

Developing with the Power BI Service API



Agenda

- Power BI Service API Overview
- Authentication with Azure Active Directory
- Developing with the Power BI SDK
- Creating and Managing Workspaces



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



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



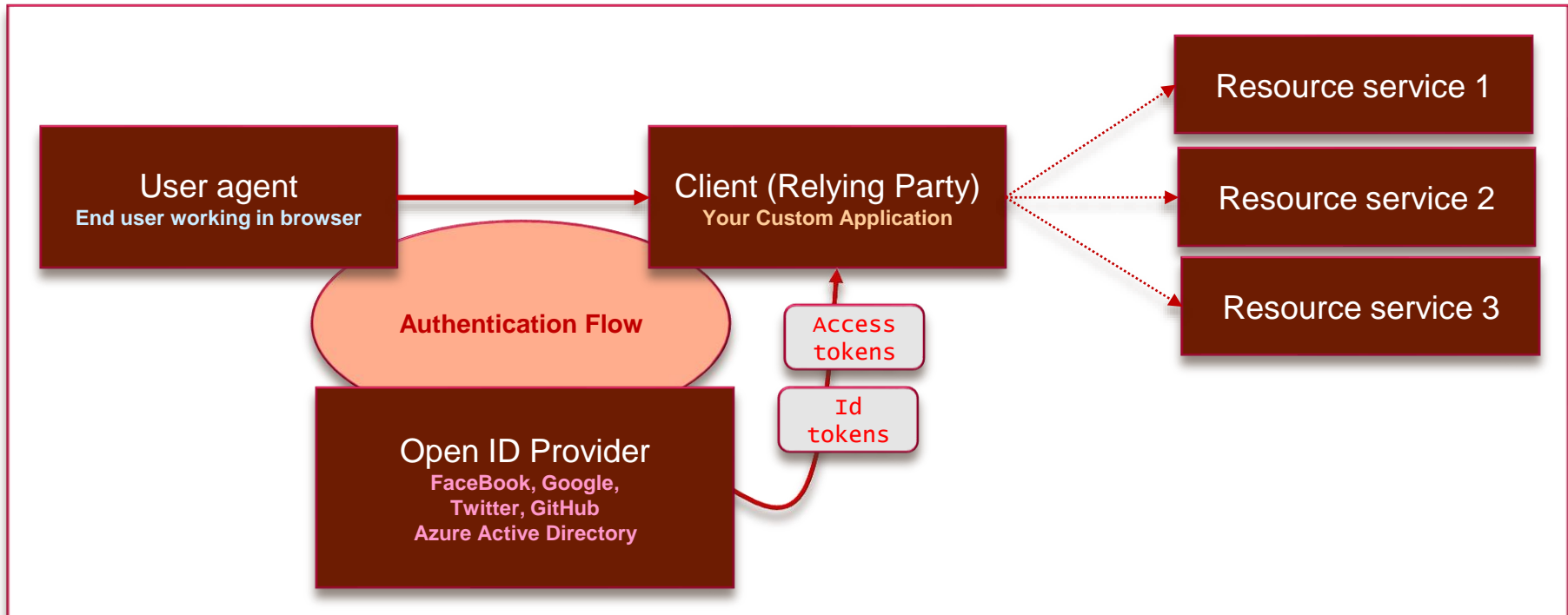
Agenda

- ✓ Power BI Service API Overview
- Authentication with Azure Active Directory
 - Developing with the Power BI SDK
 - Creating and Managing Workspaces



OAuth2 and Open ID Connect

- Power BI Service requires authentication with OAuth2
 - Your application must implement an authentication flow
 - Authentication flow used to acquire an access token
 - Access token required whenever calling Power BI Service API



Client Application Registration

- Application 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



Authentication Flows

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



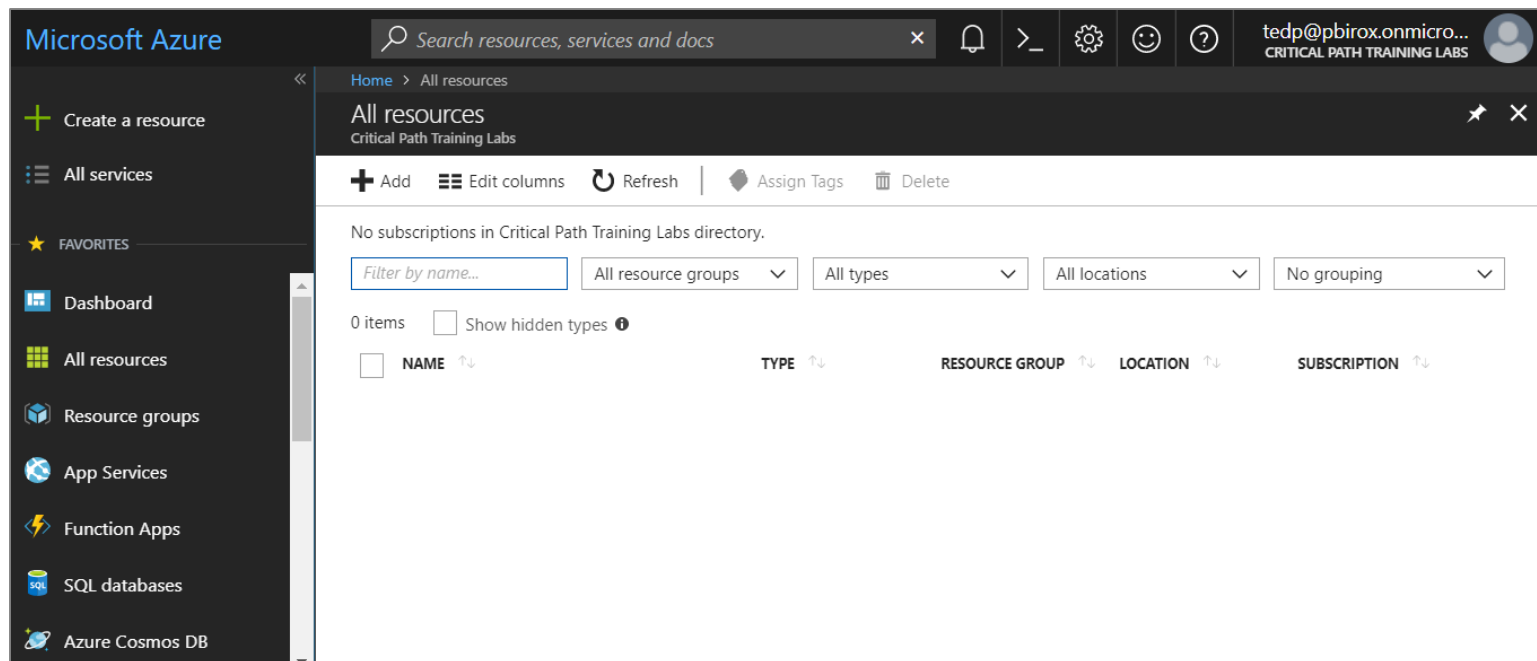
Azure Active Directory (AAD)

