

Developing with Power BI Embedding

In the five years since its initial release, Power BI has become widely-adopted by corporations and other organizations as a Software-as-a-Service (SaaS) application. The Power BI Service provides licensed users with the ability to access and interact with Power BI reports and dashboards through the *Power BI portal* which most users access through the URL of <https://app.powerbi.com>. The Power BI Service also provides licensed users with a first-class mobile experience using the *Power BI Mobile apps* that Microsoft has published for iPhone, Android and Windows 10. Microsoft promotes Power BI along with Power Apps and Power Automate as the primary services in the Power Platform which offer customers a *no code, low code* approach to building custom business solutions.

In addition to its SaaS offerings, Power BI also provides developers with Platform-as-a-Service (PaaS) capabilities which make it possible to embed Power BI resources into custom applications. By learning to use a set of APIs created by the Power BI team at Microsoft, a developer can embed Power BI reports, dashboards and dashboard tiles into custom web applications that target the browsers, tablets and mobile devices.

This whitepaper focuses on the details and the developer skills required for developing with Power BI embedding and explains the essential concepts and terminology. As you will see, there are quite a few details you must learn when you get started. However, once you understand the landscape of Power BI embedding, you will be able to effectively extend the custom applications and components you are developing with interactive reports and dashboards.

Prerequisites: This whitepaper assumes the reader is already familiar with building and publishing PBIX files using Power BI Desktop as well as building reports and dashboards within the Power BI portal.

Introduction to Power BI Embedding

Let's begin by examining how Power BI embedding works at a fundamental level. From an architectural perspective, Power BI embedding involves adding an HTML `iframe` element to a web page and configuring the `iframe` with a URL and a security token to load a Power BI report, dashboard or tile directly from the Power BI Service. Figure 1.1 depicts a simple web page in a custom web application with an embedded Power BI report.

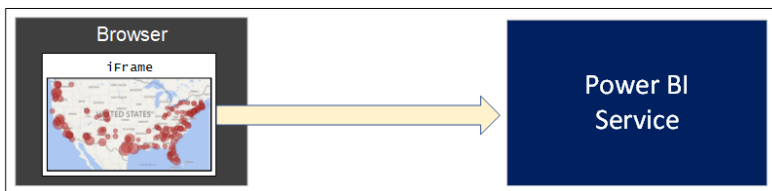


Figure 1.1: Power BI embedding is based on rendering a Power BI resource inside an `iframe` element.

The underlying architecture for Power BI embedding has been designed using open web standards including HTML, CSS, JavaScript, OData, OAuth and OpenID Connect. This means that Power BI embedding techniques can be used by developers using a wide variety of programming languages and development platforms. These open standards also allow custom applications which use Power BI embedding to provide a wide reach to target any type of modern browser as well tablets and mobile devices.

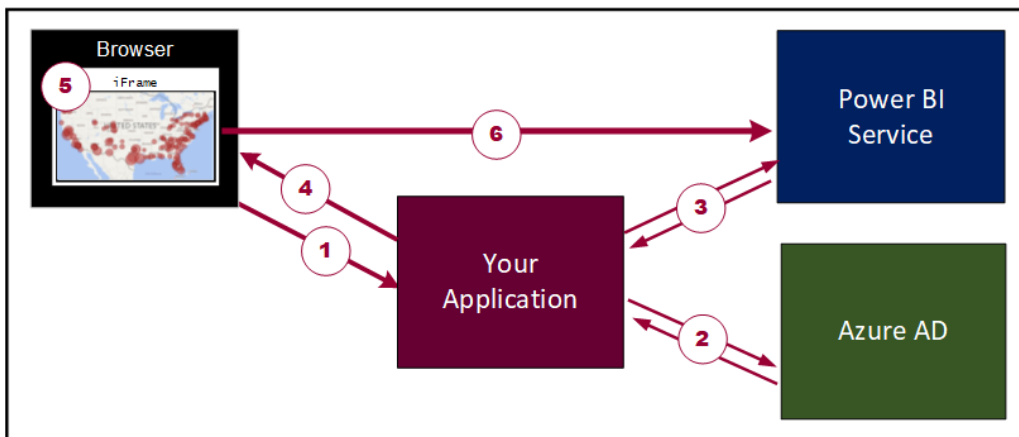


Figure 1.2: The basic set of steps required to implement Power BI embedding.

