# The AppOwnsDataMultiTenant Application

The **AppOwnsDataMultiTenant** application is a sample .NET 6 application which demonstrates how to integrate the **service principal profiles** feature recently introduced in the Power BI Service. This article also explains how service principal profiles can be used to design and implement a large-scale Power BI embedding environment with 1000's or 10,000's of customer tenants.

Let's start by explaining what is meant by a tenant. If you have experience working with Azure AD, the word **"tenant"** might make you think of an Azure AD tenant. However, the concept of a tenant is different when talking about building a multi-tenant environment for Power BI embedding. In this context, each tenant is created on behalf of a customer for which you are embedding Power BI reports using the App-Owns-Data embedding model.

In order to manage a multi-tenant environment, you must create a separate tenant for each customer. Provisioning a new customer tenant for Power BI embedding typically involves writing code which calls the Power BI REST API to create a new Power BI workspace, import a PBIX file, update datasource parameters, patch datasource credentials and start a dataset refresh operation.

## Problems to Solve With App-Owns-Data Embedding

dddd

## Service Principal Profiles to the Rescue

The **AppOwnsDataMultiTenant** application demonstrates how to deal with the Power BI Service limitation which restricts users and service principals from being a member of more than 1000 workspaces. If you are implementing App-Owns-Data embedding in an application which uses a single service principal, Microsoft will only support you in creating up to 1000 workspaces. Once your Power BI embedding environment moves beyond 1000 workspaces, you need a new design strategy to ensure your application does drift into an unsupported state.

The **AppOwnsDataMultiTenant** application demonstrates how to work around the 1000 workspace limitation by creating a new service principal profile for each customer tenant. Here is how it works. Each service principal can support up to 1000 workspaces. Therefore, creating a service principal pool of 10 service principals makes it possible to create and manage 10,000 customer tenant workspaces in a fashion that is supported by Microsoft.

In addition to implementing a service principal pooling scheme, the **AppOwnsDataMultiTenant** application also demonstrates how to create and manage a separate service principal for each customer tenant workspace. An application design which maintains a one-to-one relationship between service principals and customer tenant workspaces is what Microsoft recommends as a best practice because it provides the greatest amount of isolation especially with respect datasource credentials.

You can follow the steps in this document to set up the **AppOwnsDataMultiTenant**  application for testing. To complete these steps, you will require a Microsoft 365 tenant in which you have permissions to create and manage Azure AD applications and security groups. You will also need Power BI Service administrator permissions to configure Power BI settings to give service principals to ability to access the Power BI Service API. If you do not have a Microsoft 365 environment for testing, you can create one for free by following the steps in [Create a Development Environment for Power BI Embedding](https://github.com/PowerBiDevCamp/Camp-Sessions/raw/master/Create%20Power%20BI%20Development%20Environment.pdf).

## Set up your development environment

To set up the **AppOwnsDataMultiTenant** application for testing, you will need to configure a Microsoft 365 environment by completing the following tasks.

1. Create an Azure AD security group named **Power BI Apps**
2. Configure Power BI tenant-level settings for service principal access
3. Create the Azure AD Application for the **AppOwnsDataMultiTenant** Application

The following three sections will step through each of these setup tasks.

### Create an Azure AD security group named Power BI Apps

Begin by navigating to the [Groups management page](https://portal.azure.com/#blade/Microsoft_AAD_IAM/GroupsManagementMenuBlade/AllGroups) in the Azure portal. Once you get to the **Groups** page in the Azure portal, click the **New group** link.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image1.png)

In the **New Group** dialog, Select a **Group type** of **Security** and enter a **Group name** of **Power BI Apps**. Click the **Create** button to create the new Azure AD security group.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image2.png)

Verify that you can see the new security group named **Power BI Apps** on the Azure portal **Groups** page.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image3.png)

### Configure Power BI tenant-level settings for service principal access

Next, you need you enable a tenant-level setting for Power BI named **Allow service principals to use Power BI APIs**. Navigate to the Power BI Service admin portal at <https://app.powerbi.com/admin-portal>. In the Power BI Admin portal, click the **Tenant settings** link on the left.