- AAD plays role of an OpenID Connect Provider
 - Creates access tokens based on OAuth 2.0
 - Creates id tokens based on OpenID Connect 1.0
- AAD provides authentication & authorization for...
 - Office 365, Dynamics 365 and SharePoint Online
 - Power BI Service API and Microsoft Graph API
 - Custom Web Applications and Web Services



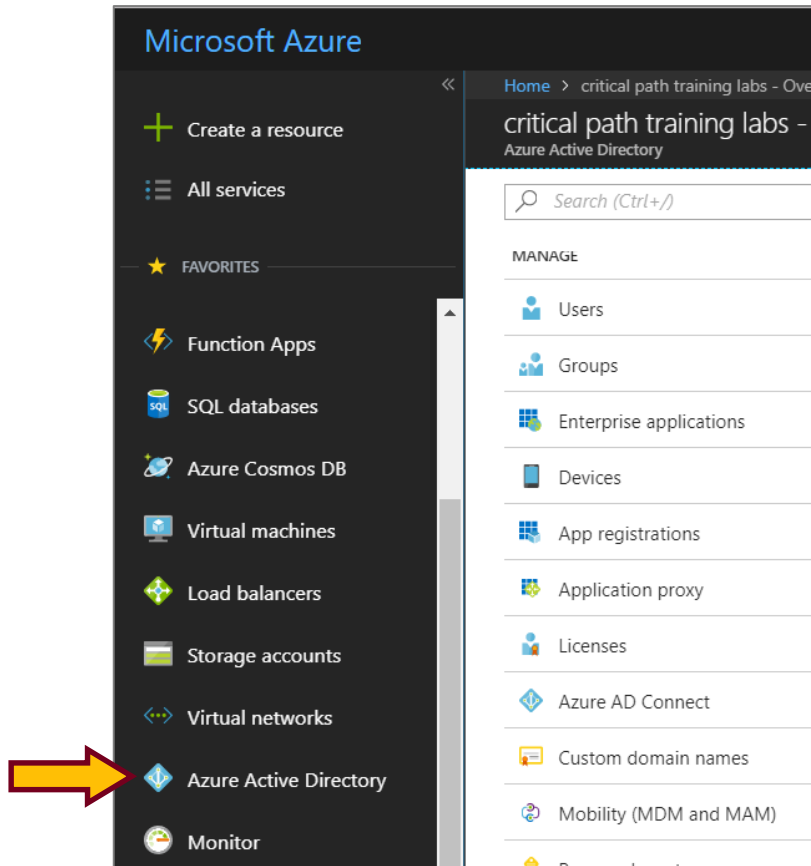
The Azure Portal

- Azure portal allows to create application
 - Azure Portal accessible at <https://portal.azure.com>
 - Azure subscription required to create resources (e.g. Web Apps, VMs)
 - No Azure subscription required to manage users, groups and applications



Azure Active Directory

- Azure portal access to Access Azure Active Directory
 - Provides ability to configure users, groups and application



Managing Users and Groups

- You can manage users and assign licenses
- You can create groups and populate members

Home > critical path training labs > Users - All users

Users - All users
critical path training labs - Azure Active Directory

Search (Ctrl+/)

+ New user + New guest user Reset password Delete user Multi-Factor Authentication

Name
Search by name or email

Show
All users

NAME	USER NAME	USER TYPE
AP Austin Powers	AustinP@pbiox.onMicrosoft.com	Member
CM Carrie Mathison	CarrieM@pbiox.onMicrosoft.com	Member
EP Emma Peel	EmmaP@pbiox.onMicrosoft.com	Member
JB Jack Bauer	JackB@pbiox.onMicrosoft.com	Member
JR Jack Ryan	JackR@pbiox.onMicrosoft.com	Member
JB James Bond	JamesB@pbiox.onMicrosoft.com	Member
JB Jason Bourne	JasonB@pbiox.onMicrosoft.com	Member
MS Maxwell Smart	MaxwellS@pbiox.onMicrosoft.com	Member
TP Ted Pattison	tedp@pbiox.onmicrosoft.com	Member

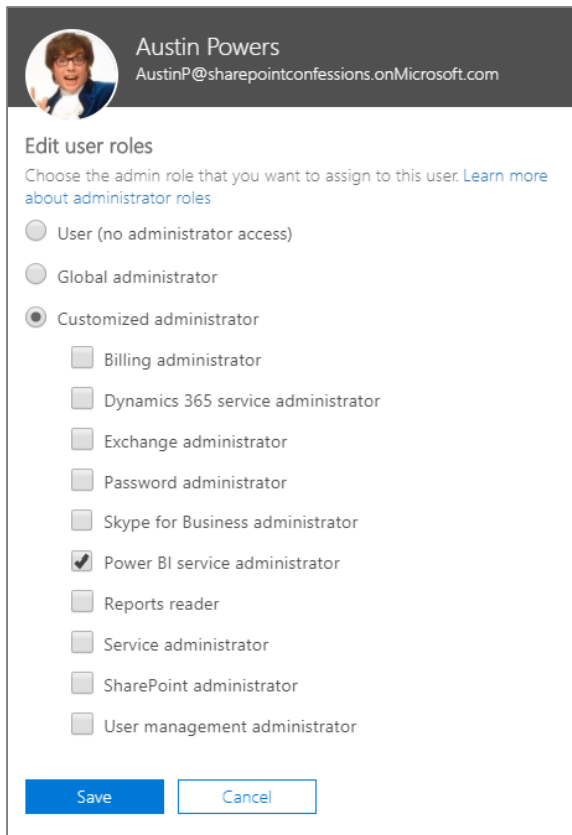
Left sidebar navigation:

- All users
- Deleted users
- Password reset
- User settings
- ACTIVITY
 - Sign-ins
 - Audit logs
- TROUBLESHOOTING + SUPPORT
 - Troubleshoot
 - New support request



Power BI Administrator

- Azure AD provides role of Power BI Service administrator
 - Provides user with tenant-level administrative permissions



The screenshot shows the 'Edit user roles' dialog for a user named Austin Powers. The user's profile picture and email address (AustinP@sharepointconfessions.onmicrosoft.com) are displayed at the top. Below the title 'Edit user roles', there is a instruction: 'Choose the admin role that you want to assign to this user. [Learn more about administrator roles](#)'. The dialog lists several roles with checkboxes. The 'Customized administrator' role is selected with a radio button. Under this role, the 'Power BI service administrator' checkbox is checked with a checkmark. Other roles listed include 'User (no administrator access)', 'Global administrator', 'Billing administrator', 'Dynamics 365 service administrator', 'Exchange administrator', 'Password administrator', 'Skype for Business administrator', 'Reports reader', 'Service administrator', 'SharePoint administrator', and 'User management administrator'. At the bottom, there are 'Save' and 'Cancel' buttons.

