Contents

What is Power BI embedded analytics?

6/30/2022 • 2 minutes to read • Edit Online

Power BI embedded analytics allows you to embed your Power BI content such as reports, dashboards and tiles, in a web application or in a website.

Using Power BI embedded analytics you can:

- Deliver compelling data experiences for your end users, enabling them to take action based on insights from your solutions data.
- Quickly and easily provide exceptional customer-facing reports, dashboards, and analytics in your own apps by using and branding Power BI as your own.
- Reduce developer resources by automating the monitoring, management, and deployment of analytics, while getting full control of Power BI features and intelligent analytics.

What are the Power BI embedded analytics solutions?

Power BI embedded analytics offers two solutions:

- Embed for your customers
- Embed for your organization

Embed for your customers

The *embed for your customers* solution allows you to build an app that uses non-interactive authentication against Power BI. Your customers are likely to be external users, and they don't need to sign in using Power BI credentials to view the embedded content. Typically, this solution is used by independent software vendors (ISVs) who are developing applications for third parties. For a tutorial, see Embed Power BI content using a sample embed for your customers application.

Embed for your organization

The *embed for your organization* solution allows you to build an app that requires signing in using Power BI credentials. Once signed in users can only consume embedded content, they have access to on Power BI service. This solution is aimed at large organizations that are building an app for internal users. For a tutorial, see Embed Power BI content into an application for your organization.

Comparing the solutions

The following table provides a comparison between the two Power BI embedded analytics solutions.

EMBED FOR YOUR CUSTOMERS	EMBED FOR YOUR ORGANIZATION
Also known as **app owns data**	Also known as **user owns data**
Aimed at external users	Aimed at internal users
To authenticate app users, use your own authentication method	App users authenticate against Azure AD
App users don't need a license	Each app user needs a Power BI license

EMBED FOR YOUR CUSTOMERS	EMBED FOR YOUR ORGANIZATION
Non-interactive authentication. Your app uses a *service principal* or a *master user* to authenticate	Interactive authentication. Your app uses the app user's credentials to authenticate

> **TIP**
> Get started with the Power BI embedded analytics setup tool.

What are Power BI capacities?

Capacity is a set of resources reserved for exclusive use. It enables you to publish dashboards, reports, and datasets to users, without having to purchase per-user licenses. It also offers dependable, consistent performance for your content.

There are two types of Power BI embedded analytics offerings, each requiring a different capacity:

- Power BI Embedded
- Embedding with Power BI

Power BI Embedded

Power BI Embedded is an Azure offer that requires A SKUs. *Power BI Embedded* is associated with the *embed for your customers* solution.

Embedding with Power BI

Embedding with Power BI is a Power BI Premium offer that requires P or EM SKUs.

Considerations and limitations

The ESRI ArcGIS visual isn't supported yet in the embed for your customers (app owns data) solution.

Next steps

Power BI Embedded Generation 2

Capacity and SKUs in Power BI embedded analytics

Tutorial: Embed Power BI content using a sample embed for your customers application

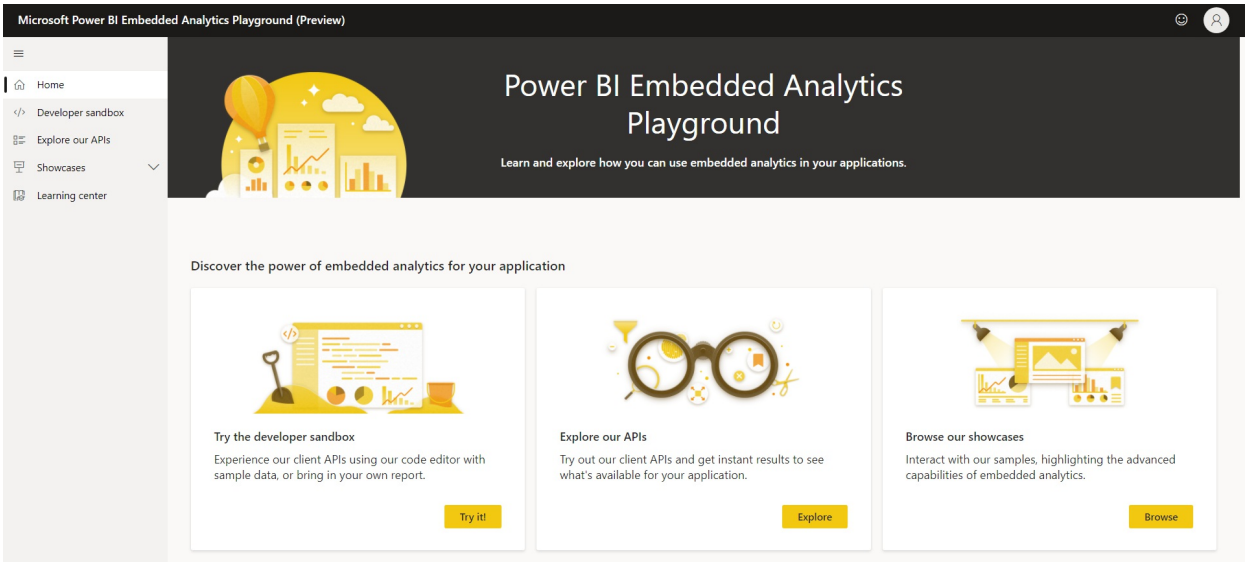
Tutorial: Embed Power BI content using a sample embed for your organization application

Power BI embedded analytics setup tool

What is the Power BI embedded analytics playground?

6/30/2022 • 2 minutes to read • Edit Online

The Power BI embedded analytics playground makes it easy for you to learn, explore, and try out Power BI embedded analytics. It's also where you can keep up with all the new features and updates of Power BI embedded.



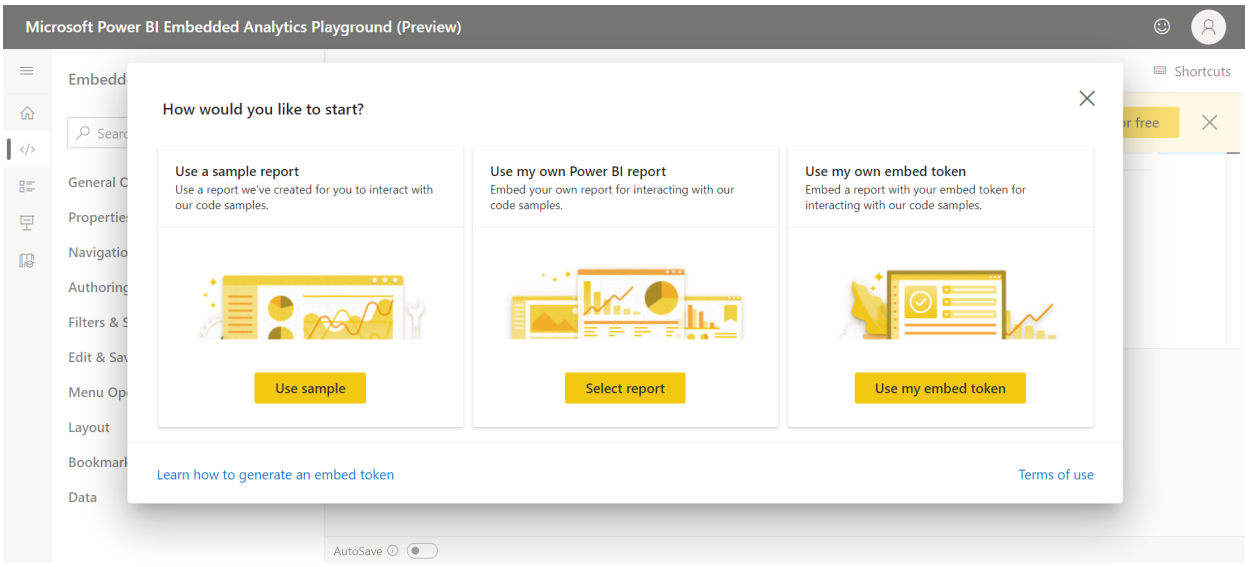
The playground gives you hands-on coding experience, and lets you embed your own reports, and interact with Power BI client APIs giving you instant results. The playground has the following main experiences:

- Developer sandbox
- Explore our APIs
- Showcases
- Learning center

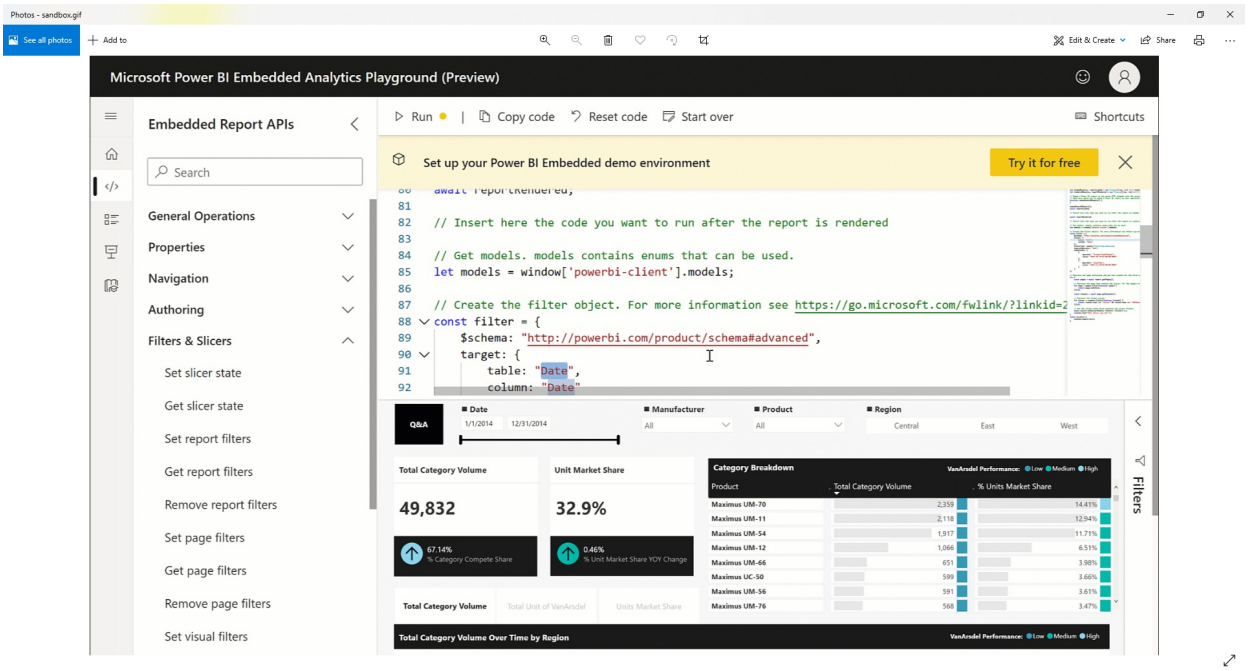
> **TIP**
> For a demonstration of how to best use the playground, watch this video

Developer sandbox

Go to the developer sandbox for hands-on experience using our client APIs. You can use the APIs with our sample report or with your own report.

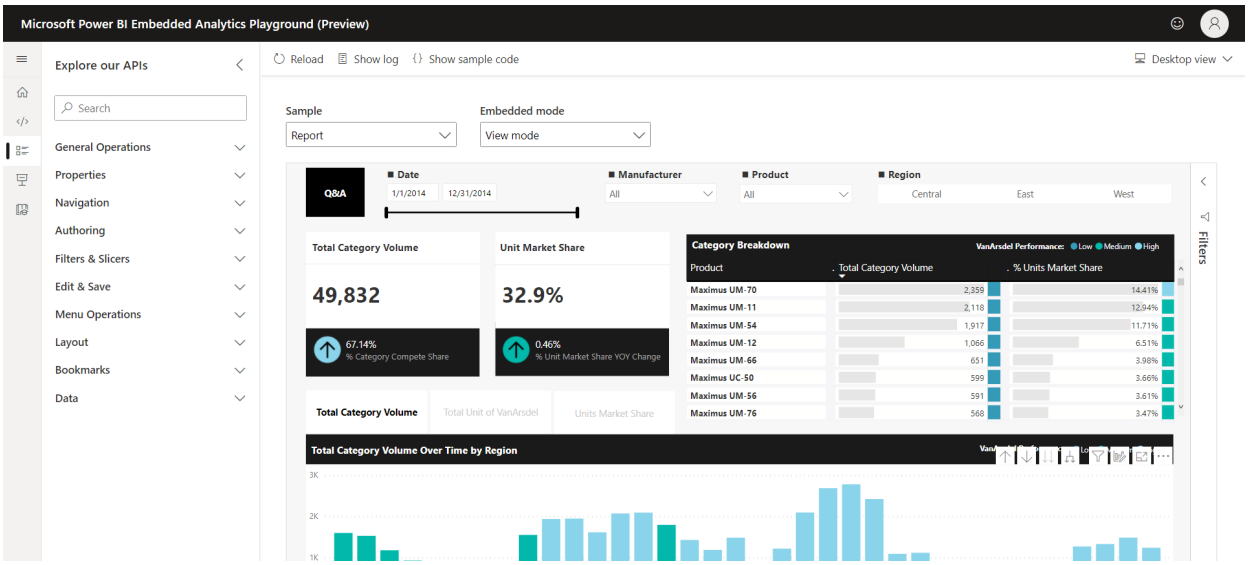


Drag and drop code snippets into the report, or type them directly into the code editor area to see how they work.



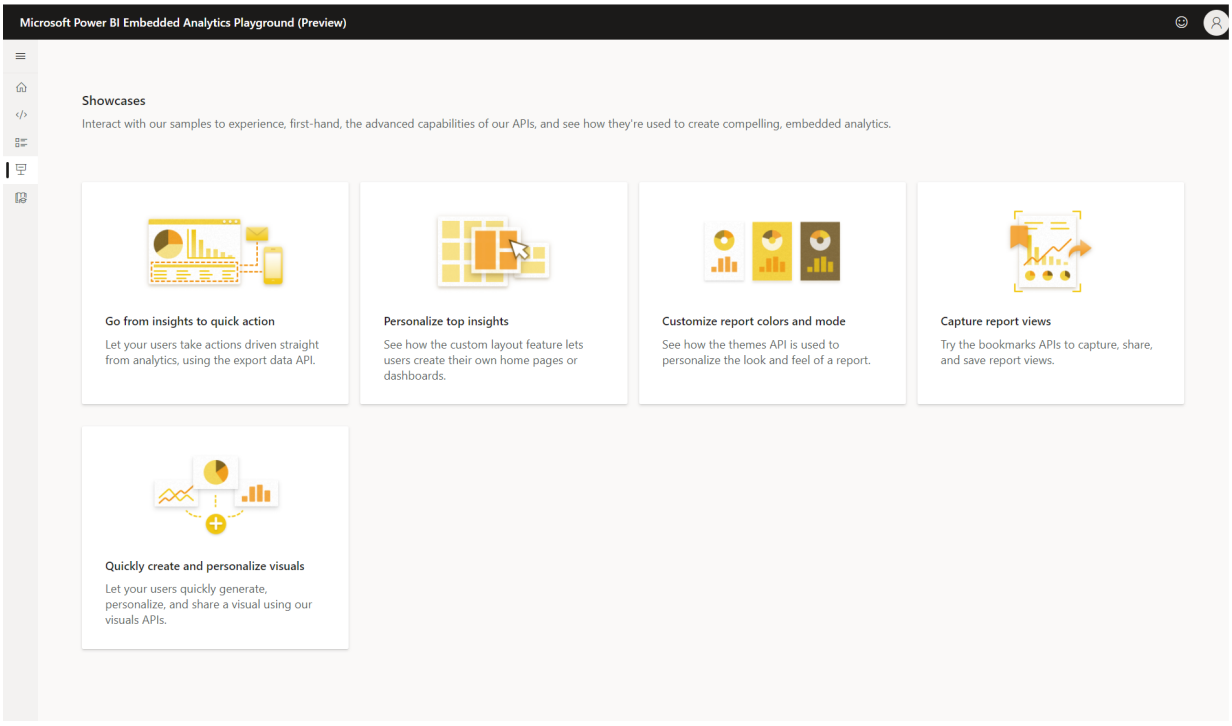
Explore our APIs

You can interact with code snippets and embed reports, dashboards, Q&As, and more in **Explore our APIs**.



Showcases

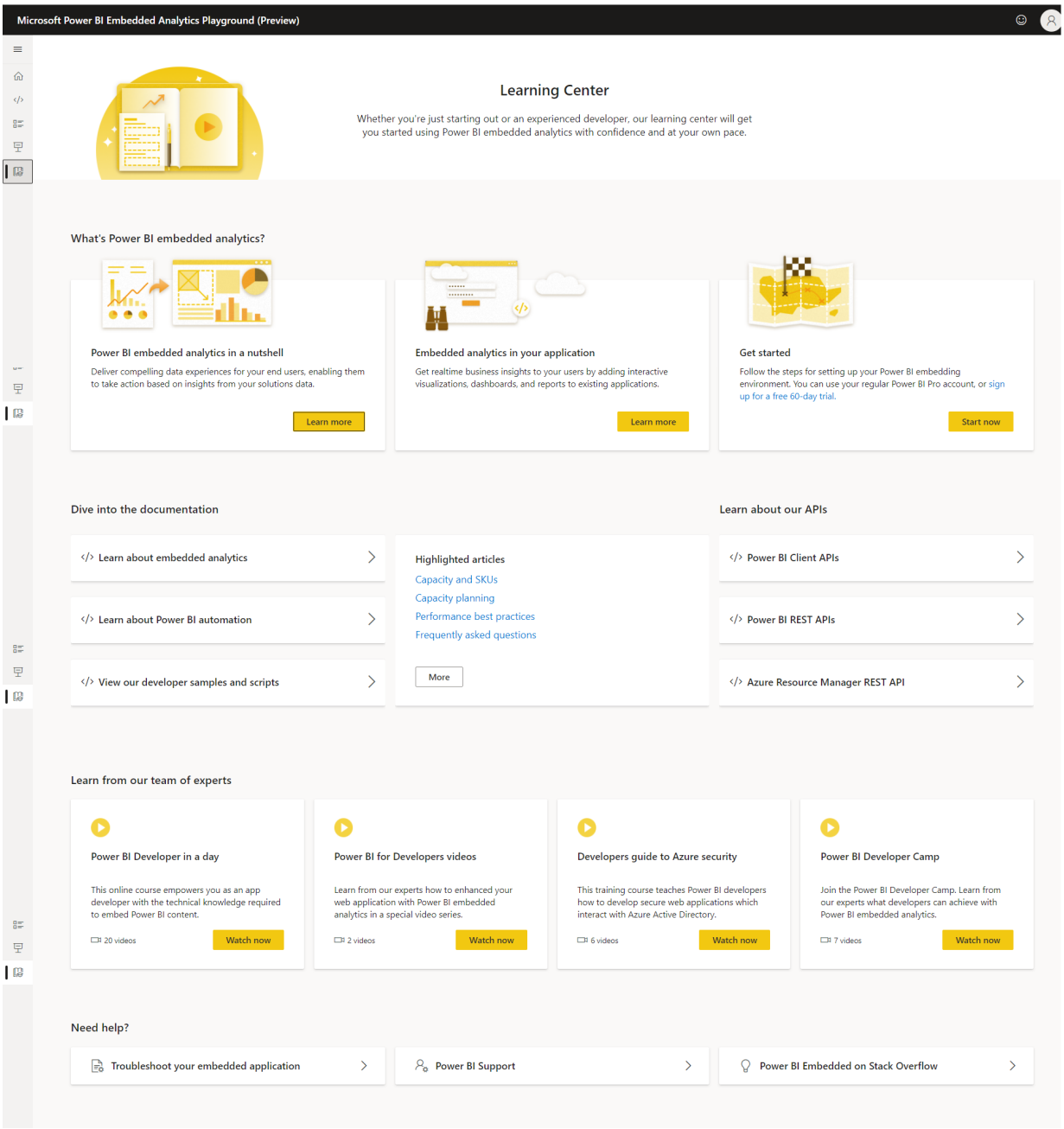
The interactive showcases let you see how to apply these features in your applications. Each showcase presents an application that demonstrates what you can do with one or more of the client APIs.



The showcases code is open-sourced, and you can find the code behind all of them in our GitHub repository.

Learning center

The Learning center is a collection of Power BI embedded analytics resources. It's where you can dive into our documentation, learn about our APIs, find our developer samples and videos, and learn where to get help.



Considerations and limitations

The Power BI embedded playground runs in *user owns data* mode only.

Next steps

Explore the embedded analytics playground

Tutorial: Embed Power BI content using a sample *embed for your customers'* application

6/30/2022 • 16 minutes to read • Edit Online

Embedded analytics and **Power BI Embedded** (the Azure offer) allow you to embed Power BI content such as reports, dashboards and tiles, into your application.

In this tutorial, you'll learn how to:

- Set up your embedded environment.
- Configure an *embed for your customers* (also known as *app owns data*) sample application.

To use your application, your users won't need to sign in to Power BI or have a Power BI license.

We recommend using the *embed for your customers* method to embed your Power BI content, if you're an independent software vendor (ISV) or a developer, who wants to create applications for third parties.

> **IMPORTANT**
> If you are embedding content for a national cloud, the first few steps of this tutorial are different. See Embed content for national clouds for details.

Code sample specifications

This tutorial includes instructions for configuring an *embed for your customers* sample application in one of the following frameworks:

- .NET Framework
- .NET Core
- Java
- Node JS
- Python

The code samples support the following browsers:

- Microsoft Edge
- Google Chrome
- Mozilla Firefox

Prerequisites

Before you start this tutorial, verify that you have both the Power BI and code dependencies listed below:

- **Power BI dependencies**
  - Your own Azure Active Directory tenant.
  - To authenticate your app against Power BI, you'll need one of the following:
    - Service principal - An Azure Active Directory (Azure AD) service principal object that allows Azure AD to authenticate your app.

- Power BI Pro license - This will be your **master user** and your app will use it to authenticate to Power BI.
- A Power BI Premium Per User (PPU) license - This will be your **master user** and your app will use it to authenticate to Power BI.

> **NOTE**
> To move to production you'll need a capacity.

- **Code dependencies**

- .NET Core
- .NET Framework
- Java
- Node JS
- Python

- .NET Core 3.1 SDK (or higher)
- An integrated development environment (IDE). We recommend using one of the following:
  - Visual Studio
  - Visual Studio Code

Method

To create an *embed for your customers* sample app, follow these steps:

1. Select your authentication method.
2. Register an Azure AD application.
3. Create a Power BI workspace.
4. Create and publish a Power BI report.
5. Get the embedding parameter values.
6. Service principal API access
7. Enable workspace access.
8. Embed your content.

Step 1 - Select your authentication method

Your embedded solution will vary depending on the authentication method you select. Therefore, it's important to understand the differences between the authentication methods, and decide which one best suits your solution.

The table below describes a few key differences between the service principal and **master user** authentication methods.

CONSIDERATION	SERVICE PRINCIPAL	MASTER USER

CONSIDERATION	SERVICE PRINCIPAL	MASTER USER
Mechanism	Your Azure AD app's service principal object allows Azure AD to authenticate your embedded solution app against Power BI.	Your Azure AD app uses the credentials (username and password) of a Power BI user, to authenticate against Power BI.
Security	*Service principal* is the Azure AD recommended authorization method. If you're using a service principal, you can authenticate using either an *application secret* or a *certificate*. This tutorial only describes using *service principal* with an *application secret*. To embed using a *service principal* and a *certificate*, refer to the service principal with a certificate article.	This authentication method isn't as secure as a *service principal*. You have to be vigilant with the *master user* credentials (username and password). For example, don't expose them in your embedding application, and change the password frequently.
Azure AD delegated permissions	Not required.	Your *master user* or an administrator has to grant consent for your app to access Power BI REST API permissions (also known as scopes). For example, *Report.ReadWrite.All*.
Power BI service access	You can't access Power BI service with a *service principal*.	You can access Power BI service with your *master user* credentials.
License	Doesn't require a Pro license. You can use content from any workspace that you're a member or an admin of.	Requires a Power BI Pro or Premium Per User (PPU) license.

Step 2 - Register an Azure AD application

Registering your application with Azure AD allows you to:

- Establish an identity for your app
- Let your app access the Power BI REST APIs
- If you're using a *master user* - Specify your app's Power BI REST permissions

To register your application with Azure AD, follow the instructions in Register your application.

> **NOTE**
> Before registering your application, you'll need to decide which authentication method to use, *service principal* or *master user*.

Step 3 - Create a Power BI workspace

Power BI keeps your reports, dashboards, and tiles in a workspace. To embed these items, you'll need to create them and upload them into a workspace.

> **TIP**
> If you already have a workspace, you can skip this step.

To create a workspace, do the following:

1. Sign in to Power BI.
2. Select **Workspaces**.
3. Select **Create a workspace**.
4. Name your workspace and select **Save**.

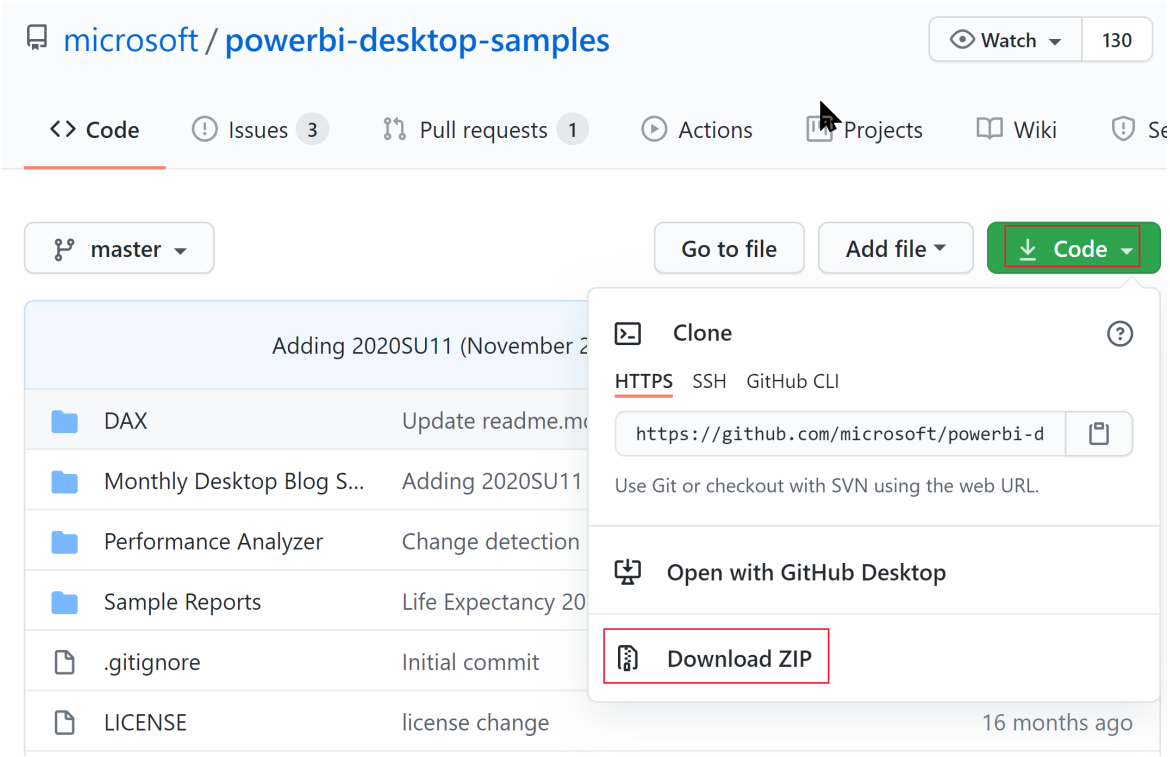
Step 4 - Create and publish a Power BI report

Your next step is to create a report and upload it to your workspace. You can create your own report using Power BI Desktop, and then publish it to your workspace. Or, you can upload a sample report to your workspace.

> **TIP**
> If you already have a workspace with a report, you can skip this step.

To download a sample report and publish it to your workspace, follow these steps:

1. Open the GitHub Power BI Desktop samples folder.
2. Select **Code** and then select **Download zip**.



3. Extract the downloaded ZIP and navigate to the **Samples Reports** folder.
4. Select a report to embed, and publish it to your workspace.

Step 5 - Get the embedding parameter values

To embed your content, you need to obtain certain parameter values. The table below shows the required values, and indicates if they're applicable to the *service principal* authentication method, the *master user* authentication method, or both.

Before you embed your content, make sure you have all the values listed below. Some of the values will differ,

depending on the authentication method you're using.

PARAMETER	SERVICE PRINCIPAL	MASTER USER
Client ID	✓	✓
Workspace ID	✓	✓
Report ID	✓	✓
Client secret	✓	✗
Tenant ID	✓	✗
Power BI username	✗	✓
Power BI password	✗	✓

Client ID

> **TIP**
> **Applies to:** ✓ Service principal ✓ Master user

To get the client ID GUID (also know as *application ID*), follow these steps:

1. Log into Microsoft Azure.
2. Search for **App registrations** and select the **App registrations** link.
3. Select the Azure AD app you're using for embedding your Power BI content.
4. From the **Overview** section, copy the **Application (client) ID** GUID.

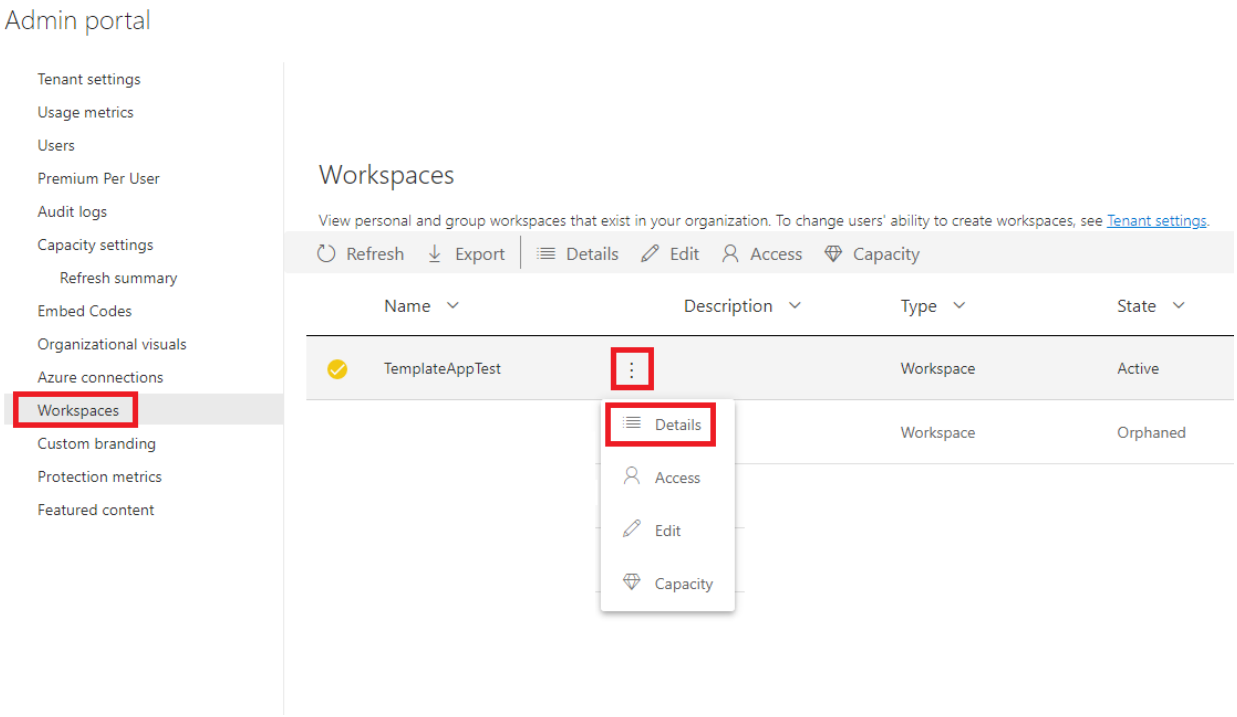
Workspace ID

> **TIP**
> **Applies to:** ✓ Service principal ✓ Master user

To get the workspace ID GUID, follow these steps:

1. Sign in to Power BI service.
2. Open the report you want to embed.
3. Copy the GUID from the URL. The GUID is the number between **/groups/** and **/reports/**.

Alternatively, you can find the workspace ID in the **Admin portal** settings by selecting **Details** next to the workspace name.



Report ID

> **TIP**
> **Applies to:** Service principal Master user

To get the report ID GUID, follow these steps:

1. Sign in to Power BI service.
2. Open the report you want to embed.
3. Copy the GUID from the URL. The GUID is the number between **/reports/** and **/ReportSection**.

   /reports/de0db6db-232f-4807-b5b5-1abe1d71da76/ReportSection

Client secret

> **TIP**
> **Applies to:** Service principal Master user

To get the client secret, follow these steps:

1. Log into Microsoft Azure.
2. Search for **App registrations** and select the **App registrations** link.
3. Select the Azure AD app you're using for embedding your Power BI content.
4. Under **Manage**, select **Certificates & secrets**.
5. Under **Client secrets**, select **New client secret**.
6. In the **Add a client secret** pop-up window, provide a description for your application secret, select when the application secret expires, and select **Add**.
7. From the **Client secrets** section, copy the string in the **Value** column of the newly created application

secret. The client secret value is your *client ID*.

> **NOTE**
> Make sure you copy the client secret value when it first appears. After navigating away from this page, the client secret will be hidden and you'll not be able to retrieve its value.

Tenant ID

> **TIP**
> **Applies to:** ✔ Service principal ✘ Master user

To get the tenant ID GUID, follow these steps:

1. Log into Microsoft Azure.
2. Search for **App registrations** and select the **App registrations** link.
3. Select the Azure AD app you're using for embedding your Power BI content.
4. From the **Overview** section, copy the **Directory (tenant) ID** GUID.

Power BI username and password

> **TIP**
> **Applies to:** ✘ Service principal ✔ Master user

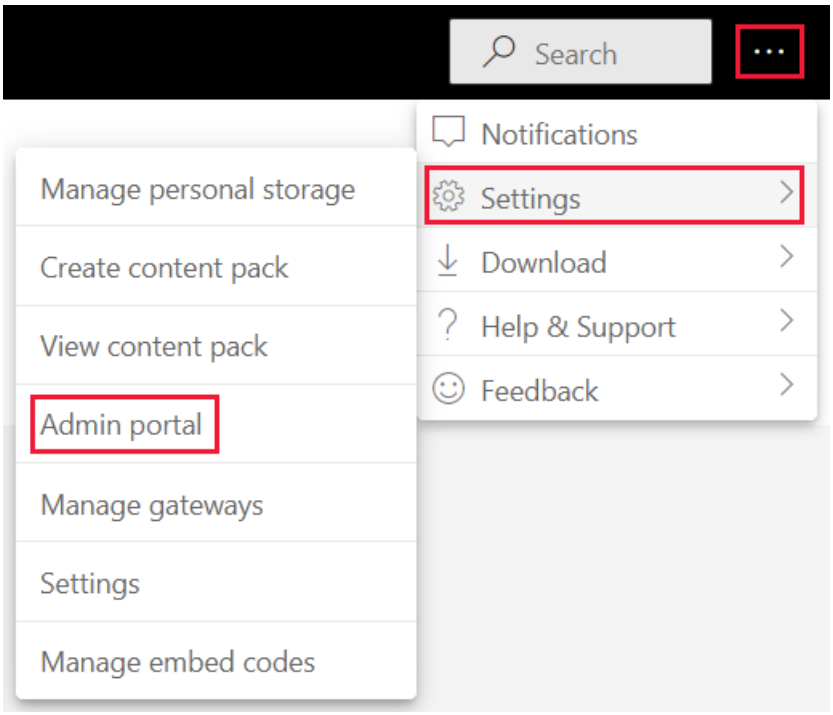
Obtain the *username* and *password* of the Power BI user you're using as your **master user**. This is the same user you used to create a workspace and upload a report to, in Power BI service.

Step 6 - Service principal API access

> **TIP**
> **Applies to:** ✔ Service principal ✘ Master user
>
> This step is only relevant if you're using the *service principal* authentication method. If you're using a *master user*, skip this step and continue with Step 7 - Enable workspace access.

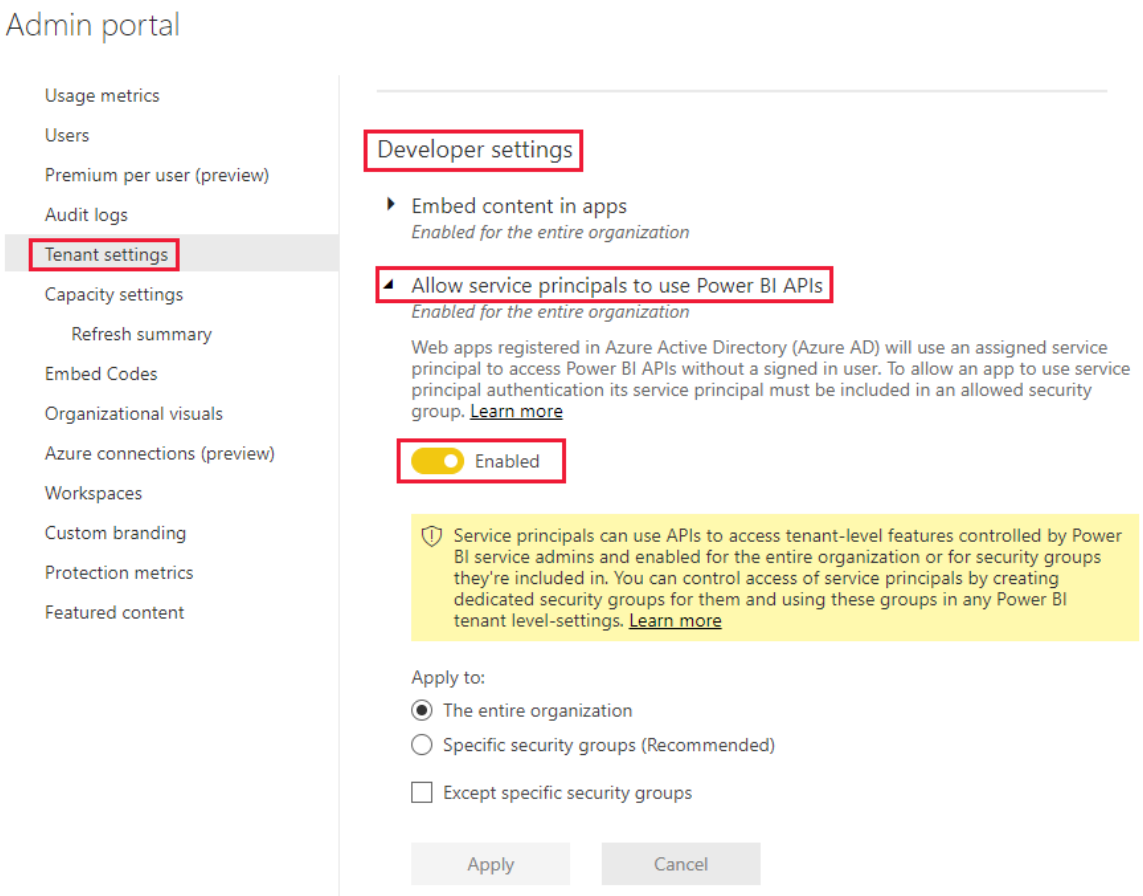
For an Azure AD app to be able to access the Power BI content and APIs, a Power BI admin needs to enable service principal access in the Power BI admin portal. If you're not the admin of your tenant, get the tenant's admin to enable the *Tenant settings* for you.

1. In *Power BI service*, select **Settings** > **Settings** > **Admin portal**.



2. Select **Tenant settings** and then scroll down to the **Developer settings** section.

3. Expand **Allow service principals to use Power BI APIs**, and enable this option.



NOTE

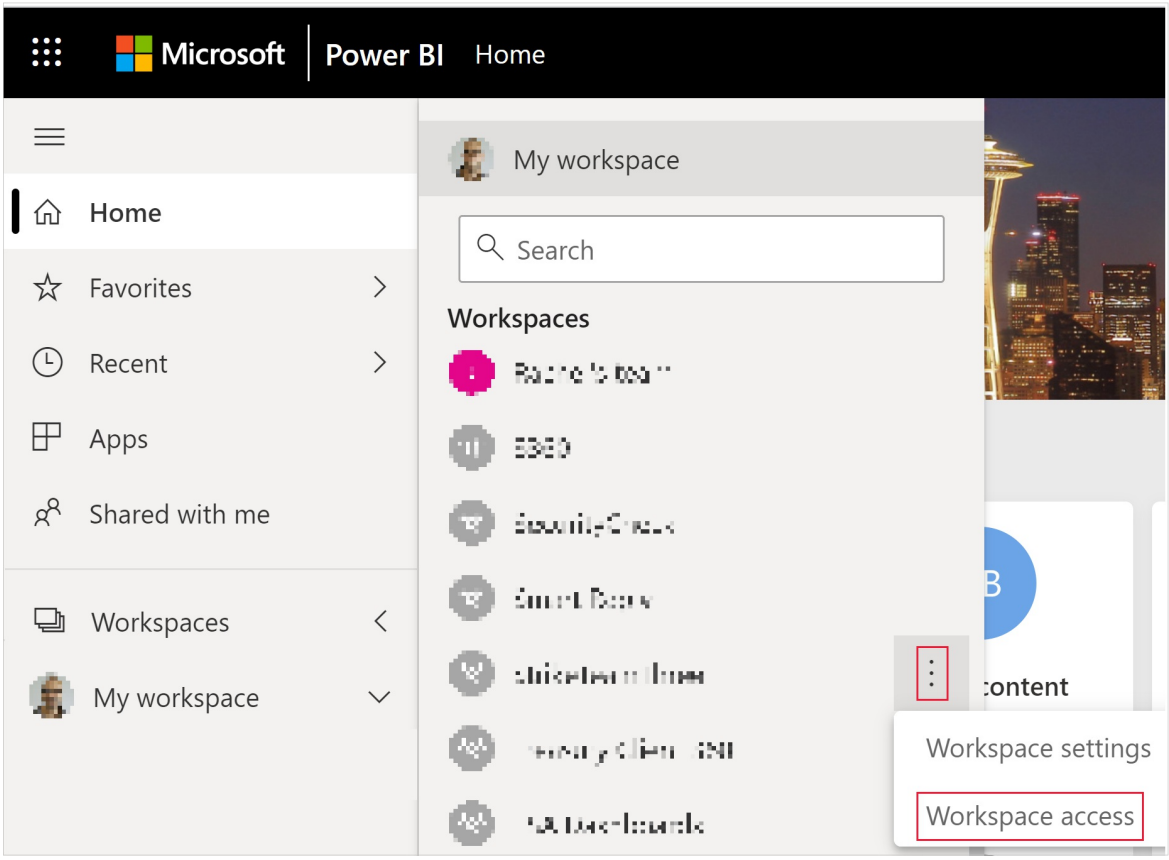
When using a *service principal*, it's recommended to limit its access to the tenant settings using a *security group*. To learn more about this feature, see these sections in the service principal article:

- Create an Azure AD security group
- Enable the Power BI service admin settings

Step 7 - Enable workspace access

To enable your Azure AD app access objects such as reports, dashboards and datasets in the Power BI service, add the *service principal* or *master user*, as a *member* or *admin* to your workspace.

1. Sign in to Power BI service.
2. Scroll to the workspace you want to enable access for, and from the **More** menu, select **Workspace access**.



3. In the **Access** pane, depending on which authentication method you're using, copy the *service principal* or *master user* to the **Enter email address** text box.

> **NOTE**
> If you're using a *service principal*, its name is the name you gave your Azure AD app.

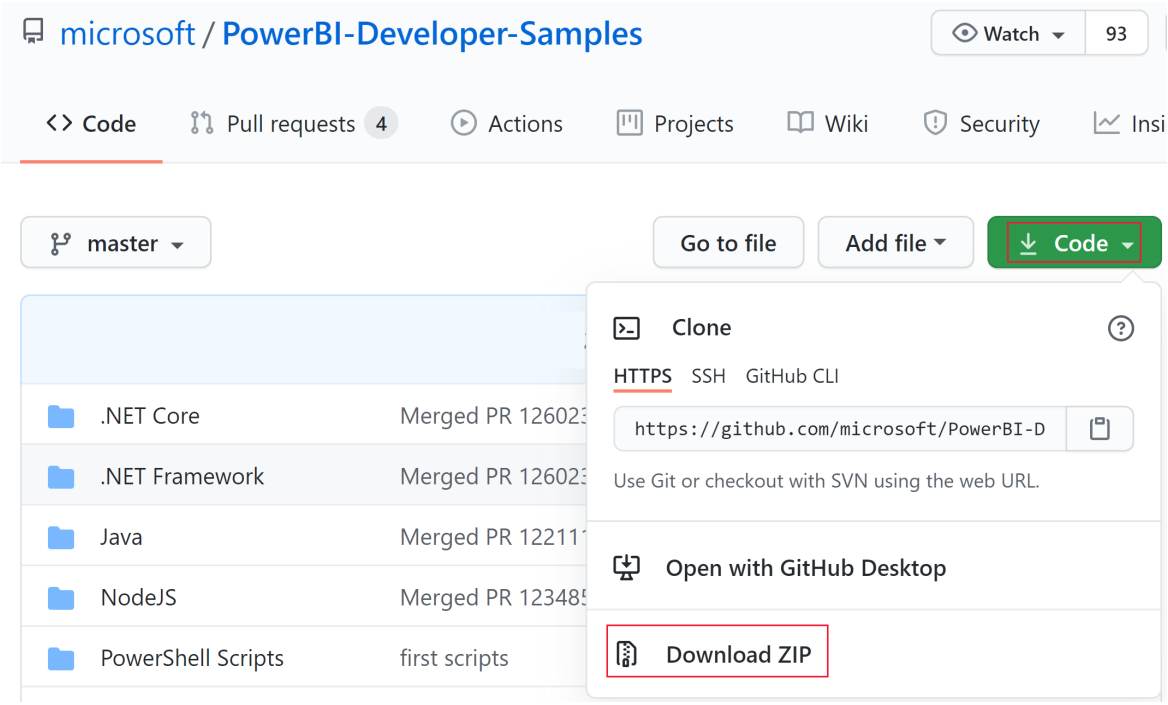
4. Select **Add**.

Step 8 - Embed your content

The Power BI embedded sample application allows you to create an *embed for your customers* Power BI app.

Follow these steps to modify the *embed for your customers* sample application, to embed your Power BI report.

1. Open the Power BI developer samples folder.
2. Select **Code** and then select **Download zip**.



3. Extract the downloaded ZIP and navigate to the **PowerBI-Developer-Samples-master** folder.

4. Depending on the language you want your app to use, open one of these folders:

   - .NET Core
   - .NET Framework
   - Java
   - Node JS
   - Python

   > **NOTE**
   > The *embed for your customers* sample applications only support the frameworks listed above. The *React* sample application only supports the *embed for your organization* solution.

5. Open the **Embed for your customers** folder.

- .NET Core
- .NET Framework
- Java
- Node JS
- Python

6. Open the *embed for your customers sample app* using one of these methods:

   - If you're using Visual Studio, open the **AppOwnsData.sln** file.
   - If you're using Visual Studio Code, open the **AppOwnsData** folder.

7. Open **appsettings.json**.

8. Depending on your authentication method, fill in the following parameter values:

PARAMETER	SERVICE PRINCIPAL	MASTER USER
`AuthenticationMode`	ServicePrincipal	MasterUser
`ClientId`	Your Azure AD app client ID	Your Azure AD app client ID
`TenantId`	Your Azure AD tenant ID	N/A
`PbiUsername`	N/A	Your *master user* username, see Power BI username and password
`PbiPassword`	N/A	Your *master user* password, see Power BI username and password
`ClientSecret`	Your Azure AD client secret	N/A
`WorkspaceId`	The ID of the workspace with your embedded report, see Workspace ID	The ID of the workspace with your embedded report, see Workspace ID
`ReportId`	The ID of the report you're embedding, see Report ID	The ID of the report you're embedding, see Report ID

9. Run the project by selecting the appropriate option:

   - If you're using **Visual Studio**, select **IIS Express** (play).
   - If you're using **Visual Studio Code**, select **Run > Start Debugging**.

Developing your application

After configuring and running the *embed for your customers* sample application, you can start developing your own application.

When you're ready, review the move to production requirements. You'll also need a capacity, and should review the capacity planning article to establish which SKU best suits your needs.

Next steps

Move to production

Embed for your organization

Embed paginated reports

Ask the Power BI Community

Tutorial: Embed Power BI content using a sample *embed for your organization* application

6/30/2022 • 9 minutes to read • Edit Online

Power BI embedded analytics allows you to embed Power BI content such as reports, dashboards and tiles, into your application.

In this tutorial, you'll learn how to:

- Set up your embedded environment.
- Configure an *embed for your organization* (also known as *user owns data*) sample application.

To use your application, your users will need to sign in to Power BI.

The embed for your organization solution is usually used by enterprises and big organizations, and is intended for internal users.

> **IMPORTANT**
> If you are embedding content for a national cloud, the first few steps of this tutorial are different. See Embed content for national clouds for details.

Code sample specifications

This tutorial includes instructions for configuring an *embed for your organization* sample application in one of the following frameworks:

- .NET Framework
- .NET Core
- React TypeScript

> **NOTE**
> The *.NET Core* and the *.NET Framework* samples will allow the end user to view any Power BI dashboard, report or tile they have access to in Power BI service. The *React TypeScript* sample lets you embed only one report that your end user already has access to on Power BI service.

The code samples support the following browsers:

- Microsoft Edge
- Google Chrome
- Mozilla Firefox

Prerequisites

Before you start this tutorial, verify that you have both the Power BI and code dependencies listed below:

- **Power BI dependencies**
  - Your own Azure Active Directory tenant.

- One of the following licenses:
  - Power BI Pro
  - Premium Per User (PPU)

> **NOTE**
> To move to production you'll need one of the following configurations:
>
> - All users with Pro licenses.
> - All users with PPU licenses.
> - A *P* or *EM* capacity. This configuration allows all users to have free licenses.

- **Code dependencies**
  - .NET Core
  - .NET Framework
  - React TypeScript
- .NET Core 3.1 SDK (or higher)
- An integrated development environment (IDE). We recommend using one of the following:
  - Visual Studio
  - Visual Studio Code

Method

To create an *embed for your organization* sample app, follow these steps:

1. Register an Azure AD application.
2. Create a Power BI workspace.
3. Create and publish a Power BI report.
4. Get the embedding parameter values.
5. Embed your content.

Step 1 - Register an Azure AD application

Registering your application with Azure AD allows you to establish an identity for your app.

To register your application with Azure AD, follow the instructions in Register your application.

Step 2 - Create a Power BI workspace

Power BI keeps your reports, dashboards, and tiles in a workspace. To embed these items, you'll need to create them and upload them into a workspace.

> **TIP**
> If you already have a workspace, you can skip this step.

To create a workspace, do the following:

1. Sign in to Power BI.
2. Select **Workspaces**.
3. Select **Create a workspace**.
4. Name your workspace and select **Save**.

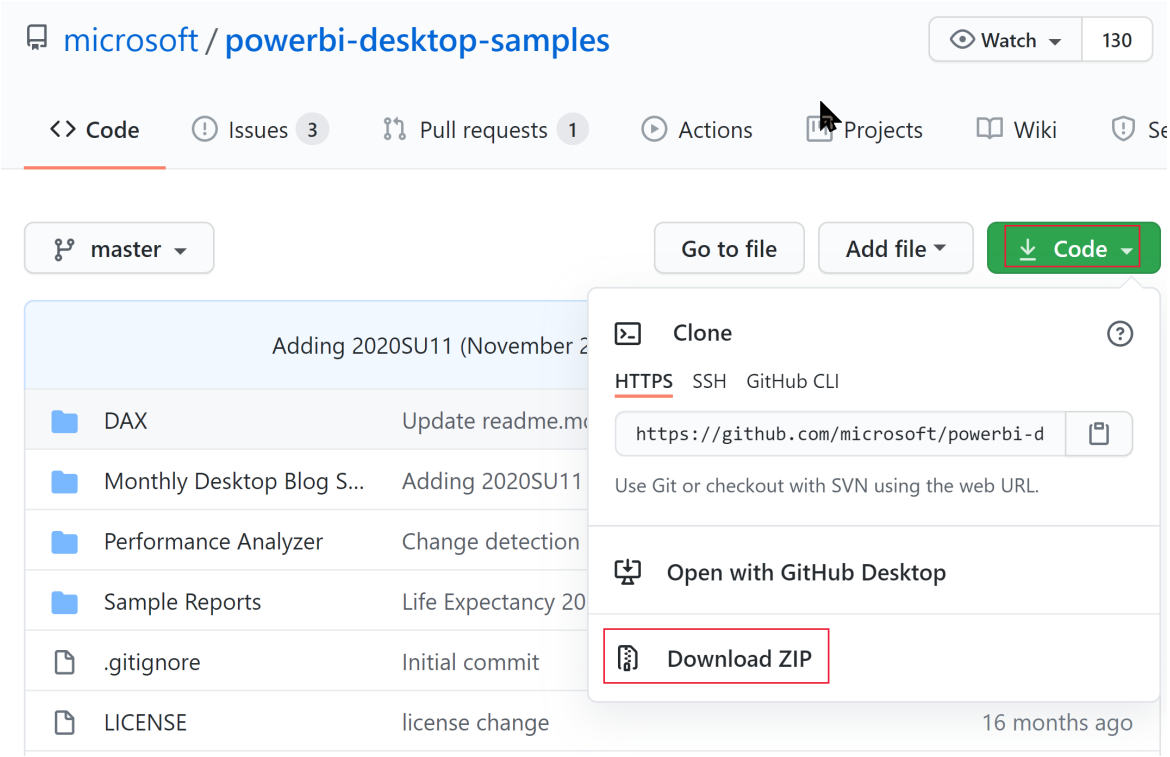
Step 3 - Create and publish a Power BI report

Your next step is to create a report and upload it to your workspace. You can create your own report using Power BI Desktop, and then publish it to your workspace. Or, you can upload a sample report to your workspace.

> **TIP**
> If you already have a workspace with a report, you can skip this step.

To download a sample report and publish it to your workspace, follow these steps:

1. Open the GitHub Power BI Desktop samples folder.
2. Select **Code** and then select **Download zip**.



3. Extract the downloaded ZIP and navigate to the **Samples Reports** folder.
4. Select a report to embed, and publish it to your workspace.

Step 4 - Get the embedding parameter values

To embed your content, you'll need to obtain a few parameter values. The parameter values you'll need depend on the language of the sample application you want to use. The table below lists which parameter values are required for each sample.

PARAMETER	.NET CORE	.NET FRAMEWORK	REACT TYPESCRIPT
Client ID	✓	✓	✓
Client secret	✓	✓	✗
Workspace ID	✗	✗	✓
Report ID	✗	✗	✓

Client ID

> **TIP**
> **Applies to:** ✓ .NET Core ✓ .NET Framework ✓ React TypeScript

To get the client ID GUID (also know as *application ID*), follow these steps:

1. Log into Microsoft Azure.
2. Search for **App registrations** and select the **App registrations** link.
3. Select the Azure AD app you're using for embedding your Power BI content.
4. From the **Overview** section, copy the **Application (client) ID** GUID.

Client secret

> **TIP**
> **Applies to:** ✓ .NET Core ✓ .NET Framework ✗ React TypeScript

To get the client secret, follow these steps:

1. Log into Microsoft Azure.
2. Search for **App registrations** and select the **App registrations** link.
3. Select the Azure AD app you're using for embedding your Power BI content.
4. Under **Manage**, select **Certificates & secrets**.
5. Under **Client secrets**, select **New client secret**.
6. In the **Add a client secret** pop-up window, provide a description for your application secret, select when the application secret expires, and select **Add**.
7. From the **Client secrets** section, copy the string in the **Value** column of the newly created application secret. The client secret value is your *client ID*.

> **NOTE**
> Make sure you copy the client secret value when it first appears. After navigating away from this page, the client secret will be hidden and you'll not be able to retrieve its value.

Workspace ID

TIP

Applies to: ✗ .NET Core ✗ .NET Framework ✓ React TypeScript

To get the workspace ID GUID, follow these steps:

1. Sign in to Power BI service.
2. Open the report you want to embed.
3. Copy the GUID from the URL. The GUID is the number between **/groups/** and **/reports/**.

Report ID

TIP

Applies to: ✗ .NET Core ✗ .NET Framework ✓ ReactTypeScript

To get the report ID GUID, follow these steps:

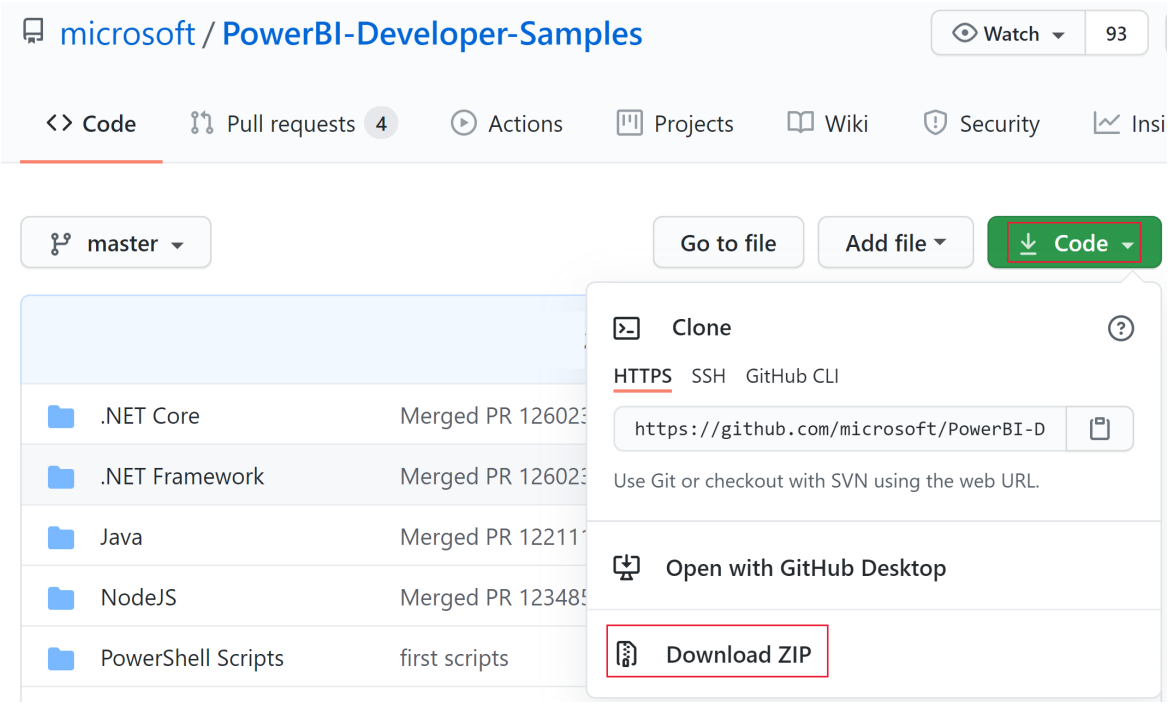
1. Sign in to Power BI service.
2. Open the report you want to embed.
3. Copy the GUID from the URL. The GUID is the number between **/reports/** and **/ReportSection**.

Step 5 - Embed your content

The Power BI embedded sample application allows you to create an *embed for your organization* Power BI app.

Follow these steps to modify the *embed for your organization* sample application, to embed your Power BI report.

1. Open the Power BI developer samples folder.
2. Select **Code** and then select **Download zip**.



3. Extract the downloaded ZIP and navigate to the **PowerBI-Developer-Samples-master** folder.

4. Depending on the language you want your application to use, open one of these folders:

   - .NET Core
   - .NET Framework
   - React-TS

   > **NOTE**
   > The *embed for your organization* sample applications only support the frameworks listed above. The *Java*, *Node JS* and *Python* sample applications, only support the *embed for your customers* solution.

- .NET Core
- .NET Framework
- React TypeScript

Configure your Azure AD app

1. Sign into the Azure portal.

2. Select **App registrations**. If you can't see this option, search for it.

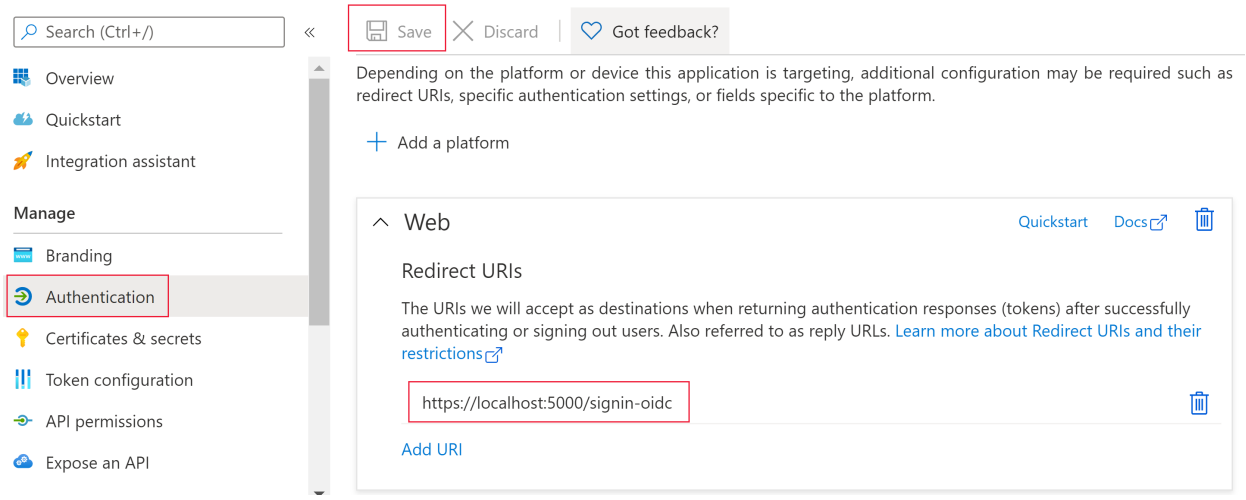
3. Open the Azure AD application you created in Step 1 - Register an Azure AD application.

4. From the *Manage* menu, select **Authentication**.

5. In *Platform configurations*, open your *Web* platform and in the **Redirect URIs** section, add `https://localhost:5000/signin-oidc`.

   > **NOTE**
   > If you don't have a **Web** platform, select **Add a platform** and in the *Configure platforms* window, select **Web**.

6. Save your changes.



Configure the sample embedding app

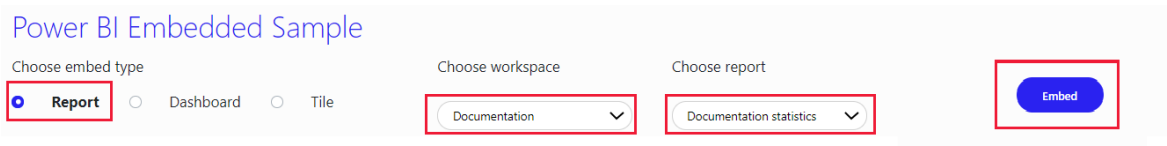
1. Open the **Embed for your organization** folder.
2. Open the *embed for your organization sample app* using one of these methods:
   - If you're using Visual Studio, open the **UserOwnsData.sln** file.
   - If you're using Visual Studio Code, open the **UserOwnsData** folder.
3. Open **appsettings.json** and fill in the following parameter values:
   - `ClientId` - Use the client ID GUID
   - `ClientSecret` - Use the client secret

Run the sample app

1. Run the project by selecting the appropriate option:
   - If you're using **Visual Studio**, select **IIS Express** (play).
   - If you're using **Visual Studio Code**, select **Run > Start Debugging**.
2. Sign into the embedding sample application.

> **NOTE**
> During your first sign in, you'll be prompted to allow Azure AD permissions for the app.

3. When the embedding sample application loads, select the Power BI content you want to embed and then select **Embed**.



Develop your application

After configuring and running the *embed for your customers* sample application, you can start developing your own application.

Update user permissions

For anyone to access the report, they need permission to access the Power BI folder the report is in. When you

grant someone permission to access a folder, the change usually takes effect only after the user logs in to the Power BI Portal. For the new permissions to take effect immediately, in the Embedded scenario, make an explicit call to the RefreshUser Permissions REST API at startup. This API call will refresh the permissions and avoid authorization failures for users with newly granted permissions.

Next steps

Embed for your customers

Embed paginated reports

Ask the Power BI Community

Tutorial: Embed Power BI content into your application for national clouds

6/30/2022 • 3 minutes to read • Edit Online

Learn how to embed analytical content within your business process applications for the national cloud. You can use the Power BI .NET SDK with the Power BI JavaScript API to embed a report, dashboard, or tile, into your web applications.

Power BI supports the following national clouds:

- U.S. Government Community Cloud (GCC)
- U.S. Government Community Cloud High (GCC High)
- U.S. Military Contractors (DoDCON)
- U.S. Military (DoD)
- Power BI for China cloud

To get started with this walkthrough, you need a **Power BI account**. If you don't have an account set up, then depending on the type of government or country you can choose the right national cloud for you. You can sign up for a U. S. government Power BI account, or a Power BI for China cloud account.

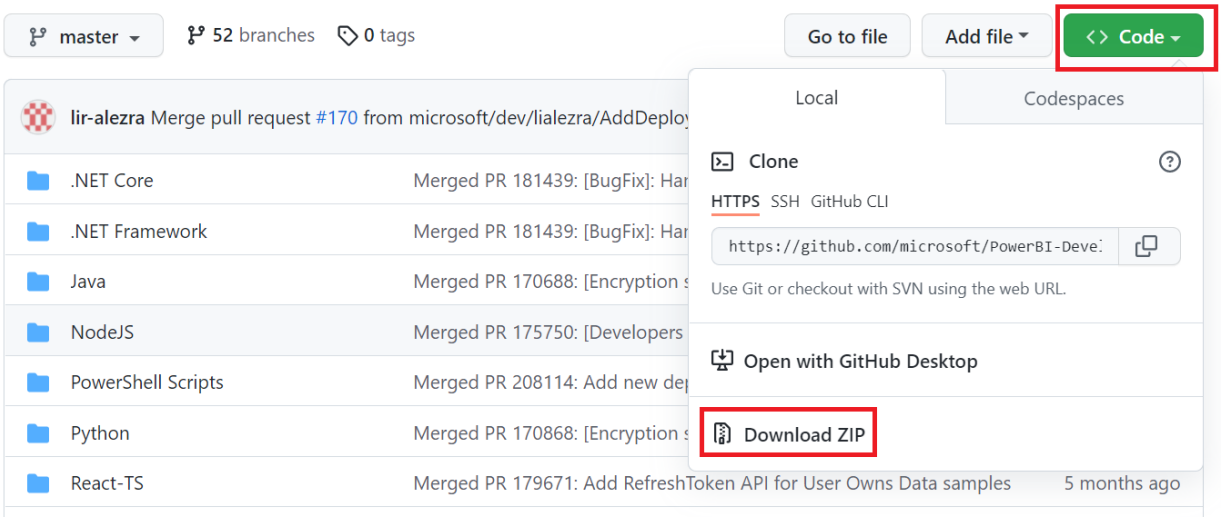
> **NOTE**
> Looking to embed a dashboard for your organization instead? See, Integrate a dashboard into an app for your organization.