Applications that implement Power BI embedding are developed using a common pattern. The diagram in Figure 1.2 shows the typical sequence of steps you go through to embed a Power BI report on a web page in a custom web application. Let's walk through the steps.

1. User launches the application.
2. The application calls to Azure AD to obtain an access token which is required to call the Power BI Service API.
3. The application calls the Power BI Service API to retrieve embedding data for a specific Power BI resource.
4. The application passes the embedding data and a security token to client-side code running in the browser.
5. Client-side code in the browser calls the Power BI JavaScript API to embed the Power BI resource.
6. The Power BI JavaScript API dynamically creates the iframe and initializes the embedded resource.

There is an important observation here. Once a Power BI report has been embedded in the iframe, it has a direct connection back to the Power BI Service. This allows the users of your web application to interact with the embedded report using familiar Power BI report features such as slicers, highlighting, drillthrough pages and bookmarks.

Embeddable Resources

The PaaS features of the Power BI Service continue to evolve offering support for new types of embedded resources. Currently, the Power BI Service supports embedding for the following types of Power BI resources.

1. Power BI reports
2. Power BI dashboards
3. Power BI dashboard tiles
4. Power BI Q&A experience
5. Power BI report visuals

While you can embed several different types of Power BI resources, the list of supported features for each type is quite different. More specifically, embedded reports support quite a few extra features that are not supported by any of the other types. For example, you can embed a report in edit mode allowing the user to update the report layout and to save these changes back to the Power BI Service. It is also possible to embed a new report on top of an existing Power BI dataset allowing users to create a report from scratch and save their work back to the Power BI Service.

In a simple *embed-it-and-forget-it* scenario, you can just embed a report and rely on the underlying report to supply its own interactive behavior using slicers, highlighting, drillthrough pages and bookmarks. But you can go far beyond that. Once you master the skills of developing with Power BI embedding, you'll be able to extend the interactive behavior of an embedded report by writing client-side code to set custom filters, apply bookmarks and set custom page layouts. Many developers today are using this approach to embed Power BI reports inside a user interface experience with a custom navigation scheme or a custom filtering experience.

Embedded dashboards do not provide the same level of interactive behavior when compared to embedded reports. For example, embedded dashboard can only exhibit read-only behavior. Unlike an embedded report, users cannot make changes to a dashboard layout nor save any changes back to the Power BI Service. However, embedded dashboards do exhibit interactive behavior because hovering over a dashboard tile will still generate popup tooltips.

There is good news if you are working with real-time dashboards which you have built on top of streaming datasets, push datasets or hybrid datasets. You can embed a real-time dashboard in a custom web application and it will continue to update itself automatically in near real-time just as with dashboards that are accessed through the Power BI portal.

While you can embed an entire dashboard, you also have the option to embed dashboard tiles individually. This can be handy when you want to select which tiles from a dashboard are displayed or you want more flexibility as to where each dashboard tile is displayed. One other thing to keep in mind that you cannot embed a tile from a real-time dashboard and see the updates in real time. When you are working with real-time dashboards, you must embed the entire dashboard and not individual dashboard tiles.

The Q&A Experience is the newest arrival to the supported list of embeddable resources. If you have used the Q&A Experience in the Power BI portal, then you can imagine what the experience would look like when isolated by itself on your custom web page. When you embed the Q&A Experience, you must configure it by referencing an underlying dataset. You can also seed questions into the Q&A Experience to provide the user a starting point that will return results and display an initial visualization.

Now let's discuss how to embed a report visual. In truth, you cannot really embed a report visual by itself. Instead, you play a little trick. You embed the report which contains the visual and apply a custom layout to hide every other visual in

the report except for the visual you want to display. You can implement a custom layout to reset the position, width and height of the visual so it occupies all the real estate inside the hosting iframe.

Custom report layouts can be applied dynamically after an embedded report has already loaded. This can lead to using custom layouts in creative ways. For example, let's say you embed a report with a custom layout that displays a single visual. You could then extend the user experience to provide some interactive control where the user can move from one visual to the next to iterate through all the visuals in the report. The user interface experience of switching between visuals will be very fast because the entire report has already been loaded into memory behind the current web page.