Austin Powers
AustinP@sharepointconfessions.onmicrosoft.com

Edit user roles
Choose the admin role that you want to assign to this user. [Learn more about administrator roles](#)

☐ User (no administrator access)

☐ Global administrator

☒ Customized administrator

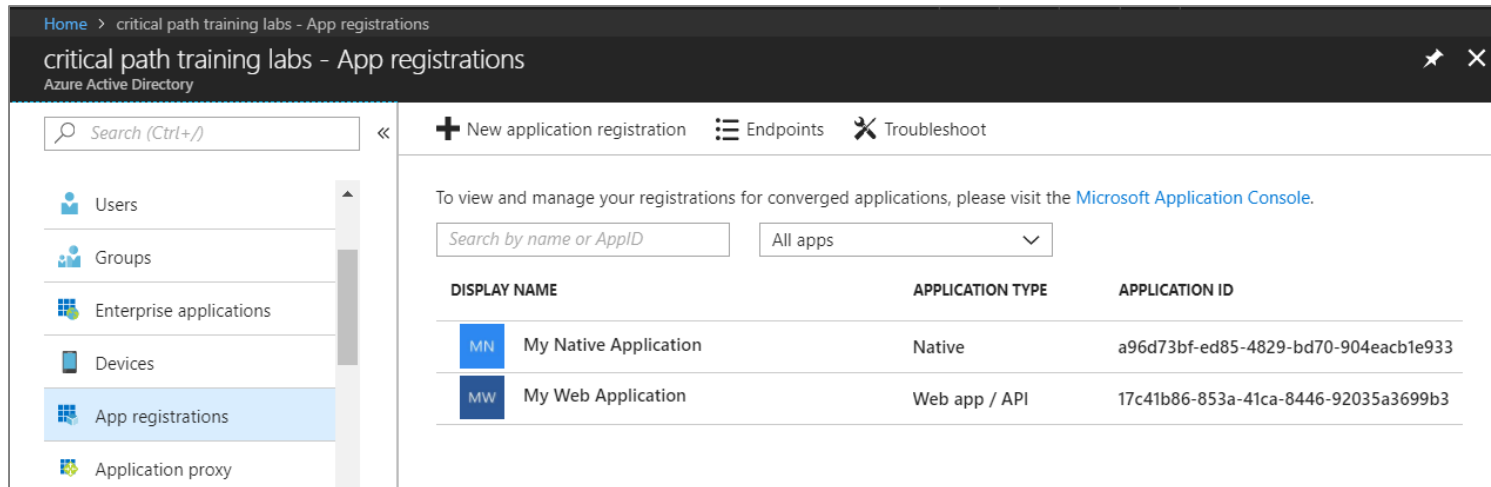
- ☐ Billing administrator
- ☐ Dynamics 365 service administrator
- ☐ Exchange administrator
- ☐ Password administrator
- ☐ Skype for Business administrator
- ☒ Power BI service administrator
- ☐ Reports reader
- ☐ Service administrator
- ☐ SharePoint administrator
- ☐ User management administrator

[Save](#) [Cancel](#)



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**



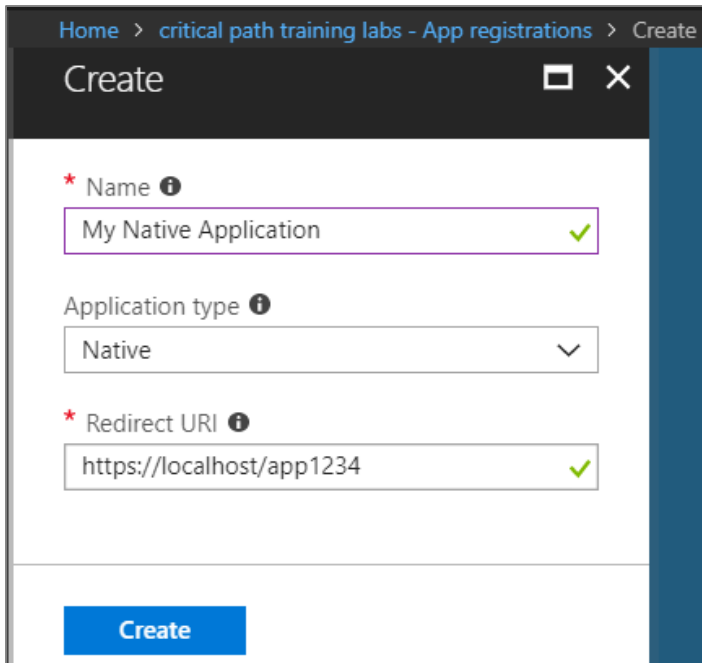
The screenshot shows the Azure Active Directory (Azure AD) App Registrations page. The left sidebar contains navigation links: Users, Groups, Enterprise applications, Devices, App registrations (selected), and Application proxy. The main content area displays a table of registered applications. The table has three columns: DISPLAY NAME, APPLICATION TYPE, and APPLICATION ID. Two applications are listed: 'My Native Application' (Native) and 'My Web Application' (Web app / API).

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



Creating a Native Application

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



The screenshot shows a 'Create' dialog box with a dark header bar containing the text 'Create' and window control icons. The breadcrumb path at the top reads 'Home > critical path training labs - App registrations > Create'. The form contains three fields, each with a red asterisk and an information icon:

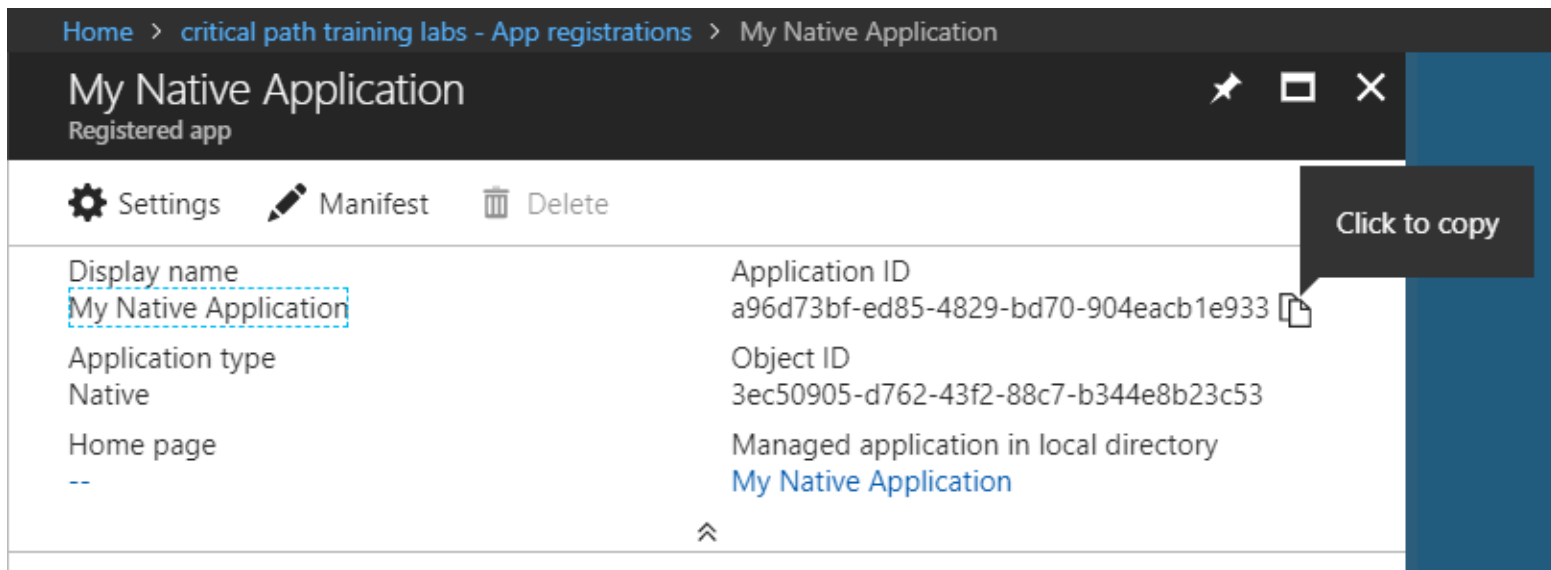
- Name**: The text 'My Native Application' is entered in the input field, followed by a green checkmark.
- Application type**: A dropdown menu is set to 'Native' with a downward arrow.
- Redirect URI**: The text 'https://localhost/app1234' is entered in the input field, followed by a green checkmark.

