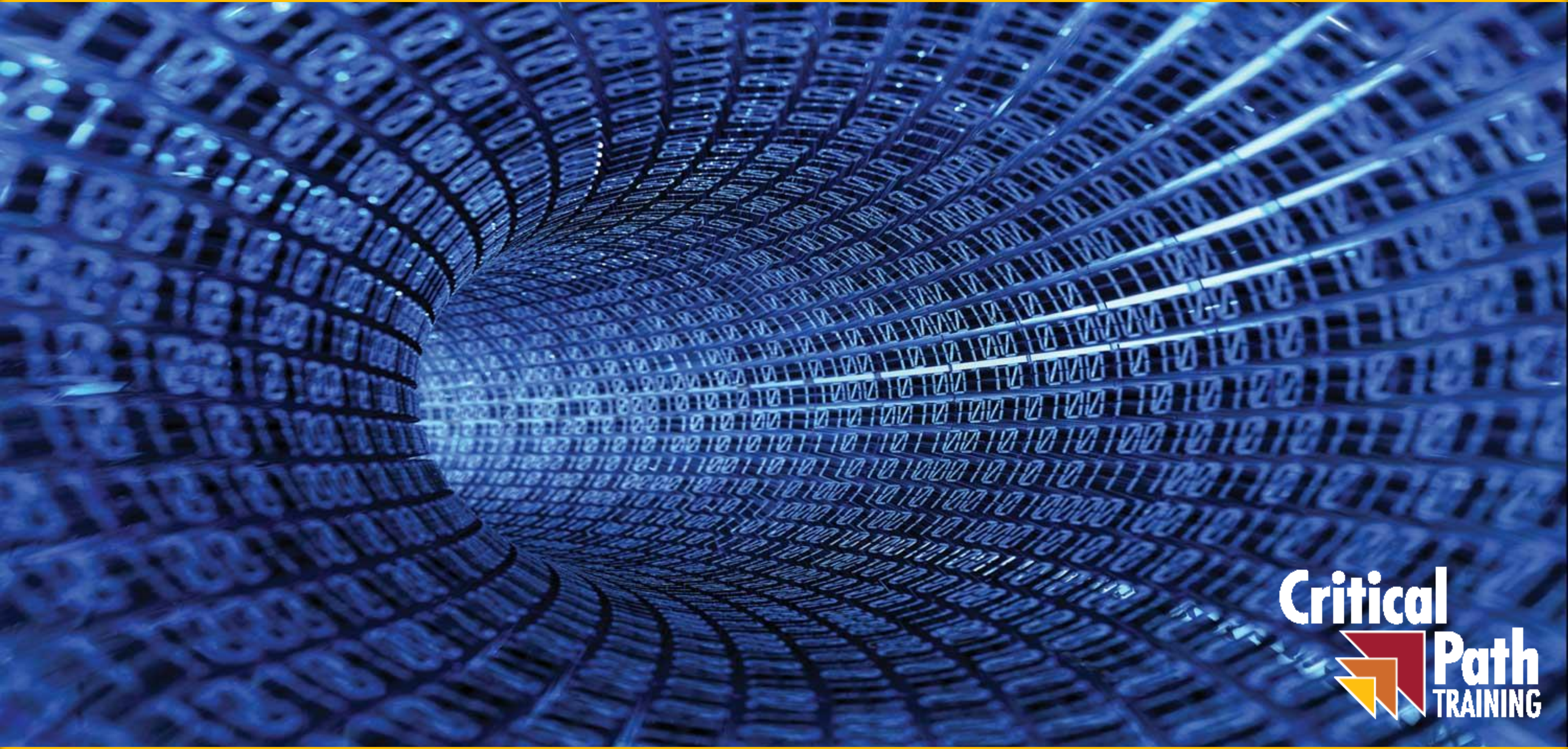
Programming the Power BI Service API



Agenda

- Power BI Service API Overview
- Understanding OAuth 2.0 and OpenID Connect
- Creating & Configuring Azure AD Applications
- Acquiring Access Tokens using ADAL
- Programming with the Power BI Service SDK
- Acquiring Access Tokens using MSAL

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

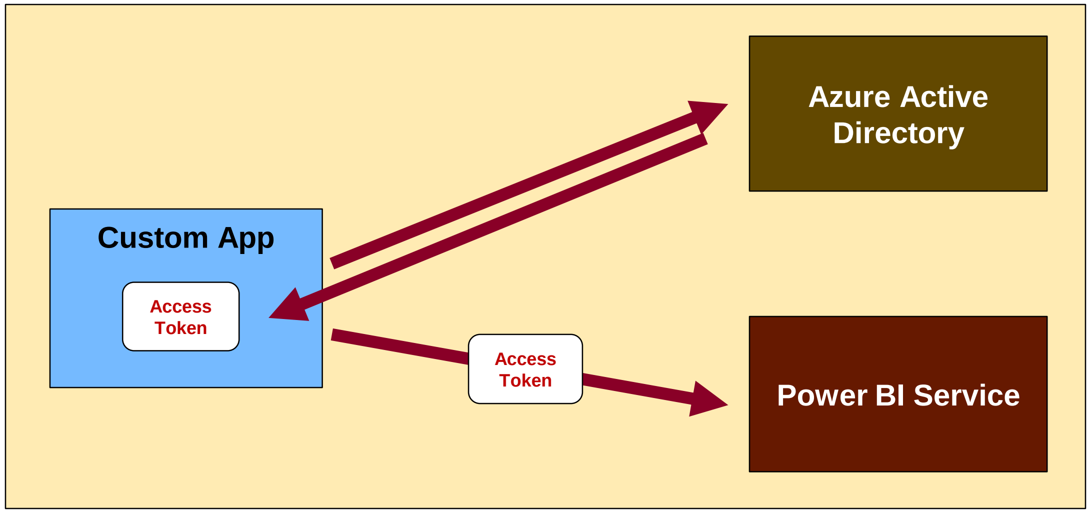
- Power BI User APIs (e.g. `GetGroupsAsync`)
  - provides users with access to personal workspace
  - provides users with access to app workspaces
  - provides service principal (SP) with access to app workspaces
- Power BI Admin APIs (e.g. `GetGroupsAsAdminAsync`)
  - provides users with tenant-level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

Getting Started

- What you need to get started?
  - Visual Studio 2017 or Visual Studio 2015
  - Organizational account in an Azure AD tenancy
  - License for Power BI Pro
  - Access to Azure portal to create Azure AD applications
- Azure subscription not required!
  - Azure portal used to create Azure AD application
  - Azure subscription helpful to create Azure resources

Authenticating with Azure AD

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

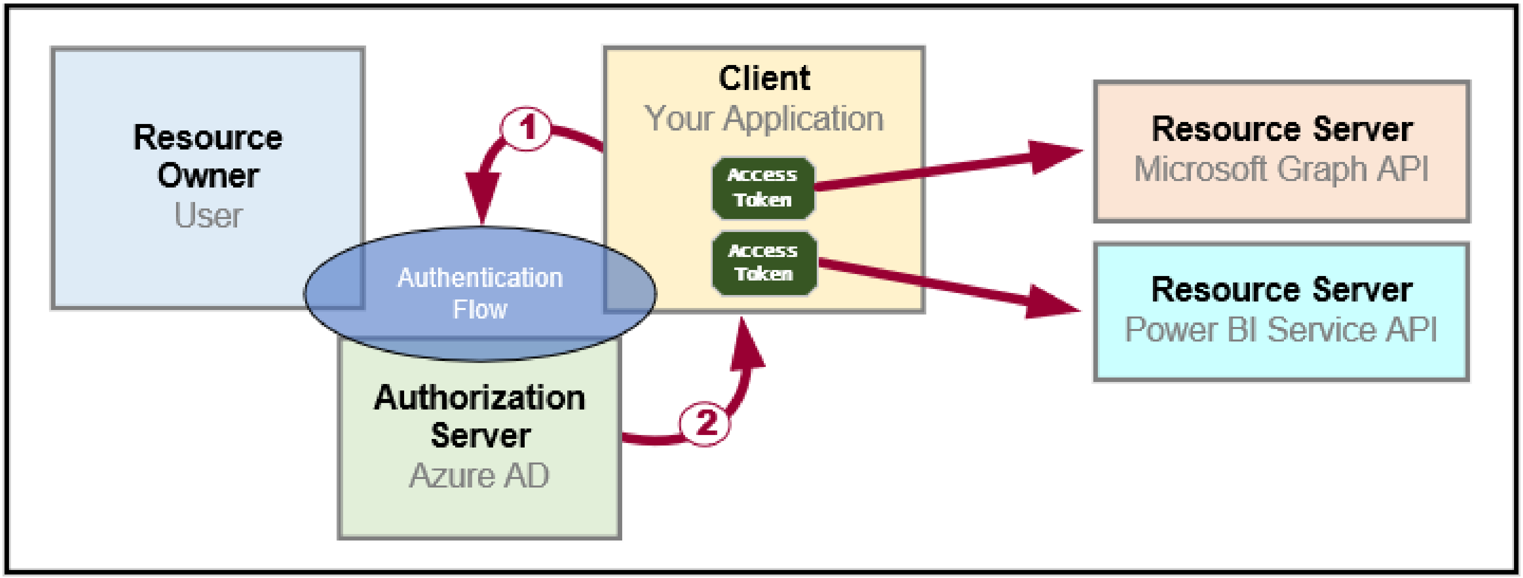


Agenda

- ✓ Power BI Service API Overview
- ➢ Understanding OAuth 2.0 and OpenID Connect
- Creating & Configuring Azure AD Applications
- Acquiring Access Tokens using ADAL
- Programming with the Power BI Service SDK
- Acquiring Access Tokens using MSAL

