Labs Tenant - App registrations > Create

Create

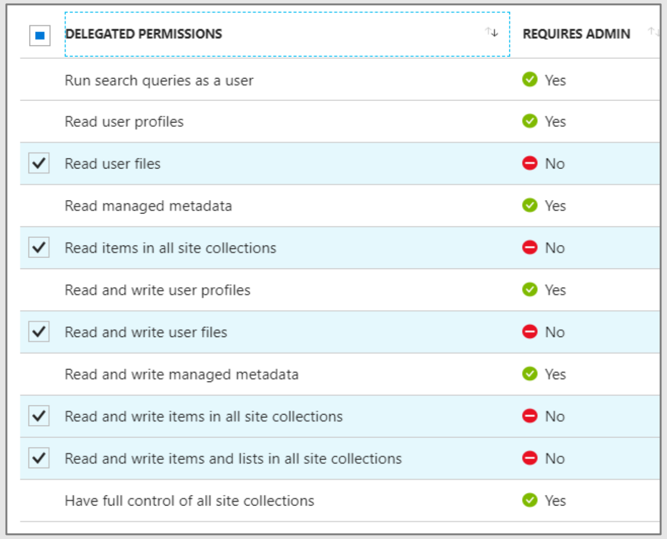
* Name: My Custom Application

Application type: Web app / API

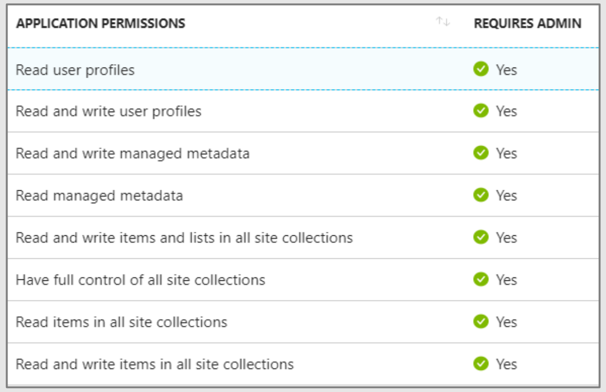
* Sign-on URL: https://localhost:44300

Delegated Permissions vs Application Permissions

- Permissions categorized into two basic types
  - Delegated permissions are (app + user) permissions
  - Application permissions are app-only permissions (far more powerful)
  - Not all application types and APIs support application permissions
  - Power BI Service API does not yet support application permissions
- Example permissions for Office 365 SharePoint Online
  - Some delegated permissions requires administrative permissions



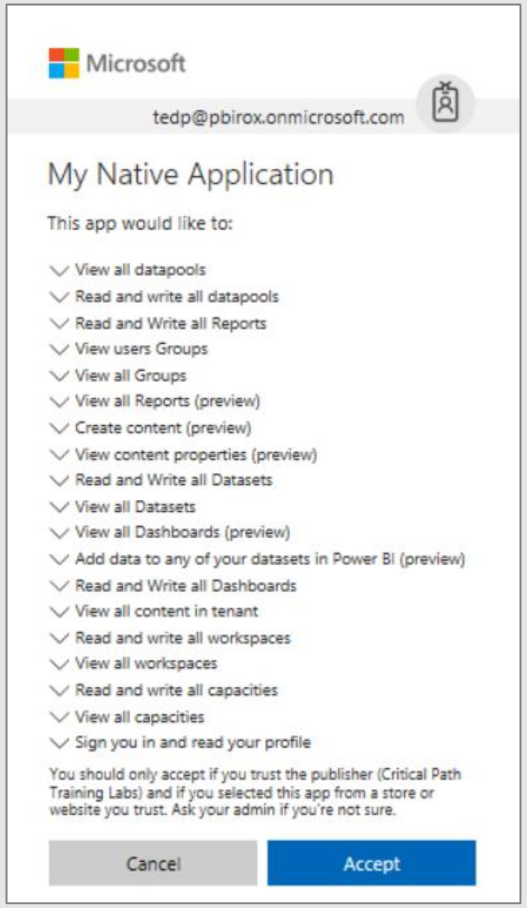
	DELEGATED PERMISSIONS	REQUIRES ADMIN
	Run search queries as a user	✓ Yes
	Read user profiles	✓ Yes
✓	Read user files	⛔ No
	Read managed metadata	✓ Yes
✓	Read items in all site collections	⛔ No
	Read and write user profiles	✓ Yes
✓	Read and write user files	⛔ No
	Read and write managed metadata	✓ Yes
✓	Read and write items in all site collections	⛔ No
✓	Read and write items and lists in all site collections	⛔ No
	Have full control of all site collections	✓ Yes



APPLICATION PERMISSIONS	REQUIRES ADMIN
Read user profiles	✓ Yes
Read and write user profiles	✓ Yes
Read and write managed metadata	✓ Yes
Read managed metadata	✓ Yes
Read and write items and lists in all site collections	✓ Yes
Have full control of all site collections	✓ Yes
Read items in all site collections	✓ Yes
Read and write items in all site collections	✓ Yes

Interactive Consent for Delegated Permissions

- Users must consent to delegated permissions
  - User prompted during first log in
  - User must click Accept
  - Only occurs once for each user



Microsoft

tedp@pbirox.onmicrosoft.com

My Native Application

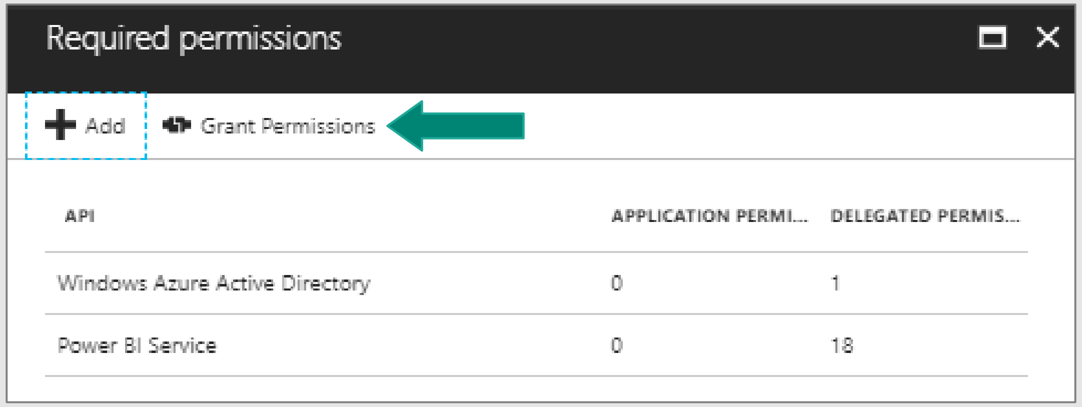
This app would like to:

- View all datapools
- Read and write all datapools
- Read and Write all Reports
- View users Groups
- View all Groups
- View all Reports (preview)
- Create content (preview)
- View content properties (preview)
- Read and Write all Datasets
- View all Datasets
- View all Dashboards (preview)
- Add data to any of your datasets in Power BI (preview)
- Read and Write all Dashboards
- View all content in tenant
- Read and write all workspaces
- View all workspaces
- Read and write all capacities
- View all capacities
- Sign you in and read your profile

You should only accept if you trust the publisher (Critical Path Training Labs) and if you selected this app from a store or website you trust. Ask your admin if you're not sure.

Granting Delegated Permissions

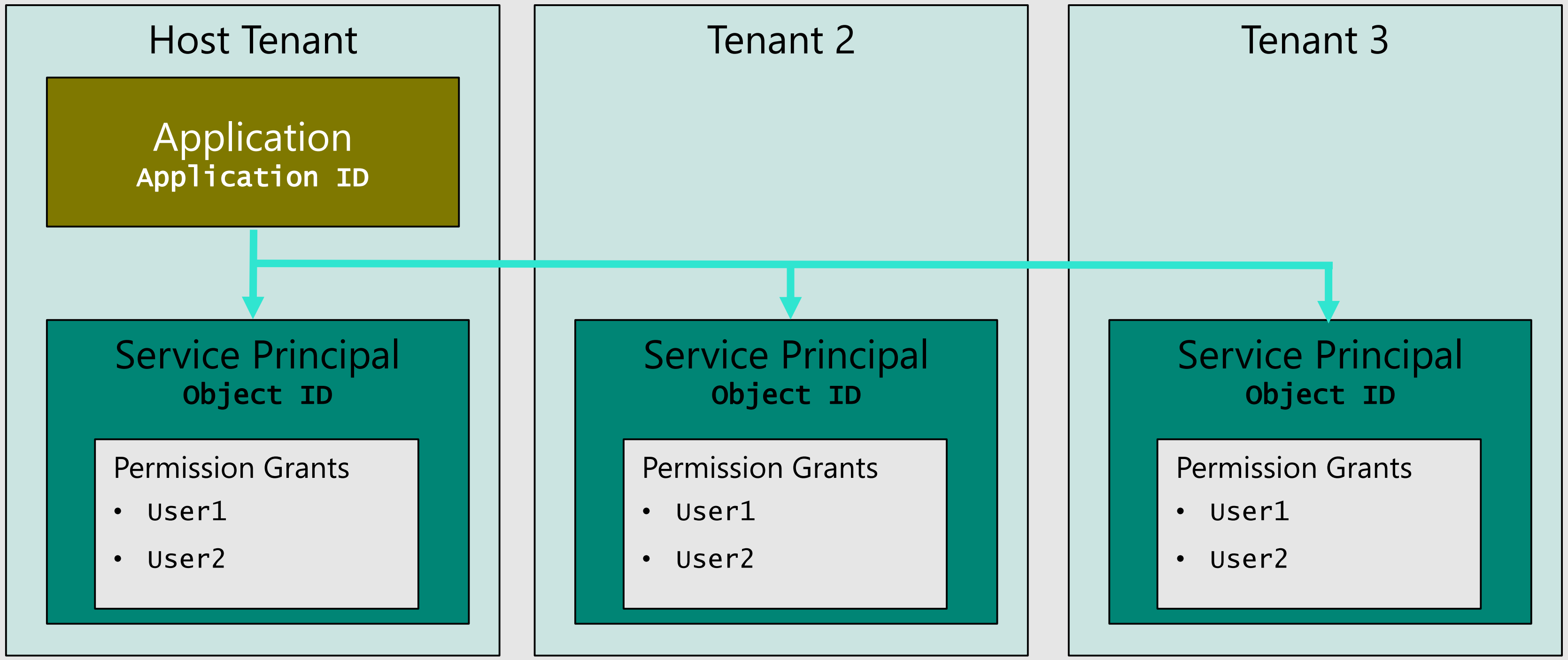
- It can be helpful to Grant Permissions in Azure portal
  - Prevents the need for interactive granting of application by user
  - Might be required when authenticating in non-interactive fashion



API	APPLICATION PERMI...	DELEGATED PERMIS...
Windows Azure Active Directory	0	1
Power BI Service	0	18

AAD Security Principals

- Azure AD creates service principal for application
  - Service principle created once per tenant
  - Service principle used to track permission grants
  - AAD creates service principal on demand when first needed
  - You can create service principal in PowerShell script



Registering AAD 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
```

Configuring Delegated Permissions

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

# configure delegated permisssions for the Power BI Service API
$requiredAccess = New-Object -TypeName "Microsoft.Open.AzureAD.Model.RequiredResourceAccess"
$requiredAccess.ResourceAppId = "00000009-0000-0000-c000-000000000000"

# create first delegated permission - Report.Read.All
$permission1 = New-Object -TypeName "Microsoft.Open.AzureAD.Model.ResourceAccess" `
                          -ArgumentList "4ae1bf56-f562-4747-b7bc-2fa0874ed46f","Scope"

# create second delegated permission - Dashboards.Read.All
$permission2 = New-Object -TypeName "Microsoft.Open.AzureAD.Model.ResourceAccess" `
                          -ArgumentList "2448370f-f988-42cd-909c-6528efd67c1a","Scope"

# add permissions to ResourceAccess list
$requiredAccess.ResourceAccess = $permission1, $permission2

# add permissions by updating application with RequiredResourceAccess object
Set-AzureADApplication -ObjectId $aadApplication.ObjectId -RequiredResourceAccess $requiredAccess
```

Registering an Azure AD Application using PowerShell

Agenda

- ✓ Power BI Embedding Fundamentals
- ✓ Authentication with Azure AD
- ➢ Programming the Power BI Service API
- App-only Authentication
- Single Page Applications (SPAs) with React.js
- Programming the Power BI JavaScript API