A blue 'Create' button is located at the bottom left of the dialog.



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 IS from the azure portal



The screenshot shows the Azure portal interface for a registered application named 'My Native Application'. The breadcrumb navigation at the top reads 'Home > critical path training labs - App registrations > My Native Application'. The application title 'My Native Application' is displayed with the subtitle 'Registered app'. Below the title are three action buttons: 'Settings' (gear icon), 'Manifest' (pencil icon), and 'Delete' (trash icon). The application details are presented in a table-like format:

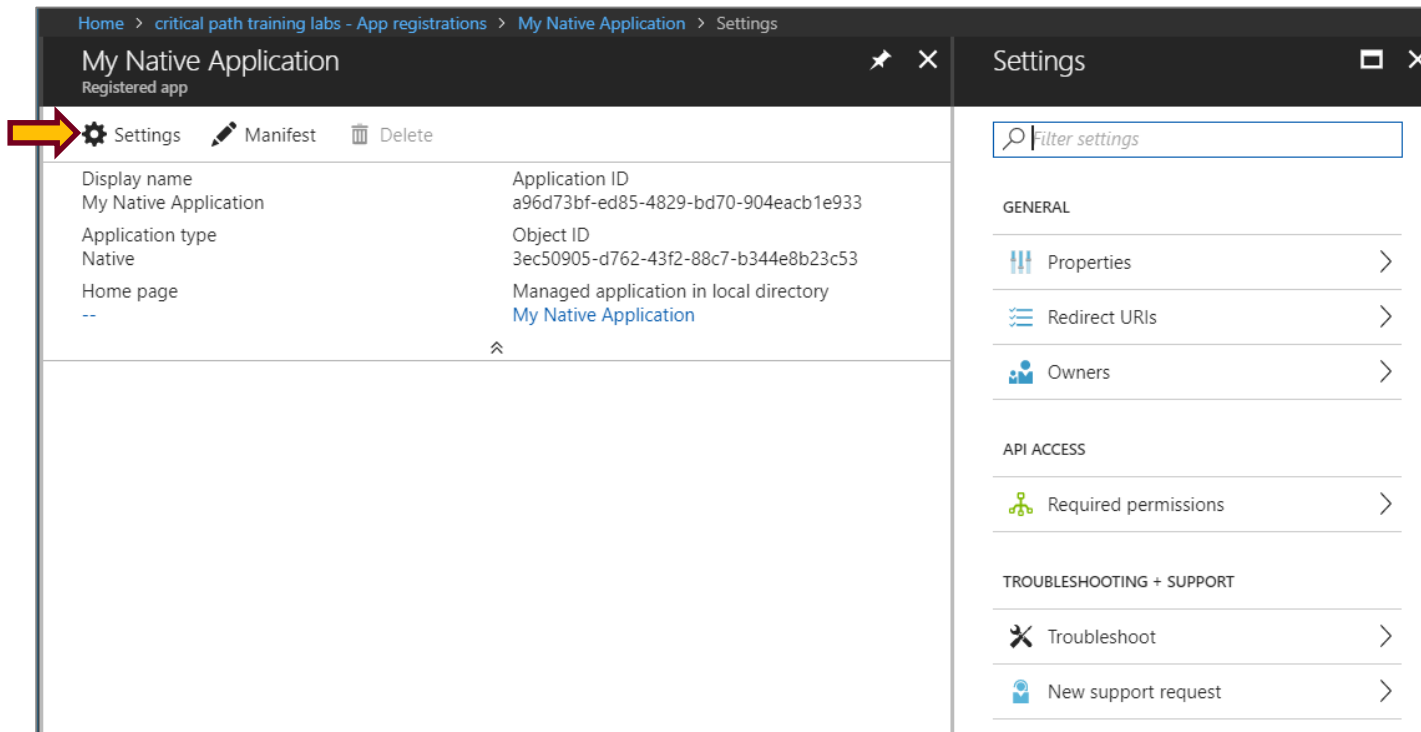
Display name	Application ID
My Native Application	a96d73bf-ed85-4829-bd70-904eacb1e933
Application type	Object ID
Native	3ec50905-d762-43f2-88c7-b344e8b23c53
Home page	Managed application in local directory
--	My Native Application

A dashed blue box highlights the 'My Native Application' text under 'Display name'. A dark grey tooltip with the text 'Click to copy' is positioned over the 'Application ID' value 'a96d73bf-ed85-4829-bd70-904eacb1e933'. A small copy icon is visible to the right of the ID value. An upward-pointing chevron icon is located at the bottom center of the details section.



Native Application Settings

- Properties
- Redirect URLs
- Owners
- Required Permissions



The screenshot displays the Azure portal interface for managing a native application. The breadcrumb navigation at the top reads: Home > critical path training labs - App registrations > My Native Application > Settings. The left-hand pane, titled 'My Native Application' (Registered app), contains three tabs: 'Settings' (highlighted with a red arrow), 'Manifest', and 'Delete'. Below the tabs, a table lists application details:

Display name	My Native Application	Application ID	a96d73bf-ed85-4829-bd70-904eacb1e933
Application type	Native	Object ID	3ec50905-d762-43f2-88c7-b344e8b23c53
Home page	--	Managed application in local directory	My Native Application