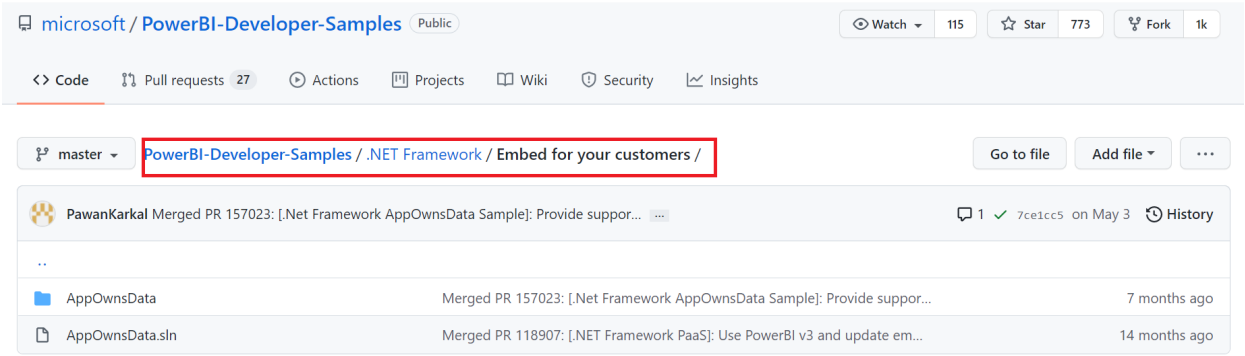
To integrate a dashboard into a web app, you use the **Power BI** API, and an Azure Active Directory (AD) authorization **access token** to get a dashboard. Then, you load the dashboard using an embed token. The **Power BI** API provides programmatic access to specific **Power BI** resources. For more information, see Power BI REST API, Power BI .NET SDK, and the Power BI JavaScript API.

Download the sample

This article shows the code used in the App Owns Data sample on GitHub. To follow along with this walkthrough, you can download the sample. We'll be using the `.NET Framework/Embed for your customers` directory.

> **IMPORTANT**
> Embedding Power BI content from a Government Community Cloud (GCC), can only be done with a Microsoft 365 SKU. Other national cloud customers can use Microsoft 365 or Azure SKUs.





- Government Community Cloud (GCC):

  1. Update applicationId (Native app applicationId), workspaceId, the username (your master user), and password in Web.config file.

  2. Add the GCC parameters in the web.config file as follows.

     ```
     <add key="authorityUrl" value="https://login.microsoftonline.com/organizations/" />
     <add key="scopeBase" value="https://analysis.usgovcloudapi.net/powerbi/api/.default" />
     <add key="urlPowerBiServiceApiRoot" value="https://api.powerbigov.us/" />
     ```

- Military Contractors (DoDCON):

  1. Update applicationId (Native app applicationId), workspaceId, the username (your master user), and password in Web.config file.

2. Add the DoDCON parameters in the web.config file as follows.

```
<add key="authorityUrl" value="https://login.microsoftonline.us/organizations/" />
<add key="scopeBase" value="https://high.analysis.usgovcloudapi.net/powerbi/api/.default" />
<add key="urlPowerBiServiceApiRoot" value="https://api.high.powerbigov.us/" />
```

- Military (DoD):

1. Update applicationId (Native app applicationId), workspaceId, the username (your master user), and password in Web.config file.

2. Add the DoDCON parameters in the web.config file as follows.

```
<add key="authorityUrl" value="https://login.microsoftonline.us/organizations/" />
<add key="scopeBase" value="https://mil.analysis.usgovcloudapi.net/powerbi/api/.default" />
<add key="urlPowerBiServiceApiRoot" value="https://api.mil.powerbigov.us/" />
```

- Power BI for China cloud parameters

1. Update applicationId (Native app applicationId), workspaceId, the username (your master user), and password in Web.config file.

2. Add the Power BI for China cloud parameters in the web.config file as follows.

```
<add key="authorityUrl" value="https://login.chinacloudapi.cn/organizations/" />
<add key="scopeBase" value="https://analysis.chinacloudapi.cn/powerbi/api/.default" />
<add key="urlPowerBiServiceApiRoot" value="https://api.powerbi.cn/" />
```

Step 1 - register an app in Azure AD

Register your application with Azure AD to make REST API calls. For more information, see Register an Azure AD app to embed Power BI content. Since there are different national cloud affiliations, there are distinct URLs to register your application.

- Government Community Cloud (GCC) - `https://app.powerbigov.us/apps`
- Military Contractors (DoDCON) - `https://app.high.powerbigov.us/apps`
- Military (DoD) - `https://app.mil.powerbigov.us/apps`
- Power BI for China cloud - `https://app.powerbi.cn/apps`

If you downloaded the Embedding for your customer sample, you would use the **applicationId** you get, so that the sample can authenticate to Azure AD. To configure the sample, change the **applicationId** in the *web.config* file.

Step 2 - get an access token from Azure AD

Within your application, you need to get an **access token**, from Azure AD, before you can make calls to the Power BI REST API. For more information, see Authenticate users and get an Azure AD access token for your Power BI app. Since there are different national cloud affiliations, there are distinct URLs to get an access token for your application.

- Government Community Cloud (GCC) - `https://login.microsoftonline.com`
- Military Contractors (DoDCON) - `https://login.microsoftonline.us`

- Military (DoD) - `https://login.microsoftonline.us`
- Power BI for China cloud - `https://login.chinacloudapi.cn`

You can see examples of these access tokens within each content item task in the **Controllers\HomeController.cs** file.

Step 3 - embed content

Now that you have an access token, you can continue embedding as you would on any other platform.

- Embed content for customers
- Embed content for your organization

Next steps

Embedding for your customers sample

Power BI JavaScript API

More questions? Try asking the Power BI Community

Tutorial: Embed a Power BI report in an application for your customers

6/30/2022 • 12 minutes to read • Edit Online

In this tutorial, you'll learn how to embed a Power BI report in a .NET 5.0 application, as part of the *embed for your customers* (also known as an *app owns data*) solution. In an *embed for your customers* solution, your app users will not need to sign in to Power BI or have a Power BI license.

In this tutorial, you'll learn how to embed:

- A Power BI report
- In an *embed for your customers* app
- Using a *service principal*
- Using .NET 5.0
- With the `Microsoft.Identity.Web` library (this library is also supported in .NET Core)

> **NOTE**
> The full solution used in this tutorial, is available from the DOTNET5-AppOwnsData-Tutorial GitHub repository.

Prerequisites

- A Power BI Pro or Premium Per User (PPU) license.
- A Power BI workspace with a report.
- Your own Azure Active Directory tenant.
- An Azure AD app.
- A .NET Core 5 model view controller (MVC) app.
- .NET Core 5 SDK (or higher).
- An integrated development environment (IDE). We recommend using one of the following:
  - Visual Studio.
  - Visual Studio Code (with the C# extension).

Resources

In this tutorial, you'll use:

- Power BI REST Reports API - Used to embed the URL and retrieve the embed token.
- Microsoft Identity Web authentication library.
- Power BI embedded analytics Client APIs - Use to embed the report.

Method

To embed Power BI content in an *embed for your customers* solution, follow these steps:

1. Configure your Azure AD app and service principal.
2. Get the embedding parameter values.
3. Add the required NuGet packages.
4. Enable server side authentication.
5. Build your app's client side.
6. Run your application.

Step 1 - Configure your Azure AD app and service principal

In this tutorial you'll use a *service principal* to authenticate you're web app against Azure AD. You'll also need an Azure AD app which will enable you to generate an Azure AD token. The *Azure AD token* enables your web app to call Power BI REST APIs and embed Power BI items such as reports, dashboards or tiles.

Follow the service principal instructions to create an Azure AD app and enable the apps service principal to work with your Power BI content.

Step 2 - Get the embedding parameter values

To embed your report, you'll need the following values:

- Domain
- Tenant ID
- Client ID
- Client secret
- Workspace ID
- Report ID

Domain and tenant ID

If you don't know what's your domain or tenant ID, see Find the Microsoft Azure AD tenant ID and primary domain name.

> **NOTE**
> To embed content for a user on a different tenant (a guest user), you'll need to adjust the `authorityUri` parameter.

Client ID

To get the client ID GUID (also know as *application ID*), follow these steps:

1. Log into Microsoft Azure.
2. Search for **App registrations** and select the **App registrations** link.
3. Select the Azure AD app you're using for embedding your Power BI content.
4. From the **Overview** section, copy the **Application (client) ID** GUID.

Client secret

To get the client secret, follow these steps:

1. Log into Microsoft Azure.
2. Search for **App registrations** and select the **App registrations** link.

3. Select the Azure AD app you're using for embedding your Power BI content.
4. Under **Manage**, select **Certificates & secrets**.
5. Under **Client secrets**, select **New client secret**.
6. In the **Add a client secret** pop-up window, provide a description for your application secret, select when the application secret expires, and select **Add**.
7. From the **Client secrets** section, copy the string in the **Value** column of the newly created application secret. The client secret value is your *client ID*.

> **NOTE**
> Make sure you copy the client secret value when it first appears. After navigating away from this page, the client secret will be hidden and you'll not be able to retrieve its value.

Workspace ID

To get the workspace ID GUID, follow these steps:

1. Sign in to Power BI service.
2. Open the report you want to embed.
3. Copy the GUID from the URL. The GUID is the number between **/groups/** and **/reports/**.

> **NOTE**
> To get the workspace ID programmatically, use the Get Groups API.

Report ID

To get the report ID GUID, follow these steps:

1. Sign in to Power BI service.
2. Open the report you want to embed.
3. Copy the GUID from the URL. The GUID is the number between **/reports/** and **/ReportSection**.

> **NOTE**
> To get the report ID programmatically, use the Get Reports In Group API.

Step 3 - Add the required NuGet packages

Before you can start, you'll need to add the `Microsoft.Identity.Web`, and `Microsoft.PowerBI.Api` NuGet packages to your app.

Add the NuGet packages listed below to your app:

- In **VS Code**, open a terminal and type in the code below.

- In **Visual studio**, navigate to *Tools* > *NuGet Package Manager* > *Package Manager Console* and type in the code below.

```
dotnet add package Microsoft.Identity.Web
dotnet add package Microsoft.Identity.Web.UI
dotnet add package Microsoft.PowerBI.Api
```

Step 4 - Enable server-side authentication

Enable server-side authentication in your app, by creating or modifying the files in the table below.

FILE	USE
Startup.cs	Initialize the `Microsoft.Identity.Web` authentication service
appsettings.json	Authentication details
PowerBiServiceApi.cs	Get the Azure AD token and embedding metadata
HomeController.cs	Pass embedding data as a model to the view

Configure your startup file to support `Microsoft.Identity.Web`

Modify the code in **Startup.cs** to properly initialize the authentication service provided by `Microsoft.Identity.Web`.

Add the code snippet below to your app's **Startup.cs** file.

> **NOTE**
>
> The code in `ConfigureServices` accomplishes several important things:
>
> 1. The call to `AddMicrosoftWebAppCallsWebApi` configures the Microsoft authentication library to acquire access tokens (Azure AD tokens).
> 2. The call to `AddInMemoryTokenCaches` configures a token cache that the Microsoft authentication library will use to cache access tokens and refresh tokens behind the scenes
> 3. The call to `services.AddScoped(typeof(PowerBiServiceApi))` configures the `PowerBiServiceApi` class as a service class that can be added to other classes using dependency injection.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.OpenIdConnect;
using Microsoft.AspNetCore.Authorization;
using Microsoft.Identity.Web;
using Microsoft.Identity.Web.UI;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc.Authorization;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using AppOwnsData.Services;
```

```
namespace AppOwnsData
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services) {

    services.AddMicrosoftIdentityWebAppAuthentication(Configuration)
            .EnableTokenAcquisitionToCallDownstreamApi()
            .AddInMemoryTokenCaches();

            services.AddScoped(typeof(PowerBiServiceApi));

            services.AddControllersWithViews(options => {
                var policy = new AuthorizationPolicyBuilder()
                    .RequireAuthenticatedUser()
                    .Build();
                options.Filters.Add(new AuthorizeFilter(policy));
            });
            services.AddRazorPages()
                    .AddMicrosoftIdentityUI();
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
                // The default HSTS value is 30 days. You may want to change this for production scenarios,
see https://aka.ms/aspnetcore-hsts.
                app.UseHsts();
            }
            app.UseHttpsRedirection();
            app.UseStaticFiles();

            app.UseRouting();

            app.UseAuthentication();
            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller=Home}/{action=Index}/{id?}");
                endpoints.MapRazorPages();
            });
        }
    }
}
```

Create an authentication details file

In this tutorial, the `appsettings.json` file contains sensitive information such as *client ID* and *client secret*. For security reasons, it's not recommended to keep this information in the settings file. When embedding in your

application, consider a more secure method such as Azure Key Vault for keeping this information.

1. In your project, create a new file and call it **appsettings.json**.

2. Add the following code to **appsettings.json**:

```
{
 "AzureAd": {
   "Instance": "https://login.microsoftonline.com/",
   "Domain": "yourtenant.onMicrosoft.com",
   "TenantId": "",
   "ClientId": "",
   "ClientSecret": "",
   "CallbackPath": "/signin-oidc",
   "SignedOutCallbackPath": "/signout-callback-oidc"
 },
 "PowerBi": {
   "ServiceRootUrl": "https://api.powerbi.com/"
 },
 "Logging": {
   "LogLevel": {
     "Default": "Information",
     "Microsoft": "Warning",
     "Microsoft.Hosting.Lifetime": "Information"
   }
 },
"AllowedHosts": "*"
}
```

3. Fill in the embedding parameter values obtained from Step 2 - Get the embedding parameter values.

   - `Domain` - Domain and tenant ID
   - `TenantId` - Domain and tenant ID
   - `ClientId` - Client ID
   - `ClientSecret` - Client secret

> **NOTE**
> In the code snippet above, the `PowerBi:ServiceRootUrl` parameter is added as a custom configuration value to track the base URL to the Power BI service. When you are programming against the Power BI service in the Microsoft public cloud, the URL is https://api.powerbi.com/. However, the root URL for the Power BI service will be different in other clouds such as the government cloud. Therefore, this value is stored as a project configuration value so it is easy to change whenever required.

Get the Azure AD access token and call the Power BI service

In order to embed Power BI content (such as reports and dashboards), your app needs to get an Azure AD token. To get the token, you'll need a configuration object.

The code in this section uses the .NET Core dependency injection pattern. When your class needs to use a service, you can add a constructor parameter for that service and the .NET Core runtime takes care of passing the service instance at run time. In this case, the constructor is injecting an instance of the .NET Core configuration service using the `IConfiguration` parameter, which is used to retrieve the `PowerBi:ServiceRootUrl` configuration value from **appsettings.json**. The `ITokenAcquisition` parameter, which is named `tokenAcquisition` holds a reference to the Microsoft authentication service provided by the `Microsoft.Identity.Web` library and will be used to acquire access tokens from Azure AD.

The `RequiredScopes` field holds a string array containing a set of delegated permissions supported by the Power BI service API. When your application calls across the network to acquire an Azure AD token, it will pass this set

of delegated permissions so that Azure AD can include them in the access token it returns.

> **NOTE**
>
> Verify that your *Azure AD app* is configured with the scopes required by your web app. For more information, see Change your Azure AD app's permissions.

1. In your app's project, create a new folder titled **Services**.
2. In the **Services** folder, create a new file titled **PowerBiServiceApi.cs**.
3. Add the following code to **PowerBiServiceApi.cs**.

```
using System;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.Extensions.Configuration;
using Microsoft.Identity.Web;
using Microsoft.Rest;
using Microsoft.PowerBI.Api;
using Microsoft.PowerBI.Api.Models;
using Newtonsoft.Json;

namespace AppOwnsData.Services {

    // A view model class to pass the data needed to embed a single report.
    public class EmbeddedReportViewModel {
        public string Id;
        public string Name;
        public string EmbedUrl;
        public string Token;
    }

    public class PowerBiServiceApi {

        private ITokenAcquisition tokenAcquisition { get; }
        private string urlPowerBiServiceApiRoot { get; }

        public PowerBiServiceApi(IConfiguration configuration, ITokenAcquisition tokenAcquisition) {
            this.urlPowerBiServiceApiRoot = configuration["PowerBi:ServiceRootUrl"];
            this.tokenAcquisition = tokenAcquisition;
        }

        public const string powerbiApiDefaultScope =
"https://analysis.windows.net/powerbi/api/.default";

        // A method to get the Azure AD token (also known as 'access token')
        public string GetAccessToken() {
            return this.tokenAcquisition.GetAccessTokenForAppAsync(powerbiApiDefaultScope).Result;
        }

        public PowerBIClient GetPowerBiClient() {
            var tokenCredentials = new TokenCredentials(GetAccessToken(), "Bearer");
            return new PowerBIClient(new Uri(urlPowerBiServiceApiRoot), tokenCredentials);
        }

        public async Task<EmbeddedReportViewModel> GetReport(Guid WorkspaceId, Guid ReportId) {

            PowerBIClient pbiClient = GetPowerBiClient();

            // Call the Power BI service API to get the embedding data
            var report = await pbiClient.Reports.GetReportInGroupAsync(WorkspaceId, ReportId);

            // Generate a read-only embed token for the report
            var datasetId = report.DatasetId;
            var tokenRequest = new GenerateTokenRequest(TokenAccessLevel.View, datasetId);
```

```
            var tokenRequest = new GenerateTokenRequest(TokenAccessLevel.View, datasetId);
            var embedTokenResponse = await pbiClient.Reports.GenerateTokenAsync(WorkspaceId,
ReportId, tokenRequest);
            var embedToken = embedTokenResponse.Token;

            // Return the report embedded data to caller
            return new EmbeddedReportViewModel {
                Id = report.Id.ToString(),
                EmbedUrl = report.EmbedUrl,
                Name = report.Name,
                Token = embedToken
            };
        }

    }
}
```

Modify the HomeController.cs file

In this code example, you'll use dependency injection. As you registered the `PowerBiServiceApi` class as a service by calling `services.AddScoped` in the `ConfigureServices` method. You can add a `PowerBiServiceApi` parameter to the constructor, and the .NET Core runtime will take care of creating a `PowerBiServiceApi` instance and passing it to the constructor.

From the **Controllers** folder, open the **HomeController.cs** file and add to it the following code snippet:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using AppOwnsData.Models;
using AppOwnsData.Services;

namespace AppOwnsData.Controllers
{
    [Authorize]
    public class HomeController : Controller
    {
        private PowerBiServiceApi powerBiServiceApi;

        public HomeController(PowerBiServiceApi powerBiServiceApi)
        {
            this.powerBiServiceApi = powerBiServiceApi;
        }

        [AllowAnonymous]
        public IActionResult Index()
        {
            return View();
        }

        public async Task<IActionResult> Embed() {

            // Replace these two GUIDs with the workspace ID and report ID you recorded earlier
            Guid workspaceId = new Guid("11111111-1111-1111-1111-111111111111");
            Guid reportId = new Guid("22222222-2222-2222-2222-222222222222");

            var viewModel = await powerBiServiceApi.GetReport(workspaceId, reportId);

            return View(viewModel);
        }

        [AllowAnonymous]
        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier
});
        }
    }
}
```

Step 5 - Build your app's client side

For client-side implementation, you'll need to create or modify the files in the table below

FILE	USE
embed.js	Contains the client-side JavaScript code
Embed.cshtml	Contains your app's document object model (DOM), and a DIV for embedding the report

Create a container for your embedded report

Create the **Embed.cshtml** file, which has a `div` element used as a container for your embedded report, and three scripts.

1. In the **View** > **Home** folder, create a file called **Embed.cshtml**.
2. Add the following code snippet to the **Embed.cshtml** file.

```
@model AppOwnsData.Services.EmbeddedReportViewModel;

<div id="embed-container" style="height:800px;"></div>

@section Scripts {

    <!-- powerbi.min.js is the JavaScript file that loads the client-side Power BI JavaScript API
library.
    Make sure that you're working with the latest library version.
    You can check the latest library available in https://cdnjs.com/libraries/powerbi-client -->
    <script src="https://cdn.jsdelivr.net/npm/powerbi-client@2.18.0/dist/powerbi.min.js"></script>

    <!-- This script creates a JavaScript object named viewModel which is accessible to the
JavaScript code in embed.js. -->
    <script>
        var viewModel = {
            reportId: "@Model.Id",
            embedUrl: "@Model.EmbedUrl",
            token: "@Model.Token"
        };
    </script>

    <!-- This script specifies the location of the embed.js file -->
    <script src="~/js/embed.js"></script>
}
```

Add client-side JavaScript to embed your report

To embed Power BI content, you need to create a configuration object. To learn more about creating the configuration object, see Embed a report.

In this tutorial, you'll create a JavaScript file named **embed.js** with a configuration object for embedding your report, using the variable `models`.

`models` is initialized using a call to `window['powerbi-client'].models`. The `models` variable is used to set configuration values such as `models.Permissions.All`, `models.TokenType.Aad` and `models.ViewMode.View`.

The `powerbi.embed` function uses the `models` configuration object to embed your report.

1. In the **wwwroot** > **js** folder, create a file called **embed.js**.
2. Add the following code snippet to the **embed.js** file.

```
$(function () {

    // 1 - Get DOM object for div that is report container
    var reportContainer = document.getElementById("embed-container");

    // 2 - Get report embedding data from view model
    var reportId = window.viewModel.reportId;
    var embedUrl = window.viewModel.embedUrl;
    var token = window.viewModel.token

    // 3 - Embed report using the Power BI JavaScript API.
    var models = window['powerbi-client'].models;

    var config = {
      type: 'report',
      id: reportId,
      embedUrl: embedUrl,
      accessToken: token,
      permissions: models.Permissions.All,
      tokenType: models.TokenType.Embed,
      viewMode: models.ViewMode.View,
      settings: {
        panes: {
          filters: { expanded: false, visible: true },
          pageNavigation: { visible: false }
        }
      }
    };

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

    // 4 - Add logic to resize embed container on window resize event
    var heightBuffer = 12;
    var newHeight = $(window).height() - ($("header").height() + heightBuffer);
    $("#embed-container").height(newHeight);
    $(window).resize(function () {
      var newHeight = $(window).height() - ($("header").height() + heightBuffer);
      $("#embed-container").height(newHeight);
    });

  });
```

Step 6 - Run your application

After you've made all the adjustments listed in this tutorial, you're ready to run your application. Execute your application and experiment with the way your Power BI report is embedded. You can use the Power BI embedded analytics Client APIs to enhance your app using client side APIs.

When your app is ready, you can move your embedded app to production.

Next steps

Embedded analytics application tokens

Move your embedded app to production

Capacity and SKUs in Power BI embedded analytics

Capacity planning in Power BI embedded analytics

Tutorial: Embed a Power BI report in an application for your organization

6/30/2022 • 12 minutes to read • Edit Online

In this tutorial, you'll learn how to embed a Power BI report in a .NET 5.0 application, as part of the *embed for your organization* (also known as *user owns data*) solution. In an *embed for your organization* solution, your app users need to authenticate against Power BI with their own credentials.

In this tutorial, you'll learn how to embed:

- A Power BI report
- In an *embed for your organization* app
- Using .NET 5.0
- With the `Microsoft.Identity.Web` library (this library is also supported in .NET Core)

> **NOTE**
> The full solution used in this tutorial, is available from the DOTNET5-UserOwnsData-Tutorial GitHub repository.

Prerequisites

- A Power BI Pro or Premium Per User (PPU) license.

  > **NOTE**
  > The *embed for your organization* solution, is not supported on capacities based on *A* SKUs. An *A* SKU can only be used for the *embed for your customers* solution.

- A Power BI workspace with a report.
- Your own Azure Active Directory tenant.
- An Azure AD app.
- A .NET Core 5 model view controller (MVC) app.
- .NET Core 5 SDK (or higher).
- An integrated development environment (IDE). We recommend using one of the following environments:
  - Visual Studio.
  - Visual Studio Code (with the C# extension).

Resources

In this tutorial, you'll use:

- Power BI REST Reports API - to embed the URL and retrieve the embed token.
- Microsoft Identity Web authentication library.
- Power BI embedded analytics Client APIs - to embed the report.

Method

To embed Power BI content in an *embed for your organization* solution, follow these steps:

1. Configure your Azure AD app.
2. Get the embedding parameter values.
3. Add the required NuGet packages
4. Enable server side authentication.
5. Build your app's client side.
6. Run your application

Step 1 - Configure your Azure AD app

When your web app calls Power BI, it needs an Azure AD token to call Power BI REST APIs and embed Power BI items such as reports, dashboards or tiles.

If you don't have an Azure AD app, create one using the instructions in Register an Azure AD application to use with Power BI.

To configure your Azure AD app, follow the instructions in Configure your Azure AD app.

Step 2 - Get the embedding parameter values

To embed your report, you'll need the following values:

- Domain
- Tenant ID
- Client ID
- Client secret
- Workspace ID
- Report ID

Domain and tenant ID

If you don't know your domain or tenant ID, see Find the Microsoft Azure AD tenant ID and primary domain name.

> **NOTE**
> To embed content for a user on a different tenant (a guest user), you need to adjust the `authorityUri` parameter.

Client ID

To get the client ID GUID (also know as *application ID*), follow these steps:

1. Log into Microsoft Azure.
2. Search for **App registrations** and select the **App registrations** link.
3. Select the Azure AD app you're using for embedding your Power BI content.
4. From the **Overview** section, copy the **Application (client) ID** GUID.

Client secret

To get the client secret, follow these steps:

1. Log into Microsoft Azure.

2. Search for **App registrations** and select the **App registrations** link.
3. Select the Azure AD app you're using for embedding your Power BI content.
4. Under **Manage**, select **Certificates & secrets**.
5. Under **Client secrets**, select **New client secret**.
6. In the **Add a client secret** pop-up window, provide a description for your application secret, select when the application secret expires, and select **Add**.
7. From the **Client secrets** section, copy the string in the **Value** column of the newly created application secret. The client secret value is your *client ID*.

> **NOTE**
> Make sure you copy the client secret value when it first appears. After navigating away from this page, the client secret will be hidden and you'll not be able to retrieve its value.

Workspace ID

To get the workspace ID GUID, follow these steps:

1. Sign in to Power BI service.
2. Open the report you want to embed.
3. Copy the GUID from the URL. The GUID is the number between **/groups/** and **/reports/**.

> **NOTE**
> To get the workspace ID programmatically, use the Get Groups API.

Report ID

To get the report ID GUID, follow these steps:

1. Sign in to Power BI service.
2. Open the report you want to embed.
3. Copy the GUID from the URL. The GUID is the number between **/reports/** and **/ReportSection**.

> **NOTE**
> To get the report ID programmatically, use the Get Reports In Group API.

Step 3 - Add the required NuGet packages

Before you start, you need to add the `Microsoft.Identity.Web`, and `Microsoft.PowerBI.Api` NuGet packages to your app.

Add the NuGet packages listed below to your app:

- In **VS Code**, open a terminal and type in the code below.
- In **Visual studio**, navigate to *Tools* > *NuGet Package Manager* > *Package Manager Console* and type in the code below.

```
dotnet add package Microsoft.Identity.Web -v 0.3.0-preview
dotnet add package Microsoft.Identity.Web.UI -v 0.3.0-preview
dotnet add package Microsoft.PowerBI.Api
```

If your app previously used `Microsoft.AspNetCore` to authenticate, remove this package from your project by typing:

```
dotnet remove package Microsoft.AspNetCore.Authentication.AzureAD.UI
```

Step 4 - Enable server-side authentication

Enable server-side authentication in your app, by creating or modifying the files in the table below.

FILE	USE
Startup.cs	Initialize the `Microsoft.Identity.Web` authentication service
appsettings.json	Authentication details
PowerBiServiceApi.cs	Get the Azure AD token and embedding metadata
HomeController.cs	Pass embedding data as a model to the view

Configure your startup file to support `Microsoft.Identity.Web`

Modify the code in **Startup.cs** to properly initialize the authentication service provided by `Microsoft.Identity.Web`.

Add the code snippet below to your app's **Startup.cs** file.

> **NOTE**
>
> The code in `ConfigureServices` accomplishes several important things:
>
> 1. The call to `AddMicrosoftWebAppCallsWebApi` configures the Microsoft authentication library to acquire access tokens (Azure AD tokens).
> 2. The call to `AddInMemoryTokenCaches` configures a token cache that the Microsoft authentication library will use to cache access tokens and refresh tokens behind the scenes
> 3. The call to `services.AddScoped(typeof(PowerBiServiceApi))` configures the `PowerBiServiceApi` class as a service class that can be added to other classes using dependency injection.

```
using Microsoft.Identity.Web;
using Microsoft.Identity.Web.TokenCacheProviders;
using Microsoft.Identity.Web.TokenCacheProviders.InMemory;
using Microsoft.Identity.Web.UI;
using UserOwnsData.Services;

namespace UserOwnsData {

  public class Startup {

    public Startup (IConfiguration configuration) {
      Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the container.
    public void ConfigureServices (IServiceCollection services) {

      services
        .AddMicrosoftIdentityWebAppAuthentication(Configuration)
        .EnableTokenAcquisitionToCallDownstreamApi(PowerBiServiceApi.RequiredScopes)
        .AddInMemoryTokenCaches();

      services.AddScoped (typeof (PowerBiServiceApi));

      var mvcBuilder = services.AddControllersWithViews (options => {
        var policy = new AuthorizationPolicyBuilder()
          .RequireAuthenticatedUser()
          .Build();
        options.Filters.Add (new AuthorizeFilter (policy));
      });

      mvcBuilder.AddMicrosoftIdentityUI();

      services.AddRazorPages();

    }
  }
}
```

Create an authentication details file

In this tutorial, the `appsettings.json` file contains sensitive information such as *client ID* and *client secret*. For security reasons, we don't recommend keeping this information in the settings file. When embedding in your application, consider a more secure method such as Azure Key Vault for keeping this information.

1. In your project, create a new file and call it **appsettings.json**.
2. Add the following code to **appsettings.json**:

```
{
    "AzureAd": {
        "Instance": "https://login.microsoftonline.com/",
        "Domain": "",
        "TenantId": "",
        "ClientId": "",
        "ClientSecret": "",
        "CallbackPath": "/signin-oidc",
        "SignedOutCallbackPath": "/signout-callback-oidc"
    },
    "PowerBi": {
        "ServiceRootUrl": "https://api.powerbi.com"
    },
    "Logging": {
        "LogLevel": {
            "Default": "Information",
            "Microsoft": "Warning",
            "Microsoft.Hosting.Lifetime": "Information"
        }
    },
    "AllowedHosts": "*"
}
```

3. Fill in the embedding parameter values obtained from Step 2 - Get the embedding parameter values.

- `Domain` - Domain and tenant ID
- `TenantId` - Domain and tenant ID
- `ClientId` - Client ID
- `ClientSecret` - Client secret

> **NOTE**
>
> In the code snippet above, the `PowerBi:ServiceRootUrl` parameter is added as a custom configuration value to track the base URL to the Power BI service. When programming against the Power BI service in the Microsoft public cloud, the URL is https://api.powerbi.com/. However, the root URL for the Power BI service will be different in other clouds such as the government cloud. Therefore, this value is stored as a project configuration value so it is easy to change when required.

Get the Azure AD access token and call the Power BI service

In order to embed Power BI content (such as reports and dashboards), your app needs to get an Azure AD token. To get the token, you need a configuration object.

The code in this section uses the .NET Core dependency injection pattern. When your class needs to use a service, you can add a constructor parameter for that service and the .NET Core runtime takes care of passing the service instance at run time. In this case, the constructor is injecting an instance of the .NET Core configuration service using the `IConfiguration` parameter, which is used to retrieve the `PowerBi:ServiceRootUrl` configuration value from **appsettings.json**. The `ITokenAcquisition` parameter, which is named `tokenAcquisition` holds a reference to the Microsoft authentication service provided by the `Microsoft.Identity.Web` library and will be used to acquire access tokens from Azure AD.

The `RequiredScopes` field holds a string array containing a set of delegated permissions supported by the Power BI service API. When your application calls across the network to acquire an Azure AD token, it will pass this set of delegated permissions so that Azure AD can include them in the access token it returns.

> **NOTE**
>
> Verify that your *Azure AD app* is configured with the scopes required by your web app. For more information, see Change your Azure AD app's permissions.

1. In your app's project, create a new folder titled **Services**.
2. In the **Services** folder, create a new file titled **PowerBiServiceApi.cs**.
3. Add the following code to **PowerBiServiceApi.cs**.

```
using Microsoft.Identity.Web;
using Microsoft.PowerBI.Api;
using Microsoft.PowerBI.Api.Models;
using Microsoft.Rest;
using Newtonsoft.Json;

namespace UserOwnsData.Services {

  // A view model class to pass the data needed to embed a single report.
  public class EmbeddedReportViewModel {
     public string Id;
     public string Name;
     public string EmbedUrl;
     public string Token;
  }

public class PowerBiServiceApi {
     private ITokenAcquisition tokenAcquisition { get; }
     private string urlPowerBiServiceApiRoot { get; }

     public PowerBiServiceApi(IConfiguration configuration, ITokenAcquisition tokenAcquisition) {
         this.urlPowerBiServiceApiRoot = configuration["PowerBi:ServiceRootUrl"];
         this.tokenAcquisition = tokenAcquisition;
     }

     public static readonly string[] RequiredScopes = new string[] {
         "https://analysis.windows.net/powerbi/api/Report.Read.All"
     };

    // A method to get the Azure AD token (also known as 'access token')
     public string GetAccessToken() {
         return this.tokenAcquisition.GetAccessTokenForUserAsync(RequiredScopes).Result;
     }

     public PowerBIClient GetPowerBiClient() {
         var tokenCredentials = new TokenCredentials(GetAccessToken(), "Bearer");
         return new PowerBIClient(new Uri(urlPowerBiServiceApiRoot), tokenCredentials);
     }

     public async Task<EmbeddedReportViewModel> GetReport(Guid WorkspaceId, Guid ReportId) {
         PowerBIClient pbiClient = GetPowerBiClient();
         // Call the Power BI Service API to get embedding data
   var report = await pbiClient.Reports.GetReportInGroupAsync(WorkspaceId, ReportId);

   // Return report embedding data to caller
   return new EmbeddedReportViewModel {
    Id = report.Id.ToString(),
    EmbedUrl = report.EmbedUrl,
    Name = report.Name,
    Token = GetAccessToken()
   };
  }

 }

}
```

Modify the HomeController.cs file

In this code example, you use dependency injection. As you registered the `PowerBiServiceApi` class as a service by calling `services.AddScoped` in the `ConfigureServices` method. You can add a `PowerBiServiceApi` parameter to the constructor, and the .NET Core runtime will take care of creating a `PowerBiServiceApi` instance and passing it to the constructor.

From the **Controllers** folder, open the **HomeController.cs** file and add it to the following code snippet:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using UserOwnsData.Models;
using UserOwnsData.Services;

namespace UserOwnsData.Controllers {
    [Authorize]
    public class HomeController : Controller {

        private PowerBiServiceApi powerBiServiceApi;

        public HomeController (PowerBiServiceApi powerBiServiceApi) {
            this.powerBiServiceApi = powerBiServiceApi;
        }

        [AllowAnonymous]
        public IActionResult Index() {
            return View();
        }

        public async Task<IActionResult> Embed() {
            Guid workspaceId = new Guid("11111111-1111-1111-1111-111111111111");
            Guid reportId = new Guid("22222222-2222-2222-2222-222222222222");
            var viewModel = await powerBiServiceApi.GetReport(workspaceId, reportId);
            return View(viewModel);
        }

        [AllowAnonymous]
        [ResponseCache (Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error() {
            return View (new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
        }
    }
}
```

Step 5 - Build your app's client side

For client-side implementation, you need to create or modify the files in the table below.

FILE	USE
embed.js	Contains the client-side JavaScript code
Embed.cshtml	Contains your app's document object model (DOM), and a DIV for embedding the report

Create a container for your embedded report

Create the **Embed.cshtml** file, which has a `div` element used as a container for your embedded report, and three scripts.

1. In the **View** > **Home** folder, create a file called **Embed.cshtml**.
2. Add the following code snippet to the **Embed.cshtml** file.

```
@model UserOwnsData.Services.EmbeddedReportViewModel;

<div id="embed-container" style="height:800px;"></div>

@section Scripts {

    <!-- powerbi.min.js is the JavaScript file that loads the client-side Power BI JavaScript API
library.
    Make sure that you're working with the latest library version.
    You can check the latest library available in https://cdnjs.com/libraries/powerbi-client -->
    <script src="https://cdn.jsdelivr.net/npm/powerbi-client@2.21.0/dist/powerbi.min.js"></script>

    <!-- This script creates a JavaScript object named viewModel which is accessible to the
JavaScript code in embed.js. -->
    <script>
        var viewModel = {
            reportId: "@Model.Id",
            embedUrl: "@Model.EmbedUrl",
            token: "@Model.Token"
        };
    </script>

    <!-- This script specifies the location of the embed.js file -->
    <script src="~/js/embed.js"></script>
}
```

Add client-side JavaScript to embed your report

To embed Power BI content, you need to create a configuration object. To learn more about creating the configuration object, see Embed a report.

In this tutorial, you'll create a JavaScript file named **embed.js** with a configuration object for embedding your report, using the variable `models`.

`models` is initialized using a call to `window['powerbi-client'].models`. The `models` variable is used to set configuration values such as `models.Permissions.All`, `models.TokenType.Aad` and `models.ViewMode.View`.

The `powerbi.embed` function uses the `models` configuration object to embed your report.

1. In the **wwwroot** > **js** folder, create a file called **embed.js**.
2. Add the following code snippet to the **embed.js** file.

```
$(function(){
    // 1 - Get DOM object for div that is report container
    let reportContainer = document.getElementById("embed-container");

    // 2 - Get report embedding data from view model
    let reportId = window.viewModel.reportId;
    let embedUrl = window.viewModel.embedUrl;
    let token = window.viewModel.token

    // 3 - Embed report using the Power BI JavaScript API.
    let models = window['powerbi-client'].models;
    let config = {
        type: 'report',
        id: reportId,
        embedUrl: embedUrl,
        accessToken: token,
        permissions: models.Permissions.All,
        tokenType: models.TokenType.Aad,
        viewMode: models.ViewMode.View,
        settings: {
            panes: {
                filters: { expanded: false, visible: true },
                pageNavigation: { visible: false }
            }
        }
    };

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

    // 4 - Add logic to resize embed container on window resize event
    let heightBuffer = 12;
    let newHeight = $(window).height() - ($("header").height() + heightBuffer);
    $("#embed-container").height(newHeight);
    $(window).resize(function() {
        var newHeight = $(window).height() - ($("header").height() + heightBuffer);
        $("#embed-container").height(newHeight);
    });

});
```

Step 6 - Run your application

After you've made all the adjustments listed in this tutorial, you're ready to run your application. Execute your application and experiment with the way your Power BI report is embedded. You can use the Power BI embedded analytics Client APIs to enhance your app using client side APIs.

When your app is ready, you can move your embedded app to production.

Next steps

Embedded analytics application tokens

Move your embedded app to production

Capacity and SKUs in Power BI embedded analytics

Capacity planning in Power BI embedded analytics

Embedded analytics application tokens

6/30/2022 • 4 minutes to read • Edit Online

Consuming Power BI content (such as reports, dashboards and tiles) requires an access token. Depending on your solution, this token can be either an Azure AD token, an embed token, or both.

In the *embed for your customers* solution, your web app users are granted access to Power BI content according to the *embed token* generated by your application.

> **NOTE**
> When using the *embed for your customers* solution, you can use any authentication method to allow access to your web app.

In the *embed for your organization* solution, your web app users authenticate against Azure AD using their own credentials. They'll then have access to the Power BI content they have permission to access on Power BI service.

Azure AD token

For both *embed for your customers* and *embed for your organization* solutions, you need an Azure AD token. This token is required for all REST API operations, and it expires after an hour.

- In the *embed for your customers*, the Azure AD token is used to generate the *embed token*.
- In the *embed for your organization*, the Azure AD token is used to access Power BI.

Embed token

When you're using the *embed for your customers* solution, your web app needs to know which Power BI content its user can access. Use the embed token REST APIs, to generate an *embed token*, which specifies the following information:

- Which content your web app user can access.
- The web app user's access level (view, create, or edit).

For more information, see Considerations when generating an embed token.

Authentication flows

This section describes the different authentication flows for the *embed for your customers* and *embed for your organization* embedding solutions.

- Embed for your customers
- Embed for your organization

The *embed for your customers* solution uses a non-interactive authentication flow. Users don't sign in to Azure AD to access Power BI. Instead, your web app uses a reserved Azure AD identity to authenticate against Azure AD, and generate the *embed token*. The reserved identity can be either a *service principal* or a *master user*.

- **Service principal**

  Your web app uses the Azure AD service principal object to authenticate against Azure AD and get an

app-only Azure AD token. This *app-only* authentication method is recommended by Azure AD.

When using a *service principal*, you need to enable Power BI APIs access in the Power BI service *admin* settings. Enabling access allows your web app to access the Power BI REST APIs. To use API operations on a workspace, the *service principal* needs to be a *member* or *admin* of the workspace.

- **Master user**

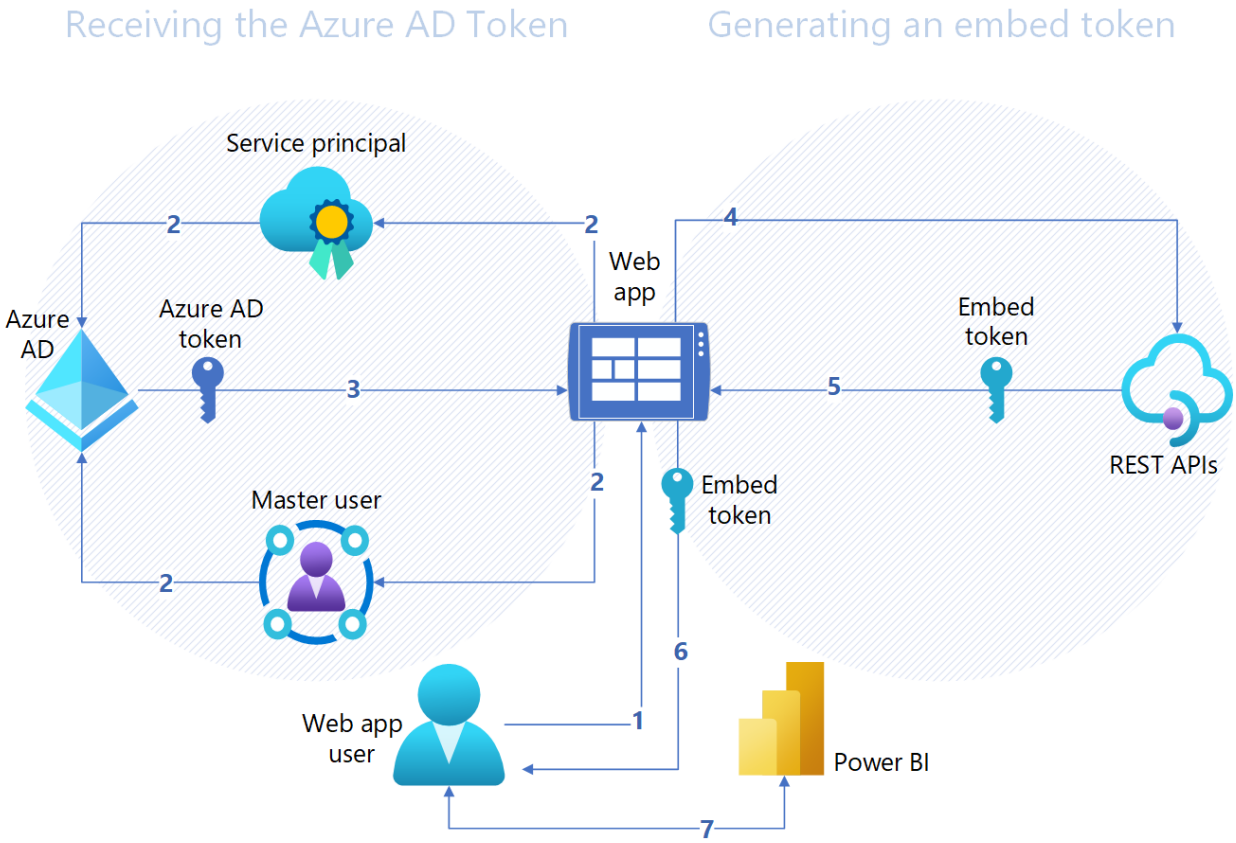
  Your web app uses a user account to authenticate against Azure AD and get the *Azure AD token*. The *master user* needs to have a Power BI Pro or a Premium Per User (PPU) license.

  When using a *master user*, you'll need to define your app's delegated permissions (also known as scopes). The *master user* or *Power BI admin* is required to grant consent for using these permissions using the Power BI REST APIs.

After successful authentication against Azure AD, your web app will generate an embed token to allow its users to access specific Power BI content.

> **NOTE**
>
> - To embed using the *embed for your customers* solution, you'll need a capacity with an A, EM, or P SKU.
> - To move to production you'll need a capacity.

The following diagram shows the authentication flow for the *embed for your customers* solution.



1. Web app user authenticates against your web app (with your authentication method).
2. Your web app uses a *service principal* or a *master user* to authenticate against Azure AD.
3. Your web app gets an *Azure AD token* from Azure AD, and uses it to access Power BI REST APIs. Access to the Power BI REST APIs is given according to your authentication method, which is either *service principal* or *master user*.

4. Your web app calls an Embed Token REST API operation, requesting the *embed token*. The *embed token* specifies which Power BI content can be embedded.
5. The REST API returns the *embed token* to your web app.
6. The web app passes the embed token to the user's web browser.
7. The web app user uses the embed token to access Power BI.

Next steps

Considerations when generating an embed token

Capacity and SKUs in Power BI embedded analytics

More questions? Try asking the Power BI Community

Generate an embed token

6/30/2022 • 5 minutes to read • Edit Online

Generate token is a REST API that lets you generate a token for embedding a Power BI item in a web app or a portal. The token is used to authorize your request against the Power BI service.

The generate token API uses a single identity (a master user or service principal) to generate a token for an individual user, depending on that user's credentials in the app (effective identity).

After successful authentication, access to the relevant data is granted.

> **NOTE**
> Generate token is only applicable when you're *embedding for your customers* (also known as the *app owns data* scenario).

You can use APIs to generate a token for the following items:

- Dashboards
- Datasets
- Generate an embed token for multiple reports, datasets, and target workspaces
- Report creation
- Reports
- Tiles

Workspace versions

Power BI has two workspace versions, a *classic* workspace, and a *new* workspace. You can learn more about the differences between these workspaces in new and classic workspace differences.

When creating an embed token, different workspaces have different considerations and require different permissions.

	CLASSIC WORKSPACE	*NEW* WORKSPACE
Considerations	• The dataset and the item must be in the same workspace • Service principal can't be used	The dataset and the item can be in two different *new* workspaces
Workspace permissions	The master user must be an admin of the workspace	The service principal or master user must be at least a member of both workspaces

> **NOTE**
> - You can't create an embed token for My workspace.
> - The word *item* refers to a Power BI item such as a dashboard, tile, Q&A or report.

Securing your data

If you are handling data from multiple customers, consider these two approaches for securing your data: *Workspace-based isolation* and *Row-level security-based isolation*. You can find a detailed comparison between them in service principal profiles and row level security.

We recommend using workspace-based isolation with profiles, but if you want to use the RLS approach, review the RLS considerations and limitations at the end of this article.

Token permissions

In the generate token APIs, the *GenerateTokenRequest* section describes the token permissions.

> **NOTE**
> The token permissions listed in this section are not applicable for the Generate token for multiple reports API.

Access Level

Use the *accessLevel* parameter to determine the user's access level.

- **View** - Grant the user viewing permissions.
- **Edit** - Grant the user viewing and editing permissions (only applies when generating an embed token for a report).

  If you're using the Generate token for multiple reports, datasets, and target workspaces API, use the *allowEdit* parameter to grant the user viewing and editing permissions.
- **Create** - Grant the user permissions to create a report (only applies when generating an embed token for creating a report).

  For report creation, you must also supply the *datasetId* parameter.

Saving a copy of the report

Use the *allowSaveAs* boolean to let users save the report as a new report. This setting is set to *false* by default.

This parameter is only applicable when generating an embed token for a report.

Row Level Security

With Row Level Security (RLS), the identity you use can be different from the identity of the service principal or master user you're using to generating the token. By using different identities you can display embedded information according to the user you're targeting. For example, in your application you can ask users to sign in, and then display a report that only contains sales information if the signed in user is a sales employee.

If you're using RLS, you can sometimes leave out the user's identity (the *EffectiveIdentity* parameter). This allows the token to have access to the entire database. This method can be used to grant access to users such as admins and managers, who have permission to view the entire dataset. However, you can't use this method in every scenario. The table below lists the different RLS types, and shows which authentication method can be used without specifying a user's identity.

The table also shows the considerations and limitation applicable to each RLS type.

RLS TYPE	CAN I GENERATE AN EMBED TOKEN WITHOUT SPECIFYING THE EFFECTIVE USER ID?	CONSIDERATIONS AND LIMITATIONS

RLS TYPE	CAN I GENERATE AN EMBED TOKEN WITHOUT SPECIFYING THE EFFECTIVE USER ID?	CONSIDERATIONS AND LIMITATIONS
Cloud Row Level Security (Cloud RLS)	✔ Master user ✘ Service principal	
RDL (paginated reports)	✘ Master user ✔ Service principal	You can't use a master user to generate an embed token for RDL.
Analysis Services (AS) on premises live connection	✔ Master user ✘ Service principal	The user generating the embed token also needs one of the following permissions: • Gateway admin permissions • Datasource impersonate permission (*ReadOverrideEffectiveIdentity*)
Analysis Services (AS) Azure live connection	✔ Master user ✘ Service principal	The identity of the user generating the embed token can't be overridden. Custom data can be used to implement dynamic RLS or secure filtering. **Note:** Service principal must provide its object ID as the effective identity (RLS username).
Single Sign On (SSO)	✔ Master user ✘ Service principal	An explicit (SSO) identity can be provided using the identity blob property in an effective identity object
SSO and cloud RLS	✔ Master user ✘ Service principal	You must provide the following: • Explicit (SSO) identity in the identity blob property in an effective identity object • Effective (RLS) identity (username)

NOTE

Service principals must always provide the following:

- An identity for any item with an RLS dataset.
- For an SSO dataset, an effective RLS identity with the contextual (SSO) identity defined.

DirectQuery for Power BI datasets

To embed Power BI report that has a dataset with a Direct Query connection to another Power BI dataset, do the following:

- In the Power BI portal, set the **XMLA endpoint** to *Read Only* or *Read Write* as described in enable read-write for a Premium capacity. You only need to do this once per capacity.
- Generate a multi-resource embed token
  - Specify all dataset IDs in the request.
  - Set the `XmlaPermissions` to *Read Only* for each dataset in the request.
  - For each Single Sign-on (SSO) enabled data source, provide the identity blob for the data source in the `DatasourceIdentity`.

Considerations and limitations

For security reasons, the lifetime of the embed token is set to the remaining lifetime of the Azure AD token used to call the `GenerateToken` API. Therefore, if you use the same Azure AD token to generate several embed tokens, the lifetime of the generated embed tokens will be shorter with each call.

Next steps

- Register an app
- Power BI Embedded for your customers
- Application and service principal objects in Azure Active Directory
- Row-level security using on-premises data gateway with service principal
- Embed Power BI content with service principal

Best practices for faster performance in Power BI embedded analytics

6/30/2022 • 4 minutes to read • Edit Online

This article provides recommendations for faster rendering of reports, dashboards, and tiles in your application.

> **NOTE**
>
> Remember that loading time mainly depends on elements relevant to the report and data itself, including visuals, the size of the data, and the complexity of the queries and measures. For more information, see the Power BI optimization guide.

Update tools and SDK packages

Keep tools and SDK packages up-to-date.

- Always use the latest version of Power BI Desktop.
- Install the latest version of the Power BI client SDK. We continuously release new enhancements, so make sure to follow up from time to time.

Embed parameters

The `powerbi.embed(element, config)` method receives an element and a config parameter. The config parameter includes fields that have performance implications.

Embed URL

Avoid generating the embed URL yourself. Instead, make sure you get the Embed URL by calling Get reports, Get dashboards, or Get tiles API. The ***config*** parameter in the URL is used for performance improvements.

Permissions

Provide **View** permissions if you don't intend to embed a report in edit mode. This way, time isn't spent initializing components that are only used in edit mode.

Filters, bookmarks, and slicers

Usually, report visuals are saved with cached data. Reports render the cached data while queries are executed. If filters, bookmarks, or slicers are provided, cached data isn't used and the visuals are rendered only after the visual query has ended.

If you embed reports with the same filters, bookmarks, and slicers, save the report with the filters, bookmarks, and slicers already applied. When you save the report this way, it will render using the cached data that includes the filters, bookmarks, and slicers, which improves performance.

Switching between reports

When embedding multiple reports to the same space, don't generate a new iFrame for each report. Instead, embed the new report in the same iFrame to overwrite the previous report. Use `powerbi.embed(element, config)` with a different config to embed the new report.

> **NOTE**
> Embedding reports using embed for your customers (also known as an 'app owns data' scenario), requires the use of an embed token with permissions to all reports and datasets. For more information, see the generate token API.

Multiple visuals

When embedding several visuals from the same report, don't generate a new iFrame for each visual. Use a single iFrame to render the report with the specified visuals.

When embedding multiple visuals into a single iFrame, consider the following:

- Power BI uses iFrames to embed a report. Sometimes you might want to add more content between the visuals (for example, text or graphics that don't come from the report). In that case, you may need a different iFrame to render different visuals. For best performance, try and arrange the visuals so that you use the fewest iFrames possible. To reduce the number of iFrames, please consider using the custom-layout feature.
- If you have visuals from different reports or different datasets, consider joining the datasets and creating a new report so that you can include all the visuals in the same iFrame.
- Another alternative, if you have non-contiguous regions, or data from multiple datasets, is to create a dashboard and pin the visuals to it. This allows you to:
  - Embed the individual tiles into non-contiguous iFrames. Dashboard tiles are lighter than reports and load faster.
  - Embed the entire dashboard into one iFrame. This allows you to have visuals from different reports or datasets in one iFrame without creating a new report.

  Keep in mind, however, that dashboard tiles aren't interactive and don't refresh with the same frequency as visuals.

Query caching

Organizations with Power BI Premium capacity or Power BI Embedded capacity can take advantage of query caching to speed up reports associated with a dataset.

Learn more about query caching in Power BI.

Preload

Use `powerbi.preload()` to improve the end-user performance. The method `powerbi.preload()` downloads JavaScript, css files, and other items, which are used later to embed a report.

Call `powerbi.preload()` if you're not embedding the report immediately. For example, if the embedded Power BI content doesn't appear in the home page, use `powerbi.preload()` to download and cache the items that are used for embedding the content.

Bootstrapping the iFrame

> **NOTE**
> Power BI client SDK version 2.9 is required to bootstrap the iFrame.

`powerbi.bootstrap(element, config)` allows you to start embedding before all required parameters are available.

The bootstrap API prepares and initializes the iFrame. When using the bootstrap API, it's still required to call `powerbi.embed(element, config)` on the same HTML element.

For example, one of the use cases for this feature, is to run the iFrame bootstrap and the back-end calls for embedding, in parallel.

> **TIP**
> Use the bootstrap API when it's possible to generate the iFrame before it's visible to the end user.

Learn more about iFrame bootstrap.

Measure performance

Performance events

To measure embedded performance, you may use two events:

1. Loaded event: The time until the report is initialized (the Power BI logo will disappear when the load is finished).
2. Rendered event: The time until the report is fully rendered, using the actual data. The rendered event is fired each time the report is re-rendered (for example, after applying filters). To measure a report, make sure you do the calculations on the first raised event.

Cached data is rendered when available but no other event is generated.

Learn more about event handling.

Performance Analyzer

To examine the performance of the report elements, you might use the Performance Analyzer in Power BI Desktop. The Performance Analyzer will allow you to see and record logs that measure how each of your report elements performs.

Learn more about Performance Analyzer.

> **NOTE**
> Always remember to compare the embedded report performance to the performance on powerbi.com. This might help you understand the origin of your performance issues

Next steps

- Power BI optimization guide
- How to troubleshoot Power BI embedded analytics issues
- Power BI embedded analytics FAQ

Service principal profiles for multi-customer apps in Power BI Embedded

6/30/2022 • 13 minutes to read • Edit Online

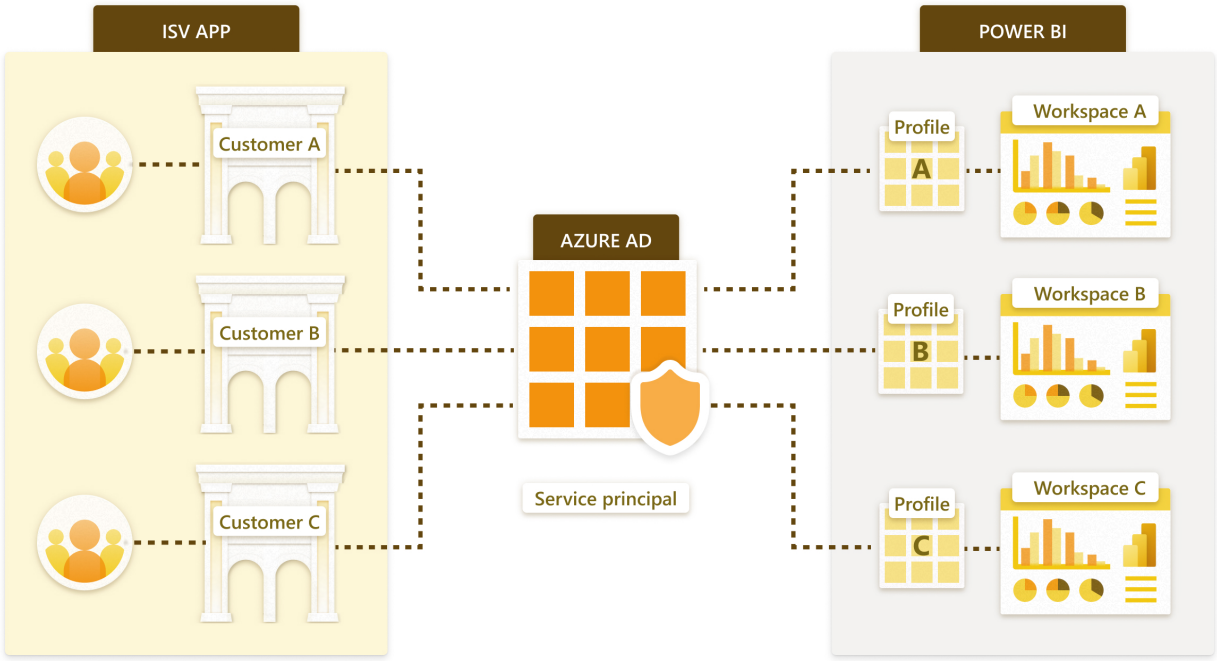
This article explains how an ISV or other Power BI Embedded app owner with many customers can use service principal profiles to map and manage each customer's data as part of their Power BI *embed for your customers* solution. Service principal profiles allow the ISV to build scalable applications that enable better customer data isolation and establish tighter security boundaries between customers. This article discusses the advantages and the limitations of this solution.

> **NOTE**
>
> For the sake of simplicity, all ISVs and Power BI Embedded Enterprise application owners will be referred to as ISVs in this article.

A *service principal profile* is a profile created by a service principal. The ISV application calls the Power BI APIs using a service principal profile, as explained in this article.

The ISV application service principal creates a different Power BI profile for each customer. When a customer visits the ISV app, the app uses the corresponding profile to generate an embed token that will be used to render a report in the browser.

Using service principal profiles enables the ISV application to host multiple customers on a single Power BI tenant. Each profile represents one customer in Power BI. In other words, each profile creates and manages Power BI content for one specific customer's data.



> **NOTE**
>
> This article is aimed at organizations that want to set up a multi-customer app using service principal profiles. If your organization already has an app that supports multiple customers from a single Power BI tenant, and you want to migrate to the service principal profile model, see Migrate multi-customer applications to the service principal profiles model.

Setting up your Power BI content involves the following steps:

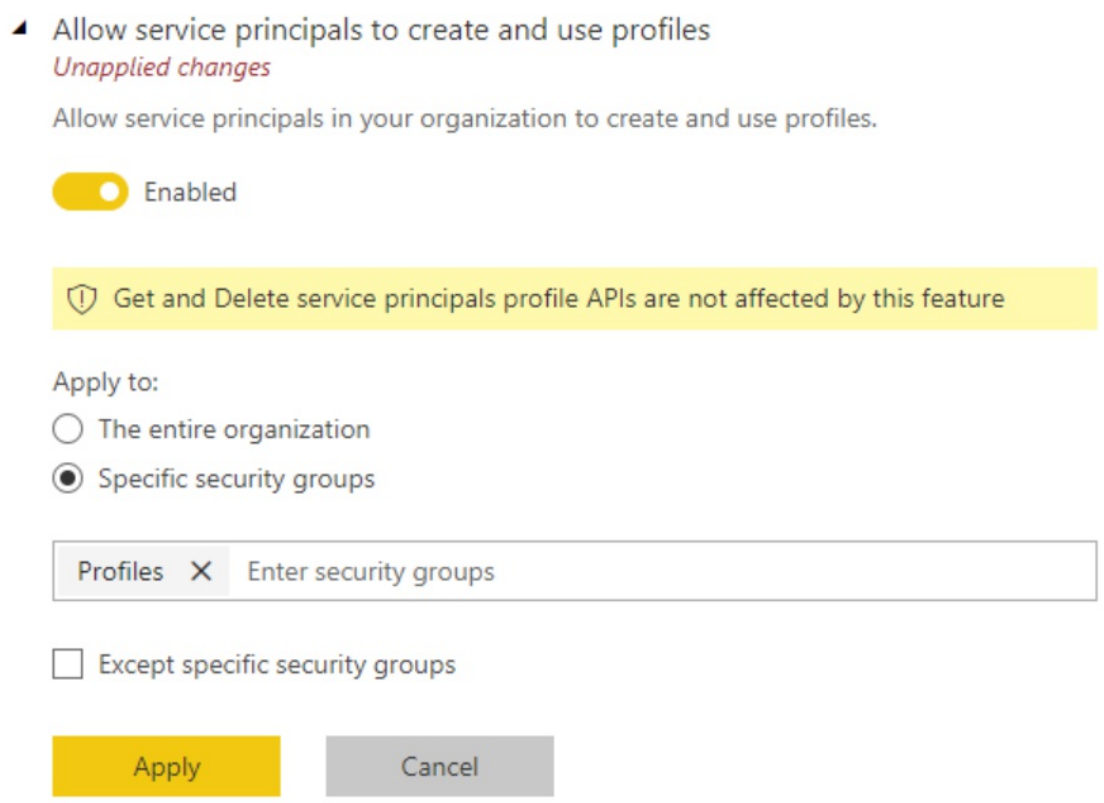
- Create a profile
- Set the profile permissions
- Create a workspace for each customer
- Import reports and datasets into the workspace
- Set the dataset connection details to connect to the customer's data
- Remove permissions from the service principal (optional, but helps with scalability)
- Embed a report into the application

All the above steps can be fully automated using Power BI REST APIs.

Prerequisites

Before you can create service principal profiles, you need to:

- Set up the service principal by following the *first three steps* of Embed Power BI content with service principal.
- From a Power BI tenant admin account, enable creating profiles in the tenant.



Create a profile

Profiles can be created, updated, and deleted using Profiles REST API.

For example, to create a profile:

```
POST https://api.powerbi.com/v1.0/myorg/profiles HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJK…UUPA
Content-Type: application/json; charset=utf-8

{"displayName":"ContosoProfile"}
```

A service principal can also call GET Profiles REST API to get a list of its profiles. For example:

```
GET https://api.powerbi.com/v1.0/myorg/profiles HTTP/1.1
Authorization: Bearer eyJ0eXA…UUPA
```

Profile permissions

Any API that grants a user permission to Power BI items can also grant a profile permission to Power BI items. For example, Add Group User API can be used to grant a profile permission to a workspace.

The following points are important to understand when using profiles:

- A profile belongs to the service principal that created it, and can only be used by that service principal.
- A profile owner can't be changed after creation.
- A profile isn't a standalone identity. It needs the service principal Azure AD token to call Power BI REST APIs.

ISV applications call Power BI REST APIs by providing the service principal Azure AD token in the *Authorization* header, and the profile ID in the *X-PowerBI-Profile-Id* header. For example:

```
GET https://api.powerbi.com/v1.0/myorg/groups HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUz.....SXBwasfg
X-PowerBI-Profile-Id: 5f1aa5ed-5983-46b5-bd05-999581d361a5
```

Create a workspace

Power BI workspaces are used to host Power BI items such as reports and datasets.

Each profile needs to:

- Create at least one Power BI workspace

  ```
  POST https://api.powerbi.com/v1.0/myorg/groups HTTP/1.1
  Authorization: Bearer eyJ0eXA…ZUiIsg
  Content-Type: application/json; charset=utf-8
  X-PowerBI-Profile-Id: a4df5235-6f18-4141-9e99-0c3512f41306

  {
    "name": "ContosoWorkspace"
  }
  ```

- Grant access permissions to the workspace
- Assign the workspace to a capacity

```
POST https://api.powerbi.com/v1.0/myorg/groups/f313e682-c86b-422c-a3e2-b1a05426c4a3/AssignToCapacity
HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJK…wNkZUiIsg
Content-Type: application/json; charset=utf-8
X-PowerBI-Profile-Id: a4df5235-6f18-4141-9e99-0c3512f41306

{
  "capacityId": "13f73071-d6d3-4d44-b9de-34590f3e3cf6"
}
```

Read more about Power BI workspaces.

Import reports and datasets

Use Power BI Desktop to prepare reports that are connected to the customer's data or sample data. Then, you can use the Import API to import the content into the created workspace.