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

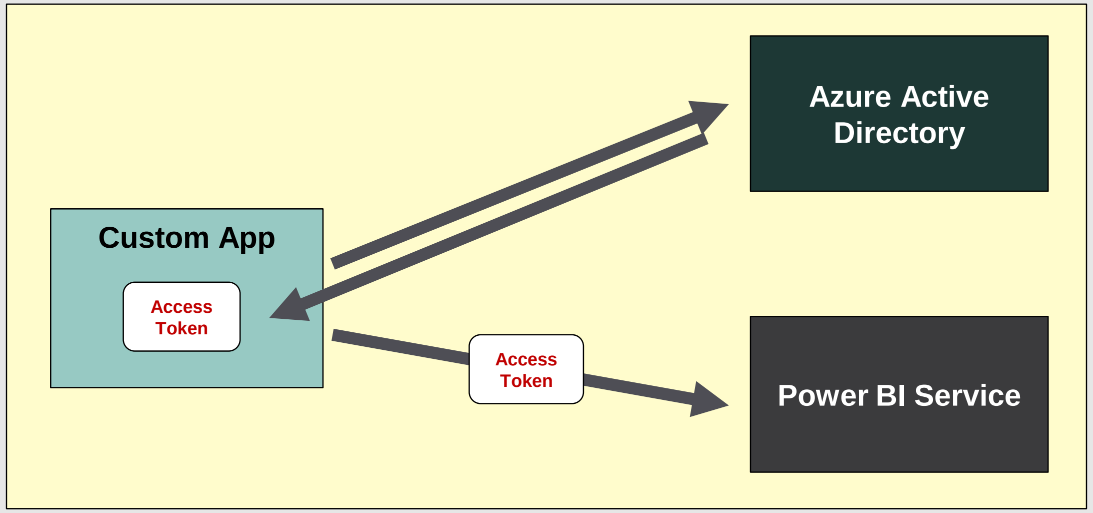
User APIs versus Admin APIs

- 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

Authenticating with Azure AD

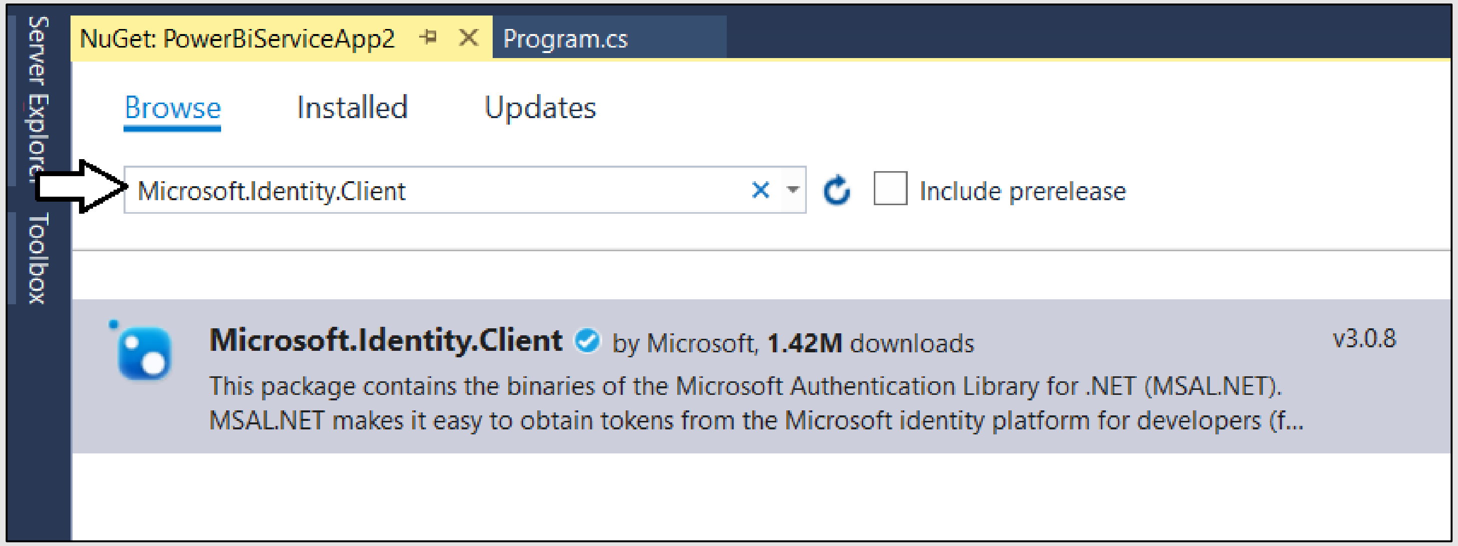
- Custom applications must authenticate with Azure AD
  - Your code implements and authentication flow to obtain access token
  - Access token must be passed when calling Power BI Service API



- 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 Library (.NET)

- Developing with the Microsoft Authentication Library
  - Provides access to Azure AD V2 Endpoint
  - Added to project as `Microsoft.Identity.Client` NuGet package
  - Provides different classes for *public clients* vs *confidential clients*



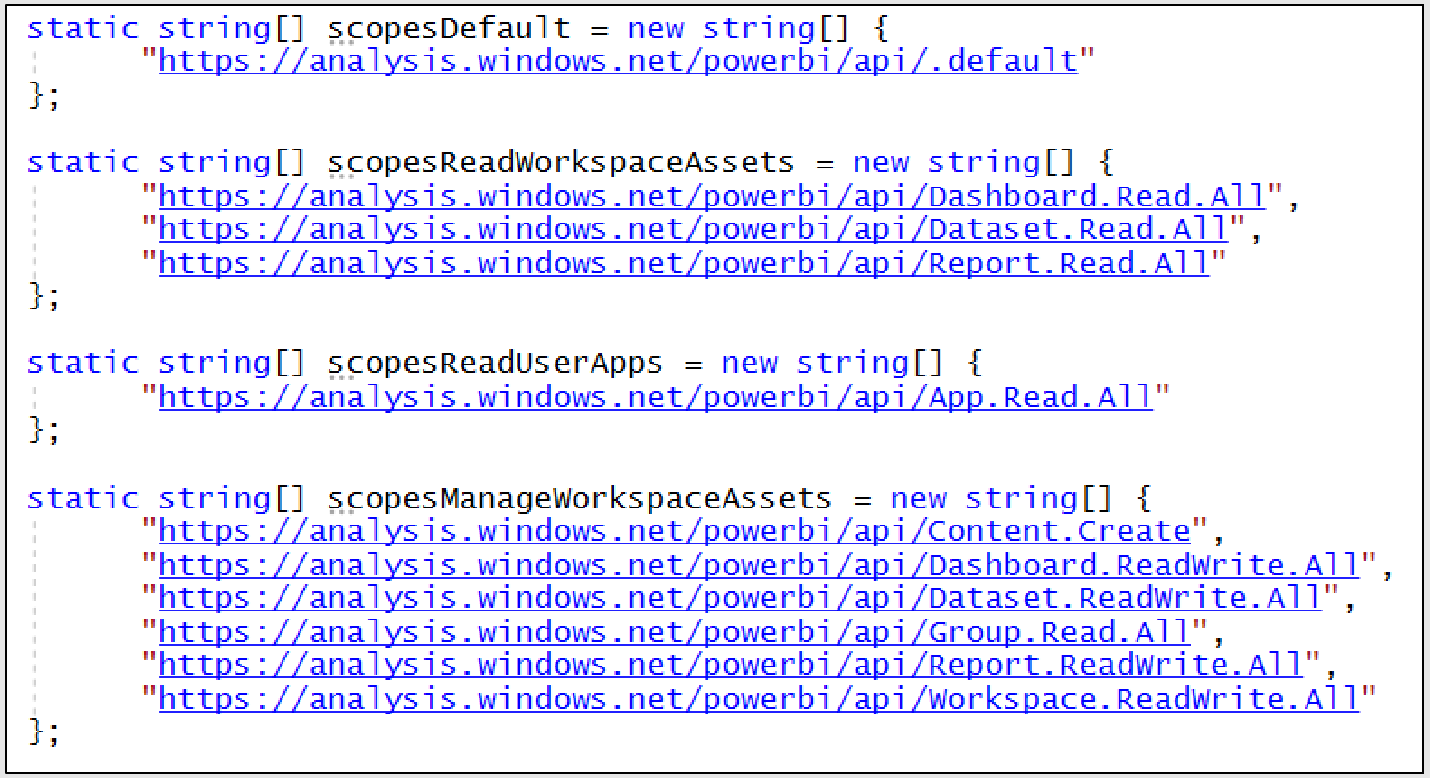
Microsoft.Identity.Client

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

Using MSAL with public client application

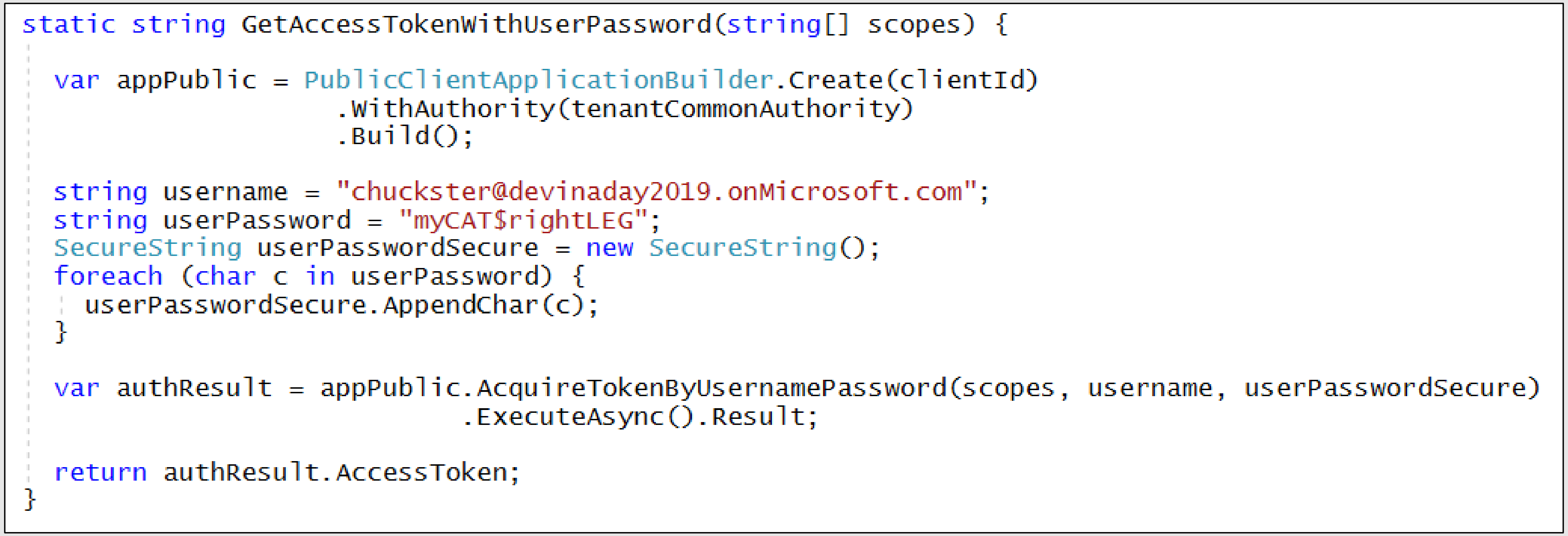
- Flow implemented using PublicClientApplication object
  - Created using PublicClientApplicationBuilder object
  - Requires passing redirect URI
  - You can control prompting behavior

```
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

Using MSAL with public client application

- MSAL supports user credential password flow
  - Supported in .NET runtime but not in .NET CORE
  - Microsoft recommends against using this flow

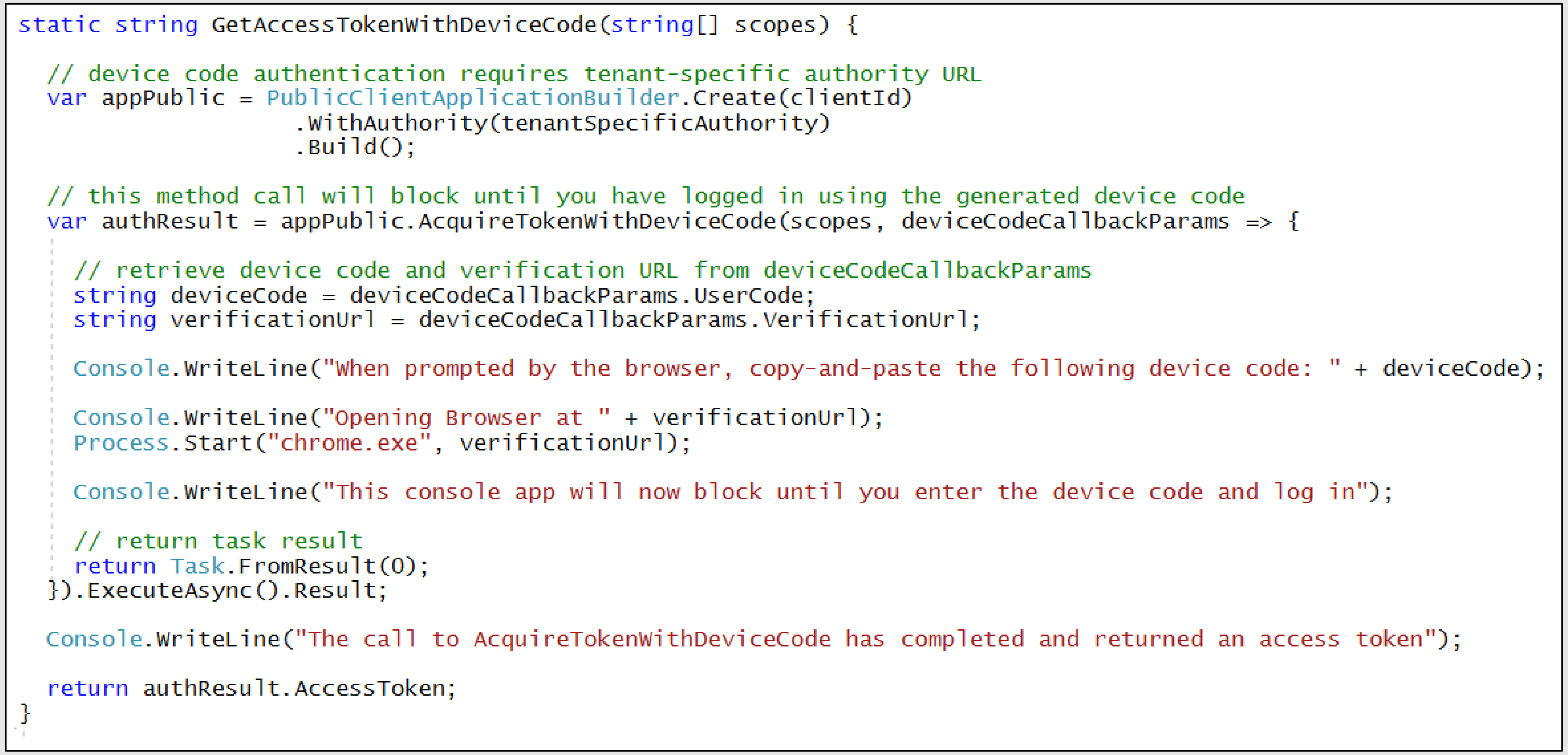