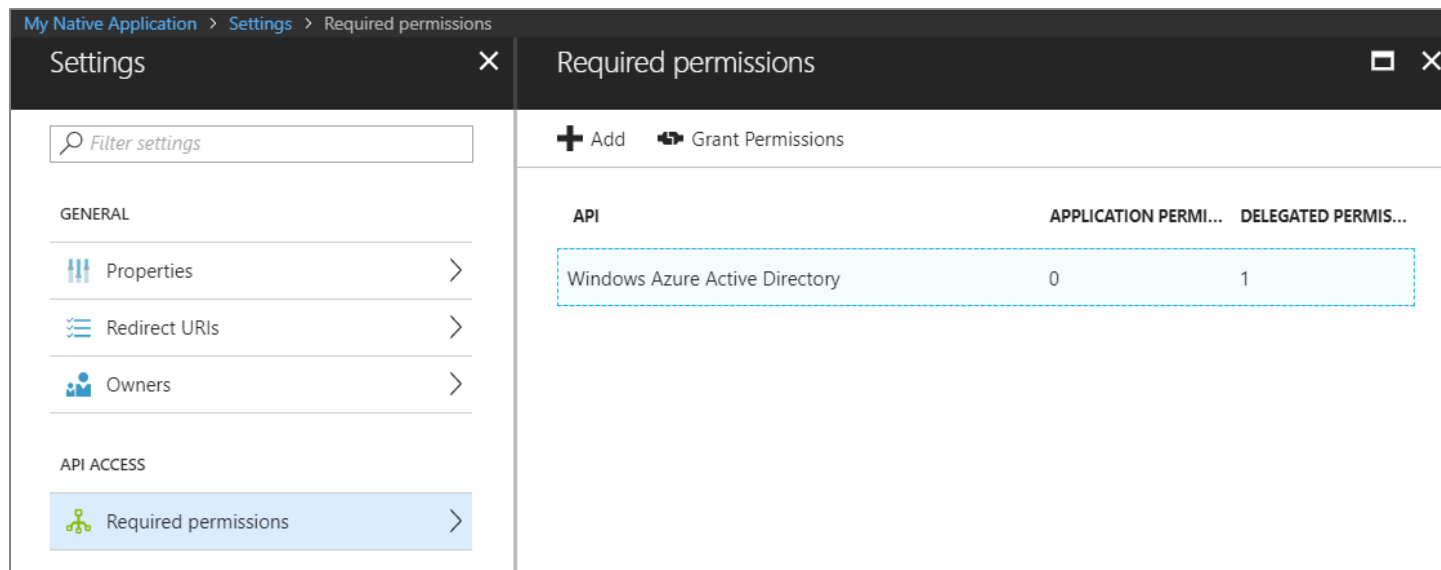
The right-hand pane, titled 'Settings', features a search bar labeled 'Filter settings'. It is organized into sections with expandable options:

- GENERAL**
 - Properties
 - Redirect URLs
 - Owners
- API ACCESS**
 - Required permissions
- TROUBLESHOOTING + SUPPORT**
 - Troubleshoot
 - New support request



Configuring Required Permissions

- Application configured with permissions
 - Default permissions allows user authentication – but that's it
 - To use APIs, you must assign permissions to the application



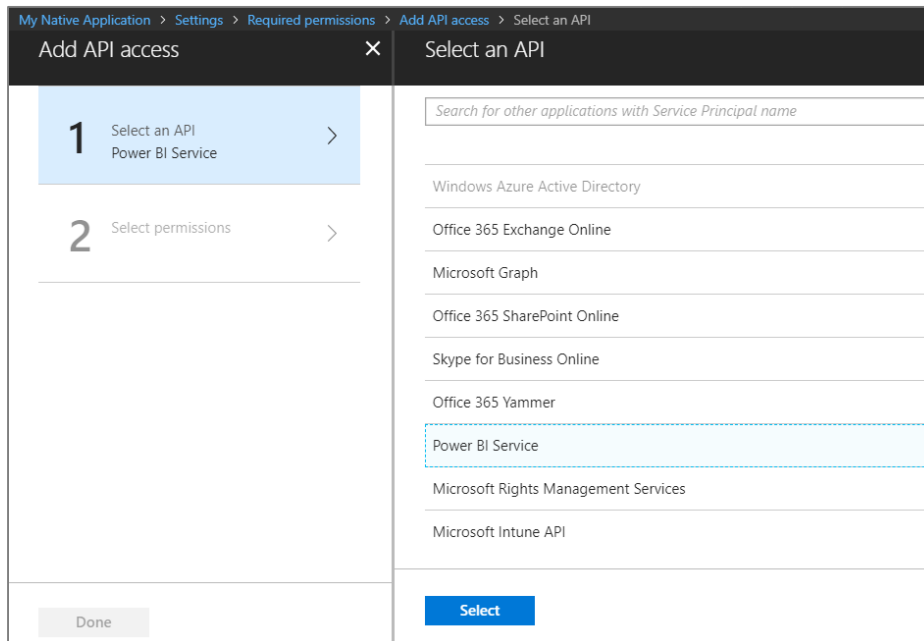
The screenshot shows the 'Required permissions' settings window for a native application. The left sidebar contains a search bar and a list of settings categories: GENERAL (Properties, Redirect URIs, Owners) and API ACCESS (Required permissions). The main panel displays a table of required permissions.

API	APPLICATION PERMI...	DELEGATED PERMIS...
Windows Azure Active Directory	0	1



Choosing APIs

- There are lots of APIs to choose from
 - Office 365 Exchange Online
 - Microsoft Graph
 - Office 365 SharePoint Online
 - Power BI Service



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
 - Note that some delegated permissions requires administrative permissions

<input type="checkbox"/> DELEGATED PERMISSIONS	REQUIRES ADMIN
Run search queries as a user	✓ Yes
Read user profiles	✓ Yes
<input checked="" type="checkbox"/> Read user files	✗ No
Read managed metadata	✓ Yes
<input checked="" type="checkbox"/> Read items in all site collections	✗ No
Read and write user profiles	✓ Yes
<input checked="" type="checkbox"/> Read and write user files	✗ No
Read and write managed metadata	✓ Yes
<input checked="" type="checkbox"/> Read and write items in all site collections	✗ No
<input checked="" type="checkbox"/> 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