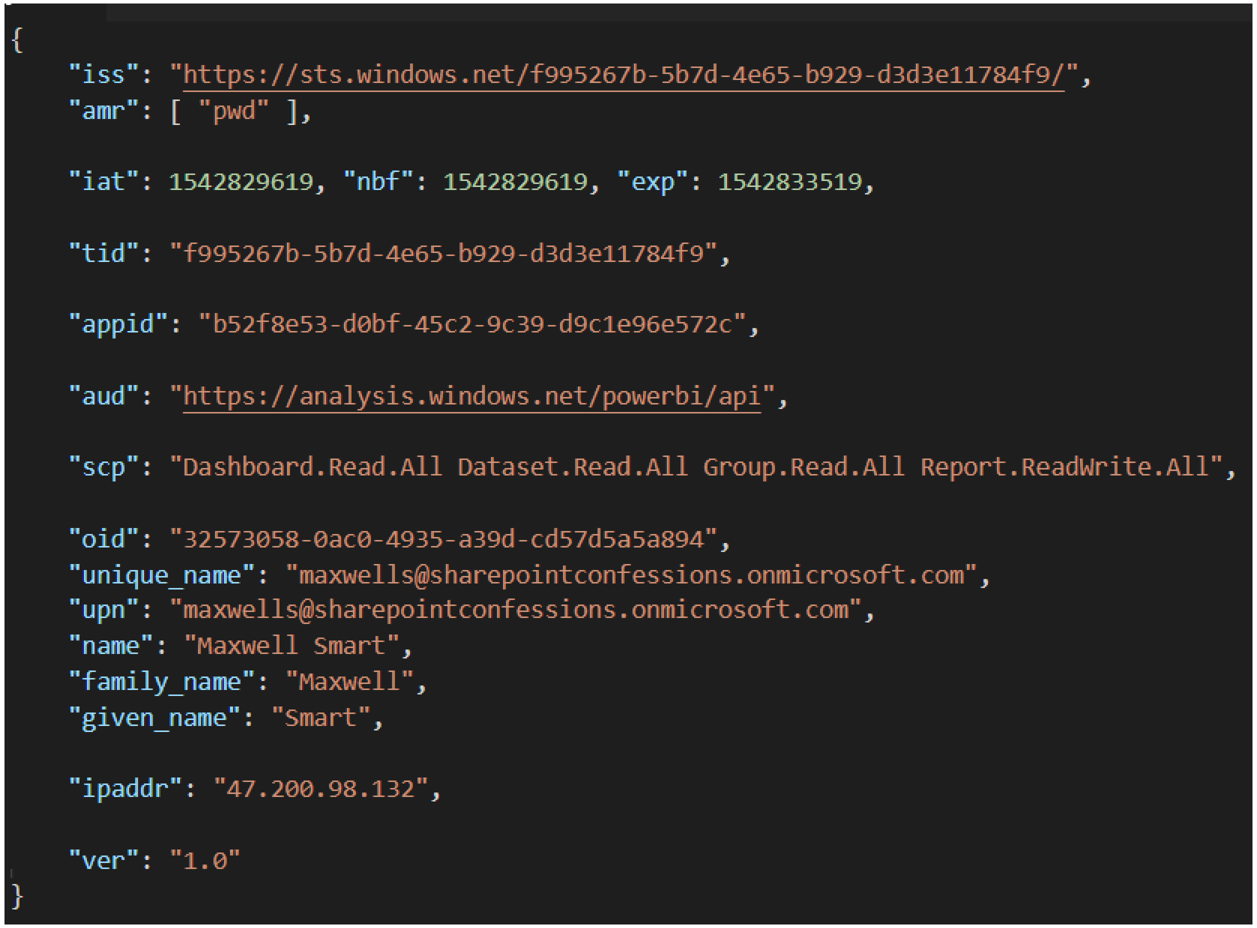
OAuth 2.0 Fundamentals

- Client application calls to resource server on behalf of a user
  - Client implements authentication flow to acquire access token
  - Access token contains permission grants for client to call resource server
  - Client passes access token when calling to resource server
  - Resource server inspects access token to ensure client has permissions



Access Token is a Bearer Token

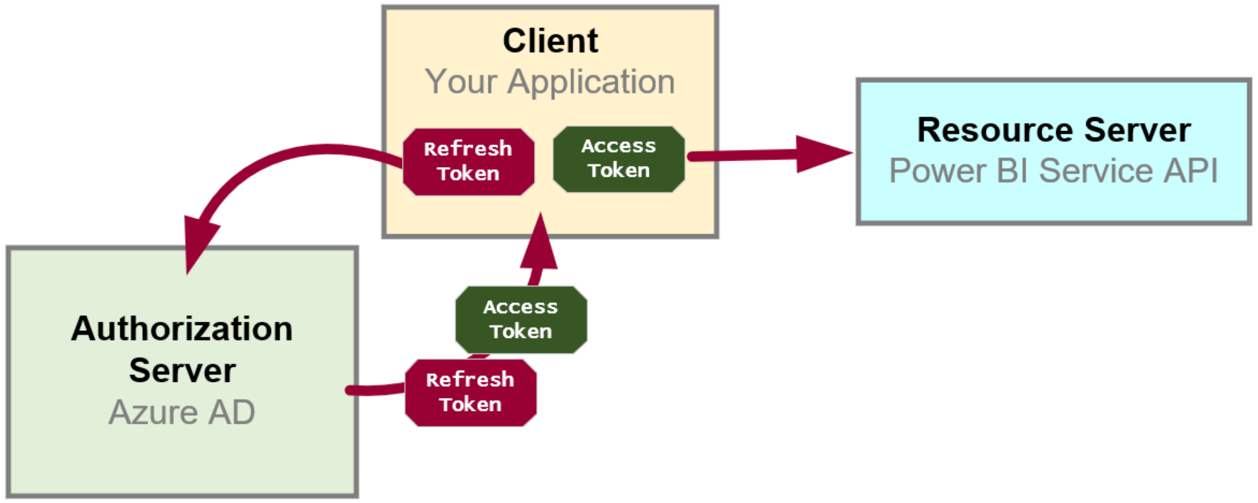
- It can be used by any who bears (e.g. steals) it
  - Always encrypt with HTTPS when transmitting access tokens



```
{
    "iss": "https://sts.windows.net/f995267b-5b7d-4e65-b929-d3d3e11784f9/",
    "amr": [ "pwd" ],

    "iat": 1542829619, "nbf": 1542829619, "exp": 1542833519,

    "tid": "f995267b-5b7d-4e65-b929-d3d3e11784f9",

    "appid": "b52f8e53-d0bf-45c2-9c39-d9c1e96e572c",

    "aud": "https://analysis.windows.net/powerbi/api",

    "scp": "Dashboard.Read.All Dataset.Read.All Group.Read.All Report.ReadWrite.All",

    "oid": "32573058-0ac0-4935-a39d-cd57d5a5a894",
    "unique_name": "maxwells@sharepointconfessions.onmicrosoft.com",
    "upn": "maxwells@sharepointconfessions.onmicrosoft.com",
    "name": "Maxwell Smart",
    "family_name": "Maxwell",
    "given_name": "Smart",

    "ipaddr": "47.200.98.132",

    "ver": "1.0"
}
```

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
  - Refresh tokens lesson need for user to enter security credentials

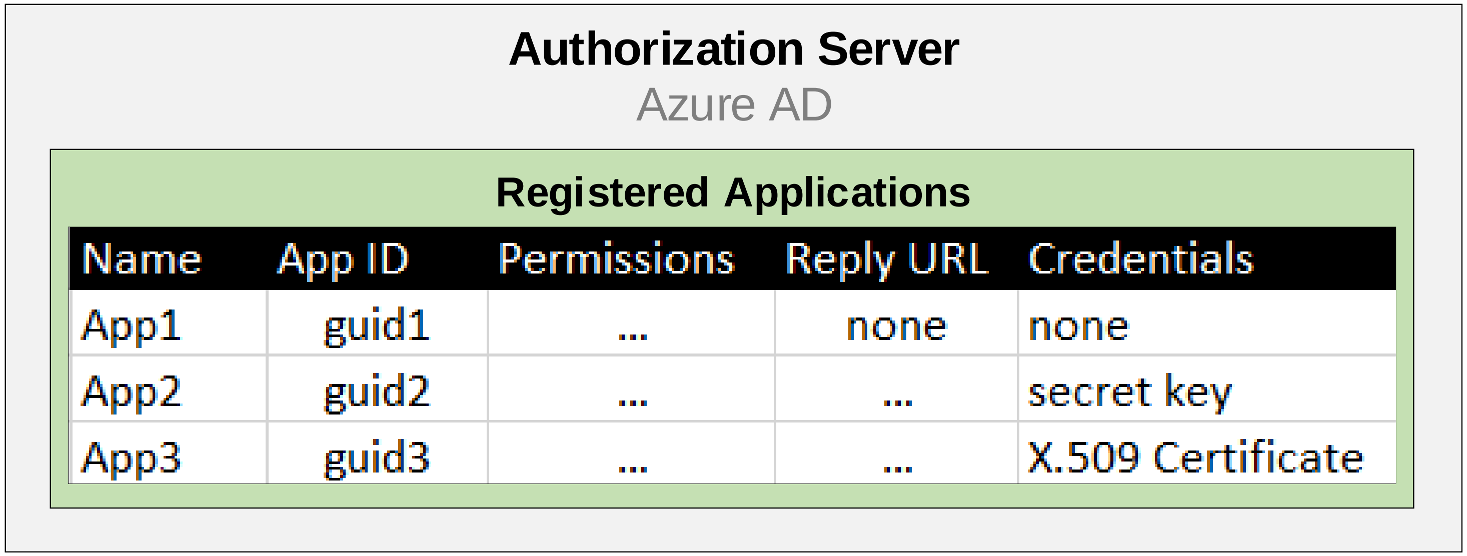


Authentication Flows

- **User Password Credential Flow** *(public client)*
  - Used in Native clients to obtain access code
  - Requires passing user name and password across network
- **Authorization Code Flow** *(confidential client)*
  - Client first obtains authorization code then access token
  - Access token acquired in server-to-server call
  - Access token never passes through browser or client device
- **Implicit Flow** *(public client)*
  - Used in SPAs built with JavaScript and AngularJS
  - Application obtains access token w/o acquiring authorization code
- **Client Credentials Flow** *(confidential client)*
  - Authentication based on SSL certificate with public-private key pair
  - Used to obtain access token when using app-only permissions

OAuth 2.0 Client Registration

- Client must be registered with authorization server
  - Authorization server tracks each client with unique Client ID
  - Client should be registered with one or more Reply URLs
  - Reply URL should be fixed endpoint on Internet
  - Reply URL used to transmit security tokens to clients
  - Client registration tracks permissions and other attributes