[Graphical user interface, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image4.png)

Move down in the **Developer settings** section and expand the **Allow service principals to use Power BI APIs** section.

[Graphical user interface, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image5.png)

Note that the **Allow service principals to use Power BI APIs** setting is initially set to **Disabled**.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image6.png)

Change the setting to **Enabled**. After that, set the **Apply to** setting to **Specific security groups** and add the **Power BI Apps** security group as shown in the screenshot below. Click the **Apply** button to save your configuration changes.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image7.png)

You will see a notification indicating it might take up to 15 minutes to apply these changes to the organization.

[Text

Description automatically generated with medium confidence](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image8.png)

Now scroll upward in the **Tenant setting** section of the Power BI admin portal and locate **Workspace settings**.

[Graphical user interface, application, Teams

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image9.png)

Note that a new Power BI tenant has an older policy where only users who have the permissions to create Office 365 groups can create new Power BI workspaces. You must reconfigure this setting so that service principals in the **Power BI Apps** group will be able to create new workspaces.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image10.png)

In **Workspace settings**, set **Apply to** to **The entire organization** and click the **Apply** button to save your changes.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image11.png)

You have now completed the configuration of Power BI tenant-level settings.

### Create the Azure AD Application for the ****AppOwnsDataMultiTenant**** Application

Login to the Azure portal to create the new Azure AD application. Begin by navigating to the [App registration](https://portal.azure.com/#blade/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/RegisteredApps) page in the Azure portal and click the **New registration** link.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image12.png)

On the **Register an application** page, enter an application name such as **Power BI Tenant Management Application** and accept the default selection for **Supported account types** of **Accounts in this organizational directory only**.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image13.png)

Complete the following steps in the **Redirect URI** section.

1. Leave the default selection of **Web** in the dropdown box
2. Enter a **Redirect URI** of [**https://localhost:44300/signin-oidc**](https://localhost:44300/signin-oidc)
3. Click the **Register** button to create the new Azure AD application.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image14.png)

After creating a new Azure AD application in the Azure portal, you should see the Azure AD application overview page which displays the **Application ID**. Note that the ***Application ID*** is often called the ***Client ID***, so don't let this confuse you. You will need to copy this Application ID and store it so you can use it later to configure the project's support for Client Credentials Flow.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image15.png)

Copy the **Client ID** (aka Application ID) and paste it into a text document so you can use it later in the setup process. Note that this **Client ID** value will be used by **AppOwnsDataMultiTenant** project to configure authentication with Azure AD.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image16.png)

Next, repeat the same step by copying the **Tenant ID** and copying that into the text document as well.

[Text

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image17.png)

Your text document should now contain the **Client ID** and **Tenant ID** as shown in the following screenshot.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image18.png)

Next, you need to create a Client Secret for the application. Click on the **Certificates & secrets** link in the left navigation to move to the **Certificates & secrets** page. On the **Certificates & secrets** page, click the **New client secret** button as shown in the following screenshot.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image19.png)

In the **Add a client secret** dialog, add a text description such as **Test Secret** and then click the **Add** button to create the new Client Secret.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image20.png)

Once you have created the Client Secret, you should be able to see its **Value** in the **Client secrets** section. Click on the **Copy to clipboard** button to copy the Client Secret into the clipboard.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image21.png)

Paste the **Client Secret** into the same text document with the **Client ID** and **Tenant ID**.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image22.png)

## Test the Tenant Management project in Visual Studio 2019

In order to run and test the **AppOwnsDataMultiTenant** project on a developer workstation, you must install the .NET 5 SDK and Visual Studio 2019. While this document will walk through the steps of opening and running the **AppOwnsDataMultiTenant** project using Visual Studio 2019, you can also open and run the project using Visual Studio Code if you prefer that IDE. Here are links to download this software if you need them.

1. .NET 5 SDK – [[download](https://dotnet.microsoft.com/download/dotnet/5.0)]
2. Visual Studio 2019 – [[download](https://visualstudio.microsoft.com/downloads/)]
3. Visual Studio Code – [[download](https://code.visualstudio.com/Download)]

### Download the Source Code

The source code for the **AppOwnsDataMultiTenant** project is maintained in a GitHub repository at the following URL.