```
static string GetAccessTokenWithUserPassword(string[] scopes) {

  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;
}
```

Device Code Flow

Using MSAL with public client application

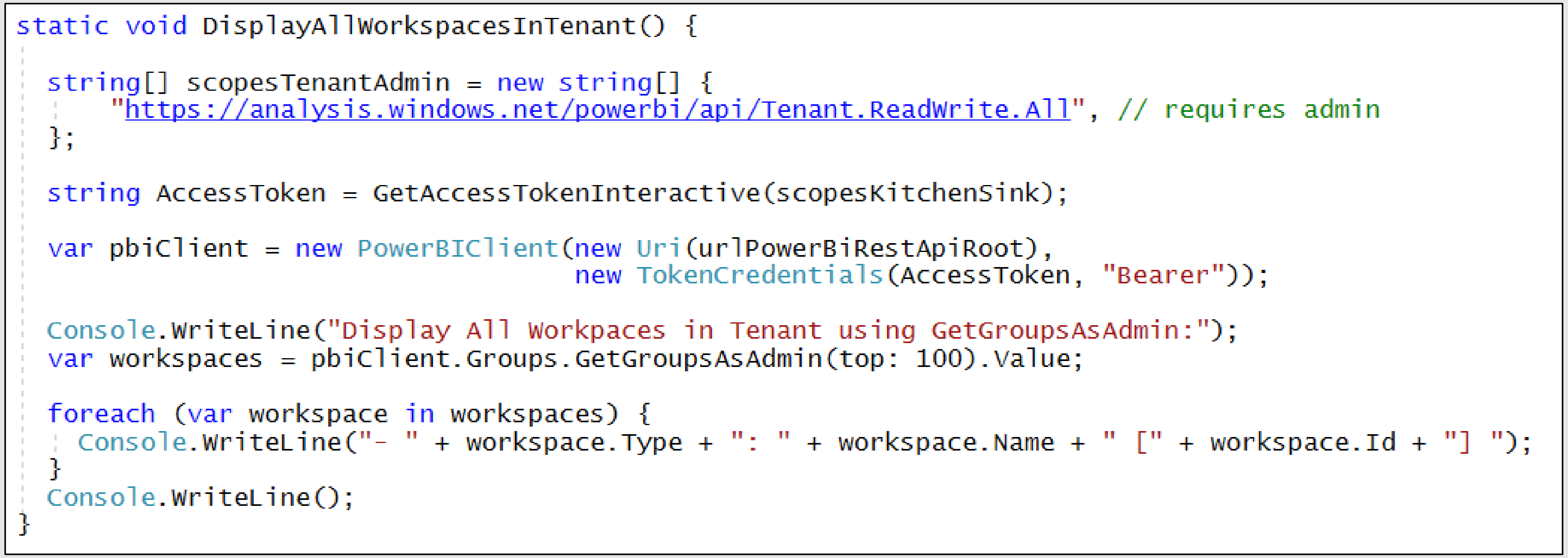
- MSAL introduced this new flow with MSAL
  - Much more secure than user password credential flow
  - Not available in ADAL



```
static string GetAccessTokenWithDeviceCode(string[] scopes) {

  // device code authentication requires tenant-specific authority URL
  var appPublic = PublicClientApplicationBuilder.Create(clientId)
                    .WithAuthority(tenantSpecificAuthority)
                    .Build();

  // this method call will block until you have logged in using the generated device code
  var authResult = appPublic.AcquireTokenWithDeviceCode(scopes, deviceCodeCallbackParams => {

    // retrieve device code and verification URL from deviceCodeCallbackParams
    string deviceCode = deviceCodeCallbackParams.UserCode;
    string verificationUrl = deviceCodeCallbackParams.VerificationUrl;

    Console.WriteLine("When prompted by the browser, copy-and-paste the following device code: " + deviceCode);

    Console.WriteLine("Opening Browser at " + verificationUrl);
    Process.Start("chrome.exe", verificationUrl);

    Console.WriteLine("This console app will now block until you enter the device code and log in");

    // return task result
    return Task.FromResult(0);
  }).ExecuteAsync().Result;

  Console.WriteLine("The call to AcquireTokenWithDeviceCode has completed and returned an access token");

  return authResult.AccessToken;
}
```

Calling into the Power BI Admin API

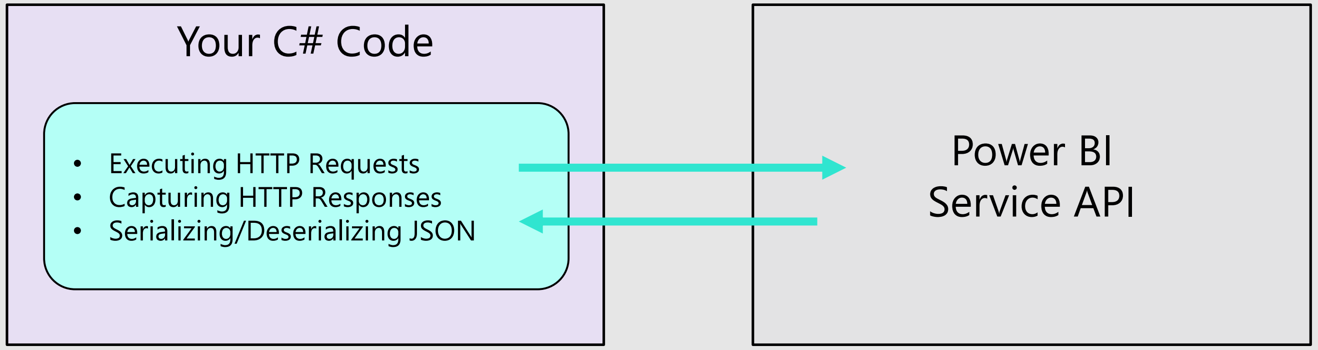
- Admin API exposed using AsAdmin methods
  - Example: `pbiClient.Groups.GetGroupsAsAdmin(top: 100).Value;`
  - Makes it possible to access every workspace in current tenant
  - Requires access token for user who is tenant or Power BI admin



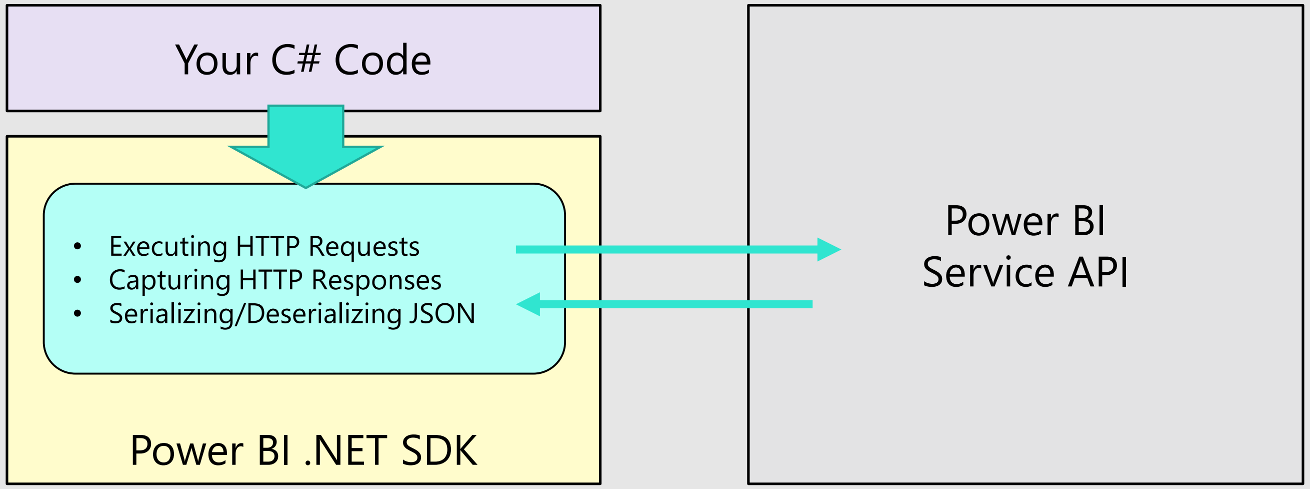
```
static void DisplayAllWorkspacesInTenant() {

  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 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();
}
```

Power BI .NET SDK

- Developing without the Power BI .NET SDK

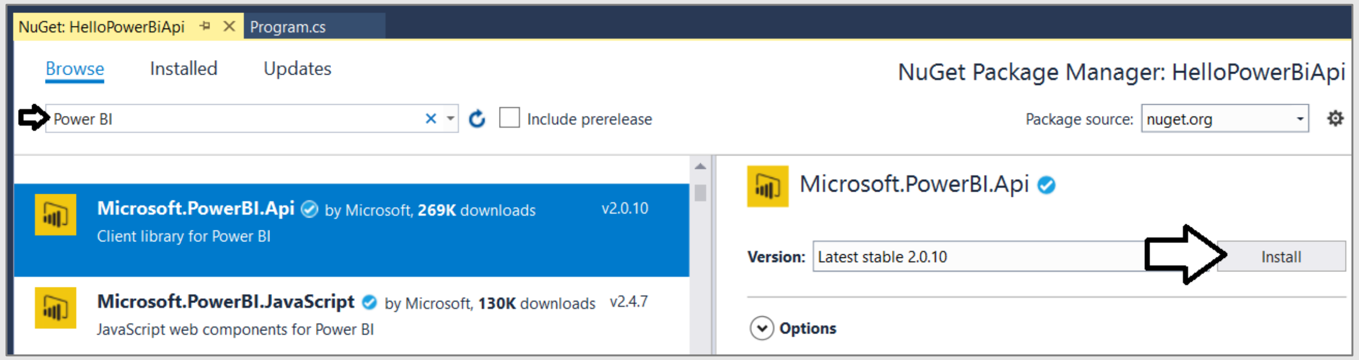


- Developing with the Power BI .NET SDK

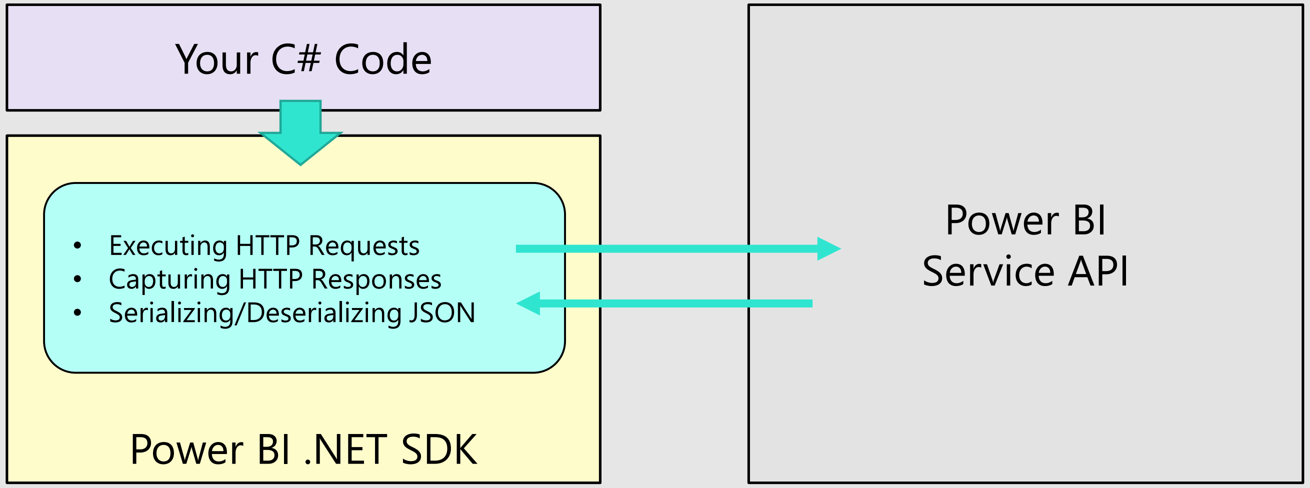


Power BI .NET SDK

- Added as a NuGet package

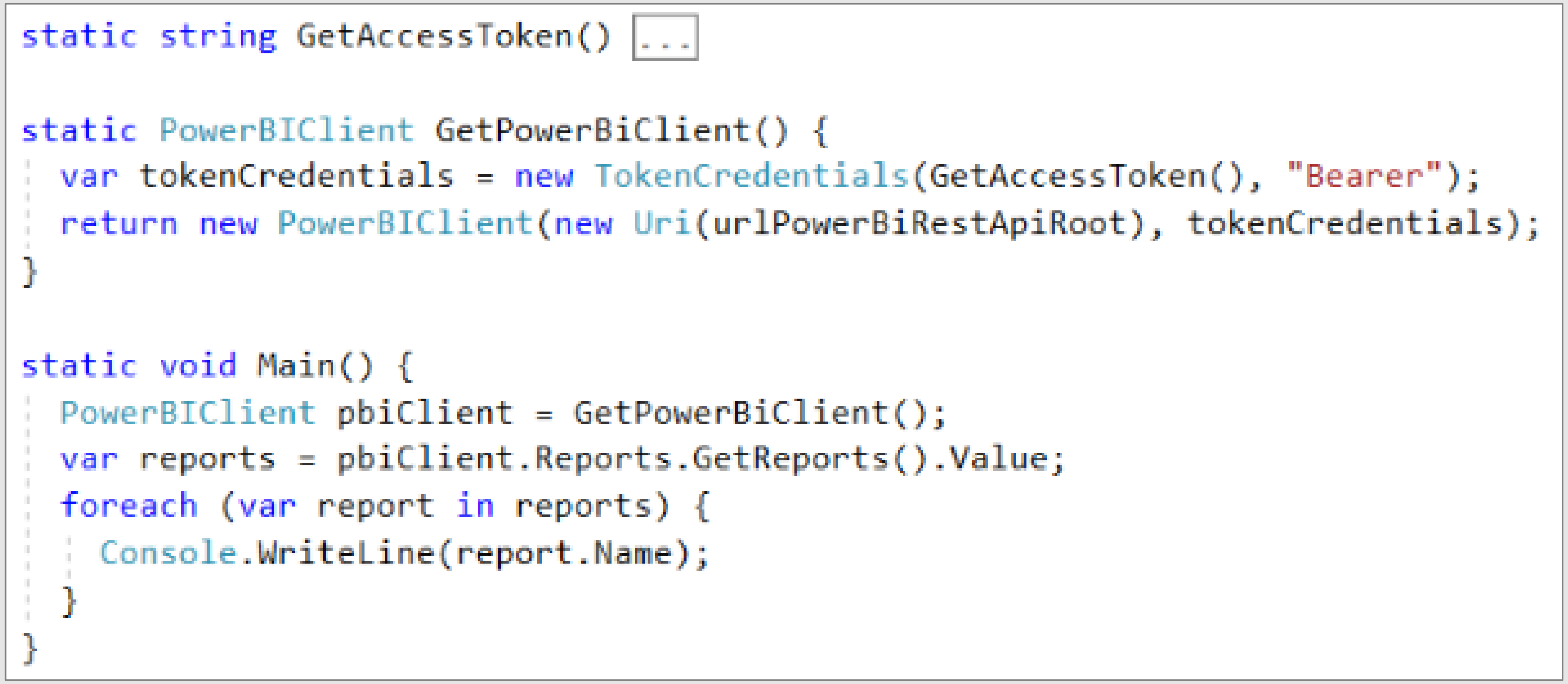


NuGet: HelloPowerBiApi
Program.cs
Browse
Installed
Updates
NuGet Package Manager: HelloPowerBiApi
Power BI
Include prerelease
Package source: nuget.org
Microsoft.PowerBI.Api by Microsoft, 269K downloads v2.0.10
Client library for Power BI
Microsoft.PowerBI.JavaScript by Microsoft, 130K downloads v2.4.7
JavaScript web components for Power BI
Microsoft.PowerBI.Api
Version: Latest stable 2.0.10
Install
Options



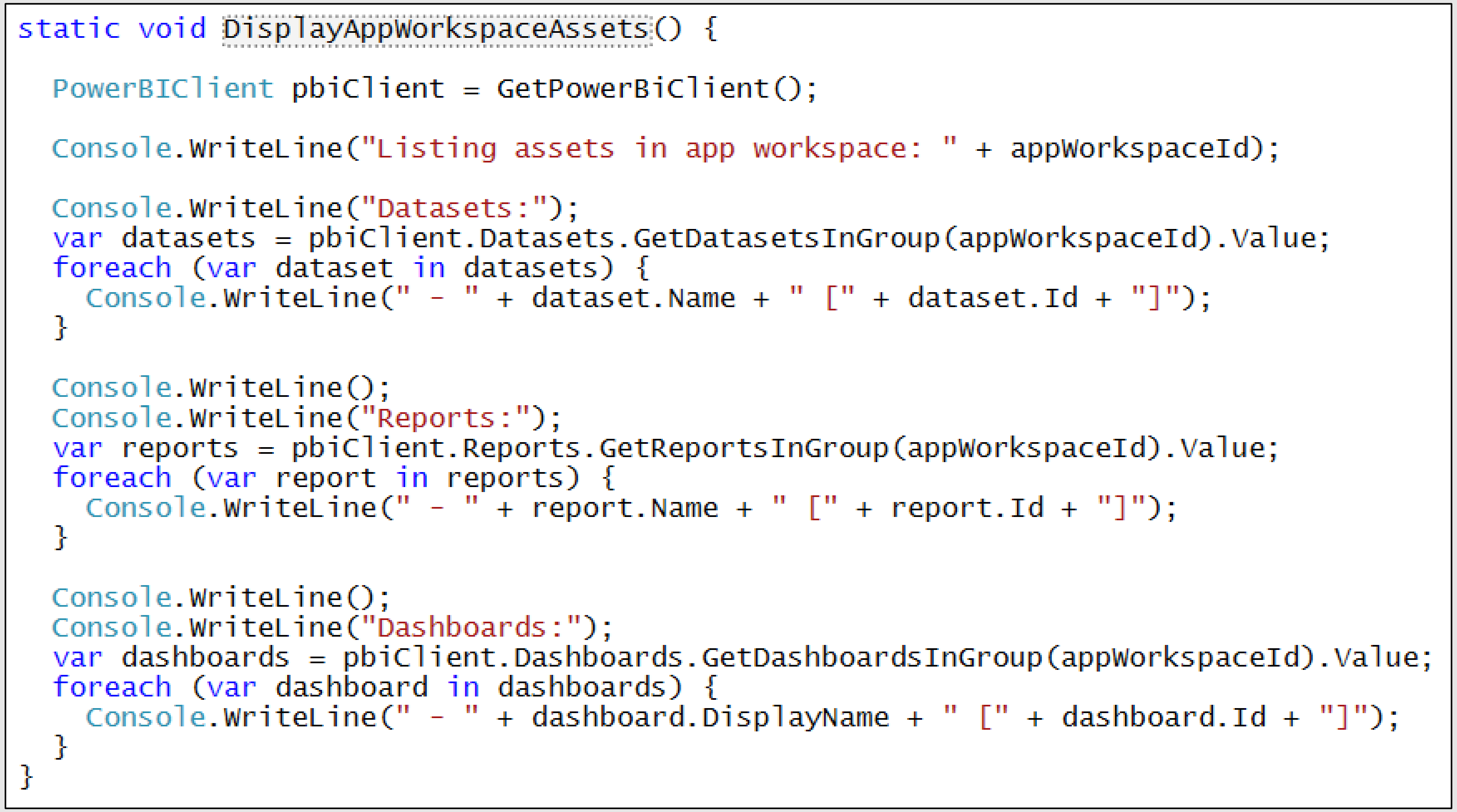
Initializing an Instance of PowerBIClient

- 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);
  }
}
```