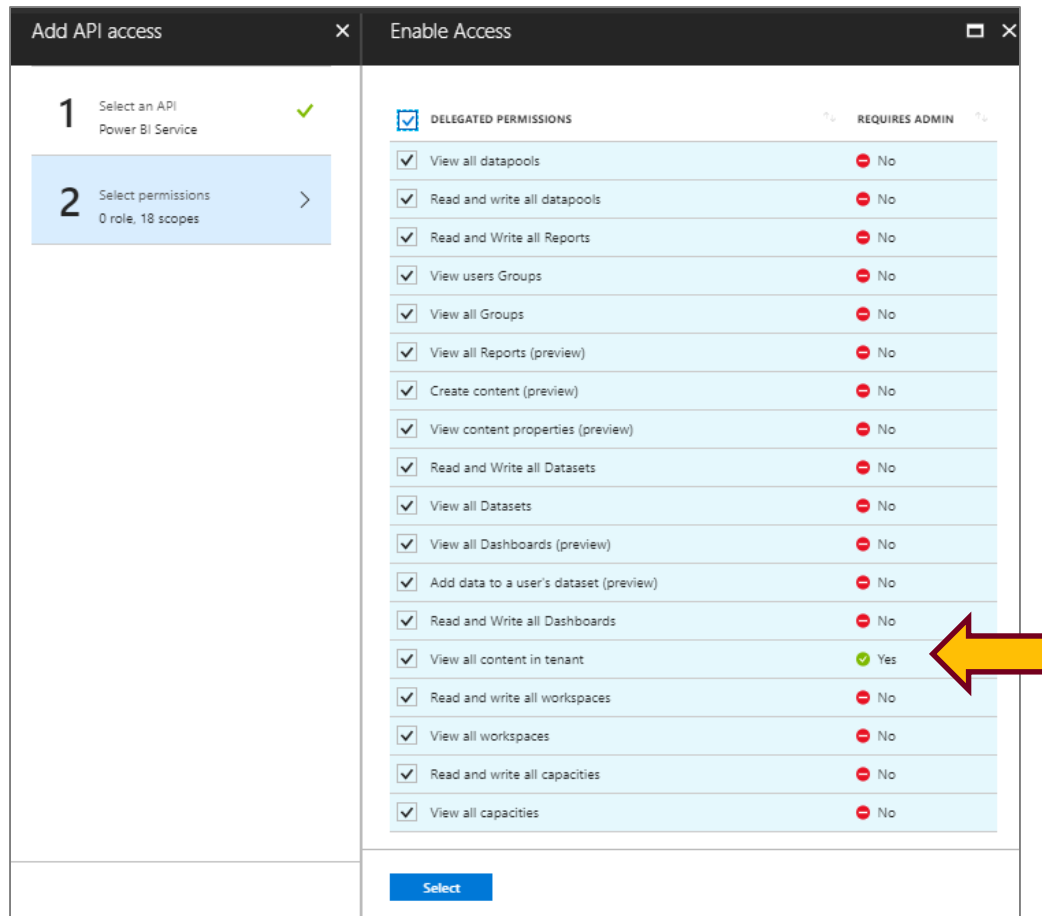
Power BI Service API Permissions

- 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 a user's dataset (preview)
- Read and Write all Dashboards
- View all content in tenant (requires admin)
- Read and write all workspaces
- View all workspaces
- Read and write all capacities
- View all capacities



Power BI Service API Permissions

- Do you really need permissions that Requires Admin
 - It makes it so that only Power BI administrators can use your app



The screenshot shows the 'Add API access' dialog with the 'Enable Access' tab selected. The left sidebar shows two steps: '1 Select an API' (Power BI Service) and '2 Select permissions' (0 role, 18 scopes). The main area displays a table of permissions under the heading 'DELEGATED PERMISSIONS'. A yellow arrow points to the 'View all content in tenant' permission, which is marked as 'Yes' under the 'REQUIRES ADMIN' column.

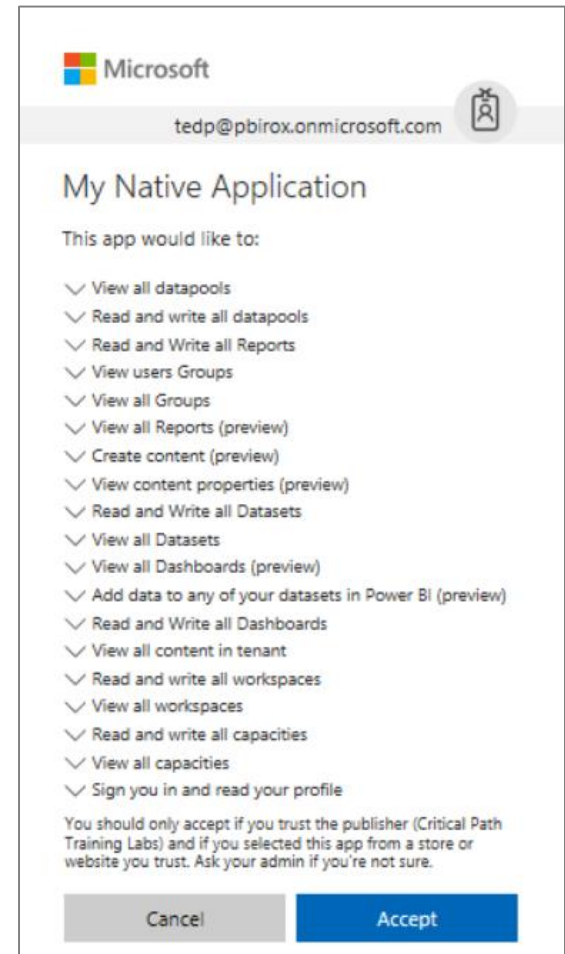
DELEGATED PERMISSIONS	REQUIRES ADMIN
<input checked="" type="checkbox"/> View all datapools	No
<input checked="" type="checkbox"/> Read and write all datapools	No
<input checked="" type="checkbox"/> Read and Write all Reports	No
<input checked="" type="checkbox"/> View users Groups	No
<input checked="" type="checkbox"/> View all Groups	No
<input checked="" type="checkbox"/> View all Reports (preview)	No
<input checked="" type="checkbox"/> Create content (preview)	No
<input checked="" type="checkbox"/> View content properties (preview)	No
<input checked="" type="checkbox"/> Read and Write all Datasets	No
<input checked="" type="checkbox"/> View all Datasets	No
<input checked="" type="checkbox"/> View all Dashboards (preview)	No
<input checked="" type="checkbox"/> Add data to a user's dataset (preview)	No
<input checked="" type="checkbox"/> Read and Write all Dashboards	No
<input checked="" type="checkbox"/> View all content in tenant	Yes
<input checked="" type="checkbox"/> Read and write all workspaces	No
<input checked="" type="checkbox"/> View all workspaces	No
<input checked="" type="checkbox"/> Read and write all capacities	No
<input checked="" type="checkbox"/> View all capacities	No

Select



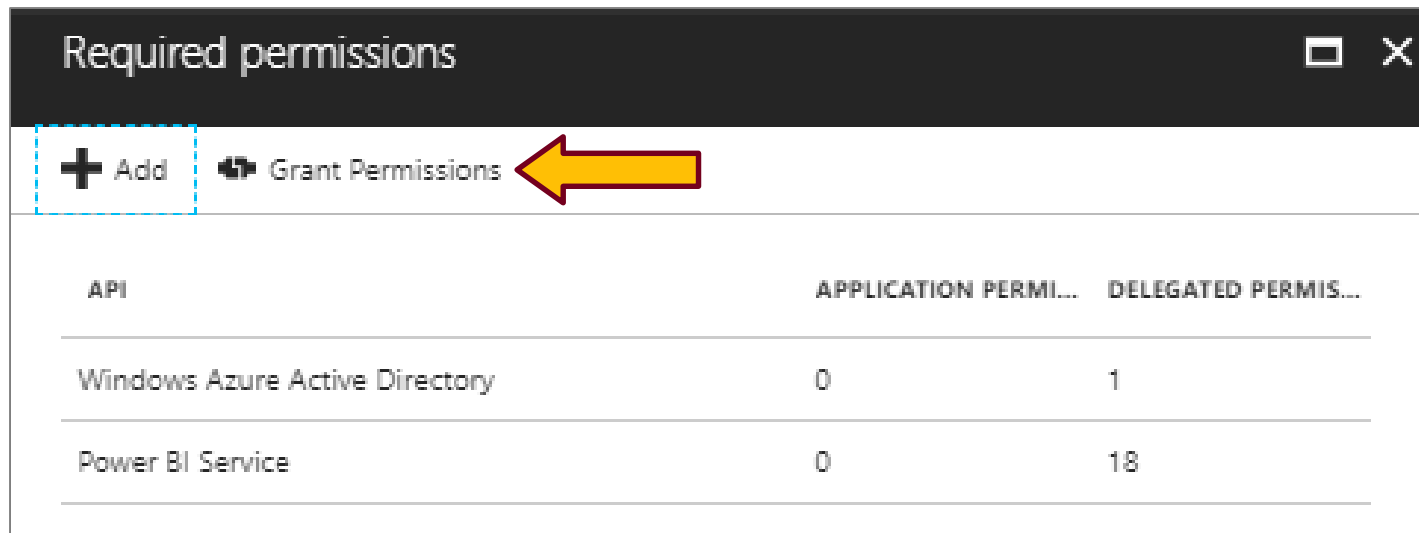
Interactive Consent for Delegated Permissions