[**https://github.com/PowerBiDevCamp/AppOwnsDataMultiTenant**](https://github.com/PowerBiDevCamp/TenantManagement)

You can download the **AppOwnsDataMultiTenant** project source files in a single ZIP archive using [this link](https://github.com/PowerBiDevCamp/TenantManagement/archive/refs/heads/main.zip). If you are familiar with the **git** utility, you can clone the project source files to your local developer workstation using the following **git** command.

git clone https://github.com/PowerBiDevCamp/AppOwnsDataMultiTenant.git

Once you have downloaded the source files for the **AppOwnsDataMultiTenant** repository to your developer workstation, you will see there is a top-level project folder named **AppOwnsDataMultiTenant** which contains several files including a solution file named **AppOwnsDataMultiTenant.sln** and a project file named **AppOwnsDataMultiTenant.csproj**.

[Graphical user interface, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image23.png)

### Open the Project in Visual Studio 2019

Launch Visual Studio 2019 and use the **File > Open > Project/Solution** menu command to open the solution file named **AppOwnsDataMultiTenant.sln**. You should note that this development project has been built as a .NET 5 MVC Web Application as shown in the following screenshot.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image24.png)

Let's quickly review the NuGet packages that have been installed in the **AppOwnsDataMultiTenant** project. There are several NuGet packages which add Entity Framework support which make it possible to quickly create the SQL Server database associated with this project.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image25.png)

There are several packages included to add Azure AD authentication support including **Microsoft.Identity.Web** and **Microsoft.Identity.Web.UI**. The package named **Microsoft.Graph** has been included to support .NET programming with the Microsoft Graph API. The package named **Microsoft.PowerBI.Api** has been included to support .NET programming with the Power BI REST API.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image26.png)

### Update application settings in the appsettings.json file

Before you can run the application in the Visual Studio debugger, you must update several critical application settings in the **appsettings.json** file. Open the **appsettings.json** file and examine the JSON content inside. There is three important sections named **AzureAd**, **AppOwnsDataMultiTenantDB** and **DemoSettings**.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image27.png)

Inside the **AzureAd** section, update the **TenantId**, **ClientId** and **ClientSecret** with the data you collected when creating the Azure AD application named **Power BI Tenant Management Application.**

[Text

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image28.png)

If you are using Visual Studio 2019, you shoukd be able leave the database connection string the way it is with the **Server** setting of **(localdb)\\MSSQLLocalDB**. You can change this connection string to point to a different server if you'd rather create the project database named **AppOwnsDataMultiTenantDB** in a different location.

[Text

Description automatically generated with low confidence](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image29.png)

In the **DemoSettings** section there is a property named **AdminUser**. The reason that this property exists has to with you being able to see Power BI workspaces as they are created by service principals. There is code in the **AppOwnsDataMultiTenant** application that will add the user specified by the **AdminUser** setting as a workspace admin any times it creates a new Power BI workspace. This just makes things much easier for you to see what's going on when you begin to run and test the application.

Update the **AdminUser** setting with your Azure AD account name so that you will be able to see all the Power BI workspaces created by this application.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image30.png)

### Create the ****AppOwnsDataMultiTenantDB**** database

Before you can run the application in Visual Studio, you must create the project database named **AppOwnsDataMultiTenantDB**. This database schema has been created using the .NET 5 version of the Entity Framework. In this step, you will execute two PowerShell cmdlets provided by Entity Framework to create the database.

Before creating the **AppOwnsDataMultiTenantDB** database, take a moment to understand how it’s been structured. Start by opening the file named **AppOwnsDataMultiTenantDB.cs** in the **Models** folder. Note that you shouldn't make any change to **AppOwnsDataMultiTenantDB.cs**. You are just going to inspect the file you understand how the **AppOwnsDataMultiTenantDB** database is generated.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image31.png)

When you inspect the code inside **AppOwnsDataMultiTenantDB.cs**, you will see a class named **AppOwnsDataMultiTenantDB** that derives from **DbContext** to add support for automatic database generation using Entity Framework. The **AppOwnsDataMultiTenantDB** class serves as the top-level class for the Entity Framework which contains two **DBSet** properties named **AppIdentites** and **Tenants**. When you generate the database, each of these **DBSet** properties will be created as database tables. The **AppIdentites** table is generated using the table schema defined by the **PowerBiAppIdentity** class.