Enumerating Collections with PowerBiClient

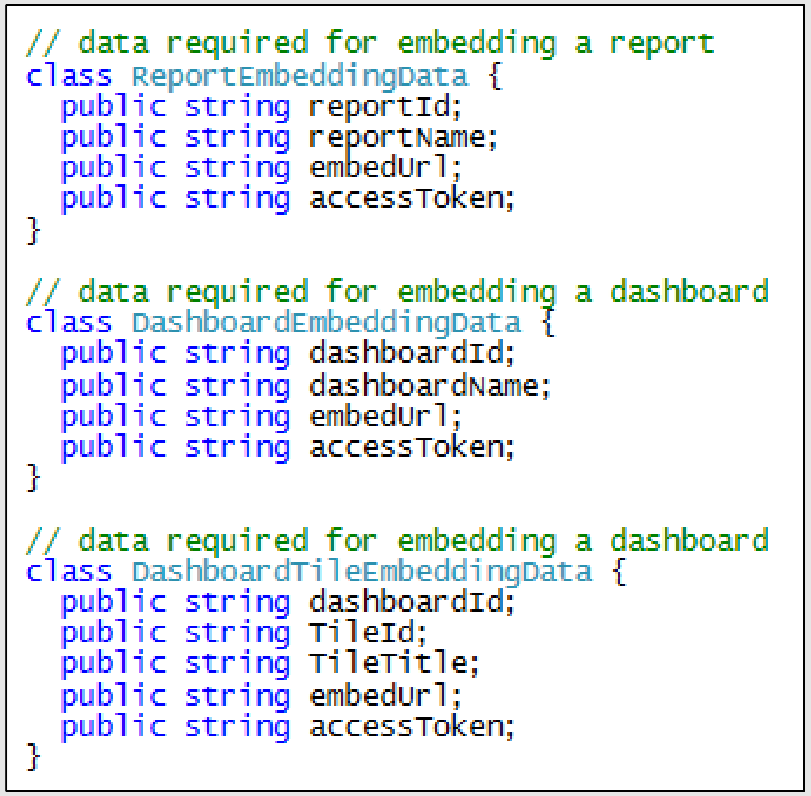


```
static void DisplayAppWorkspaceAssets() {

  PowerBIClient pbiClient = GetPowerBiClient();

  Console.WriteLine("Listing assets in app workspace: " + appWorkspaceId);

  Console.WriteLine("Datasets:");
  var datasets = pbiClient.Datasets.GetDatasetsInGroup(appWorkspaceId).Value;
  foreach (var dataset in datasets) {
    Console.WriteLine(" - " + dataset.Name + " [" + dataset.Id + "]");
  }

  Console.WriteLine();
  Console.WriteLine("Reports:");
  var reports = pbiClient.Reports.GetReportsInGroup(appWorkspaceId).Value;
  foreach (var report in reports) {
    Console.WriteLine(" - " + report.Name + " [" + report.Id + "]");
  }

  Console.WriteLine();
  Console.WriteLine("Dashboards:");
  var dashboards = pbiClient.Dashboards.GetDashboardsInGroup(appWorkspaceId).Value;
  foreach (var dashboard in dashboards) {
    Console.WriteLine(" - " + dashboard.DisplayName + " [" + dashboard.Id + "]");
  }
}
```

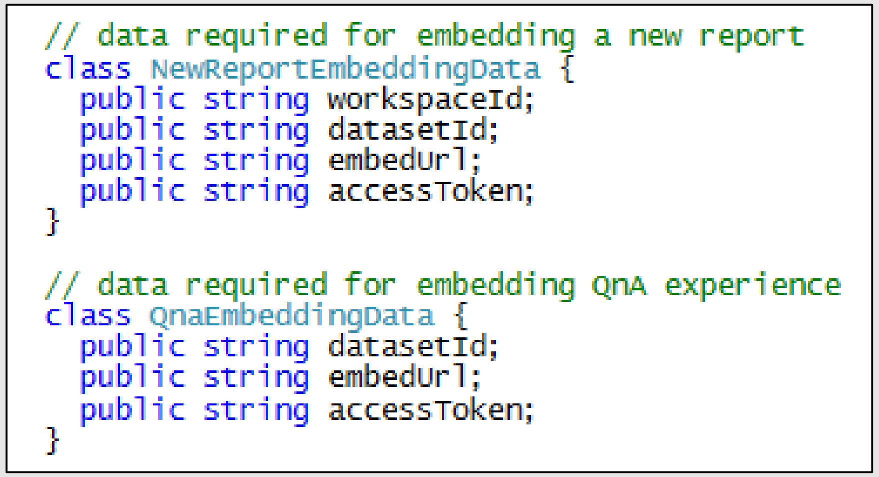
Agenda

- ✓ Power BI Embedding Fundamentals
- ✓ Authentication with Azure AD
- ✓ Programming the Power BI Service API
- ➢ App-only Authentication
- Single Page Applications (SPAs) with React.js
- Programming the Power BI JavaScript API

Report and Dataset Info



```
// data required for embedding a report
class ReportEmbeddingData {
  public string reportId;
  public string reportName;
  public string embedUrl;
  public string accessToken;
}

// data required for embedding a dashboard
class DashboardEmbeddingData {
  public string dashboardId;
  public string dashboardName;
  public string embedUrl;
  public string accessToken;
}

// data required for embedding a dashboard
class DashboardTileEmbeddingData {
  public string dashboardId;
  public string TileId;
  public string TileTitle;
  public string embedUrl;
  public string accessToken;
}
```

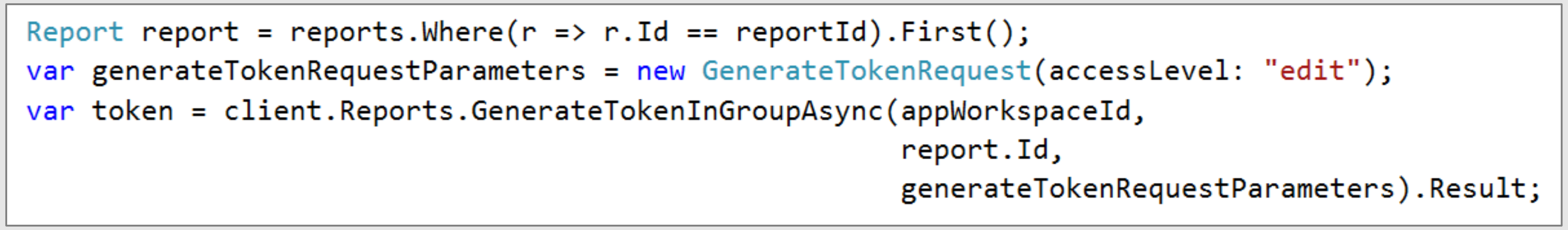


```
// data required for embedding a new report
class NewReportEmbeddingData {
  public string workspaceId;
  public string datasetId;
  public string embedUrl;
  public string accessToken;
}

// data required for embedding QnA experience
class QnaEmbeddingData {
  public string datasetId;
  public string embedUrl;
  public string accessToken;
}
```

Embed Tokens

- You can embed reports using master user 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;
```

Data for Third Party Report Embedding

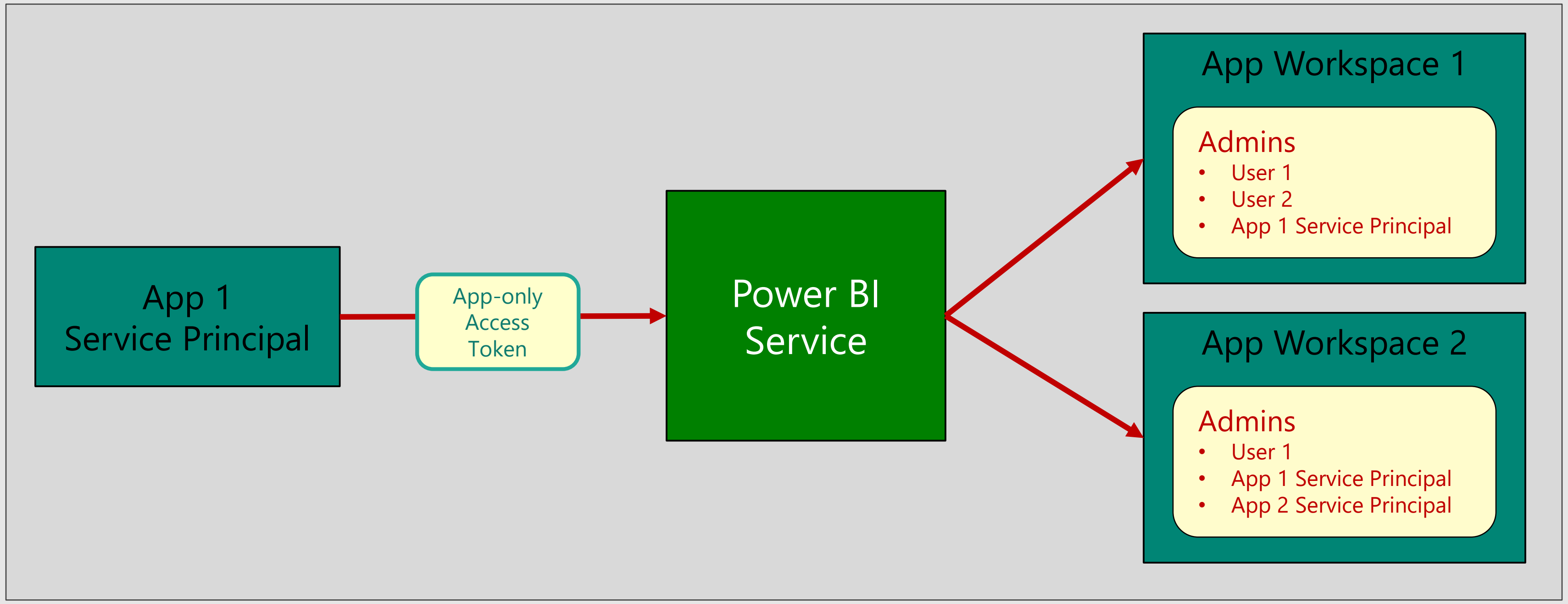
```
public static ReportEmbeddingData GetReportEmbeddingData() {

  PowerBIClient pbiClient = GetPowerBiClient();

  var report = pbiClient.Reports.GetReportInGroup(workspaceId, reportId);
  var embedUrl = report.EmbedUrl;
  var reportName = report.Name;