```
POST https://api.powerbi.com/v1.0/myorg/groups/f313e682-c86b-422c-a3e2-b1a05426c4a3/imports?
datasetDisplayName=ContosoSales HTTP/1.1
Authorization: Bearer eyJ...kZUiIsg
Content-Type: multipart/form-data; boundary="8b071895-b380-4769-9c62-7e586d717ed7"
X-PowerBI-Profile-Id: a4df5235-6f18-4141-9e99-0c3512f41306
Fiddler-Encoding: base64

LS04YjA3MTg5NS1iMzgwLTQ3...Tg2ZDcxN2VkNy0tDQo=
```

Use dataset parameters to create a dataset that can connect to different customers' data sources. Then, use the *Update parameters* API to define which customers' data the dataset connects to.

Set the dataset connection

ISVs usually store their customers' data in one of two ways:

- A separate database for each customer
- A single multi-customer database

In either case, you should end up with single-customer datasets (one dataset per customer) in Power BI.

A separate database for each customer

If the ISV application has a separate database for each customer, create single-customer datasets in Power BI. Provide each dataset with connection details that point to its matching database. Use one of the following methods to avoid creating multiple identical reports with different connection details:

- **Dataset parameters:** Create a dataset with parameters in the connection details (such as SQL server name, SQL database name). Then, import a report into a customer's workspace and change the parameters to match the customer's database details.
- **Update Dataseource API:** Create a .pbix that points to a data source with sample content. Then, import the .pbix into a customer's workspace and change the connection details using the Update Datasource API.

A single multi-customer database

If the ISV application uses one database for all its customers, separate the customers into different datasets in Power BI as follows:

Create a report using parameters that only retrieve the relevant customer's data. Then, import a report into a customer's workspace and change the parameters to retrieve the relevant customer's data only.

Embed a report

After the setup is complete, you can embed customer reports and other items into your application using an embed token.

When a customer visits your application, use the corresponding profile to call the GenerateToken API. Use the generated embed token to embed a report or other items in the customer's browser.

To generate an embed token:

```
POST https://api.powerbi.com/v1.0/myorg/GenerateToken HTTP/1.1
Authorization: Bearer eyJ0eXAiOi…kZUiIsg
Content-Type: application/json; charset=utf-8
X-PowerBI-Profile-Id: a4df5235-6f18-4141-9e99-0c3512f41306

{
  "datasets": [
    {
      "id": "3b1c8f74-fbbe-45e0-bf9d-19fce69262e3"
    }
  ],
  "reports": [
    {
      "id": "3d474b89-f2d5-43a2-a436-d24a6eb2dc8f"
    }
  ]
}
```

Design aspects

Before setting up a profile-based multi-customer solution, you should be aware of the following issues:

- Scalability
- Automation & operational complexity
- Multi-Geo needs
- Cost efficiency
- Row level security

Scalability

By separating the data into separate datasets for each customer, you minimize the need for large datasets. When the capacity gets overloaded, it can evict unused datasets to free memory for active datasets. This optimization is impossible for a single large dataset. By using multiple datasets, you can also separate tenants into multiple Power BI capacities if necessary.

Without profiles, a service principal is limited to 1,000 workspaces. To overcome this limit, a service principal can create multiple profiles, where each profile can create up to 1,000 workspaces. With multiple profiles, the ISV app can isolate each customer's content using distinct logical containers.

Once a service principal profile has access to a workspace, its parent service principal's access to the workspace can be removed to avoid scalability problems.

Even with these advantages, you should consider the potential scale of your application. For example, the number of workspace items a profile can access is limited. Today, a profile has the same limits as a regular user. If the ISV application allows end users to save a personalized copy of their embedded reports, a customer's profile will have access to all the created reports of all its users. This model can eventually exceed the limit.

Automation and operational complexity

With Power BI profile-based separation, you might need to manage hundreds or thousands of items. Therefore,

it's important to define the processes that frequently happen in your application lifecycle management, and ensure you have the right set of tools to do these operations at scale. Some of these operations include:

- Adding a new customer
- Updating a report for some or all customers
- Updating the dataset schema for some or all customers
- Unplanned customizations for specific customers
- Frequency of dataset refreshes

For example, creating a profile and a workspace for a new customer is a common task, which can be fully automated with the Power BI REST API.

Multi-Geo needs

Multi-Geo support for Power BI Embedded means that ISVs and organizations that build applications using Power BI Embedded to embed analytics into their apps can now deploy their data in different regions around the world. To support different customers in different regions, assign the customer's workspace to a capacity in the desired region. This task is a simple operation that doesn't involve extra cost. However, if you have customers that need data from multiple regions, the customer profile should duplicate all items into multiple workspaces that are assigned to different regional capacities. This duplication may increase both cost and management complexity.

For compliance reasons, you may still want to create multiple Power BI tenants in different regions. Read more about multi-geo.

Cost efficiency

Application developers using Power BI Embedded need to purchase a Power BI Embedded capacity. The profile-based separation model works well with capacities because:

- The smallest object you can independently assign to a capacity is a workspace (you can't assign a report, for example). By separating customers by profiles, you get different workspaces - one per customer. This way, you get full flexibility to manage each customer according to their performance needs, and optimize capacity utilization by scaling up or down. For example, you can manage large and essential customers with high volume and volatility in a separate capacity to ensure a consistent level of service, while grouping smaller customers in another capacity to optimize costs.
- Separating workspaces also means separating datasets between customers so that data models are in smaller chunks, rather than a single large dataset. These smaller models enable the capacity to manage memory usage more efficiently. Small, unused datasets can be evicted after a period of inactivity, in order to maintain good performance.

When buying a capacity, consider the limit on the number of parallel refreshes, as refresh processes might need extra capacity when you have multiple datasets.

Row Level Security

This article explains how to use profiles to create a separate dataset for each customer. Alternatively, ISV applications can store all their customers' data in one large dataset and use Row-level security (RLS) to protect each customer's data. This approach can be convenient for ISVs that have relatively few customers and small to medium-sized datasets because:

- There's only one report and one dataset to maintain
- The onboarding process for new customers can be simplified

Before using RLS, however, make sure you understand its limitations. All the data for all customers are in one large Power BI dataset. This dataset runs in a single capacity with its own scaling and other limitations.

Even if you use service principal profiles to separate your customers' data, you can still use RLS within a single

customer's dataset to give different groups access to different parts of the data. For example, you could use RLS to prevent members of one department from accessing data of another department in the same organization.

Considerations and limitations

Service principal profiles are not supported with Azure Analysis Services (AAS) in live connection mode.

Power BI capacity limitations

- Each capacity can only use its allocated memory and V-cores, according to the SKU purchased. For the recommended dataset size for each SKU, reference Premium large datasets.
- To use a dataset larger than 10 GB, use a Premium capacity and enable the Large datasets setting.
- For the number of refreshes that can run concurrently on a capacity, reference resource management and optimization.
- Scaling a capacity in Gen 1, on average, takes between 1-2 minutes. During that time, the capacity isn't available. We recommend using a scale-out approach to avoid downtime. For Gen 2, scaling is instantaneous.

Manage service principals

Change a service principal

In Power BI, a profile belongs to the service principal that created it. That means, a profile can't be shared with other principals. With this limitation, if you want to change the service principal for any reason, you'll need to recreate all the profiles and provide the new profiles access to the relevant workspaces. Often, the ISV application needs to save a mapping between a profile ID and a customer ID in order to pick the right profile when needed. If you change the service principal and recreate the profiles, the IDs will also change, and you may need to update the mapping in the ISV application database.

Delete a service principal

> **WARNING**
> Be very careful when deleting a service principal. You don't want to accidentally lose data from all its associated profiles.

If you delete the service principal in the active directory, all its profiles in Power BI will be deleted. However, Power BI won't delete the content immediately, so the tenant admin can still access the workspaces. Be careful when deleting a service principal used in a production system, especially when you created profiles using this service principal in Power BI. If you do delete a service principal that has created profiles, be aware that you will need to recreate all the profiles, provide the new profiles access to the relevant workspaces, and update the profile IDs in the ISV application database.

Data separation

When data is separated by service principal profiles, a simple mapping between a profile and customer prevents one customer from seeing another customer's content. Using a single *service principal* requires that the service principal has access to all the different workspaces in all the profiles.

To add extra separation, you can assign a separate service principal to each customer, instead of having a single service principal access multiple workspaces using different profiles. This has several advantages, including:

- Human error or a credential leak won't cause multiple customers' data to be exposed.
- Certificate rotation can be done separately for each customer.

However, using multiple service principals comes with a high management cost. Select this path only if you need the extra separation. Keep in mind that the configuration of which data to show an end user is defined when you generate the embed token, a backend-only process that end users can't see or change. Requesting an embed token using a customer-specific profile will generate an embed token for that specific profile, which will give you customer separation in consumption.

One report, multiple datasets

Generally, you have one report and one dataset per customer. If you have hundreds of reports, you'll have hundreds of datasets. Sometimes, you might have one report format, with different datasets (for example, different parameters or connection details). Each time you have a new version of the report, you'll need to update all the reports for all customers. Although you can automate this, it's easier to manage if you have just one copy of the report. This can be achieved by creating a workspace that contains the report to embed. During a session runtime, you can bind the report to the customer-specific dataset. Read dynamic bindings for more details.

Customizing and authoring content

When you create content, carefully consider who you allow to edit it. If you permit multiple users in each tenant to edit it's easy to exceed dataset limitations. If you decide to give users editing capability, we recommend monitor their content generation closely, and scale up as needed. For the same reason, we don't recommend using this capability for content personalization, where each user can make small changes to a report and save it for themselves. If the ISV application allows content personalization, consider introducing and communicating workspace retention policies for user-specific content. Retention policies make it easier to delete content when users move to a new position, leave the company, or stop using the platform.

System-Managed identity

Instead of a service principal, you can use a user-assigned or system-assignedmanaged identity to create profiles. Managed identities reduce the need for secrets and certificates.

Be careful when using a system-managed identity because:

- If a system-assigned managed identity is accidentally turned off, you'll lose access to the profiles. This situation is similar to a case where a service principal is deleted.
- A system-assigned managed identity is connected to a resource in Azure (web app). If you delete the resource, the identity will be deleted.
- Using multiple resources (different web apps in different regions) will result in multiple identities that need to be managed separately (each identity will have its own profiles).

Due to the above considerations, we recommend that you use a user-assigned managed identity.

Next steps

Learn more about service principals

Use the Power BI SDK with service principals

Migrate multi-customer applications to the service principal profiles model

Migrate multi-customer applications to the service principal profiles model

6/30/2022 • 4 minutes to read • Edit Online

This article describes how you can get better scalability by migrating your Power BI embedded analytics multi-customer apps to the service principal profiles model.

Service principal profiles make it easier to manage organizational content in Power BI and use your capacities more efficiently.

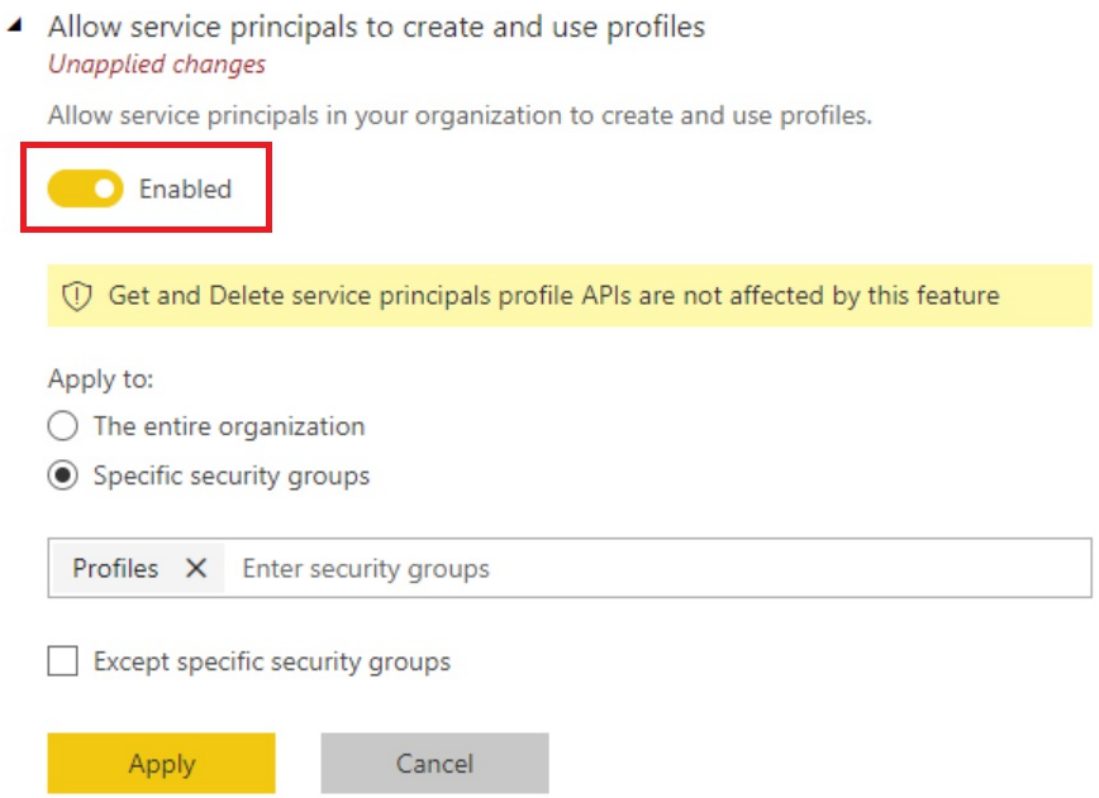
> **NOTE**
>
> This article is aimed at organizations that already have an app that supports multiple customers from a single Power BI tenant.
>
> Not all apps benefit from the service principal model. For example, the following apps shouldn't migrate:
>
> - Small apps that maintain one service principal with a small number of objects.
> - Apps that use one multiple service principal per customer

Prerequisites

It's important to read about service principal profiles before you start the migration.

You also need to do the following steps:

- Set up the service principal by following **the first three steps** of Embed Power BI content with service principal.
- From a Power BI tenant admin account, enable creating profiles in the tenant.



Migrate to service principal profiles

Migrating to service principal profiles involves the following steps:

1. Create profiles, one profile per customer.
2. Organize your workspaces.
3. Change the application code to use profiles.
4. Test your application with the profiles model.
5. Clean up redundant permissions.

Create Profiles (Required)

Use Profiles REST API with the service principal you created to create one profile for each customer.

It's a good idea to save a mapping of each data customer ID with its corresponding profile ID in your database. You'll need this mapping later to make API calls with the tenant profile.

Organize your workspaces

The easiest way to manage your data is by maintaining one workspace per customer. If your app already uses this model, you don't need to create new workspaces. However, you still have to provide each profile with access to the corresponding workspace using the Add Group User API.

If you don't have one workspace per customer, use the corresponding profile to call Create Group User API to create a new workspace for each customer.

Organize items in workspaces

You should now have a profile and a workspace for each customer. If you created new workspaces in the previous step, you need to import items (like reports and datasets) into these workspaces. The datasets you import depend on your current solution:

- If your app uses a separate dataset for each customer, the dataset design can work as it is.
- If your app uses one dataset withrow level security (RLS)to provide different data to different customers,

you can get better scalability by creating a separate dataset for each customer and using profiles as described in this article.

- After overcoming scalability limitations by using profiles and separate data sources, you can get even more data separation by using RLS with profiles.
  - If you rely on Dynamic RLS, the name of the profile will be returned in the DAX function `UserName()`.
  - If you use static RLS and override roles when generating the embed token, you can continue doing this.

Once the items are ready, import them into the relevant workspaces. To automate the process, consider using the Import API.

Change the application codes to use profiles

Once you have profiles with access to the relevant workspaces, and a database with mapping that shows you which profile represents which customer, you can make the necessary code changes. We recommend that you keep two code flows side by side and gradually expose the profiles code flow to your customers.

Make the following code changes:

- **Authorization code change**
  - If you're using a *master user* in the Azure AD app, change the acquire token code. Read embed with service principal to learn about creating an app-only Azure AD token.
  - If you're using a *service principal* and you created a new one for profiles, adjust the code to use the correct service principal ID and secrets.
- **Management code change**

  Some apps have management code that automates onboarding a new customer upon registration. Often, the management code uses Power BI REST APIs to create workspaces and import content. Most of this code should remain the same, but you may need to adapt the following details:
  - Each time you create a new customer tenant, create a new service profile to be the creator and administrator of the workspace for that tenant.
  - If you decide to reorganize your Power BI content, edit the code to reflect the changes.
- **Embed token code change**

  Replace the API caller. Make sure a profile calls the GenerateToken API because in the profiles model, only the specific profile has access to the customer's content.

Validate

It's best practice to test your app thoroughly before moving it to the profiles model. Reports may load even if there are bugs in the SaaS application code because you didn't delete the older permissions on the workspaces.

Clean up after migration

Now that you finished the migration and validated the results, remove what you don't need anymore.

- Clean up code: You might want to disable old code paths to ensure that you're only running new code that relies on profiles.
- Clean up workspaces and permissions in Power BI: If you created new workspaces, you can delete the old workspaces that are no longer in use. If you reused the same workspaces, you may want to delete the older permissions (such as *master user* permissions) on the workspace.

Next steps

Manage service principal profiles

More questions? Try asking the Power BI Community

Connect a report to a dataset using dynamic binding

6/30/2022 • 2 minutes to read • Edit Online

When a report is connected to a dataset, you can use dynamic binding. The connection between the report and the dataset, is known as *binding*. When the binding is determined at the point of embedding, as opposed to being predetermined earlier, the binding is known as dynamic binding.

When embedding a Power BI report using *dynamic binding*, you can connect the same report to different datasets depending on the user's credentials.

This means that you can use one report to display different information, depending on the dataset it's connected to. For example, a report showing retail sale values can be connected to different retailer datasets, and produce different results, depending on the dataset of the retailer it's connected to.

The report and the dataset don't need to reside in the same workspace. Both workspaces (the one containing the report, and the one containing the dataset) must be assigned to a capacity.

As part of the embedding process, make sure you *generate a token with sufficient permissions*, and *adjust the config object*.

Generating a token with sufficient permissions

Dynamic binding is supported for both *Embedding for your organization* and *Embedding for your customers* scenarios. The table below describes the considerations for each scenario.

SCENARIO	DATA OWNERSHIP	TOKEN	REQUIREMENTS
Embedding for your organization	User owns data	Access token for Power BI users	The user who's Azure AD token is used, must have appropriate permissions for all items (reports, datasets, etc.).
Embedding for your customers	App owns data	Access token for non-Power BI users	Must include permissions for both the report and the dynamically bound dataset. Use the API for generating an embed token for multiple items, to generate an embed token that supports multiple items.

NOTE

The maximum number of data sources allowed per user is 1000. This limit implies that the combined number of datas sources used in the dynamic binding between reports and datasets by this user cannot exceed 1000.

Adjusting the config object

For dynamic binding to work, you need to add `datasetBinding` to the config object. To learn how this is done,

see Bind datasets dynamically to a report.

Next steps

If you're new to embedding in Power BI, review these tutorials to learn how to embed your Power BI content.

Embed Power BI content into an application for your customers

Embed Power BI content into an application for your organization

Monitor Power BI Embedded

6/30/2022 • 5 minutes to read • Edit Online

When you have critical applications and business processes relying on Azure resources, you want to monitor those resources for their availability, performance, and operation. This article describes the monitoring data generated by Power BI Embedded and how you can use the features of Azure Monitor to analyze and alert on this data.

> **TIP**
> You can also use the Premium Gen2 Monitoring App to monitor your Embedded Gen 2 capacity.

Monitor overview

The **Overview** page in the Azure portal for each *Power BI Embedded* instance, includes the following information:

- **Resource group** - The resource group the instance belongs to
- **Status** - The status of your Power BI Embedded instance
- **Location** - The location of your Power BI Embedded instance
- **Resource name** - The name of your Power BI Embedded instance
- **SKU** - The SKU your Power BI Embedded instance is using
- **Subscription name** - Your Power BI Embedded instance subscription name
- **Subscription ID** - Your Power BI Embedded instance subscription ID

What is Azure Monitor?

Power BI Embedded creates monitoring data using Azure Monitor, which is a full stack monitoring service in Azure that provides a complete set of features to monitor your Azure resources in addition to resources in other clouds and on-premises.

Start with the article Monitoring Azure resources with Azure Monitor, which describes the following concepts:

- What is Azure Monitor?
- Costs associated with monitoring
- Monitoring data collected in Azure
- Configuring data collection
- Standard tools in Azure for analyzing and alerting on monitoring data

The following sections build on this article by describing the specific data gathered for Power BI Embedded and providing examples for configuring data collection and analyzing this data with Azure tools.

Monitoring data

Power BI Embedded collects the same kinds of monitoring data as other Azure resources that are described in Monitoring data from Azure resources.

See Monitoring *Power BI Embedded* data reference for detailed information on the metrics and logs metrics created by Power BI Embedded.

Collection and routing

Platform metrics and the Activity log are collected and stored automatically, but can be routed to other locations by using a diagnostic setting.

Resource Logs are not collected and stored until you create a diagnostic setting and route them to one or more locations.

See Create diagnostic setting to collect platform logs and metrics in Azure for the detailed process for creating a diagnostic setting using the Azure portal, CLI, or PowerShell. When you create a diagnostic setting, you specify which categories of logs to collect. The categories for *Power BI Embedded* are listed in Power BI Embedded monitoring data reference.

Using PowerShell to enable diagnostics

To enable metrics and diagnostics logging by using PowerShell, use the commands listed below. To learn more about using PowerShell to enable diagnostics, see Create and configure a Log Analytics workspace in Azure Monitor using PowerShell.

- To enable storage of diagnostics logs in a storage account, use this command:

  ```
  Set-AzDiagnosticSetting -ResourceId [your resource id] -StorageAccountId [your storage account id] -
  Enabled $true
  ```

  The storage account ID is the resource ID for the storage account where you want to send the logs.

- To enable streaming of diagnostics logs to an event hub, use this command:

  ```
  Set-AzDiagnosticSetting -ResourceId [your resource id] -ServiceBusRuleId [your service bus rule id] -
  Enabled $true
  ```

- The Azure Service Bus rule ID is a string with this format:

  ```
  {service bus resource ID}/authorizationrules/{key name}
  ```

- To enable sending diagnostics logs to a Log Analytics workspace, use this command:

  ```
      Set-AzDiagnosticSetting -ResourceId [your resource id] -WorkspaceId [resource id of the log
  analytics workspace] -Enabled $true
  ```

- You can obtain the resource ID of your Log Analytics workspace by using the following command:

  ```
  (Get-AzOperationalInsightsWorkspace).ResourceId
  ```

You can combine these parameters to enable multiple output options.

The metrics and logs you can collect are discussed in the following sections.

Analyzing metrics

You can analyze metrics for *Power BI Embedded* with metrics from other Azure services using metrics explorer by opening **Metrics** from the **Azure Monitor** menu. See Getting started with Azure Metrics Explorer for details on using this tool.

For a list of the platform metrics collected for Power BI Embedded, see Monitoring *Power BI Embedded* data

reference metrics

For reference, you can see a list of all resource metrics supported in Azure Monitor.

Analyzing logs

Data in Azure Monitor Logs is stored in tables where each table has its own set of unique properties.

All resource logs in Azure Monitor have the same fields followed by service-specific fields. The common schema is outlined in Azure Monitor resource log schema The schema for Power BI Embedded resource logs is found in the Power BI Embedded Data Reference.

The Activity log is a platform login Azure that provides insight into subscription-level events. You can view it independently or route it to Azure Monitor Logs, where you can do much more complex queries using Log Analytics.

For a list of the types of resource logs collected for Power BI Embedded, see Monitoring Power BI Embedded data reference

For a list of the tables used by Azure Monitor Logs and queryable by Log Analytics, see Monitoring Power BI Embedded data reference

Sample Kusto queries

> **IMPORTANT**
>
> When you select **Logs** from the Power BI Embedded menu, Log Analytics is opened with the query scope set to the current Power BI Embedded resource. This means that log queries will only include data from that resource. If you want to run a query that includes data from other Power BI Embedded resource or data from other Azure services, select **Logs** from the **Azure Monitor** menu. See Log query scope and time range in Azure Monitor Log Analytics for details.

Here is an example of a query that is completed in less than five minutes (300,000 milliseconds).

```
search *
| where Type == "AzureDiagnostics"
| where ( OperationName == "QueryEnd" )
| where toint(Duration_s) < 300000
```

Alerts

Azure Monitor alerts proactively notify you when important conditions are found in your monitoring data. They allow you to identify and address issues in your system before your customers notice them. You can set alerts on metrics, logs, and the activity log.

Next steps

You can learn more about Azure resource diagnostic logging.

Monitoring Power BI Embedded data reference

Monitoring Azure resources with Azure Monitor

Embed Power BI content with service principal and an application secret

6/30/2022 • 6 minutes to read • Edit Online

Service principal is an authentication method that can be used to let an Azure AD application access Power BI service content and APIs.

When you create an Azure Active Directory (Azure AD) app, a service principal object is created. The service principal object, also known simply as *service principal*, allows Azure AD to authenticate your app. Once authenticated, the app can access Azure AD tenant resources.

To authenticate, the service principal uses the Azure AD app's *Application ID*, and one of the following:

- Certificate
- Application secret

This article describes service principal authentication using *Application ID* and *Application secret*.

> **NOTE**
>
> Azure AD recommends that you secure your backend services using certificates, rather than secret keys.
>
> - Learn more about getting access tokens from Azure AD using secret keys or certificates.
> - To secure your solution using a certificate, complete the instructions in this article and then follow the steps described in Embed Power BI content with service principal and a certificate.

Method

To use service principal and an application ID embedded analytics, follow these steps:

1. Create an Azure AD app.
    a. Create the Azure AD app's secret.
    b. Get the app's *Application ID* and *Application secret*.

    > **NOTE**
    >
    > These steps are described in **step 1**. For more information about creating an Azure AD app, see create an Azure AD app.

2. Create an Azure AD security group.
3. Enable the Power BI service admin settings.
4. Add the service principal to your workspace.
5. Embed your content.

IMPORTANT

Once you enable service principal to be used with Power BI, the application's AD permissions don't take effect anymore. The application's permissions are then managed through the Power BI admin portal.

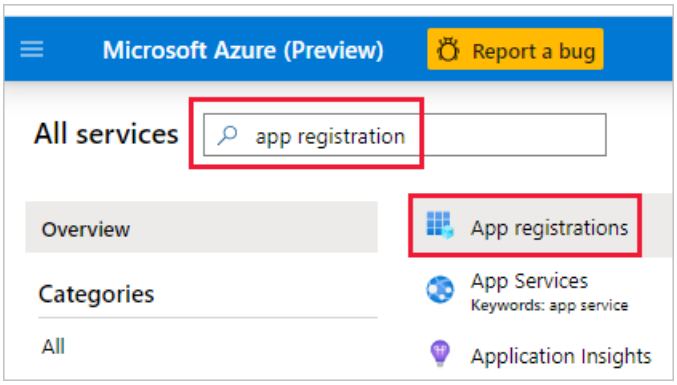
Step 1 - Create an Azure AD app

Create an Azure AD app using one of these methods:

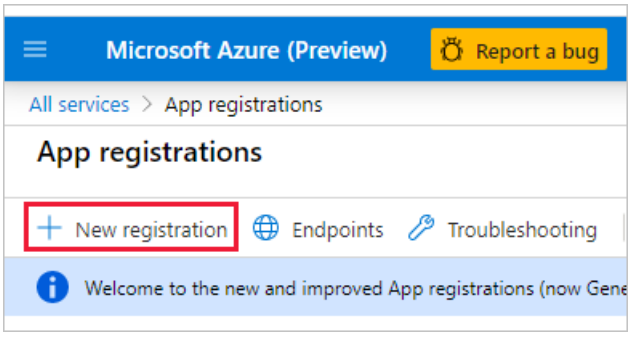
- Create the app in the Microsoft Azure portal
- Create the app using PowerShell

Creating an Azure AD app in the Microsoft Azure portal

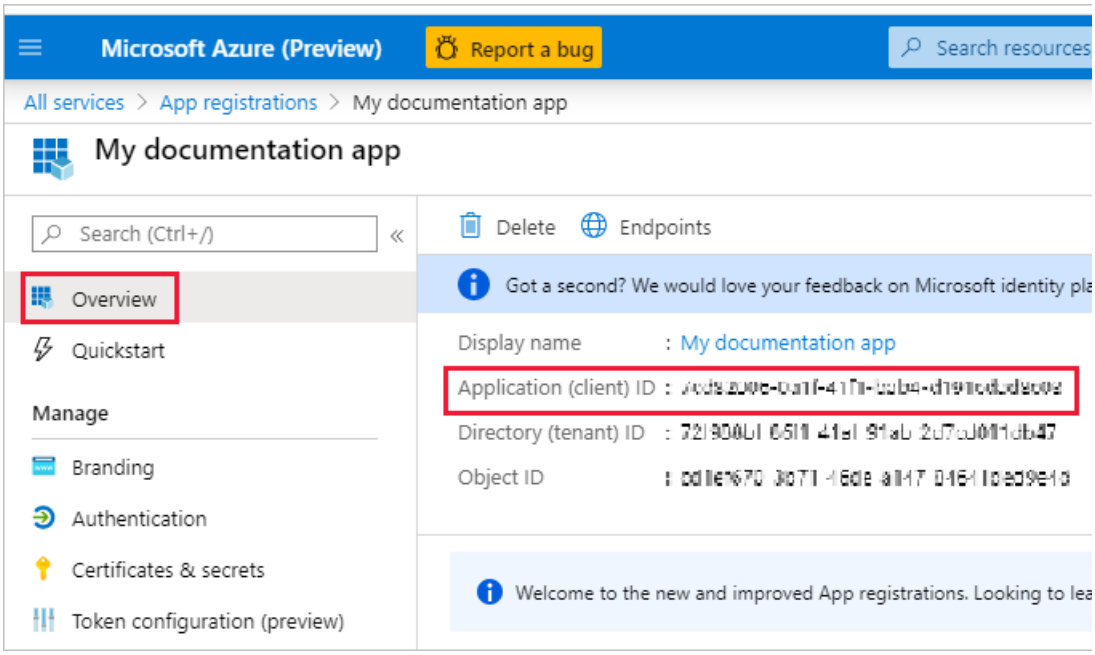
1. Log into Microsoft Azure.
2. Search for **App registrations** and click the **App registrations** link.



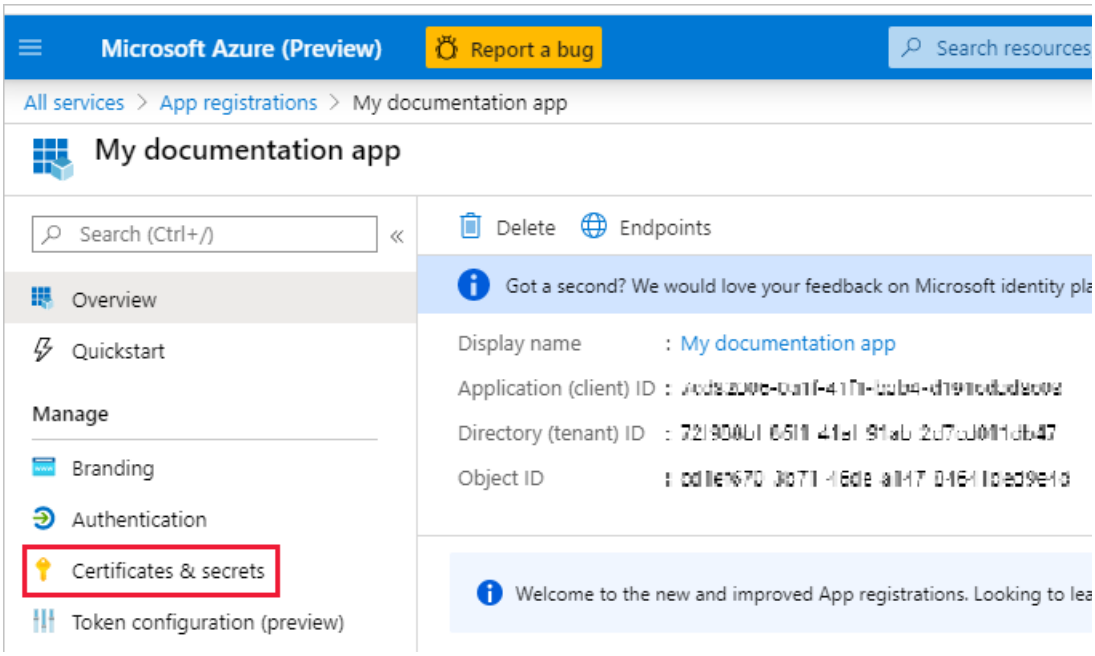
3. Click **New registration**.



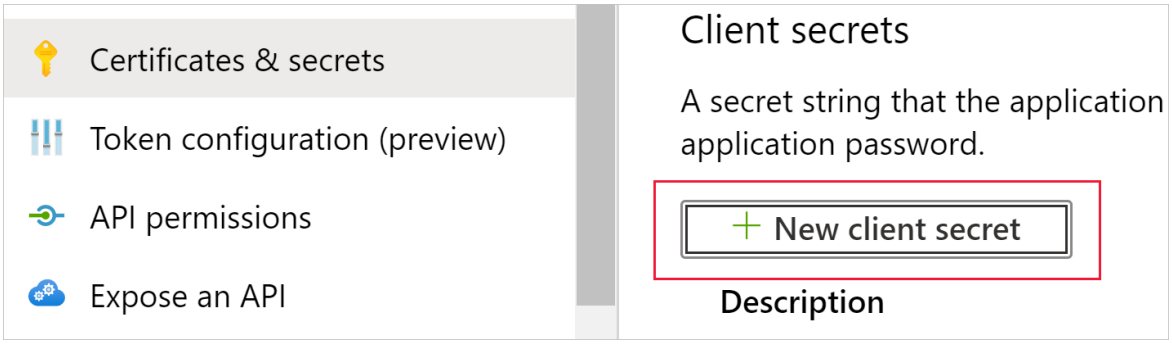
4. Fill in the required information:
   - **Name** - Enter a name for your application
   - **Supported account types** - Select supported account types
   - (Optional) **Redirect URI** - Enter a URI if needed
5. Click **Register**.
6. After registering, the *Application ID* is available from the **Overview** tab. Copy and save the *Application ID* for later use.



7. Click the **Certificates & secrets** tab.

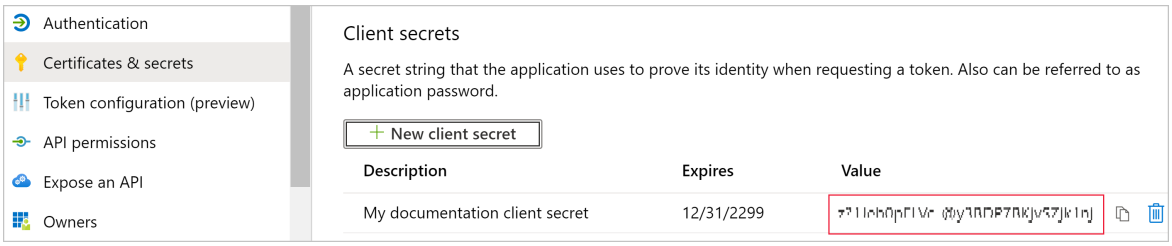


8. Click **New client secret**



9. In the *Add a client secret* window, enter a description, specify when you want the client secret to expire, and click **Add**.

10. Copy and save the *Client secret* value.



> **NOTE**
>
> After you leave this window, the client secret value will be hidden, and you'll not be able to view or copy it again.

Creating an Azure AD app using PowerShell

This section includes a sample script to create a new Azure AD app using PowerShell.

```
# The app ID - $app.appid
# The service principal object ID - $sp.objectId
# The app key - $key.value

# Sign in as a user that's allowed to create an app
Connect-AzureAD

# Create a new Azure AD web application
$app = New-AzureADApplication -DisplayName "testApp1" -Homepage "https://localhost:44322" -ReplyUrls
"https://localhost:44322"

# Creates a service principal
$sp = New-AzureADServicePrincipal -AppId $app.AppId

# Get the service principal key
$key = New-AzureADServicePrincipalPasswordCredential -ObjectId $sp.ObjectId
```

Step 2 - Create an Azure AD security group

Your service principal doesn't have access to any of your Power BI content and APIs. To give the service principal access, create a security group in Azure AD, and add the service principal you created to that security group.

There are two ways to create an Azure AD security group:

- Manually (in Azure)
- Using PowerShell

Create a security group manually

To create an Azure security group manually, follow the instructions in create a basic group and add members.

Create a security group using PowerShell

Below is a sample script for creating a new security group and adding an app to that security group.

> **NOTE**
>
> If you want to enable service principal access for the entire organization, skip this step.

```
# Required to sign in as admin
Connect-AzureAD

# Create an Azure AD security group
$group = New-AzureADGroup -DisplayName <Group display name> -SecurityEnabled $true -MailEnabled $false -
MailNickName notSet

# Add the service principal to the group
Add-AzureADGroupMember -ObjectId $($group.ObjectId) -RefObjectId $($sp.ObjectId)
```

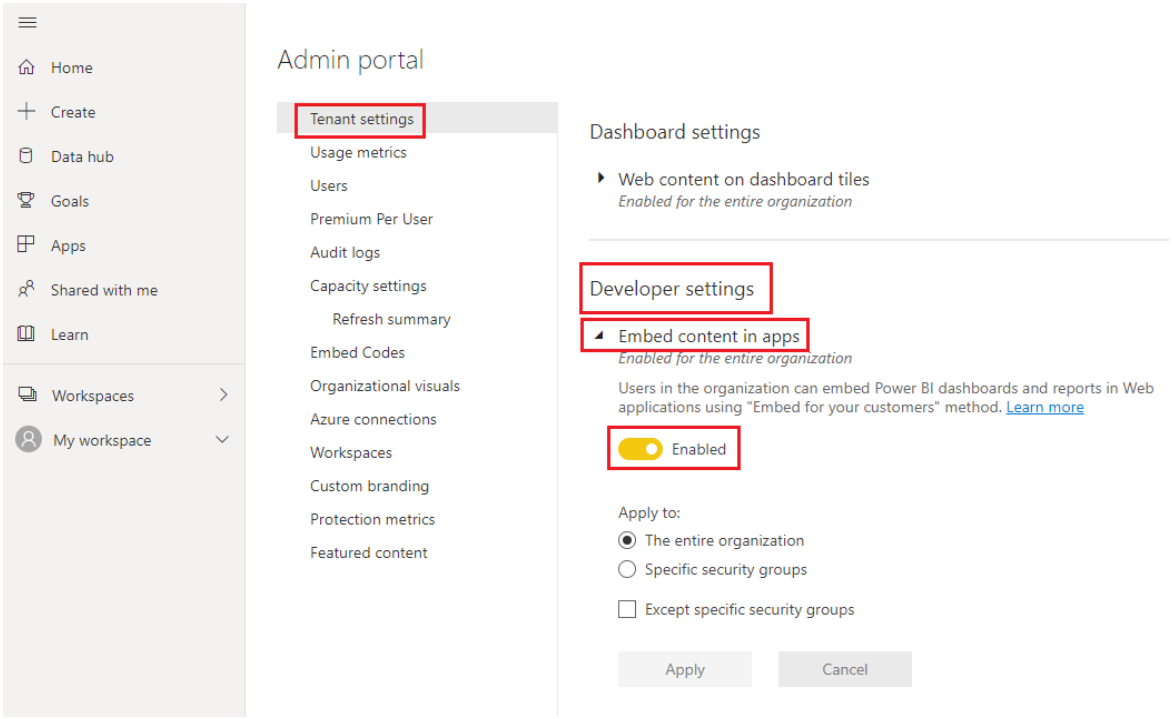
Step 3 - Enable the Power BI service admin settings

For an Azure AD app to be able to access the Power BI content and APIs, a Power BI admin needs to enable the following setting:

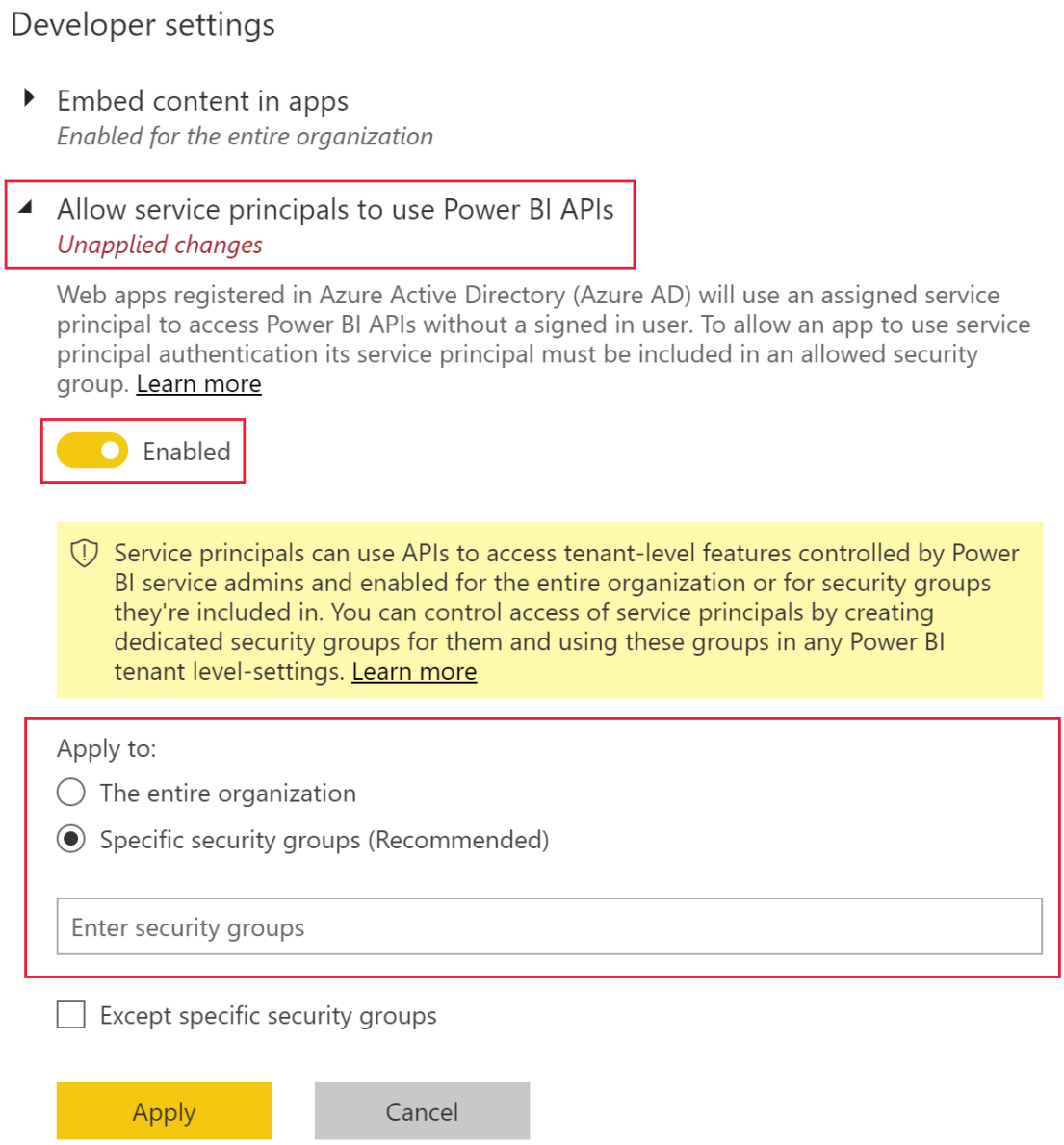
- Embed content in apps
- Allow service principals to use Power BI APIs.

Go to **Tenant settings** in the Admin portal, and scroll down to **Developer settings**.

- Enable to the **Embed content in apps** switch either for the entire organization or for the specific security group you created in Azure AD.



- Enable the **Allow service principals to use Power BI APIs** switch either for the entire organization or for the specific security group you created in Azure AD.



IMPORTANT

Service principals have access to any tenant settings they're enabled for. Depending on your admin settings, this includes specific security groups or the entire organization.

To restrict service principal access to specific tenant settings, allow access only to specific security groups. Alternatively, you can create a dedicated security group for service principals, and exclude it from the desired tenant settings.

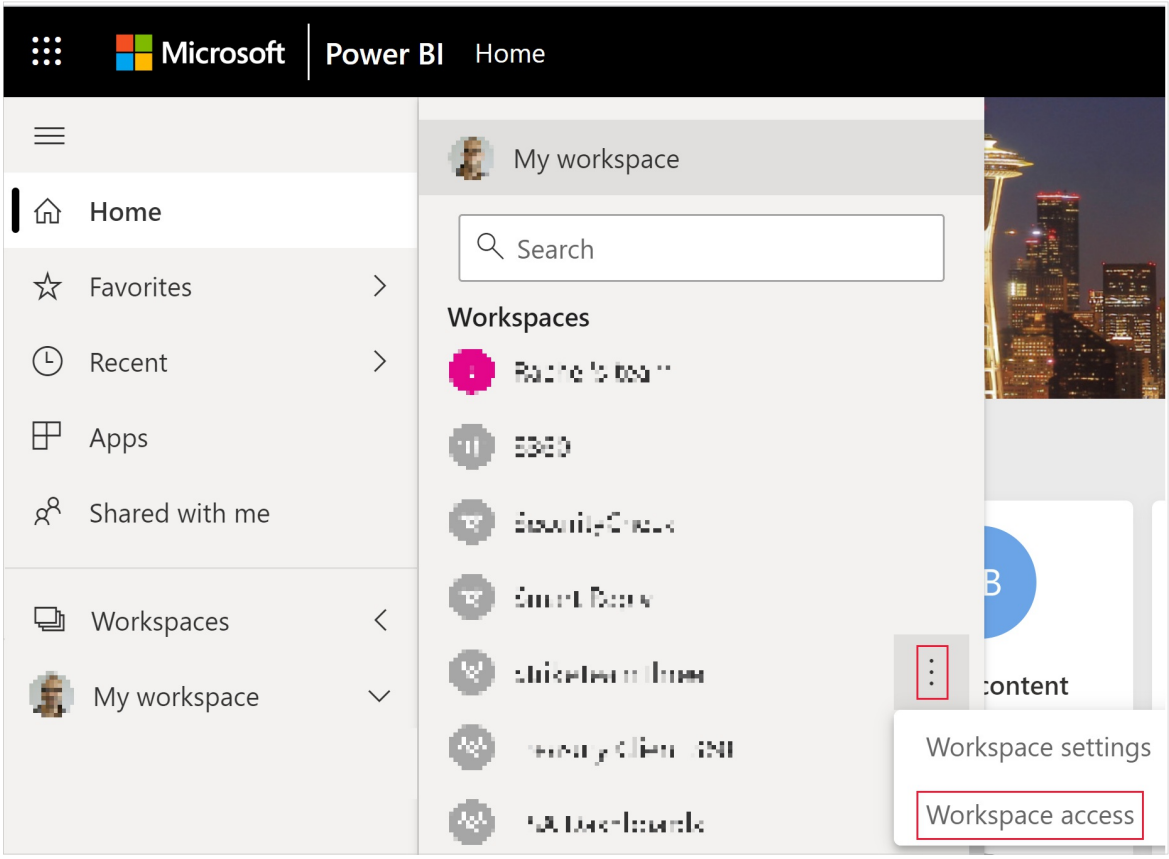
Step 4 - Add the service principal to your workspace

To enable your Azure AD app to access items such as reports, dashboards and datasets in the Power BI service, add the service principal entity, or the security group that includes your service principal, as a member or admin to your workspace.

NOTE

This section provides UI instructions. You can also add a service principal or a security group to a workspace, using the Groups - add group user API.

1. Scroll to the workspace you want to enable access for, and from the **More** menu, select **Workspace access**.



2. In the **Access** pane, text box, add one of the following:

   - Your **service principal**. The name of your service principal is the *Display name* of your Azure AD app, as it appears in your Azure AD app's overview tab.
   - The **security group** that includes your service principal.

3. From the drop-down menu, select **Member** or **Admin**.

4. Select **Add**.

Add a service principal as a workspace member using PowerShell

This section includes a sample script to add a service principal as a workspace member using PowerShell.

```
Login-PowerBI

# Service Principal Object ID for the created Service Principal
$SPObjectId = 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX'

$pbiWorkspace = Get-PowerBIWorkspace -Name "YourWorkspaceName"

Add-PowerBIWorkspaceUser -Id $pbiWorkspace.Id -AccessRight Member -PrincipalType App -Identifier $SPObjectId
```

Add a security group as a workspace member using PowerShell

This section includes a sample script to add a security group as a workspace member using PowerShell.

```
Login-PowerBI

# Security Group Object ID for the created Security Group
$SGObjectId = 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX'

$pbiWorkspace = Get-PowerBIWorkspace -Name "YourWorkspaceName"

Add-PowerBIWorkspaceUser -Id $pbiWorkspace.Id -AccessRight Member -PrincipalType Group -Identifier
$SGObjectId
```

Step 5 - Embed your content

You can embed your content within a sample application, or within your own application.

Once your content is embedded, you're ready to move to production.

> **NOTE**
> To secure your content using a certificate, follow the steps described in Embed Power BI content with service principal and a certificate.

Considerations and limitations

- Service principal only works with new workspaces.
- **My Workspace** isn't supported when using service principal.
- A capacity is required when moving to production.
- You can't sign into the Power BI portal using service principal.
- Power BI admin rights are required to enable service principal in developer settings within the Power BI admin portal.
- Embed for your organization applications can't use service principal.
- Dataflows management is not supported.
- Service principal only supports some read-only admin APIs. To enable service principal support for read-only admin APIs, you have to enable the Power BI service admin settings in your tenant. For more information, see Enable service principal authentication for read-only admin APIs.
- When using service principal with an Azure Analysis Services data source, the service principal itself must have an Azure Analysis Services instance permissions. Using a security group that contains the service principal for this purpose, doesn't work.

Next steps

Register an app

Power BI Embedded for your customers

Embed using a service principal and a certificate

Application and service principal objects in Azure Active Directory

Row-level security using on-premises data gateway with service principal

Embed Power BI content with service principal and a certificate

6/30/2022 • 3 minutes to read • Edit Online

Certificate-based authentication enables you to be authenticated by Azure Active Directory (Azure AD) with a client certificate. The client certificate can be on a Windows, Android or iOS device, or kept in an Azure Key Vault.

Using this method of authentication allows managing certificates from a central place, using the CA, for rotation or revocation.

You can learn more about certificates in Azure AD in the Client credential flows GitHub page.

Method

1. Embed your content with service principal.
2. Create a certificate.
3. Set up certificate authentication.
4. Get the certificate from Azure Key Vault.
5. Authenticate using service principal and a certificate.

Step 1 - Embed your content with service principal

To embed your content with service principal, follow the instructions in Embed Power BI content with service principal and an application secret.

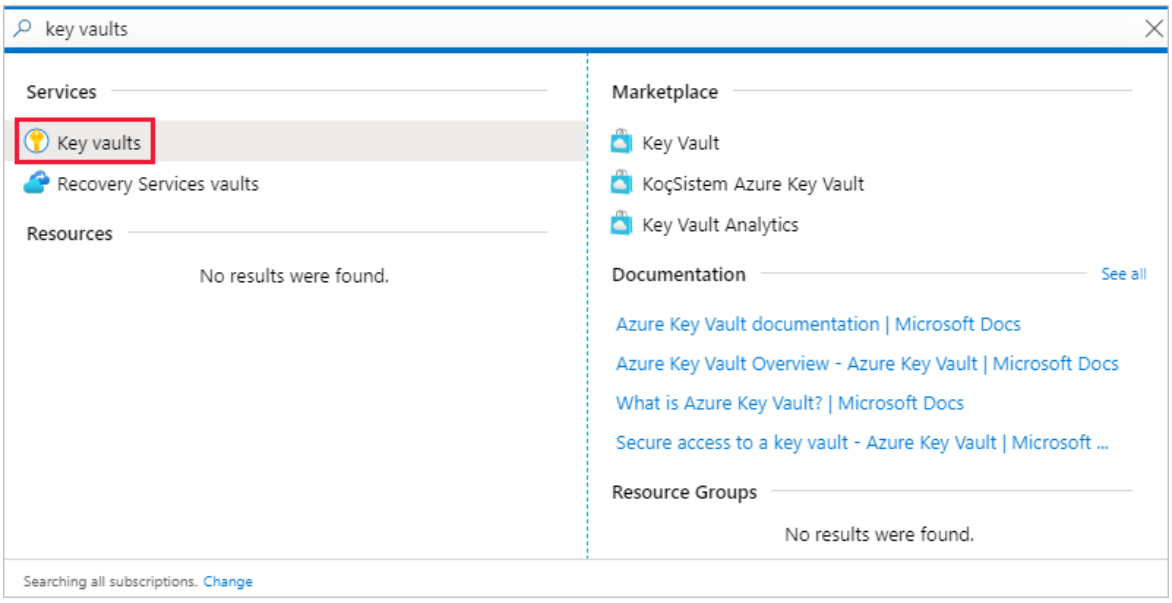
> **NOTE**
> If you already have content that's embedded using a service principal, skip this step and advance to step 2.

Step 2 - Create a certificate

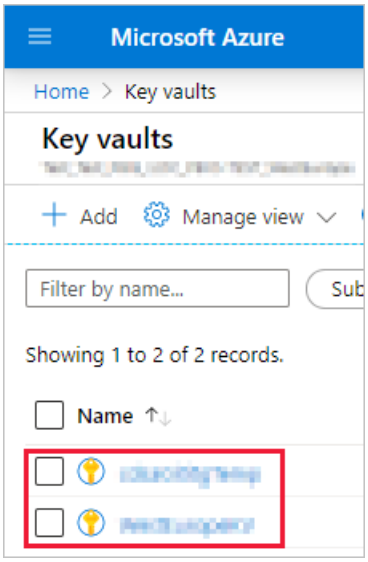
You can procure a certificate from a trusted *Certificate Authority*, or generate a certificate yourself.

This section describes creating a certificate using Azure Key Vault, and downloading the *.cer* file, which contains the public key.

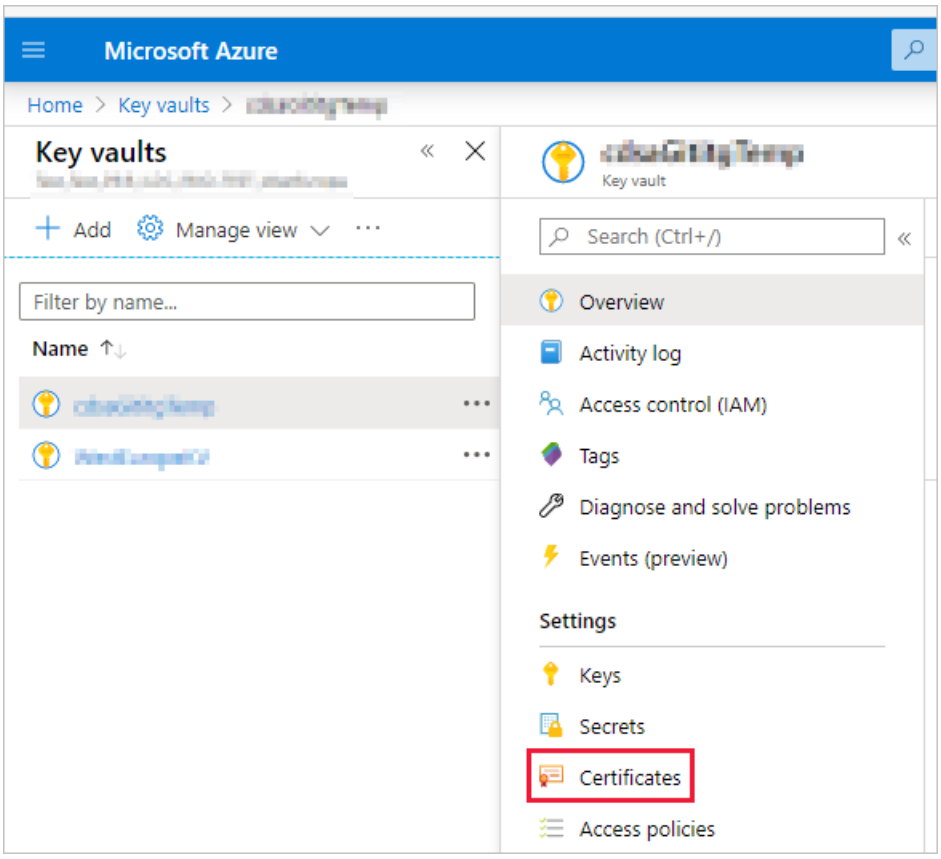
1. Log into Microsoft Azure.
2. Search for and select the **Key Vaults** link.



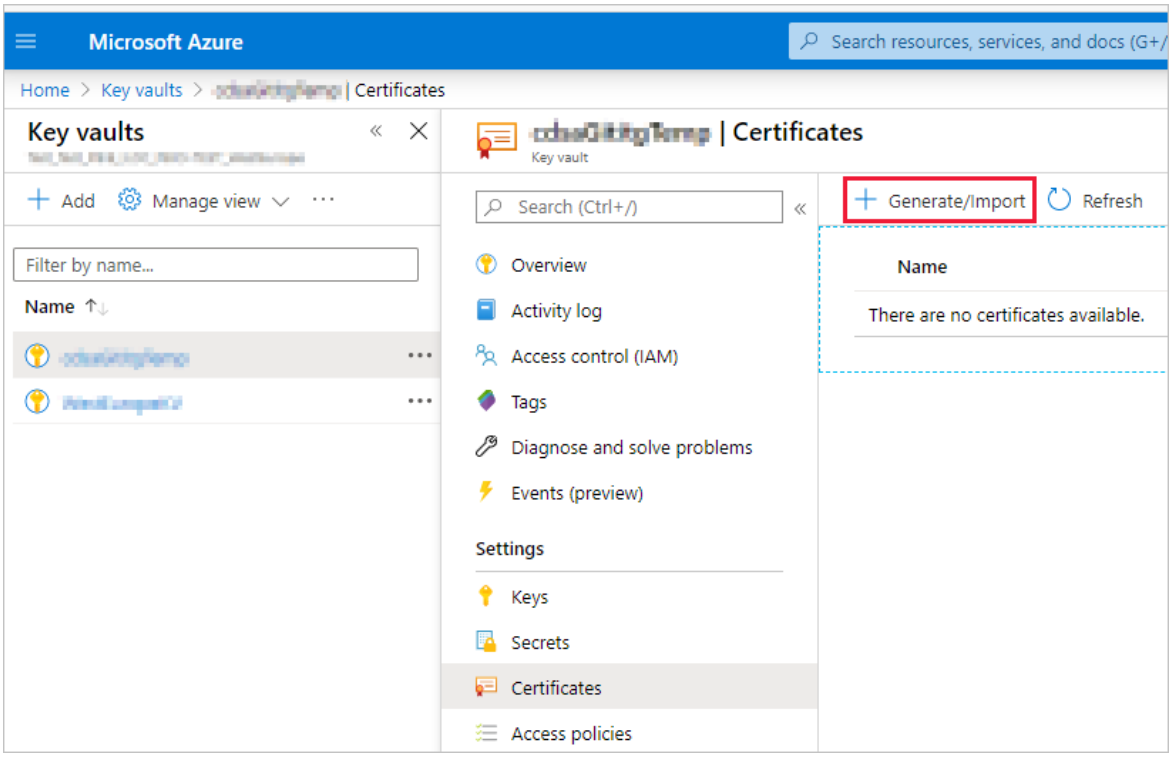
3. Select the key vault you want to add a certificate to.



4. Select **Certificates**.



5. Select **Generate/Import**.

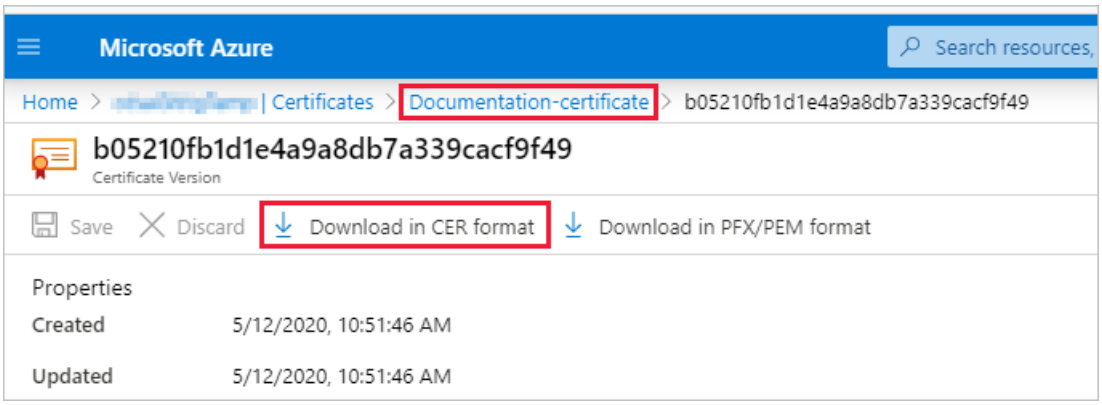


6. Configure the **Create a certificate** fields as follows:

- **Method of Certificate Creation** - General
- **Certificate Name** - Enter a name for your certificate
- **Type of Certificate Authority (CA)** - Self-signed certificate
- **Subject** - An X.500 distinguished name
- **DNS Names** - 0 DNS names

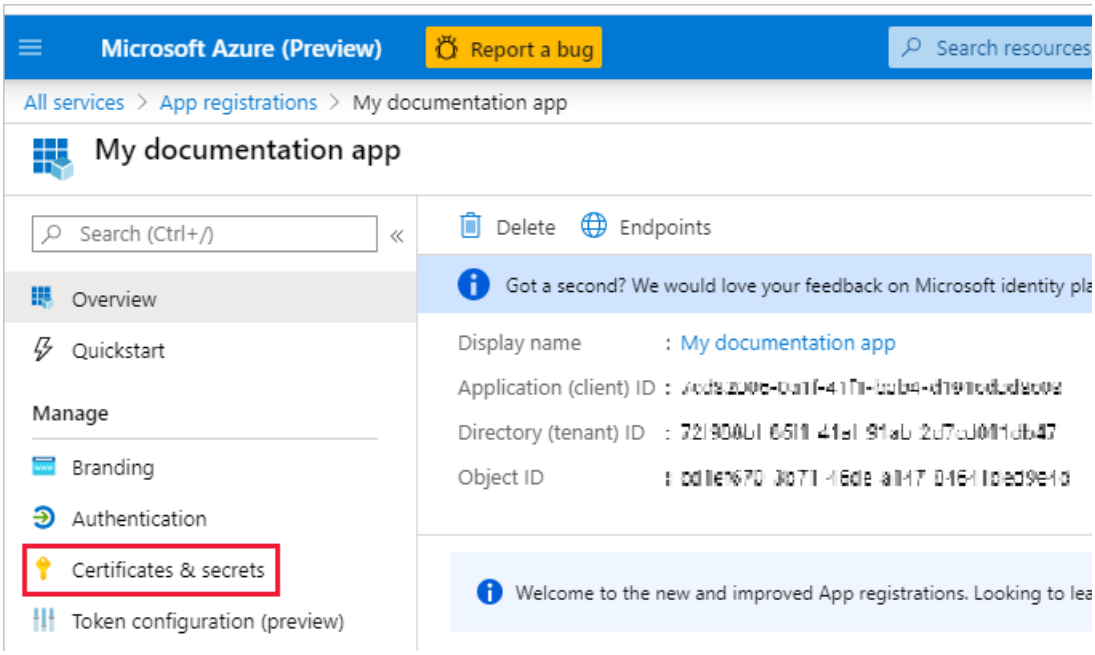
- **Validity Period (in months)** - Enter the certificate's validity duration
- **Content Type** - PKCS #12
- **Lifetime Action Type** - Automatically renew at a given percentage lifetime
- **Percentage Lifetime** - 80
- **Advanced Policy Configuration** - Not configured

7. Select **Create**. The newly created certificate is disabled by default. It can take up to five minutes to become enabled.
8. Select the certificate you created.
9. Select **Download in CER format**. The downloaded file contains the public key.



Step 3 - Set up certificate authentication

1. In your Azure AD application, select the **Certificates & secrets** tab.



2. Select **Upload certificate** and upload the *.cer* file you created and downloaded in step 2 of this tutorial. The *.cer* file contains the public key.

Step 4 - Get the certificate from Azure Key Vault

Use Managed Service Identity (MSI) to get the certificate from Azure Key Vault. This process involves getting the *.pfx* certificate that contains both the public and private keys.

Refer to the code example for reading the certificate from Azure Key Vault. If you want to use Visual Studio, refer to Configure Visual Studio to use MSI.

```
private X509Certificate2 ReadCertificateFromVault(string certName)
{
    var serviceTokenProvider = new AzureServiceTokenProvider();
    var keyVaultClient = new KeyVaultClient(new
KeyVaultClient.AuthenticationCallback(serviceTokenProvider.KeyVaultTokenCallback));
    CertificateBundle certificate = null;
    SecretBundle secret = null;

    certificate = keyVaultClient.GetCertificateAsync($"https://{KeyVaultName}.vault.azure.net/",
certName).Result;
    secret = keyVaultClient.GetSecretAsync(certificate.SecretIdentifier.Identifier).Result;

    return new X509Certificate2(Convert.FromBase64String(secret.Value));
}
```

Step 5 - Authenticate using service principal and a certificate

You can authenticate your app using service principal and a certificate that's stored in Azure Key Vault, by connecting to Azure Key Vault.

To connect and read the certificate from Azure Key Vault, refer to the code below.

> **NOTE**
> If you already have a certificate created by your organization, upload the *.pfx* file to Azure Key Vault.