[Text

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image32.png)

The **Tenants** table is generated using the table schema defined by the **PowerBiTenant** class.

[Text

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image33.png)

After you have inspected the code used to generated the database, close the source file named **AppOwnsDataMultiTenantDB.cs** without saving any changes. The next step is to run the PowerShell commands to create the project database named **AppOwnsDataMultiTenantDB**.

Open the Package Manager console using **Tools > NuGet Package Manager > Package Manager Console**.

[Graphical user interface, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image34.png)

You should see the **Package Manager Console** command prompt where you can execute PowerShell commands.

[Graphical user interface, text

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image35.png)

Type and execute the following **Add-Migration** command to create a new Entity Framework migration in the project.

Add-Migration InitialCreate

The **Add-Migration** command should run without errors. If this command fails you might have to modify the database connection string in **appsettings.json**.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image36.png)

After running the Add-Migration command, you will see a new folder has been added to the project named **Migrations** with several C# source files. There is no need to change anything in thee source files but you can inspect what's inside them if you are curious how the Entity Framework does its work.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image37.png)

Return to the **Package Manager Console** and run the following **Update-Database** command to generate the database named **AppOwnsDataMultiTenantDB**.

Update-Database

The **Update-Database** command should run without errors and generate the database named **AppOwnsDataMultiTenantDB**.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image38.png)

In Visual Studio, you can use the **SQL Server Object Explorer** to see the database that has just been created. Open the **SQL Server Object Explorer** by invoking the **View >** **SQL Server Object Explorer** menu command.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image39.png)

Expand the **Databases** node for the server you are using and verify you an see the new database named **AppOwnsDataMultiTenantDB**.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image40.png)

If you expand the **Tables** node for **AppOwnsDataMultiTenantDB**, you should see the two tables named **AppIdentities** and **Tenants**.

[A picture containing text

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image41.png)

The **AppOwnsDataMultiTenantDB** database has now been set up and you are ready to run the application in the Visual Studio debugger.

## Test the Tenant Management Application

Launch the **AppOwnsDataMultiTenant** web application in the Visual Studio debugger by pressing the **{F5}** key or clicking the Visual Studio **Play** button with the green arrow and the caption **IIS Express**.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image42.png)

When the application starts, click the **Sign in** link in the upper right corner to begin the user login sequence.

[Graphical user interface, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image43.png)

The first time you authenticate with Azure AD, you'll be prompted with the **Permissions requested** dialog asking you to accept the delegated permissions for the Microsoft Graph API requested by the application. Click the **Accept** button to grant these permissions and continue.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image44.png)

Once you have logged in, you should see your name in the welcome message.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image45.png)

### Create App Identities

Start by creating a few new App Identities. Click the **App Identities** link to navigate to the **App Identities** page.

[Graphical user interface

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image46.png)

Click the **Add New App Identity to Pool** button to display the **Create New App Identity** page.

[Graphical user interface, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image47.png)

When you open the **Create New App Identity** page, it will automatically populate the **App Identity Name** textbox with a value of **ServicePrincipal01**. Click the **Add New App Identity to Pool** button to create the new app identity.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image48.png)

After a few second, you should see the new app identity named **ServicePrinicpal01** on the **App Identities** page.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image49.png)

Follow the same steps to create two more app identities named **ServicePrincipal02** and **ServicePrincipal03**. When you're done, the **App Identities** page should match the following screenshot.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image50.png)

Note that behind the scenes the **AppOwnsDataMultiTenant** application is using the Microsoft Graph API to create new Azure AD application each time you create a new app identity. If you return pack to the [App registration page](https://portal.azure.com/#blade/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/RegisteredApps) in the Azure portal, you will see that an Azure AD application has been created for each app identity you've created.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image51.png)