  // create token request object
  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "view");

  // call to Power BI Service API and pass GenerateTokenRequest object to generate embed token
  string embedToken = pbiClient.Reports.GenerateTokenInGroup(workspaceId,
                                                             report.Id,
                                                             generateTokenRequestParameters).Token;

  return new ReportEmbeddingData {
    reportId = reportId,
    reportName = reportName,
    embedUrl = embedUrl,
    accessToken = embedToken
  };

}
```

Getting the Data for Dashboard Embedding

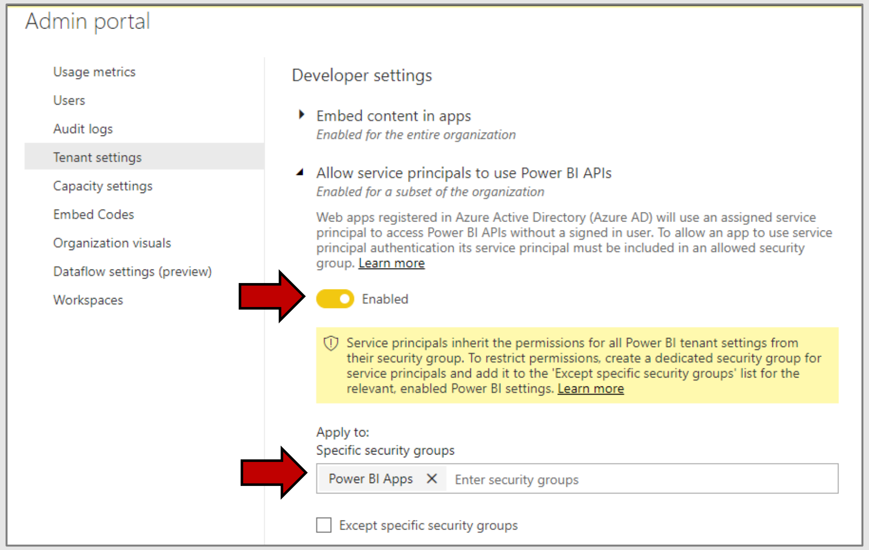
```
public static DashboardEmbeddingData GetDashboardEmbeddingData() {

  PowerBIClient pbiClient = GetPowerBiClient();

  var dashboard = pbiClient.Dashboards.GetDashboardInGroup(workspaceId, dashboardId);
  var embedUrl = dashboard.EmbedUrl;
  var dashboardDisplayName = dashboard.DisplayName;

  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "view");
  string embedToken = pbiClient.Dashboards.GenerateTokenInGroup(workspaceId,
                                                                dashboardId,
                                                                generateTokenRequestParameters).Token;

  return new DashboardEmbeddingData {
    dashboardId = dashboardId,
    dashboardName = dashboardDisplayName,
    embedUrl = embedUrl,
    accessToken = embedToken
  };

}
```

App-only Access Control

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



Tenant Setup



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

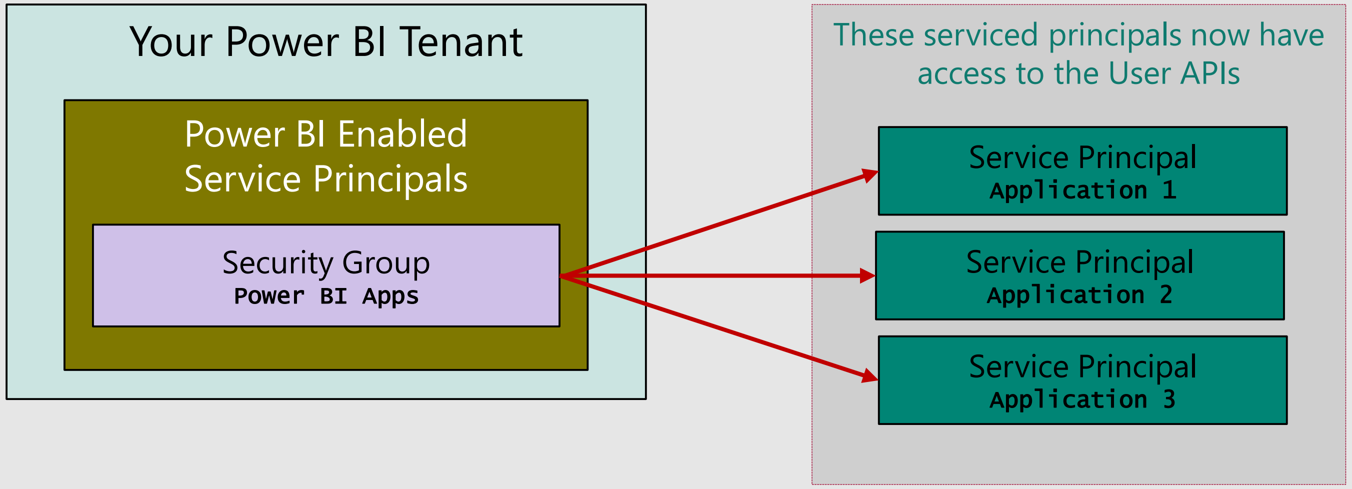
Enabled

Service principals 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

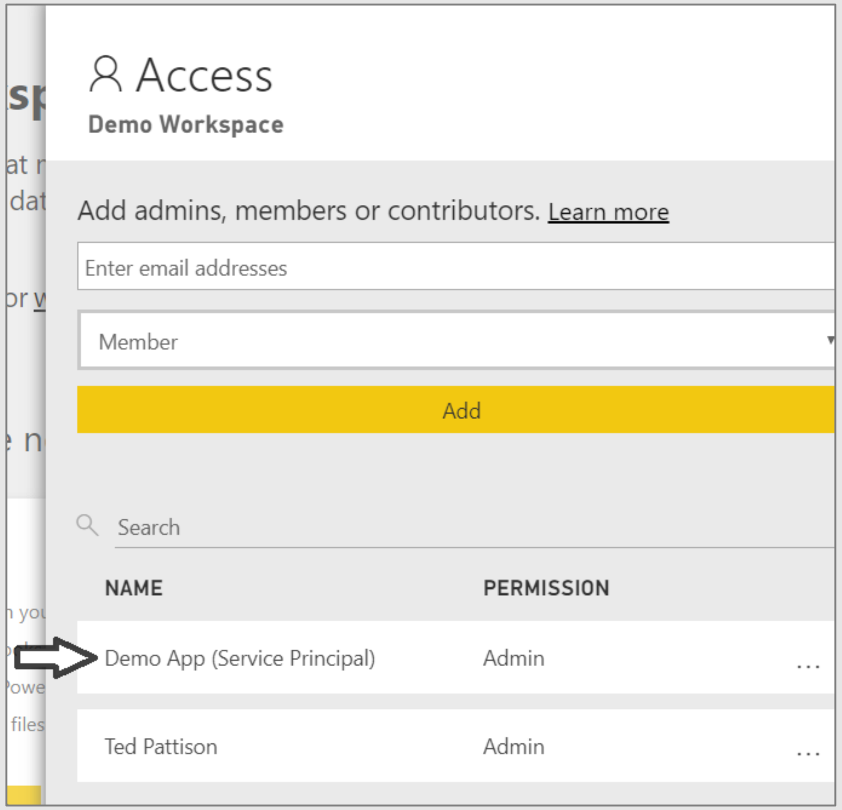
Power BI Apps ✕ Enter security groups

☐ 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



Access

Demo Workspace

Add admins, members or contributors. Learn more

Enter email addresses

Member

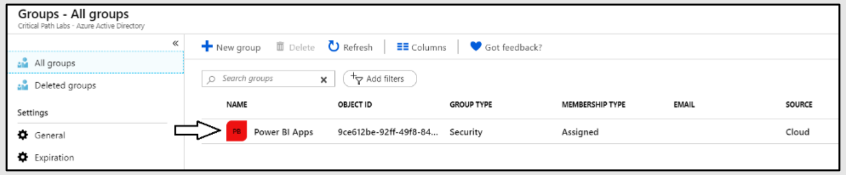
Add

Search

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

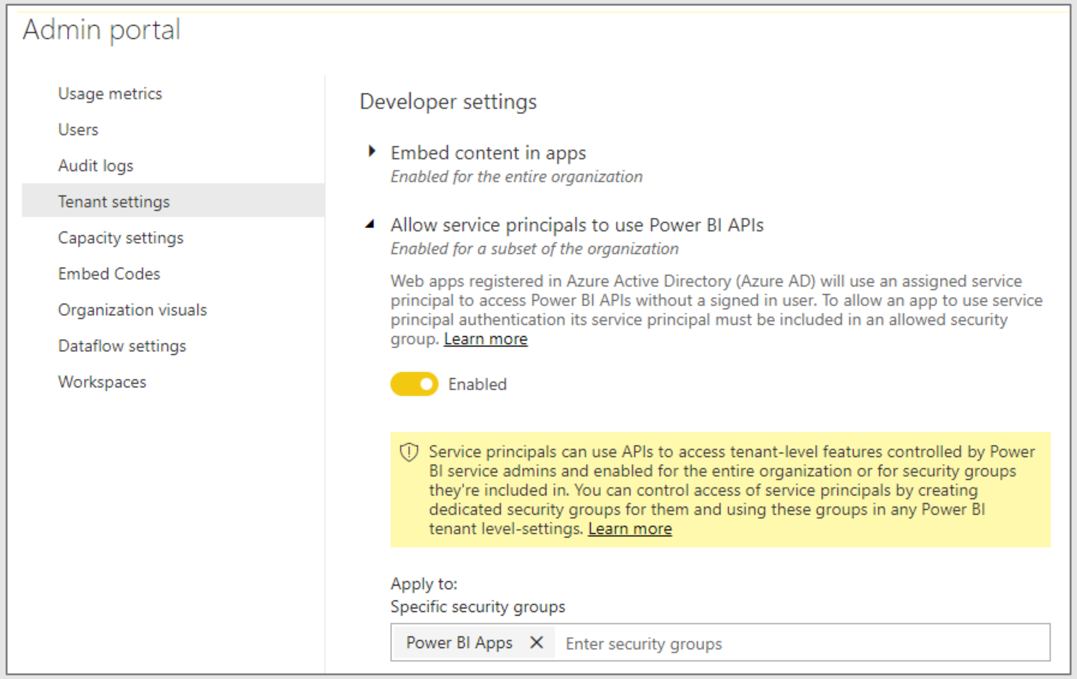
Setting Up for App-Owns-Data – Part 1

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



	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*



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

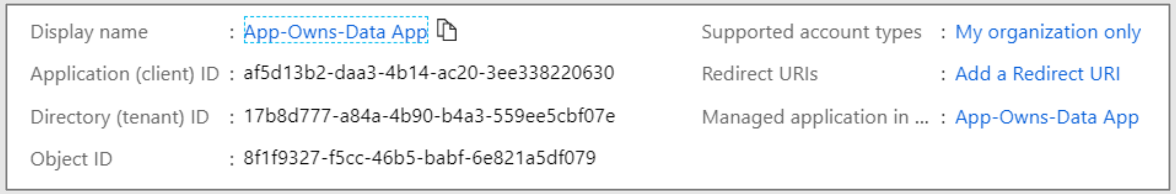
Service principals can use APIs to access tenant-level features controlled by Power BI service admins and enabled for the entire organization or for security groups they're included in. You can control access of service principals by creating dedicated security groups for them and using these groups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

Apply to:
Specific security groups

Power BI Apps ✕ Enter security groups

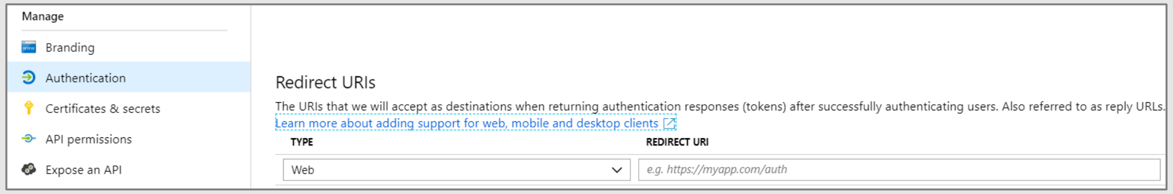
Setting Up for App-Owns-Data – Part 2

- 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

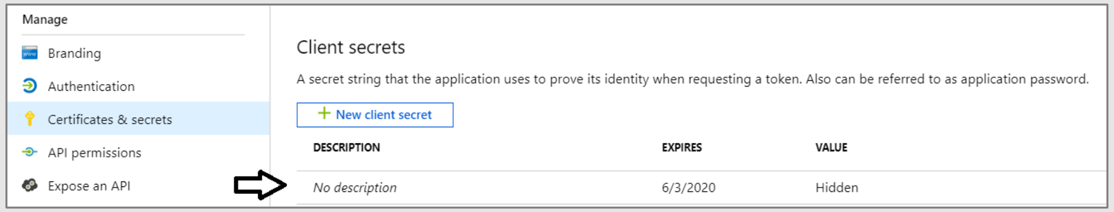


Redirect URIs

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

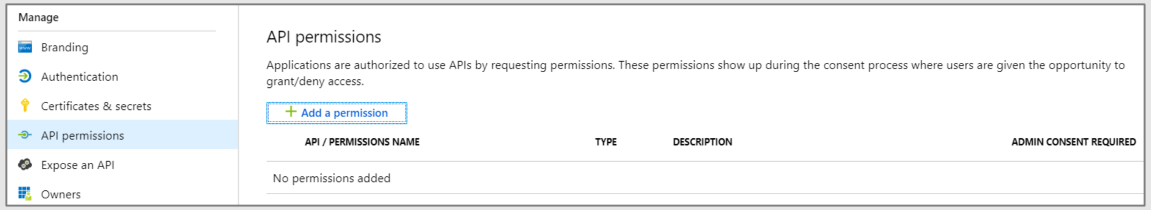


Client secrets

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



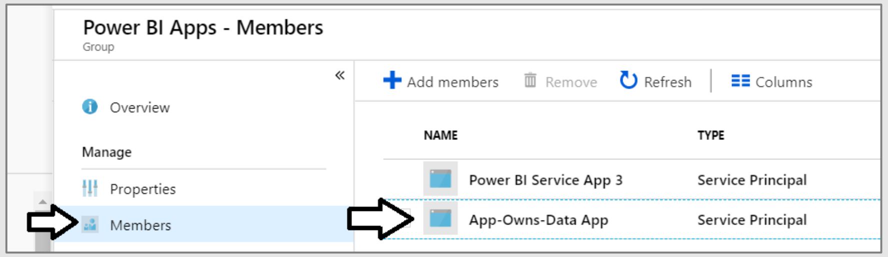
API permissions

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

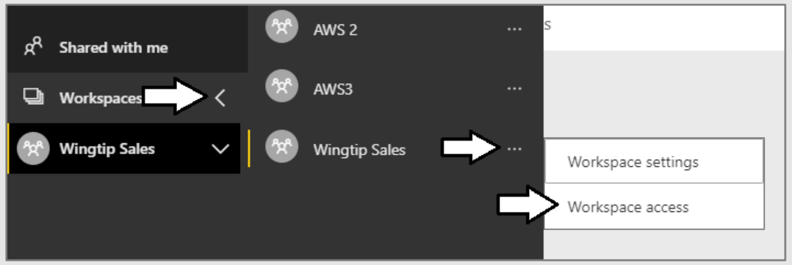
- Add application's service principal in **Power BI Apps** security group

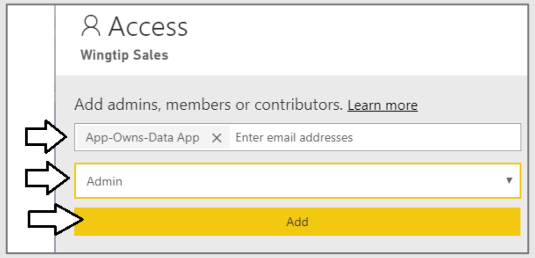


Power BI Apps - Members
Group

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

- Configure application's service principal as workspace admin





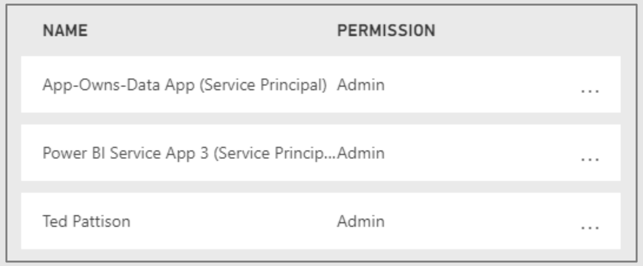
Access
Wingtip Sales

Add admins, members or contributors. Learn more

App-Owns-Data App ✕ Enter email addresses

Admin

- Service principal should now be workspace admin

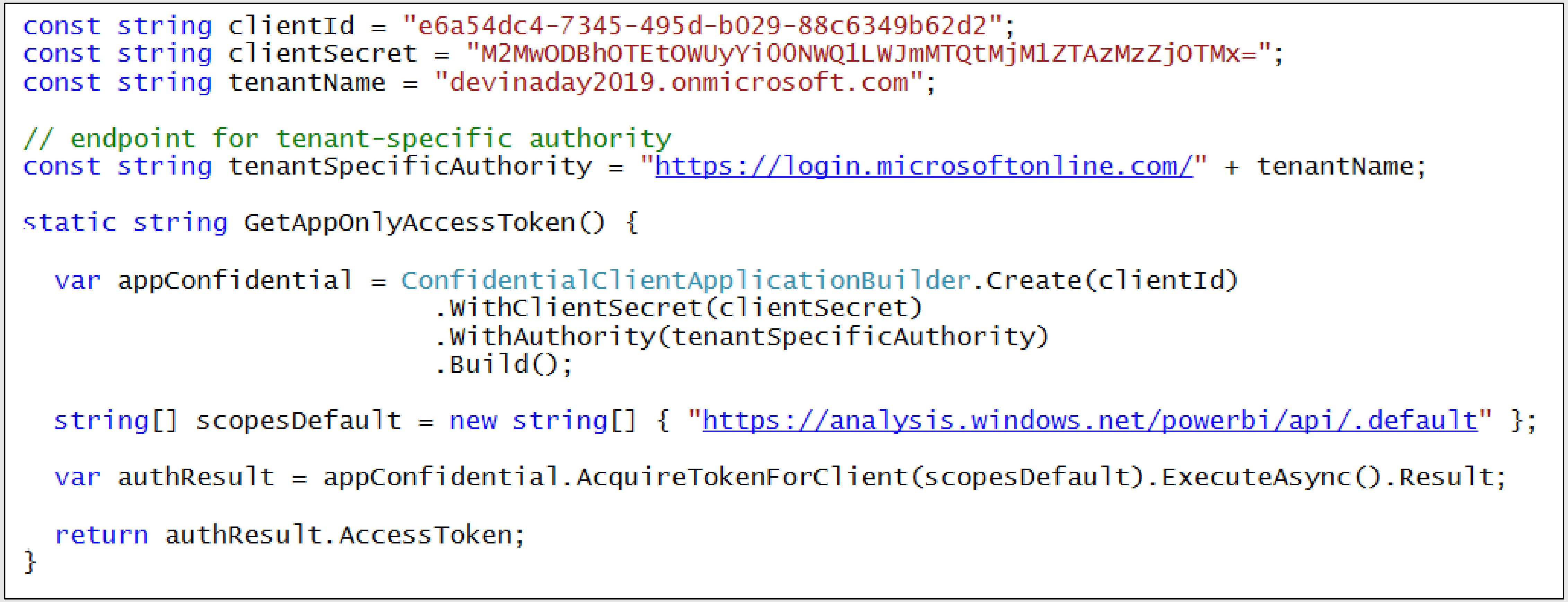


NAME	PERMISSION	
App-Owns-Data App (Service Principal)	Admin	...
Power BI Service App 3 (Service Princip...	Admin	...
Ted Pattison	Admin	...

Client Credentials Flow

Using MSAL with confidential client application

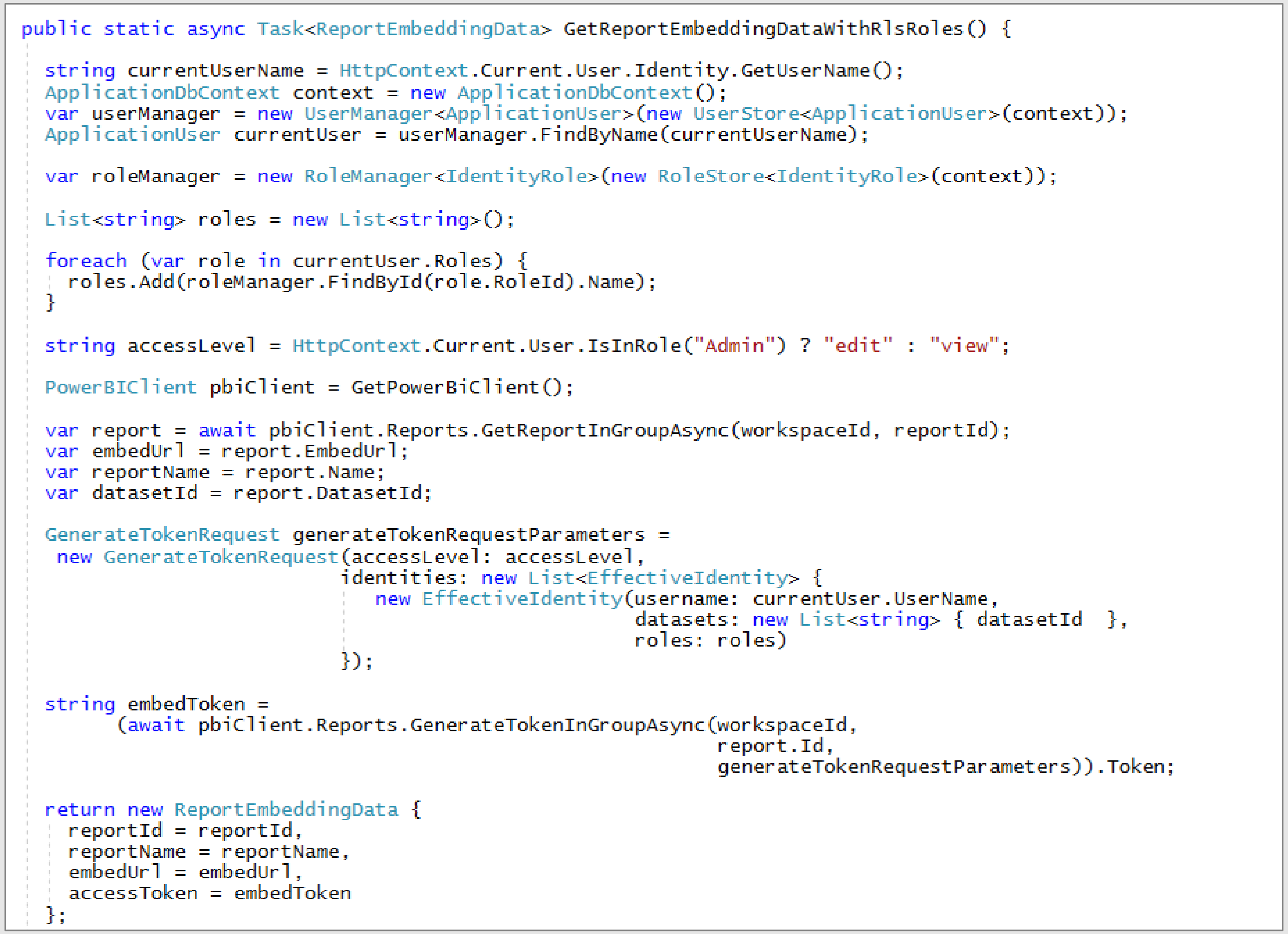
- Client credentials flow used to obtain app-only token
  - Requires passing app secret (e.g. app password or certificate)
  - Requires passing tenant-specific endpoint