```
// Preparing needed variables
var Scope = "https://analysis.windows.net/powerbi/api/.default"
var ApplicationId = "{YourApplicationId}"
var tenantSpecificURL = "https://login.microsoftonline.com/{YourTenantId}/"
X509Certificate2 certificate = ReadCertificateFromVault(CertificateName);

// Authenticating with a SP and a certificate
public async Task<AuthenticationResult> DoAuthentication(){
    IConfidentialClientApplication clientApp = null;
    clientApp = ConfidentialClientApplicationBuilder.Create(ApplicationId)
                                                    .WithCertificate(certificate)
                                                    .WithAuthority(tenantSpecificURL)
                                                    .Build();
    return await clientApp.AcquireTokenForClient(Scope).ExecuteAsync();
}
```

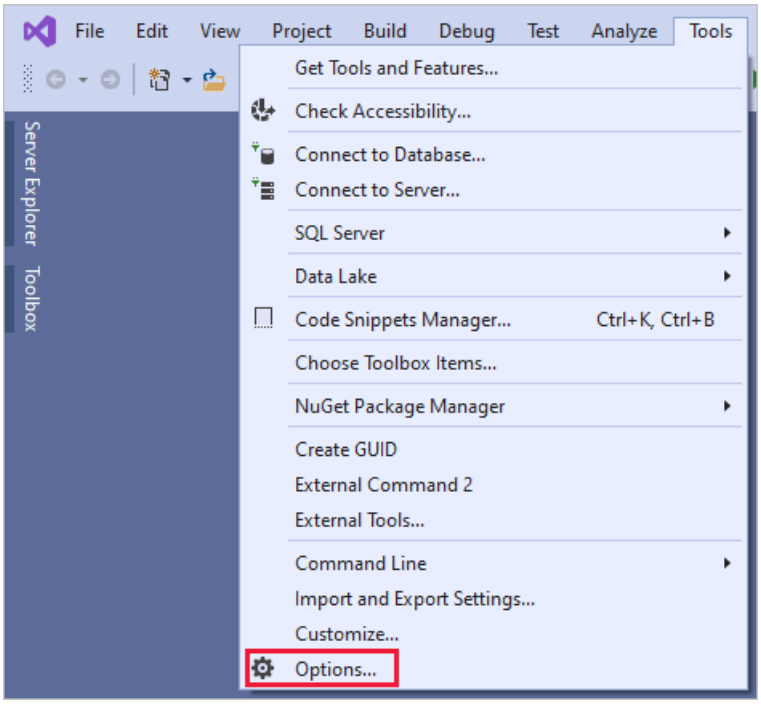
Configure Visual Studio to use MSI

When you create an embedded solution, it may be useful to configure Visual Studio to use Managed Service Identity (MSI). MSI is a feature that enables you to manage your Azure AD identity. Once configured, it will let Visual Studio authenticate against your Azure Key Vault.

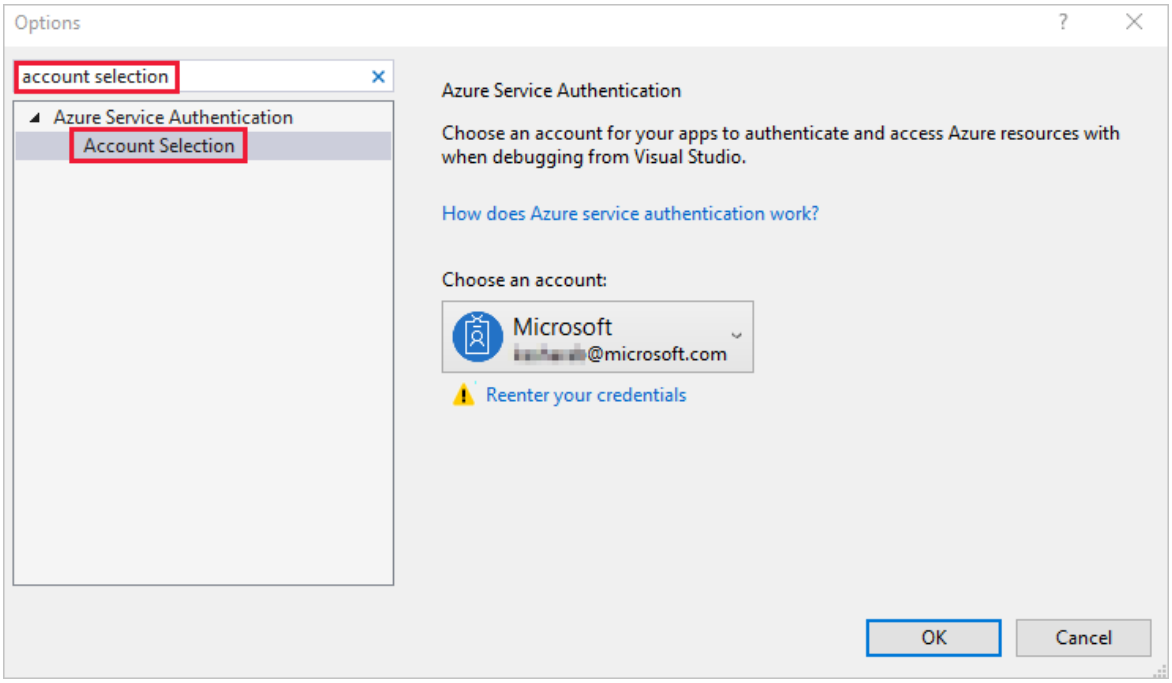
> **NOTE**
> The user that signs into Visual Studio has to have Azure Key Vault permissions to get the certificate.

1. Open your project in Visual Studio.

2. Select **Tools** > **Options**.



3. Search for and select **Account Selection**.



4. Add the account that has access to your Azure Key Vault.

Next steps

Register an app

Power BI Embedded for your customers

Application and service principal objects in Azure Active Directory

Row-level security using on-premises data gateway with service principal

Embed Power BI paginated reports

6/30/2022 • 6 minutes to read • Edit Online

With Power BI embedded analytics, you can create Power BI content that displays paginated reports in a fully integrated and interactive application. Embed paginated reports using the solution that works best for you, embed for your customers or embed for your organization.

This article describes how to embed paginated reports using the embedding sample tutorials.

Prerequisites

To get started, you're required to have:

- Embed for your customers
- Embed for your organization
- A service principal
- Your own Azure Active Directory tenant setup
- A capacity, with paginated reports workload enabled

If you don't have an Azure subscription, create a free account before you begin.

Method

To embed a paginated report using the sample app, follow these steps:

1. Create a workspace.
2. Create a capacity.
3. Assign a workspace to a capacity.
4. Enable paginated reports workload.
5. Create and upload your paginated report.
6. Embed content using the sample application.

Step 1 - Create a workspace

- Embed for your customers
- Embed for your organization

As you're using a service principal to sign into your application, you'll need to create a new workspace.

Your *service principal* must also be an admin or member of the Power BI workspaces.

Step 2 - Create a capacity

- Embed for your customers
- Embed for your organization

Before you import or upload a paginated report to embed, the workspace containing the report must be assigned to a capacity. There are two types of capacity you can choose from:

- **Power BI Premium** - For embedding a paginated report, an *EM* or *P* SKU is required. For more information about this subscription, see What is Power BI Premium?
- **Azure Power BI Embedded** - You can purchase a capacity from the Microsoft Azure portal. This subscription uses the *A* SKUs. For details on how to create a Power BI Embedded capacity, see Create Power BI Embedded capacity in the Azure portal.

> **NOTE**
> Power BI Embedded recently released a new version, called **Embedded Gen2**. Embedded Gen2 simplifies the management of embedded capacities, and improves the Power BI Embedded experience. For more information, see Paginated reports and Premium Gen2.

The following table describes the resources and limits of each SKU. To determine which capacity best fits your needs, see the which SKU should I purchase for my scenario table.

CAPACITY NODES	TOTAL V-CORES	BACKEND V-CORES	FRONTEND V-CORES	RAM (GB)
EM1/A1 with Embedded Gen2	1	0.5	0.5	2.5
EM2/A2 with Embedded Gen2	2	1	1	5
EM3/A3 with Embedded Gen2	4	2	2	10
P1/A4	8	4	4	25
P2/A5	16	8	8	50
P3/A6	32	16	16	100

Step 3 - Assign a workspace to a capacity

- Embed for your customers
- Embed for your organization

Once you create a capacity, you can assign your app workspace to that capacity.

To assign a capacity to a workspace using a service principal, use the Power BI REST API. When you're using the Power BI REST APIs, make sure to use the service principal object ID.

> **NOTE**
> You can also import paginated reports into a *new workspace* using the Power BI REST APIs.

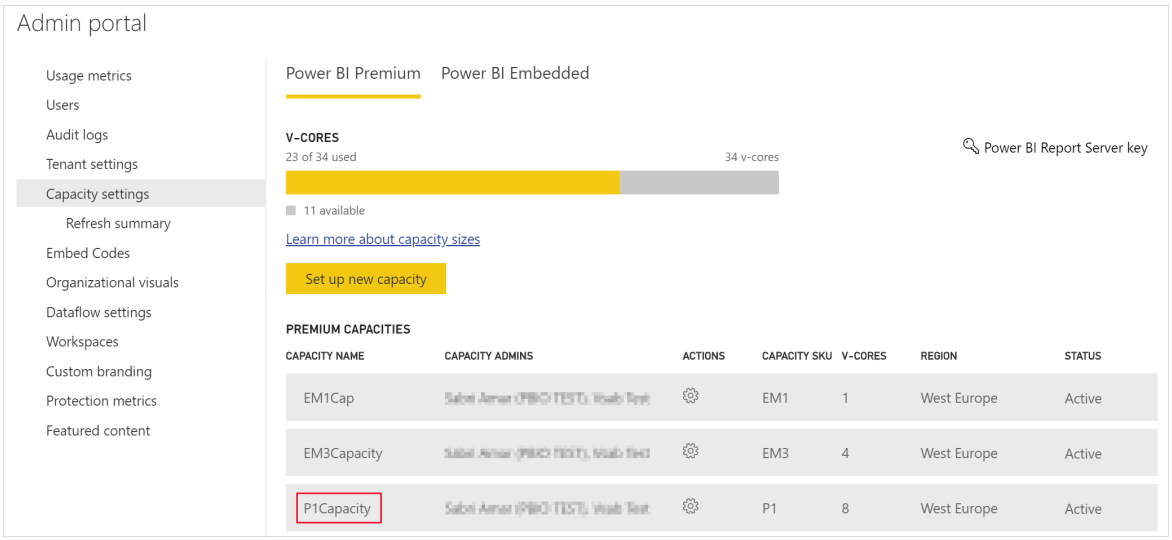
Step 4 - Enable paginated reports workload

NOTE

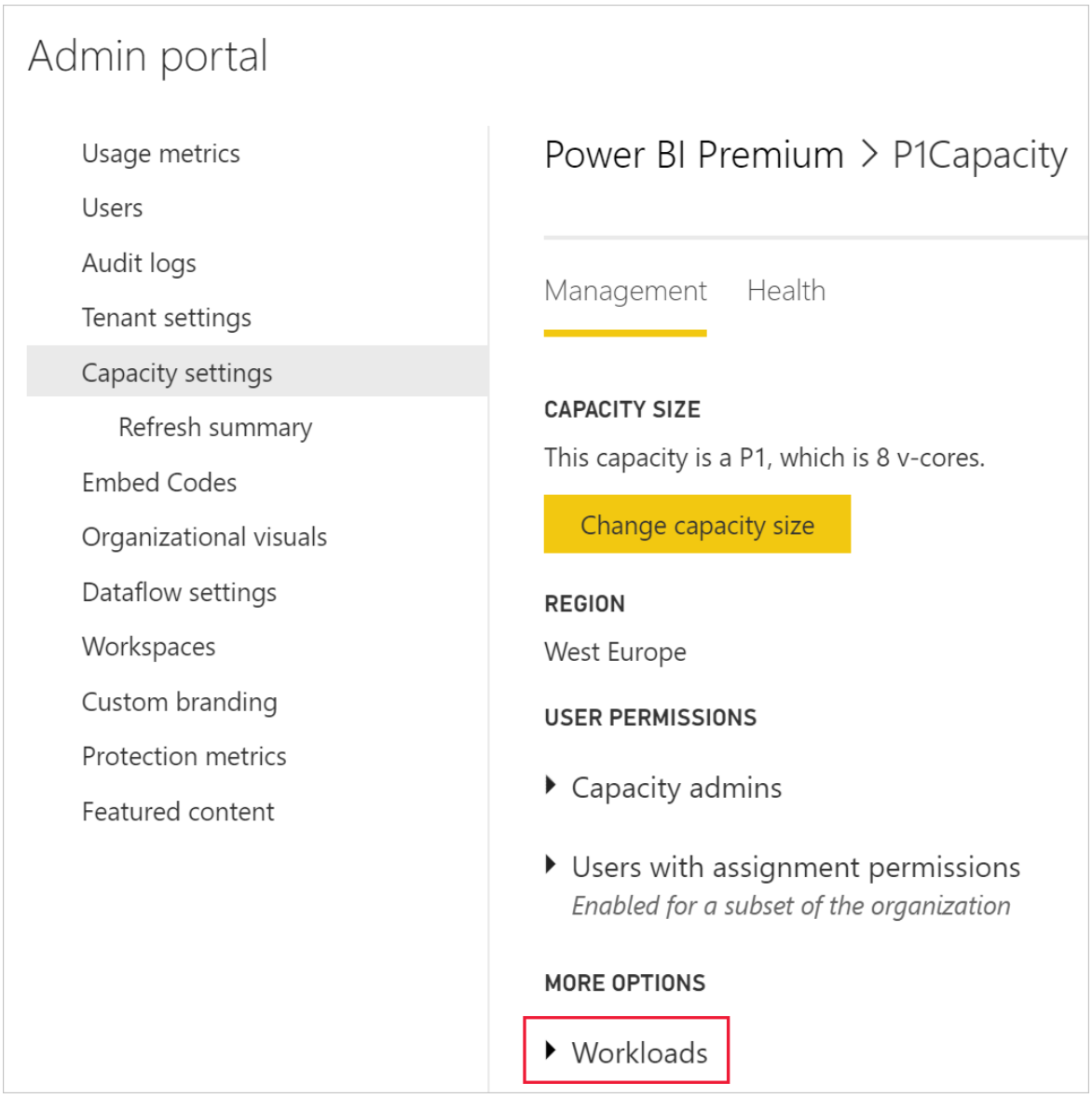
This step is only necessary for **Embedded Gen1**. If your capacity is Gen2, continue to Step 5.

After creating a capacity and assigning your workspace to it, you need to enable the paginated report workload on your capacity.

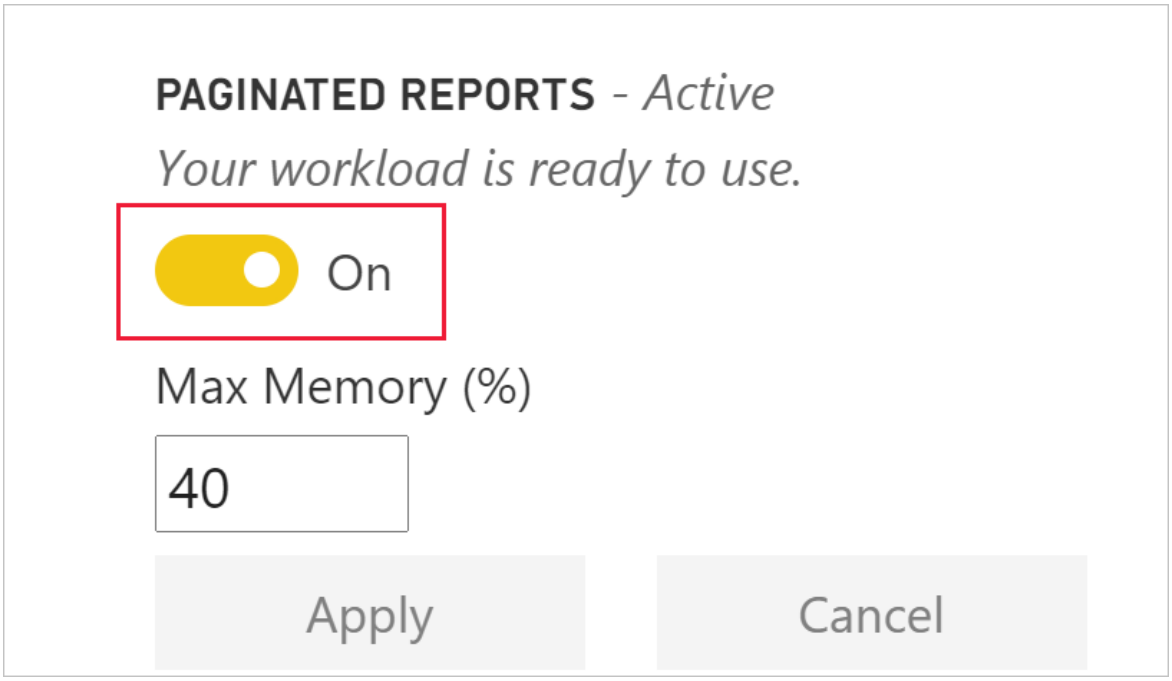
1. Sign into Power BI > Admin portal > Capacity settings.
2. Select the capacity that has the workspace you want to add a paginated report to.



3. Expand **Workloads**.



4. Activate the paginated reports workload.



Step 5 - Create and upload your paginated report

You can create your paginated report using Power BI Report Builder, and then upload the report to the service.

> **NOTE**
> The user uploading the paginated report needs a Power BI Pro or Premium Per User (PPU) license to publish to a workspace.

Step 6 - Embed content using the sample application

- Embed for your customers
- Embed for your organization

Follow the instructions in the embed content for your customers tutorial. Skip Step 4 - Create and publish a Power BI report and use the paginated report you uploaded, instead of the sample report suggested in the tutorial.

- To use a **Power BI dataset** as a data source:
  - In the Power BI portal, set the **XMLA endpoint** to *Read Only* or *Read Write* as described in enable read-write for a Premium capacity. You only need to do this once per capacity.
  - Generate a multi-resource embed token with the dataset ID specified in the request, and the XmlaPermissions set to *Read Only*.
- To use a **Single Sign-on (SSO)** enabled data source:

  SSO-enabled data sources are supported if they're either directly connected to the paginated report, or connected to a Power BI dataset which is the data source of the paginated report. When embedding a paginated report with SSO-enabled data sources, the identity blob for the data source must be provided in the `DatasourceIdentity` when you generate a multi-resource embed token.

Considerations and limitations

- Embed for your customers limitations
- Embed for your organization limitations

- You must use a **service principal**. Master user isn't supported.
- Premium Per User (PPU) isn't supported.
- When embedding a paginated report with a Power BI dataset, both the paginated report and the Power BI dataset must reside in a Premium per capacity or Embedded workspace (they can reside in two different workspaces), and the user generating the embed token should have *Write* permissions in the workspaces of the report and the dataset.

For a full list of supported datasets and their authentication methods, see Supported data sources for Power BI paginated reports.

Next steps

Tutorial: Embed a Power BI report in an application for your organization

Capacity and SKUs in Power BI embedded analytics

Considerations when generating an embed token

Use the Power BI SDK with service principal profiles

6/30/2022 • 2 minutes to read • Edit Online

This article explains how to use the SDK with service principal profiles. There are two ways to connect a Power BI client to a service principal profile. You can:

- Create a client with a profile object ID
- Specify the profile ID in the API request call

Once the client is associated with a profile, you can get the current service principal profile from the Power BI client.

Create a Power BI client with a service principal profile

```
var profileObjectId = new Guid("81f24a6d-7ebb-4478-99c7-2c36f7870a26");
var powerBIclient = new PowerBIClient(credentials, profileObjectId: profileObjectId);
```

When you create a Power BI client with the profile object ID, every API call that uses the client is issued with the `X-PowerBI-profile-id` in the request header.

For example,

```
GET https://powerbiapi.analysis-df.windows.net/v1.0/myorg/groups


Authorization: Bearer eyJ0eXAiO.....5U_g
X-PowerBI-profile-id: 81f24a6d-7ebb-4478-99c7-2c36f7870a26
```

Set profile on API request call

Alternatively, you can specify the profile ID in the API request by using the `customHeaders` property in the API's overloaded PowerBIClient method `WithHttpMessagesAsync`.

```
var powerBIclient = new PowerBIClient(credentials);
var profileHeader = new Dictionary<string, List<string>>();
profileHeader.Add("X-PowerBI-profile-id", new List<string> { "81f24a6d-7ebb-4478-99c7-2c36f7870a26" });
var groups = await powerBIclient.Groups.GetGroupsWithHttpMessagesAsync(customHeaders: profileHeader);
```

For example,

```
GET https://powerbiapi.analysis-df.windows.net/v1.0/myorg/groups

Authorization: Bearer eyJ0eXAiO.....5U_g
X-PowerBI-profile-id: 81f24a6d-7ebb-4478-99c7-2c36f7870a26
```

In the case, the profile header will *not* be added to the client default headers. You need to specify it with every API request.

Make sure you avoid duplications. For example, creating a client with a profile object ID and then specifying the header with the API request will result in unauthorized errors.

Get the current service principal profile from Power BI client

To retrieve the current service principal profile from the SDK client, call `GetServicePrincipalProfileObjectId`.

```
var profileObjectId = new Guid("81f24a6d-7ebb-4478-99c7-2c36f7870a26");
var powerBIclient = new PowerBIClient(credentials, profileObjectId: profileObjectId);
var currentProfileObjectId = powerBIclient.GetServicePrincipalProfileObjectId();
```

Considerations and Limitations

Service principal profiles are not supported with Azure Analysis Services (AAS) in live connection mode.

Next steps

Service principal profiles in Power BI Embedded

Move your embedded app to production

6/30/2022 • 3 minutes to read • Edit Online

IMPORTANT

This article only applies to *embed for your customers* applications. If you're using the *embed for your organization* scenario, you need to use either a Pro or Premium license.

After you've completed developing your application, to move to production you'll need to back your workspace with a capacity. Note that **all workspaces** (the ones containing the reports or dashboards, and the ones containing the datasets) must be assigned to a capacity.

Create a capacity

By creating a capacity, you can take advantage of having a resource for your customers. There are two types of capacities you can choose from:

- **Power BI Premium** - A tenant-level Microsoft 356 subscription available in two SKU families, *EM* and *P*. When embedding Power BI content, this solution is referred to as *Power BI embedding*. For more information regarding this subscription, see What is Power BI Premium?
- **Azure Power BI Embedded** - You can purchase a capacity from the Microsoft Azure portal. This subscription uses the *A* SKUs. For details on how to create a Power BI Embedded capacity, see Create Power BI Embedded capacity in the Azure portal.

  NOTE

  With A SKUs, you can't access Power BI content with a FREE Power BI license.

Capacity specifications

The table below describes the resources and limits of each SKU. To determine which capacity best fits your needs, see the which SKU should I purchase for my scenario table.

CAPACITY NODES	TOTAL V-CORES	BACKEND V-CORES	RAM (GB)	FRONTEND V-CORES	DIRECTQUERY /LIVE CONNECTION (PER SEC)	MODEL REFRESH PARALLELISM
EM1/A1	1	0.5	2.5	0.5	3.75	1
EM2/A2	2	1	5	1	7.5	2
EM3/A3	4	2	10	2	15	3
P1/A4	8	4	25	4	30	6
P2/A5	16	8	50	8	60	12
P3/A6	32	16	100	16	120	24

CAPACITY NODES	TOTAL V-CORES	BACKEND V-CORES	RAM (GB)	FRONTEND V-CORES	DIRECTQUERY /LIVE CONNECTION (PER SEC)	MODEL REFRESH PARALLELISM
P4/A7	64	32	200	32	240	48
P5/A8	128	64	400	64	480	96

Development testing

For development testing, you can use embed trial tokens with a Pro license. To embed in a production environment, use a capacity.

The number of embed trial tokens a Power BI *service principal* or *master user* (master account) can generate, is limited. Use the Available Features API to check the percentage of your current embedded usage. The usage amount is displayed per service principal or master account.

If you run out of embed tokens while testing, you need to purchase a Power BI Embedded or Premium capacity. There's no limit to the number of embed tokens you can generate with a capacity.

Assign a workspace to a capacity

Once you create a capacity, you can assign your workspace to that capacity.

Each workspace that contains a Power BI item related to the embedded content (including datasets, reports, and dashboards), must be assigned to capacities. For example, if an embedded report and the dataset bound to it reside in different workspaces, both workspaces must be assigned to capacities.

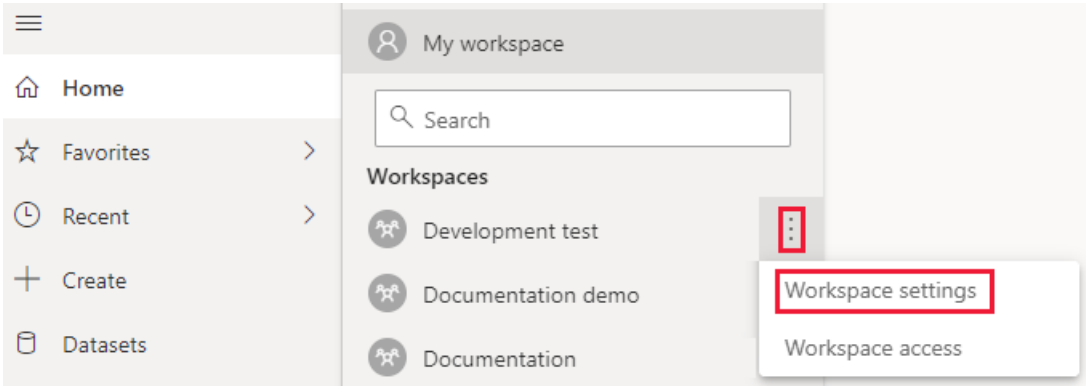
Assign a workspace to a capacity using a service principal

To assign a workspace to a capacity using a service principal, use the Power BI REST API. When you're using the Power BI REST APIs, make sure to use the service principal object ID.

Assign a workspace to a capacity using a master user

You can also assign a workspace to a capacity from the settings of that workspace using a **master user**. The master user must have admin permissions to that workspace, and also capacity assignment permissions to that capacity.

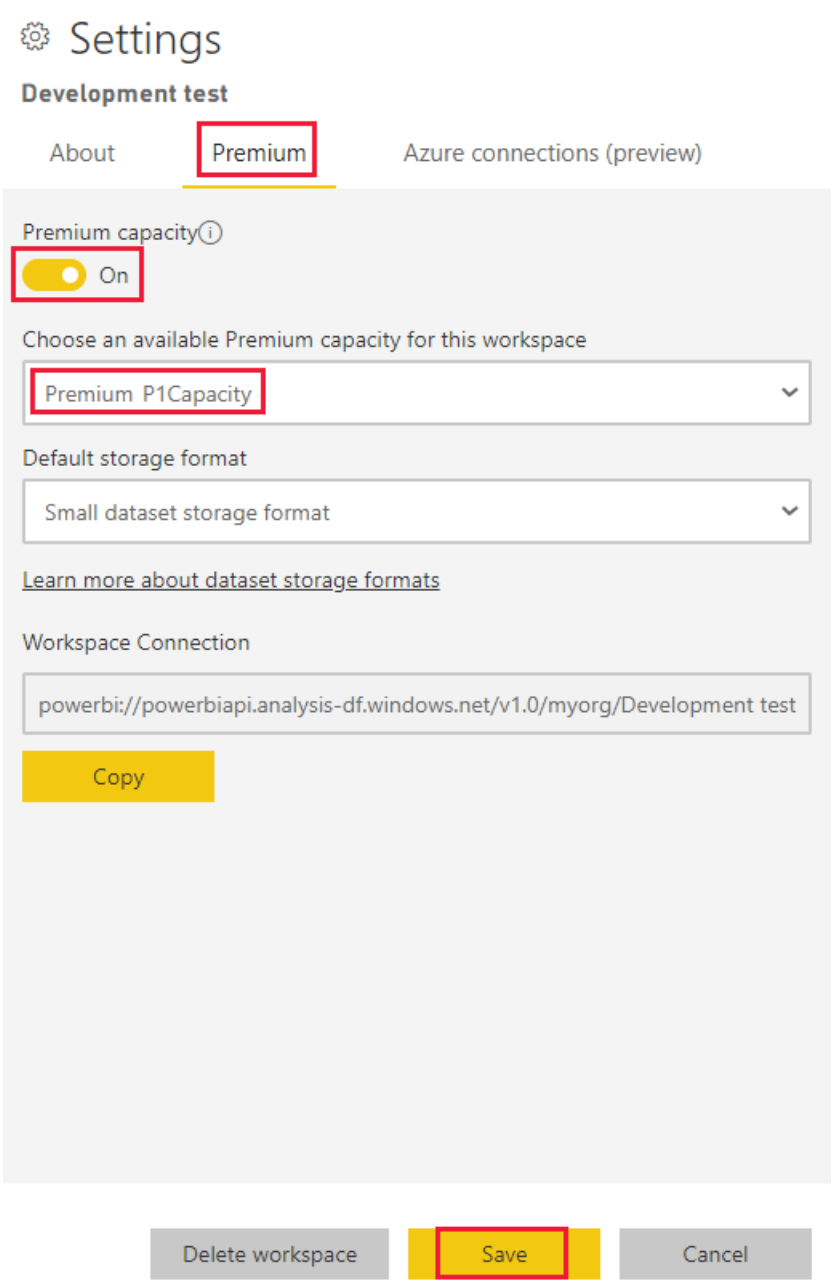
1. Within the **Power BI service**, expand workspaces and select the ellipsis for the workspace you're using for embedding your content. Then select **Workspace settings**.



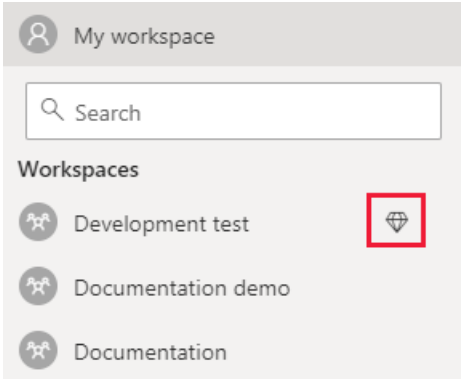
2. Select the **Premium** tab, and do the following:

   - Enable **Capacity**.

- Select the capacity you created.
- Select **Save**.



After assigning your workspace to a capacity, a diamond appears next to it



Next steps

Capacity and SKUs in Power BI embedded analytics

Export Power BI report to file

6/30/2022 • 12 minutes to read • Edit Online

The `exportToFile` API enables exporting a Power BI report by using a REST call. The following file formats are supported:

- **.pptx** (PowerPoint)
- **.pdf**
- **.png**
  - When you export to a .png, a report with multiple pages is compressed into a .zip file
  - Each file in the .zip represents a report page
  - The page names are the same as the return values of the Get Pages or Get Pages in Group APIs

> **IMPORTANT**
> The `exporttofile` API is only available in for Gen2 capacities. If you are using a gen1 capacity, upgrade to Gen2

Usage examples

You can use the export feature in several ways. Here are a couple of examples:

- **Send to print button** - In your application, create a button that when clicked on triggers an export job. The job can export the viewed report as a .pdf or a .pptx. When it's complete, the user can receive the file as a download. Using bookmarks you can export the report in a specific state, including configured filters, slicers, and other settings. As the API is asynchronous, it may take some time for the file to be available.
- **Email attachment** - Send an automated email at set intervals, with an attached .pdf report. This scenario can be useful if you want to automate sending a weekly report to executives. For more information, see Export and email a Power BI report with Power Automate

Using the API

Before using the API, verify that the following admin tenant settings are enabled:

- **Export reports as PowerPoint presentations or PDF documents** - Enabled by default.
- **Export reports as image files** - Required only for *.png* and disabled by default.

The API is asynchronous. When the exportToFile API is called, it triggers an export job. After triggering an export job, use polling to track the job, until it's complete.

During polling, the API returns a number that represents the amount of work completed. The work in each export job is calculated based on the total of exports in the job. An export includes exporting a single visual, or a page with or without bookmarks. All exports have the same weight. If, for example, your export job includes exporting a report with 10 pages, and the polling returns 70, it means the API has processed seven out of the 10 pages in the export job.

When the export is complete, the polling API call returns a Power BI URL for getting the file. The URL will be available for 24 hours.

Supported features

This section describes the operation of the following supported features:

- Selecting which pages to print
- Exporting a page or a single visual
- Bookmarks
- Filters
- Authentication
- Row Level Security (RLS)
- Data protection
- Localization

Selecting which pages to print

Specify the pages you want to print according to the Get Pages or Get Pages in Group return value. You can also specify the order of the pages you're exporting.

Exporting a page or a single visual

You can specify a page or single visual to export. Pages can be exported with or without bookmarks.

Depending on the type of export, you need to pass different attributes to the ExportReportPage object. The table below specifies which attributes are required for each export job.

> **NOTE**
> Exporting a single visual has the same weight as exporting a page (with or without bookmarks). This means that in terms of system calculations, both operations carry the same value.

ATTRIBUTE	PAGE	SINGLE VISUAL	COMMENTS
`bookmark`	Optional	⊗	Use to export a page in a specific state
`pageName`	✓	✓	Use the GetPages REST API or the `getPages` client API. For more information, see Get pages and visuals.
`visualName`	⊗	✓	There are two ways to get the name of the visual: • Use the `getVisuals` client API. For more information, see Get pages and visuals. • Listen and log the *visualClicked* event, which is triggered when a visual is selected. For more information, see How to handle events .

Bookmarks

Bookmarks can be used to save a report in a specific configuration, including applied filters and the state of the report's visuals. You can use the exportToFile API to programmatically export a report's bookmark, in two ways:

- Export an existing bookmark

To export an existing report bookmark, use the `name` property, a unique (case sensitive) identifier, which you can get using the bookmarks JavaScript API.

- **Export the report's state**

  To export the current state of the report, use the `state` property. For example, you can use the bookmark's `bookmarksManager.capture` method to capture the changes a specific user made to a report, and then export it in its current state using `capturedBookmark.state`.

> **NOTE**
> Personal bookmarks and persistent filters are not supported.

Filters

Using `reportLevelFilters` in PowerBIReportExportConfiguration, you can export a report in a filtered condition.

To export a filtered report, insert the URL query string parameters you want to use as your filter, to ExportFilter. When you enter the string, you must remove the `?filter=` part of the URL query parameter.

The table below includes a few syntax examples of strings you can pass to `ExportFilter`.

FILTER	SYNTAX	EXAMPLE
A value in a field	Table/Field eq 'value'	Store/Territory eq 'NC'
Multiple values in a field	Table/Field in ('value1', 'value2')	Store/Territory in ('NC', 'TN')
A distinct value in one field, and a different distinct value in another field	Table/Field1 eq 'value1' and Table/Field2 eq 'value2'	Store/Territory eq 'NC' and Store/Chain eq 'Fashions Direct'

Authentication

You can authenticate using a user (or master user) or a service principal.

Row Level Security (RLS)

With Row Level Security (RLS), you can export a report showing data that's only visible to certain users. For example, if you're exporting a sales report that's defined with regional roles, you can programmatically filter the report so that only a certain region is displayed.

To export using RLS, you must have the following permissions:

- Write and reshare permissions for the dataset the report is connected to
- If the report resides on a v1 workspace, you need to be the workspace admin
- If the report resides on a v2 workspace, you need to be a workspace member or admin

Data protection

The .pdf and .pptx formats support sensitivity labels. If you export a report with a sensitivity label to a .pdf or a .pptx, the exported file will display the report with its sensitivity label.

A report with a sensitivity label can't be exported to a .pdf or a .pptx using a service principal.

Localization

When using the `exportToFile` API, you can pass your desired local. The localization settings affect the way the report is displayed, for example by changing formatting according to the selected local.

Concurrent requests

`exportToFile` supports concurrent export job requests. The table below shows the number of jobs you can run at the same time, depending on the SKU your report resides on. Concurrent requests refer to report pages. For example, 55 pages in one export request on an A4 SKU, will be processed concurrently. This process will take roughly the same amount of time as sending 55 export requests with one page each.

A job that exceeds its number of concurrent requests doesn't terminate. For example, if you export 30 pages in an A2 SKU, the first 25 jobs will run, and the remaining five will wait for the next execution cycle.

Only five pages of a report are processed concurrently. For example, if you're exporting a report with 50 pages, the export job will be processed in 10 sequential intervals. When optimizing your export job, you may want to consider executing a few jobs in parallel. For example, if you have an A1 SKU with a limit of processing 20 max concurrent pages per export, you can process four 50 page reports at the same time. Only five pages from each job are being processed at a given time. As a result, the overall time to complete the four jobs will be shorter than exporting the entire report in one job.

> **NOTE**
> Exporting a Power BI report to file using the `exportToFile` API, is not supported for Premium Per User (PPU).

AZURE SKU	OFFICE SKU	MAXIMUM CONCURRENT REPORT PAGES
A1	EM1	20
A2	EM2	25
A3	EM3	35
A4	P1	55
A5	P2	95
A6	P3	175
A7[1]	P4[1]	200
A8[1]	P5[1]	200

[1] SKUs greater than 100 GB aren't available in all regions. To request using these SKUs in regions where they're not available, contact your Microsoft account manager.

Code examples

When you create an export job, there are four steps to follow:

1. Sending an export request.
2. Polling.
3. Getting the file.
4. Using the file stream.

This section provides examples for each step.

Step 1 - sending an export request

The first step is sending an export request. In this example, an export request is sent for a specific page.

```
private async Task<string> PostExportRequest(
    Guid reportId,
    Guid groupId,
    FileFormat format,
    IList<string> pageNames = null, /* Get the page names from the GetPages REST API */
    string urlFilter = null)
{
    var powerBIReportExportConfiguration = new PowerBIReportExportConfiguration
    {
        Settings = new ExportReportSettings
        {
            Locale = "en-us",
        },
        // Note that page names differ from the page display names
        // To get the page names use the GetPages REST API
        Pages = pageNames?.Select(pn => new ExportReportPage(Name = pn)).ToList(),
        // ReportLevelFilters collection needs to be instantiated explicitly
        ReportLevelFilters = !string.IsNullOrEmpty(urlFilter) ? new List<ExportFilter>() { new
ExportFilter(urlFilter) } : null,

    };

    var exportRequest = new ExportReportRequest
    {
        Format = format,
        PowerBIReportConfiguration = powerBIReportExportConfiguration,
    };

    // The 'Client' object is an instance of the Power BI .NET SDK
    var export = await Client.Reports.ExportToFileInGroupAsync(groupId, reportId, exportRequest);

    // Save the export ID, you'll need it for polling and getting the exported file
    return export.Id;
}
```

Step 2 - polling

After you've sent an export request, use polling to identify when the export file you're waiting for is ready.

```
private async Task<HttpOperationResponse<Export>> PollExportRequest(
    Guid reportId,
    Guid groupId,
    string exportId /* Get from the PostExportRequest response */,
    int timeOutInMinutes,
    CancellationToken token)
{
    HttpOperationResponse<Export> httpMessage = null;
    Export exportStatus = null;
    DateTime startTime = DateTime.UtcNow;
    const int c_secToMillisec = 1000;
    do
    {
        if (DateTime.UtcNow.Subtract(startTime).TotalMinutes > timeOutInMinutes ||
token.IsCancellationRequested)
        {
            // Error handling for timeout and cancellations
            return null;
        }

        // The 'Client' object is an instance of the Power BI .NET SDK
        httpMessage = await Client.Reports.GetExportToFileStatusInGroupWithHttpMessagesAsync(groupId,
reportId, exportId);
        exportStatus = httpMessage.Body;

        // You can track the export progress using the PercentComplete that's part of the response
        SomeTextBox.Text = string.Format("{0} (Percent Complete : {1}%)", exportStatus.Status.ToString(),
exportStatus.PercentComplete);
        if (exportStatus.Status == ExportState.Running || exportStatus.Status == ExportState.NotStarted)
        {
            // The recommended waiting time between polling requests can be found in the RetryAfter header
            // Note that this header is not always populated
            var retryAfter = httpMessage.Response.Headers.RetryAfter;
            var retryAfterInSec = retryAfter.Delta.Value.Seconds;
            await Task.Delay(retryAfterInSec * c_secToMillisec);
        }
    }
    // While not in a terminal state, keep polling
    while (exportStatus.Status != ExportState.Succeeded && exportStatus.Status != ExportState.Failed);

    return httpMessage;
}
```

Step 3 - getting the file

Once polling returns a URL, use this example to get the received file.

```
private async Task<ExportedFile> GetExportedFile(
    Guid reportId,
    Guid groupId,
    Export export /* Get from the PollExportRequest response */)
{
    if (export.Status == ExportState.Succeeded)
    {
        // The 'Client' object is an instance of the Power BI .NET SDK
        var fileStream = await Client.Reports.GetFileOfExportToFileAsync(groupId, reportId, export.Id);
        return new ExportedFile
        {
            FileStream = fileStream,
            FileSuffix = export.ResourceFileExtension,
        };
    }
    return null;
}

public class ExportedFile
{
    public Stream FileStream;
    public string FileSuffix;
}
```

Step 4 - Using the file stream

When you have the file stream, you can handle it in the way that best fits your needs. For example, you can email it or use it to download the exported reports.

End-to-end example

This is an end-to-end example for exporting a report. This example includes the following stages:

1. Sending the export request.
2. Polling.
3. Getting the file.

```
private async Task<ExportedFile> ExportPowerBIReport(
 Guid reportId,
 Guid groupId,
 FileFormat format,
 int pollingtimeOutInMinutes,
 CancellationToken token,
 IList<string> pageNames = null,  /* Get the page names from the GetPages REST API */
    string urlFilter = null)
{
 const int c_maxNumberOfRetries = 3; /* Can be set to any desired number */
 const int c_secToMillisec = 1000;
 try
 {
  Export export = null;
  int retryAttempt = 1;
  do
  {
   var exportId = await PostExportRequest(reportId, groupId, format, pageNames, urlFilter);
   var httpMessage = await PollExportRequest(reportId, groupId, exportId, pollingtimeOutInMinutes, token);
   export = httpMessage.Body;
   if (export == null)
   {
    // Error, failure in exporting the report
    return null;
   }
   if (export.Status == ExportState.Failed)
   {
    // Some failure cases indicate that the system is currently busy. The entire export operation can be
```

```
retried after a certain delay
    // In such cases the recommended waiting time before retrying the entire export operation can be found
in the RetryAfter header
    var retryAfter = httpMessage.Response.Headers.RetryAfter;
    if(retryAfter == null)
    {
        // Failed state with no RetryAfter header indicates that the export failed permanently
        return null;
                }

                var retryAfterInSec = retryAfter.Delta.Value.Seconds;
                await Task.Delay(retryAfterInSec * c_secToMillisec);
            }
        }
        while (export.Status != ExportState.Succeeded && retryAttempt++ < c_maxNumberOfRetries);

        if (export.Status != ExportState.Succeeded)
        {
            // Error, failure in exporting the report
            return null;
        }

        var exportedFile = await GetExportedFile(reportId, groupId, export);

        // Now you have the exported file stream ready to be used according to your specific needs
        // For example, saving the file can be done as follows:
        /*
            var pathOnDisk = @"C:\temp\" + export.ReportName + exportedFile.FileSuffix;

            using (var fileStream = File.Create(pathOnDisk))
            {
                exportedFile.FileStream.CopyTo(fileStream);
            }
        */

        return exportedFile;
    }
    catch
    {
        // Error handling
        throw;
    }
}
```

Considerations and limitations

- The `exporttofile` API is only available in for Gen2 capacities. If you are using a gen1 capacity, upgrade to Gen2.
- An export API operation load will be evaluated as a slow-running background operation, as described in Premium Gen2 capacity load evaluation.
- The report you're exporting must reside on a Premium or Embedded capacity.
- The dataset of the report you're exporting must reside on a Premium or Embedded capacity.
- Exported reports can't exceed a file size of 250 MB.
- When exporting to .png, sensitivity labels aren't supported.
- The number of exports (single visuals or report pages) that can be included in an exported report is 50 (not including exporting paginated reports). If the request includes more exports, the API returns an error and the export job is canceled.
- Personal bookmarks and persistent filters aren't supported.
- Dynamic Dataset binding isn't supported.
- Exporting a Power BI report to file using the `exportToFile` API, isn't supported for **Premium Per User (PPU)**.

- The Power BI visuals listed below aren't supported. When you export a report containing these visuals, the parts of the report that contain these visuals won't render, and will display an error symbol.
  - Uncertified Power BI custom visuals
  - R visuals
  - PowerApps
  - Python visuals
  - Power Automate
  - Paginated report visual
  - Visio

Next steps

Review how to embed content for your customers and your organization:

Export paginated report to file

Embed for your customers

Embed for your organization

Export and email a Power BI report with Power Automate

Export paginated report to file

6/30/2022 • 6 minutes to read • Edit Online

The `exportToFile` API enables exporting a Power BI paginated report by using a REST call. The following file formats are supported:

- **.pptx** (PowerPoint)
- **.pdf** (and Accessible PDF, or PDF/UA)
- **.xlsx** (Excel)
- **.docx** (Word)
- **.csv**
- **.xml**
- **.mhtml**
- **Image** When exporting to an image, set the image format via the `OutputFormat` format setting. The supported `OutputFormat` values are:
  - *.tiff* (default)
  - *.bmp*
  - *.emf*
  - *.gif*
  - *.jpeg*
  - *.png*

Usage examples

You can use the export feature in a variety of ways. Here are a couple of examples:

- **Send to print button** - In your application, create a button that when clicked on triggers an export job. The job can export the viewed report as a .pdf or a .pptx. When it's complete, the user can receive the file as a download. Using report parameters and format settings you can export the report in a specific state, including filtered data, custom page sizes, and other format-specific settings. As the API is asynchronous, it may take some time for the file to be available.
- **Email attachment** - Send an automated email at set intervals, with an attached .pdf report. This scenario can be useful if you want to automate sending a weekly report to executives.

Using the API

The API is asynchronous. When the exportToFile API is called, it triggers an export job. After triggering an export job, use polling to track the job, until it's complete.

When the export is complete, the polling API call returns a Power BI URL for getting the file. The URL will be available for 24 hours.

Supported features

Format settings

Specify a variety of format settings for each file format. The supported properties and values are equivalent to Device Info parameters for paginated report URL parameters.

Here are two examples. The first is for exporting the first four pages of a report using the report page size to a .pptx file. The second example is for exporting the third page of a report to a .jpeg file.

Exporting the first four pages to a .pptx

```
{
      "format": "PPTX",
      "paginatedReportConfiguration":{
            "formatSettings":{
                  "UseReportPageSize": "true",
                  "StartPage": "1",
                  "EndPage": "4"
            }
      }
}
```

Exporting the third page to a .jpeg

```
{
      "format": "IMAGE",
      "paginatedReportConfiguration":{
            "formatSettings":{
                  "OutputFormat": "JPEG",
                  "StartPage": "3",
                  "EndPage": "3"
            }
      }
}
```

Report parameters

You can use the `exportToFile` API to programmatically export a report with a set of report parameters. This is done using report parameter capabilities.

Here's an example for setting report parameter values.

```
{
      "format": "PDF",
      "paginatedReportConfiguration":{
            "parameterValues":[
                  {"name": "State", "value": "WA"},
                  {"name": "City", "value": "Seattle"},
                  {"name": "City", "value": "Bellevue"},
                  {"name": "City", "value": "Redmond"}
            ]
      }
}
```

Authentication

You can authenticate using a user (or master user) or a service principal.

Row Level Security (RLS)

When using a Power BI dataset that has Row Level Security (RLS) defined as a data source, you can export a report showing data that's only visible to certain users. For example, if you're exporting a sales report that's defined with regional roles, you can programmatically filter the report so that only a certain region is displayed.

To export using RLS, you must have read permission for the Power BI dataset the report is using as a data source.

Here's an example of supplying an effective user name for RLS.

```
{
      "format": "PDF",
      "paginatedReportConfiguration":{
            "identities": [
                  {"username": "john@contoso.com"}
            ]
      }
}
```

Single Sign-on SQL and Dataverse (SSO)

In Power BI, you have the option to set OAuth with SSO. When you do, the credentials for the user viewing the report are used to retrieve data. The access token in the request header isn't used to access the data. The token must be passed in with the effective identity in the post body.

Getting the correct access token for the resource that you want to access can sometimes be tricky.

- For Azure SQL, the resource is `https://database.windows.net`.
- For Dataverse, the resource is the `https://` address for your environment. For example, `https://contoso.crm.dynamics.com`.

Access the token API using the AuthenticationContext.AcquireTokenAsync method.

Here's an example for supplying an effective identity (user name) with an access token.

```
{
       "format":"PDF",
       "paginatedReportConfiguration":{
          "formatSettings":{
             "AccessiblePDF":"true",
             "PageHeight":"11in",
             "PageWidth":"8.5in",
             "MarginBottom":"2in"
          },
          "identities":[
             {
                "username":"john@contoso.com",
                "identityBlob": {
                "value": "eyJ0eX....full access token"
         }
        }
     ]
   }
}
```

PPU concurrent requests

The `exportToFile` API allows one request in a five-minute window when using Premium Per User (PPU). Multiple (greater than one) requests within a five-minute window will result in a *Too Many Requests* (429) error.

Code examples

The Power BI API SDK used in the code examples can be download here.

When you create an export job, there are three steps to follow:

1. Sending an export request.
2. Polling.
3. Getting the file.

This section provides examples for each step.

Step 1 - sending an export request

The first step is sending an export request. In this example, an export request is sent for a specific page range, size, and report parameter values.

```
private async Task<string> PostExportRequest(
    Guid reportId,
    Guid groupId)
{
    // For documentation purposes the export configuration is created in this method
    // Ordinarily, it would be created outside and passed in
    var paginatedReportExportConfiguration = new PaginatedReportExportConfiguration()
    {
        FormatSettings = new Dictionary<string, string>()
        {
            {"PageHeight", "14in"},
            {"PageWidth", "8.5in" },
            {"StartPage", "1"},
            {"EndPage", "4"},
        },
        ParameterValues = new List<ParameterValue>()
        {
            { new ParameterValue() {Name = "State", Value = "WA"} },
            { new ParameterValue() {Name = "City", Value = "Redmond"} },
        },
    };

    var exportRequest = new ExportReportRequest
    {
        Format = FileFormat.PDF,
        PaginatedReportExportConfiguration = paginatedReportExportConfiguration,
    };

    var export = await Client.Reports.ExportToFileInGroupAsync(groupId, reportId, exportRequest);

    // Save the export ID, you'll need it for polling and getting the exported file
    return export.Id;
}
```

Step 2 - polling

After you've sent an export request, use polling to identify when the export file you're waiting for is ready.

```
private async Task<Export> PollExportRequest(
    Guid reportId,
    Guid groupId,
    string exportId /* Get from the ExportToAsync response */,
    int timeOutInMinutes,
    CancellationToken token)
{
    Export exportStatus = null;
    DateTime startTime = DateTime.UtcNow;
    const int secToMillisec = 1000;
    do
    {
        if (DateTime.UtcNow.Subtract(startTime).TotalMinutes > timeOutInMinutes ||
token.IsCancellationRequested)
        {
            // Error handling for timeout and cancellations
            return null;
        }

        var httpMessage =
            await Client.Reports.GetExportToFileStatusInGroupWithHttpMessagesAsync(groupId, reportId,
exportId);

        exportStatus = httpMessage.Body;
        if (exportStatus.Status == ExportState.Running || exportStatus.Status == ExportState.NotStarted)
        {
            // The recommended waiting time between polling requests can be found in the RetryAfter header
            // Note that this header is only populated when the status is either Running or NotStarted
            var retryAfter = httpMessage.Response.Headers.RetryAfter;
            var retryAfterInSec = retryAfter.Delta.Value.Seconds;

            await Task.Delay(retryAfterInSec * secToMillisec);
        }
    }
    // While not in a terminal state, keep polling
    while (exportStatus.Status != ExportState.Succeeded && exportStatus.Status != ExportState.Failed);

    return exportStatus;
}
```

Step 3 - getting the file

Once polling returns a URL, use this example to get the received file.

```
private async Task<ExportedFile> GetExportedFile(
    Guid reportId,
    Guid groupId,
    Export export /* Get from the GetExportStatusAsync response */)
{
    if (export.Status == ExportState.Succeeded)
    {
        var httpMessage =
            await Client.Reports.GetFileOfExportToFileInGroupWithHttpMessagesAsync(groupId, reportId,
export.Id);

        return new ExportedFile
        {
            FileStream = httpMessage.Body,
            ReportName = export.ReportName,
            FileExtension = export.ResourceFileExtension,
        };
    }

    return null;
}

public class ExportedFile
{
    public Stream FileStream;
    public string ReportName;
    public string FileExtension;
}
```

End-to-end example

This is an end-to-end example for exporting a report. This example includes the following stages:

1. Sending the export request.
2. Polling.
3. Getting the file.

```
private async Task<ExportedFile> ExportPaginatedReport(
    Guid reportId,
    Guid groupId,
    int pollingtimeOutInMinutes,
    CancellationToken token)
{
    try
    {
        var exportId = await PostExportRequest(reportId, groupId);

        var export = await PollExportRequest(reportId, groupId, exportId, pollingtimeOutInMinutes, token);
        if (export == null || export.Status != ExportState.Succeeded)
        {
           // Error, failure in exporting the report
            return null;
        }

        return await GetExportedFile(reportId, groupId, export);
    }
    catch
    {
        // Error handling
        throw;
    }
}
```

Considerations and limitations

- Exporting a paginated report that has a Power BI dataset as its data source, isn't supported for service principals.
- When exporting a paginated report with an effective identity, the username must be an existing user from your tenant's Azure Active Directory.
- Export of a report is limited to 60 minutes, which matches the life of the user access token.
- If you get a timeout error past the 60-minute mark while exporting large amounts of data, consider minimizing the data using appropriate filters.

Next steps

Review how to embed content for your customers and your organization:

Export report to file

Embed for your customers

Embed for your organization

Configure credentials programmatically for Power BI

6/30/2022 • 3 minutes to read • Edit Online

Follow the steps in this article, to configure credentials programmatically for Power BI.

> **NOTE**
>
> - The calling user must be a dataset owner, or a gateway admin. You can also use a service principal. For example, the service principal can be the dataset owner.
> - Cloud data sources and their corresponding credentials are managed at the user level.

Update credentials flow for data sources

1. Call Get Datasources to discover the data sources of the dataset. In the response body for each data source, are the type, connection details, gateway, and data source ID.

```
// Select a datasource
var datasources = pbiClient.Datasets.GetDatasources(datasetId).Value;
var datasource = datasources.First();
```

2. Build credentials string according to Update Datasource Examples depending on the credentials type.

   - .NET SDK v3
   - .NET SDK v2

```
var credentials =  new BasicCredentials(username: "username", password :"*****");
```

> **NOTE**
>
> If you're using cloud data sources don't follow the next steps in this section. Set the credentials using the gateway ID and data source ID obtained in step 1, by calling Update Datasource.

3. Call Get Gateway to retrieve the gateway public key.

```
var gateway = pbiClient.Gateways.GetGatewayById(datasource.GatewayId);
```

4. Encrypt the credentials.

   - .NET SDK v3
   - .NET SDK v2

```
var credentialsEncryptor = new AsymmetricKeyEncryptor(gateway.publicKey);
```

5. Build credential details with encrypted credentials.

   - .NET SDK v3

- .NET SDK v2

Use the AssymetricKeyEncryptor class with the public key retrieved in **Step 3**.

```
var credentialDetails = new CredentialDetails(
        credentials,
        PrivacyLevel.Private,
        EncryptedConnection.Encrypted,
        credentialsEncryptor);
```

6. Call Update Datasource to set credentials.

```
pbiClient.Gateways.UpdateDatasource(datasource.GatewayId.Value, datasource.DatasourceId.Value, new
UpdateDatasourceRequest(credentialDetails));
```

Configure a new data source for a data gateway

1. Install the On-premises data gateway on your machine.
2. Call Get Gateways to retrieve the gateway ID and public key.

```
// Select a gateway
var gateways = pbiClient.Gateways.GetGateways().Value;
var gateway = gateways.First();
```

3. Build credential details in the same way as described in update credentials flow for data sources, using the gateway public key retrieved in **step 2**.
4. Build the request body.

```
var request = new PublishDatasourceToGatewayRequest(
        dataSourceType: "SQL",
        connectionDetails: "{\"server\":\"myServer\",\"database\":\"myDatabase\"}",
        credentialDetails: credentialDetails,
        dataSourceName: "my sql datasource");
```

5. Call the Create Datasource API.

```
pbiClient.Gateways.CreateDatasource(gateway.Id, request);
```

Credential types

When you call Create Datasource or Update Datasource under an **enterprise on-prem gateway** using Power BI Rest API, the credentials value needs to be encrypted using the gateway's public key.

> **NOTE**
> .NET SDK v3 can also run the .NET SDK v2 examples listed below.

Windows and basic credentials

- .NET SDK v3
- .NET SDK v2

```
// Windows credentials
var credentials = new WindowsCredentials(username: "john", password: "*****");

// Or

// Basic credentials
var credentials = new BasicCredentials(username: "john", password: "*****");

var credentialsEncryptor = new AsymmetricKeyEncryptor(publicKey);
var credentialDetails = new CredentialDetails(credentials, PrivacyLevel.Private,
EncryptedConnection.Encrypted, credentialsEncryptor);
```

Key credentials

- .NET SDK v3
- .NET SDK v2

```
var credentials = new KeyCredentials("TestKey");
var credentialsEncryptor = new AsymmetricKeyEncryptor(publicKey);
var credentialDetails = new CredentialDetails(credentials, PrivacyLevel.Private,
EncryptedConnection.Encrypted, credentialsEncryptor);
```

OAuth2 credentials

- .NET SDK v3
- .NET SDK v2

```
var credentials = new OAuth2Credentials("TestToken");
var credentialsEncryptor = new AsymmetricKeyEncryptor(publicKey);
var credentialDetails = new CredentialDetails(credentials, PrivacyLevel.Private,
EncryptedConnection.Encrypted, credentialsEncryptor);
```

Anonymous credentials

- .NET SDK v3
- .NET SDK v2