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



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



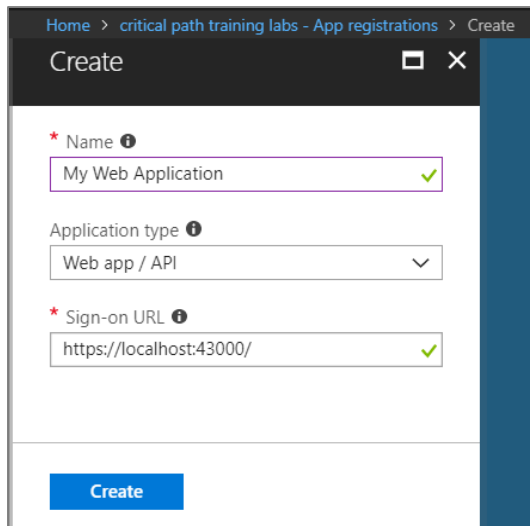
The screenshot shows a 'Required permissions' dialog box. At the top, there are two buttons: '+ Add' and 'Grant Permissions'. A yellow arrow points to the 'Grant Permissions' button. Below the buttons is a table with three columns: 'API', 'APPLICATION PERMI...', and 'DELEGATED PERMIS...'. The table lists two APIs: 'Windows Azure Active Directory' and 'Power BI Service'.

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



Creating Applications for Web Applications

- Web applications more secure than native applications
 - Requires Redirect URI which improves security
 - Authentication can be used on client secret (application password)
 - Can use application permissions – Native applications cannot



Home > critical path training labs - App registrations > Create

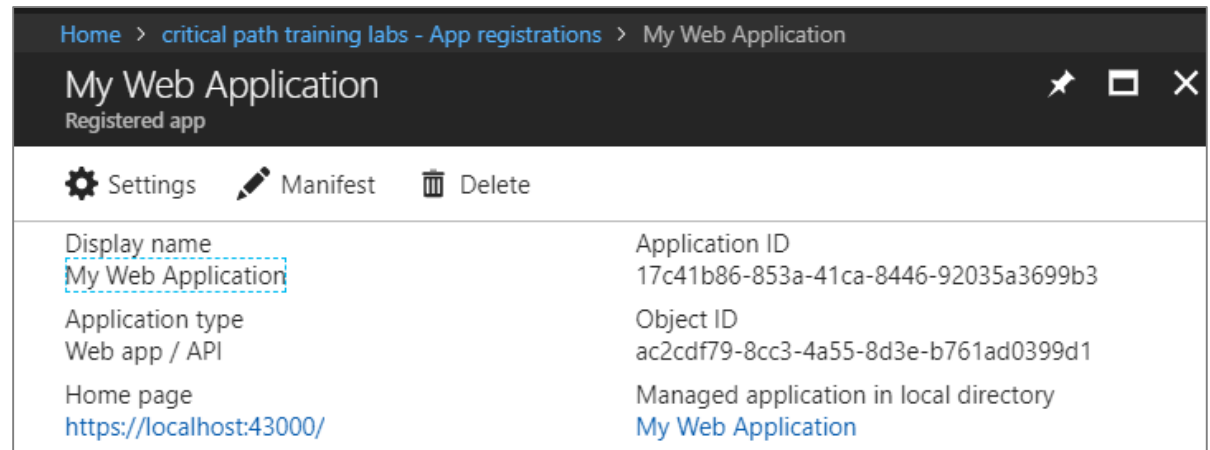
Create

* Name ⓘ
My Web Application ✓

Application type ⓘ
Web app / API ▾

* Sign-on URL ⓘ
https://localhost:43000/ ✓

Create



Home > critical path training labs - App registrations > My Web Application

My Web Application
Registered app






⚙ Settings ✎ Manifest 🗑 Delete

Display name	Application ID
My Web Application	17c41b86-853a-41ca-8446-92035a3699b3
Application type	Object ID
Web app / API	ac2cdf79-8cc3-4a55-8d3e-b761ad0399d1
Home page	Managed application in local directory
https://localhost:43000/	My Web Application



Reply URLs

- Reply URLs required for web applications
 - Your application must be accessible through the reply URL
 - Provides extra security dimension not available to native apps
 - Application can be configured with multiple reply URLs for single
 - Application must pass Reply URL matching registered Reply URL

Settings	×	Reply URLs
<input type="text" value="Filter settings"/>		 Save  Discard
GENERAL		<input type="text" value="https://localhost:43000/"/>
 Properties >		<input type="text" value="https://MyProductionApp.Azurewebsites.net"/>
 Reply URLs >		<input type="text"/>
 Owners >		



Creating Keys for Application Passwords

- Web applications authenticate using keys
 - Key acts as application-level password
 - Application requires copy of key value

Settings

×

GENERAL

Properties >

Reply URLs >

Owners >

API ACCESS

Required permissions >

Keys >

Keys

Save Discard Upload Public Key

⚠ Copy the key value. You won't be able to retrieve after you leave this blade.

Passwords

DESCRIPTION	EXPIRES	VALUE
My App Password	4/14/2020	O4p6IP2hi5PnB3k92xD9z/qWqCZBz6iMDRDkpx+VWos=
<input type="text" value="Key description"/>	<input type="text" value="Duration"/> ▼	<input type="text" value="Value will be displayed on save"/>



Power BI App Registration

- Power BI provides page to create Azure AD application
 - Accessible through <https://app.powerbi.com/apps>
 - Does not give the same level of control as Azure AD
 - RECOMMENDATION: use Azure portal instead of this page

Power BI Development Co. X

← → ↻ <https://app.powerbi.com/apps>

Power BI for Developers

Register an Application for Power BI

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

Step 1 Login to your Power BI account
Welcome, Ted Pattison! (Wrong account? No problem, logout and try again.)

Step 2 Tell us about your app
Let's start with some basic details.

App Name
Power BI Rest API Demo Console App

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

Redirect URL
A valid URL.
<https://localhost/PowerBI/RestApi/Demo>

Step 3 Choose APIs to access
Select the APIs and the level of access your app needs.

Dataset APIs

☐ Read All Datasets
☒ Read and Write All Datasets

Report and Dashboard APIs

☐ Read All Dashboards (preview)
☐ Read All Reports (preview)

Other APIs

☐ Read All Groups

Step 4 Register your app
Once you've set everything the way you want it, click the button below and we'll register your app. Your client ID and secret (for web apps only) will appear below. Be sure to copy the values into your app. By clicking the Register App button, you have accepted the [terms of use](#).