```
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;
}
```

Examining the ThirdPartyEmbeddingApp

Working with RLS Roles and EffectiveIdentity



```
public static async Task<ReportEmbeddingData> GetReportEmbeddingDataWithRlsRoles() {

  string currentUserName = HttpContext.Current.User.Identity.GetUserName();
  ApplicationDbContext context = new ApplicationDbContext();
  var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(context));
  ApplicationUser currentUser = userManager.FindByName(currentUserName);

  var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));

  List<string> roles = new List<string>();

  foreach (var role in currentUser.Roles) {
    roles.Add(roleManager.FindById(role.RoleId).Name);
  }

  string accessLevel = HttpContext.Current.User.IsInRole("Admin") ? "edit" : "view";

  PowerBIClient pbiClient = GetPowerBiClient();

  var report = await pbiClient.Reports.GetReportInGroupAsync(workspaceId, reportId);
  var embedUrl = report.EmbedUrl;
  var reportName = report.Name;
  var datasetId = report.DatasetId;

  GenerateTokenRequest generateTokenRequestParameters =
   new GenerateTokenRequest(accessLevel: accessLevel,
                           identities: new List<EffectiveIdentity> {
                              new EffectiveIdentity(username: currentUser.UserName,
                                                    datasets: new List<string> { datasetId  },
                                                    roles: roles)
                           });

  string embedToken =
        (await pbiClient.Reports.GenerateTokenInGroupAsync(workspaceId,
                                                            report.Id,
                                                            generateTokenRequestParameters)).Token;