User-Owns-Data Embedding versus App-Owns-Data Embedding

When initially designing a custom application which will implement Power BI embedding, you must decide between two development models: *user-owns-data embedding* versus *app-owns-data embedding*. Whichever of these two development models you choose will determine which identity will be used to authenticate with Azure AD to obtain the access token needed to call into the Power BI Service API.

User-owns-data embedding can be thought of as the *first-party embedding model*. That's because you implement an Azure AD authentication flow to obtain an access token for the current user and you use the current user's identity and permissions to call into the Power BI Service API. User-owns-data embedding is used in scenarios in which all users have an Azure AD user account and a Power BI user license. For example, a company using Office 365 and Power BI might decide to use user-owns-data embedding to develop a custom Intranet-style application which surfaces Power BI reports and dashboards to an audience of users within a single Azure AD tenant.

While user-owns-data embedding is a good fit for organizations using Power BI, Office 365 and Dynamics 365, it is not as useful for independent software vendors (ISVs) and other types of companies that provide SaaS applications to their customers. App-owns-data embedding is more attractive to ISVs because it provides a *third-party embedding model* which can be used in scenarios in which users require neither Power BI licenses nor Azure AD user accounts. Therefore, an ISV can use app-owns-data embedding in commercial applications that use a custom identity provider or some other identity provider service other than Azure AD. It can also be used in scenarios in which obtaining a Power BI license for each user is either impossible or impractical.

When developing application which use app-owns-data embedding, there's no need to acquire a Power BI license for each user. However, the lack of user licensing with app-owns-data embedding begs an important question. If you are not paying Microsoft for the use of Power BI through user licensing, how do you pay them for using the PaaS capabilities of the Power BI Service? The answer involves purchasing dedicated capacities which is explained in the next section.

There is an important security aspect of Power BI embedding that differs between applications that use user-owns-data embedding versus app-owns-data embedded. While you must pass a security token to the browser to properly initialize the iframe with the embedded resource, the type of security token used between these two scenarios is different. When you use user-owns-data embedding, you embed resources using an access token you retrieve from Azure AD when authenticating the current user as shown in Figure 1.3.

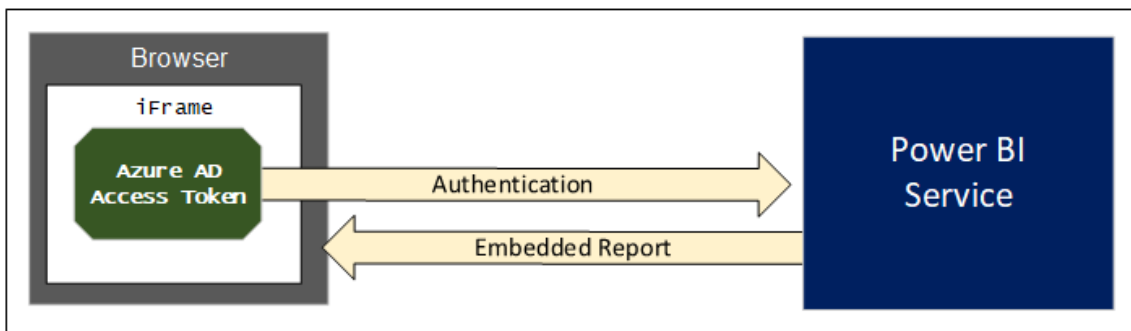


Figure 1.3: User-owns-data embedding involves embedding a resource using an Azure AD access token.

When using app-owns-data embedding, you do not authenticate the current user with Azure AD, so you need to acquire a security token by some other means. This is accomplished by calling into the Power BI Service API to acquire a different type of security token known as an *embed token*.

Figure 1.4 shows how app-owns-data embedding uses a Power BI embed token instead of an Azure AD access token. A key point is that an embed token is far more constrained than an Azure AD access token. That's because an embed token

is specific to one Power BI resource such as a single report or dashboard. When generating an embed token, you can specify the permissions you are granting such as view, edit and create. When using app-owns-data embedding, you will generate a separate embed token for each Power BI resource that you need to embed on a web page.