```
var credentials = new AnonymousCredentials();
var credentialDetails = new CredentialDetails(credentials, PrivacyLevel.Private,
EncryptedConnection.NotEncrypted);
```

Troubleshooting

No gateway and data source ID found when calling get data sources

This issue means the dataset isn't bound to a gateway. When creating a new dataset, for each cloud connection a data source with no credentials is created automatically on the user's cloud gateway. This gateway is used to store the credentials for cloud connections.

After you create the dataset, an automatic binding is created between the dataset and a suitable gateway, which contains matching data sources for all connections. If there's no such gateway or multiple suitable gateways, the automatic binding fails.

If you're using on-premises datasets, create the missing on-premises data sources, and bind the dataset to a gateway manually by using Bind To Gateway.

To discover gateways that could be bound, use Discover Gateways.

Next steps

Power BI REST APIs

Embed Power BI content in Salesforce

6/30/2022 • 2 minutes to read • Edit Online

Salesforce is a world renowned customer relationship management (CRM) solution. Embedded analytics lets you embed Power BI content such as reports and dashboards in Salesforce.

This article links to two samples that show how to develop using Power BI embedded analytics in a Salesforce environment.

Prerequisites

You need to be familiar with both Power BI and Salesforce.

- You should have experience developing with Power BI embedded analytics.
- You should be familiar with the Salesforce environment; preferably with the salesforce developer experience (SFDX).

Embed for your customers

In the *embed for your customers* solution, also known as *app owns data*, you create an embedded application using your own Power BI account. Your customers don't need to use Power BI credentials to view and interact with the embedded content.

Apex class

The Salesforce *embed for your customers* solution uses a service principal, and is built on top of an Apex class named `PowerBiEmbedManager`.

Here are some advantages of the `PowerBiEmbedManager` Apex class:

- It's programmed to interact with both Azure AD and the Power BI REST APIs.
- It implements Client Credentials Flow when it interacts with Azure AD to acquire an app-only access token. App-only access tokens are important because they make it possible to call the Power BI REST APIs under the identity of a service principal, instead of calling under the identity of a user.

Lightning Aura

The Salesforce *embed for your customers* solution contains a Lightning Aura component named `powerBiReportAura`. When you add an instance of the `powerBiReportAura` component to a Lightning application page, you must configure it with a specific Power BI *Workspace ID* and *Report ID*. This design makes it possible to add multiple instances of the `powerBiReportAura` component, and configure each instance to embed a different Power BI report.

Access the *embed for your customers* Salesforce developer sample

To embed Power BI content using the *embed for your customers* Salesforce solution, follow the instructions in the SalesforceAppOwnsDataEmbedding GitHub repository.

Embed for your organization

In the embed for your organization solution, also known as *user owns data*, you create an embedded application that requires your customers to sign in with their own Power BI credentials. Signed in customers can view and interact with the embedded content according to their Power BI permissions.

Single page application

The Salesforce *embed for your organization* Salesforce solution uses a simple single page application (SPA), that implements Power BI reports. The solution is built using three essential files:

- index.html
- app.css
- app.js

Client-side libraries

In the Salesforce *embed for your organization* Salesforce solution, `powerbi.js` is used to access the Power BI embedded analytics Client APIs.

Access the *embed for your organization* Salesforce developer sample

To embed Power BI content using the *embed for your organization* solution, follow the instructions in the SalesforceUserOwnsDataEmbedding GitHub repository.

Next steps

Embed Power BI content into an application for your customers

Embed Power BI content into an application for your organization

Embed a Power BI report in an application for your organization

Power BI embedded analytics Client APIs

Power BI embedded analytics playground

Auto-install Power BI apps when embedding for your organization

6/30/2022 • 2 minutes to read • Edit Online

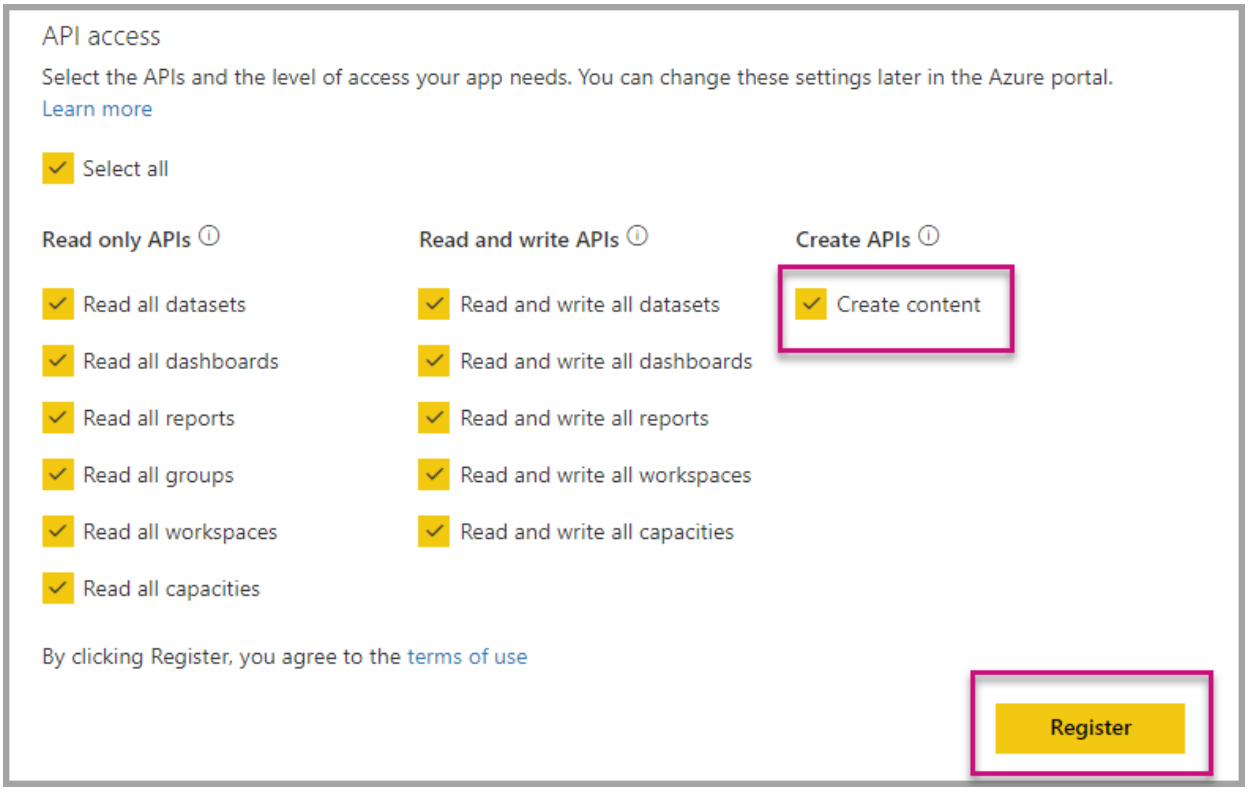
To embed content from an app, the user that is embedding must have access to the app. If the app is installed for the user, then embedding works smoothly. For more information, see Embed reports or dashboards from app. It's possible to define in PowerBI.com that all apps can be installed automatically. However, this action is done at the tenant level and applies to all apps.

Auto-install app on embedding

If a user has access to an app, but the app isn't installed, then embedding fails. So you can avoid these failures when embedding from an app, you can allow auto installation of the app upon embedding. This action means if the app the user tries to embed isn't installed, it's automatically installed for you. So the content you want gets embedded immediately, resulting in a smooth experience for the user.

Embed for Power BI users (User owns data)

To allow auto install of apps for your users, you need to give your application the 'Content Create' permission when registering your application, or add it if you already registered your app.



Next, you need to provide the app ID in the embed URL. To provide the app ID, the app creator first needs to install the app then use one of the supported Power BI REST API calls - Get Reports or Get Dashboards. Then the app creator needs to take the embed Url from the REST API response. The app ID appears in the URL if the content is from an app. After you have the embed URL, you can use it to embed regularly.

Secure Embed

To use auto install of apps, the app creator first needs to install the app then go to the app on PowerBI.com, navigate to the report, and get the link in a usual fashion. All other users with access to the app that can use the link can embed the report.

Considerations and limitations

- You can only embed reports and dashboards for this scenario.
- This feature is currently not supported for app owns data and SharePoint embed scenarios.

Create an Azure Active Directory tenant to use with Power BI

6/30/2022 • 3 minutes to read • Edit Online

Learn how to create a new Azure Active Directory (Azure AD) tenant for a custom application that calls Power BI REST APIs.

A tenant represents an organization in Azure Active Directory. It's a reserved Azure AD service instance that an organization receives and owns when it signs up for a Microsoft cloud service such as Azure, Microsoft Intune, or Microsoft 365. Each Azure AD tenant is distinct and separate from other Azure AD tenants.

Once you have an Azure AD tenant, you can define an application and assign it permissions so it can call Power BI REST APIs.

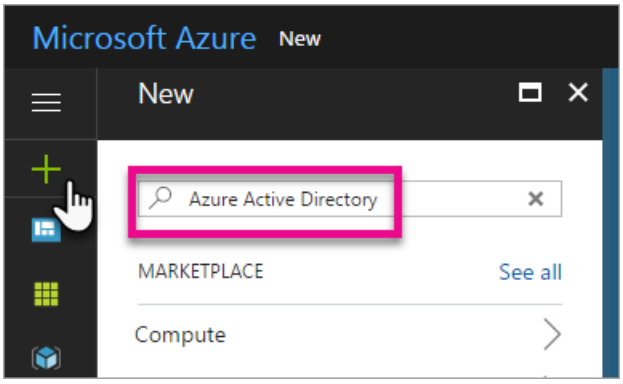
Your organization may already have an Azure AD tenant that you can use for your application. You can also create a new tenant specifically for your application. This article shows how to create a new tenant.

Create an Azure Active Directory tenant

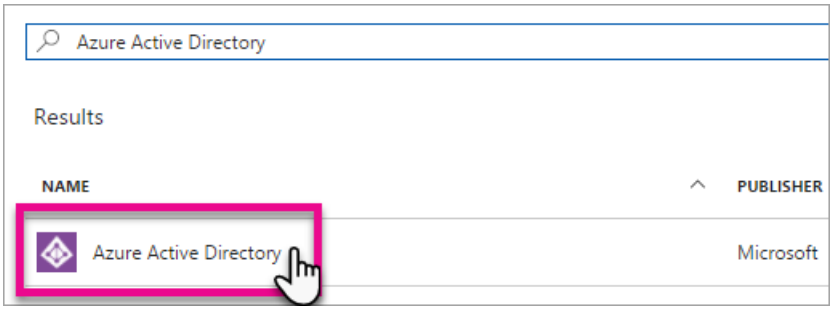
To integrate Power BI into your custom application, you need to define an application within Azure AD, which requires an Azure AD directory. This directory is your *tenant*. If your organization doesn't have a tenant yet, because they aren't using Power BI or Microsoft 365, then you need to set up a dev environment. You also need to create one if you don't want your application mixing with your organization's tenant, allowing you to keep things isolated. Or, you may just want to create a tenant for testing purposes.

To create a new Azure AD tenant:

1. Browse to the Azure portal and sign in with an account that has an Azure subscription.
2. Select the **plus icon (+)** and search for **Azure Active Directory**.

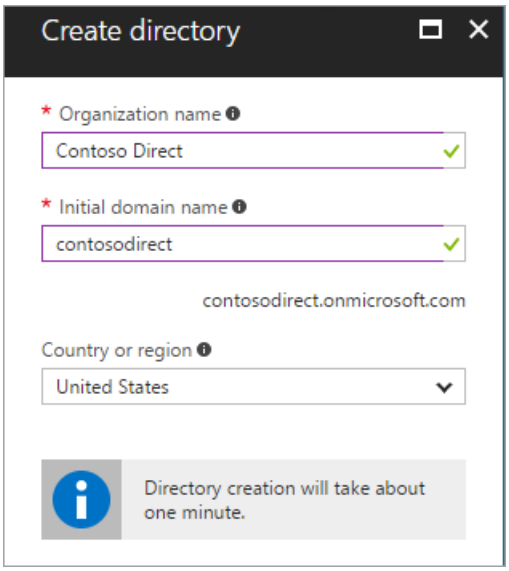


3. Select **Azure Active Directory** in the search results.



4. Select **Create**.

5. Provide an **Organization name** and an **Initial domain name**. Then select **Create**. Your directory is created.



> **NOTE**
> Your initial domain is part of onmicrosoft.com. You can add other domain names later. A tenant directory can have multiple domains assigned to it.

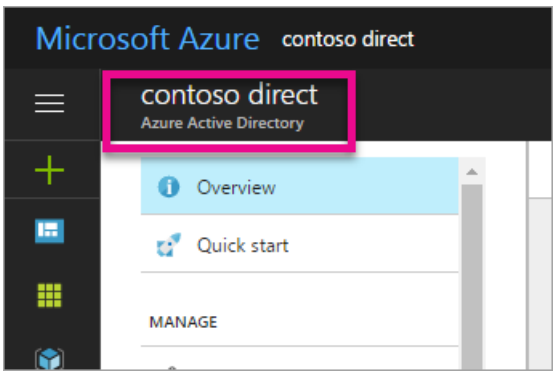
6. After directory creation is complete, select the information box to manage your new directory.

Next, you're going to add tenant users.

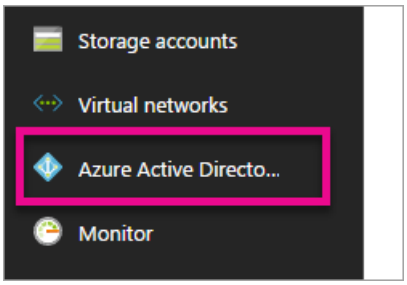
Create Azure Active Directory tenant users

Now that you have a directory, let's create at least two users. One is a tenant Global Admin and another is a master user for embedding. You can think of the latter as a service account.

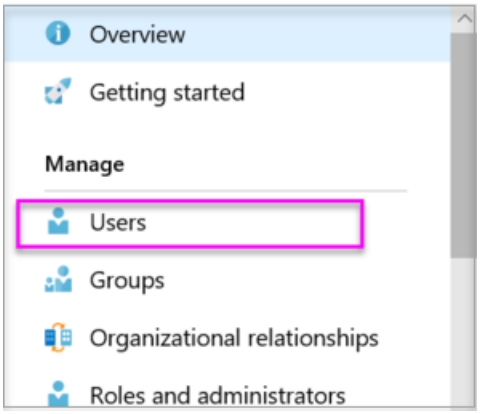
1. In the Azure portal, make sure you are on the Azure Active Directory fly out.



   If not, select the Azure Active Directory icon from the left services navigation.

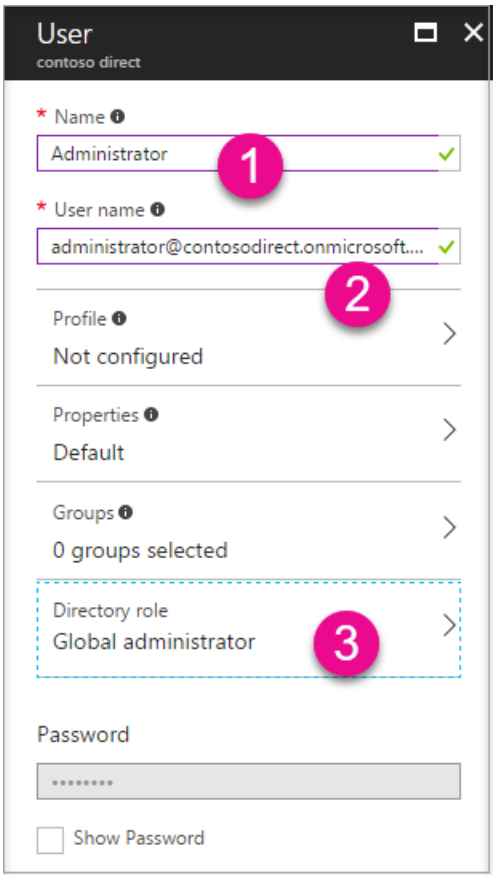


2. Under **Manage**, select **Users**.

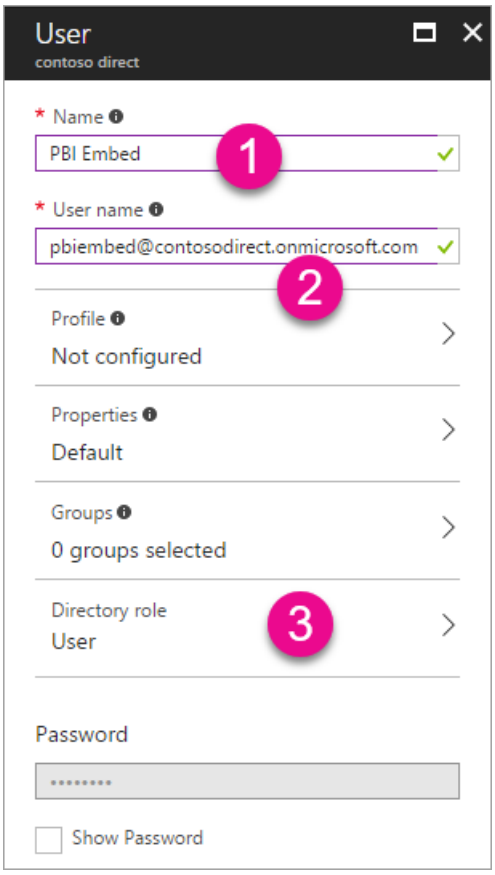


3. Select **All users** and then select **+ New user**.

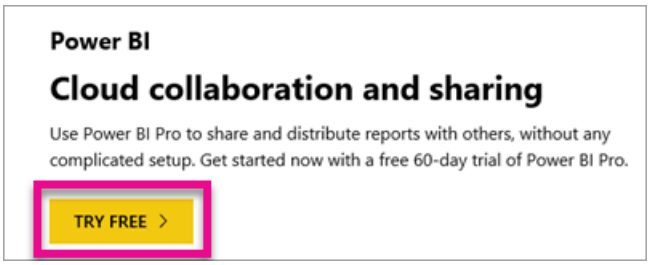
4. Provide a **Name** and **User name** for your tenant Global Admin. Change the **Directory role** to **Global administrator**. You can also show the temporary password. When you're done, select **Create**.



5. Do the same thing for a regular tenant user. You can use this account for your master embedding account. This time, for **Directory role**, leave it as **User**. Note the password, then select **Create**.



6. Sign up for Power BI with the user account that you created in step 5. Go to powerbi.com and select **Try free** under **Power BI - Cloud collaboration and sharing**.



When you sign up, you're prompted to try Power BI Pro free for 60 days. You can opt into that to become a Pro user, which gives you the option to start developing an embedded solution.

> **NOTE**
> Make sure you sign up with your user account's email address.

Next steps

Now that you have an Azure AD tenant, you can use this tenant to test items within Power BI. You can also embed Power BI dashboards and reports in your application. For more information, see How to embed your Power BI dashboards, reports, and tiles.

What is an Azure Active directory?

Quickstart: Set up a dev environment

More questions? Try asking the Power BI Community

Register an Azure AD application to use with Power BI

6/30/2022 • 10 minutes to read • Edit Online

To use Power BI embedded analytics, you need to register an Azure Active Directory (Azure AD) application in Azure. The Azure AD app establishes permissions for Power BI REST resources, and allows access to the Power BI REST APIs.

Determine your embedding solution

Before registering your app, decide which of the following solutions is best suited for you:

- Embed for your customers
- Embed for your organization

Embed for your customers

Use the embed for your customers solution, also known as *app owns data*, if you're planning to create an application that's designed for your customers. Users will not need to sign in to Power BI or have a Power BI license, to use your application. Your application will use one of the following methods to authenticate against Power BI:

- **Master user** account (a Power BI Pro license used for signing in to Power BI)
- Service principal

The embed for your customers solution is usually used by independent software vendors (ISVs) and developers who are creating applications for a third party.

Embed for your organization

Use the embed for your organization solution, also known as *user owns data*, if you're planning to create an application that requires users to use their credentials to authenticate against Power BI.

The embed for your organization solution is usually used by enterprises and big organizations, and is intended for internal users.

Register an Azure AD app

The easiest way to register an Azure AD app is by using the Power BI embedding setup tool. The tool offers a quick registration process for both embedding solutions, using a simple graphical interface.

If you're creating an *embed for your organization* application, and want more control over your Azure AD app, you can register it manually in the Azure portal.

> **IMPORTANT**
> Before you register a Power BI app you need an Azure Active Directory tenant and an organizational user.

- Embed for your customers
- Embed for your organization
- Manual registration

These steps describe how to register an Azure AD application for the Power BI embed for your customers solution.

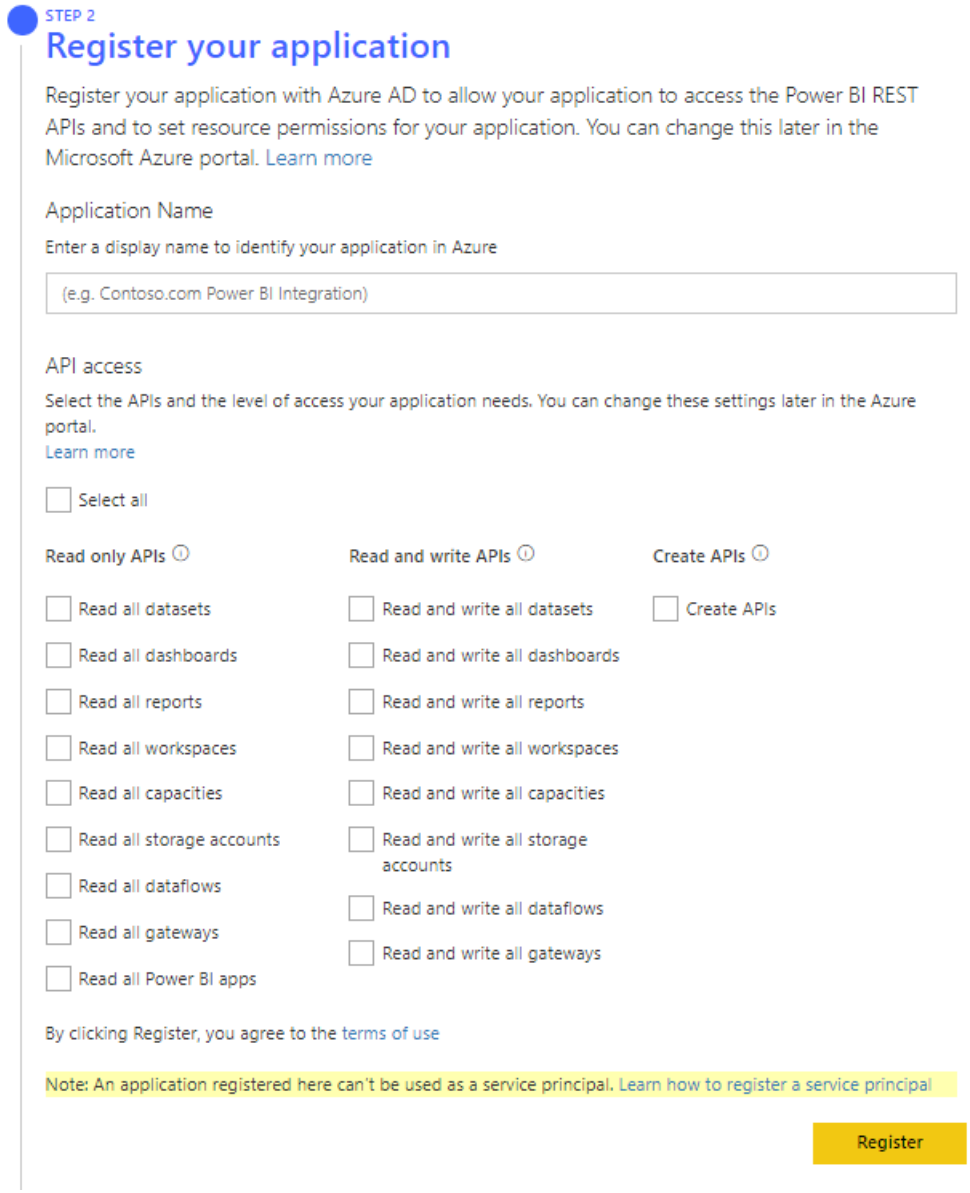
> **IMPORTANT**
>
> The following instructions will not work for GCC customers. If you are embedding for a GCC, follow the instructions for **Manual registration**.

1. Open the Power BI App Registration Tool.
2. In the *Choose an embedding solution* section, select **Embed for your customers**.
3. In *Step 1 - sign in to Power BI*, sign in with a user that belongs to your Power BI tenant. The Azure AD app will be registered under this user.

   > **NOTE**
   >
   > If you're already signed in, verify that you're signed in with the user you want to use for creating the Azure AD app. To change a user, select the *sign out* link and once the tool restarts, sign in again

   Click **Next**.
4. In *Step 2 - Register your application*, fill in the following fields:

   - **Application Name** - Give your application a name.
   - **API access** - Select the Power BI APIs (also known as scopes) that your application needs. You can use *Select all* to select all the APIs. For more information about Power BI access permissions, see Permissions and consent in the Microsoft identity platform endpoint.



5. Select **Register**.

   Your Azure AD app **Application ID** is displayed in the *Summary* box. Copy this value for later use.

6. (Optional) In *Step 3 - Create a workspace*, you can create a workspace in Power BI service.

   - If you already have a Power BI workspace, select **Skip**.
   - To create a workspace, enter a name for your workspace and select **Create workspace**. Your Workspace name and ID appear in the *Summary* box. Copy these values for later use.

   > **NOTE**
   > For the **embedded analytics sample app** to work as expected, you have to create a workspace using the tool.

7. (Optional) In *Step 4 - Import content*, select one of following options:

   - If you have your own Power BI app, you can select **Skip**.
   - If you want to create a sample Power BI app using a sample report, select **Sample Power BI report** and then select **Import**.
   - If you want to create a sample Power BI app using your own report, select **Upload a .pbix file**, browse for your file and then select **Import**.

8. In *Step 5 - Grant permissions*, select **Grant permissions** and in the pop-up window select **accept**. This allows your Azure AD app to access the APIs you selected (also known as scopes) with your signed in user. This user is also known as the **master user**.

9. (Optional) If you created a Power BI workspace and uploaded content to it using the tool, you can now select **Download sample application**. Make sure you copy all the information in the *Summary* Box.

> **NOTE**
>
> If you skipped the optional stages, you can still download a sample Power BI app. However, the code in the downloaded app, will lack the properties that you didn't fill in during registration. For example, if you didn't create a workspace, the sample app will not include the *workspace ID*.

Change your Azure AD app's permissions

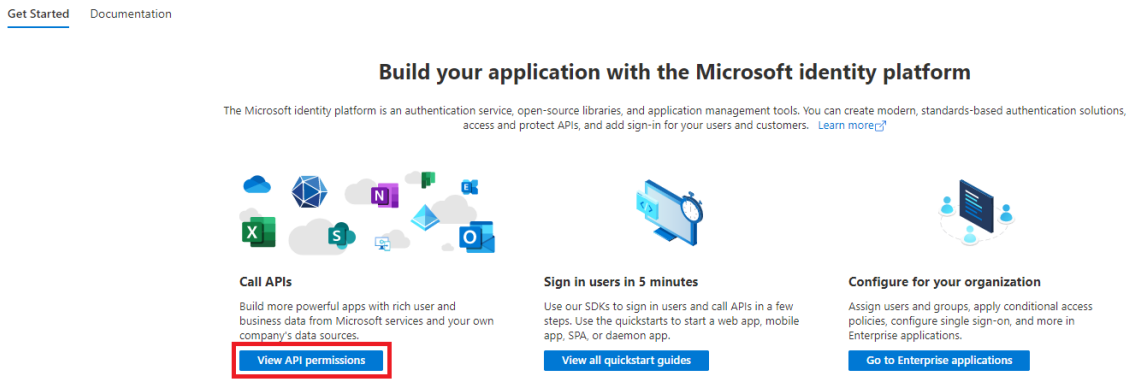
After you register your application, you can make changes to its permissions. Permission changes can be made programmatically, or in the Azure portal.

> **NOTE**
>
> Azure AD app permissions are only applicable for these scenarios:
>
> - *Embed for your organization*
> - *Embed for your customers* with the *master user* authentication method

- Azure
- HTTP
- C#

In the Azure portal, you can view your app and make changes to its permissions.

1. Sign into the Azure portal.
2. Select your Azure AD tenant by selecting your account in the upper right corner of the page.
3. Select **App registrations**. If you can't see this option, search for it.
4. From the **Owned applications** tab, select your app. The application opens in the *Overview* tab, where you can review the *Application ID*.
5. Select the **View API permissions** tab.

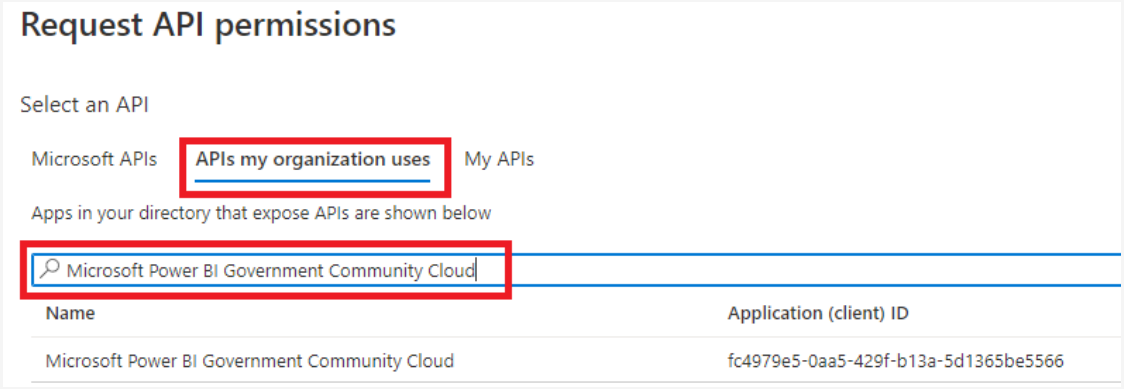


6. Select **Add a permission**.

7. To add permissions, follow these steps (note that the first step is different for GCC apps):

   a. From the **Microsoft APIs** tab, select **Power BI service**.

> **NOTE**
>
> For GCC apps, Select the **APIs my organization uses** tab, and search for either *Microsoft Power BI Government Community Cloud* OR *fc4979e5-0aa5-429f-b13a-5d1365be5566*.



   b. Select **Delegated Permissions** and add or remove the specific permissions you need.

   c. When you're done, select **Add permissions** to save your changes.

8. To remove a permission, follow these steps:

   a. Select the ellipsis (...) to the right of the permission.

   b. Select **Remove permission**.

   c. In the *Remove permission* pop-up window, select **Yes, remove**.

Next steps

Considerations when generating an embed token

Capacity and SKUs in Power BI embedded analytics

Monitor Power BI Embedded

6/30/2022 • 5 minutes to read • Edit Online

When you have critical applications and business processes relying on Azure resources, you want to monitor those resources for their availability, performance, and operation. This article describes the monitoring data generated by Power BI Embedded and how you can use the features of Azure Monitor to analyze and alert on this data.

> **TIP**
> You can also use the Premium Gen2 Monitoring App to monitor your Embedded Gen 2 capacity.

Monitor overview

The **Overview** page in the Azure portal for each *Power BI Embedded* instance, includes the following information:

- **Resource group** - The resource group the instance belongs to
- **Status** - The status of your Power BI Embedded instance
- **Location** - The location of your Power BI Embedded instance
- **Resource name** - The name of your Power BI Embedded instance
- **SKU** - The SKU your Power BI Embedded instance is using
- **Subscription name** - Your Power BI Embedded instance subscription name
- **Subscription ID** - Your Power BI Embedded instance subscription ID

What is Azure Monitor?

Power BI Embedded creates monitoring data using Azure Monitor, which is a full stack monitoring service in Azure that provides a complete set of features to monitor your Azure resources in addition to resources in other clouds and on-premises.

Start with the article Monitoring Azure resources with Azure Monitor, which describes the following concepts:

- What is Azure Monitor?
- Costs associated with monitoring
- Monitoring data collected in Azure
- Configuring data collection
- Standard tools in Azure for analyzing and alerting on monitoring data

The following sections build on this article by describing the specific data gathered for Power BI Embedded and providing examples for configuring data collection and analyzing this data with Azure tools.

Monitoring data

Power BI Embedded collects the same kinds of monitoring data as other Azure resources that are described in Monitoring data from Azure resources.

See Monitoring *Power BI Embedded* data reference for detailed information on the metrics and logs metrics created by Power BI Embedded.

Collection and routing

Platform metrics and the Activity log are collected and stored automatically, but can be routed to other locations by using a diagnostic setting.

Resource Logs are not collected and stored until you create a diagnostic setting and route them to one or more locations.

See Create diagnostic setting to collect platform logs and metrics in Azure for the detailed process for creating a diagnostic setting using the Azure portal, CLI, or PowerShell. When you create a diagnostic setting, you specify which categories of logs to collect. The categories for *Power BI Embedded* are listed in Power BI Embedded monitoring data reference.

Using PowerShell to enable diagnostics

To enable metrics and diagnostics logging by using PowerShell, use the commands listed below. To learn more about using PowerShell to enable diagnostics, see Create and configure a Log Analytics workspace in Azure Monitor using PowerShell.

- To enable storage of diagnostics logs in a storage account, use this command:

  ```
  Set-AzDiagnosticSetting -ResourceId [your resource id] -StorageAccountId [your storage account id] -
  Enabled $true
  ```

  The storage account ID is the resource ID for the storage account where you want to send the logs.

- To enable streaming of diagnostics logs to an event hub, use this command:

  ```
  Set-AzDiagnosticSetting -ResourceId [your resource id] -ServiceBusRuleId [your service bus rule id] -
  Enabled $true
  ```

- The Azure Service Bus rule ID is a string with this format:

  ```
  {service bus resource ID}/authorizationrules/{key name}
  ```

- To enable sending diagnostics logs to a Log Analytics workspace, use this command:

  ```
      Set-AzDiagnosticSetting -ResourceId [your resource id] -WorkspaceId [resource id of the log
  analytics workspace] -Enabled $true
  ```

- You can obtain the resource ID of your Log Analytics workspace by using the following command:

  ```
  (Get-AzOperationalInsightsWorkspace).ResourceId
  ```

You can combine these parameters to enable multiple output options.

The metrics and logs you can collect are discussed in the following sections.

Analyzing metrics

You can analyze metrics for *Power BI Embedded* with metrics from other Azure services using metrics explorer by opening **Metrics** from the **Azure Monitor** menu. See Getting started with Azure Metrics Explorer for details on using this tool.

For a list of the platform metrics collected for Power BI Embedded, see Monitoring *Power BI Embedded* data

reference metrics

For reference, you can see a list of all resource metrics supported in Azure Monitor.

Analyzing logs

Data in Azure Monitor Logs is stored in tables where each table has its own set of unique properties.

All resource logs in Azure Monitor have the same fields followed by service-specific fields. The common schema is outlined in Azure Monitor resource log schema The schema for Power BI Embedded resource logs is found in the Power BI Embedded Data Reference.

The Activity log is a platform login Azure that provides insight into subscription-level events. You can view it independently or route it to Azure Monitor Logs, where you can do much more complex queries using Log Analytics.

For a list of the types of resource logs collected for Power BI Embedded, see Monitoring Power BI Embedded data reference

For a list of the tables used by Azure Monitor Logs and queryable by Log Analytics, see Monitoring Power BI Embedded data reference

Sample Kusto queries

> **IMPORTANT**
>
> When you select **Logs** from the Power BI Embedded menu, Log Analytics is opened with the query scope set to the current Power BI Embedded resource. This means that log queries will only include data from that resource. If you want to run a query that includes data from other Power BI Embedded resource or data from other Azure services, select **Logs** from the **Azure Monitor** menu. See Log query scope and time range in Azure Monitor Log Analytics for details.

Here is an example of a query that is completed in less than five minutes (300,000 milliseconds).

```
search *
| where Type == "AzureDiagnostics"
| where ( OperationName == "QueryEnd" )
| where toint(Duration_s) < 300000
```

Alerts

Azure Monitor alerts proactively notify you when important conditions are found in your monitoring data. They allow you to identify and address issues in your system before your customers notice them. You can set alerts on metrics, logs, and the activity log.

Next steps

You can learn more about Azure resource diagnostic logging.

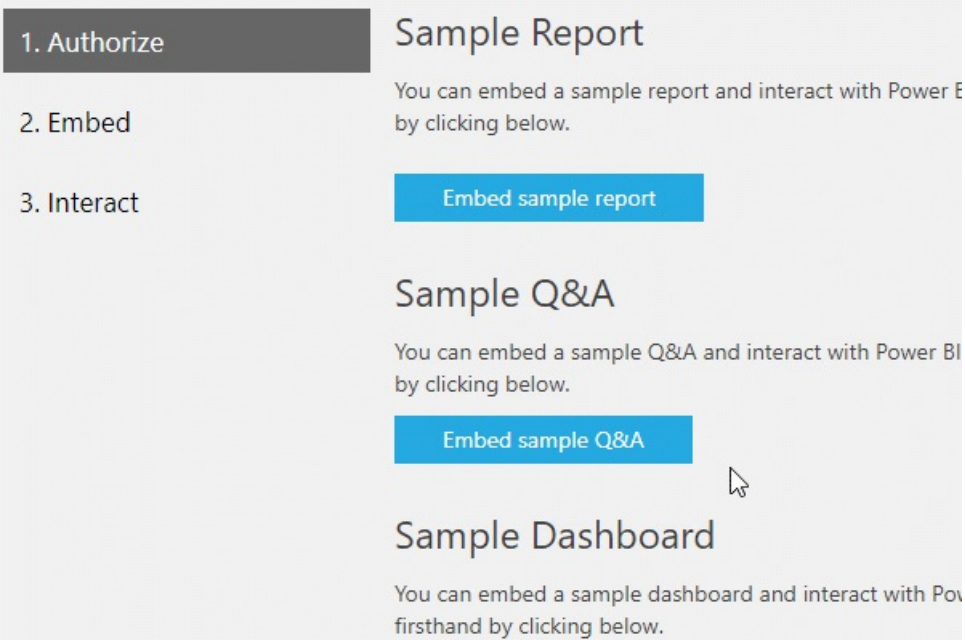
Monitoring Power BI Embedded data reference

Monitoring Azure resources with Azure Monitor

Q&A in Power BI embedded analytics

6/30/2022 • 2 minutes to read • Edit Online

Power BI embedded analytics offers you a way to incorporate Q&A into an application. Your users can ask questions using natural language, and receive immediate answers in the form of visuals like charts or graphs.



There are two modes for embedding Q&A within your application: **interactive** and **result only**. **Interactive** mode allows you to type in questions and have them displayed within the visual. If you have a saved question, or a set question you want to display, you can use the **result only** mode by populating the question in your embed config.

Here's what the JavaScript code will look like.

```
// Embed configuration used to describe the what and how to embed.
// This object is used when calling powerbi.embed within the JavaScript API.
// You can find more information at https://github.com/Microsoft/PowerBI-JavaScript/wiki/Embed-
Configuration-Details.
var config= {
    type: 'qna',
    tokenType:   models.TokenType.Embed | models.TokenType.Aad,
    accessToken: access token value,
    embedUrl:    https://app.powerbi.com/qnaEmbed (groupId to be appended as query parameter if required),
    datasetIds:  array of requested data set ids (at the moment we support only one dataset),
    viewMode:    models.QnaMode.Interactive | models.QnaMode.ResultOnly,
    question:    optional parameter for Explore mode (QnaMode.Interactive) and mandatory for Render Result
mode (QnaMode.ResultOnly)
};

// Get a reference to the embedded QNA HTML element
var qnaContainer = $('#qnaContainer')[0];

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

Set question

If you use **result mode** with a set question, you can inject more questions into the frame. The answer to the new question will immediately replace the previous result. A new visual is rendered matching the new question.

One example of this usage would be a frequently asked question list. The user could go through the questions and have them answered within the same embedded part.

Code snippet for JS SDK usage:

```
// Get a reference to the embedded Q&A HTML element
var qnaContainer = $('#qnaContainer')[0];

// Get a reference to the embedded Q&A.
qna = powerbi.get(qnaContainer);

qna.setQuestion("This year sales")
    .then(function (result) {
        …….
    })
    .catch(function (errors) {
        …….
    });
```

Visual rendered event

For **interactive** mode, the application can be notified with a data changed event each time the rendered visual changes to target the updated input query as it is being typed.

Listening to the *visualRendered* event allows you to save questions for use later.

Code snippet for JS SDK usage:

```
// Get a reference to the embedded Q&A HTML element
var qnaContainer = $('#qnaContainer')[0];

// Get a reference to the embedded Q&A.
qna = powerbi.get(qnaContainer);

// qna.off removes a given event listener if it exists.
qna.off("visualRendered");

// qna.on will add an event listener.
qna.on("visualRendered", function(event) {
     …….
});
```

Embed token

Create an embed token off of a dataset to start a Q&A part. For more information, see Generate token.

Next steps

Try out Q&A embedding with the JavaScript embed sample

More questions? Try asking the Power BI Community

Power BI Embedded Generation 2

6/30/2022 • 2 minutes to read • Edit Online

Power BI Embedded recently released a new version of Power BI Embedded, **Power BI Embedded Generation 2**, referred to as **Embedded Gen2** for convenience. You can still create the original version of the Power BI Embedded resource, called *Embedded Gen 1*, or create a new *Embedded Gen 2* resource. You can run both types of Power BI Embedded capacities in parallel, and assign any workspace to either a Gen1 or a Gen2 capacity".

All of the Power BI Embedded Gen 1 capabilities such as pausing and resuming the capacity, are preserved in Gen 2. In addition, the Gen 2 capacity resource provides the following updates and improved experience:

- **Enhanced performance** - Better performance on any capacity size, anytime. Operations will always perform at top speed and won't slow down when the load on the capacity approaches the capacity limits.
- **Greater scale** - Including the following enhancements:
  - *No limits on refresh concurrency* - You no longer need to track schedules for datasets being refreshed on your capacity
  - *Fewer memory restrictions*
  - *Complete separation between report interaction and scheduled refreshes*
- **Lower entry level for paginated reports and AI workloads** – Start with an *A1* SKU and grow as you need.
- **Scaling a resource instantly** - Instantly scale your Power BI Embedded resource. From scaling a Gen1 resource in minutes, to scaling a Gen2 resource in seconds.
- **Scaling without downtime** - With Embedded Gen2 you can scale your Power BI Embedded resource without experiencing any downtime.
- **Improved metrics** - Including clear and normalized capacity utilization data, depending only on the complexity of the analysis operations the capacity performs. Metrics are not impacted by other factors such as the size of the capacity, and the level of load on the system while performing analytics. When using the improved metrics, the built-in reporting tool allows you to clearly see:
  - Utilization analysis
  - Budget planning
  - Chargebacks
  - The need to upgrade your capacity

> **NOTE**
> You can get usage metrics for Embedded Gen2 capacities by using the Premium Gen 2 Capacity Utilization Metrics app.

Using Embedded Gen2

Create an Embedded Gen2 capacity resource to take advantage of its updates. To create an Embedded Gen2 capacity resource, follow the instructions in Create Power BI Embedded capacity in the Azure portal.

Understanding Embedded Gen2

Embedded Gen 2 has the same architecture improvements, capacity load evaluation and background operation scheduling as Premium Gen2.

Autoscaling in Embedded Gen2

Embedded Gen2 does not provide an out-of-the-box vertical autoscale feature. Instead, customers can configure autoscale using these options:

- Power BI Embedded Azure Resource Manager REST APIs, for example Capacities - Update.
- Power BI Embedded Gen2 capacity metrics such as *CPU*, *CPU Per Workload*, and *Overload*.
- Azure alerts. You can use the Power BI Embedded sample script as a reference for scaling a capacity.

Considerations and limitations

- Memory allocation settings for specific workloads don't apply to Embedded Gen2 capacities. For more information, see Embedded Gen 2 memory enhancements
- If you're using XMLA with Embedded Gen2, make sure you're using the most recent versions of the data modeling and management tools.
- Analysis services features in Embedded Gen2 are only supported on the latest client libraries. Estimated release dates for dependent tools to support this requirement are listed in limitations in Premium Gen2.
- For a list of Embedded Gen2 memory restrictions, see limitations in Premium Gen2.

Next steps

Create Power BI Embedded capacity in the Azure portal

Capacity and SKUs in Power BI embedded analytics

What is Power BI Premium?

Monitoring Power BI Embedded data reference

Plan your transition to Power BI Embedded Gen2

6/30/2022 • 2 minutes to read • Edit Online

This article provides information about key dates for migrating Power BI Embedded capacity to the latest platform.

Over the last several months, we've been working to make many improvements to Power BI Embedded. Changes include updates to performance, scaling, management overhead, and improved insight to utilization metrics. This next generation of Power BI Embedded, referred to as Power BI Embedded Gen2, has officially moved from preview to general availability as of October 4, 2021.

If your organization is using the previous version of Power BI Embedded, you're required to migrate capacities to the modern Gen2 platform. Microsoft began migrating all Embedded capacities to Gen2 capacities. If you have a Embedded capacity that requires migrating, **you'll receive an email notification 60 days before the migration is scheduled to start**.

Self-migration to Embedded Generation 2

If you want to perform your own migration to the latest platform before January 15, 2022, it's easy to transition. You can upgrade through either the Azure portal or the API. Migrating doesn't interrupt your Power BI service. The change typically completes within a minute and won't take more than 10 minutes.

Ready for the next generation? Follow these instructions in Upgrade a capacity to Gen2.

Transition from preview to Embedded Gen 2 general availability

Customers using Power BI Embedded Gen2 in preview don't need to take any action to transition to the general availability release.

Migration notification

Following the general availability of gen2, we'll begin to notify affected customers so that you can prepare your organization for changes. We'll post additional awareness, along with specific migration timelines to Microsoft 365 Message Center. Admins will receive 60 days advance notice of changes. The timeline will vary by cloud.

Next steps

What is Power BI Embedded Gen2?

Capacity and SKUs in Power BI embedded analytics

6/30/2022 • 5 minutes to read • Edit Online

When moving to production, Power BI embedded analytics requires a capacity (*A*, *EM*, or *P* SKU) for publishing embedded Power BI content.

Capacity is a dedicated set of resources reserved for exclusive use. It offers dependable, consistent performance for your content.

> **NOTE**
> For publishing, you'll need one Power BI Pro or Premium Per User (PPU) account.

What is embedded analytics?

Power BI embedded analytics includes two solutions:

- *Power BI Embedded* - Azure offering
- Embedding Power BI as part of *Power BI Premium* - Microsoft Office offering

Power BI Embedded

Power BI Embedded is for ISVs and developers who want to embed visuals into their applications.

Applications using Power BI Embedded allow users to consume content stored on Power BI Embedded capacity.

> **NOTE**
> Power BI Embedded recently released a new version, called **Embedded Gen2**. Embedded Gen2 simplifies the management of embedded capacities, and improves the Power BI Embedded experience. For more information, see Power BI Embedded Generation 2.

Power BI Premium

Power BI Premium is geared toward enterprises who want a complete BI solution that provides a single view of its organization, partners, customers, and suppliers.

Power BI Premium is a SaaS product that allows users to consume content through mobile apps, internally developed apps, or at the Power BI portal (Power BI service). This enables Power BI Premium to provide a solution for both internal and external customer facing applications.

Capacity and SKUs

Each capacity offers a selection of SKUs, and each SKU provides different resource tiers for memory and computing power. The type of SKU you require, depends on the type of solution you wish to deploy.

To understand which workloads are supported for each tier, refer to the Configure workloads in a Premium capacity article.

To plan and test your capacity, use these links:

- Capacity planning
- Testing approaches

Power BI Embedded SKUs

Power BI Embedded is shipped with an *A* SKU.

Power BI Premium SKUs

Power BI premium offers two SKUs, *P* and *EM*.

- Understand the differences between the *P* and *EM* SKUs
- Buy a Premium SKU

Which SKU should I use?

The table below provides a summary of features, the capacity they require, and the specific SKU that is needed for each one.

In this table, a custom app refers to a web app created using embedded analytics. When you embed to a custom web app as a developer (using the JavaScript or .NET SDKs, or the REST APIs), you can control and customize the UX. This ability isn't available with other embedding options, such as Power BI service and Power BI Mobile.

SCENARIO	AZURE	OFFICE
	(A SKU)	(P and EM SKUs)
Embed for your customers (app owns data)	✔	✔
Embed for your organization (user owns data)	✘	✔
Microsoft 365 apps (formerly known as Office 365 apps) • Embed in Teams • Embed in SharePoint	✘	✔
Secure URL embedding (embed from Power BI service)	✘	✔

NOTE

- A Power BI Pro or Premium Per User (PPU) license is needed for publishing content to a Power BI app workspace.
- Only the **P SKU** allows free Power BI users to consume Power BI apps and shared content, in Power BI service.

Capacity considerations

The table below lists payment and usage considerations per capacity.

PAYMENT AND USAGE	POWER BI EMBEDDED	POWER BI PREMIUM	POWER BI PREMIUM
Offer	Azure	Office	Office
SKU	A	EM	P
Billing	Hourly	Monthly	Monthly
Commitment	None	Yearly	Monthly or yearly

PAYMENT AND USAGE	POWER BI EMBEDDED	POWER BI PREMIUM	POWER BI PREMIUM
Usage	Azure resources can be: • Scaled up or down • Paused and resumed	Embed in apps, and in Microsoft applications	Embed in apps, and in Power BI service

SKU memory and computing power

The table below describes the resources and limits of each SKU.

- Premium Gen2
- Premium Gen1

CAPACITY SKUS	TOTAL V-CORES	BACKEND V-CORES	FRONTEND V-CORES	MAX MEMORY PER DATASET (GB)[1, 2, 3]	DIRECTQUERY/LIVE CONNECTION (PER SECOND)[1, 2]	MAX MEMORY PER QUERY (GB)[1, 2]	MODEL REFRESH PARALLELISM[2]
EM1/A1	1	0.5	0.5	3	3.75	1	5
EM2/A2	2	1	1	5	7.5	2	10
EM3/A3	4	2	2	10	15	2	20
P1/A4	8	4	4	25	30	6	40
P2/A5	16	8	8	50	60	6	80
P3/A6	32	16	16	100	120	10	160
P4/A7[4]	64	32	32	200	240	10	320
P5/A8[4]	128	64	64	400	480	10	640

[1] The Power BI Premium Utilization and Metrics app doesn't currently expose these metrics.

[2] These limits only apply to the datasets workload per capacity.

[3] The *Max memory per dataset (GB)* column represents an upper bound for the dataset size. However, some memory must be reserved for operations such as dataset refreshes and queries. The maximum dataset size permitted on a capacity may be smaller than the numbers in this column.

[4] SKUs greater than 100 GB aren't available in all regions. To request using these SKUs in regions where they're not available, contact your Microsoft account manager.

Embedded Gen 2 memory enhancements

The amount of memory available on each node size is described in the *RAM (GB)* column in the SKU memory and computing power table. With Power BI Embedded Generation 2 (also known as Embedded Gen 2), it's set to the memory footprint limit of a single Power BI item (such as a dataset, report or dashboard), and not to the cumulative consumption of memory. For example, in an Embedded Gen2 A4 capacity, a single dataset size is limited to 25 GB, compared to the original Power BI Embedded capacity, where the total memory footprint of *all* datasets handled at the same time was limited to 25 GB.

Next steps

Embed for your customers

Embed for your organization

Embed from apps

Capacity planning in Power BI embedded analytics

6/30/2022 • 2 minutes to read • Edit Online

Calculating the type of capacity needed for a Power BI embedded analytics deployment can be complicated since the calculation is based on multiple parameters, some of them hard to predict.

Some things you should take into consideration when planning your capacity are:

- The data models you're using
- The number and complexity of required queries
- The hourly distribution of the usage of your application
- Data refresh rates
- Additional usage patterns that are hard to predict.

This article is designed to make capacity planning for Power BI embedded analytics easier by introducing the Power BI Capacity Load Assessment Tool, created for automating load testing for Power BI embedded analytics capacities (*A*, *EM* or *P* SKUs).

Planning tool

The Power BI Capacity Load Assessment Tool can help you understand how much user load your capacity can handle. It uses PowerShell to create automated load tests against your capacities, and lets you choose which reports to test, and how many concurrent users to simulate.

The tool generates load on a capacity by continuously rendering each report with new filter values (to prevent unrealistically good performance due to report caching), until the token required for authenticating the tool against the service, expires.

Using the planning tool

When running the tool, be mindful of the existing load on your capacities and make sure not to run load tests during top usage times.

Here are some examples of how you can use the planning tool.

- Capacity administrators can get a better understanding of how many users their capacity can handle in a given time frame.
- Report authors can understand the user load effect, as measured with Power BI desktop's Performance Analyzer.
- You can see renders happening in real time on your browser.
- Using SQL Server Profiler, you can connect to the XMLA endpoints of the capacities being measured, to see the queries being executed.
- The load test effects are visible in the premium capacity metrics app's Datasets page. Capacity admins can use this tool to generate load, and see how that load shows up.

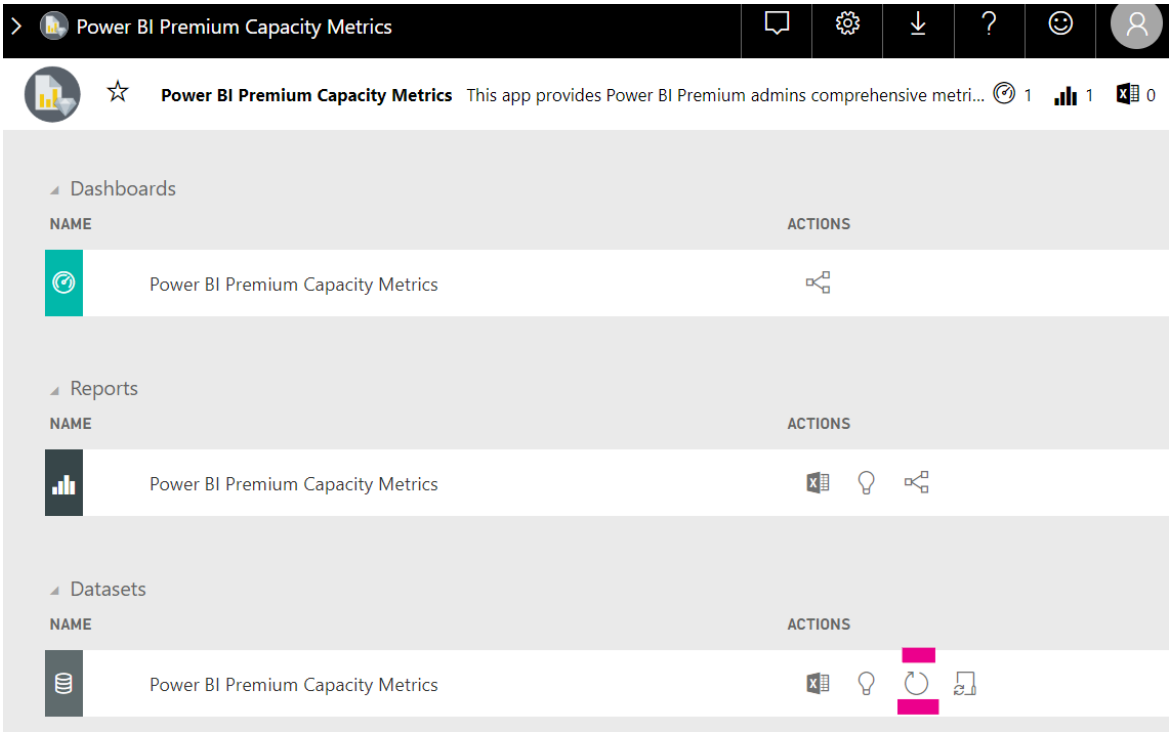
Reviewing the test results

The following section explains how to review the load test results of a Gen 1 capacity:

To see the effects of the load assessment test in the metrics app after the test runs, follow the instructions below. Expect up to a 15 minute lag from the time the test starts generating load, until the load is visible in the metrics.

1. Expand the **Datasets** tab of your metrics app landing page.

2. Initiate an on-demand refresh by clicking **refresh now**.



Power BI capacity tools GitHub repository

The Power BI capacity tools GitHub repository was created to host the capacity planning tool and other future tools and utilities.

The repository is open source and users are encouraged to contribute, add additional tools related to Power BI Premium and Embedded capacities, and improve the existing ones.

Next steps

- Capacity and SKUs in Power BI embedded analytics
- Power BI Embedded performance best practices

Create Power BI Embedded capacity in the Azure portal

6/30/2022 • 8 minutes to read • Edit Online

This article walks you through how to create a Power BI Embedded capacity in Microsoft Azure. Power BI Embedded simplifies Power BI capabilities by helping you quickly add stunning visuals, reports, and dashboards to your apps.

> **IMPORTANT**
> Within the next few months, all Gen1 capacities will be deprecated and only Power BI Embedded Gen2 capacities will available. We recommend that you upgrade your embedded capacities to Gen2 using the Azure portal or the ARM API.

Before you begin

To complete this quickstart, you need:

- **Azure subscription:** Visit Azure Free Trial to create an account.
- **Azure Active Directory:** Your subscription must be associated with an Azure Active Directory (Azure AD) tenant. Also, ***you need to be signed in to Azure with an account in that tenant***. Microsoft accounts aren't supported. To learn more, see Authentication and user permissions.
- **Power BI tenant:** At least one account in your Azure AD tenant must be signed up for Power BI.
- **Resource group:** Use a resource group you already have or create a new one.

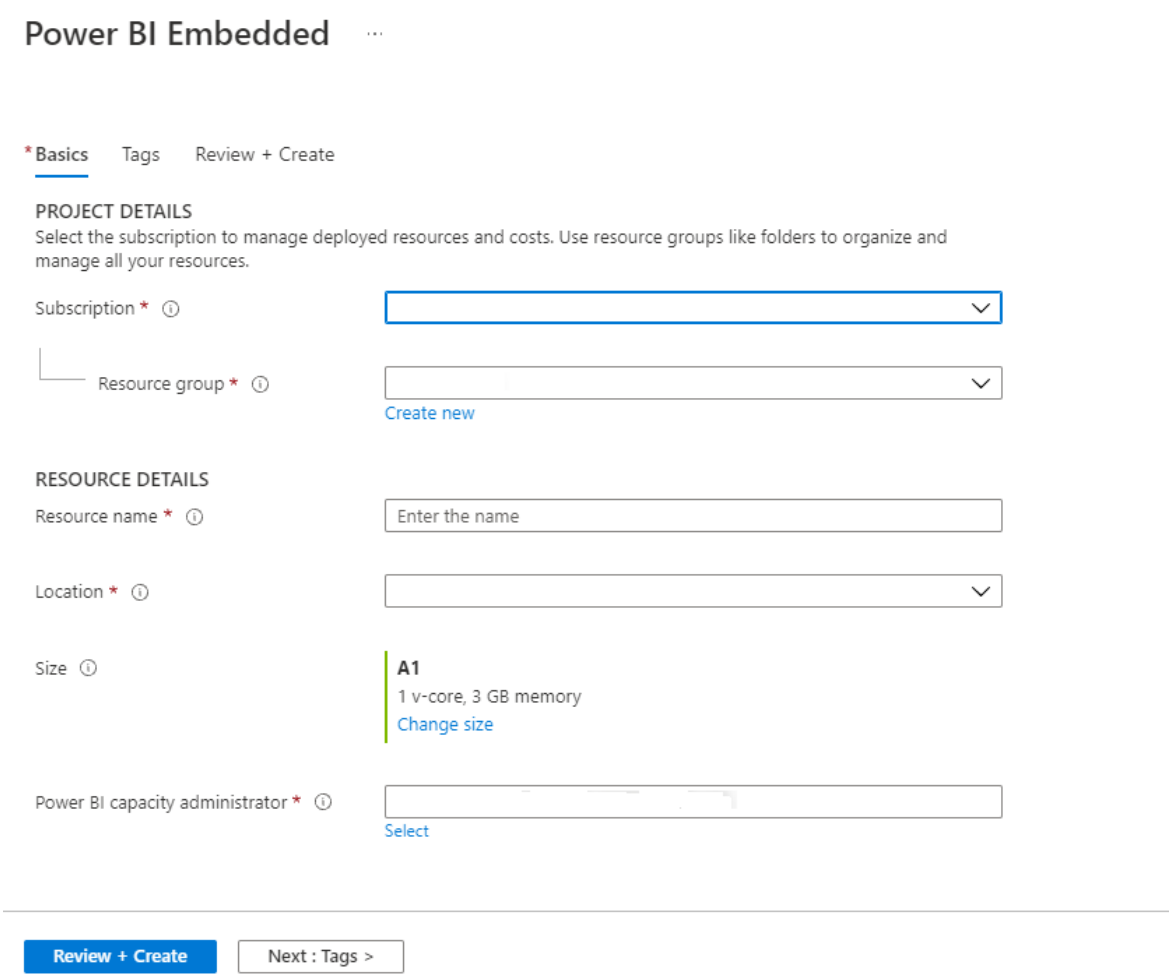
Create a capacity

> **NOTE**
> To create or manage a capacity, you must have the built-in role of contributor or higher.

Before creating a Power BI Embedded capacity, make sure you have signed into Power BI at least once.

- Portal
- Azure CLI
- ARM template

1. Sign into the Azure portal.
2. Under **Azure services**, select *Power BI Embedded*.
3. Within Power BI Embedded, select **Create**.
4. Fill in the required information and then select **Review + Create**.



- **Subscription** - The subscription you would like to create the capacity against.
- **Resource group** - The resource group that contains this new capacity. Pick from an existing resource group, or create another. For more information, see Azure Resource Manager overview.
- **Resource name** - The resource name of the capacity.
- **Location** - The location where Power BI is hosted for your tenant. Your default location is your home region, but you can change the location using Multi-Geo options.
- **Size** - The A SKU you require. For more information, see SKU memory and computing power.
- **Power BI capacity administrator** - An admin for the capacity.

> **NOTE**
>
> - By default, the capacity administrator is the user creating the capacity.
> - You can select a different user or service principal, as capacity administrator.
> - The capacity administrator must belong to the tenant where the capacity is provisioned. Business to business (B2B) users cannot be capacity administrators.

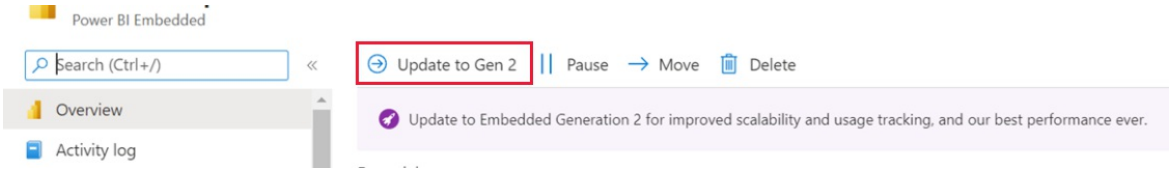
Upgrade a capacity to Gen2

You can upgrade a Gen1 capacity to Gen2 in either the Azure UI portal, or the ARM API.

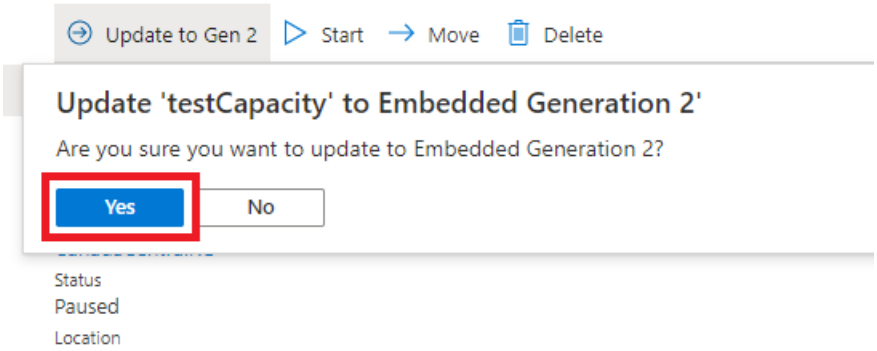
- Azure UI
- ARM API

To upgrade a Gen1 capacity to Gen2 in the Azure UI:

1. Select the capacity.
2. From the overview section select **Update to Gen 2**.



3. Select **Yes** to confirm.



The capacity will update automatically in a few seconds.

Next steps

Manage capacities

Pause and start your Power BI Embedded capacity in the Azure portal

Embed Power BI content into an application for your customers

More questions? Try asking the Power BI Community

Multi-Geo support for Power BI Embedded

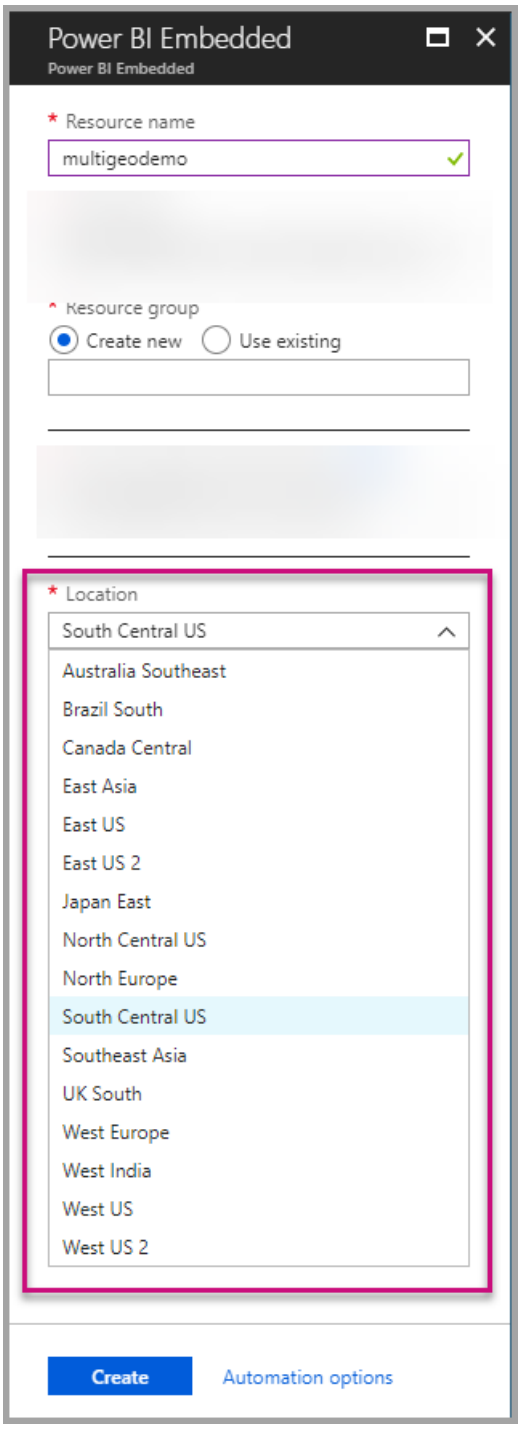
6/30/2022 • 2 minutes to read • Edit Online

Multi-Geo support for Power BI Embedded means that ISVs and organizations that build applications using Power BI Embedded to embed analytics into their apps can now deploy their data in different regions around the world.

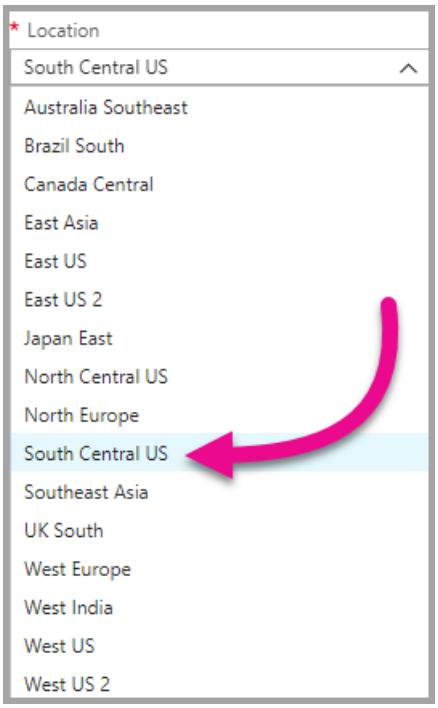
Now, customers using **Power BI Embedded** can set up an **A capacity** using **Multi-Geo** options, based on the same features and limitations that Power BI Premium supports using Multi-Geo.

Creating new Power BI Embedded Capacity resource with Multi-Geo

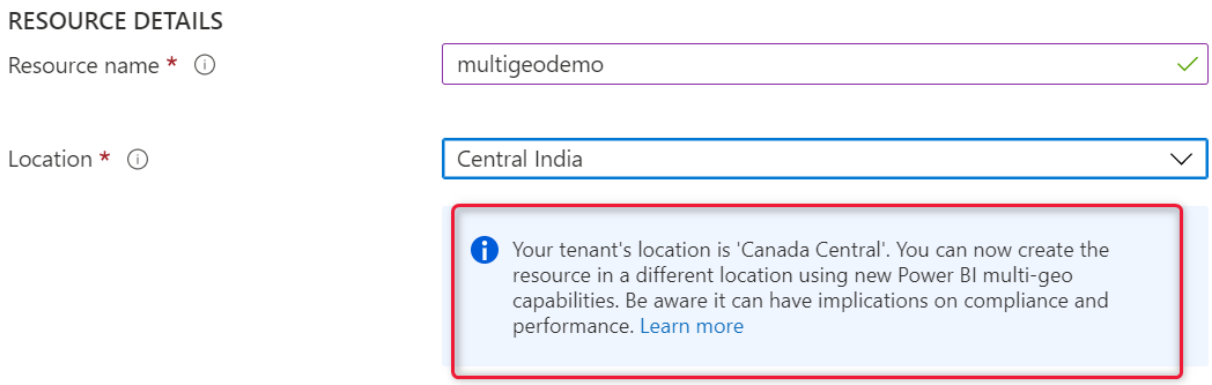
In the **Create resource** screen, choose the location of your capacity. Previously, the location was limited to the location of your Power BI tenant, so only a single location was available. With Multi-Geo, you can choose between different regions to deploy your capacity.



Notice that when you open the *location* drop-down menu, your home tenant is the default selection.

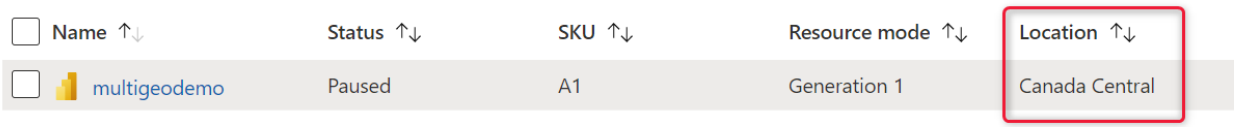


When you choose a location other than the default tenant location, a message prompts you to make sure you're aware of the selection.

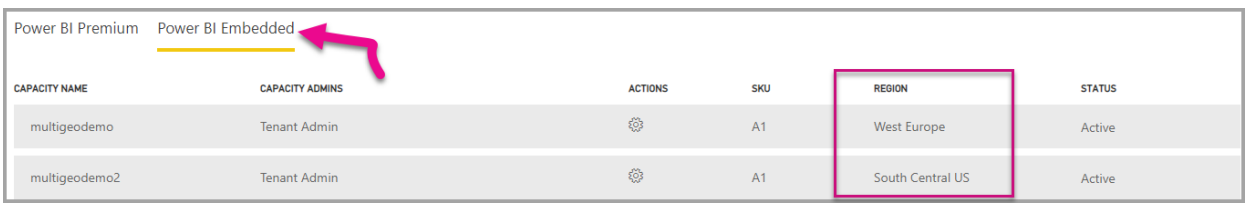


View Capacity location

You can see your capacities' locations easily from the main Power BI Embedded management page in the Azure portal.



The locations are also available in the Admin portal at `PowerBI.com`. In the Admin portal, choose **Capacity settings**, and then switch to the **Power BI Embedded** tab.



Learn more about creating capacities with Power BI Embedded.

Manage existing capacities location

You can't change a Power BI Embedded resource location once you created a new capacity.

To move your Power BI content to a different region, follow these steps:

1. Create a new capacity in a different region.
2. Assign all workspaces from the existing capacity to the new capacity.
3. Delete or pause the old capacity.

It's important to note that if you decide to delete a capacity without reassigning its content, all the content in that capacity moves to a shared capacity in your home region.

API support for Multi-Geo

To manage capacities with Multi-Geo through APIs, use the following APIs:

1. **Get Capacities** - The API returns a list of capacities with access to the user. The response includes a property called `region`, that specifies the capacity's location.
2. **Assign To Capacity** - The API allows you to assign a workspace to a capacity outside of your home region, or to move workspaces between capacities in different regions. To perform this operation, the user or service principal needs admin permissions on the workspace, and admin or assign permissions on the target capacity.
3. **Azure Resource Manager API** - all of the Azure Resource Manager API operations, including *Create* and *Delete*, supports Multi-Geo.

Considerations and limitations

- The Power BI Embedded multi-geo limitations are similar to the Power BI Premium multi-geo considerations and limitations.

Next steps

What is Power BI Embedded?

Create a Power BI Embedded capacity

Multi-Geo in Power BI Premium capacities

More questions? Try asking the Power BI Community

Scale your Power BI Embedded capacity in the Azure portal

6/30/2022 • 2 minutes to read • Edit Online

This article walks through how to scale a Power BI Embedded capacity in Microsoft Azure. Scaling allows you to increase or decrease the size of your capacity.

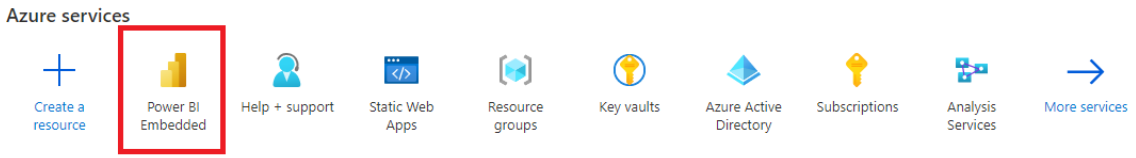
This assumes you created a Power BI Embedded capacity. If you haven't, see Create Power BI Embedded capacity in the Azure portal to get started.

For information about autoscaling in Embedded Gen 2, see Autoscaling in Embedded Gen 2.

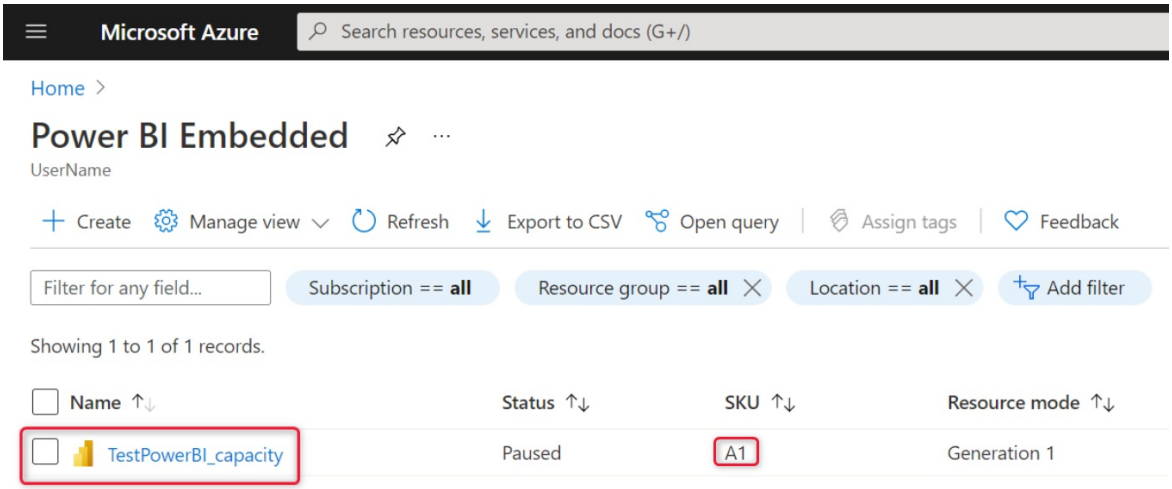
> **NOTE**
> With Gen1, a scaling operation can take about a minute. During this time, the capacity won't be available and embedded content may fail to load. With Embedded Gen2 you can scale your Power BI Embedded resource without experiencing any downtime.

Scale a capacity

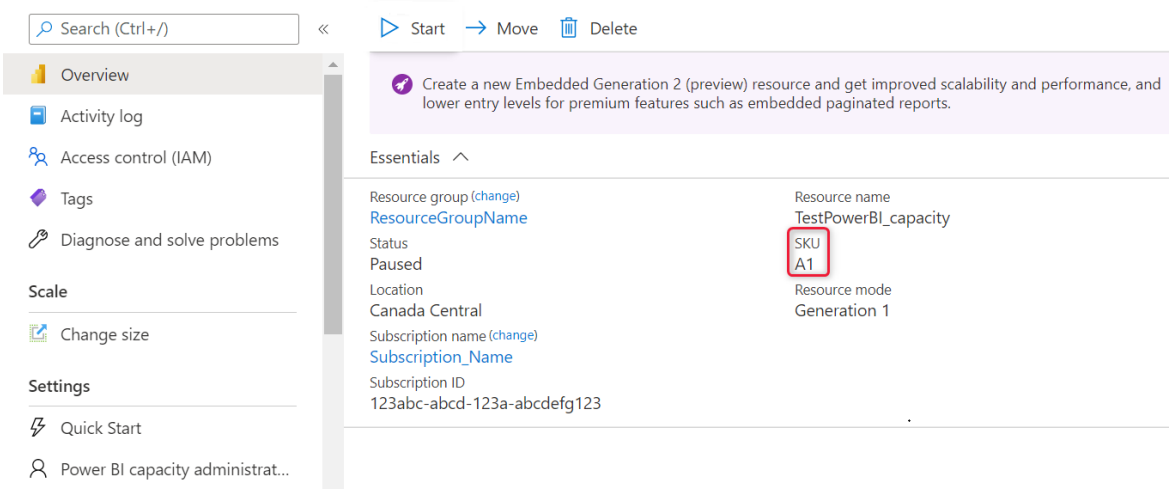
1. Sign into the Azure portal.
2. Under **Azure services**, select **Power BI Embedded** to see your capacities.