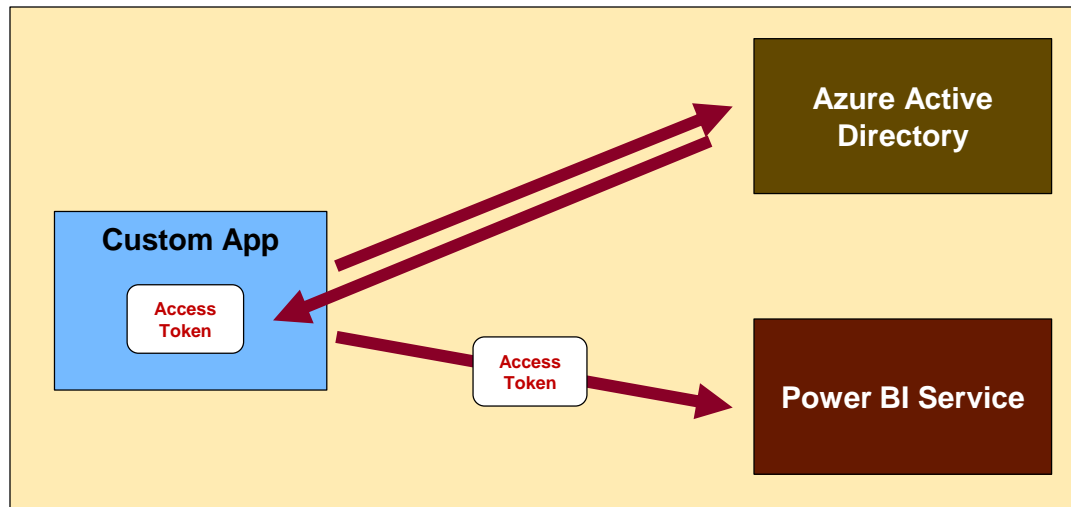
Register App

Client ID
58883475-5508-438f-80d7-9e16a573d70f



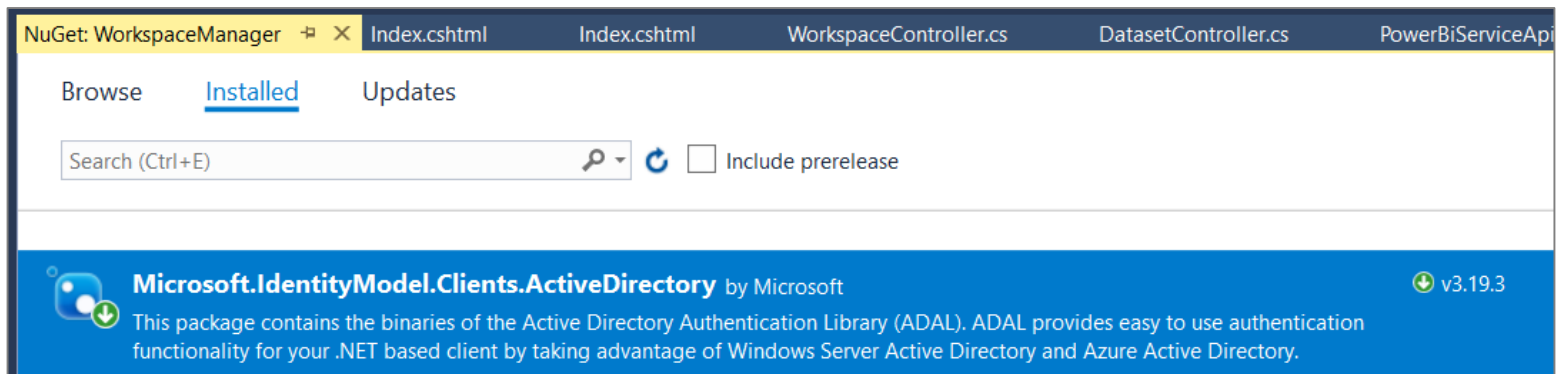
Authenticating with Azure AD

- 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



ADAL for .NET

- Active Directory Authentication Library for .NET
 - Used in Native Clients and in Web Clients
 - Handles authentication flow behind the scenes
 - Provides caching for access tokens and refresh tokens
- ADAL .NET installs as a NuGet Package
 - Package name is `Microsoft.IdentityModel.Clients.ActiveDirectory`



Access Token Acquisition

```
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;
}
```

Agenda

- ✓ Power BI Service API Overview
- ✓ Authentication with Azure Active Directory
- Developing with the Power BI SDK
 - Creating and Managing Workspaces



Programming with PowerBIClient

- PowerBIClient is top-level object in API

```
private static PowerBIClient GetPowerBiClient() {  
    string accessToken = GetAccessTokenAsync().Result;  
    TokenCredentials tokenCredentials = new TokenCredentials(accessToken, "Bearer");  
    return new PowerBIClient(new Uri(urlPowerBiRestApiRoot), tokenCredentials);  
}
```

```
public static async Task<IList<Group>> GetWorkspacesAsync() {  
    PowerBIClient pbiClient = GetPowerBiClient();  
    return (await pbiClient.Groups.GetGroupsAsync()).Value;  
}
```



Uploading 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;  
}
```



Getting and Refreshing Datasets

```
public static async Task<DatasetViewModel> GetDatasetAsync(string WorkspaceId, string DatasetId) {  
    PowerBIClient pbiClient = GetPowerBiClient();  
    Dataset dataset = (await pbiClient.Datasets.GetDatasetByIdInGroupAsync(WorkspaceId, DatasetId));  
    IList<DataSource> datasources = (await pbiClient.Datasets.GetDataSourcesInGroupAsync(WorkspaceId, DatasetId)).Value;  
    IList<Refresh> refreshHistory = null;  
  
    if (dataset.IsRefreshable == true) {  
        refreshHistory = (await pbiClient.Datasets.GetRefreshHistoryInGroupAsync(WorkspaceId, DatasetId)).Value;  
    }  
  
    DatasetViewModel viewModel = new DatasetViewModel {  
        WorkspaceId=WorkspaceId,  
        Id = dataset.Id,  
        Name = dataset.Name,  
        Dataset = dataset,  
        Datasources = datasources,  
        RefreshHistory = refreshHistory  
    };  
  
    return viewModel;  
}
```

```
public static async Task RefreshDatasetAsync(string WorkspaceId, string DatasetId) {  
    PowerBIClient pbiClient = GetPowerBiClient();  
    await pbiClient.Datasets.RefreshDatasetInGroupAsync(WorkspaceId, DatasetId);  
    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;
}
```



Agenda

- ✓ Power BI Service API Overview
- ✓ Authentication with Azure Active Directory
- ✓ Calling the API with Direct REST Calls
- ✓ Developing with the Power BI SDK
- Creating and Managing Workspaces



App Workspace Management

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



Summary

- ✓ Power BI Service API Overview
- ✓ Authentication with Azure Active Directory
- ✓ Developing with the Power BI SDK
- ✓ Creating and Managing Workspaces