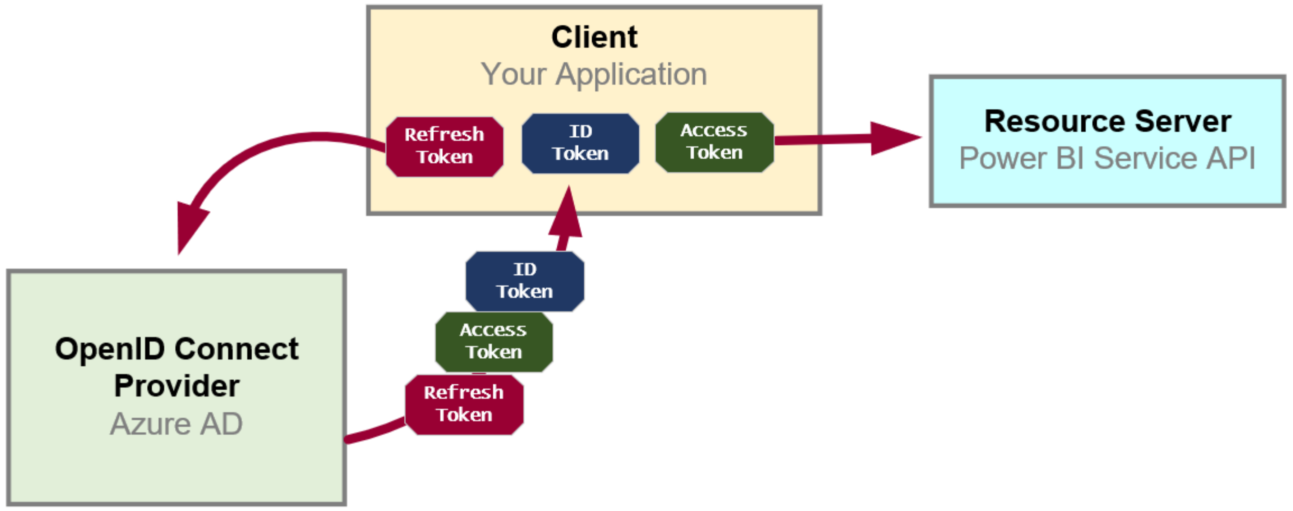

Authorization Server
Azure AD

Registered Applications

Name	App ID	Permissions	Reply URL	Credentials
App1	guid1	...	none	none
App2	guid2	...	...	secret key
App3	guid3	...	...	X.509 Certificate

OpenID Connect Extends OAuth 2.0

- 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

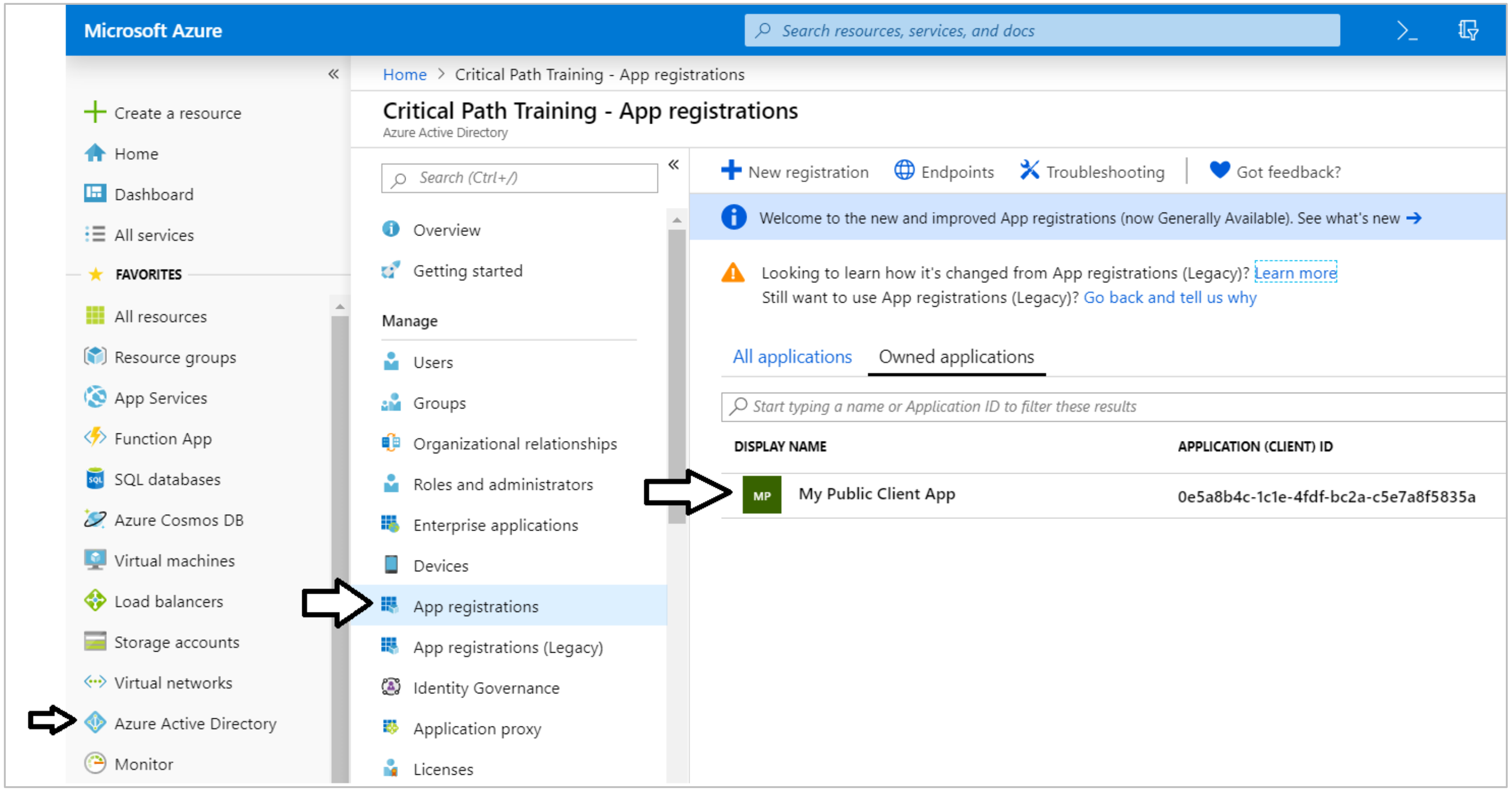


Agenda

- ✓ Power BI Service API Overview
- ✓ Understanding OAuth 2.0 and OpenID Connect
- ➢ Creating & Configuring Azure AD Applications
- Acquiring Access Tokens using ADAL
- Programming with the Power BI Service SDK
- Acquiring Access Tokens using MSAL

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



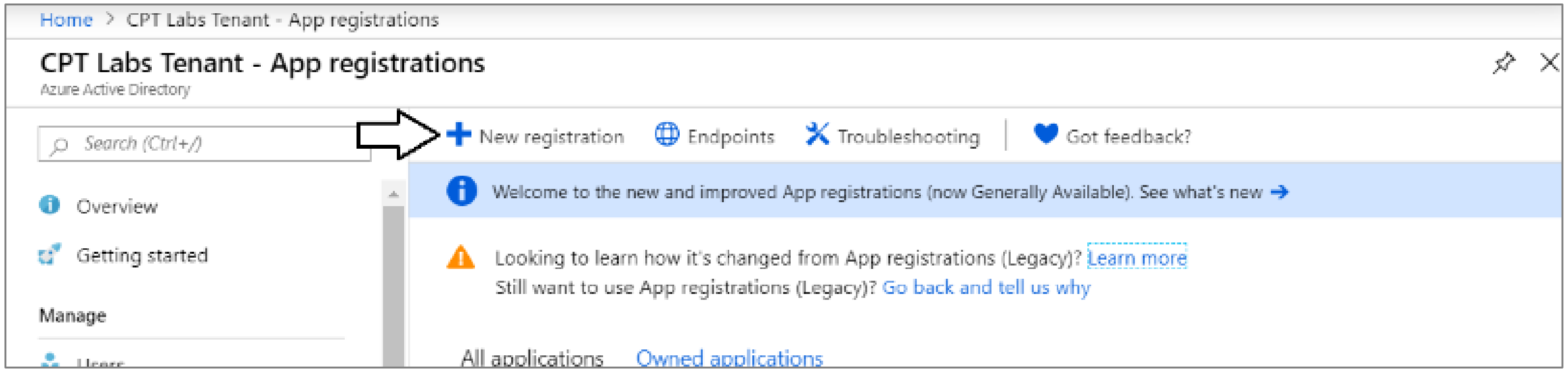
Welcome to the new and improved App registrations (now Generally Available). See what's new →

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 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



Home > CPT Labs Tenant - App registrations

CPT Labs Tenant - App registrations

Azure Active Directory

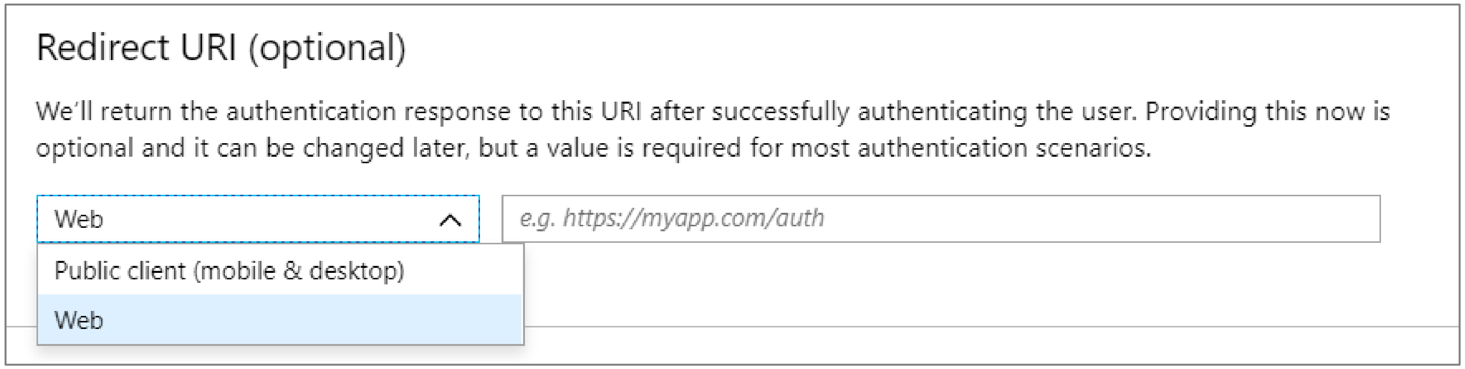
Welcome to the new and improved App registrations (now Generally Available). See what's new

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

Application Types

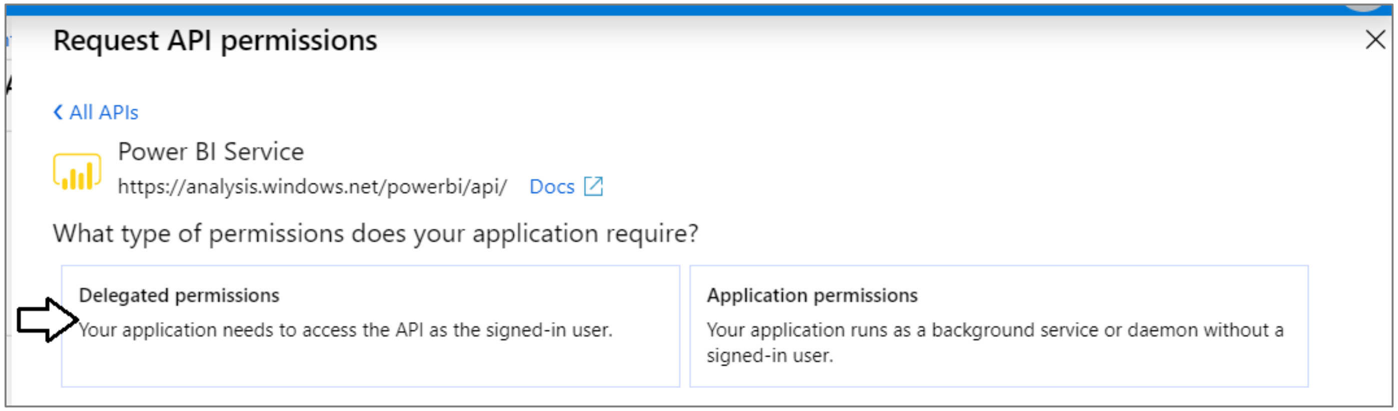
- Azure AD Application Types
  - Public client (mobile and desktop)
  - Web


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.