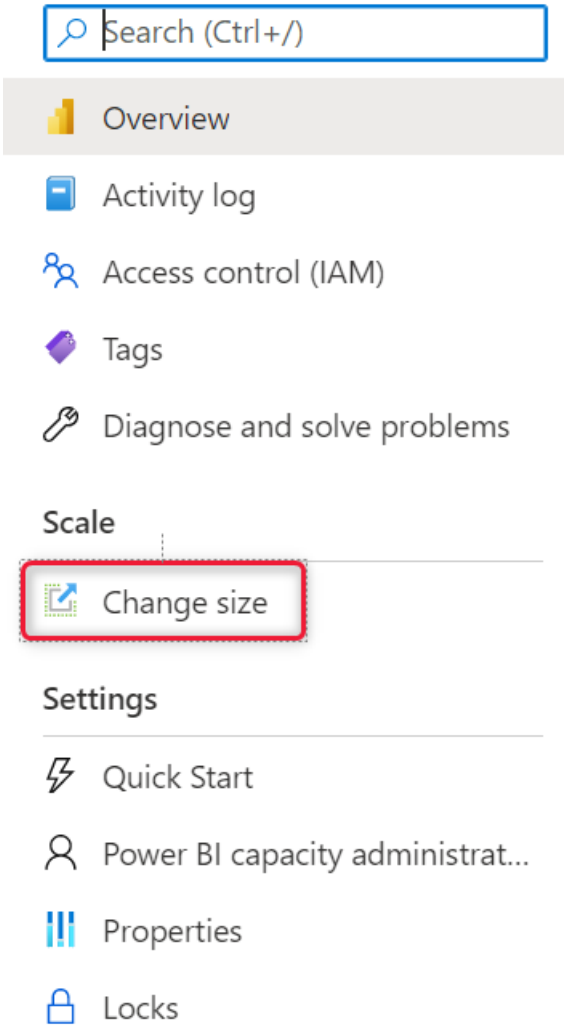
3. Select the capacity you want to scale. Notice that the current scale for each capacity is listed under **SKU**.



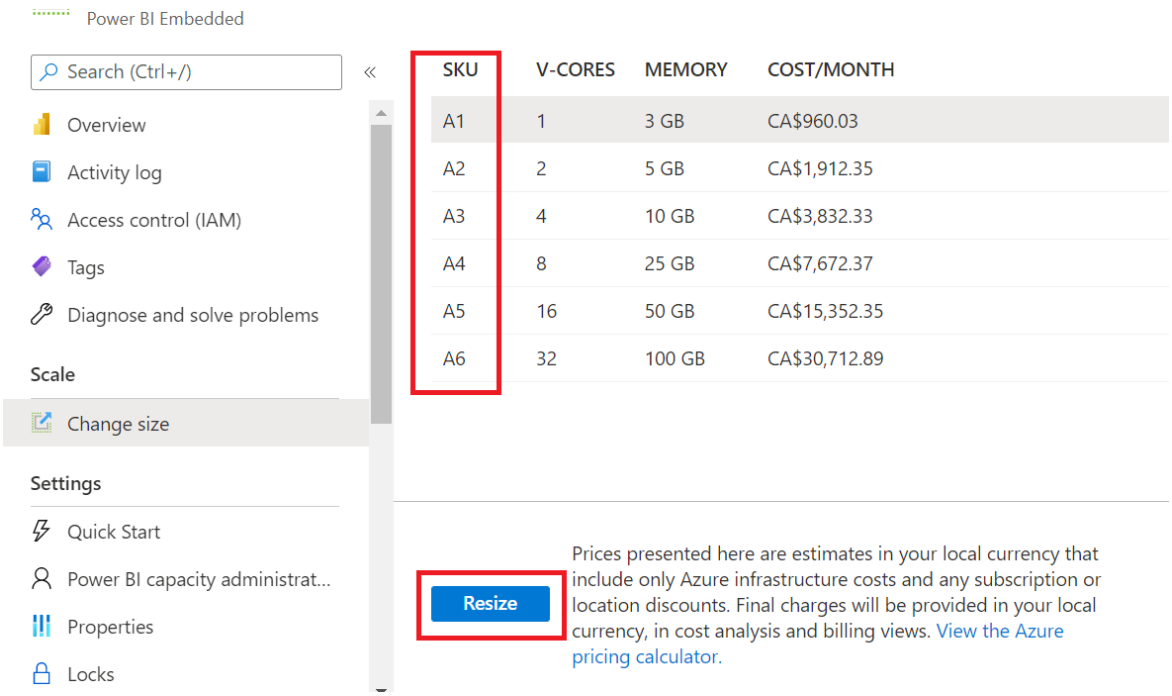
When you make your selection, information about that capacity is displayed next to it. This information again includes the current scaling under **SKU**.



4. Under **Scale**, select **Change size**.

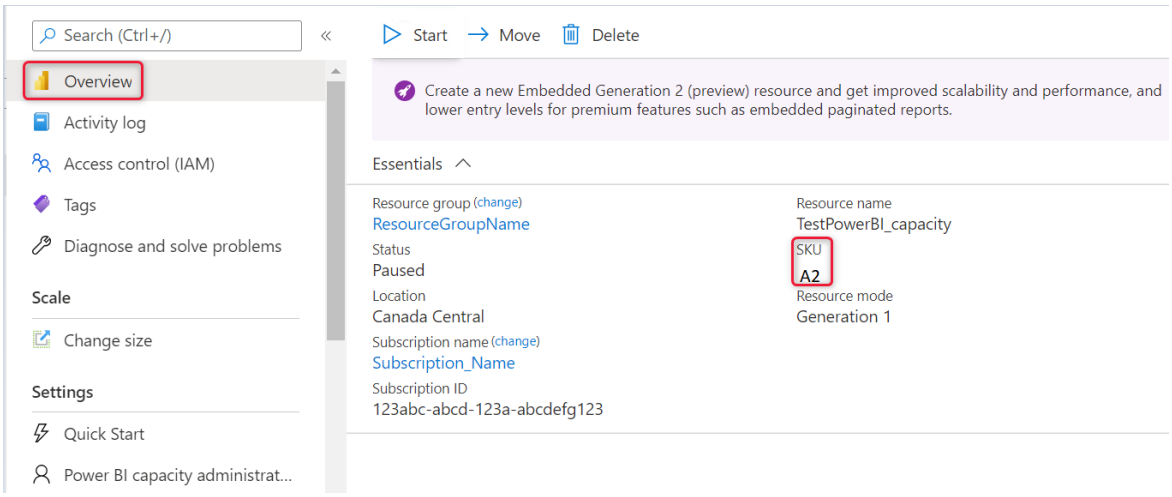


5. Select a scale and click **Resize**.



For Gen1 capacities, scaling might take a minute or two to complete.

6. Confirm your tier by viewing the overview tab. The current pricing tier is listed.



Next steps

Pause and start your Power BI Embedded capacity in the Azure portal

How to embed your Power BI dashboards, reports, and tiles.

More questions? Try asking the Power BI Community

Pause and start your Power BI Embedded capacity in the Azure portal

6/30/2022 • 2 minutes to read • Edit Online

This article walks through how to pause and start a Power BI Embedded capacity in Microsoft Azure. This assumes you have created a Power BI Embedded capacity. If you have not, see Create Power BI Embedded capacity in the Azure portal to get started.

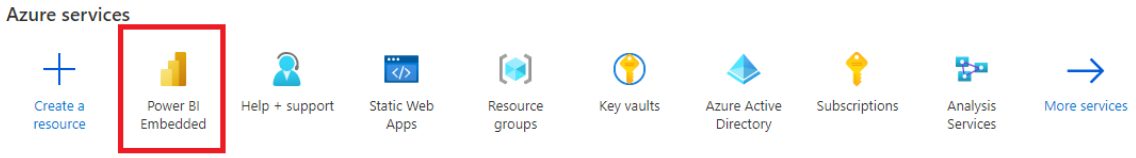
If you don't have an Azure subscription, create a free account before you begin.

Pause your capacity

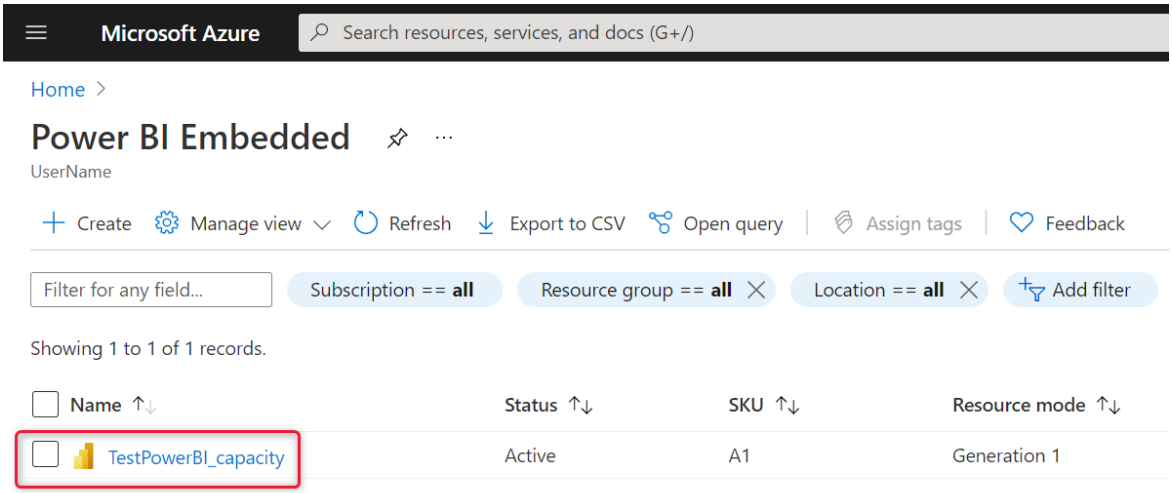
Pausing your capacity prevents you from being billed. Pausing your capacity is great if you don't need to use the capacity for a period of time. Use the following steps to pause your capacity.

> **NOTE**
> Pausing a capacity may prevent content from being available within Power BI. Make sure to unassign workspaces from your capacity before pausing to prevent interruption.

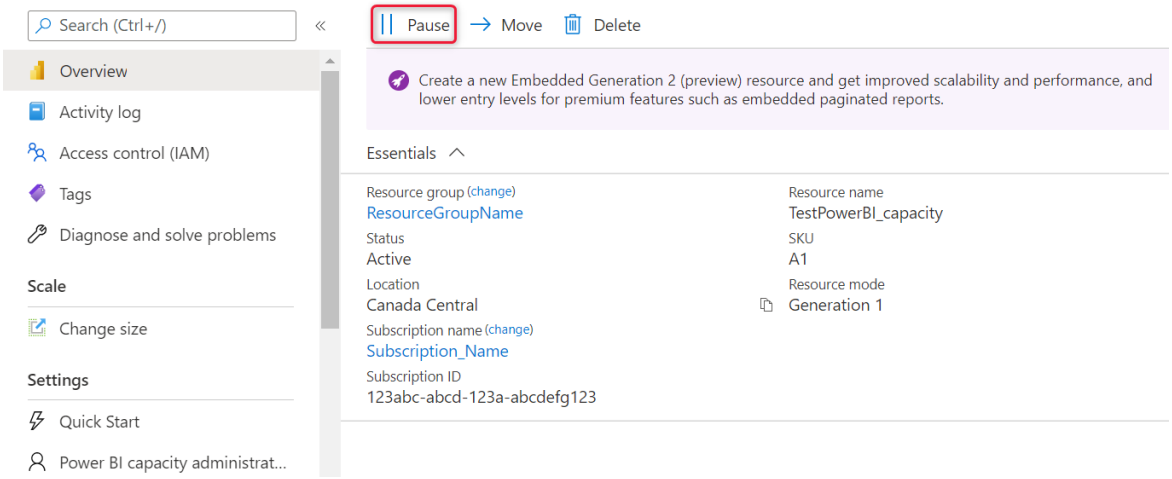
1. Sign into the Azure portal.
2. Under **Azure services**, select **Power BI Embedded** to see your capacities.



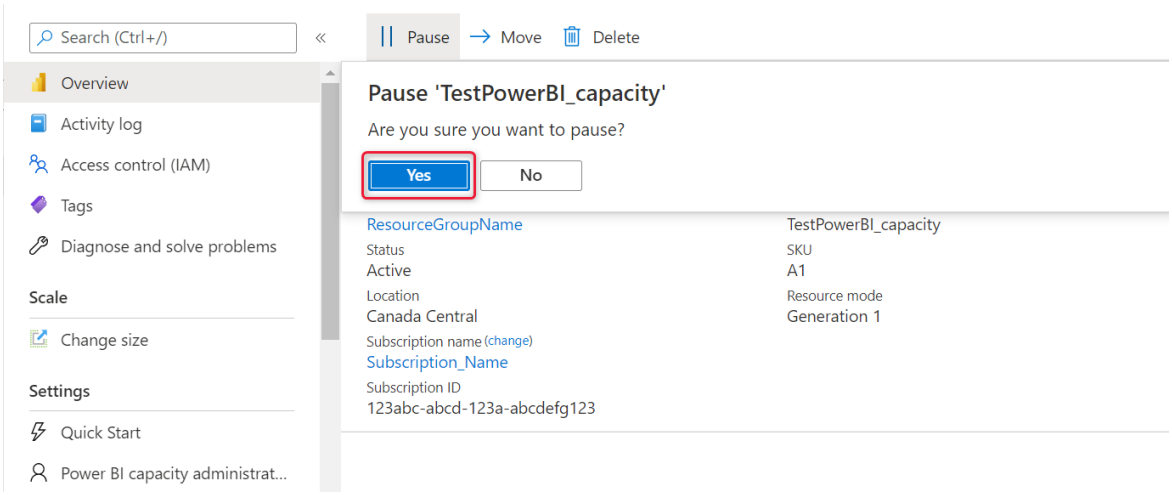
3. Select the capacity you want to pause.



4. Select **Pause** above the capacity details.



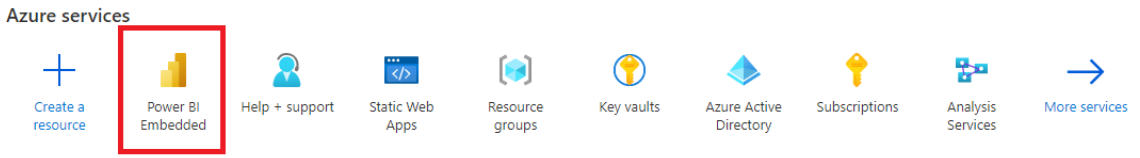
5. Select **Yes** to confirm you want to pause the capacity.



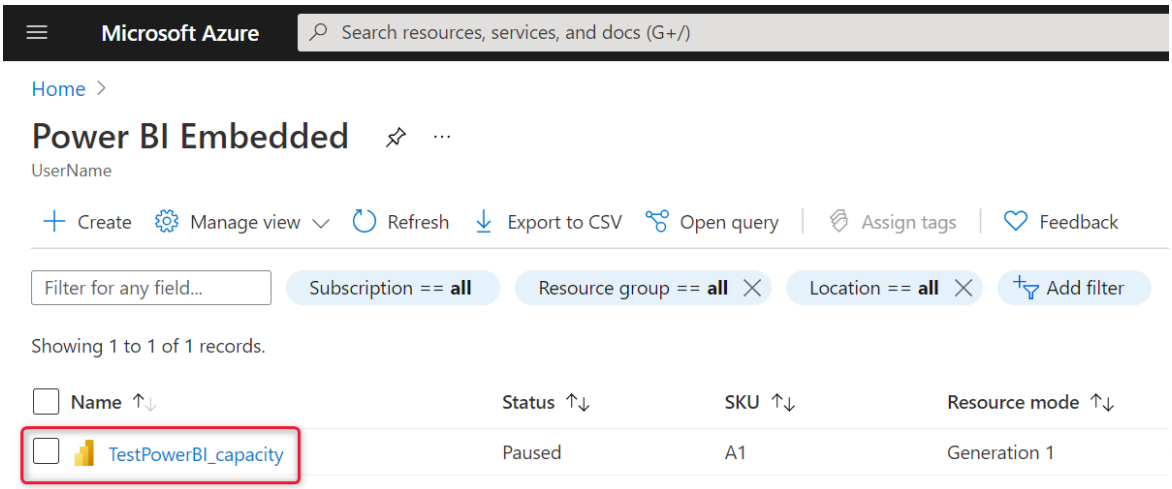
Start your capacity

Resume usage by starting your capacity. Starting your capacity also resumes billing.

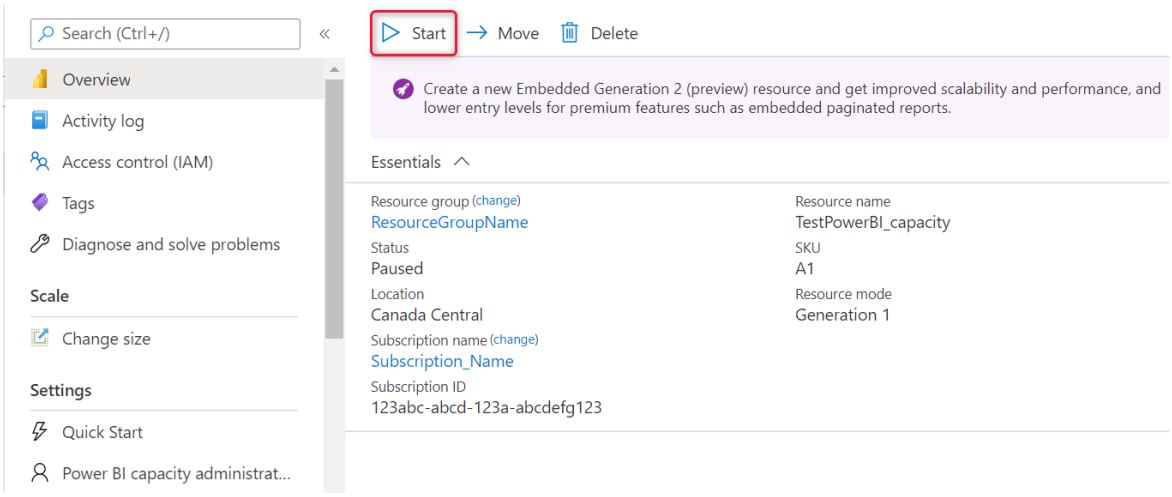
1. Sign into the Azure portal.
2. Select **All services** > **Power BI Embedded** to see your capacities.



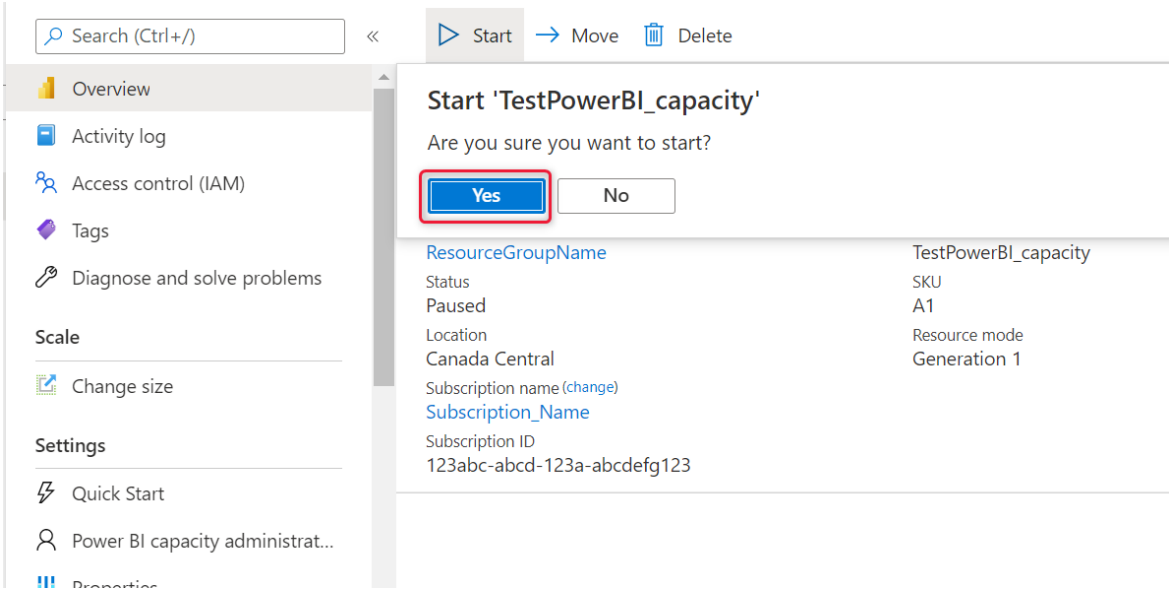
3. Select the capacity you want to start.



4. Select **Start** above the capacity details.



5. Select **Yes** to confirm you want to start the capacity.



If any content is assigned to this capacity, it becomes available once started.

Use CLI to start or pause your capacity

You can also start or pause your capacity from the command line using:

- ARM API references
- PS references
  - Suspend
  - Resume

Next steps

Scale your Power BI Embedded capacity.

How to embed your Power BI dashboards, reports and tiles.

More questions? Try asking the Power BI Community

Service principal profiles for multi-customer apps in Power BI Embedded

6/30/2022 • 13 minutes to read • Edit Online

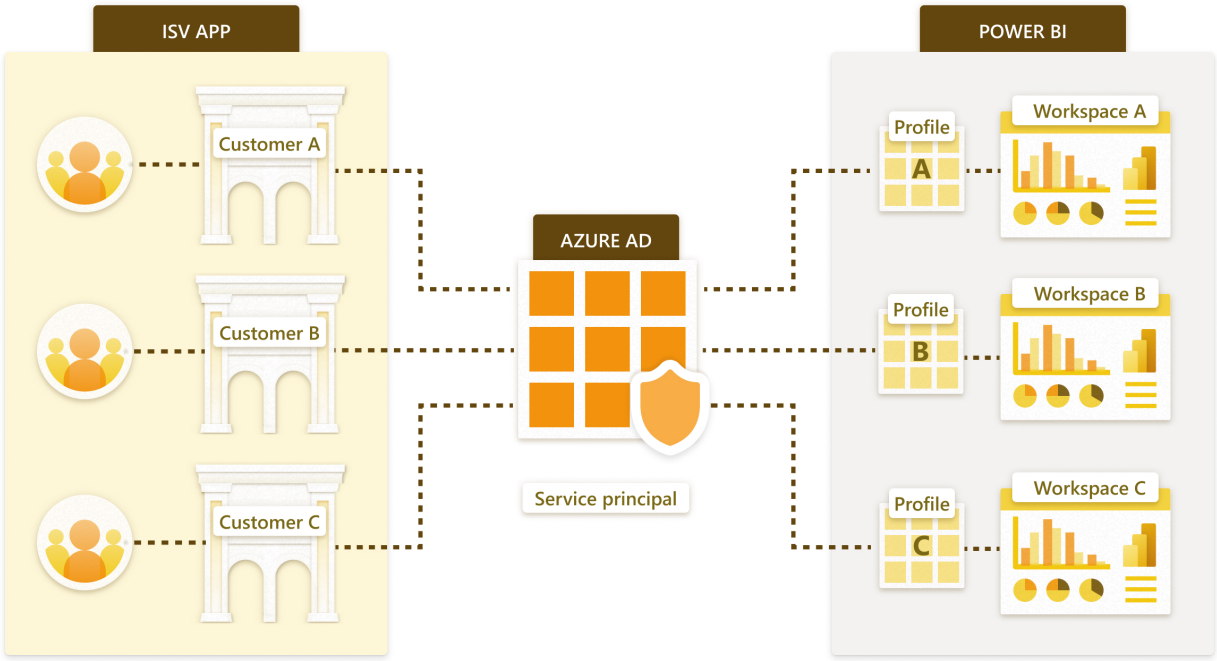
This article explains how an ISV or other Power BI Embedded app owner with many customers can use service principal profiles to map and manage each customer's data as part of their Power BI *embed for your customers* solution. Service principal profiles allow the ISV to build scalable applications that enable better customer data isolation and establish tighter security boundaries between customers. This article discusses the advantages and the limitations of this solution.

> **NOTE**
>
> For the sake of simplicity, all ISVs and Power BI Embedded Enterprise application owners will be referred to as ISVs in this article.

A *service principal profile* is a profile created by a service principal. The ISV application calls the Power BI APIs using a service principal profile, as explained in this article.

The ISV application service principal creates a different Power BI profile for each customer. When a customer visits the ISV app, the app uses the corresponding profile to generate an embed token that will be used to render a report in the browser.

Using service principal profiles enables the ISV application to host multiple customers on a single Power BI tenant. Each profile represents one customer in Power BI. In other words, each profile creates and manages Power BI content for one specific customer's data.



> **NOTE**
>
> This article is aimed at organizations that want to set up a multi-customer app using service principal profiles. If your organization already has an app that supports multiple customers from a single Power BI tenant, and you want to migrate to the service principal profile model, see Migrate multi-customer applications to the service principal profiles model.

Setting up your Power BI content involves the following steps:

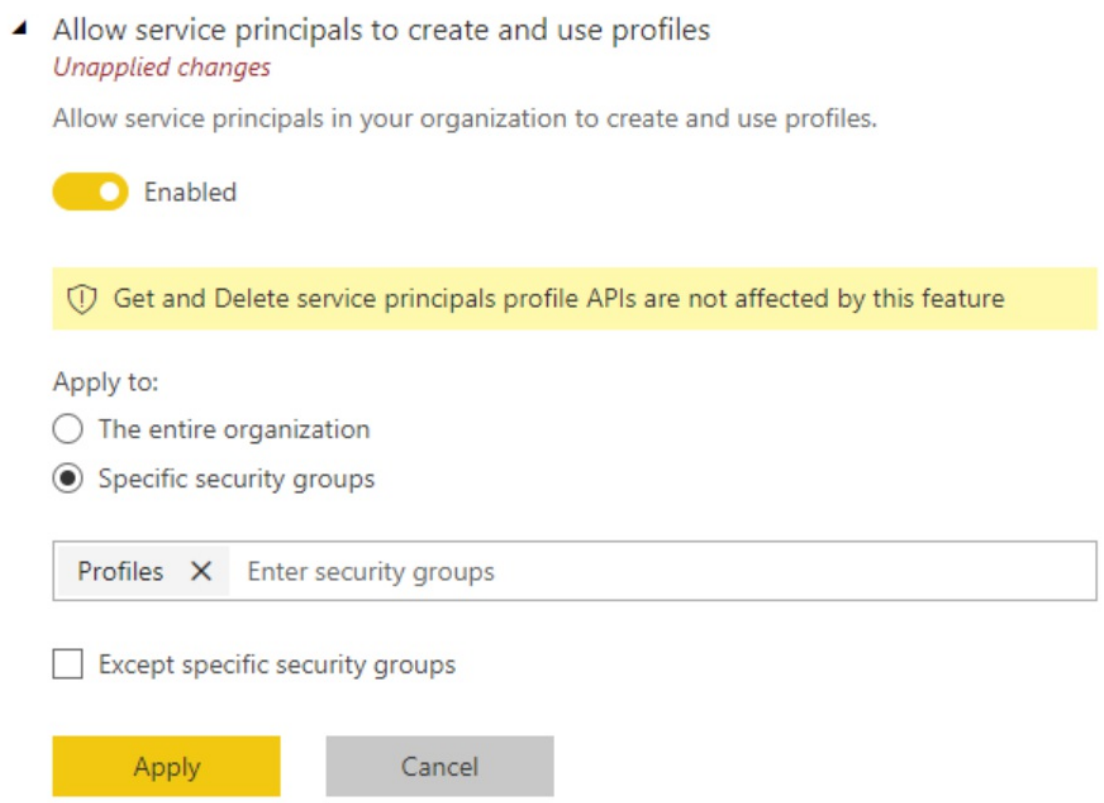
- Create a profile
- Set the profile permissions
- Create a workspace for each customer
- Import reports and datasets into the workspace
- Set the dataset connection details to connect to the customer's data
- Remove permissions from the service principal (optional, but helps with scalability)
- Embed a report into the application

All the above steps can be fully automated using Power BI REST APIs.

Prerequisites

Before you can create service principal profiles, you need to:

- Set up the service principal by following the *first three steps* of Embed Power BI content with service principal.
- From a Power BI tenant admin account, enable creating profiles in the tenant.



Create a profile

Profiles can be created, updated, and deleted using Profiles REST API.

For example, to create a profile:

```
POST https://api.powerbi.com/v1.0/myorg/profiles HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJK…UUPA
Content-Type: application/json; charset=utf-8

{"displayName":"ContosoProfile"}
```

A service principal can also call GET Profiles REST API to get a list of its profiles. For example:

```
GET https://api.powerbi.com/v1.0/myorg/profiles HTTP/1.1
Authorization: Bearer eyJ0eXA…UUPA
```

Profile permissions

Any API that grants a user permission to Power BI items can also grant a profile permission to Power BI items. For example, Add Group User API can be used to grant a profile permission to a workspace.

The following points are important to understand when using profiles:

- A profile belongs to the service principal that created it, and can only be used by that service principal.
- A profile owner can't be changed after creation.
- A profile isn't a standalone identity. It needs the service principal Azure AD token to call Power BI REST APIs.

ISV applications call Power BI REST APIs by providing the service principal Azure AD token in the *Authorization* header, and the profile ID in the *X-PowerBI-Profile-Id* header. For example:

```
GET https://api.powerbi.com/v1.0/myorg/groups HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUz.....SXBwasfg
X-PowerBI-Profile-Id: 5f1aa5ed-5983-46b5-bd05-999581d361a5
```

Create a workspace

Power BI workspaces are used to host Power BI items such as reports and datasets.

Each profile needs to:

- Create at least one Power BI workspace

  ```
  POST https://api.powerbi.com/v1.0/myorg/groups HTTP/1.1
  Authorization: Bearer eyJ0eXA…ZUiIsg
  Content-Type: application/json; charset=utf-8
  X-PowerBI-Profile-Id: a4df5235-6f18-4141-9e99-0c3512f41306

  {
    "name": "ContosoWorkspace"
  }
  ```

- Grant access permissions to the workspace
- Assign the workspace to a capacity

```
POST https://api.powerbi.com/v1.0/myorg/groups/f313e682-c86b-422c-a3e2-b1a05426c4a3/AssignToCapacity
HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJK…wNkZUiIsg
Content-Type: application/json; charset=utf-8
X-PowerBI-Profile-Id: a4df5235-6f18-4141-9e99-0c3512f41306

{
  "capacityId": "13f73071-d6d3-4d44-b9de-34590f3e3cf6"
}
```

Read more about Power BI workspaces.

Import reports and datasets

Use Power BI Desktop to prepare reports that are connected to the customer's data or sample data. Then, you can use the Import API to import the content into the created workspace.

```
POST https://api.powerbi.com/v1.0/myorg/groups/f313e682-c86b-422c-a3e2-b1a05426c4a3/imports?
datasetDisplayName=ContosoSales HTTP/1.1
Authorization: Bearer eyJ...kZUiIsg
Content-Type: multipart/form-data; boundary="8b071895-b380-4769-9c62-7e586d717ed7"
X-PowerBI-Profile-Id: a4df5235-6f18-4141-9e99-0c3512f41306
Fiddler-Encoding: base64

LS04YjA3MTg5NS1iMzgwLTQ3...Tg2ZDcxN2VkNy0tDQo=
```

Use dataset parameters to create a dataset that can connect to different customers' data sources. Then, use the *Update parameters* API to define which customers' data the dataset connects to.

Set the dataset connection

ISVs usually store their customers' data in one of two ways:

- A separate database for each customer
- A single multi-customer database

In either case, you should end up with single-customer datasets (one dataset per customer) in Power BI.

A separate database for each customer

If the ISV application has a separate database for each customer, create single-customer datasets in Power BI. Provide each dataset with connection details that point to its matching database. Use one of the following methods to avoid creating multiple identical reports with different connection details:

- **Dataset parameters:** Create a dataset with parameters in the connection details (such as SQL server name, SQL database name). Then, import a report into a customer's workspace and change the parameters to match the customer's database details.
- **Update Dataseource API:** Create a .pbix that points to a data source with sample content. Then, import the .pbix into a customer's workspace and change the connection details using the Update Datasource API.

A single multi-customer database

If the ISV application uses one database for all its customers, separate the customers into different datasets in Power BI as follows:

Create a report using parameters that only retrieve the relevant customer's data. Then, import a report into a customer's workspace and change the parameters to retrieve the relevant customer's data only.

Embed a report

After the setup is complete, you can embed customer reports and other items into your application using an embed token.

When a customer visits your application, use the corresponding profile to call the GenerateToken API. Use the generated embed token to embed a report or other items in the customer's browser.

To generate an embed token:

```
POST https://api.powerbi.com/v1.0/myorg/GenerateToken HTTP/1.1
Authorization: Bearer eyJ0eXAiOi…kZUiIsg
Content-Type: application/json; charset=utf-8
X-PowerBI-Profile-Id: a4df5235-6f18-4141-9e99-0c3512f41306

{
  "datasets": [
    {
      "id": "3b1c8f74-fbbe-45e0-bf9d-19fce69262e3"
    }
  ],
  "reports": [
    {
      "id": "3d474b89-f2d5-43a2-a436-d24a6eb2dc8f"
    }
  ]
}
```

Design aspects

Before setting up a profile-based multi-customer solution, you should be aware of the following issues:

- Scalability
- Automation & operational complexity
- Multi-Geo needs
- Cost efficiency
- Row level security

Scalability

By separating the data into separate datasets for each customer, you minimize the need for large datasets. When the capacity gets overloaded, it can evict unused datasets to free memory for active datasets. This optimization is impossible for a single large dataset. By using multiple datasets, you can also separate tenants into multiple Power BI capacities if necessary.

Without profiles, a service principal is limited to 1,000 workspaces. To overcome this limit, a service principal can create multiple profiles, where each profile can create up to 1,000 workspaces. With multiple profiles, the ISV app can isolate each customer's content using distinct logical containers.

Once a service principal profile has access to a workspace, its parent service principal's access to the workspace can be removed to avoid scalability problems.

Even with these advantages, you should consider the potential scale of your application. For example, the number of workspace items a profile can access is limited. Today, a profile has the same limits as a regular user. If the ISV application allows end users to save a personalized copy of their embedded reports, a customer's profile will have access to all the created reports of all its users. This model can eventually exceed the limit.

Automation and operational complexity

With Power BI profile-based separation, you might need to manage hundreds or thousands of items. Therefore,

it's important to define the processes that frequently happen in your application lifecycle management, and ensure you have the right set of tools to do these operations at scale. Some of these operations include:

- Adding a new customer
- Updating a report for some or all customers
- Updating the dataset schema for some or all customers
- Unplanned customizations for specific customers
- Frequency of dataset refreshes

For example, creating a profile and a workspace for a new customer is a common task, which can be fully automated with the Power BI REST API.

Multi-Geo needs

Multi-Geo support for Power BI Embedded means that ISVs and organizations that build applications using Power BI Embedded to embed analytics into their apps can now deploy their data in different regions around the world. To support different customers in different regions, assign the customer's workspace to a capacity in the desired region. This task is a simple operation that doesn't involve extra cost. However, if you have customers that need data from multiple regions, the customer profile should duplicate all items into multiple workspaces that are assigned to different regional capacities. This duplication may increase both cost and management complexity.

For compliance reasons, you may still want to create multiple Power BI tenants in different regions. Read more about multi-geo.

Cost efficiency

Application developers using Power BI Embedded need to purchase a Power BI Embedded capacity. The profile-based separation model works well with capacities because:

- The smallest object you can independently assign to a capacity is a workspace (you can't assign a report, for example). By separating customers by profiles, you get different workspaces - one per customer. This way, you get full flexibility to manage each customer according to their performance needs, and optimize capacity utilization by scaling up or down. For example, you can manage large and essential customers with high volume and volatility in a separate capacity to ensure a consistent level of service, while grouping smaller customers in another capacity to optimize costs.
- Separating workspaces also means separating datasets between customers so that data models are in smaller chunks, rather than a single large dataset. These smaller models enable the capacity to manage memory usage more efficiently. Small, unused datasets can be evicted after a period of inactivity, in order to maintain good performance.

When buying a capacity, consider the limit on the number of parallel refreshes, as refresh processes might need extra capacity when you have multiple datasets.

Row Level Security

This article explains how to use profiles to create a separate dataset for each customer. Alternatively, ISV applications can store all their customers' data in one large dataset and use Row-level security (RLS) to protect each customer's data. This approach can be convenient for ISVs that have relatively few customers and small to medium-sized datasets because:

- There's only one report and one dataset to maintain
- The onboarding process for new customers can be simplified

Before using RLS, however, make sure you understand its limitations. All the data for all customers are in one large Power BI dataset. This dataset runs in a single capacity with its own scaling and other limitations.

Even if you use service principal profiles to separate your customers' data, you can still use RLS within a single

customer's dataset to give different groups access to different parts of the data. For example, you could use RLS to prevent members of one department from accessing data of another department in the same organization.

Considerations and limitations

Service principal profiles are not supported with Azure Analysis Services (AAS) in live connection mode.

Power BI capacity limitations

- Each capacity can only use its allocated memory and V-cores, according to the SKU purchased. For the recommended dataset size for each SKU, reference Premium large datasets.
- To use a dataset larger than 10 GB, use a Premium capacity and enable the Large datasets setting.
- For the number of refreshes that can run concurrently on a capacity, reference resource management and optimization.
- Scaling a capacity in Gen 1, on average, takes between 1-2 minutes. During that time, the capacity isn't available. We recommend using a scale-out approach to avoid downtime. For Gen 2, scaling is instantaneous.

Manage service principals

Change a service principal

In Power BI, a profile belongs to the service principal that created it. That means, a profile can't be shared with other principals. With this limitation, if you want to change the service principal for any reason, you'll need to recreate all the profiles and provide the new profiles access to the relevant workspaces. Often, the ISV application needs to save a mapping between a profile ID and a customer ID in order to pick the right profile when needed. If you change the service principal and recreate the profiles, the IDs will also change, and you may need to update the mapping in the ISV application database.

Delete a service principal

> **WARNING**
> Be very careful when deleting a service principal. You don't want to accidentally lose data from all its associated profiles.

If you delete the service principal in the active directory, all its profiles in Power BI will be deleted. However, Power BI won't delete the content immediately, so the tenant admin can still access the workspaces. Be careful when deleting a service principal used in a production system, especially when you created profiles using this service principal in Power BI. If you do delete a service principal that has created profiles, be aware that you will need to recreate all the profiles, provide the new profiles access to the relevant workspaces, and update the profile IDs in the ISV application database.

Data separation

When data is separated by service principal profiles, a simple mapping between a profile and customer prevents one customer from seeing another customer's content. Using a single *service principal* requires that the service principal has access to all the different workspaces in all the profiles.

To add extra separation, you can assign a separate service principal to each customer, instead of having a single service principal access multiple workspaces using different profiles. This has several advantages, including:

- Human error or a credential leak won't cause multiple customers' data to be exposed.
- Certificate rotation can be done separately for each customer.

However, using multiple service principals comes with a high management cost. Select this path only if you need the extra separation. Keep in mind that the configuration of which data to show an end user is defined when you generate the embed token, a backend-only process that end users can't see or change. Requesting an embed token using a customer-specific profile will generate an embed token for that specific profile, which will give you customer separation in consumption.

One report, multiple datasets

Generally, you have one report and one dataset per customer. If you have hundreds of reports, you'll have hundreds of datasets. Sometimes, you might have one report format, with different datasets (for example, different parameters or connection details). Each time you have a new version of the report, you'll need to update all the reports for all customers. Although you can automate this, it's easier to manage if you have just one copy of the report. This can be achieved by creating a workspace that contains the report to embed. During a session runtime, you can bind the report to the customer-specific dataset. Read dynamic bindings for more details.

Customizing and authoring content

When you create content, carefully consider who you allow to edit it. If you permit multiple users in each tenant to edit it's easy to exceed dataset limitations. If you decide to give users editing capability, we recommend monitor their content generation closely, and scale up as needed. For the same reason, we don't recommend using this capability for content personalization, where each user can make small changes to a report and save it for themselves. If the ISV application allows content personalization, consider introducing and communicating workspace retention policies for user-specific content. Retention policies make it easier to delete content when users move to a new position, leave the company, or stop using the platform.

System-Managed identity

Instead of a service principal, you can use a user-assigned or system-assignedmanaged identity to create profiles. Managed identities reduce the need for secrets and certificates.

Be careful when using a system-managed identity because:

- If a system-assigned managed identity is accidentally turned off, you'll lose access to the profiles. This situation is similar to a case where a service principal is deleted.
- A system-assigned managed identity is connected to a resource in Azure (web app). If you delete the resource, the identity will be deleted.
- Using multiple resources (different web apps in different regions) will result in multiple identities that need to be managed separately (each identity will have its own profiles).

Due to the above considerations, we recommend that you use a user-assigned managed identity.

Next steps

Learn more about service principals

Use the Power BI SDK with service principals

Migrate multi-customer applications to the service principal profiles model

Use the Power BI SDK with service principal profiles

6/30/2022 • 2 minutes to read • Edit Online

This article explains how to use the SDK with service principal profiles. There are two ways to connect a Power BI client to a service principal profile. You can:

- Create a client with a profile object ID
- Specify the profile ID in the API request call

Once the client is associated with a profile, you can get the current service principal profile from the Power BI client.

Create a Power BI client with a service principal profile

```
var profileObjectId = new Guid("81f24a6d-7ebb-4478-99c7-2c36f7870a26");
var powerBIclient = new PowerBIClient(credentials, profileObjectId: profileObjectId);
```

When you create a Power BI client with the profile object ID, every API call that uses the client is issued with the `X-PowerBI-profile-id` in the request header.

For example,

```
GET https://powerbiapi.analysis-df.windows.net/v1.0/myorg/groups


Authorization: Bearer eyJ0eXAiO.....5U_g
X-PowerBI-profile-id: 81f24a6d-7ebb-4478-99c7-2c36f7870a26
```

Set profile on API request call

Alternatively, you can specify the profile ID in the API request by using the `customHeaders` property in the API's overloaded PowerBIClient method `WithHttpMessagesAsync`.

```
var powerBIclient = new PowerBIClient(credentials);
var profileHeader = new Dictionary<string, List<string>>();
profileHeader.Add("X-PowerBI-profile-id", new List<string> { "81f24a6d-7ebb-4478-99c7-2c36f7870a26" });
var groups = await powerBIclient.Groups.GetGroupsWithHttpMessagesAsync(customHeaders: profileHeader);
```

For example,

```
GET https://powerbiapi.analysis-df.windows.net/v1.0/myorg/groups

Authorization: Bearer eyJ0eXAiO.....5U_g
X-PowerBI-profile-id: 81f24a6d-7ebb-4478-99c7-2c36f7870a26
```

In the case, the profile header will *not* be added to the client default headers. You need to specify it with every API request.

Make sure you avoid duplications. For example, creating a client with a profile object ID and then specifying the header with the API request will result in unauthorized errors.

Get the current service principal profile from Power BI client

To retrieve the current service principal profile from the SDK client, call `GetServicePrincipalProfileObjectId`.

```
var profileObjectId = new Guid("81f24a6d-7ebb-4478-99c7-2c36f7870a26");
var powerBIclient = new PowerBIClient(credentials, profileObjectId: profileObjectId);
var currentProfileObjectId = powerBIclient.GetServicePrincipalProfileObjectId();
```

Considerations and Limitations

Service principal profiles are not supported with Azure Analysis Services (AAS) in live connection mode.

Next steps

Service principal profiles in Power BI Embedded

Migrate multi-customer applications to the service principal profiles model

6/30/2022 • 4 minutes to read • Edit Online

This article describes how you can get better scalability by migrating your Power BI embedded analytics multi-customer apps to the service principal profiles model.

Service principal profiles make it easier to manage organizational content in Power BI and use your capacities more efficiently.

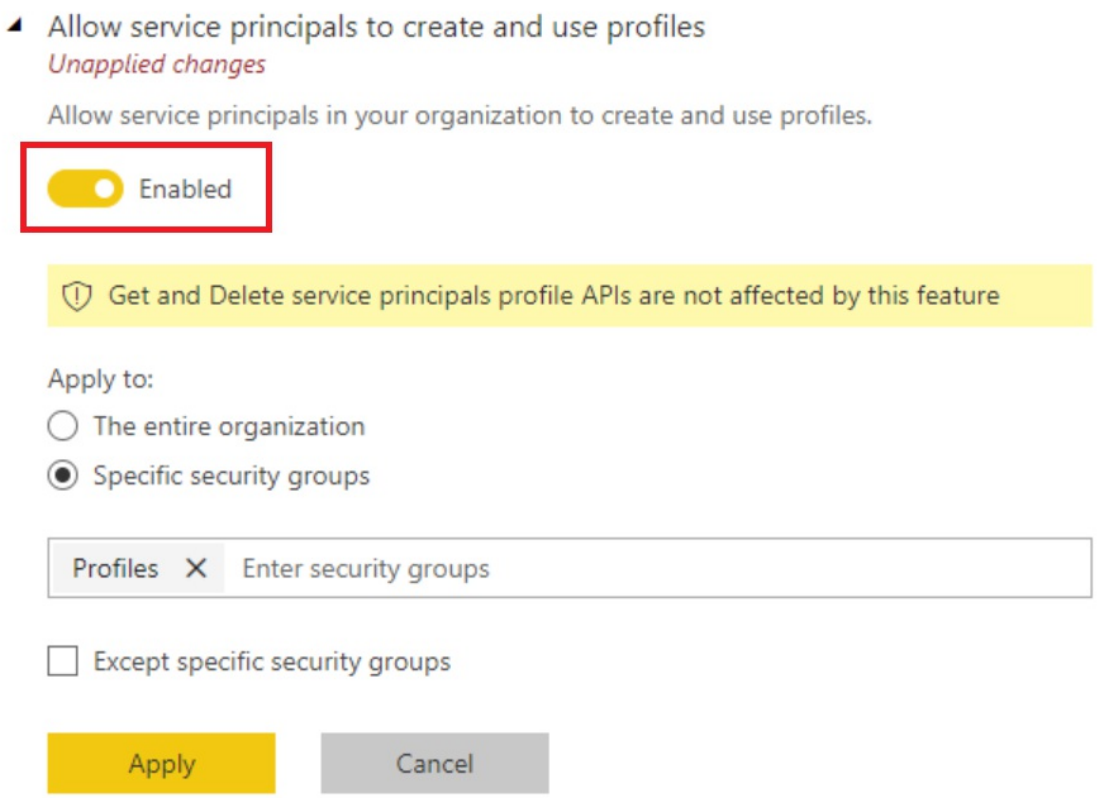
> **NOTE**
>
> This article is aimed at organizations that already have an app that supports multiple customers from a single Power BI tenant.
>
> Not all apps benefit from the service principal model. For example, the following apps shouldn't migrate:
>
> - Small apps that maintain one service principal with a small number of objects.
> - Apps that use one multiple service principal per customer

Prerequisites

It's important to read about service principal profiles before you start the migration.

You also need to do the following steps:

- Set up the service principal by following **the first three steps** of Embed Power BI content with service principal.
- From a Power BI tenant admin account, enable creating profiles in the tenant.



Migrate to service principal profiles

Migrating to service principal profiles involves the following steps:

1. Create profiles, one profile per customer.
2. Organize your workspaces.
3. Change the application code to use profiles.
4. Test your application with the profiles model.
5. Clean up redundant permissions.

Create Profiles (Required)

Use Profiles REST API with the service principal you created to create one profile for each customer.

It's a good idea to save a mapping of each data customer ID with its corresponding profile ID in your database. You'll need this mapping later to make API calls with the tenant profile.

Organize your workspaces

The easiest way to manage your data is by maintaining one workspace per customer. If your app already uses this model, you don't need to create new workspaces. However, you still have to provide each profile with access to the corresponding workspace using the Add Group User API.

If you don't have one workspace per customer, use the corresponding profile to call Create Group User API to create a new workspace for each customer.

Organize items in workspaces

You should now have a profile and a workspace for each customer. If you created new workspaces in the previous step, you need to import items (like reports and datasets) into these workspaces. The datasets you import depend on your current solution:

- If your app uses a separate dataset for each customer, the dataset design can work as it is.
- If your app uses one dataset withrow level security (RLS)to provide different data to different customers,

you can get better scalability by creating a separate dataset for each customer and using profiles as described in this article.

- After overcoming scalability limitations by using profiles and separate data sources, you can get even more data separation by using RLS with profiles.
  - If you rely on Dynamic RLS, the name of the profile will be returned in the DAX function `UserName()`.
  - If you use static RLS and override roles when generating the embed token, you can continue doing this.

Once the items are ready, import them into the relevant workspaces. To automate the process, consider using the Import API.

Change the application codes to use profiles

Once you have profiles with access to the relevant workspaces, and a database with mapping that shows you which profile represents which customer, you can make the necessary code changes. We recommend that you keep two code flows side by side and gradually expose the profiles code flow to your customers.

Make the following code changes:

- **Authorization code change**
  - If you're using a *master user* in the Azure AD app, change the acquire token code. Read embed with service principal to learn about creating an app-only Azure AD token.
  - If you're using a *service principal* and you created a new one for profiles, adjust the code to use the correct service principal ID and secrets.
- **Management code change**

  Some apps have management code that automates onboarding a new customer upon registration. Often, the management code uses Power BI REST APIs to create workspaces and import content. Most of this code should remain the same, but you may need to adapt the following details:
  - Each time you create a new customer tenant, create a new service profile to be the creator and administrator of the workspace for that tenant.
  - If you decide to reorganize your Power BI content, edit the code to reflect the changes.
- **Embed token code change**

  Replace the API caller. Make sure a profile calls the GenerateToken API because in the profiles model, only the specific profile has access to the customer's content.

Validate

It's best practice to test your app thoroughly before moving it to the profiles model. Reports may load even if there are bugs in the SaaS application code because you didn't delete the older permissions on the workspaces.

Clean up after migration

Now that you finished the migration and validated the results, remove what you don't need anymore.

- Clean up code: You might want to disable old code paths to ensure that you're only running new code that relies on profiles.
- Clean up workspaces and permissions in Power BI: If you created new workspaces, you can delete the old workspaces that are no longer in use. If you reused the same workspaces, you may want to delete the older permissions (such as *master user* permissions) on the workspace.

Next steps

Manage service principal profiles

More questions? Try asking the Power BI Community

Row-level security with Power BI Embedded

6/30/2022 • 14 minutes to read • Edit Online

Row-level security (RLS) can be used to restrict user access to data within dashboards, tiles, reports, and datasets. Different users can work with those same items all while seeing different data. Embedding supports RLS.

If you're embedding for non-Power BI users (app owns data), which is typically an ISV scenario, then this article is for you! Configure the embed token to account for the user and role.

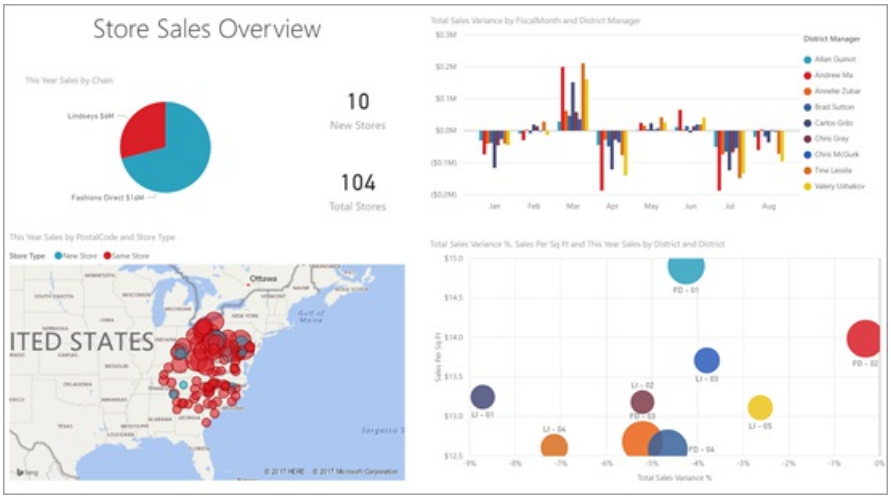
If you're embedding to Power BI users (user owns data), within your organization, RLS works the same as it does within the Power BI service directly. There's nothing more you need to do in your application. For more information, see Row-Level security (RLS) with Power BI.

To take advantage of RLS, it's important you understand three main concepts; Users, Roles, and Rules. Let's take a closer look at these concepts:

Users – End users viewing the item (dashboard, tile, report, or dataset). In Power BI Embedded, users are identified by the username property in an embed token.

Roles – Users belong to roles. A role is a container for rules and can be named something like *Sales Manager* or *Sales Rep*. You create roles within Power BI Desktop. For more information, see Row-level security (RLS) with Power BI Desktop.

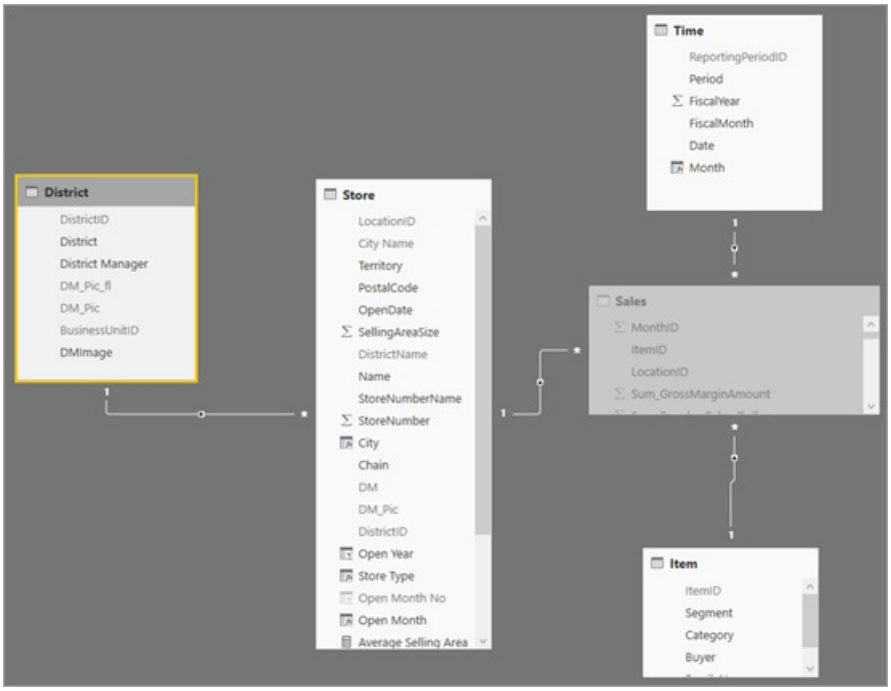
Rules – Roles have rules, and those rules are the actual filters that are going to be applied to the data. The rules could be as simple as "Country = USA" or something much more dynamic. For the rest of this article, there's an example of authoring RLS, and then consuming that within an embedded application. Our example uses the Retail Analysis Sample PBIX file.



Adding roles with Power BI Desktop

Our **Retail Analysis sample** shows sales for all the stores in a retail chain. Without RLS, no matter which district manager signs in and views the report, they all see the same data. Senior management has determined each district manager should only see the sales for the stores they manage. Using RLS allows Senior management to restrict data based on a district manager.

RLS is authored in Power BI Desktop. When the dataset and report are opened, we can switch to diagram view to see the schema:



Here are a few things to notice with this schema:

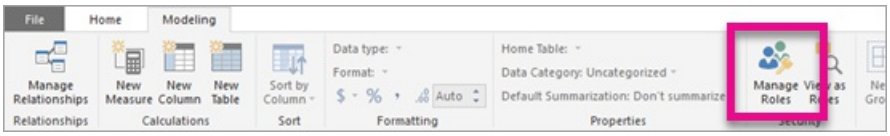
- All measures, like **Total Sales**, are stored in the **Sales** fact table.
- There are four additional related dimension tables: **Item**, **Time**, **Store**, and **District**.
- The arrows on the relationship lines indicate which way filters can flow from one table to another. For example, if a filter is placed on **Time[Date]**, in the current schema it would only filter down values in the **Sales** table. No other tables are affected by this filter since all the arrows on the relationship lines point to the sales table and not away.
- The **District** table indicates who the manager is for each district:

DistrictID	District	District Manager
1	FD - 01	Valery Ushakov
2	FD - 02	Tina Lassila
3	FD - 03	Carlos Grilo
4	FD - 04	Andrew Ma
6	LI - 01	Allan Guinot
7	LI - 02	Chris McGurk
9	LI - 03	Chris Gray
10	LI - 04	Brad Sutton
11	LI - 05	Annelie Zubar

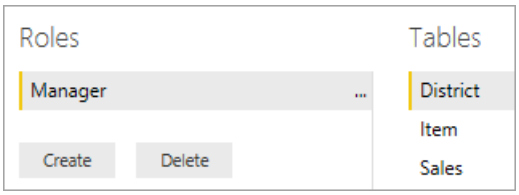
Based on this schema, if we apply a filter to the **District Manager** column in the **District** table, and if that filter matches the user viewing the report, that filter down the **Store** and **Sales** tables to show data for that district manager.

Here's how:

1. On the **Modeling** tab, select **Manage Roles**.

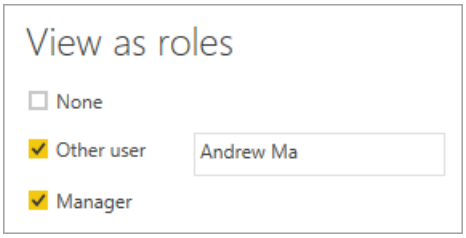


2. Create a new role called **Manager**.



3. In the **District** table, enter this DAX expression: **[District Manager] = USERNAME()**.

4. To make sure the rules are working, on the **Modeling** tab, select **View as Roles**, and then select both the **Manager** role you created, along with **Other users**. Enter **Andrew Ma** for the user.



   The reports show data as if you're signed in as **Andrew Ma**.

Applying the filter, the way we did here, filters down all records in the **District**, **Store**, and **Sales** tables. However, because of the filter direction on the relationships between **Sales** and **Time**, **Sales** and **Item**, and **Item** and **Time** tables aren't filtered down. To learn more about bidirectional cross-filtering, download the Bidirectional cross-filtering in SQL Server Analysis Services 2016 and Power BI Desktop whitepaper.

Applying user and role to an embed token

Now that you have your Power BI Desktop roles configured, some more work needs to be done in your application to take advantage of the roles.

Users are authenticated and authorized by your application and embed tokens are used to grant a user access to a specific Power BI Embedded report. Power BI Embedded doesn't have any specific information on who your user is. For RLS to work, you need to pass some additional context as part of your embed token in the form of identities. You can pass the identities by using the Embed Token API.

The API accepts a list of identities with indication of the relevant datasets. For RLS to work, you need to pass the below pieces as part of the identity.

- **username (mandatory)** – A string that can be used to help identify the user when applying RLS rules. Only a single user can be listed. Your username can be created with *ASCII* characters.
- **roles (mandatory)** – A string containing the roles to select when applying Row Level Security rules. If passing more than one role, they should be passed as a string array.
- **dataset (mandatory)** – The dataset that is applicable for the item you're embedding.

You can create the embed token by using the **GenerateTokenInGroup** method on **PowerBIClient.Reports**.

For example, you could change the *PowerBI-Developer-Samples > .NET Framework > Embed for your customers > **PowerBIEmbedded_AppOwnsData*** sample.

```
public EmbedToken GetEmbedToken(Guid reportId, IList<Guid> datasetIds, [Optional] Guid targetWorkspaceId)
    {
        PowerBIClient pbiClient = this.GetPowerBIClient();

        // Create a request for getting an embed token
        // This method works only with new Power BI V2 workspace experience
        var tokenRequest = new GenerateTokenRequestV2(
            reports: new List<GenerateTokenRequestV2Report>() { new GenerateTokenRequestV2Report(reportId)
},
            datasets: datasetIds.Select(datasetId => new
GenerateTokenRequestV2Dataset(datasetId.ToString())).ToList(),
            targetWorkspaces: targetWorkspaceId != Guid.Empty ? new
List<GenerateTokenRequestV2TargetWorkspace>() { new GenerateTokenRequestV2TargetWorkspace(targetWorkspaceId)
} : null,
            identities: new List<EffectiveIdentity> { rls }
        );

        // Generate an embed token
        var embedToken = pbiClient.EmbedToken.GenerateToken(tokenRequest);

        return embedToken;
    }
```

If you're calling the REST API, the updated API now accepts an additional JSON array, named **identities**, containing a username, list of string roles and list of string datasets.

Use the following code below as an example:

```
{
    "accessLevel": "View",
    "identities": [
        {
            "username": "EffectiveIdentity",
            "roles": [ "Role1", "Role2" ],
            "datasets": [ "fe0a1aeb-f6a4-4b27-a2d3-b5df3bb28bdc" ]
        }
    ]
}
```

Now, with all the pieces together, when someone logs into your application to view this item, they'll only see the data that they're allowed to see, as defined by our row-level security.

Working with Analysis Services live connections

Row-level security can be used with Analysis Services live connections for on-premises servers. There are a few specific concepts that you should understand when using this type of connection.

The effective identity that is provided for the username property must be a Windows user with permissions on the Analysis Services server.

> **NOTE**
>
> When using service principal with an Azure Analysis Services data source, the service principal itself must have an Azure Analysis Services instance permissions. Using a security group that contains the service principal for this purpose, doesn't work.

On-premises data gateway configuration

An On-premises data gateway is used when working with Analysis Services live connections. When generating an embed token, with an identity listed, the master account needs to be listed as an admin of the gateway. If the master account isn't listed, the row-level security isn't applied to the property of the data. A non-admin of the gateway can provide roles, but must specify its own username for the effective identity.

Use of roles

Roles can be provided with the identity in an embed token. If no role is provided, the username that was provided can be used to resolve the associated roles.

Using the CustomData feature

The CustomData feature allows you to add a Row filter by passing a free text (string) using the CustomData connection string property. Unlike users and roles, CustomData can't be set within a .pbix file.

CustomData can be used in a *role* DAX query, and can be used without a role in a *measure* DAX query.

The CustomData feature is part of token generation functionality for dashboard, report, and tile items. Dashboards can have multiple CustomData identities (one per tile/model or dataset).

> **NOTE**
>
> When generating a token with the CustomData feature, you must specify a username of no more than 256 characters.

CustomData SDK additions

The CustomData string property was added to our effective identity in the token generation scenario.

```
[JsonProperty(PropertyName = "customData")]
public string CustomData { get; set; }
```

The identity can be created with custom data using the following call:

```
public EffectiveIdentity(string username, IList<string> datasets, IList<string> roles = null, string
customData = null);
```

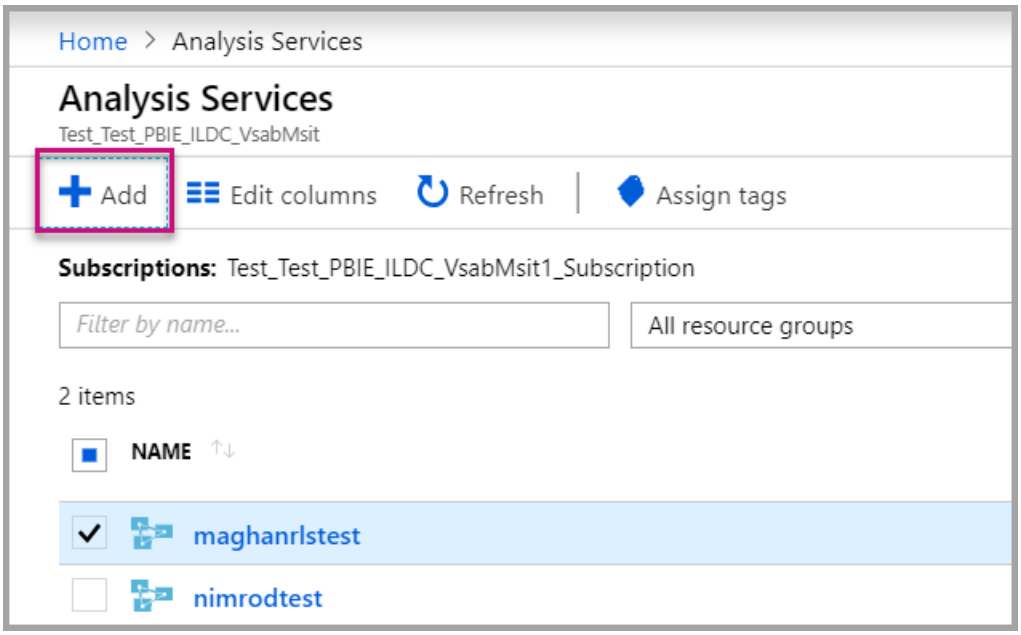
CustomData SDK usage

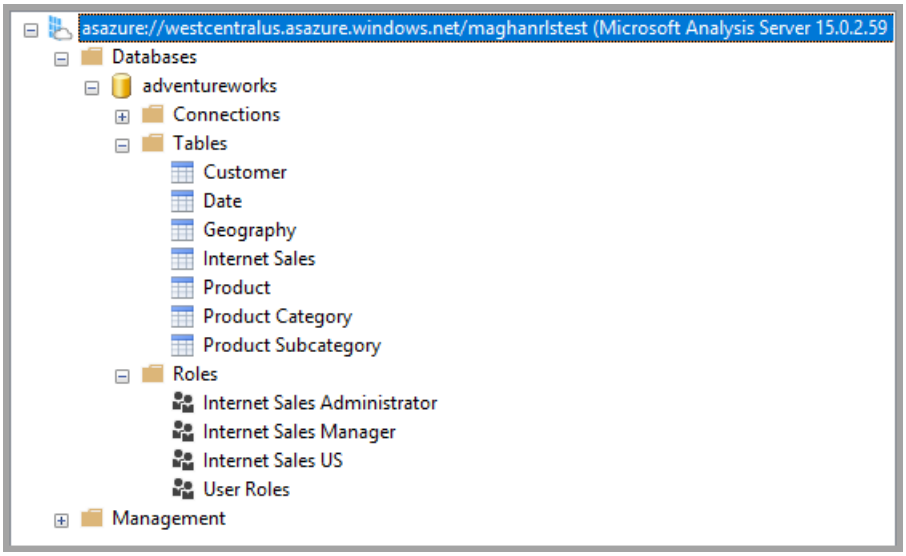
If you're calling the REST API, you can add custom data inside each identity, for example:

```
{
    "accessLevel": "View",
    "identities": [
        {
            "username": "EffectiveIdentity",
            "roles": [ "Role1", "Role2" ],
            "customData": "MyCustomData",
            "datasets": [ "fe0a1aeb-f6a4-4b27-a2d3-b5df3bb28bdc" ]
        }
    ]
}
```

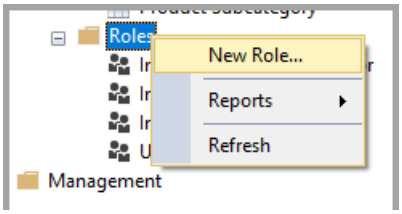
Here are the steps to begin setting up the CustomData() feature with your Power BI Embedded application.

1. Create your Azure Analysis Services database. Then sign in to your Azure Analysis Services server via SQL Server Management Studio.