  return new ReportEmbeddingData {
    reportId = reportId,
    reportName = reportName,
    embedUrl = embedUrl,
    accessToken = embedToken
  };
```

DEMO

Examining the RowLevelSecurityDemo

Agenda

- ✓ Power BI Embedding Fundamentals
- ✓ Authentication with Azure AD
- ✓ Programming the Power BI Service API
- ✓ App-only Authentication
- ✓ Single Page Applications (SPAs) with React.js
- ➢ Programming the Power BI JavaScript API

Examining the PowerBiDaySpa

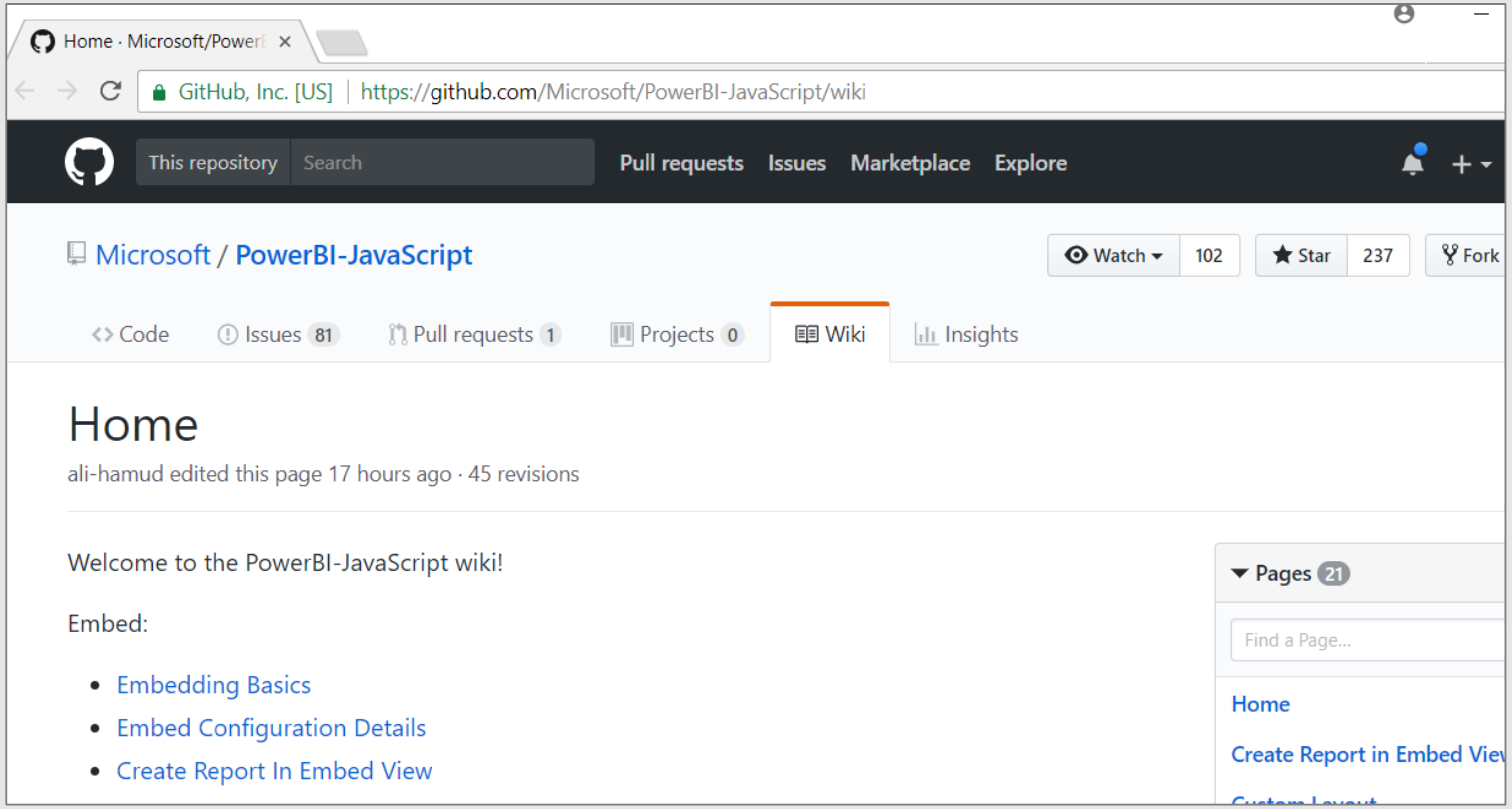
Examining the UserOwsDataConsumerApp

Agenda

- ✓ Power BI Embedding Fundamentals
- ✓ Authentication with Azure AD
- ✓ Programming the Power BI Service API
- ✓ App-only Authentication
- ➢ Single Page Applications (SPAs) with React.js
- Programming the Power BI JavaScript API

Power BI JavaScript API (powerbi.js)

- Power BI JavaScript API used to embed resources in browser
  - GitHub repo at https://github.com/Microsoft/PowerBI-JavaScript/wiki
  - GitHub repository contains code, docs, wiki and issues list



Home

ali-hamud edited this page 17 hours ago · 45 revisions

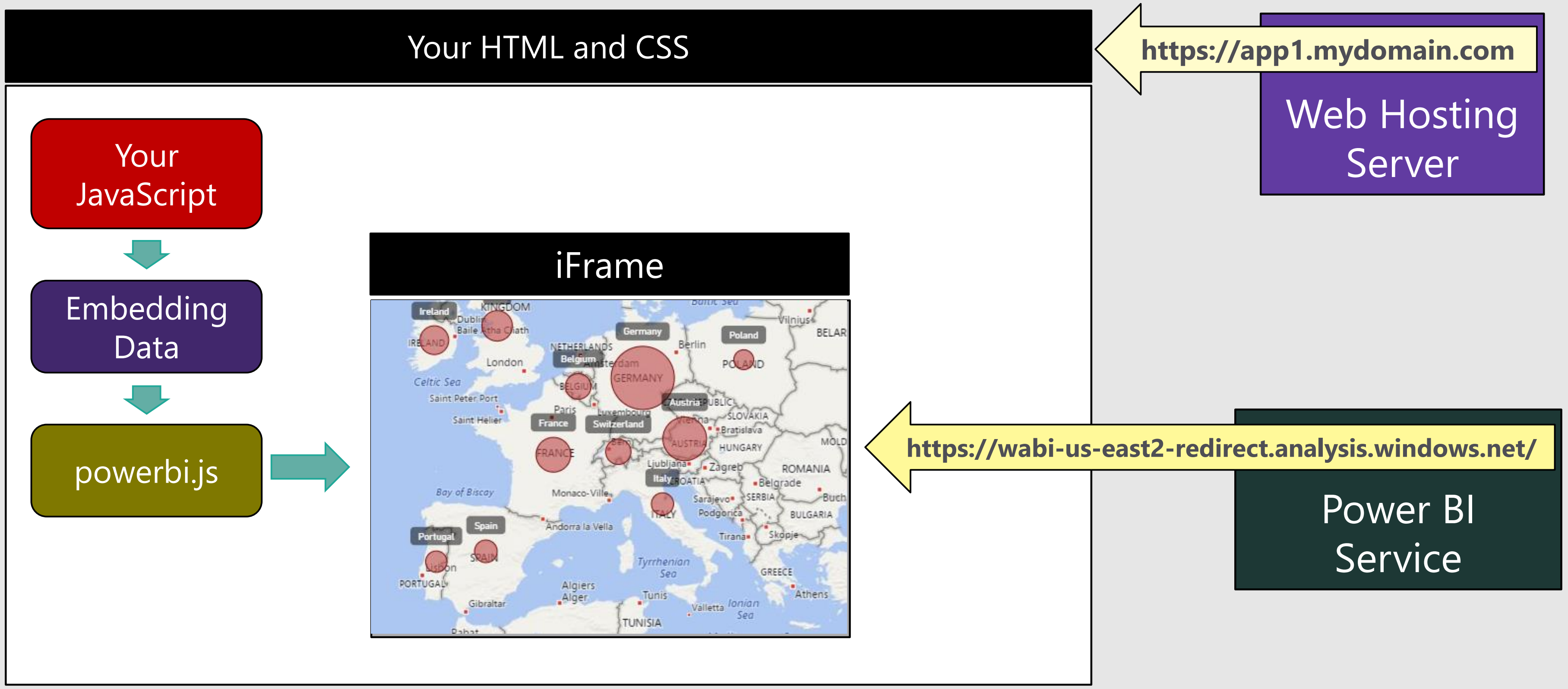
Welcome to the PowerBI-JavaScript wiki!

Embed:

- Embedding Basics
- Embed Configuration Details
- Create Report In Embed View

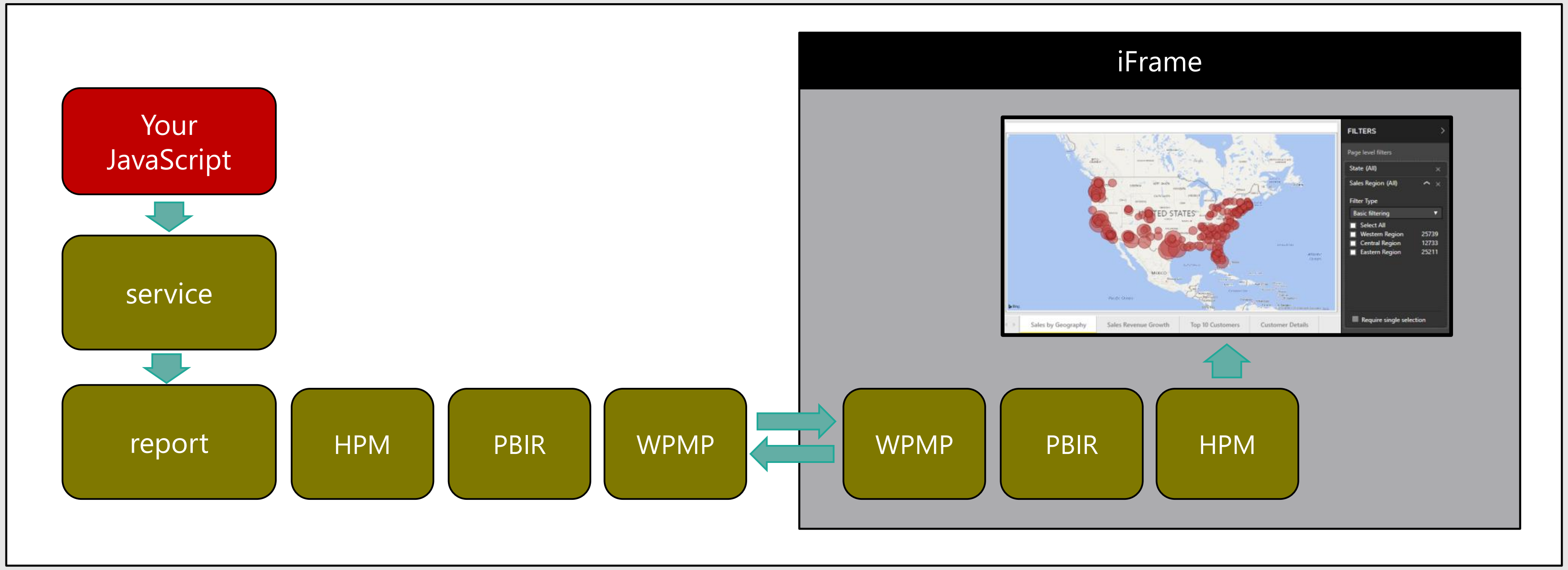
Report Embedding Architecture

- Embedding involves creating an iFrame on the page
  - PBIJS transparently creates iFrame and sets source to Power BI Service
  - ***The iFrame and hosting page originate from different DNS domains***



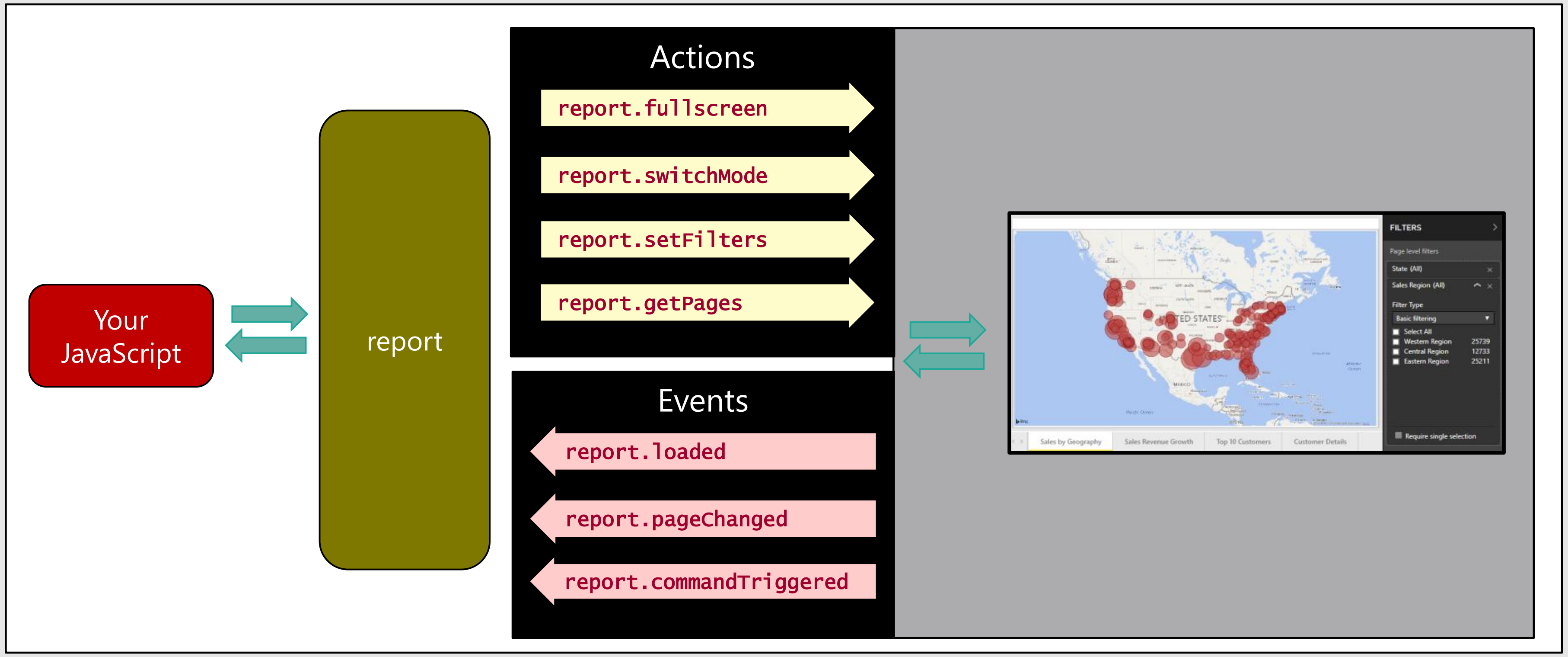
Post Message Communications Flow

- 4 extra libraries used communicate with report in iFrame
  - window-post-message-proxy (WPMP)
  - http-post-message (HPM)
  - powerbi-router (PBIR)
  - powerbi-models (PBIM)



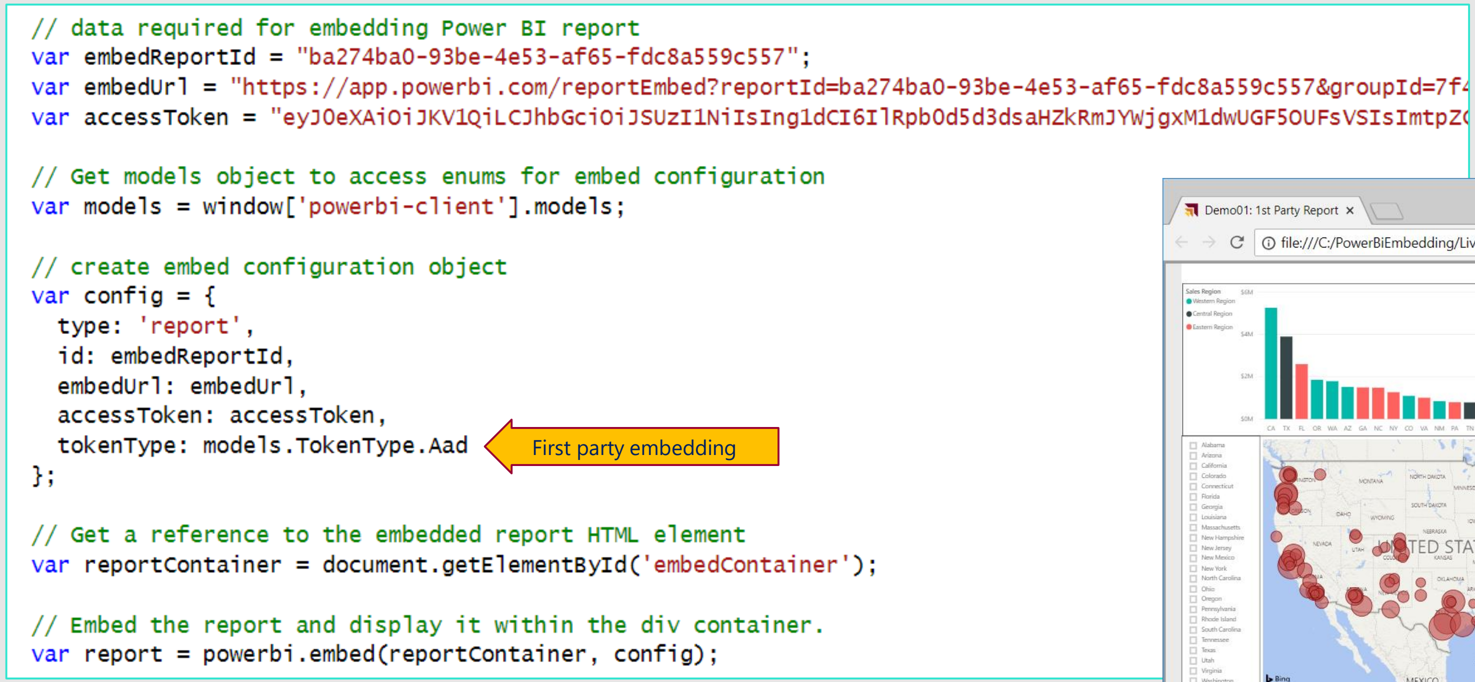
A Promise-based Programming Model

- Design of PBIJS simulates HTTP protocol
  - Creates more intuitive programming model for developers
  - Programming based on asynchronous requests and promises
  - Embedded objects programmed using actions and events



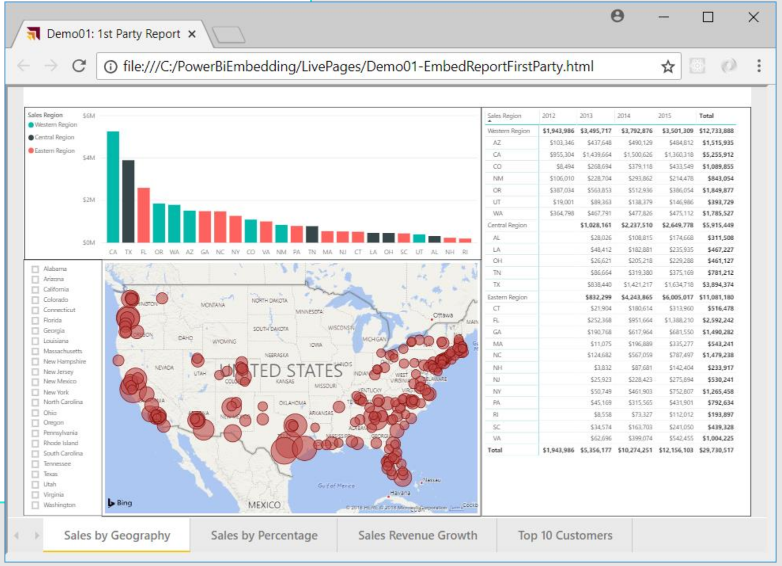
Hello World with Power BI Embedding

- powerbi.js library provides powerbi as top-level service object
  - You call powerbi.embed and pass configuration object with access token
  - models object available to supply configuration settings
  - configuration object sets tokenType to either TokenType.Embed or TokenType.Aad