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 support application permission



Request API permissions

< All APIs

Power BI Service

https://analysis.windows.net/powerbi/api/ Docs

What type of permissions does your application require?

Delegated permissions

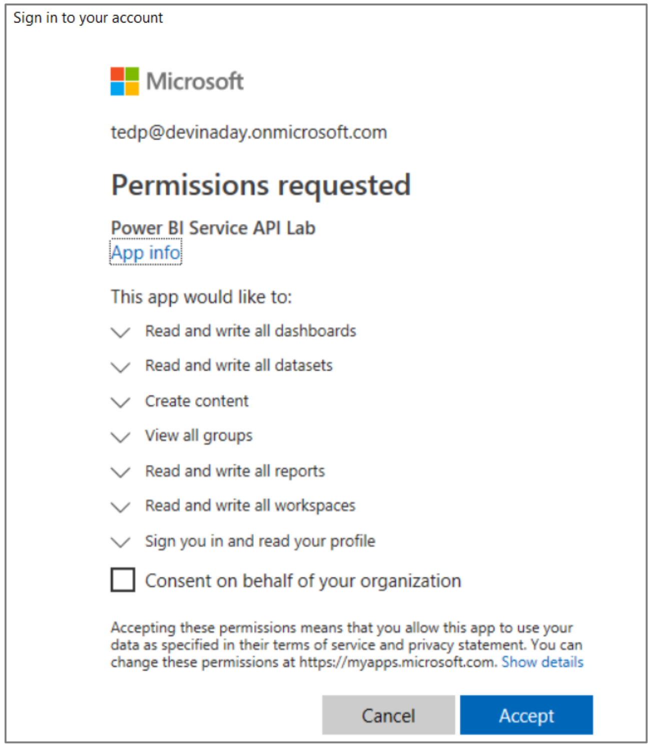
Your application needs to access the API as the signed-in user.

Application permissions

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

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


Sign in to your account

Microsoft

tedp@devinaday.onmicrosoft.com

Permissions requested

Power BI Service API Lab
App info

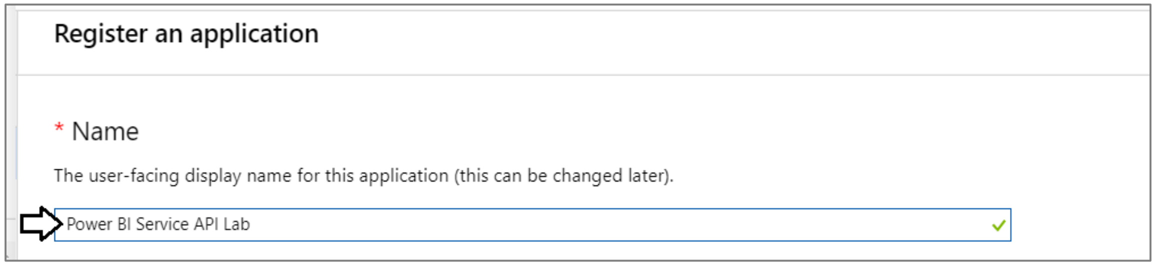
This app would like to:
- Read and write all dashboards
- Read and write all datasets
- Create content
- View all groups
- Read and write all reports
- Read and write all workspaces
- Sign you in and read your profile

☐ 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

Creating a Native Application

- Power BI supports Native applications
  - Can be used for desktop applications and Console applications
  - Can be used in third party embedding (App Owns Data model)
  - Application type should be configured as Public client
  - Requires Redirect URI with unique string - not an actual URL

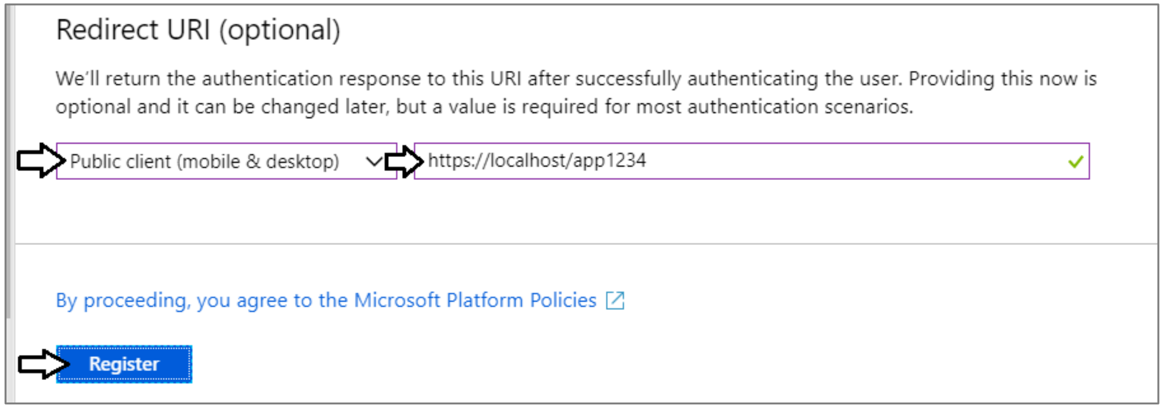


Register an application

* Name

The user-facing display name for this application (this can be changed later).

Power BI Service API Lab



Redirect URI (optional)

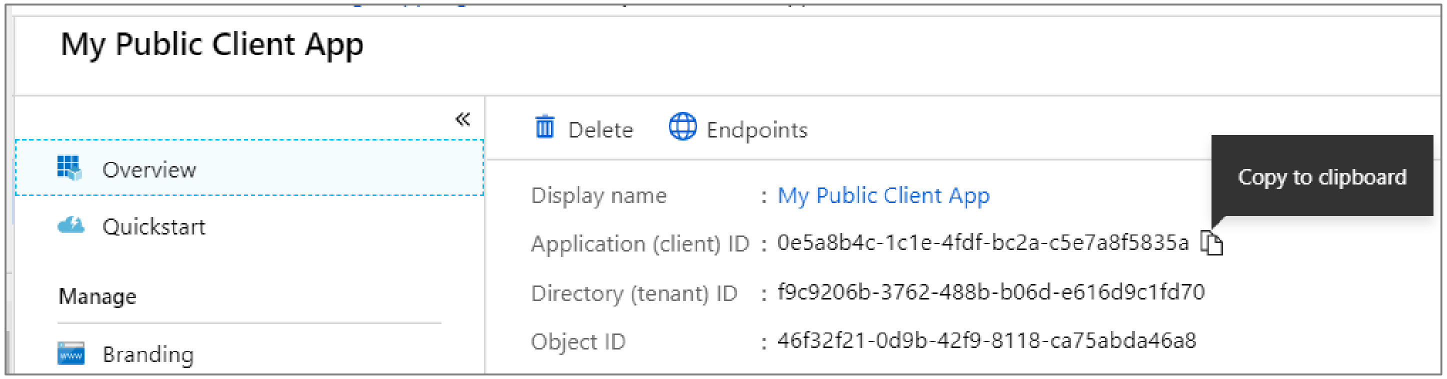
We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Public client (mobile & desktop) | https://localhost/app1234

By proceeding, you agree to the Microsoft Platform Policies

Copying the Application ID

- Each new application created with Application ID
  - You cannot supply your own GUID for application ID
  - Azure AD will always create this GUID
  - You can copy the application ID from the Azure portal

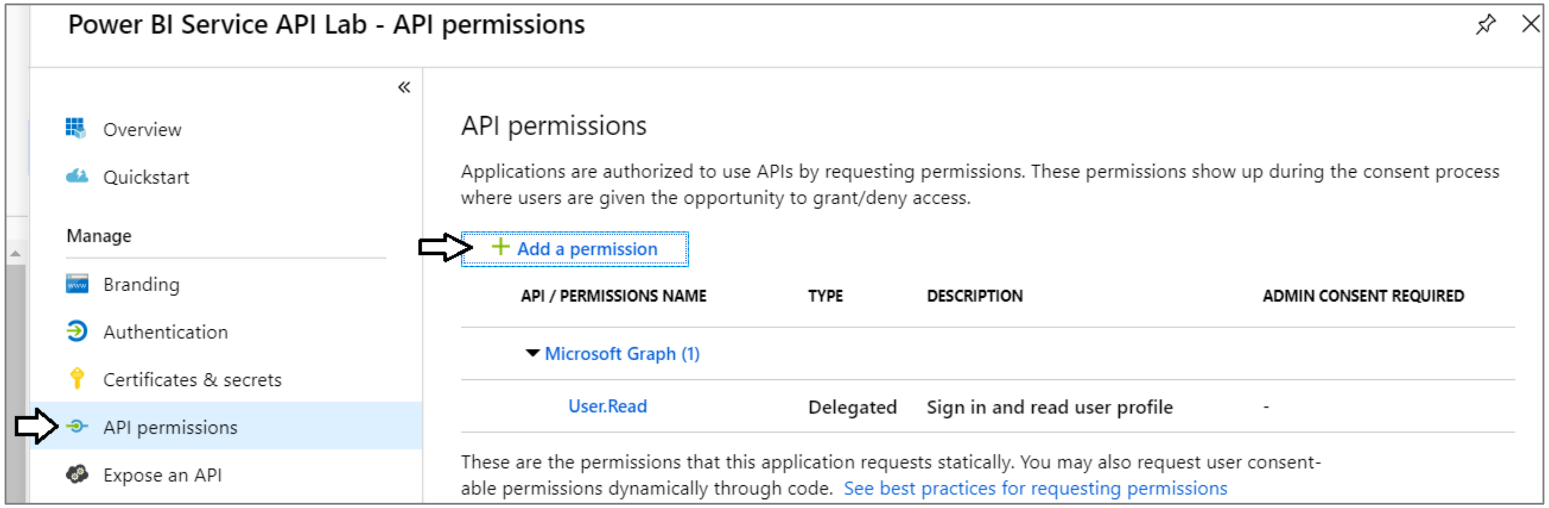


My Public Client App

Display name : My Public Client App
Application (client) ID : 0e5a8b4c-1c1e-4fdf-bc2a-c5e7a8f5835a
Directory (tenant) ID : f9c9206b-3762-488b-b06d-e616d9c1fd70
Object ID : 46f32f21-0d9b-42f9-8118-ca75abda46a8