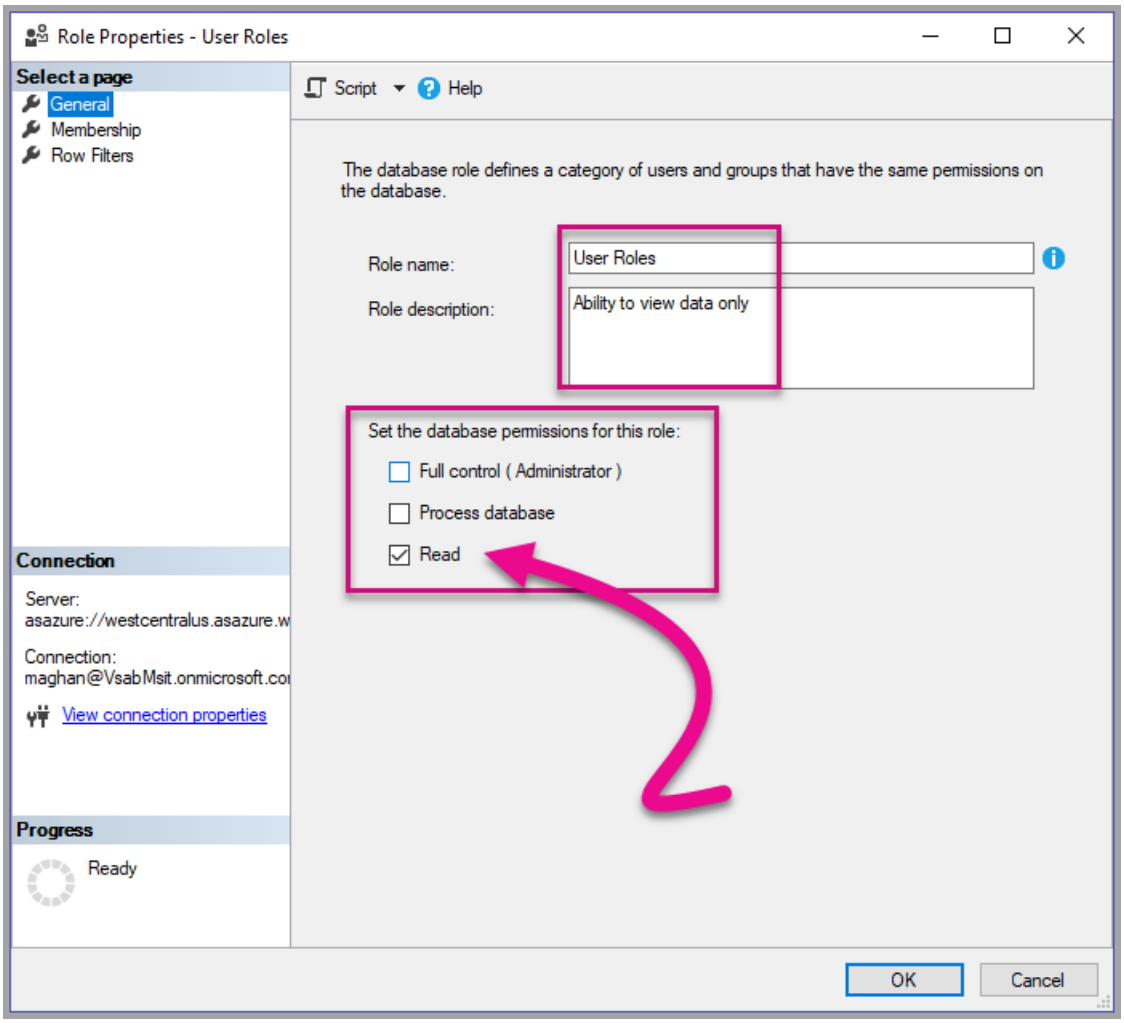




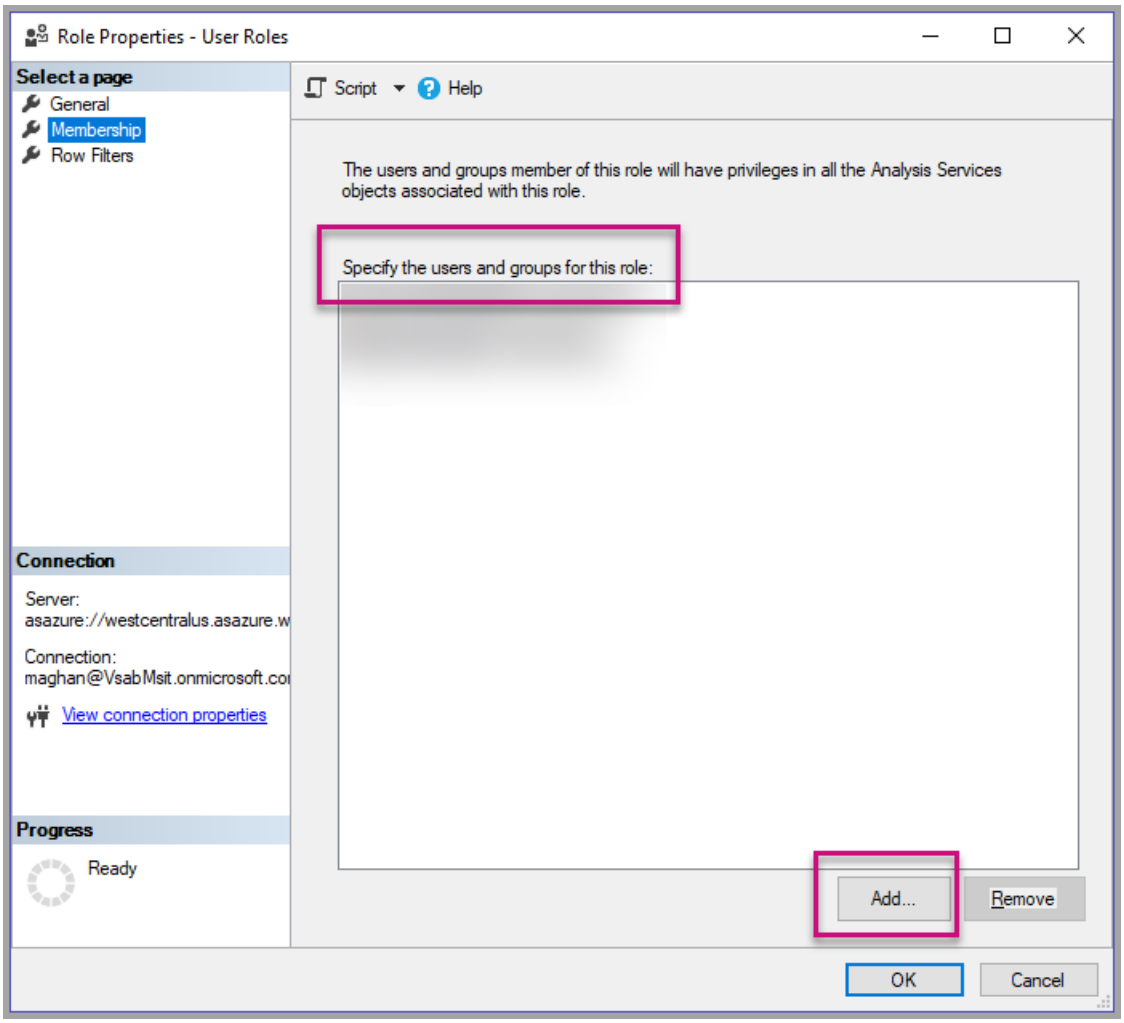
2. Create a Role in the Analysis Services server.



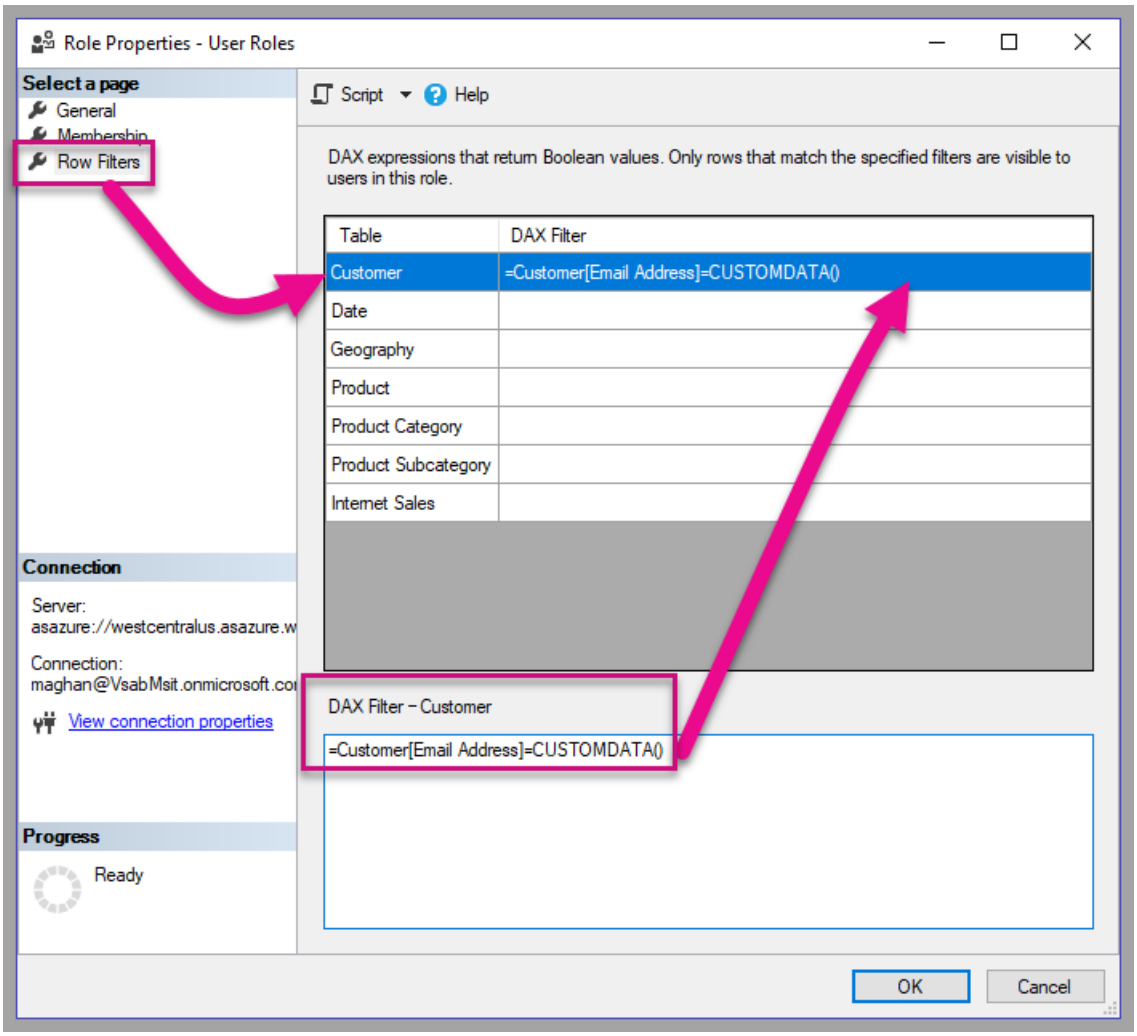
3. Set your **General** settings. Here you give the **Role Name** and set the database permissions to **Read** only.



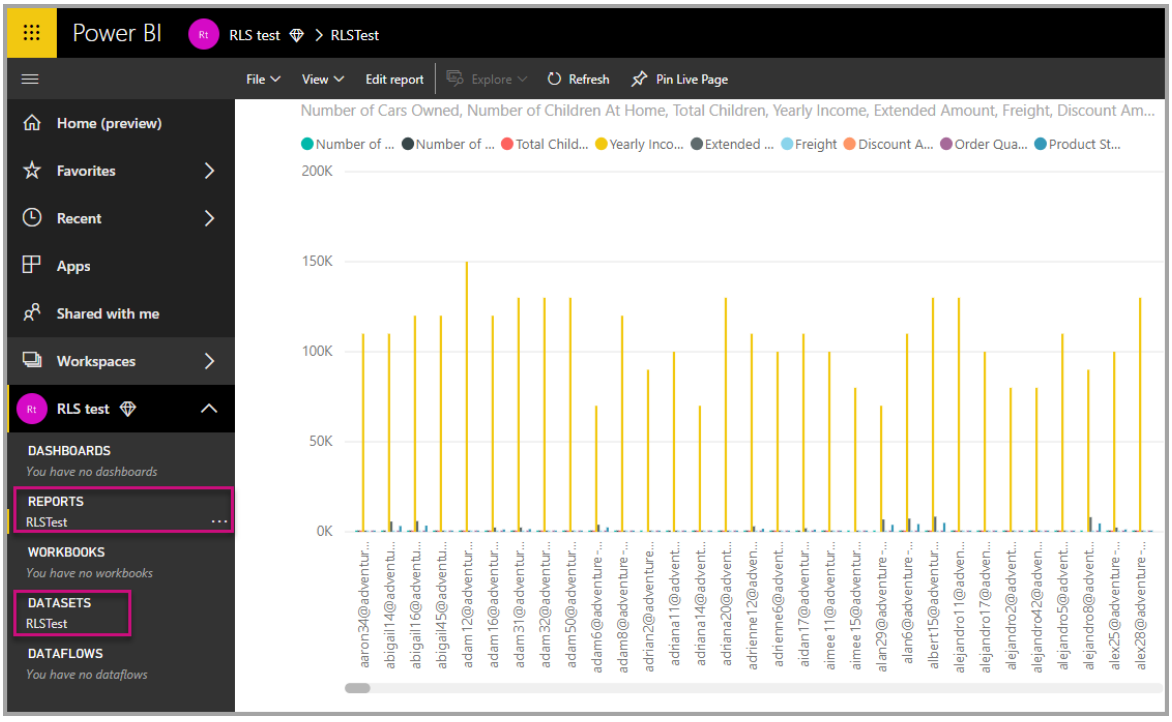
4. Set the **Membership** settings. Here you add te users that are affected by this role.



5. Set your **Row filters** DAX query using the *CUSTOMDATA()* function.



6. Build a PBI report and publish it to a workspace with capacity.



7. Use the Power BI APIs to use the CustomData feature in your application. When generating a token with the Custom data feature, you need to have a username. The username must be equal to the UPN of the master user. The master user must be a member of the role(s) you created. If no role(s) are specified, then all the roles the master user is a member of are used for RLS evaluation.

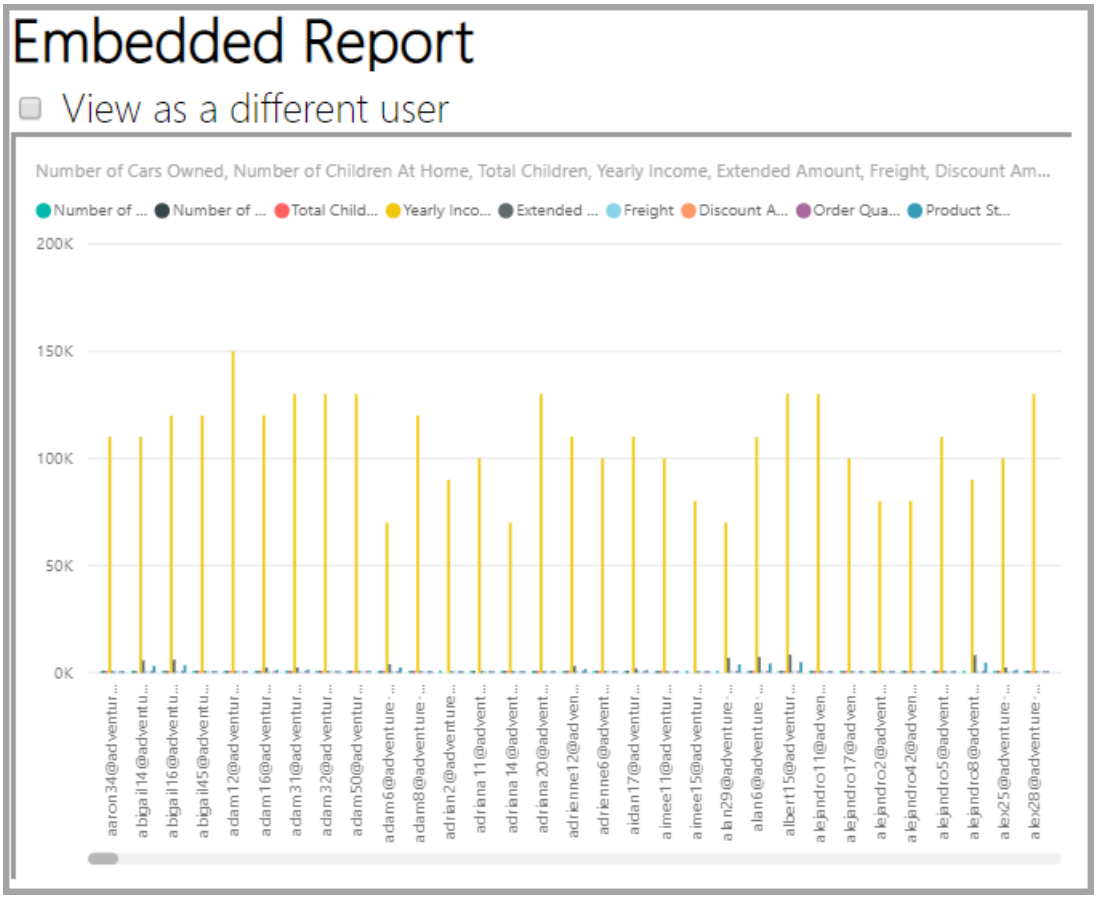
   When working with a service principal, you also need to do the above steps in place of using a master

account. When generating embed token, use the service principal object ID as the username.

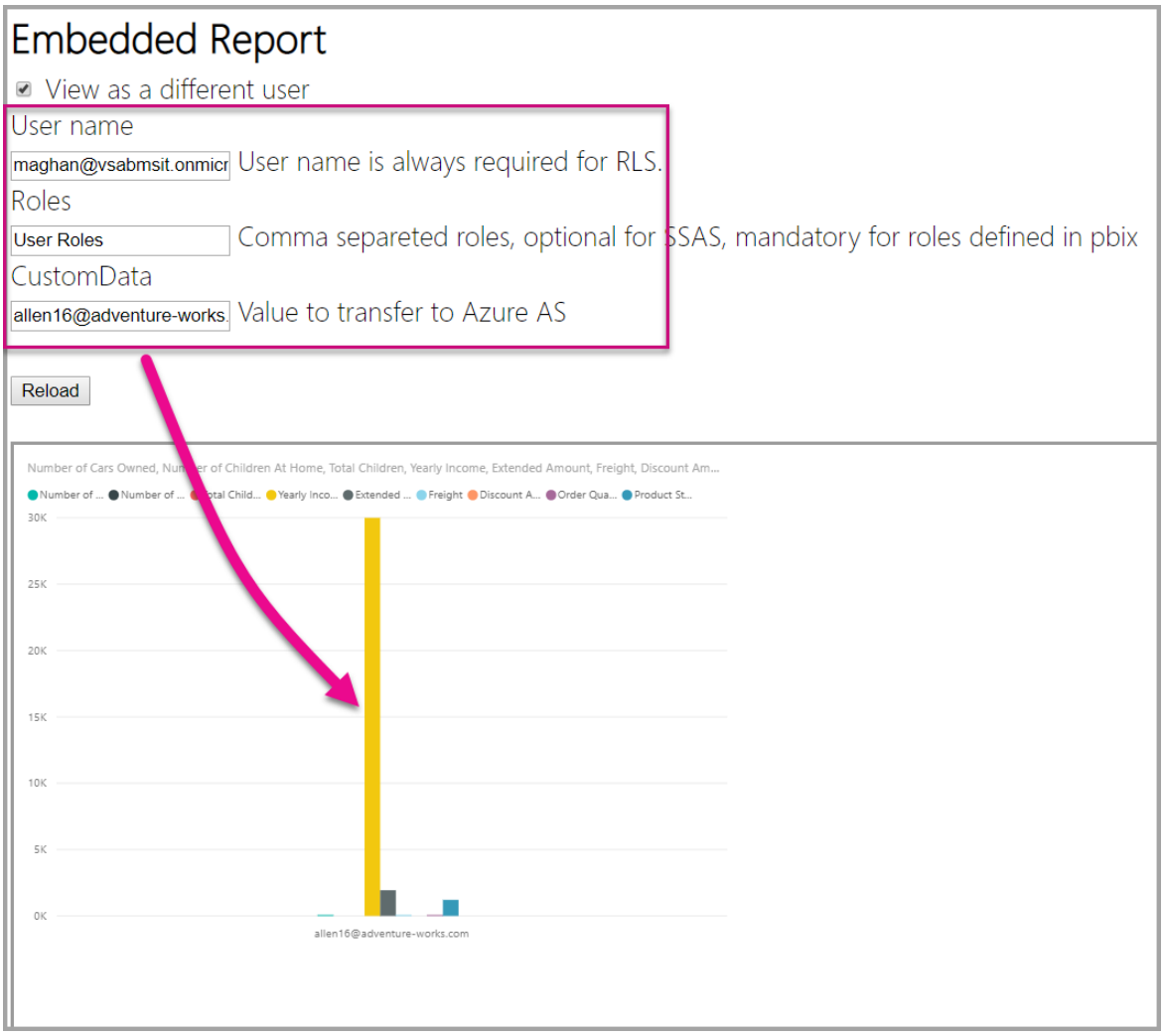
> **NOTE**
>
> When you're ready to deploy your application to production, the master user account field or option should not be visible to the end user.

View the code to add the CustomData feature.

8. Now you can view the report in your application before applying the Custom data value(s) to see all the data your report holds.



Then apply the Custom data value(s) to see how the report displays a different set of data.



Using RLS vs. JavaScript filters

When deciding on filtering your data in a report, you can use **row-level security (RLS)** or **JavaScript filters**.

Row-level security is a feature that filters data at the data model level. Your backend data source controls your RLS settings. Based on your data model, the embed token generation sets the username and the roles for the session. It cannot be overridden, removed, or controlled by the client-side code and that's why it's considered secure. We recommend using RLS for filtering data securely. You can filter data with RLS by using one of the options below.

- Configuring roles in a Power BI report.
- Configuring roles at the data source level (Analysis Services live connection only).
- Programmatically with an Embed Token using `EffectiveIdentity`. When using an embed token, the actual filter passes through the embed token for a specific session.

JavaScript filters are used to allow the user to consume reduced, scoped, or a filtered view of the data. However, the user still has access to the model schema tables, columns, and measures and potentially can access any data there. Restricted access to the data can only be applied with RLS and not through client-side filtering APIs.

Token-based Identity with Azure SQL Database

The **token-based identity** allows you to specify the effective identity for an embed token using an Azure Active Directory (AAD) access token for an **Azure SQL Database**.

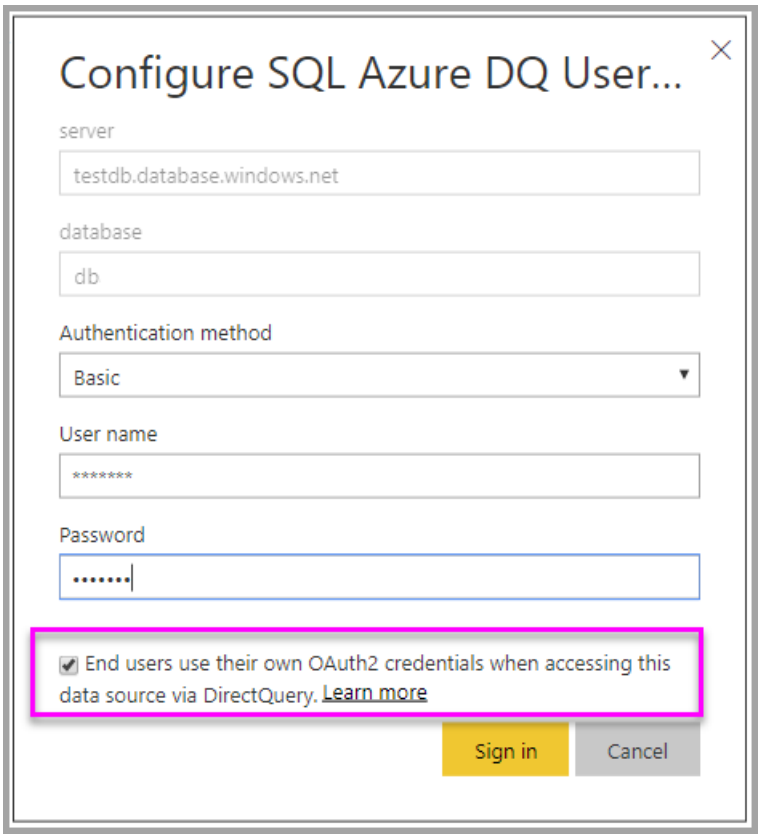
Customers that hold their data in **Azure SQL Database** can now enjoy a new capability to manage users and their access to data in Azure SQL when integrating with **Power BI Embedded**.

When you're generating the embed token, you can specify the effective identity of a user in Azure SQL. You can specify the effective identity of a user by passing the AAD access token to the server. The access token is used to pull only the relevant data for that user from Azure SQL, for that specific session.

It can be used to manage each user's view in Azure SQL or to sign in to Azure SQL as a specific customer in a multi-tenant DB. It can also apply row-level security on that session in Azure SQL and retrieve only the relevant data for that session, removing the need to manage RLS in Power BI.

Such effective identity issues apply to RLS rules directly on the Azure SQL Server. Power BI Embedded uses the provided access token when querying data from the Azure SQL Server. The UPN of the user (for which the access token was provided) is accessible as a result of the USER_NAME() SQL function.

The token-based identity only works for DirectQuery models on a capacity - connected to an Azure SQL Database, which is configured to allow AAD authentication (learn more about AAD authentication for Azure SQL Database). The dataset's data source must be configured to use end users' OAuth2 credentials, to use a token-based identity.



Token-based Identity SDK additions

The identity blob property was added to our effective identity in the token generation scenario.

```
[JsonProperty(PropertyName = "identityBlob")]
public IdentityBlob IdentityBlob { get; set; }
```

The IdentityBlob type is a simple JSON structure holding a value string property

```
[JsonProperty(PropertyName = "value")]
public string value { get; set; }
```

The EffectiveIdentity can be created with identity blob using the following call:

```
public EffectiveIdentity(string username, IList<string> datasets, IList<string> roles = null, string
customData = null, IdentityBlob identityBlob = null);
```

Identity blob can be created using the following call.

```
public IdentityBlob(string value);
```

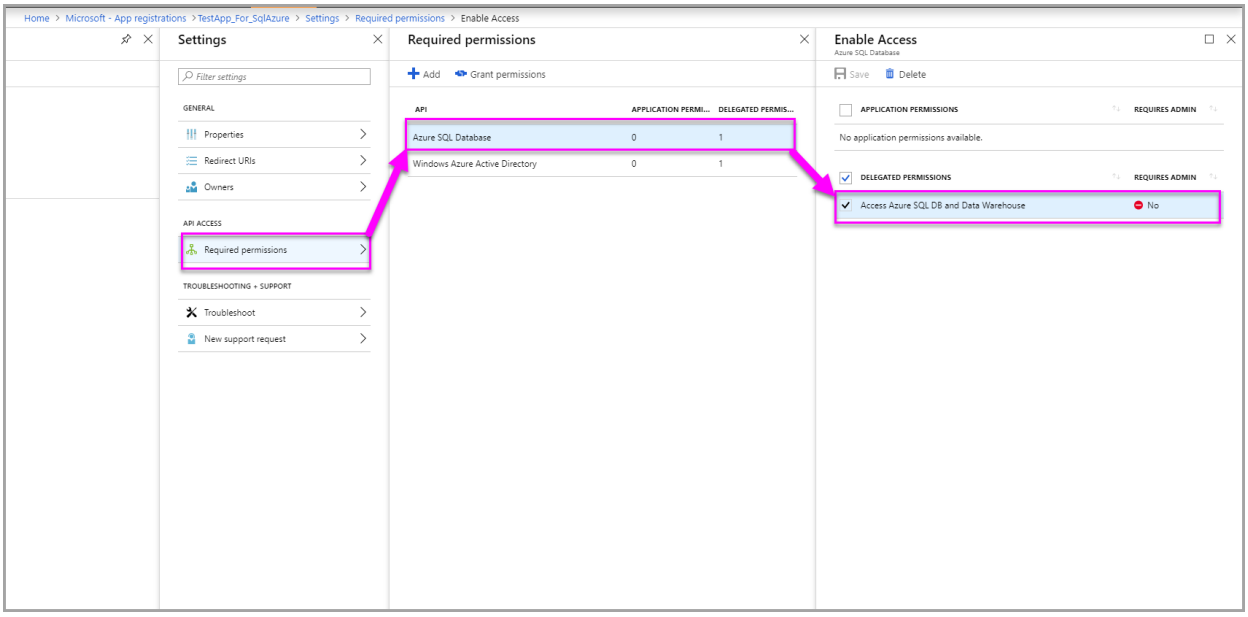
Token-based Identity REST API Usage

If you're calling the REST API, you can add identity blob inside each identity.

```
{
    "accessLevel": "View",
    "identities": [
        {
            "datasets": ["fe0a1aeb-f6a4-4b27-a2d3-b5df3bb28bdc"],
        "identityBlob": {
            "value": "eyJ0eXAiOiJKV1QiLCJh…."
         }
        }
    ]
}
```

The value provided in the identity blob should be a valid access token to Azure SQL Server.

> **NOTE**
>
> To be able to create an access token for Azure SQL, the application must have **Access Azure SQL DB and Data Warehouse** delegated permission to **Azure SQL Database** API on AAD app registration configuration in the Azure portal.



On-premises data gateway with service principal

Customers that are using SQL Server Analysis Services (SSAS) on-premises live connection data source, can enjoy the service principal capability to manage users and their access to data in SSAS when integrating with **Power BI Embedded**.

Using Power BI REST APIs, allows you to specify the effective identity for SSAS on-premises live connections for

an embed token using a service principal object.

Until now, to be able to specify the effective identity for SSAS on-premises live connection, the *master user* generating the embed token had to be a gateway admin. Now, instead of requiring the user to be gateway admin, the gateway admin can give the user dedicated permission to that data source, that allows the user to override the effective identity when generating the embed token. This new ability enables embedding with service principal for a live SSAS connection.

To enable this scenario, the gateway admin uses the Add Datasource User REST API to give the service principal the *ReadOverrideEffectiveIdentity* permission for the SSAS data source.

You can't set this permission using the admin portal. This permission is only set with the API. In the admin portal, you see an indication for users and SPNs with such permissions.

> **NOTE**
> If you are connected to an SSAS database without RLS configured on it, you still need to supply an effective identity (the identity of the SSAS server admin) in the embed token generation call.

Considerations and limitations

- Assignment of users to roles within the Power BI service doesn't affect RLS when using an embed token.
- While the Power BI service doesn't apply RLS setting to admins or members with edit permissions, when you supply an identity with an embed token, it applies to the data.
- Analysis Services live connections are supported for on-premises servers.
- Azure Analysis Services live connections support filtering by roles. Dynamic filtering can be done using CustomData.
- If the underlying dataset doesn't require RLS, the GenerateToken request must **not** contain an effective identity.
- If the underlying dataset is a cloud model (cached model or DirectQuery), the effective identity must include at least one role, otherwise role assignment doesn't occur.
- A list of identities enables multiple identity tokens for dashboard embedding. For all others items, the list contains a single identity.

Token-based Identity limitations

- You can use RLS only if you have a capacity.
- RLS doesn't work with SQL Server on-premises.

More questions? Try asking the Power BI Community

Embed a report with cloud-based RLS

6/30/2022 • 4 minutes to read • Edit Online

This article explains how to embed Power BI content that uses RLS into a standard Power BI app owns data application.

> **NOTE**
> This article is only relevant for app owns data customers.

Prerequisites

For a detailed explanation on how to set up RLS, refer to Row-level security (RLS) with Power BI.

When you define your RLS roles, keep in mind that the DAX expression you use determines if the RLS model is static or dynamic.

When to use static and dynamic security

Static security uses a fixed value in the DAX filter to define each role. It's simple to implement but difficult to maintain when there are many users or organizations involved.

Static security works best for an ISV that serves one or a few big customers where each department needs to access different data.

Dynamic security uses a DAX function (`username()` or `userprincipalname()`) to define the roles. Dynamic security provides more flexibility and allows you to manage your data using fewer roles and less maintenance.

Static security

With static roles, you pass the role to Power BI when you generate an embed token, and the user sees data according to that role. To create static security roles, enter a fixed value in the DAX filter.

For example, you can define the role of *Eastern US* as `[Region] = "East"`

Let's say john@contoso.com is a user of your app. You want to give John access to data from the *Eastern US* role. To embed a report for john@contoso.com, generate an embed token using the *Eastern US* role. The resulting data will be filtered for `[Region] = "East"`.

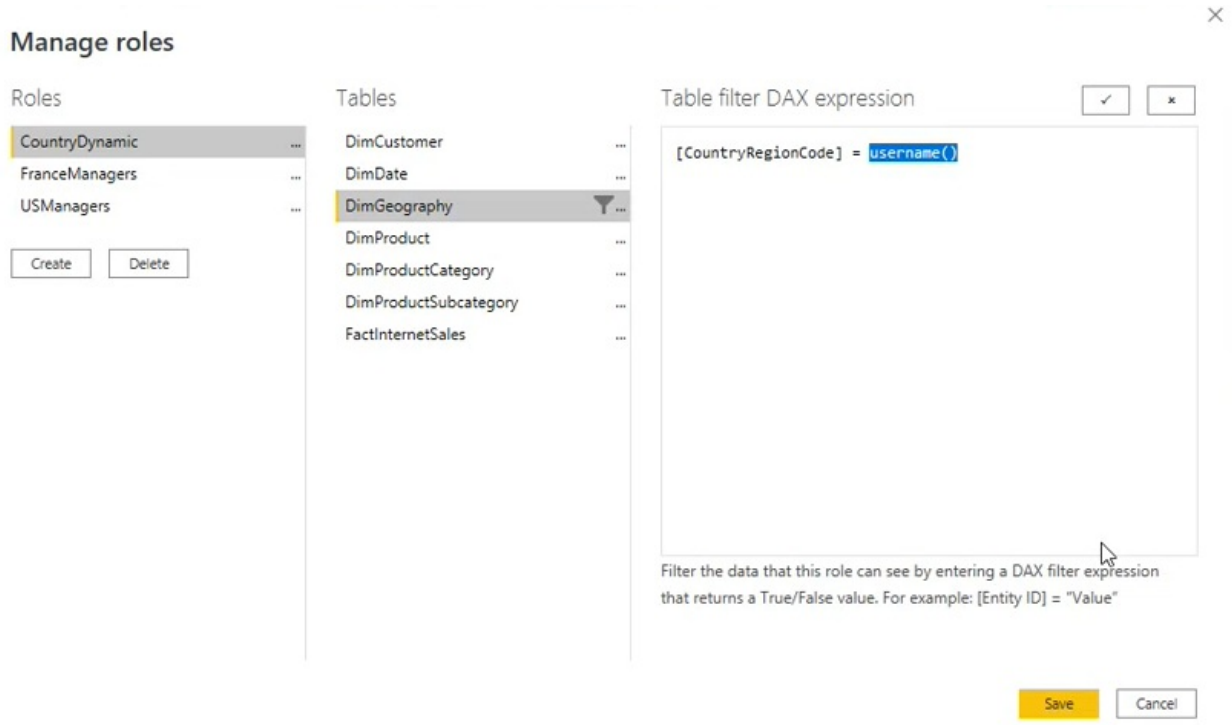
> **NOTE**
>
> When you generate the embed token, you need to supply a username, but the username can be any string. Static roles have a fixed value that isn't dependent on a username, so once the ISV determines the user's role and passes it to the embed token, the data is filtered according to that role regardless of what username was passed.

Dynamic security

Dynamic security uses the DAX function (`username()` or `userprincipalname()`) to define the role.

In the *user owns data* scenario, the RLS model automatically filters data based on the roles of the specific user. With *app owns data*, Power BI doesn't know the usernames of the ISV's customers, so you can use the `username()` function to dynamically filter the data.

Create a role in Power BI Desktop using the username() function. For example, you can create a role called *CountryDynamic* and define it as `[CountryRegionCode] = username()`



Let's say you want to give your user, jane@contoso.com, access to data for *France*. When you generate an embed token for jane@contose.com, you pass the string *France* as the *username* in the *CountryDynamic* role. Your data will be filtered according to [CountryRegionCode] = *France*.

```
{
    "accessLevel": "View",
    "identities": [
        {
            "username": "France",
            "roles": [ "CountryDynamic"],
            "datasets": [ "fe0a1aeb-f6a4-4b27-a2d3-b5df3bb28bdc" ]
        }
    ]
}
```

When using dynamic security in this scenario, you only need one role for all regions. The region name is used as the effective identity.

Generate an embed token

When you're ready to embed the report into your app, you need to generate an embed token. To generate a token using the Embed Token API, pass the following information to the API.

- **username** (mandatory) – If the roles are dynamic, the *username* string is used as the filter. For static roles, the *username* doesn't affect the RLS and can be any string at all. Only a single username can be listed.
- **roles** (mandatory) – The role(s) used when applying Row Level Security rules. If passing more than one role, they should be passed as a string array.
- **dataset** (mandatory) – The dataset that is applicable for the item you're embedding.

You can now embed your report into your app. The report will filter data according to the RLS applied.

```
public EmbedToken GetEmbedToken(Guid reportId, IList<Guid> datasetIds, [Optional] Guid targetWorkspaceId)
    {
        PowerBIClient pbiClient = this.GetPowerBIClient();

       // Defines the user identity and roles.
        var rlsIdentity = new EffectiveIdentity(
            username: "France",
            roles: new List<string>{ "CountryDynamic" },
            datasets: new List<string>{ datasetId.ToString()}
        );

        // Create a request for getting an embed token for the rls identity defined above
        // This method works only with new Power BI V2 workspace experience
        var tokenRequest = new GenerateTokenRequestV2(
            reports: new List<GenerateTokenRequestV2Report>() { new GenerateTokenRequestV2Report(reportId)
},
            datasets: datasetIds.Select(datasetId => new
GenerateTokenRequestV2Dataset(datasetId.ToString())).ToList(),
            targetWorkspaces: targetWorkspaceId != Guid.Empty ? new
List<GenerateTokenRequestV2TargetWorkspace>() { new GenerateTokenRequestV2TargetWorkspace(targetWorkspaceId)
} : null,
            identities: new List<EffectiveIdentity> { rlsIdentity }
        );

        // Generate an embed token
        var embedToken = pbiClient.EmbedToken.GenerateToken(tokenRequest);

        return embedToken;
    }
```

Considerations and limitations

- The user that generates the embed token has to be a *member* or *admin* in both workspaces (the dataset workspace and the report workspace).
- When generating the embed token, you need to provide a username and a role. If you don't, one of the following will occur, depending on if the token is being generated by service principal or master user:
  - For a **service principal**, token generation will fail.
  - For a **master user**, token generation will succeed but the data will not be filtered (all the data is returned).

Next steps

- RLS guidance
- Generate an embed token More Questions? Try the Power BI Community

Implementing row-level security in embedded paginated reports

6/30/2022 • 2 minutes to read • Edit Online

This article explains how to embed a paginated report that uses **RLS** (row-level security) into your *app owns data* application.

> **NOTE**
> This article is only relevant for app owns data customers.

To use RLS for your paginated reports:

1. Set up the environment to filter the report
2. Filter the data at report or query level
3. Pass the configured parameter using an embed token

Prerequisites

- This article assumes that you know how to embed a Power BI paginated report. It explains how to generate the embed token so that the report only shows what the user has permission to access.
- Paginated reports are created using the SQL Server Reporting Services engine, and not the Power BI (Analysis Services) engine, so the RLS filtering is set up in Power BI Report Builder.

Set up the environment

To apply row-level security to a Power BI paginated report, use the built-in field **UserID** to assign a parameter. This parameter will be used to filter or query your data.

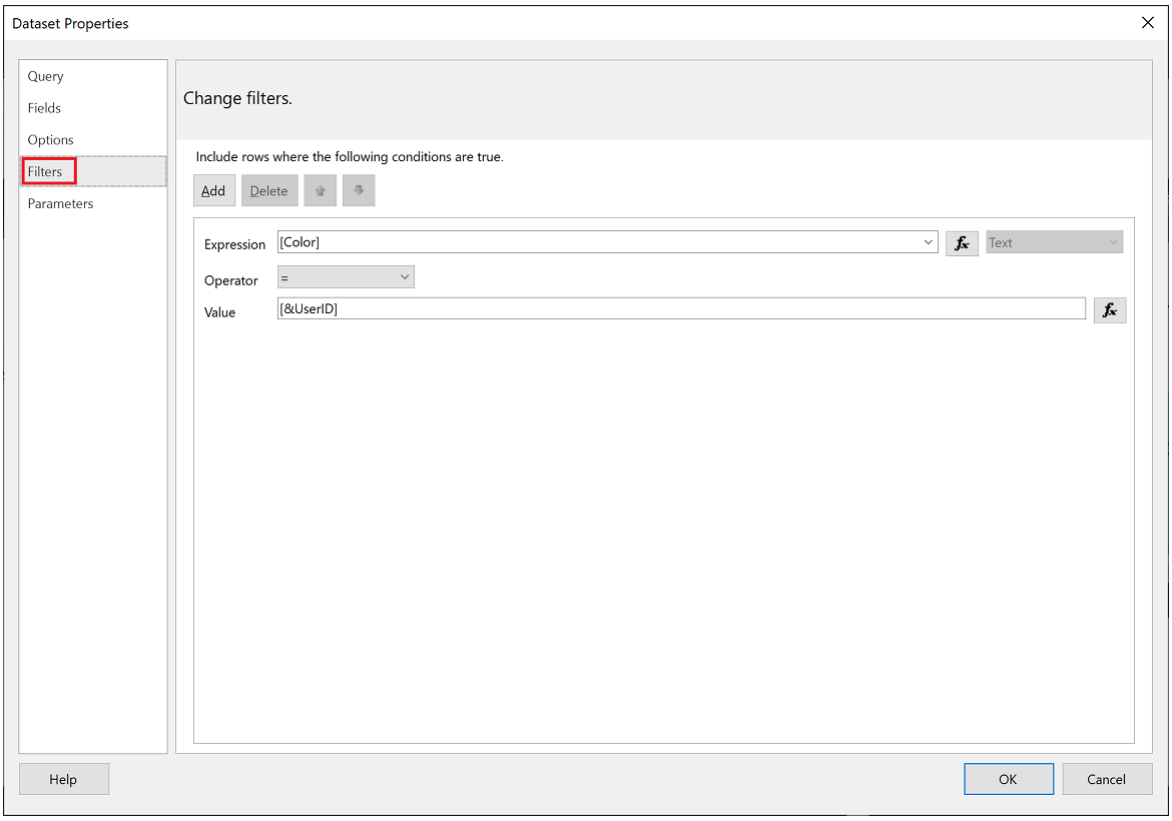
Then, pass the **UserID** to the Embed Token - Generate Token API to get the embed token.

Use UserID as a filter at report or query level

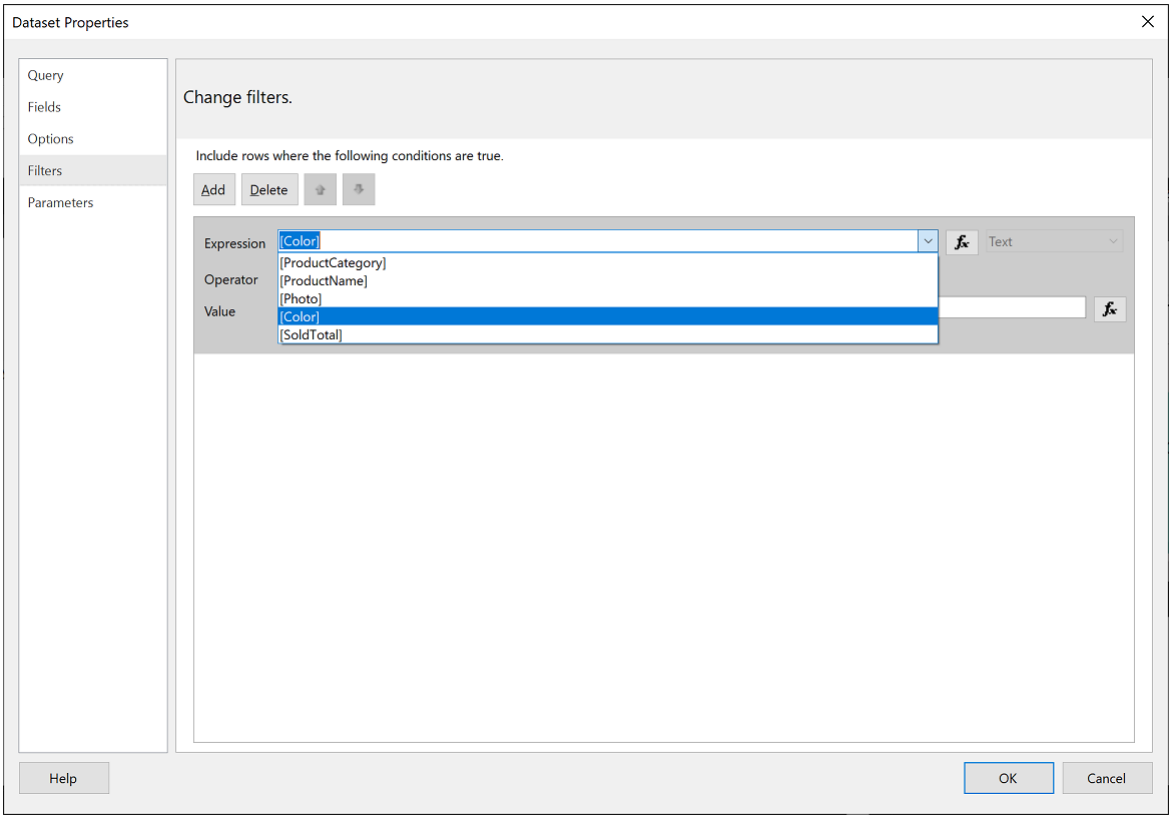
You can use **UserId** as a *filter* or in a *query* to the data source.

Filter the data

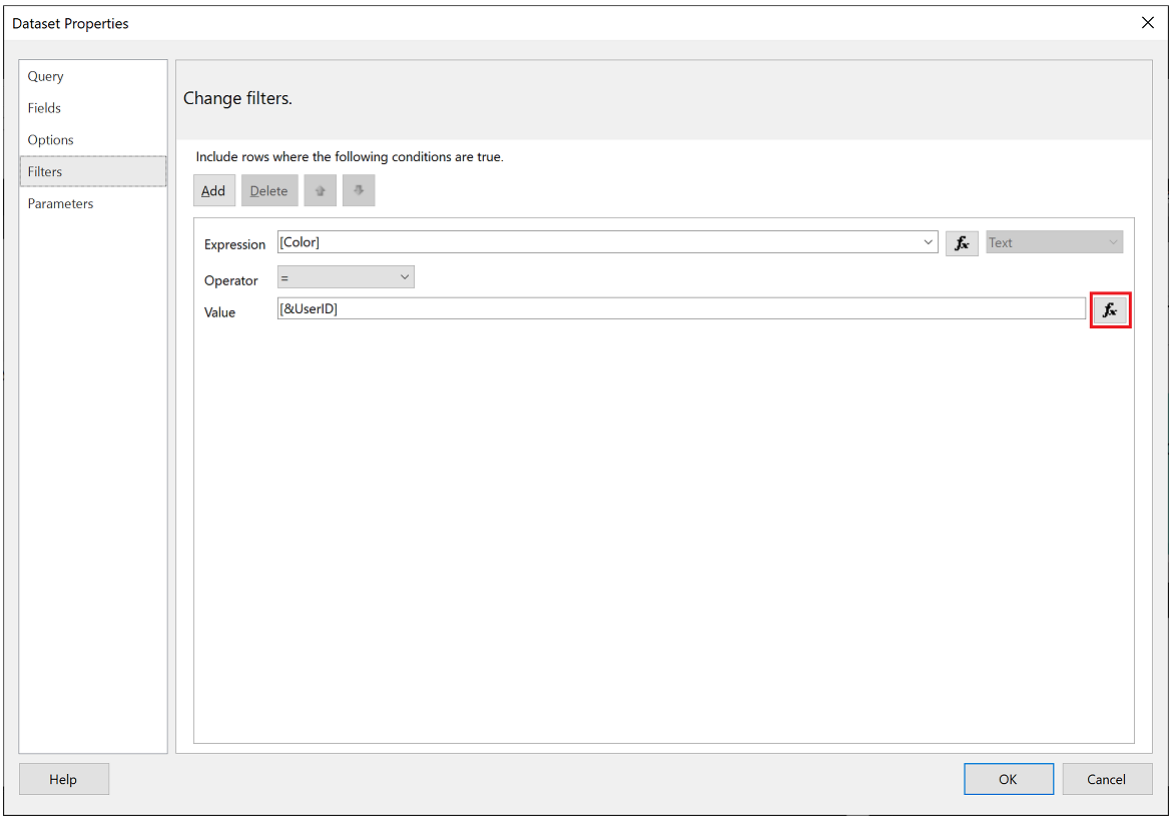
1. In the **Dataset Properties** window, from the left pane, select **Filter**.



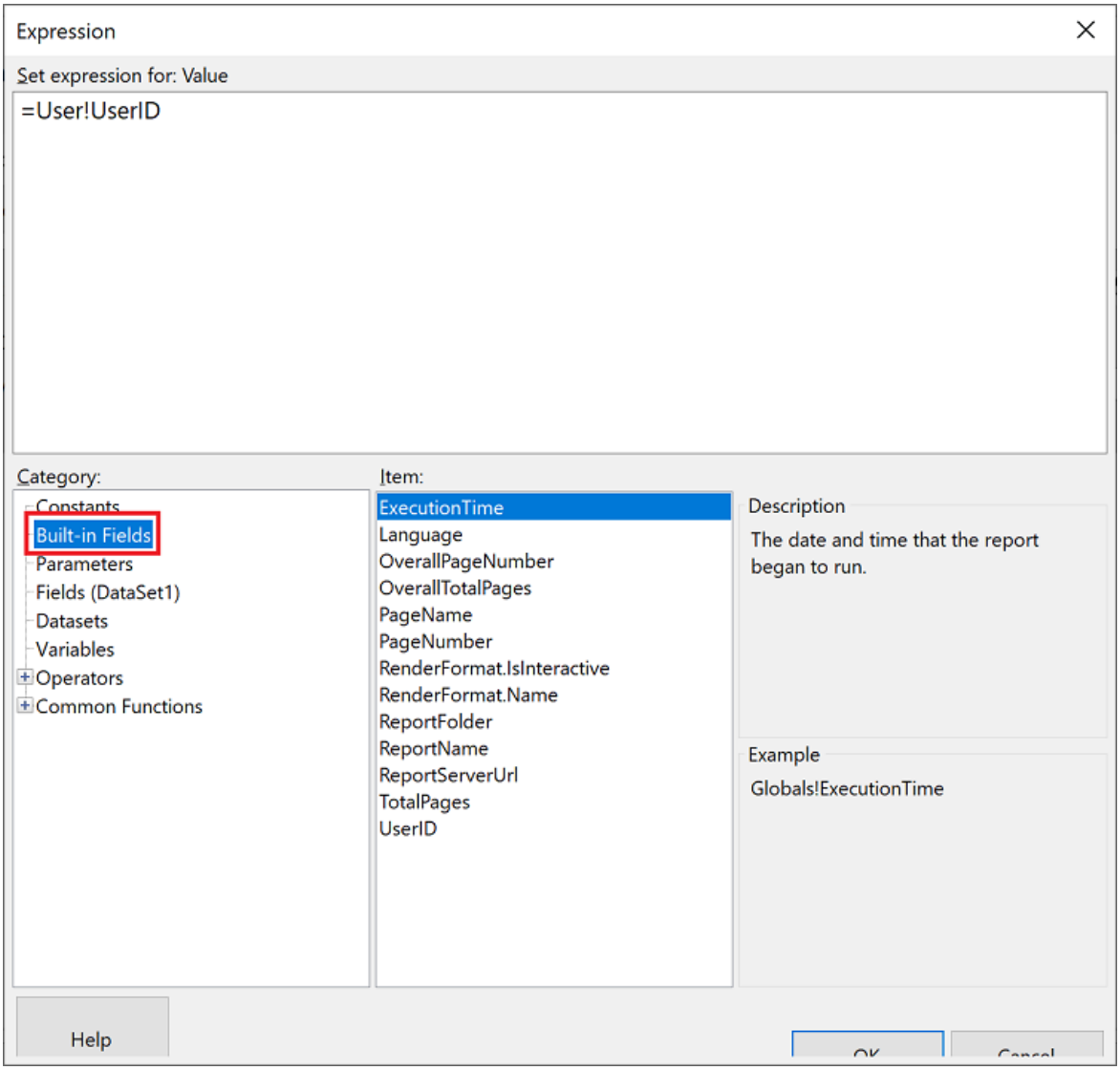
2. From the **Expression** dropdown menu, select the parameter you want to use for filtering the data.



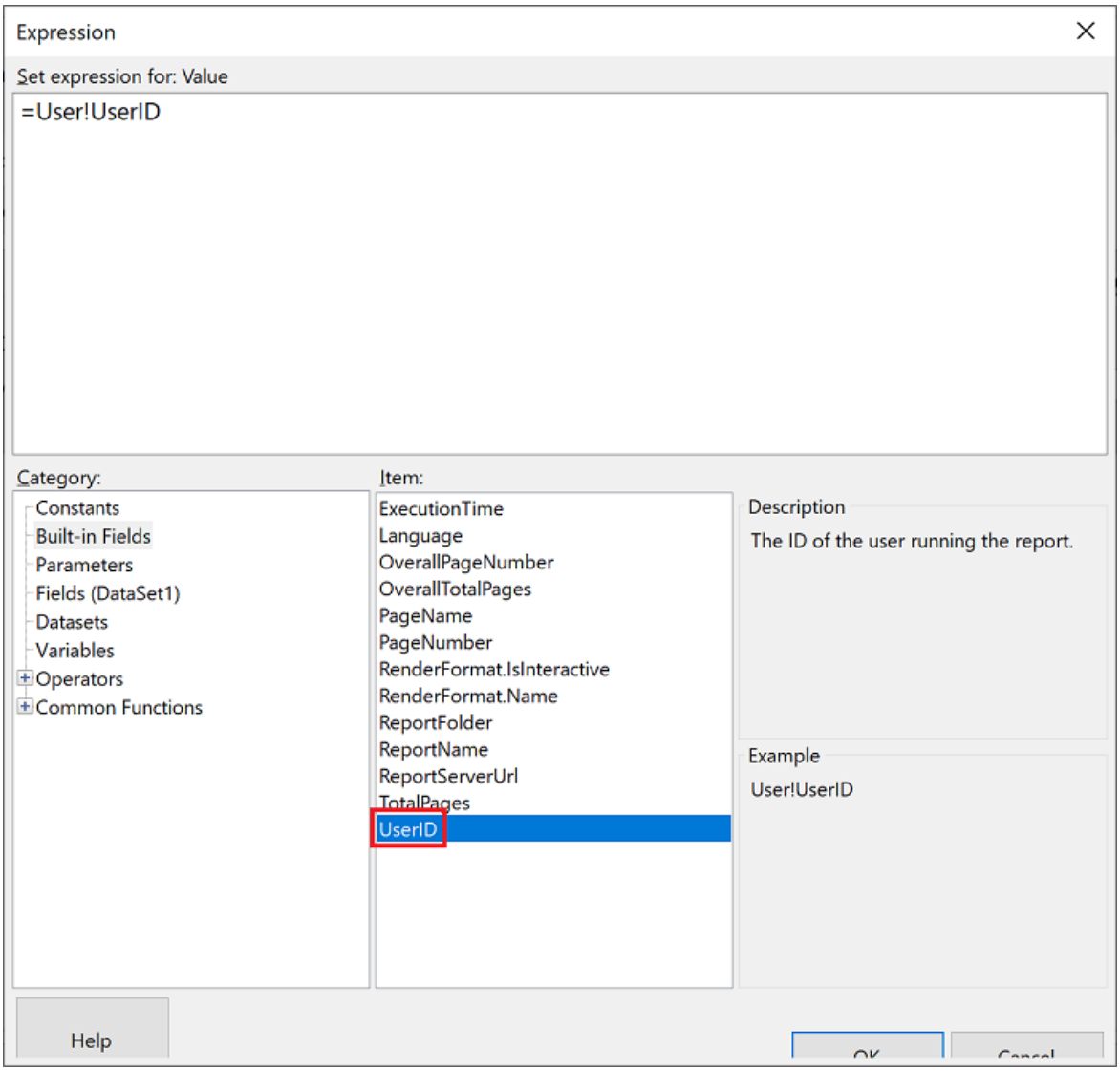
3. Click the **Value** function button.



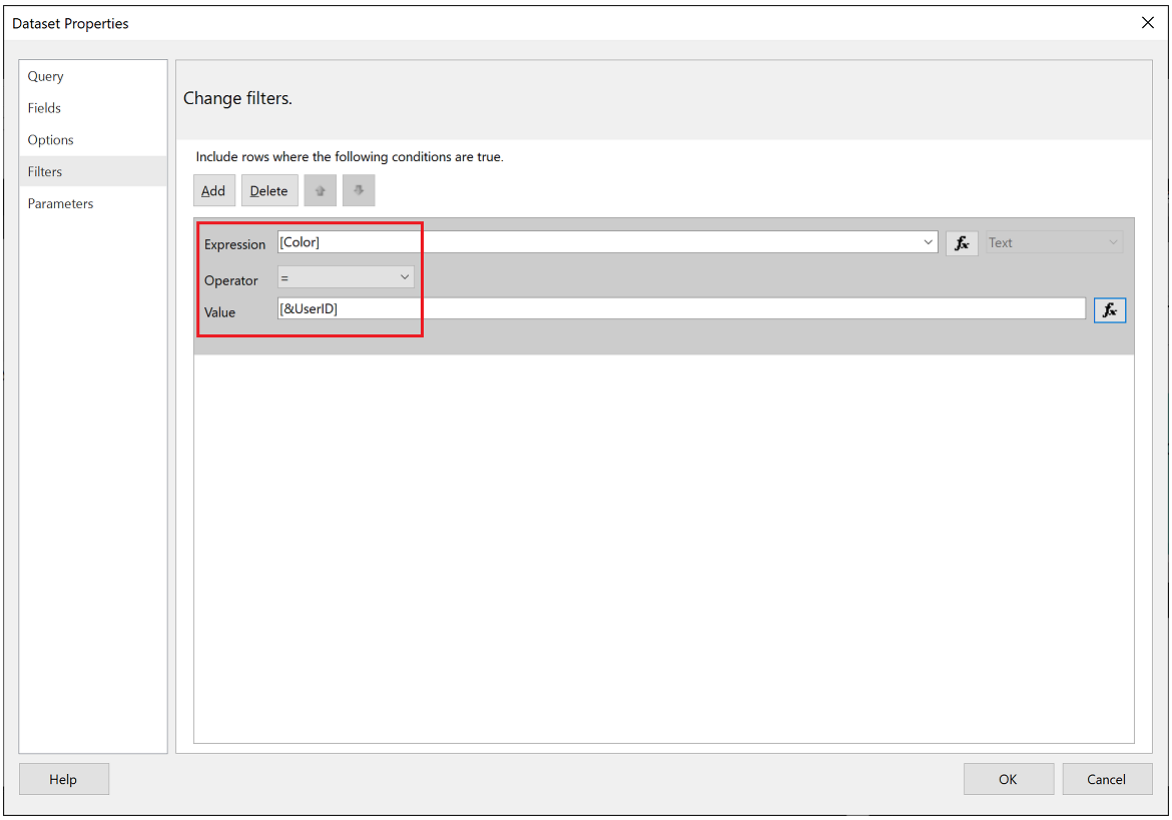
4. In the **Expression** window, from the **Category** list, select **Built-in Fields**.



5. From the **Item** list, select **UserID** and click **OK**.

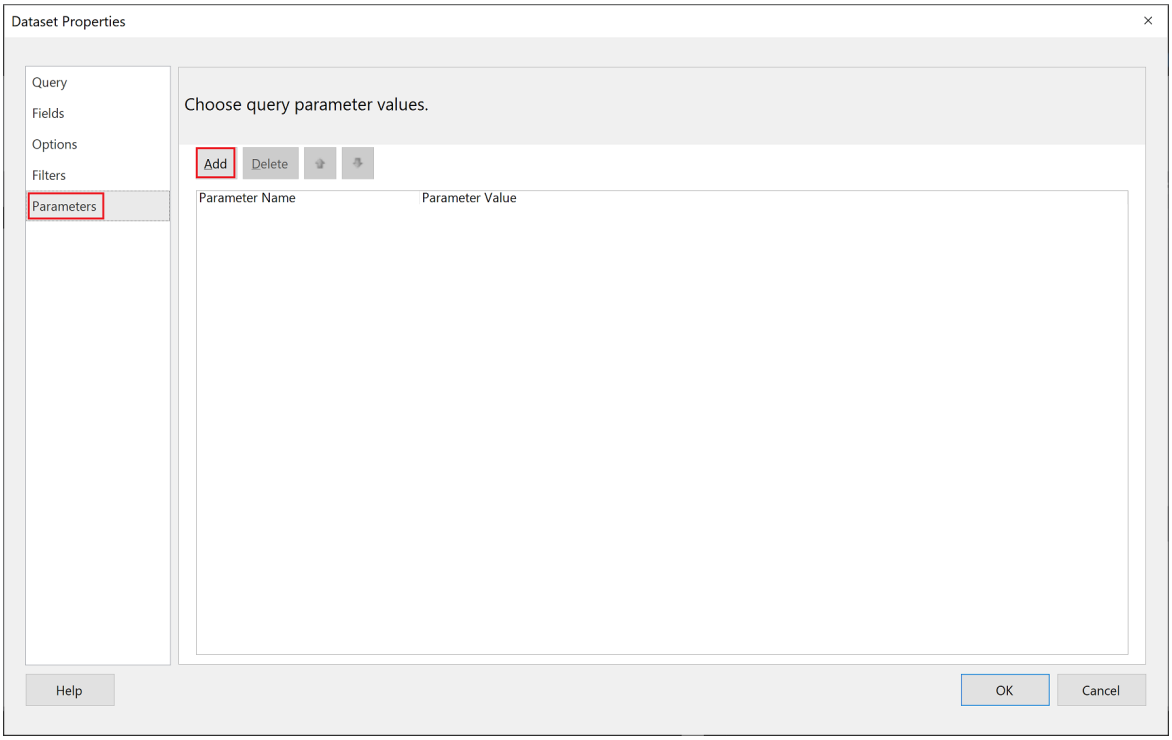


6. In the **Dataset Properties** window, verify that the expression is *your selected parameter = UserID*, and click **OK**.

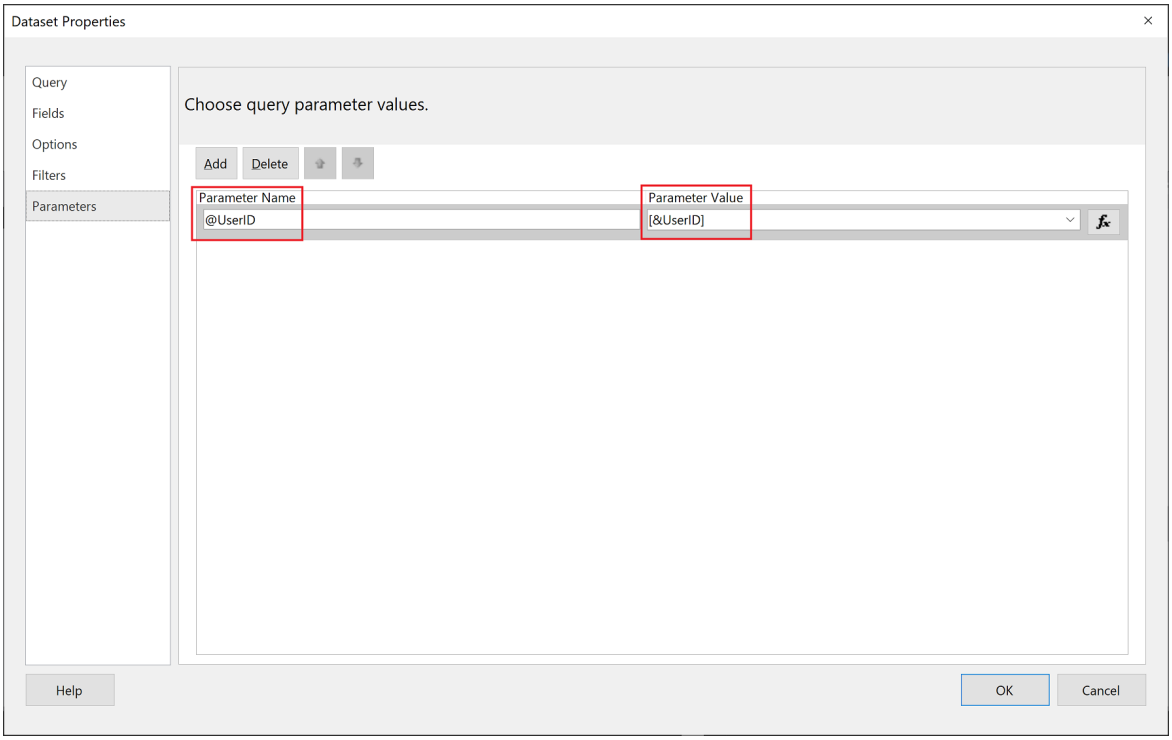


Using a query

1. In the **Dataset Properties** window, from the left navigation pane, select **Parameters** and click **Add**.

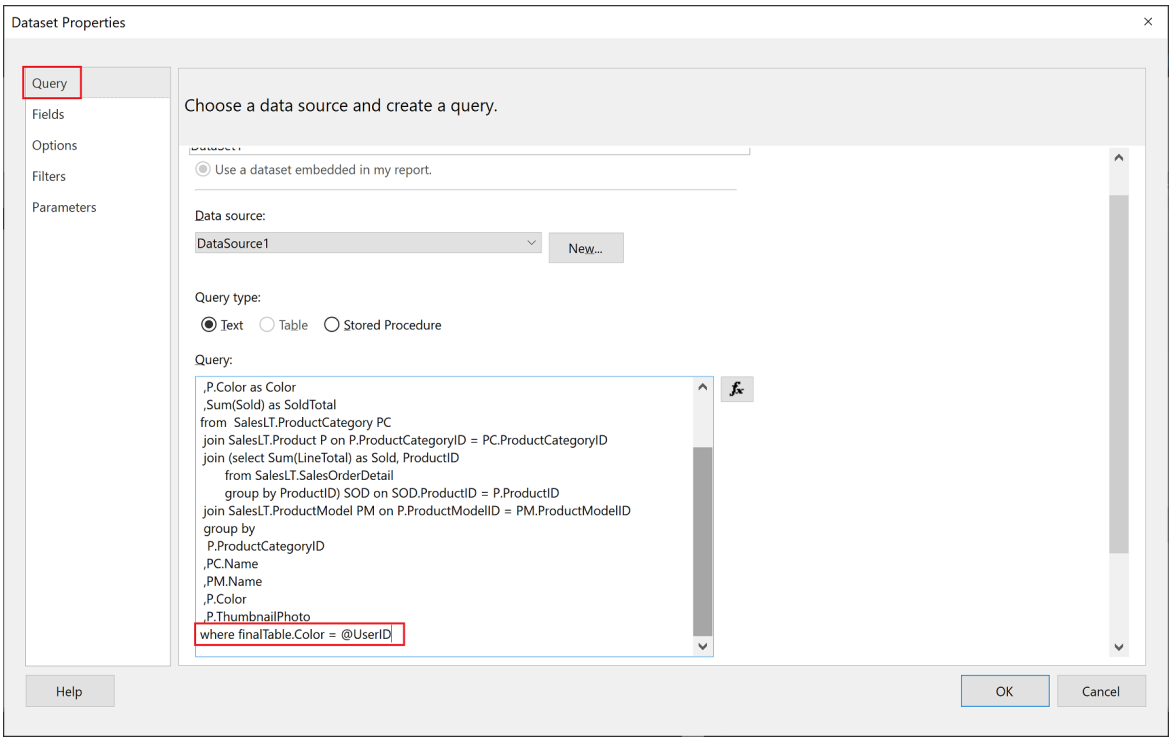


2. In the **Parameter Name** field enter **@UserID**, and in the **Parameter Value** add **[&UserID]**.



3. From the left pane, select **Query**, in the Query add the **UserID** parameter as part of your query, and click **OK**.