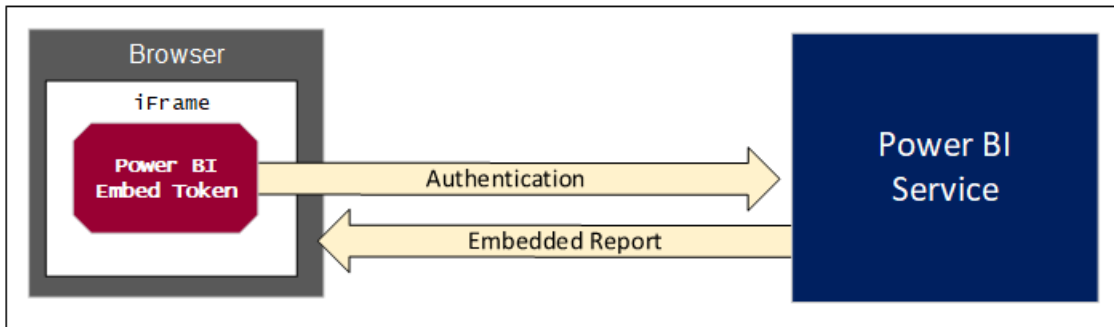


Figure 1.4: App-owns-data embedding involves embedding a resource using a Power BI embed token.

Dedicated Capacities

It has never been a goal of Microsoft to become non-profit organization. Therefore, you should expect to pay licensing fees whenever using Power BI embedding features in a production environment. This is also true when using the PaaS features of the Power BI Service to implement Power BI embedding. For simple user-owns-data embedding scenarios, acquiring a Power BI Pro license for each user might be all you need. For scenarios involving app-owns-data embedding, you cannot pay Microsoft by purchasing user licenses because your users are unknown to Azure AD and the Power BI Service. Instead, you pay for app-owns-data embedding by purchasing a license for a dedicated capacity.

A *dedicated capacity* is an isolated execution environment inside the Power BI cloud. When you purchase a license for a dedicated capacity, you are effectively paying Microsoft for a set of resources including the memory and the processing cycles used to serve up your Power BI content. In many (but not all) scenarios, a dedicated capacity runs on its own isolated hardware within a Microsoft data center. Running a dedicated capacity on its own isolated hardware improves performance and scalability because an organization does not have to worry about issues that arise from sharing memory and processing cycles with other organizations in what is known as the *noisy neighbor* problem.

Once you have acquired a dedicated capacity, you take advantage of it by creating associations with Power BI workspaces. This includes both app workspaces and personal workspaces. Once a workspace has been associated with a dedicated capacity, all the content inside that workspace is then served up by the dedicated capacity.

Any workspace that has not been associated with a dedicated capacity runs within the context of a common execution environment known as the *shared capacity*. Since the shared capacity stores content for and serves content to many organizations at the same time, constraints are placed at the individual level and at the organizational level so that no one organization can monopolize shared capacity resources such as processing cycles and memory.

When you are working with an app workspace running in the shared capacity, there are some important limitations. For example, the datasets you create are limited to 1GB in size in memory. You are also limited in scheduling dataset refreshes to eight times a day. In order to move beyond these limitations, you must acquire a dedicated capacity.

When you need to acquire a dedicated capacity, you must choose between several different choices. First, you can acquire a dedicated capacity through Office 365 by purchasing a Power BI Premium subscription. Power BI Premium subscriptions are available through two families of SKUs known as the P SKUs and EM SKUs. Alternatively, you can provision a dedicated capacity using the Power BI Embedded service in Microsoft Azure. When you create a dedicated capacity in Microsoft Azure using the Power BI Embedded service, you set its pricing tier using an A SKU. Therefore, acquiring a dedicated capacity requires you to choose between these three options.

- Power BI Premium P SKU
- Power BI Premium EM SKU
- Power BI Embedded A SKU

Power BI Premium Capacities

Let's start by examining the P SKUs available through Power BI Premium. Many people refer to P SKUs as the *all-in* SKUs because they offer the greatest number of features. P SKUs are also the most expensive. Table 1.1 shows the five different P SKUs available for Power BI Premium along with their specifications and price. Note that a dedicated capacity created from a Power BI Premium P SKU is always run on its own dedicated hardware.

Power BI Premium P SKUs	P1	P2	P3	P4	P5
Virtual cores (aka v-cores)	8	16	32	64	128
Memory (GB)	25	50	100	200	400
Peek renders per hour	2,400	4,800	9,600	19,200	38,400
DirectQuery executions per second	30	60	120	240	480
Dedicated hardware	Yes	Yes	Yes	Yes	Yes
Price per month	\$4,995	\$9,995	\$19,995	\$39,995	\$79,995

Table 1.1: You can purchase Power BI Premium capacities using five different P SKUs.