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



Power BI Service API Lab - API permissions

API permissions

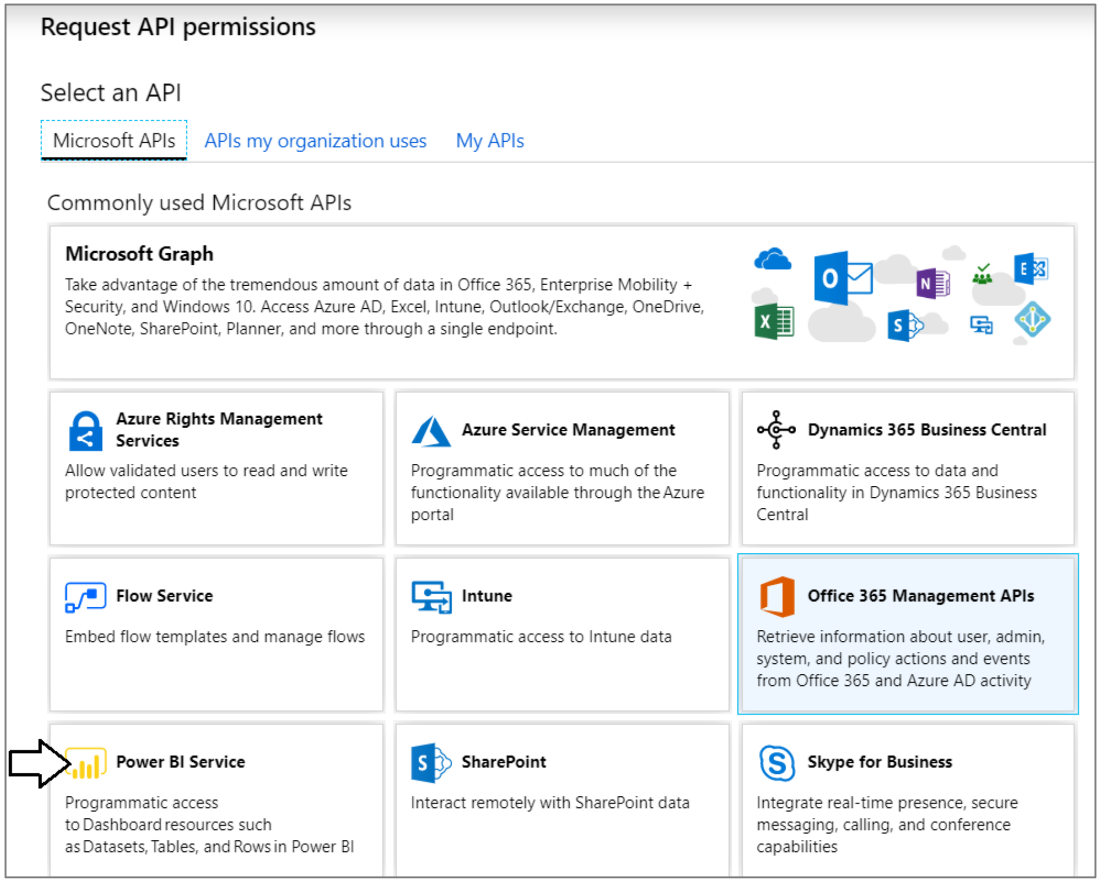
Applications are authorized to use APIs by requesting permissions. These permissions show up during the consent process where users are given the opportunity to grant/deny access.

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

Choosing an API

- There are lots of APIs to choose from
  - Microsoft Graph, Power BI Service, etc.



Request API permissions

Select an API

Microsoft APIs | APIs my organization uses | My APIs

Commonly used Microsoft APIs

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

Azure Rights Management Services
Allow validated users to read and write protected content

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

Dynamics 365 Business Central
Programmatic access to data and functionality in Dynamics 365 Business Central

Flow Service
Embed flow templates and manage flows

Intune
Programmatic access to Intune data

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

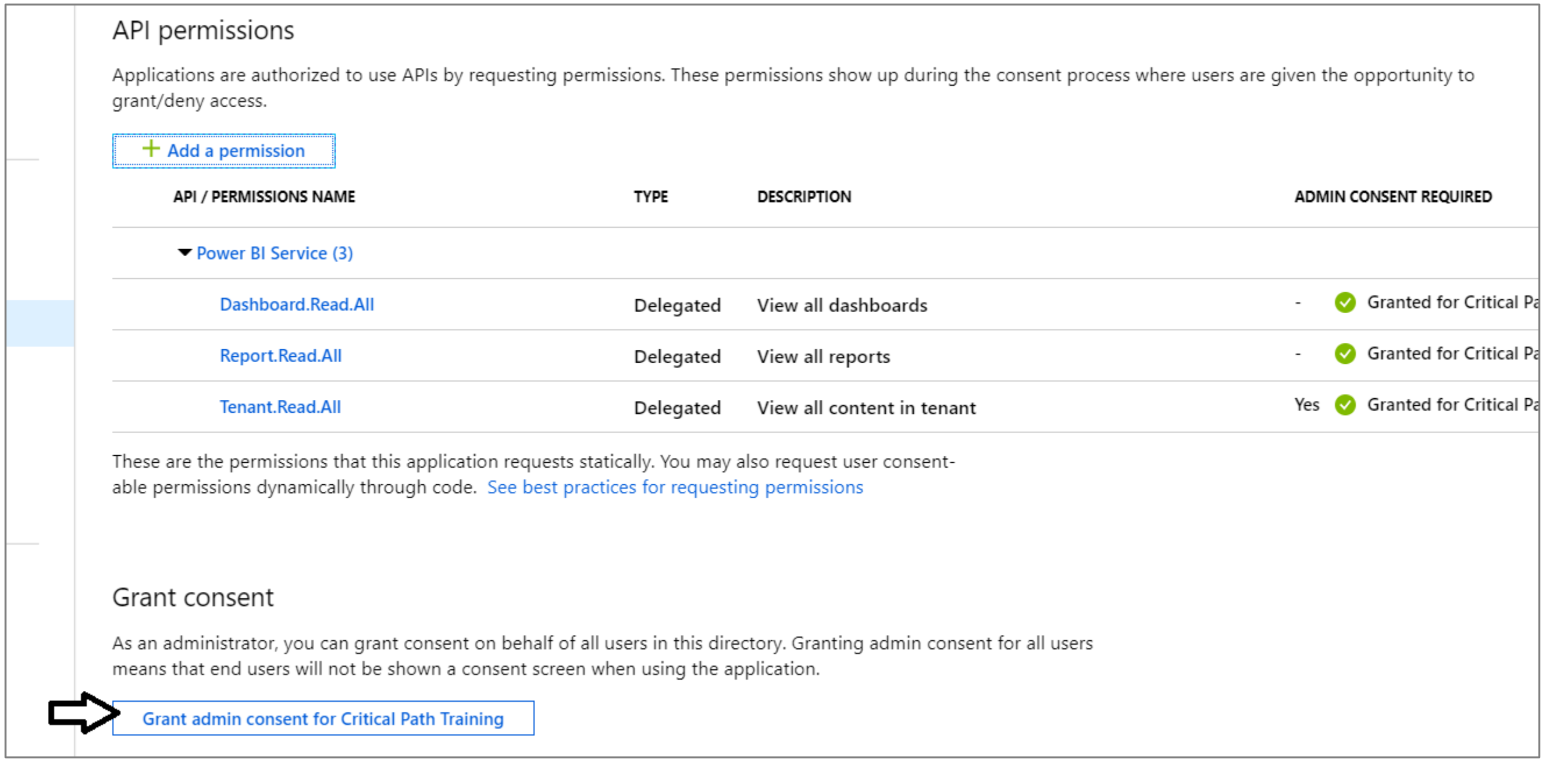
Power BI Service
Programmatic access to Dashboard resources such as Datasets, Tables, and Rows in Power BI

SharePoint
Interact remotely with SharePoint data

Skype for Business
Integrate real-time presence, secure messaging, calling, and conference capabilities

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 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	
▾ Power BI Service (3)				
Dashboard.Read.All	Delegated	View all dashboards	-	✓ Granted for Critical Pa
Report.Read.All	Delegated	View all reports	-	✓ Granted for Critical Pa
Tenant.Read.All	Delegated	View all content in tenant	Yes	✓ Granted for Critical Pa

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

Grant consent

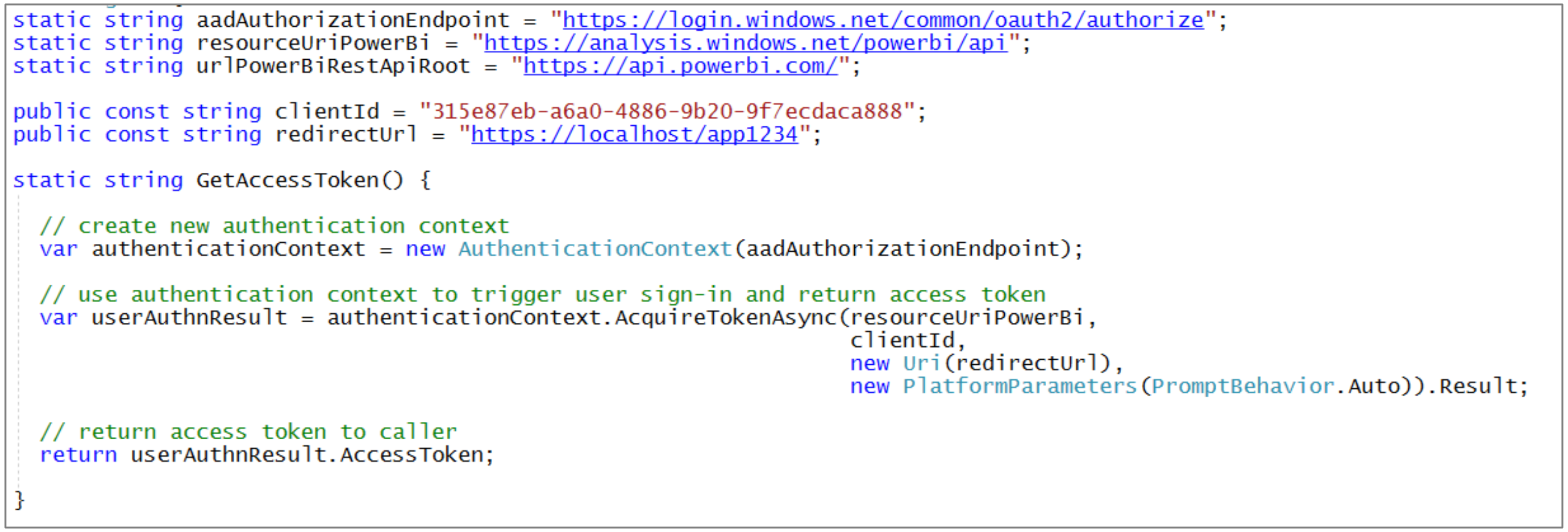
As an administrator, you can grant consent on behalf of all users in this directory. Granting admin consent for all users means that end users will not be shown a consent screen when using the application.