```
// data required for embedding Power BI report
var embedReportId = "ba274ba0-93be-4e53-af65-fdc8a559c557";
var embedUrl = "https://app.powerbi.com/reportEmbed?reportId=ba274ba0-93be-4e53-af65-fdc8a559c557&groupId=7f4
var accessToken = "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IlRpb0d5d3dsaHZkRmJYWjgxM1dwUGF5OUFsVSIsImtpZC

// Get models object to access enums for embed configuration
var models = window['powerbi-client'].models;

// create embed configuration object
var config = {
  type: 'report',
  id: embedReportId,
  embedUrl: embedUrl,
  accessToken: accessToken,
  tokenType: models.TokenType.Aad
};

// Get a reference to the embedded report HTML element
var reportContainer = document.getElementById('embedContainer');

// Embed the report and display it within the div container.
var report = powerbi.embed(reportContainer, config);
```

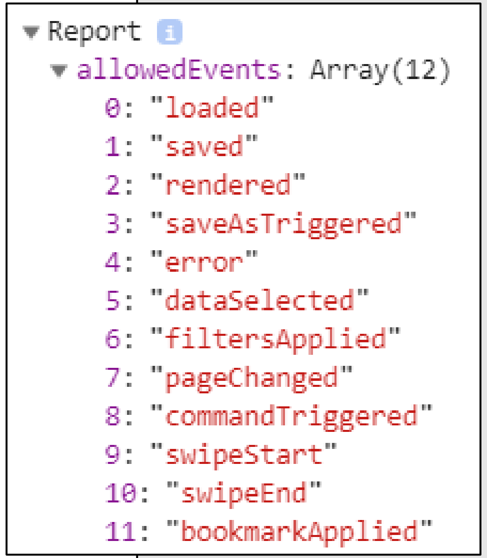
First party embedding



Sales Region	2012	2013	2014	2015	Total
Western Region	$1,943,986	$3,495,717	$3,792,876	$3,501,309	$12,733,888
AZ	$103,346	$437,648	$490,129	$484,812	$1,515,935
CA	$955,304	$1,439,664	$1,500,626	$1,360,318	$5,255,912
CO	$8,494	$268,694	$379,118	$433,549	$1,089,855
NM	$106,010	$228,704	$293,862	$214,478	$843,054
OR	$387,034	$563,853	$512,936	$386,054	$1,849,877
UT	$19,001	$89,363	$138,379	$146,986	$393,729
WA	$364,798	$467,791	$477,826	$475,112	$1,785,527
Central Region		$1,028,161	$2,237,510	$2,649,778	$5,915,449
AL		$28,026	$108,815	$174,668	$311,508
LA		$48,412	$182,881	$235,935	$467,227
OH		$26,621	$205,218	$229,288	$461,127
TN		$86,664	$319,380	$375,169	$781,212
TX		$838,440	$1,421,217	$1,634,718	$3,894,374
Eastern Region		$832,299	$4,243,865	$6,005,017	$11,081,180
CT		$21,904	$180,614	$313,960	$516,478
FL		$252,368	$951,664	$1,388,210	$2,592,242
GA		$190,768	$617,964	$681,550	$1,490,282
MA		$11,075	$196,889	$335,277	$543,241
NC		$124,682	$567,059	$787,497	$1,479,238
NH		$3,832	$87,681	$142,404	$233,917
NJ		$25,923	$228,423	$275,894	$530,241
NY		$50,749	$461,903	$752,807	$1,265,458
PA		$45,169	$315,565	$431,901	$792,634
RI		$8,558	$73,327	$112,012	$193,897
SC		$34,574	$163,703	$241,050	$439,328
VA		$62,696	$399,074	$542,455	$1,004,225
Total	$1,943,986	$5,356,177	$10,274,251	$12,156,103	$29,730,517

Handling Report Events

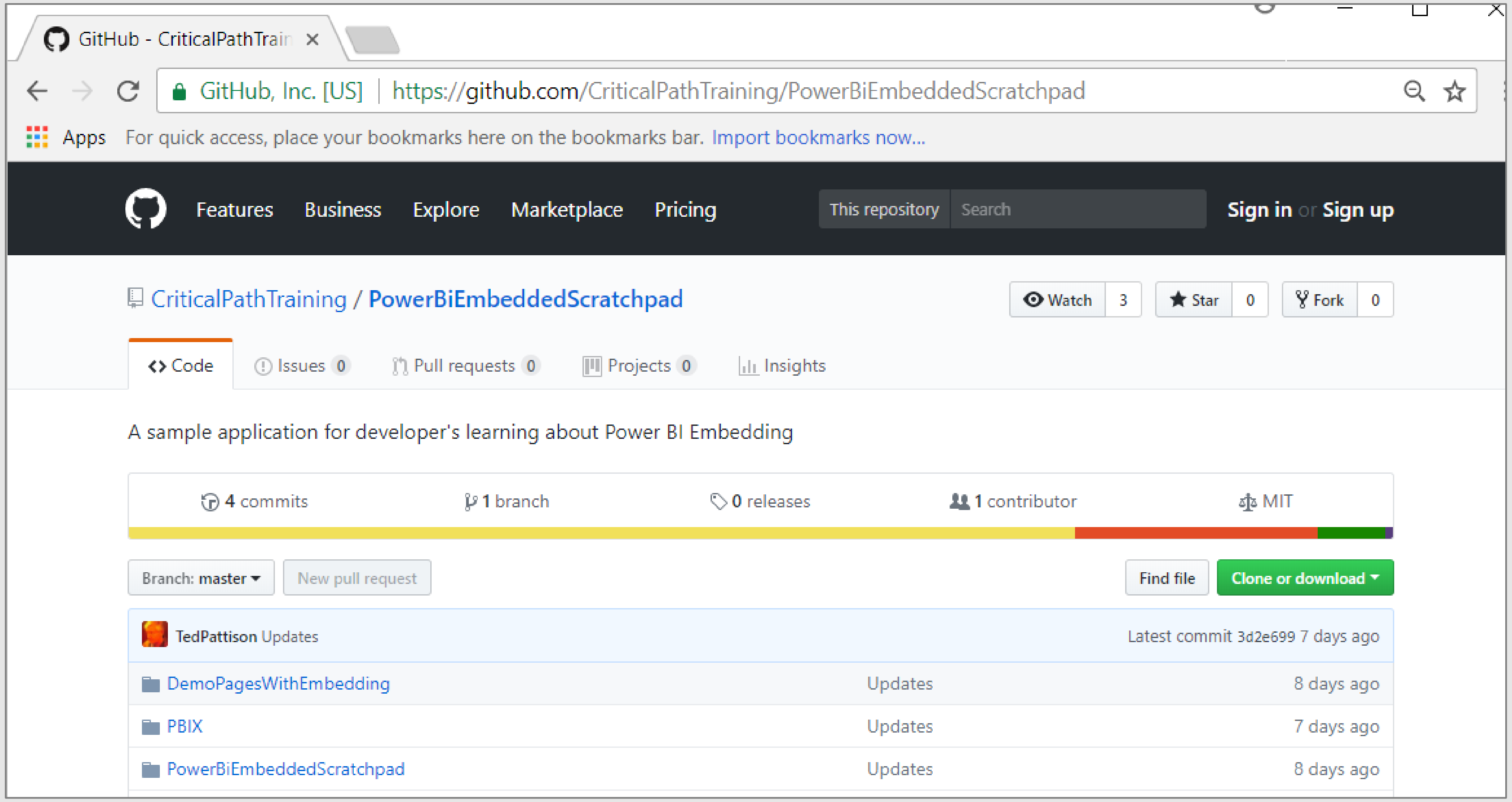
```
var report = powerbi.embed(embedContainer, config);

var pages;

report.on('loaded', function () {
  // call getPages with callback
  report.getPages().then(
    function (reportPages) {
      pages = reportPages;
      // call method to load pages into nav menu
      loadReportPages(pages);
    });
});

var loadReportPages = function (pages) {
  for (var index = 0; index < pages.length; index++) {
    // determine which pages are visible and not hidden
    if (pages[index].visibility == 0) { // 0 means visible and 1 means hidden
      var reportPageDisplayName = pages[index].displayName;
      pageNavigation.append($("<li>")
        .append($('<a href="javascript:;" >')
        .text(pages[index].displayName))
        .click(function (domEvent) {
          var targetPageName = domEvent.target.textContent;
          // get target page from pages collection
          var targetPage = pages.find(function (page) { return page.displayName === targetPageName; });
          // navigate report to target page
          targetPage.setActive();
        }));
    }
  }
}
```



```
▼ Report
  ▼ allowedEvents: Array(12)
      0: "loaded"
      1: "saved"
      2: "rendered"
      3: "saveAsTriggered"
      4: "error"
      5: "dataSelected"
      6: "filtersApplied"
      7: "pageChanged"
      8: "commandTriggered"
      9: "swipeStart"
      10: "swipeEnd"
      11: "bookmarkApplied"
```

PowerBiEmbeddedScratchpad Sample

- https://github.com/CriticalPathTraining/PowerBiEmbeddedScratchpad



CriticalPathTraining / PowerBiEmbeddedScratchpad

A sample application for developer's learning about Power BI Embedding

4 commits | 1 branch | 0 releases | 1 contributor | MIT

TedPattison Updates — Latest commit 3d2e699 7 days ago

DemoPagesWithEmbedding	Updates	8 days ago
PBIX	Updates	7 days ago
PowerBiEmbeddedScratchpad	Updates	8 days ago

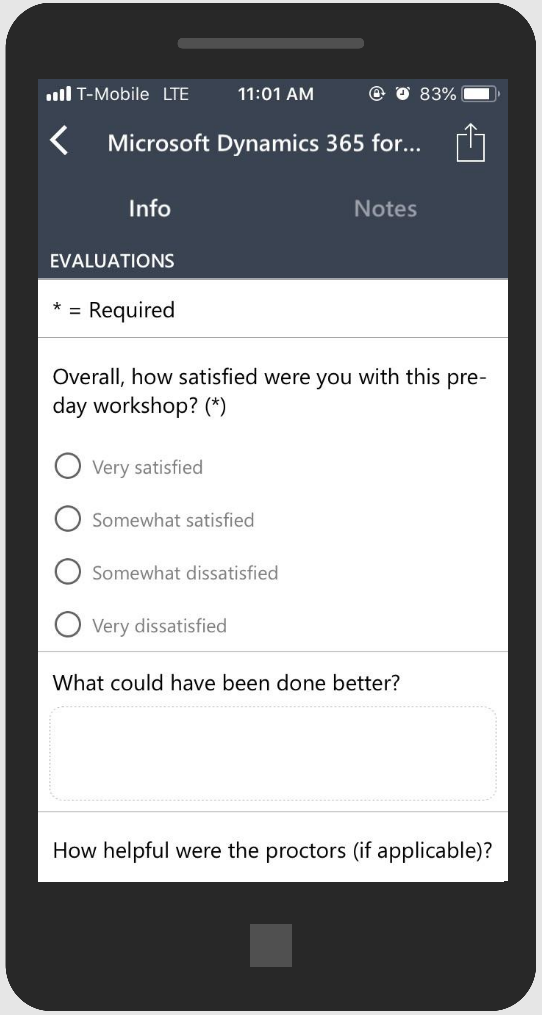
DEMO

The Power BI Embedded Scratchpad App

Summary

- ✓ Power BI Embedding Fundamentals
- ✓ Authentication with Azure AD
- ✓ Programming the Power BI Service API
- ✓ App-only Authentication
- ✓ Single Page Applications (SPAs) with React.js
- ✓ Programming the Power BI JavaScript API

Evaluate this session



T-Mobile LTE 11:01 AM 83%

Microsoft Dynamics 365 for...

Info Notes

EVALUATIONS

* = Required

Overall, how satisfied were you with this pre-day workshop? (*)

- Very satisfied
- Somewhat satisfied
- Somewhat dissatisfied
- Very dissatisfied

What could have been done better?

How helpful were the proctors (if applicable)?

Complete your session and conference evaluations through the mobile app or https://aka.ms/MBASeventapp

Please select Microsoft Business Applications Summit from the menu of events and then log in with your registration credentials

Microsoft