If you return to the [Groups](https://portal.azure.com/#blade/Microsoft_AAD_IAM/GroupsManagementMenuBlade/AllGroups) page in the Azure portal and drill into the **Members** page of the **Power BI Apps** security group, you will see that the **AppOwnsDataMultiTenant** application has also added the service principal for each azure AD application as a group member. This is important because these service principals must be added to this Azure AD security group or they will not be able to call the Power BI REST API.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image52.png)

In addition to communicating with Azure AD to create and configure Azure AD application, the **AppOwnsDataMultiTenant** application also captures application metadata and authentication credentials so it can store them in the **AppOwnsDataMultiTenantDB** database. Soon you will see how the **AppOwnsDataMultiTenant** application is able to retrieve these credentials on demand and authenticate with Azure AD under the identity of any of these Azure AD applications.

[Graphical user interface, application, Word

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image53.png)

**CAVEAT**: Keep in mind that the **AppOwnsDataMultiTenant** application has been designed as a proof-of-concept (POC) application to teach concepts and provide a starting point for other developers. This application does not include certain aspects that are important to include in a real-world applications such as hiding secrets. If you plan to extend this POC sample application into a production application, it will be your responsibility to add support for hiding credentials such as the Client Secret. You can consider an approach such as using the [Always Encrypted](https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-database-engine?view=sql-server-ver15) feature in Azure SQL or extending the **AppOwnsDataMultiTenant** application to store client secrets or client certificates in [Azure Key Vault](https://docs.microsoft.com/en-us/azure/key-vault/general/basic-concepts).

### Create New Power BI Tenants

Return to the **AppOwnsDataMultiTenant** application and navigate to the **Tenants** page.

[Graphical user interface

Description automatically generated with low confidence](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image54.png)

Click the **Onboard New Tenant** button to display the **Onboard New Tenant** page.

[Graphical user interface

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image55.png)

You can create the first tenant using the default values supplied by the **Onboard New Tenant** page. Click to **Create New Tenant** button to begin the process of creating a new customer tenant.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image56.png)

After a few seconds, you should see the new customer tenant has been created.

[Graphical user interface, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image57.png)

Click the **Onboard New Tenant** button again to create a second tenant.

[Graphical user interface

Description automatically generated with low confidence](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image58.png)

This time, select a different database for **Database Name** and then click **Create New Tenant**.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image59.png)

You should now have two customer tenants. Note they each tenant has a different app identity as its **Owner**.

[Table

Description automatically generated with medium confidence](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image60.png)

Follow the same steps to create two more customer tenants so that there are 3 app identities and 4 customer tenants. Once you have created more tenants then app identities, you should see app identity pooling where multiple customer tenants share the same app identity.

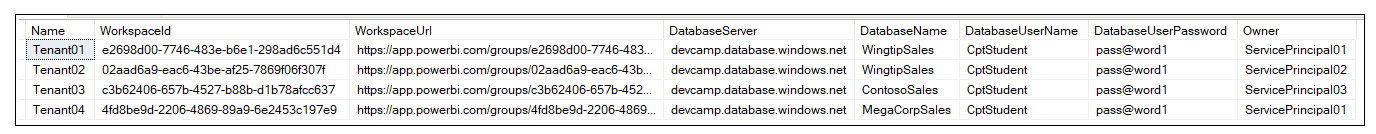
[Graphical user interface, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image61.png)

Now let's discuss what's going on behind the scenes. As you create a new customer tenant, the **AppOwnsDataMultiTenant** application uses the Power BI REST API to implement the following onboarding logic.

1. Create a new Power BI workspace
2. Upload a [template PBIX file](https://github.com/PowerBiDevCamp/TenantManagement/raw/main/TenantManagement/wwwroot/PBIX/DatasetTemplate.pbix) to create the **Sales** dataset and the **Sales** report
3. Update dataset parameters on **Sales** dataset to point to this customer's database
4. Patch credentials for the SQL datasource used by the **Sales** dataset
5. Start a refresh operation on the **Sales** database

The **AppOwnsDataMultiTenant** application also create a new record in the **Tenants** table of the **AppOwnsDataMultiTenantDB** database. Note that the application identity associated with this customer tenant is tracked in the **Owner** column.