> **NOTE**
>
> In the screenshot below the color parameter is used as an example (WHERE FinalTable.Color = @UserID). If needed, you can create a more complex query.



Generate an embed token

When you embed a paginated report for your customers, use the Reports GenerateTokenInGroup API to get the embed token. This token can also be used to filter some data out of the paginated report.

You can only generate a token using a **service principal**. You can't generate a token as a master user. The service principal has to have at least member permissions to the workspace in the Power BI service. (If the service principal is a contributor or viewer it won't be able to generate a token).

To generate a token, assign the `username` field with the information you want to display. For example, in a paginated report that has a color parameter, if you enter *green* in the `username` field, the embed token will restrict the embedded data to just the data that has *green* as its value in the color column.

```
{
 "reports": [
 {
  "id": "8d57615e-cfed-4d60-bd21-7dc05727193c"
 }
 ],
 "identities": [
 {
  "username": "green",
  "reports": [
  "8d57615e-cfed-4d60-bd21-7dc05727193c"
  ]
 }
 ]
}
```

> **NOTE**
>
> If you generate embed token without specifying a *user-id*, the *object-id* of service principal will be used.

Considerations and limitations

- Master-user isn't supported with paginated reports for *embed for your customers*. Master-user **is** supported

for *embed for your organization*.

- The service principal must have workspace permissions of member or above (not viewer or contributor).

Next steps

Generate an embed token

Troubleshoot your embedded application

6/30/2022 • 11 minutes to read • Edit Online

This article discusses some common issues that can come up when embedding content from Power BI.

Troubleshooting tools

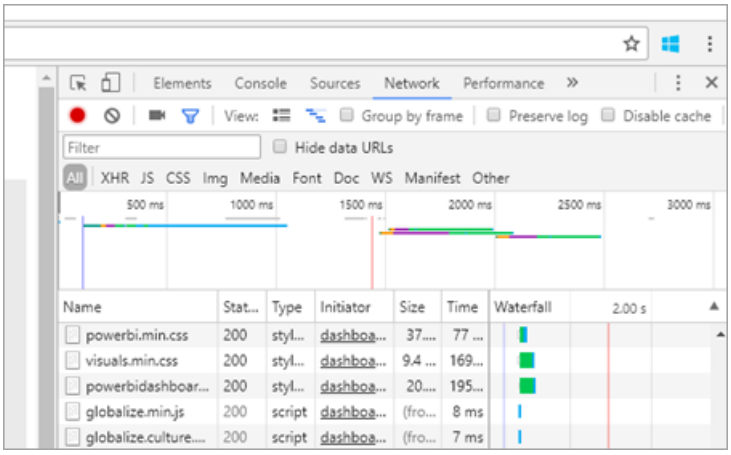
Fiddler Trace

Fiddler is a free tool from Telerik that monitors HTTP traffic. You can see the traffic with the Power BI APIs from the client machine. This tool may show errors and other related information.

8	200	HTTPS	wabi-west-us-redirect.analysis.windows.net	/powerbi/dmm/gateways/discover
9	200	HTTPS	wabi-west-us-redirect.analysis.windows.net	/powerbi/dmm/aggregateDataSource/147516?testConnection=true
11	200	HTTPS	wabi-west-us-redirect.analysis.windows.net	/powerbi/content/packages/147029/refresh/
13	200	HTTPS	wabi-west-us-redirect.analysis.windows.net	/powerbi/metadata/models/147516/?modelOptions=Default
14	200	HTTPS	wabi-west-us-redirect.analysis.windows.net	/powerbi/refresh/subscribe
16	200	HTTPS	wabi-west-us-redirect.analysis.windows.net	/powerbi/metadata/dashboard/95433/tiles
17	200	HTTPS	wabi-west-us-redirect.analysis.windows.net	/powerbi/metadata/models/147516/?modelOptions=Default
18	-	HTTPS	wabi-west-us-redirect.analysis.windows.net	/powerbi/refresh/subscribe

F12 in Browser for front-end debugging

The F12 key launches the developer window within your browser. This tool lets you look at network traffic and see other valuable information.



Extract error details from Power BI response

This code snippet shows how to extract the error details from HTTP exception:

```
public static string GetExceptionText(this HttpOperationException exc)
{
    var errorText = string.Format("Request: {0}\r\nStatus: {1} ({2})\r\nResponse: {3}",
    exc.Request.Content, exc.Response.StatusCode, (int)exc.Response.StatusCode, exc.Response.Content);
    if (exc.Response.Headers.ContainsKey("RequestId"))
    {
        var requestId = exc.Response.Headers["RequestId"].FirstOrDefault();
        errorText += string.Format("\r\nRequestId: {0}", requestId);
    }

    return errorText;
}
```

We recommend logging the Request ID (and error details for troubleshooting). Provide the Request ID when approaching Microsoft support.

App registration

App registration failure

Error messages within the Azure portal or the Power BI app registration page will notify you if you don't have sufficient privileges to register your app. To register an application, you must be an admin in the Azure AD tenant, or application registrations must be enabled for non-admin users.

Power BI Service doesn't appear in the Azure portal when registering a new App

At least one user must be signed up for Power BI. If you don't see **Power BI Service** listed within the API list, no user is signed up for Power BI.

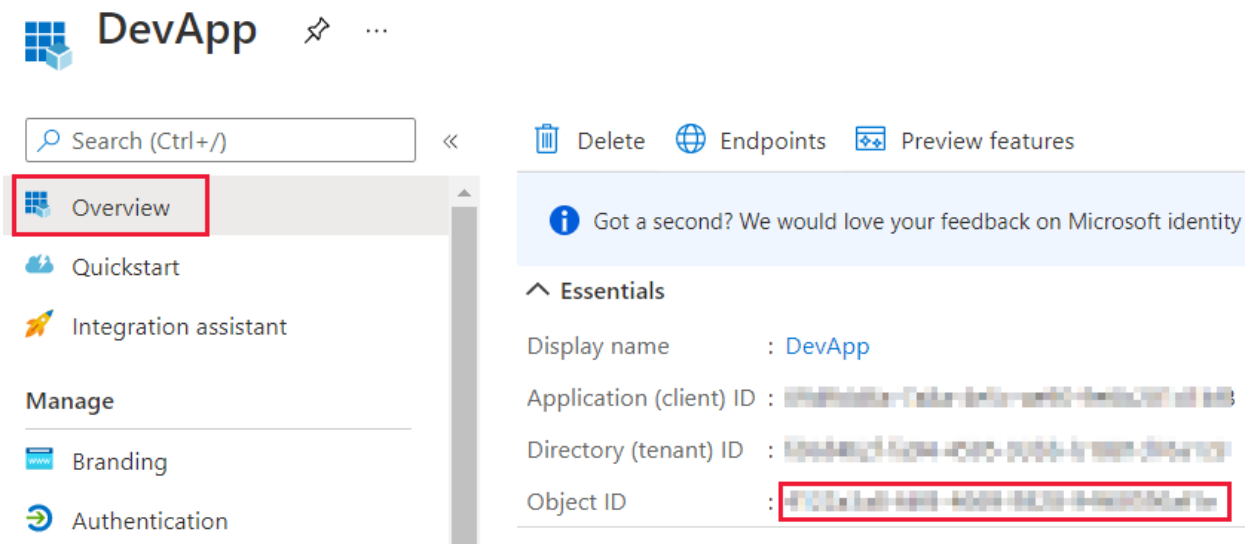
What's the difference between an application object ID and a principal object ID?

When you register an Azure AD app, there are a two parameters called *object ID*. This section explains the purpose of each parameter, and how to obtain it.

Application object ID

The application object ID, also known simply as *object ID*, is the unique ID of your Azure AD application object.

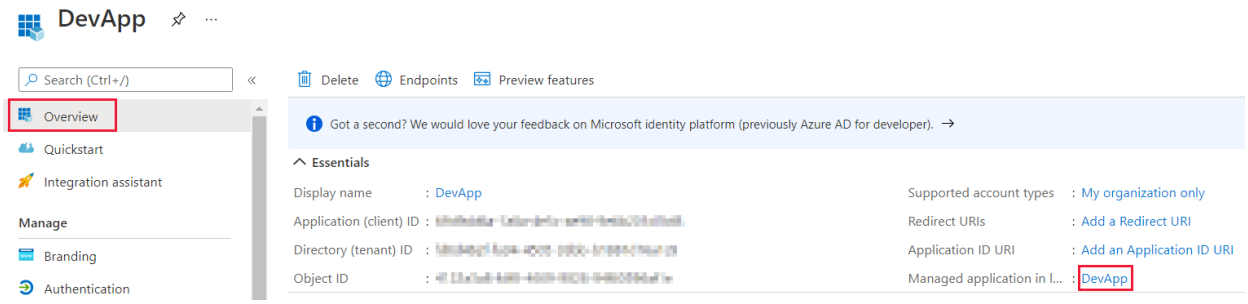
To get the application object ID, navigate to your Azure AD app, and copy it from the *Overview*.



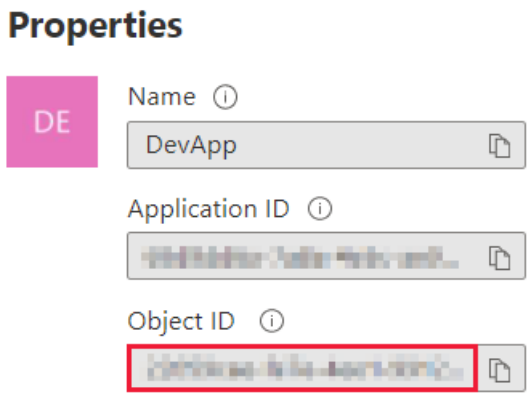
Principal object ID

The principal object ID, also known simply as *object ID*, is the unique ID of the service principal object associated with your Azure AD application.

To get your principal object ID, navigate to your Azure AD app, and from the *Overview* select the app link in **Managed application in local directory**.



From the *Properties* section, copy the **Object ID**.



Authentication

Authentication failed with AADSTS70002 or AADSTS50053

(AADSTS70002: Error validating credentials. AADSTS50053: You've tried to sign in too many times with an incorrect User ID or password)

If you're using Power BI Embedded and Azure AD Direct authentication, you may receive a message like the one above when you try to log in, because direct authentication is no longer in use.

You can turn direct authentication back on using an Azure AD Policy that is scoped to the organization or a service principal.

We recommend you enable this policy only on a per-app basis.

To create this policy, you need to be a **Global Administrator** for the directory where you're creating the policy and assigning it. Here is a sample script for creating the policy and assigning it to the SP for this application:

1. Install the Azure AD Preview PowerShell Module.
2. Run the following PowerShell commands line-by-line (making sure the variable $sp doesn't have more than one application as a result).

```
Connect-AzureAD
```

```
$sp = Get-AzureADServicePrincipal -SearchString "Name_Of_Application"
```

```
$policy = New-AzureADPolicy -Definition @("{`"HomeRealmDiscoveryPolicy`":
{`"AllowCloudPasswordValidation`":true}}") -DisplayName EnableDirectAuth -Type HomeRealmDiscoveryPolicy -
IsOrganizationDefault $false
```

```
Add-AzureADServicePrincipalPolicy -Id $sp.ObjectId -RefObjectId $policy.Id
```

After assigning the policy, wait approximately 15-20 seconds for propagation before testing.

Generate token fails when providing effective identity

`GenerateToken` can fail, with effective identity supplied, for a few different reasons.

- Dataset doesn't support effective identity
- Username wasn't provided
- Role wasn't provided

- DatasetId wasn't provided
- User doesn't have the correct permissions

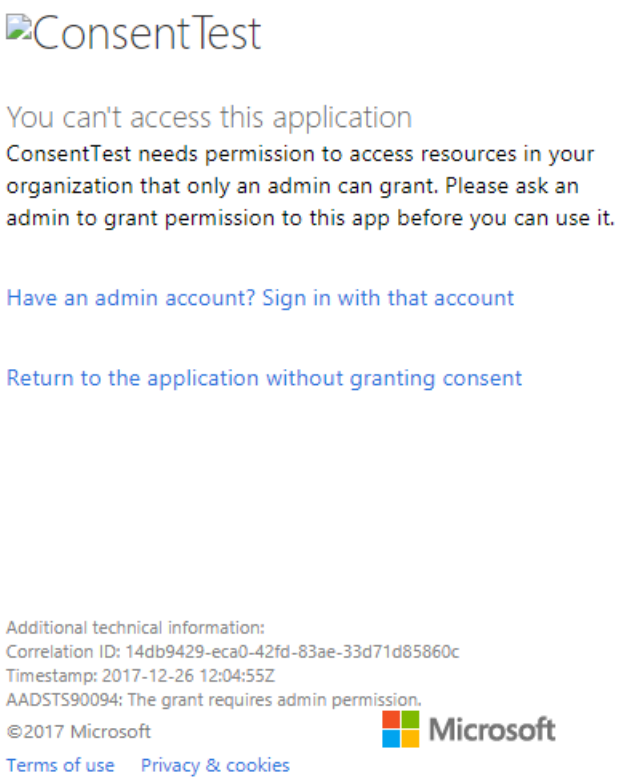
To determine the problem, try the following steps:

- Execute get dataset. Is the property IsEffectiveIdentityRequired true?
- Username is mandatory for any EffectiveIdentity.
- If IsEffectiveIdentityRolesRequired is true, Role is required.
- DatasetId is mandatory for any EffectiveIdentity.
- For Analysis Services, the master user has to be a gateway admin.

AADSTS90094: The grant requires admin permission

Symptoms:
When a non-admin user tries to sign in to an application for the first time while granting consent, then gets one of the following errors:

- ConsentTest needs permission to access resources in your organization that only an admin can grant. Ask an admin to grant permission to this app before you can use it.
- AADSTS90094: The grant requires admin permission.



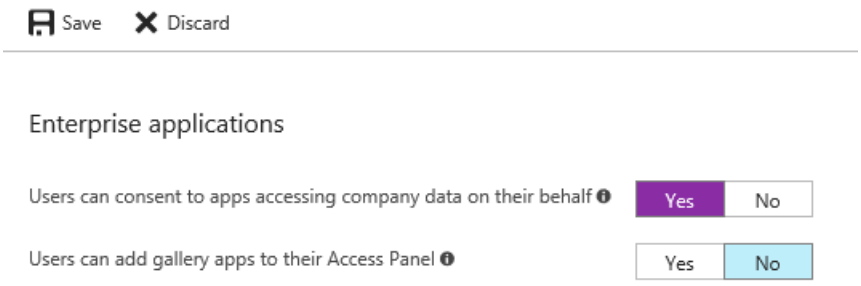
An admin user can sign in and grant consent successfully.

Root cause:
User consent is disabled for the tenant.

Several fixes are possible:

- Enable user consent for the entire tenant (all users, all applications)

1. In the Azure portal, navigate to "Azure Active Directory" => "Users and groups" => "User settings"
2. Enable the "Users can consent to apps accessing company data on their behalf" setting and save the changes



- An admin can grant permissions to the application - either for the entire tenant or a specific user.

CS1061 error

Download Microsoft.IdentityModel.Clients.ActiveDirectory if you experience an "'AuthenticationContext' does not contain a definition for 'AcquireToken' and no accessible 'AcquireToken' accepting a first argument of type 'AuthenticationContext' could be found (are you missing a using directive or an assembly reference?)" error.

Azure AD token for a different tenant (guest user)

When you *embed for your organization*, to allow Azure AD guest users access to your content, you need to specify the tenant ID in the `authorityUri` parameter.

- URL for authenticating in your organization's tenant:

  ```
  https://login.microsoftonline.com/common/v2.0
  ```

- URL for authenticating a guest Azure AD user:

  ```
  https://login.microsoftonline.com/<tenant ID>
  ```

To find your tenant ID, you can use the instructions in Find the Microsoft Azure AD tenant ID and primary domain name.

For more information, see How to: Sign in any Azure Active Directory user using the multi-tenant application pattern.

Data sources

ISV wants to have different credentials for the same data source

A data source can have a single set of credentials for one master user. If you need to use different credentials, create additional master users. Then, assign the different credentials in each of the master users contexts, and embed using the Azure AD token of that user.

Troubleshoot your embedded application with the IError object

Use the **IError object** returned by the *error* event from the **JavaScript SDK** to debug your application and better understand the cause of your errors.

After acquiring the IError object, you should look at the appropriate common errors table that fits the embed type you're using. Compare the **IError properties** with the ones in the table and find the possible reason(s) for the failure.

Typical errors when embedding for Power BI users

MESSAGE	DETAILED MESSAGE	ERROR CODE	POSSIBLE REASON(S)

MESSAGE	DETAILED MESSAGE	ERROR CODE	POSSIBLE REASON(S)
TokenExpired	Access token has expired, resubmit with a new access token	403	Expired token
PowerBIEntityNotFound	Get report failed	404	• Wrong Report ID • Report doesn't exist
Invalid parameters	powerbiToken parameter not specified	N/A	• No access token provided • No Report ID provided
LoadReportFailed	Fail to initialize - Couldn't resolve cluster	403	• Bad access token • Embed type doesn't match token type
PowerBINotAuthorizedException	Get report failed	401	• Wrong group ID • Unauthorized group
TokenExpired	Access token has expired, resubmit with a new access token. Couldn't render a report visual titled: *visual title*	N/A	Query data Expired token
OpenConnectionError	Can't display the visual. Couldn't render a report visual titled: *visual title*	N/A	Capacity paused or deleted while a report related to the capacity was open in a session
ExplorationContainer_FailedToLoadModel_DefaultDetails	Couldn't load the model schema associated with this report. Make sure you have a connection to the server and try again.	N/A	• Capacity paused • Capacity deleted

Typical errors when embedding for non-Power BI users (using an Embed Token)

MESSAGE	DETAILED MESSAGE	ERROR CODE	REASON(S)
TokenExpired	Access token has expired, resubmit with a new access token	403	Expired token
LoadReportFailed	Get report failed	404	• Wrong Report ID • Report doesn't exist
LoadReportFailed	Get report failed	403	Report ID doesn't match token
LoadReportFailed	Get report failed	500	Report provided ID isn't a guid
Invalid parameters	powerbiToken parameter not specified	N/A	• No access token provided • No Report ID provided

MESSAGE	DETAILED MESSAGE	ERROR CODE	REASON(S)
LoadReportFailed	Fail to initialize - Couldn't resolve cluster	403	Wrong token type, Bad Token
PowerBINotAuthorizedException	Get report failed	401	Wrong/unauthorize group ID
TokenExpired	Access token has expired, resubmit with a new access token. Couldn't render a report visual titled: *visual title*	N/A	Query data Expired token
OpenConnectionError	Can't display the visual. Couldn't render a report visual titled: *visual title*	N/A	Capacity paused or deleted while a report related to the capacity was open in a session
ExplorationContainer_FailedToLoadModel_DefaultDetails	Couldn't load the model schema associated with this report. Make sure you have a connection to the server and try again.	N/A	• Capacity paused • Capacity deleted

Get report fails - error 401 - resolve themselves

Sometimes users in the *user owns data* scenario will get a 401 error that resolves itself after they access the Power BI portal. When this happens, add the RefreshUser Permissions call in the app as explained in Update user permissions.

Datasets

Manage which portion of the data your users can see

Any user with read permissions for a dataset can see the entire schema (tables, columns and measures) and all the data. You can't control viewing permissions to raw and aggregated data separately in the same dataset.

To manage which portion of the data your users can view, use one of these methods:

- Row-level filtering using Power BI row-level security (RLS).
- Object level security (OLS).
- Separate the data into different datasets. For example, you can create a dataset that only contains aggregated data and give your users access to that dataset only.

Content rendering

To resolve rendering issues in embedded Power BI items (such as reports and dashboards), review this section.

Verify that the Power BI item loads in Power BI service

To rule out issues with your application *or the embedding APIs*, verify that the item can be viewed in the Power BI service (powerbi.com).

Verify that the Power BI item loads in the Power BI embedded analytics playground

To rule out issues with your application, verify that the Power BI item can be viewed in the Power BI embedded analytics playground.

Verify that your access token didn't expire

For security purposes, access tokens (An Azure AD token or an embed token) have a limited lifetime. You should constantly monitor your access token and refresh it if needed. For more information see Refresh the access token.

Performance

To get the best performing embedded content, we recommend that you follow the Power BI embedded analytics best practices.

Embed setup tool

You can go through the Embedding setup tool to quickly download a sample application. Then you can compare your application to the sample.

Prerequisites

Verify that you have all the proper prerequisites before using the Embedding setup tool. You need a **Power BI Pro** account and a **Microsoft Azure** subscription.

- If you're not signed up for **Power BI Pro**, sign up for a free trial before you begin.
- If you don't have an Azure subscription, create a free account before you begin.
- You need to have your own Azure Active Directory tenant setup.
- You need Visual Studio installed (version 2013 or later).

Common Issues

Some common issues you might encounter while testing with the Embed setup tool are:

Using the Embed for your customers sample application

If you're working with the **Embed for your customers** experience, save and unzip the *PowerBI-Developer-Samples.zip* file. Then open the *PowerBI-Developer-Samples-master\App Owns Data* folder and run the *PowerBIEmbedded_AppOwnsData.sln* file.

- When selecting **Grant permissions** (the Grant permissions step), you get the following error:

```
AADSTS70001: Application with identifier <client ID> wasn't found in the directory <directory ID>
```

The solution is to close the popup, wait a few seconds and try again. You might need to repeat this action a few times. A time interval causes the issue from completing the application registration process to when it's available to external APIs.

- The following error message appears when running the sample app:

```
Password is empty. Please fill password of Power BI username in web.config.
```

This error occurs because the only value that isn't being injected into the sample application is your user password. Open the Web.config file in the solution and fill the pbiPassword field with your user's password.

- If you get the error - AADSTS50079: The user is required to use multi-factor authentication.

You need to use an Azure AD account that doesn't have MFA enabled.

Using the Embed for your organization sample application

If you're working with the **Embed for your organization** experience, save and unzip the *PowerBI-Developer-Samples.zip* file. Then open the *PowerBI-Developer-Samples-master\User Owns Data\integrate-report-web-app* folder and run the *pbi-saas-embed-report.sln* file.

- When you run the **Embed for your organization** sample app, you get the following error:

```
AADSTS50011: The reply URL specified in the request doesn't match the reply URLs configured for the
application: <client ID>
```

This error is because the redirect URL specified for the web-server application is different from the sample's URL. If you want to register the sample application, use `https://localhost:13526/` as the redirect URL.

If you want to edit the registered application, update the Azure AD-registered application, so the application can provide access to the web APIs.

If you want to edit your Power BI user profile or data, learn how to edit your Power BI data.

- If you get the error - AADSTS50079: The user is required to use multi-factor authentication.

You need to use an Azure AD account that doesn't have MFA enabled.

For more information, please see Power BI Embedded FAQ.

More questions? Try the Power BI Community

For further assistance, contact support or create a support ticket via the Azure portal and provide the error messages you encounter.

Next steps

Power BI Embedded Frequently Asked Questions

More questions? Try the Power BI Community

Troubleshoot REST APIs

6/30/2022 • 2 minutes to read • Edit Online

API call returns 401

A fiddler capture may be required to investigate further. The required permission scope may be missing for the registered application within Azure AD. Verify the required scope is present within the app registration for Azure AD within the Azure portal.

API call returns 403

A 403 error can occur for any of the following reasons. A fiddler capture may be required to investigate further.

- The user has exceeded the amount of embed token that can be generated on a shared capacity. Purchase Azure capacities to generate embed tokens and assign the workspace to that capacity. See Create Power BI Embedded capacity in the Azure portal.
- The Azure AD auth token expired.
- The authenticated user isn't a member of the group (workspace).
- The authenticated user isn't an admin of the group (workspace).
- The authenticated user doesn't have permissions. Permissions can be updated using refreshUserPermissions API
- The authorization header may not be listed correctly. Make sure there are no typos.

The backend of the application may need to refresh the auth token before calling GenerateToken. For more information, see Refresh the access token.

```
GET https://wabi-us-north-central-redirect.analysis.windows.net/metadata/cluster HTTP/1.1
Host: wabi-us-north-central-redirect.analysis.windows.net
...
Authorization: Bearer eyJ0eXAiOi...
...

HTTP/1.1 403 Forbidden
...

{"error":{"code":"TokenExpired","message":"Access token has expired, resubmit with a new access token"}}
```

Fix timeout exceptions when using import and export APIs

When you send a Power BI REST API request, it might arrive at a cluster that doesn't contain your tenant's data. In that case, redirecting the request may fail due to a timeout.

To fix the timeout exception, resend the request with the `preferClientRouting` parameter set to `true`. If your request arrives at the wrong cluster, the Power BI service returns a *307 Temporary Redirect* HTTP response. In such cases, you need to redirect your request to the new address specified in the response *HTTPS Location header*.

Next steps

Power BI Embedded Frequently Asked Questions

More questions? Try the Power BI Community

Push datasets limitations

6/30/2022 • 2 minutes to read • Edit Online

Push datasets are limited in their functionality. They're designed *only* for a near real-time streaming scenario to be consumed by a streaming tile in a dashboard, and not by a Power BI report.
This article lists limitations of the Power BI REST APIs push datasets.

Limitations

Review the following list of limitations before using the *push datasets* APIs.

- 75 max columns
- 75 max tables
- 10,000 max rows per single POST rows request
- 1,000,000 rows added per hour per dataset
- 5 max pending POST rows requests per dataset
- 120 POST rows requests per minute per dataset
- If table has 250,000 or more rows, 120 POST rows requests per hour per dataset
- 200,000 max rows stored per table in FIFO dataset
- 5,000,000 max rows stored per table in 'none retention policy' dataset
- 4,000 characters per value for string column in POST rows operation

Next steps

- Power BI REST APIs.
- Push datasets.

Datasets permissions

6/30/2022 • 3 minutes to read • Edit Online

This article describes Power BI permissions in general, and dataset permissions in the context of the Power BI REST APIs.

Power BI permissions

Power BI has two sets of permissions:

- Workspace permissions
- Item permissions

Workspace permissions

Workspace permissions, also known as folder permissions or roles, are the highest level of permissions in Power BI. These permissions override permissions that are given to a specific item in the workspace folder.

The table below lists the four types of folder roles. It shows each role's level, and the code string returned by the Power BI REST APIs. Admin is the highest workspace permission level, and viewer is the lowest. Every permission level includes the capabilities of the permissions below it. You can review the capabilities of each permission in Workspace roles.

FOLDER ROLE	LEVEL	DERIVED PERMISSIONS FOR DATASETS CREATED IN THE WORKSPACE
Admin	4	`ReadWriteReshareExplore`
Member	3	`ReadWriteReshareExplore`
Contributor	2	`ReadWriteExplore`
Viewer	1	`Read`

> **NOTE**
> The *write* permission is applied to Power BI datasets created by *admin*, *member* and *contributor* users in a workspace they own. The write permission can be granted or deleted using workspace permissions only. It cannot directly be granted to, or deleted from, a Power BI item.

Get and add workspace permissions with APIs

To get and add workspace permissions programmatically, use these APIs:

- Groups - Add Group User - A POST API for adding workspace permissions
- Groups - Update Group User - A PUT API for changing workspace permissions
- Groups - Get Group Users - A GET API for getting workspace permissions

Item permissions

Power BI items, such as reports, datasets, and dashboards have their own permissions. Item permissions can't override workspace permissions, and can only be granted by someone who has at least the same level of

permission.

Dataset permissions and REST APIs

Dataset permissions are part of the item permissions. The table below lists the Power BI dataset permissions and their representation in the Power BI REST APIs.

> **TIP**
> Although the API permissions are identical to the Power BI service permissions, `build` permissions are referred to as *explore* permissions in the APIs.

PERMISSION	READ	EXPLORE	RESHARE
Description	Allows the user to read the content of the dataset	Equivalent to build permissions	Allows the user to share the content of the dataset with other users who will get read, reshare, or explore permissions for it
ReadReshareExplore	✔	✔	✔
ReadReshare	✔	✘	✔
ReadExplore	✔	✔	✘
Read	✔	✘	✘

> **NOTE**
> To allow a user to perform write operations on a dataset, first change the workspace permissions.

Build permissions and REST APIs

In the Power BI REST APIs, the `build` permission is returned as *explore*. For example, a string with the *read*, *reshare* and *build* permissions, will look like this: `ReadReshareExplore`.

When you give a user `build` permission, they can build new content on your dataset. Examples of content they can build are reports, dashboards, pinned tiles from Q&A, paginated reports, and Insights Discovery.

Users also need `build` permissions to work with data outside Power BI:

- To export the underlying data.
- To build new content on the dataset such as with Analyze in Excel.
- To access the data via the XMLA endpoint.

Row-level security

For a dataset that uses row-level security (RLS), any permissions *higher* than `build` will enable the user to view all the data in the dataset. `Build`, and permissions lower than `build`, will only give the dataset user access to the data they're allowed to see as configured in your RLS settings.

Get and update dataset permissions with APIs

- POST APIs let you add new permissions to a dataset. You can use these APIs to add permissions for users

but not to remove permissions. For example, you can add the `Reshare` permission to a user that has the `Read` permission. However, you can't remove the `Reshare` permission from a user that has both `Read` and `Reshare` permissions, by attempting to add the `Read` permission.

  - Datasets - Post Dataset User
  - Datasets - Post Dataset User In Group

- PUT APIs update the user's permissions to a given dataset. The PUT API can't be used for changing write permissions or any folder level inherited permissions. This API also supports removing all permissions for a dataset for a given target.

  - Datasets - Put Dataset User
  - Datasets - Put Dataset User In Group

- GET APIs return a list of principals that have access to the specified dataset.

  - Datasets - Get Dataset Users
  - Datasets - Get Dataset Users In Group

Considerations and limitations

Each of the above APIs comes with certain limitations regarding who can use them and how. To see the limitations of each API, select the link for that API.

Next steps

Power BI REST APIs.

Push datasets.

Power BI Developer in a Day course

6/30/2022 • 4 minutes to read • Edit Online

The **Power BI Developer in a Day** video-based course empowers you as an app developer with the technical knowledge required to embed Power BI content. It comprises 2 hours 20 minutes of viewable content—available on demand, and is free of charge. There's also a self-study kit that you can download and use to complete a series of six hands-on labs.

The course was designed specifically for experienced app developers. So, it's an advantage if you have development experience with:

- ASP.NET
- Visual C#
- HTML
- JavaScript

Familiarity with Power BI will be beneficial, but not essential. We'll introduce you to the core concepts.

After you complete the course, you'll know how to:

- Acquire access using Azure AD apps and tokens
- Work with the Power BI REST API
- Embed Power BI content in your apps
- Integrate Power BI content in your apps using the Power BI JavaScript API
- Enforce row-level security (RLS) to ensure app users see the right data
- Choose the right license to suit your requirements

Watch the welcome and introduction video to start the course.

> **NOTE**
> This video might use earlier versions of Power BI Desktop or the Power BI service.

Course outline

The course of 21 videos is organized into eight modules. We recommend you watch the videos in the recorded sequence, starting with video 01 and ending with video 21.

- **Introduction**
  - Video 01: Welcome and Course Introduction
  - Video 02: Self-study Kit
- **Module 01: Power BI Overview**
  - Video 03: Power BI Overview - Part 1
  - Video 04: Power BI Overview - Part 2
  - Video 05: Power BI Overview - Part 3
  - Video 06: Power BI Overview - Part 4
- **Module 02: Power BI Embedded Analytics**

    - Video 07: Power BI Embedded Analytics - Part 1
    - Video 08: Power BI Embedded Analytics - Part 2
- **Module 03: Configure Permissions**
    - Video 09: Configure Permissions - Part 1
    - Video 10: Configure Permissions - Part 2
- **Module 04: Embed Power BI Content**
    - Video 11: Embed Power BI Content - Part 1
    - Video 12: Embed Power BI Content - Part 2
- **Module 05: Integrate Content with the Power BI Client APIs**
    - Video 13: Integrate Content with the Power BI Client APIs - Part 1
    - Video 14: Integrate Content with the Power BI Client APIs - Part 2
- **Module 06: Configure Data Permissions**
    - Video 15: Configure Data Permissions - Part 1
    - Video 16: Configure Data Permissions - Part 2
    - Video 17: Configure Data Permissions - Part 3
- **Module 07: Automate Solution Management**
    - Video 18: Automate Solution Management - Part 1
    - Video 19: Automate Solution Management - Part 2
- **Module 08: Power BI Embedded Analytics Licensing**
    - Video 20: Power BI Embedded Analytics Licensing
- **Bonus Content**
    - Video 21: Power BI Embedded Analytics Playground

Self-study kit

You can download and set up a self-study kit, which consists of the presentation content and six hands-on labs. For more information, watch the Self-study Kit video.

To complete the labs, you'll need a Windows PC (Windows 7, or later) and the following software installed:

- The latest version of Power BI Desktop.
- Visual Studio 2015, or later. We recommend Visual Studio 2019. You can use **Community** edition, which is free and suited for learning scenarios. It must have the **ASP.NET and web development** workload installed.
- A web browser supported by Power BI. We recommend Microsoft Edge.

Follow these steps to get set up:

1. Use this link to download the self-study kit (.zip) locally to your PC.
2. Open the file properties, and then check "unblock" (Windows may flag the file as potentially untrusted).
3. Extract the file contents to a folder in your file system. We recommend you create a folder that will be easy to find, perhaps naming it **Training**. The lab documents will refer to this location as **<CourseFolder>**.

Once extracted, you'll have the **PowerBIDevIAD** folder, and within it you'll find the following folders:

- **Lab00A** (and all other lab folders). The lab folders contain the lab document and lab resources, which may include assets and solution files.
- **MySolution**: This folder stores your solution files. The lab instructions will direct you when to use it.
- **Presentation**: This folder contains the course presentation file, which is available as a PDF document.

Get started with the kit

We recommend you watch the online course first. You can refer back to the presentation theory by opening the

<CourseFolder>\PowerBIDevIAD\Presentation\PowerBIDevIAD_Presentation.pdf file. The presentation includes five lab slides, which indicate when it's time to put the theory to practice. It also includes many resource links to help you find related content.

When you're ready to commence the first lab, open the **<CourseFolder>\PowerBIDevIAD\Lab00A\PowerBIDevIAD_Lab00A.pdf** file. This document guides you to sign in to the Power BI service and prepare a Power BI report.

> **NOTE**
>
> You're responsible for having your own Power BI account. If you don't already have one, see Sign up for Power BI as an individual.
>
> Your account must have a Power BI Pro license, or you can still accept a Power BI Pro Trial license—an offer that can only be accepted once. Also, your account must not have depleted the reserve of free embed tokens, available with the Power BI Pro license.
>
> Consider creating a Power BI account for exclusive use in the labs. You can create a free account with a public domain like https://outlook.live.com, and then using it to sign in to Power BI and accept the Power BI Pro Trial license.

Instructor kit

Use this link to download the instructor kit (.zip) locally to your PC. You'll find classroom setup notes in slide one of the PowerPoint slide deck.

Next steps

For more information related to this article, check out the following resources:

- Questions? Try asking the Power BI Community
- Suggestions? Contribute ideas to improve Power BI

Monitoring Power BI Embedded data reference

6/30/2022 • 4 minutes to read • Edit Online

See Monitor Power BI Embedded for details on collecting and analyzing monitoring data for Power BI Embedded.

> **TIP**
> You can also use the Premium Gen2 Monitoring App to monitor your Embedded Gen 2 capacity.

Metrics

This section lists all the automatically collected platform metrics collected for Power BI Embedded.

METRIC TYPE	RESOURCE PROVIDER / TYPE NAMESPACE AND LINK TO INDIVIDUAL METRICS
Capacities	Microsoft.PowerBIDedicated/capacities

Capacities

Resource Provider and Type: Microsoft.PowerBIDedicated/capacities

NAME	METRIC	UNIT	DESCRIPTION
CPU (Gen2)	cpu_metric	Percent	CPU utilization. Supported only for Power BI Embedded Generation 2 resources.
CPU Per Workload (Gen2)	cpu_workload_metric	Percent	CPU utilization per workload. Supported only for Power BI Embedded Generation 2 resources.
Overload (Gen2)	overload_metric	0/1	Resource overload, 1 if resource is overloaded, otherwise 0. Supported only for Power BI Embedded Generation 2 resources.
Memory (Gen1)	memory_metric	Bytes	Memory. Range 0-3 GB for A1, 0-5 GB for A2, 0-10 GB for A3, 0-25 GB for A4, 0-50 GB for A5 and 0-100 GB for A6. Supported only for Power BI Embedded Generation 1 resources.

NAME	METRIC	UNIT	DESCRIPTION
Memory Thrashing (Datasets) (Gen1)	memory_thrashing_metric	Percent	Average memory thrashing. Supported only for Power BI Embedded Generation 1 resources.
QPU High Utilization (Gen1)	qpu_high_utilization_metric	Count	QPU High Utilization In Last Minute, 1 For High QPU Utilization, Otherwise 0. Supported only for Power BI Embedded Generation 1 resources.
Query Duration (Datasets) (Gen1)	QueryDuration	Milliseconds	DAX Query duration in last interval. Supported only for Power BI Embedded Generation 1 resources.
Query Pool Job Queue Length (Datasets) (Gen1)	QueryPoolJobQueueLength	Count	Number of jobs in the queue of the query thread pool. Supported only for Power BI Embedded Generation 1 resources.

Metric dimensions

For more information on what metric dimensions are, see Multi-dimensional metrics.

Power BI Embedded does not have any metrics that contain dimensions.

Resource logs

This section lists the types of resource logs you can collect for Power BI Embedded.

For reference, see a list of all resource logs category types supported in Azure Monitor.

This section lists all the resource log category types collected for Power BI Embedded.

RESOURCE LOG TYPE	RESOURCE PROVIDER / TYPE NAMESPACE AND LINK TO INDIVIDUAL METRICS
Capacities	Microsoft.PowerBIDedicated/capacities

Azure Monitor Logs tables

This section refers to all of the Azure Monitor Logs Kusto tables relevant to Power BI Embedded and available for query by Log Analytics.

RESOURCE TYPE	NOTES
Power BI Embedded	See a list of tables below

Power BI Embedded

TABLE	DESCRIPTION
AzureActivity	Entries from the Azure Activity log that provides insight into any subscription-level or management group level events that have occurred in Azure.
AzureDiagnostics	Stores resource logs for Azure services that use Azure Diagnostics mode. Resource logs describe the internal operation of Azure resources.
AzureMetrics	Metric data emitted by Azure services that measure their health and performance.

For a reference of all Azure Monitor Logs / Log Analytics tables, see the Azure Monitor Log Table Reference.

Activity log

You can select **Engine** and/or the **AllMetrics** categories.

Engine

The engine category instructs the resource to log the events listed below. For each event, there are properties.

EVENT NAME	EVENT DESCRIPTION
Audit Login	Records all new connection to the engine events since the trace started.
Session Initialize	Records all session initialization events since the trace started.
Vertipaq Query Begin	Records all VertiPaq SE query begin events since the trace started.
Query Begin	Records all query begin events since the trace started.
Query End	Records all query end events since the trace started.
Vertipaq Query End	Records all VertiPaq SE query end events since the trace started.
Audit Logout	Records all disconnect from engine events since the trace started.
Error	Records all engine error events since the trace started.

Event example

The table below shows an event example.

PROPERTY NAME	VERTIPAQ QUERY END EXAMPLE	PROPERTY DESCRIPTION
EventClass	XM_SEQUERY_END	Event Class is used to categorize events.

PROPERTY NAME	VERTIPAQ QUERY END EXAMPLE	PROPERTY DESCRIPTION
EventSubclass	0	Event Subclass provides additional information about each event class. (for example, 0: VertiPaq Scan)
RootActivityId	ff217fd2-611d-43c0-9c12-19e202a94f70	Root activity ID.
CurrentTime	2018-04-06T18:30:11.9137358Z	Time at which the event started when available.
StartTime	2018-04-06T18:30:11.9137358Z	Time at which the event started when available.
JobID	0	Job ID for progress.
ObjectID	464	Object ID
ObjectType	802012	ObjectType
EndTime	2018-04-06T18:30:11.9137358Z	Time at which the event ended.
Duration	0	Amount of time (in milliseconds) taken by the event.
SessionType	User	Session type (what entity caused the operation).
ProgressTotal	0	Progress total.
IntegerData	0	Integer data.
Severity	0	Severity level of an exception.
Success	1	1 = success. 0 = failure (for example, a 1 means success of a permissions check and a 0 means a failure of that check).
Error	0	Error number of a given event.
ConnectionID	3	Unique connection ID.
DatasetID	5eaa550e-06ac-4adf-aba9-dbf0e8fd1527	Id of the dataset in which the statement of the user is running.
SessionID	3D063F66-A111-48EE-B960-141DEBDA8951	Session GUID.
SPID	180	Server process ID. This uniquely identifies a user session. This directly corresponds to the session GUID used by XML/A.

PROPERTY NAME	VERTIPAQ QUERY END EXAMPLE	PROPERTY DESCRIPTION
ClientProcessID	null	The process ID of the client application.
ApplicationName	null	Name of the client application that created the connection to the server.
CapacityName	pbi641fb41260f84aa2b778a85891ae2d97	The name of the Power BI Embedded capacity resource.

AllMetrics

Checking the **AllMetrics** option logs the data of all the metrics that you can use with a Power BI Embedded resource.

The following table lists the operations related to Power BI Embedded that may be created in the Activity log.

Schemas

Power BI Embedded uses the **Power BI Dedicated** schema.

Example script for scaling a capacity

To scale a capacity resource, you can use the ScaleUp-Automation-RunBook.ps1 PowerShell runbook script.

The script uses Power BI and ARM REST APIs, and can be called in Azure automation, and triggered by Azure alert.

You can either copy the script, or download it as part of the PowerBI-Developer-Samples repository, by selecting the green *code* button, and downloading the ZIP.

Next steps

Monitor Azure Power BI Embedded

Azure resource diagnostic logging

Set-AzureRmDiagnosticSetting

Glossary for Power BI developers

6/30/2022 • 21 minutes to read • Edit Online

Power BI may introduce terminology that is unfamiliar or confusing. This glossary is a great place to look up terminology, you might even want to keep it bookmarked. Other great resources for learning about the building blocks that make up Power BI are Power BI service basic Concepts and Basic concepts for designers. These articles give a high-level overview of the Power BI *pieces* and how they're connected.

This glossary is a community effort. Don't see a word here? Ask us to add it (you can use the documentation feedback button at the bottom of this article).

A

Account

Use your work or school account to sign in to Power BI. Administrators manage work or school accounts in Azure Active Directory. Your level of access is determined by the Power BI license associated with that account and the capacity type where content is stored. See **license** and **Premium**, below.

Admin portal

The location where Power BI admins manage users, features, and settings for Power BI in their organization. (Note: Microsoft 365, Azure, and PowerApps use admin center.)

Aggregates

When the values of multiple rows are grouped together as input on criteria to form a single value of more significant meaning or measurement. Only implicit measures (see definition below) can be aggregated.

Aggregation

The reduction of rows in underlying data sources to fit in a model. The result is an aggregate.

Alert, alerts

A feature that notifies users of changes in the data based on limits they set. Alerts can be set on tiles pinned from report visuals. Users receive alerts on the service and on their mobile app.

Annotate

To write lines, text, or stamps on a snapshot copy of a tile, report, or visual on the Power BI mobile app for iOS and Android devices.

App, apps

A bundle of dashboards, reports, and datasets. It also refers to the mobile apps for consuming content, such as Power BI app for iOS.

AppSource

Centralized online repository where you can browse and discover dashboards, reports, datasets, and apps to download.

ArcGIS for Power BI

ArcGIS is a mapping and analytics platform created by the company ESRI. The name of the visual included in the Power BI visuals library is called ArcGIS for Power BI.

Artifacts

See item

Auto Insights

Auto insights are now called Quick Insights.

Azure AD, Azure Active Directory

The identity service in Microsoft Azure that provides identity management and access control capabilities through a REST-based API.

Azure Key Vault

The pay-as-you-go security solution that uses a single location (the vault) in Azure to store, secure and manage cryptographic keys and secrets that can then be invoked by URI and used by applications for small, high-value security data, such as authentication keys, storage account keys, data encryption keys, .pfx files, and passwords.

B

BI, business intelligence

Bookmark

A view of data captured in the Bookmarks pane of a report in Power BI Desktop or service. In Desktop, the bookmarks are saved in the .pbix report file for sharing on the Power BI service

Breadcrumbs

The navigation at the top left to quickly navigate between reports and dashboards.

C

Capacity

[Power BI Premium] Data models running on hardware fully managed by Microsoft in Microsoft cloud data centers to help ensure consistent performance at scale. BI solutions are delivered to the entire organization regardless of Power BI license.

Card (visual type)

A Power BI visual type.

Card (Power BI Home)

Power BI Home displays rectangular and square pictures that represent dashboards, reports, apps, and more. These pictures are referred to as *cards*.

Certified custom visual

A Power BI custom visual that has met requirements and passed strict security testing.

Connect live

A method of connecting to SQL Server Analysis Services data models. Also called a live connection.

Connector

Power BI Desktop includes an ever-growing collection of data connectors that are built to connect to a specific data source. Examples include: GitHub, MailChimp, Power BI dataflows, Google Analytics, Python, SQL Server, Zendesk, and more than 100 other data sources.

Container

The areas on the navigation pane are *containers*. In the nav pane you'll find containers for: Favorites, Recent, Apps, Shared with me, and Home.

Content

Content for the Power BI service is generally dashboards, reports, and apps. It can also include workbooks and datasets.

Content list

The content index for an app.

Content pack

Deprecated. A collection of pre-packaged datasets, reports, and dashboards. Content packs were replaced with **apps**. Learn about the new workspace experience.

Content view

The view in Windows Explorer that displays the most relevant content for each item based on its file name extension or Kind association.

Continuous variable

A continuous variable can be any value between its minimum and maximum limits, otherwise it's a discrete variable. Examples are temperature, weight, age, and time. Continuous variables can include fractions or portions of the value. The total number of blue skateboards sold is a discrete variable since we can't sell half a skateboard.

Correlation

A correlation tells us how the behaviors of things are related. If their patterns of increase and decrease are similar, then they're positively correlated. And if their patterns are opposite, then they're negatively correlated. For example, if sales of our red skateboard increase each time we run a tv marketing campaign, then sales of the red skateboard and the tv campaign are positively correlated.

Cross-filter

Applies to visual interactions. Cross-filtering removes data that doesn't apply. For example, selecting **Moderation** in the doughnut chart cross-filters the line chart. The line chart now only displays data points that apply to the Moderation segment.

Cross-highlight

Applies to visual interactions. Cross-highlighting retains all the original data points but dims the portion that doesn't apply to your selection. For example, selecting **Moderation** in the doughnut chart cross-highlights the column chart. The column chart dims all the data that doesn't apply to the Moderation segment, and highlights all the data that does apply to the Moderation segment.

Custom visual

Visuals that are created by the community and Microsoft. They can be downloaded from the Office store for use in Power BI reports. You can also develop your own personalized custom visual.

D

Dashboard

In the Power BI service, a dashboard is a single page, often called a canvas, that uses visualizations to tell a story. Because it's limited to one page, a well-designed dashboard contains only the most important elements of that story. Dashboards can only be created and viewed in the Power BI service, not in Power BI Desktop. For more information, see basic concepts, dashboards.

Data connector

See connectors

Data model, Excel Data Model

In Power BI content, a data model refers to a map of data structures in a table format. The data model shows the relationships that are being used to build databases. Report designers, administrators, and developers create and work with data models to create Power BI content.

Dataflow

Dataflows ingest, transform, integrate, and enrich big data by defining data source connections, ETL logic, refresh schedules, and more. Formerly data pool.

Dataset

A collection of data used to create visualizations and reports.

Desktop or Power BI Desktop

Free Power BI tool used primarily be report designers, admins, and developers.

Diamond

Power BI Premium. The shape of the icon that signifies a workspace is a Premium capacity workspace.

Dimension

Dimensions are categorical (text) data. A dimension describes a person, object, item, products, place, and time. In a dataset, dimensions are a way to group *measures* into useful categories. For our skateboard company, some dimensions might include looking at sales (a measure) by model, color, country, or marketing campaign.

Drill up, drill down, drill through

In Power BI, "drill down" and "drill up" refer to the ability to explore the next level of detail in a report or visual. "Drill through" refers to the ability to select a part of a visual and be taken to another page in the report, filtered to the data that relates to the part of the visual you selected on the original page. Drill to details commonly means to show the underlying records.

E

Editing View

The mode in which report *designers* can explore, design, build, and share a report.

Effective identity

The identity used by the generate token API to generate a token for an individual user, depending on that user's credentials in the app. It could be a master user or service principal.

Ellipsis

Three dots - ... Selecting an ellipsis displays more menu options. Also referred to as the **More actions** menu.

Embed code

A common standard across the internet. In Power BI, the customer can generate an embed code and copy it to place content such as a report visual on a website or blog.

Embed token

A string of encrypted characters that is used for authentication, which specifies the content the web app user can access and the user's access level.

Embedded

See Power BI Embedded.

Embedding

In the Power BI developer offering, the process of integrating analytics into apps using the Power BI REST APIs and the Power BI SDK.

Environment

[Power BI Desktop, Power BI Mobile, the Power BI service, etc.] Another way to refer to one of the Power BI tools. It's okay to use Power BI environment (tenant) in documentation where it may help business analysts who are familiar with the term tenant to know it's the same thing.

Explicit measures

Power BI uses explicit measures and implicit measures (see definition below). Explicit measures are created by report designers and saved with the dataset. They're displayed in Power BI as fields, and can therefore be used over and over. For example, a report designer creates an explicit measure *TotalInvoice* that sums all invoice amounts. Colleagues who use that dataset and who have edit access to the report, can select and use that field to create a visual. When an explicit measure is added or dragged onto a report canvas, Power BI doesn't apply an aggregation. Creating explicit measures requires edit access to the dataset.

F

Favorite, unfavorite

Verb meaning to add to the Favorites list for quick access to frequently visited dashboards and reports in Power BI. When you no longer want them as a favorite, you unfavorite them.

Filter versus highlight

A filter *removes* data that doesn't apply. A highlight *grays out* the data that doesn't apply.

Focus mode

Use focus mode to pop out a visual or tile to see more detail. You can still interact with the visual or tile while in focus mode.

Free account

See *account*

Full screen, full-screen mode

Use full screen mode to view Power BI content without the distraction of menus and navigation panes. Full screen mode is sometimes referred to as TV mode.

G

Gateways or on-premises data gateways

A bridge to underlying data sources. It provides quick and secure data transfer between the Power BI service and on-premises data sources that support refresh. Managed by IT.

H

High-density visuals

Visuals with more data points than Power BI can render. Power BI samples the data to show the shape and outliers.

Home

The default landing page for Power BI service users. Doesn't modify anything. Can be called Power BI Home or simply Home.

I

Inline frame, iFrame

An iFrame is component of an HTML element that allows you to embed reports and other items inside your app. It's essentially a way to display a secondary webpage inside the main page.

Implicit measures

Power BI uses implicit measures and explicit measures (see definition above). Implicit measures are created dynamically. For example, when you drag a field onto the report canvas to create a visual. Power BI

automatically aggregates the value using one of the built-in standard aggregations (SUM, COUNT, MIN, AVG, etc.). Creating implicit measures requires edit access to the report.

Independent Software Vendor, ISV

A third-party software developer. An ISV can be an individual or an organization that independently creates computer software.

Insights

See quick insights.

Item

A component type of the Power BI workspace that includes dashboards, reports, datasets, and data flows. Formerly known as artifacts.

J

K

KPIs

Key performance indicators. A type of visual.

L

Left navigation (left nav)

The controls along the left edge of Power BI service.

License

Your level of access is determined by the Power BI license associated with your account and the capacity type where content is stored. For example, in shared capacity, a user with a Power BI Pro license can collaborate only with users who are also assigned a Pro license. In shared capacity, a free license enables access to only the user's personal workspace. However, when content is in Premium capacity, users with a Pro license can share that content with users who are assigned a free license.

A license is assigned to a user and can be a free or Pro license. Depending on how the license was acquired, it may be paid or unpaid. The accounts are either: per-user or organizational. Per-user accounts are available as *free* or *Pro*. A Power BI *free* user is either using stand-alone Power BI Desktop or is using Power BI service stand-alone or is using Power BI service within an organization that has a Premium organizational subscription. The Power BI per-user *Pro* account is a paid monthly subscription that allows for collaboration and sharing of content with other *Pro* users.

The organizational *Premium* (also known as *Premium capacity*) subscription adds a layer of features on top of per-user licenses. For example, *free* per-user account holders within an organization that has a *Premium* subscription, are able to do much more with Power BI than *free* users without *Premium*. For example, *free* users in *Premium* organizational accounts, can collaborate with colleagues and can view content that's hosted on Power BI Premium capacity.

List page or content list

One of the section pages for the elements in the nav pane. For example, Favorites, Recents, My workspace, etc.

M

Measure

A measure is a quantitative (numeric) field that can be used to do calculations. Common calculations are sum, average, and minimum. For example, if our company makes and sells skateboards, our measures might be

number of skateboards sold and average profit per year.

Mobile app

Apps that allow you to run Power BI on iOS, Android, and Windows devices.

Modeling

[Power BI Desktop] Getting the data you've connected to ready for use in Power BI. Modeling includes creating relationships between tables in multiple data sources, creating measures, and assigning metrics.

My workspace

The workspace for each Power BI customer to use to create content. If a customer wants to bundle anything created here into an app, and they have *designer* permissions, they upload it to the appropriate workspace or create a new one.

N

Native

Included with the product. For example, Power BI comes with a set of *native* visualization types. But you can also import other types, such as Power BI visuals.

Navigation pane or nav pane

The controls along the left edge of the Power BI service.

Notification

Messages sent by and to the Power BI Notification Center.

Notification Center

The location in the service where messages are delivered to users, such as notice of sunsetting certain features.

O

OneDrive for Business vs OneDrive

OneDrive is a personal account and OneDrive for Business is for work accounts.

On-premises (on-prem)

The term used to distinguish local computing (in which computing resources are located on a customer's own facilities) from cloud computing.

P

PaaS

PaaS stands for platform as a service. For example, Power BI Embedded.

Page

Reports have one or more pages. Each tab on the report canvas represents a page.

Paginated reports

Paginated reports are designed to be printed or shared. They're called *paginated* because they're formatted to fit well on a page. They display all the data in a table, even if the table spans multiple pages. They're also called pixel perfect because you can control their report page layout exactly. Power BI Report Builder is the standalone tool for authoring paginated reports.

PBIVIZ

The file extension for a Power BI custom visual.

PBIX

The file extension for a Power BI Desktop file.

Permissions

What a user can and can't do in Power BI is based on permissions. As a *consumer* you won't have the same permissions as a *designer*, *administrator*, or *developer*.

Phone report

The name for a Power BI report that's been formatted for viewing on the phone.

Phone view

The user interface in the Power BI service for laying out a phone report.

Pin, unpin

The action a report *designer* takes of placing a visual, usually from a report, onto a dashboard.

Power BI Desktop

Also referred to as *Desktop*. The free Windows application of Power BI you can install on your local computer that lets you connect to, transform, and visualize your data. Used by report designers and admins. For more information, see What is Power BI.

Power BI Embedded

A product used by developers to embed Power BI dashboards and reports into their own apps, sites, and tools.

Power BI Premium

An add-on to the Power BI Pro license that enables organizations to predictably scale BI solutions through the purchasing of reserved hardware in the Microsoft cloud. See *account*.

Power BI Pro

A monthly per-user license that provides the ability to build reports and dashboards, collaborate on shared data, keep data up-to date automatically, audit and govern how data is accessed and used, and the ability to package content to distribute (Power BI apps). See *account*.

Power BI Report Builder

A free, standalone Windows Desktop application used for authoring paginated reports. Used by report designers. For more information, see Power BI Report Builder. Power BI Report Builder can be downloaded from the Power BI site.

Power BI Report Server

An on-premises report server with a web portal in which you display and manage reports and KPIs. It allows organizations to build distributed, hybrid BI systems (a mix of cloud and on-premises deployments).

Power BI service

An online SaaS (Software as a service) service. For more information, see What is Power BI.

Power BI tenant

A Power BI account for an organization which has its data stored separately from other organizations but which is accessed through a shared service.

Premium workspace

A workspace running in a capacity, signified to customers by a diamond icon.

Pro license or Pro account

See *account*.

Publish

Power BI service report *designers* bundle the contents of a Power BI workspace to make it available to others as a Power BI app. Power BI Desktop report *designers* use publish to refer to sending a Power BI Desktop report in .pbix format to the Power BI service so that they can build dashboards from it and easily share it with others.

Q

Q&A

The ability to type natural language questions about a dataset and get responses in the form of visualizations. Appears in the Power BI service and Desktop.

Q&A virtual analyst

[Power BI Mobile] For iOS, the conversational UI for Q&A.

QR codes

[Power BI Mobile] A matrix barcode that can be generated for dashboards or tiles in the Power BI service to identify products. QR codes can be scanned with a QR code reader, or with the Power BI Mobile app on iOS or Android, to link directly to the dashboard or tile.

Query string parameter

Add to a URL to pre-filter the results seen in a Power BI report. In the broadest sense, a query string recovers information from a database.

Quick Insights

Quick Insights refer to automatically generated insights that reveal trends and patterns in data.

R

R, Microsoft R

A programming language and software environment for statistical computing and graphics.

Reading View

Read-only view for reports (as opposed to Editing View).

Real-time streaming

The ability to stream data and update dashboards in real time from sources such as sensors, social media, usage metrics, and anything else from which time-sensitive data can be collected or transmitted.

Recent

The container in the nav pane that holds all the individual items (reports, dashboards, etc.) that were accessed last.

Related content

Shows the individual pieces of content that contribute to the current content. For example, for a dashboard, you can see the reports and datasets providing the data and visualizations on the dashboard.

Relative links

Links from dashboard tiles to other dashboards and reports that have been shared directly or distributed through a Power BI app. This enables richer dashboards that support drill through.

Report

A multi-perspective view into a single dataset, with visualizations that represent different findings and insights from that dataset. It can have a single visualization or many, a single page or many pages.

Report editor

The report editor is where new reports are created and changes are made to existing reports by report *designers*.

Report measures

Also called custom calculations. Excel calls these *calculated fields*. See also *measures*.

Responsive visuals

Visuals that change dynamically to display the maximum amount of data and insights, no matter the screen size.

Row-level security, RLS

Power BI feature that enables database administrators to control access to rows in a database table based on the characteristics of the user executing a query (for example, group membership).

Administrators can configure RLS for data models imported into Power BI with Power BI Desktop.

S

SaaS

Software as a service (or SaaS) is a way of delivering applications over the internet—as a web-based service. Also referred to as: web-based software, on-demand software, or hosted software.

Scalability

The capability of a piece of hardware or software or network to expand or shrink to meet future needs and circumstances.

Screenshot

Simple screenshots of a report can be emailed using the send a screenshot feature.

Service

See *Power BI service* A standalone resource available to customers by subscription or license. A service is a product offering delivered exclusively via the cloud.

Service principal

An identity created for use with applications, hosted services, and automated tools to access Azure resources. The service principal tokens can be used to authenticate and grant access to specific Azure resources from a user-app, service or automation tool, when an organization is using Azure Active Directory.

It can sometimes replace the *master user* to authenticate with Power BI.

Settings

The location for Power BI users to manage their own general settings, such as whether to preview new features, set the default language, close their account, etc. Also, users manage individual settings for content assets, alerts, and subscriptions. Represented by a cog icon.

Share, sharing

In Power BI, sharing typically means directly sharing an individual item (a dashboard or report) with one or more people by using their email address. Requires a Power BI Pro license for sender and recipient. On mobile devices, share can refer to native OS share functionality, such as "annotate and share."

Shared with me

The container in the nav pane that holds all the individual items that were directly shared by another Power BI user.

Single sign-on, SSO

An authentication process that permits a user to log on to a system once with a single set of credentials to

access multiple applications or services.

Snapshot

In Power BI, a snapshot is a static image vs. a live image of a tile, dashboard, or report.

SQL Server Analysis Services (SSAS)

An online analytical data engine used in decision support and business analytics, providing the analytical data for business reports and client applications such as Power BI, Excel, Reporting Services reports, and other data visualization tools.

SQL Server Reporting Services (SSRS)

A set of on-premises tools and services to create, deploy, and manage report servers and paginated reports.

Streaming data

See *real-time streaming*.

Subscriptions, Subscribe

You can subscribe to report pages, apps, and dashboards and receive emails containing a snapshot. Requires a Power BI Pro license.

Summarization

[Power BI Desktop] The operation being applied to the values in one column.

T

Tenant

A client organization that is served from a web service (SaaS) which also serves other client organizations, and each organization's data is stored in a separate database.

Tiles

A tile is a snapshot of your data, pinned to the dashboard. A tile can be created from a report, dataset, dashboard, the Q&A box, Excel, SQL Server Reporting Services (SSRS) reports, and more.

Time series

A time series is a way of displaying time as successive data points. Those data points could be increments such as seconds, hours, months, or years.

U

V

Value, values

Numerical data to be visualized.

Visual, visualization

A chart. Some visuals are: bar chart, treemap, doughnut chart, map.

Visual interaction

One of the great features of Power BI is the way all visuals on a report page are interconnected. If you select a data point on one of the visuals, all the other visuals on the page that contain that data change, based on that selection.

Visualizations pane

Name for the visualization templates that ship in the shared report canvas for Power BI Desktop and the Power

BI service. Contains small templates, also called icons, for each native visualization type.

W

Workbook

An Excel workbook to be used as a data source. Workbooks can contain a data model with one or more tables of data loaded into it by using linked tables, Power Query, or Power Pivot.

Workspace

Containers for dashboards, reports, and datasets in Power BI. Users can collaborate on the content in any workspace except My workspace. The contents can be bundled into a Power BI app. Those stored in Premium capacity can be shared with Free users. Personal workspaces (under My workspace) can be hosted in Premium capacity.

X

X-axis

The axis along the bottom, the horizontal axis.

Y

Y-axis

The axis along the side, the vertical axis.

Z

Next steps

Basic concepts for Power BI service consumer

Basic concepts for designers

Power BI Dev Camp

6/30/2022 • 3 minutes to read • Edit Online

Power BI Dev Camp (third-party site) is an educational resource for developers who want to learn more about Power BI as a developer platform. It covers topics such as Power BI embedding and the development of Power BI visuals and custom connectors.

This article provides links to some of the Power BI Dev Camp resources.

Tutorials

The Power BI Dev Camp tutorials are targeted towards developers who are interested in understanding how to embed Power BI items (such as reports, dashboards and tiles) and do other basic embedding tasks.

- App owns data tutorials
- User owns data tutorials

- App owns data embedding with .NET 5 - A tutorial that shows developers how to create a .NET 5 MVC web application that implements Power BI embedding using the *app owns data* embedding model. The tutorial covers advanced topics such as adding project support for TypeScript using `node.js`, and programming the Power BI REST API to generate multi-resource access tokens.

- Tabular-Object-Model-Tutorial A tutorial and sample code programming datasets using the Tabular Object Model with Power BI Desktop and the Power BI Service via the XMLA endpoint.

- PowerBI-PowerShell-Tutorial Students files for the Power BI PowerShell tutorial from Power BI Dev Camp.

Samples

The Power BI Dev Camp samples provide additional resources for developers who are looking to embed Power BI items (such as reports, dashboards and tiles).

- App owns data hello world - A minimal .NET 5 sample application to embed either a standard Power BI report or a paginated report. You can run and test the sample in either Visual Studio Code or Visual Studio 2019. It also includes details for using a Power BI cloud other than the public cloud, which has different URLs and different Azure ID resource IDs.

- App owns data custom web API - A .NET 5 custom web API sample, with a single page application (SPA) client created using JavaScript. The custom web API interacts with the Power BI service API as a service principal, and returns embedding data and embed tokens back to the client. This sample demonstrates collecting telemetry data from the SPA client and storing it in a custom database to monitor report loading performance.

- App owns data and RLS - A sample web application built using .NET 5 and Power BI *app owns data* embedding. Created to demonstrate how to design a security authorization model, which leverages `EffectiveIdentity` and Row-level Security (RLS).

- App owns data multi-tenant - A developer sample project demonstrating how to use service principal profiles to manage a multi-tenant environment with Power BI and App-Owns-Data embedding.

- Tenant management application for Power BI - A .NET 5 sample application that demonstrates how to manage service principals within a large-scale Power BI embedding environment with thousands of customer tenants.

- Salesforce app owns data embedding - A sample project that demonstrates how to implement *app owns data* embedding with Power BI reports. This project has been created using the Salesforce developer experience (SFDX) and the Salesforce command line interface (CLI). The goal of this sample project is to provide guidance and demonstrate best practices for developers who need to implement Power BI embedding in a Salesforce environment.
- Salesforce user owns data embedding - A sample project that includes a simple single page application (SPA) that implements *user owns data* embedding with Power BI reports. The solution is built using three essential files, which include `index.html`, `app.css` and `app.js`. This solution contains an option fourth file named `loading.gif` which is used to demonstrate a white-label loading technique in which the developer can display a custom loading image, instead of the standard Power BI branded loading image.

Videos

To view the Power BI Dev Camp videos, go to the video page.

Presentations

The Power BI Dev Camp presentations can be downloaded from the Camp-Sessions GitHub repository.

Next steps

Embed Power BI content into an application for your customers

Embed Power BI content into an application for your organization

Embed a Power BI report in an application for your organization

Power BI embedded analytics Client APIs

Power BI embedded analytics playground