Agenda

- ✓ Power BI Service API Overview
- ✓ Understanding OAuth 2.0 and OpenID Connect
- ✓ Creating & Configuring Azure AD Applications
- ➢ Acquiring Access Tokens using ADAL
- Programming with the Power BI Service SDK
- Acquiring Access Tokens using MSAL

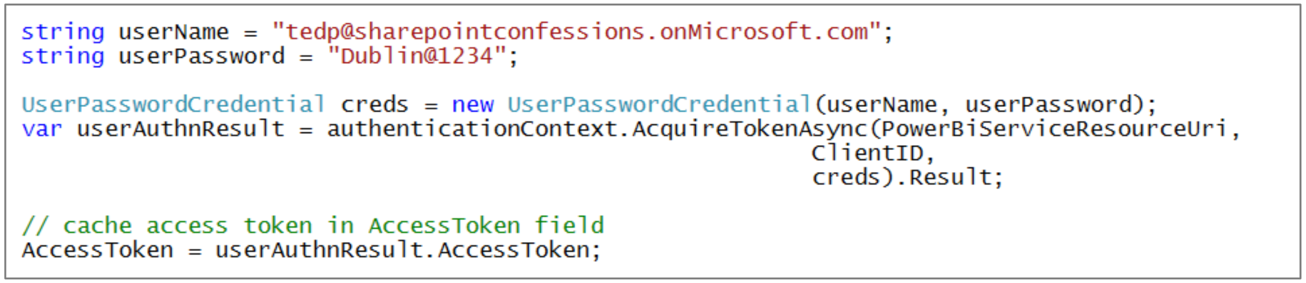
Access Token Acquisition (Native Client)

- With interactive login



```
static string aadAuthorizationEndpoint = "https://login.windows.net/common/oauth2/authorize";
static string resourceUriPowerBi = "https://analysis.windows.net/powerbi/api";
static string urlPowerBiRestApiRoot = "https://api.powerbi.com/";

public const string clientId = "315e87eb-a6a0-4886-9b20-9f7ecdaca888";
public const string redirectUrl = "https://localhost/app1234";

static string GetAccessToken() {

  // create new authentication context
  var authenticationContext = new AuthenticationContext(aadAuthorizationEndpoint);

  // use authentication context to trigger user sign-in and return access token
  var userAuthnResult = authenticationContext.AcquireTokenAsync(resourceUriPowerBi,
                                                                clientId,
                                                                new Uri(redirectUrl),
                                                                new PlatformParameters(PromptBehavior.Auto)).Result;

  // return access token to caller
  return userAuthnResult.AccessToken;

}
```

- With User Password Credential flow (non-interactive)



```
string userName = "tedp@sharepointconfessions.onMicrosoft.com";
string userPassword = "Dublin@1234";

UserPasswordCredential creds = new UserPasswordCredential(userName, userPassword);
var userAuthnResult = authenticationContext.AcquireTokenAsync(PowerBiServiceResourceUri,
                                                              ClientID,
                                                              creds).Result;

// cache access token in AccessToken field
AccessToken = userAuthnResult.AccessToken;
```

Access Token Acquisition (web app)

```
private static string aadInstance = "https://login.microsoftonline.com/";
private static string resourceUrlPowerBi = "https://analysis.windows.net/powerbi/api";
private static string urlPowerBiRestApiRoot = "https://api.powerbi.com/";

private static string clientId = ConfigurationManager.AppSettings["client-id"];
private static string clientSecret = ConfigurationManager.AppSettings["client-secret"];
private static string redirectUrl = ConfigurationManager.AppSettings["reply-url"];

private static async Task<string> GetAccessTokenAsync() {

  // determine authorization URL for current tenant
  string tenantID = ClaimsPrincipal.Current.FindFirst("http://schemas.microsoft.com/identity/claims/tenantid").Value;
  string tenantAuthority = aadInstance + tenantID;

  // create ADAL cache object
  ApplicationDbContext db = new ApplicationDbContext();
  string signedInUserID = ClaimsPrincipal.Current.FindFirst(ClaimTypes.NameIdentifier).Value;
  ADALTokenCache userTokenCache = new ADALTokenCache(signedInUserID);

  // create authentication context
  AuthenticationContext authenticationContext = new AuthenticationContext(tenantAuthority, userTokenCache);

  // create client credential object using client ID and client Secret"];
  ClientCredential clientCredential = new ClientCredential(clientId, clientSecret);

  // create user identifier object for logged on user
  string objectIdentifierId = "http://schemas.microsoft.com/identity/claims/objectidentifier";
  string userObjectID = ClaimsPrincipal.Current.FindFirst(objectIdentifierId).Value;
  UserIdentifier userIdentifier = new UserIdentifier(userObjectID, UserIdentifierType.UniqueId);

  // get access token for Power BI Service API from AAD
  AuthenticationResult authenticationResult =
    await authenticationContext.AcquireTokenSilentAsync(
        resourceUrlPowerBi,
        clientCredential,
        userIdentifier);

  // return access token back to user
  return authenticationResult.AccessToken;

}
```

REST Calls to the Power BI Service API

```
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();
  }
}
```

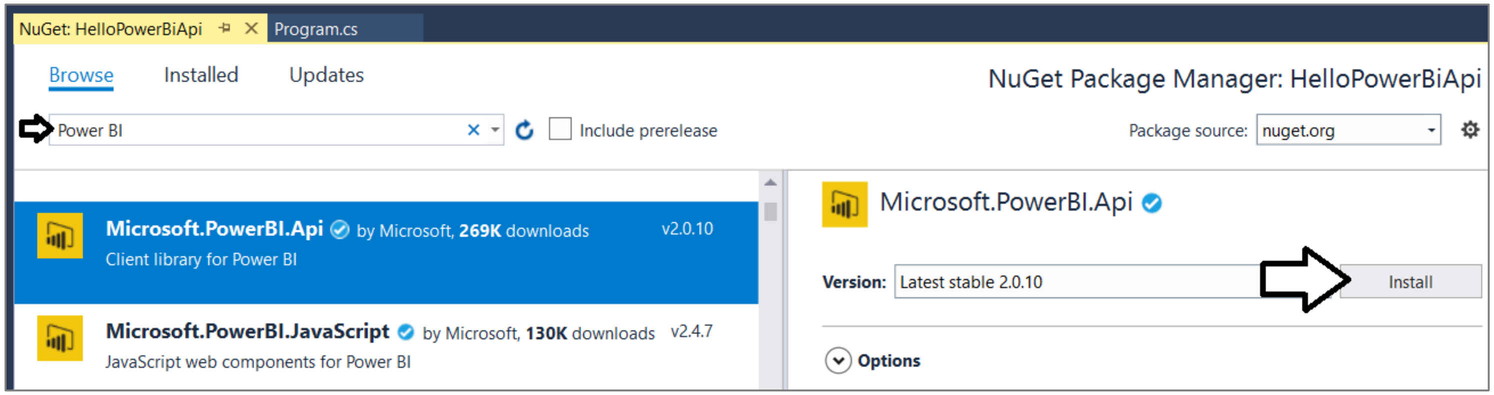
```
public class Report {
  public string id { get; set; }
  public string name { get; set; }
  public string webUrl { get; set; }
  public string embedUrl { get; set; }
  public bool isOwnedByMe { get; set; }
  public string datasetId { get; set; }
}

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

Agenda

- ✓ Power BI Service API Overview
- ✓ Understanding OAuth 2.0 and OpenID Connect
- ✓ Creating & Configuring Azure AD Applications
- ✓ Acquiring Access Tokens using ADAL
- ➢ Programming with the Power BI Service SDK
- • Acquiring Access Tokens using MSAL

Power BI Service 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

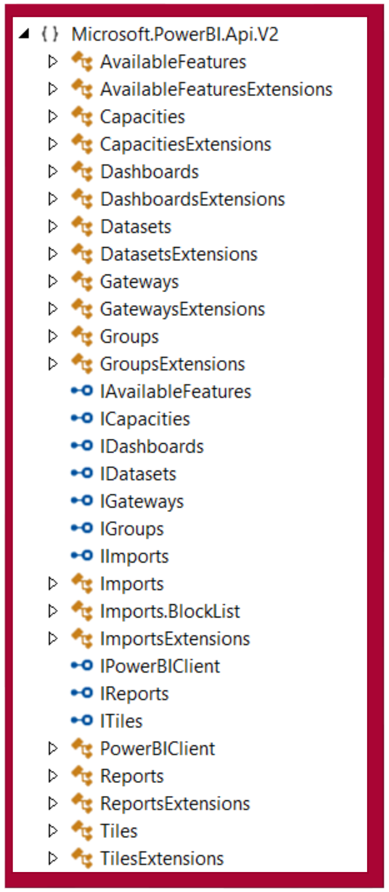
Microsoft.PowerBI.Api by Microsoft, 269K downloads v2.0.10
Client library for Power BI

Microsoft.PowerBI.JavaScript by Microsoft, 130K downloads v2.4.7
JavaScript web components for Power BI

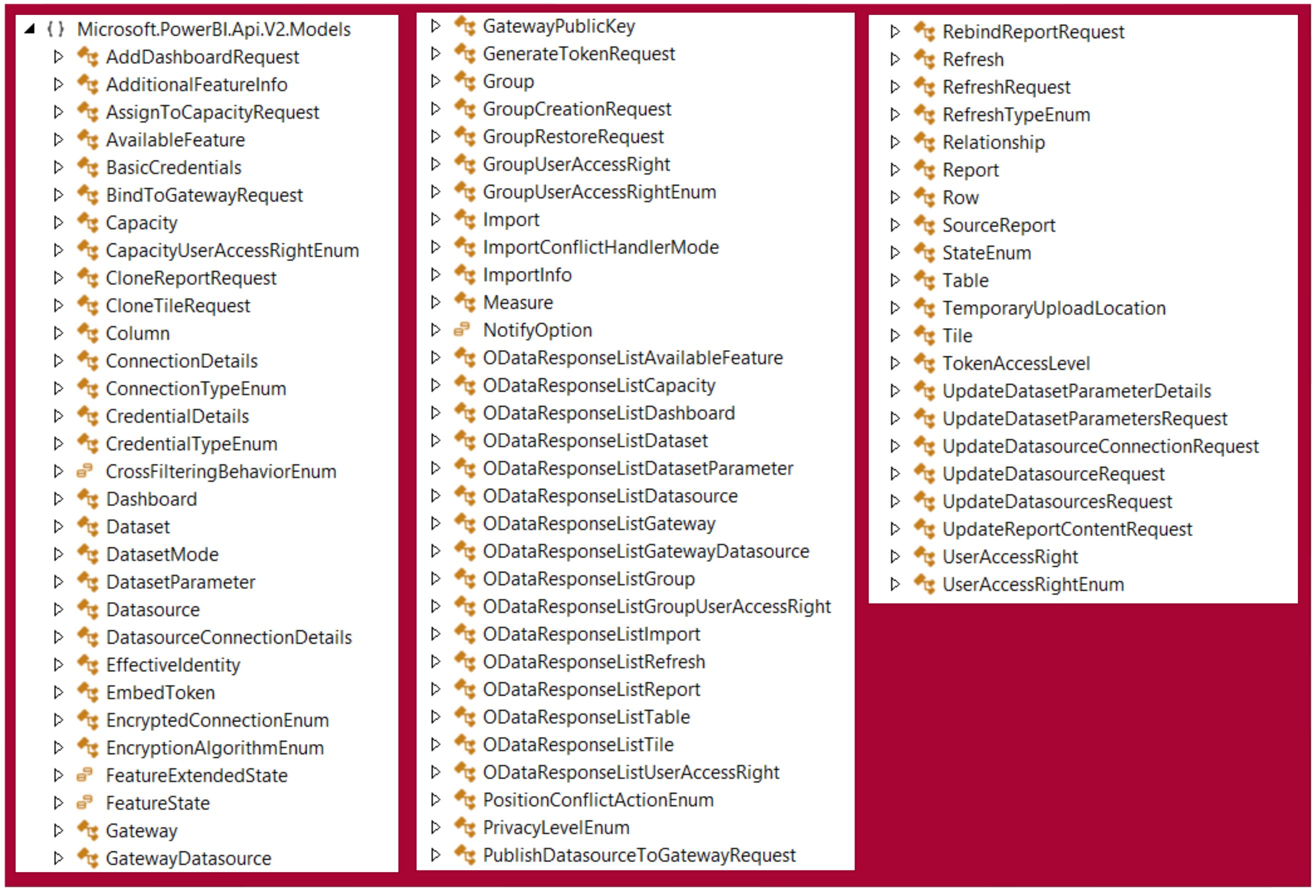
Microsoft.PowerBI.Api
Version: Latest stable 2.0.10
Install
Options

The Power BI SDK Classes

- SDK provides object model of classes