The P SKUs offer valuable features to both the PasS capabilities and the SaaS capabilities of the Power BI Service. When using the PaaS capabilities of the Power BI Service, you can use a P SKU to support either user-owns-data embedding or app-owns-data embedding. If your organization is leveraging the SaaS capabilities of the Power BI Service to serve reports and dashboards to a large number of content consumers, a P SKU can save your organization money. This requires a bit more explanation.

Microsoft sells Power BI using both user-based licensing and capacity-based licensing. In organizations that rely exclusively on user-based licensing, users with the Power BI free license are not allowed to share content with other users nor can they consume content shared by other users. This means that in an environment without a dedicated capacity that all users require a Power BI Pro license regardless of whether they are authoring content for other users or just consuming content created by someone else.

Capacity-based licensing using Power BI Premium makes it possible for users with the free Power BI license to consume content created by others. When an organization is using Power BI Premium, they still require a Power BI Pro license for each author, but they are able to eliminate all user-based licensing costs for their read-only consumers.

Consider a simple example of an organization which has 20 Power BI content authors and 1000 other users who will be read-only consumers of Power BI content. Without capacity-based licensing, a company would have to purchase a Power BI Pro license for all users which will cost \$10,200 per month.

\$10,200/month = 1020 Power BI Pro licenses @ \$10/month

If an organization purchases the Power BI Premium P1 SKU for \$4,995 per month, the total cost of licensing drops down to \$5,195 per month.

\$4995/month - 1 Power BI Premium P1 license @ \$4995/month

\$200/month - 20 Power BI Pro licenses @ \$10/month

\$5195/month - Total cost

In addition to the Power BI Premium P SKUs, Microsoft also offers Power BI Premium EM SKUs which can be purchased at a much lower price point. The key difference between these two SKUs is the P SKUs make it possible for users with the Power BI free license to consume content through the Power BI portal and through the Power BI Mobile apps. The Power BI Premium EM SKUs, on the other hand, do not provide users with the Power BI free license to access content through the Power BI portal or through the Power BI Mobile apps. Therefore, the EM SKUs are primarily valuable in cases where you are embedding reports outside of the user interface experience of the Power BI portal.

Currently, Microsoft offers three Power BI Premium EM SKUs that are shown in Table 1.2.

Power BI Premium EM SKUs	EM1	EM2	EM3
Virtual cores (aka v-cores)	1	2	4
Memory (GB)	2.5	5	10
Peak renders per hour	300	600	1200
DirectQuery executions per second	3.75	7.5	15
Dedicated hardware	No	No	Yes
Price per month	\$625	\$1,245	\$2,495

Table 1.2: You can purchase Power BI Premium capacities using three different EM SKUs.

The primary use cases for using Power BI Premium EM SKUs typically involve a small number of users. The first use case involves no-code scenarios where business users are embedding Power BI reports in a Microsoft Teams tab or on a site page in SharePoint Online using the Power BI webpart which is provided as a standard webpart. The second use cases involves development scenarios with Power BI embedding where you are using user-owns-data embedding to serve up content to users who do not possess Power BI Pro licenses.

While Power BI Premium EM SKUs are cheaper than the others, they have a few important limitations which you should understand. While EM1 and EM2 are the least inexpensive, they do not run on dedicated hardware and they require a yearly commitment through Microsoft's volume discounting program. A dedicated capacity based on a EM3 can be acquired with a monthly commitment and it will run on dedicated hardware. However, what happens when you compare the EM3 SKU to the P1 SKU?

A EM3 capacity is half the cost of a P1 capacity and it gets half of what the P1 capacity gets in terms of v-cores and memory. The EM3 SKU is half the price of the P1 SKU so you might think it's a good deal. However, you should remember that P SKUs provide free license users with the ability to access to content created by someone else through the Power BI portal at <https://app.powerbi.com> and through the Power BI Mobile apps while the EM SKUs do not.

You can purchase a monthly subscription to Power BI Premium through the Purchases Services link in the Office 365 admin Center. The act of purchasing a Power BI Premium subscription will trigger the Power BI Service to provision a new dedicated capacity inside your Office 365 tenant. Once this dedicated capacity has been provisioned, you can view and manage it from the Power BI Admin portal as shown in Figure 1.5.