[](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image62.png)

Click on the **View** button for a specific tenant on the **Power BI Tenants** page to drill into the **Tenant Details** page.

[Graphical user interface

Description automatically generated with medium confidence](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image63.png)

The **Tenant Details** page displays Power BI workspace details including its members, datasets and reports.

[Graphical user interface

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image64.png)

Click on the back arrow to return to the **Power BI Tenants** page.

[Graphical user interface, text, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image65.png)

If you're interested, you can examine the details of other tenants as well.

### Embed Reports

Now it's time to make use of the **AppOwnsDataMultiTenant** application's ability to embed reports. When navigate to the **Embed** page for a customer tenant, the **AppOwnsDataMultiTenant** application must acquire an access token for whichever app identity was used to create the customer tenant. The service principal that is configured as the **Owner** of a tenant will be the only service principal who will have access to access the target workspace in Power BI.

Move to the **Power BI Tenants** page and click on the **Embed** button for the first customer tenant.

[Graphical user interface

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image66.png)

You should now see a page with an embedded report for that tenant. When you click the **Embed** button to embed a report for a customer tenant, the **TenanantManagement** application retrieves credentials for the app identity associated with the tenant from the **AppOwnsDataMultiTenantDB** database. It then uses those credentials to acquire an access token from Azure AD using Client Credentials Flow. That access token is then used to communicate with the Power BI Service to retrieve report metadata and generate an embed token for the embedding process.

[Graphical user interface

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image67.png)

Click on the back arrow button to return to the **Tenants** page.

[Graphical user interface, website

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image68.png)

Now test clicking the **Embed** button for other customer tenants. As you can see, the **AppOwnsDataMultiTenant** application has the ability to acquire access tokens for any of the Azure AD applications that it has created.

[Graphical user interface

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image69.png)

### Inspect the Power BI Workspaces

If you're curious about what's been created in Power BI, you can see for yourself by navigating to the Power BI Service portal at [https://app.powerbi.com](https://app.powerbi.com/). You should be able to see and navigate to any of the Power BI workspaces that have been created by the **AppOwnsDataMultiTenant** application.

[A picture containing graphical user interface

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image70.png)

Navigate to one of these workspaces such as **Tenant01**.

[Graphical user interface, text, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image71.png)

Drill into the **Setting** page for the dataset named **Sales**.

[Graphical user interface, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image72.png)

You should be able to verify that the **Sales** dataset has been configured by one of the Azure AD applications that was created by the **AppOwnsDataMultiTenant** application. You should also be able to see the **Last refresh succeeded** message for the dataset refresh operation that was started by the **AppOwnsDataMultiTenant** as part of its tenant onboarding logic.

[Graphical user interface, application

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image73.png)

### Create a Separate Service Principal for Each Tenant

At this point you have used the **AppOwnsDataMultiTenant** application to pool app identities where one service principal can be the owner of multiple customer tenants. While this approach will work for many organizations and ISVs using Power BI embedding, you can take things one step further by creating a new service principal each time you create a new customer tenant. An application design which maintains a one-to-one relation between service principals and customer tenants will provide the most secure level of isolation. When you pool a service principal across tenants, that service principal will be the owner of datasource credentials for more than a single client.

When creating a new customer tenant using the **Onboard New Tenant** page, you can select the option to **Create Tenant using New Exclusive App Identity**. If you select this option, the **AppOwnsDataMultiTenant** application will create a new Azure AD application and then use the service principal from that application to create the workspace in Power BI.

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image74.png)

You should take note that The **AppIdentities** table in the **AppOwnsDataMultiTenantDB** database contains a boolean column named **Exclusive**. When an app identity has an **Exclusive** column value of **true**, there is logic in the application which knows it should not include that app identity in the pool of app identities available on the **Onboard New Tenant** page.

[A picture containing table

Description automatically generated](https://github.com/PowerBiDevCamp/TenantManagement/blob/main/Images/ReadMe/media/image76.png)

This concludes the walkthrough of the **AppOwnsDataMultiTenant** application.