```
◢ { } Microsoft.PowerBI.Api.V2
    ▷ AvailableFeatures
    ▷ AvailableFeaturesExtensions
    ▷ Capacities
    ▷ CapacitiesExtensions
    ▷ Dashboards
    ▷ DashboardsExtensions
    ▷ Datasets
    ▷ DatasetsExtensions
    ▷ Gateways
    ▷ GatewaysExtensions
    ▷ Groups
    ▷ GroupsExtensions
      IAvailableFeatures
      ICapacities
      IDashboards
      IDatasets
      IGateways
      IGroups
      IImports
    ▷ Imports
    ▷ Imports.BlockList
    ▷ ImportsExtensions
      IPowerBIClient
      IReports
      ITiles
    ▷ PowerBIClient
    ▷ Reports
    ▷ ReportsExtensions
    ▷ Tiles
    ▷ TilesExtensions
```



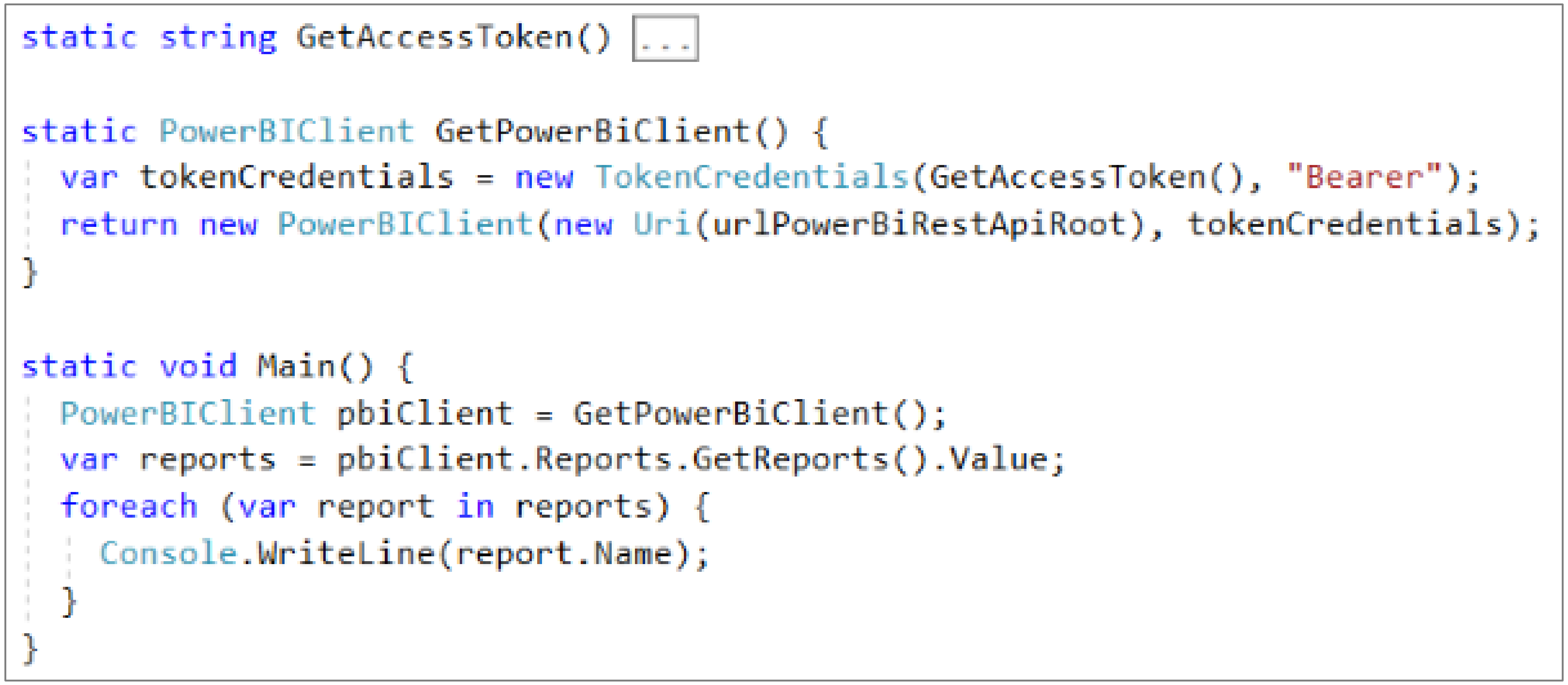
```
◢ { } Microsoft.PowerBI.Api.V2.Models
    ▷ AddDashboardRequest
    ▷ AdditionalFeatureInfo
    ▷ AssignToCapacityRequest
    ▷ AvailableFeature
    ▷ BasicCredentials
    ▷ BindToGatewayRequest
    ▷ Capacity
    ▷ CapacityUserAccessRightEnum
    ▷ CloneReportRequest
    ▷ CloneTileRequest
    ▷ Column
    ▷ ConnectionDetails
    ▷ ConnectionTypeEnum
    ▷ CredentialDetails
    ▷ CredentialTypeEnum
    ▷ CrossFilteringBehaviorEnum
    ▷ Dashboard
    ▷ Dataset
    ▷ DatasetMode
    ▷ DatasetParameter
    ▷ Datasource
    ▷ DatasourceConnectionDetails
    ▷ EffectiveIdentity
    ▷ EmbedToken
    ▷ EncryptedConnectionEnum
    ▷ EncryptionAlgorithmEnum
    ▷ FeatureExtendedState
    ▷ FeatureState
    ▷ Gateway
    ▷ GatewayDatasource
```

```
▷ GatewayPublicKey
▷ GenerateTokenRequest
▷ Group
▷ GroupCreationRequest
▷ GroupRestoreRequest
▷ GroupUserAccessRight
▷ GroupUserAccessRightEnum
▷ Import
▷ ImportConflictHandlerMode
▷ ImportInfo
▷ Measure
▷ NotifyOption
▷ ODataResponseListAvailableFeature
▷ ODataResponseListCapacity
▷ ODataResponseListDashboard
▷ ODataResponseListDataset
▷ ODataResponseListDatasetParameter
▷ ODataResponseListDatasource
▷ ODataResponseListGateway
▷ ODataResponseListGatewayDatasource
▷ ODataResponseListGroup
▷ ODataResponseListGroupUserAccessRight
▷ ODataResponseListImport
▷ ODataResponseListRefresh
▷ ODataResponseListReport
▷ ODataResponseListTable
▷ ODataResponseListTile
▷ ODataResponseListUserAccessRight
▷ PositionConflictActionEnum
▷ PrivacyLevelEnum
▷ PublishDatasourceToGatewayRequest
```

```
▷ RebindReportRequest
▷ Refresh
▷ RefreshRequest
▷ RefreshTypeEnum
▷ Relationship
▷ Report
▷ Row
▷ SourceReport
▷ StateEnum
▷ Table
▷ TemporaryUploadLocation
▷ Tile
▷ TokenAccessLevel
▷ UpdateDatasetParameterDetails
▷ UpdateDatasetParametersRequest
▷ UpdateDatasourceConnectionRequest
▷ UpdateDatasourceRequest
▷ UpdateDatasourcesRequest
▷ UpdateReportContentRequest
▷ UserAccessRight
▷ UserAccessRightEnum
```

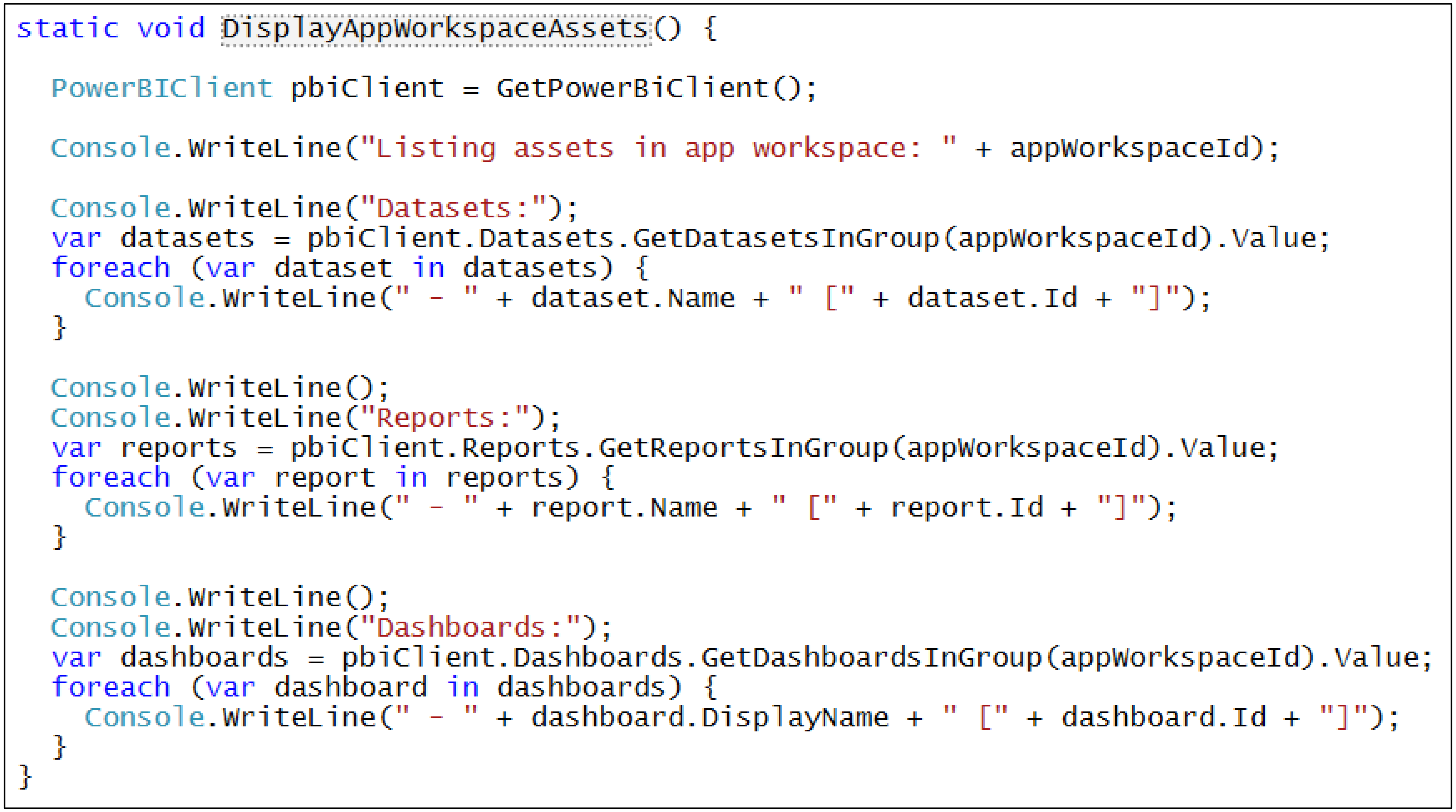
Initializing an Instance of PowerBIClient

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



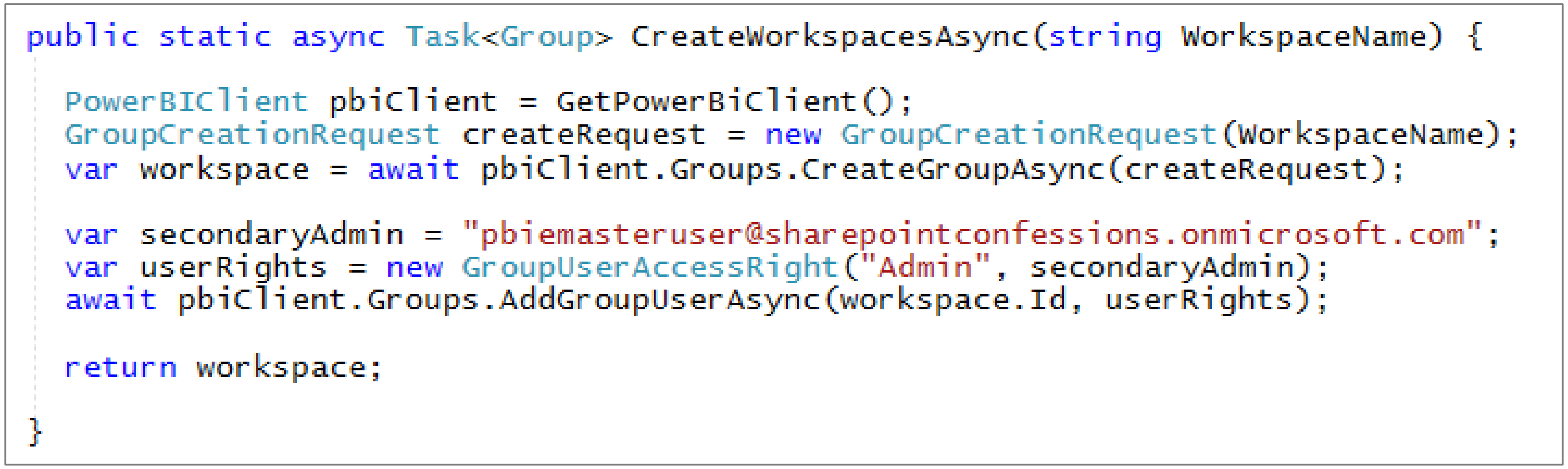
```
static string GetAccessToken() ...

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 + "]");
  }
}
```

Creating App Workspaces



```
public static async Task<Group> CreateWorkspacesAsync(string WorkspaceName) {

  PowerBIClient pbiClient = GetPowerBiClient();
  GroupCreationRequest createRequest = new GroupCreationRequest(WorkspaceName);
  var workspace = await pbiClient.Groups.CreateGroupAsync(createRequest);

  var secondaryAdmin = "pbiemasteruser@sharepointconfessions.onmicrosoft.com";
  var userRights = new GroupUserAccessRight("Admin", secondaryAdmin);
  await pbiClient.Groups.AddGroupUserAsync(workspace.Id, userRights);

  return workspace;

}
```

Importing a PBIX File

```
public static async Task UploadPBIX(string WorkspaceId, string pbixName, string importName, bool updateSqlCredentials = false) {

  string PbixFilePath = HttpContext.Current.Server.MapPath("/PBIX/" + pbixName);
  PowerBIClient pbiClient = GetPowerBiClient();
  FileStream stream = new FileStream(PbixFilePath, FileMode.Open, FileAccess.Read);
  var import = await pbiClient.Imports.PostImportWithFileAsyncInGroup(WorkspaceId, stream, importName);

  if (updateSqlCredentials) {
    await PatchSqlDatasourceCredentials(WorkspaceId, importName);
  }

  return;
}
```

Patching Datasource Credentials

```
public static async Task PatchSqlDatasourceCredentials(string WorkspaceId, string importName) {
  PowerBIClient pbiClient = GetPowerBiClient();
  var datasets = (await pbiClient.Datasets.GetDatasetsInGroupAsync(WorkspaceId)).Value;
  foreach (var dataset in datasets) {
    if (importName.Equals(dataset.Name)) {
      string datasetId = dataset.Id;
      var datasources = (await pbiClient.Datasets.GetDatasourcesInGroupAsync(WorkspaceId, datasetId)).Value;
      foreach (var datasource in datasources) {
        if (datasource.DatasourceType == "SQL") {
          var datasourceId = datasource.DatasourceId;
          var gatewayId = datasource.GatewayId;
          // create credentials for Azure SQL database log in
          Creds.BasicCredentials creds = new Creds.BasicCredentials("CptStudent", "pass@word1");
          CredentialDetails details = new CredentialDetails(creds);
          UpdateDatasourceRequest req = new UpdateDatasourceRequest(details);
          // Update credentials through gateway
          await pbiClient.Gateways.UpdateDatasourceAsync(gatewayId, datasourceId, details);
        }
      }
    }
  }
  return;
}
```

Exporting/Importing PBIX Files

```
var reports = pbiClient.Reports.GetReportsInGroup(sourceAppWorkspaceId).Value;

string downloadPath = @"C:\Student\downloads\";
// create download folder if it doesn't exist
if (!Directory.Exists(downloadPath)) {
  Directory.CreateDirectory(downloadPath);
}

foreach (var report in reports) {

  var reportStream = pbiClient.Reports.ExportReportInGroup(sourceAppWorkspaceId, report.Id);
  string filePath = downloadPath + report.Name + ".pbix";
  Console.WriteLine("Downloading PBIX file for " + report.Name + "to " + filePath);
  FileStream stream1 = new FileStream(filePath, FileMode.Create, FileAccess.ReadWrite);
  reportStream.CopyToAsync(stream1).Wait();
  reportStream.Close();
  stream1.Close();
  stream1.Dispose();

  FileStream stream = new FileStream(filePath, FileMode.Open, FileAccess.Read);
  Console.WriteLine("Publishing " + filePath + " to " + targetAppWorkpaceName);
  var import = pbiClient.Imports.PostImportWithFileInGroup(targetAppWorkspaceId, stream, report.Name);

  Console.WriteLine("Deleing file " + filePath);
  stream.Close();
  stream.Dispose();
  File.Delete(filePath);

  Console.WriteLine();
}

Console.WriteLine("Export/Import process completed");
```

Agenda

- ✓ Power BI Service API Overview
- ✓ Understanding OAuth 2.0 and OpenID Connect
- ✓ Creating & Configuring Azure AD Applications
- ✓ Acquiring Access Tokens using ADAL
- ✓ Programming with the Power BI Service SDK
- ➢ Acquiring Access Tokens using MSAL

DEMO

Authenticating with MSAL

Summary

- ✓ Power BI Service API Overview
- ✓ Understanding OAuth 2.0 and OpenID Connect
- ✓ Creating & Configuring Azure AD Applications
- ✓ Acquiring Access Tokens using ADAL
- ✓ Programming with the Power BI Service SDK
- ✓ Acquiring Access Tokens using MSAL