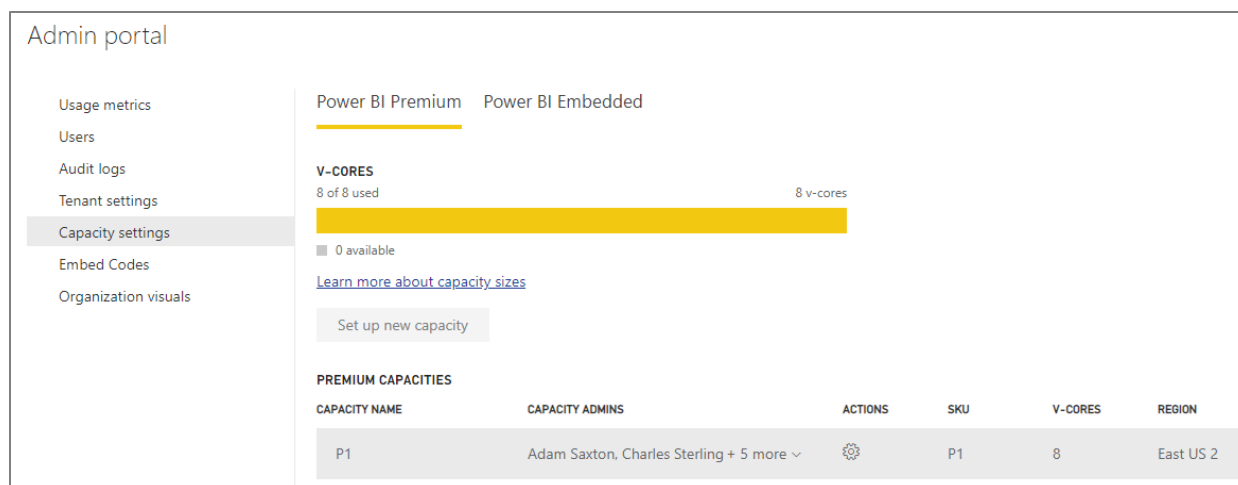


Figure 1.5: The Power BI Admin portal allows you to view and manage dedicated capacities.

As you can see from the screenshot shown in Figure 5, the Power BI Admin portal provides one tab to display *Power BI Premium* capacities and a second tab to display *Power BI Embedded* capacities. Power BI Premium capacities and Power BI Embedded capacities are both types of a dedicated capacity. Both types of dedicated capacities are useful in various scenarios involving software development with Power BI embedding. However, there are also important differences between Power BI Premium capacities and Power BI Embedded capacities which you should understand.

One big difference between a Power BI Premium capacity and a Power BI Embedded capacity has to do with where you go to manage and monitor them. Power BI Premium capacities can be managed and monitored from within the Power BI Admin portal. Power BI Embedded capacities are managed and monitored using the Azure portal or by using PowerShell commands or REST API calls available through the Azure Resource Manager.

If you drill into the view for a Power BI Premium capacity in the Power BI Admin portal, you can see statistics on its CPU and memory usage, memory thrashing and the number of queries executed against the DirectQuery limit. By monitoring these statistics, you can assess on a day-by-day basis whether a dedicated capacity is being overloaded. You can also see if you are paying more than you need to service your audience of users.

When you purchase a new subscription for a Power BI Premium P SKU, the Power BI Service responds by provisioning a single Power BI Premium capacity using all the resources that come with the SKU. For example, imagine you purchase a Power BI Premium P3 subscription through the Office 365 Admin center. The Power BI Service will initially create a new Power BI Premium capacity with 32 v-cores. Once the capacity has been created, you can spread its resources across two or more capacities. For example, you could split up the 32 v-cores from a P3 subscription into two capacities with 16 v-cores each. Alternatively, you could split the 32 v-cores by creating one capacity with 24 v-cores for high priority workloads and a second capacity with 8 v-cores for less important workloads.

See the following link for more information on configuring Power BI Premium capacities:

<https://docs.microsoft.com/en-us/power-bi/service-admin-premium-manage>

Keep in mind that a dedicated capacity doesn't provide any real value until you populate it with content. You add content to a dedicated capacity by creating workspace associations. To publish content, you first create a new app workspace and associate it with a dedicated capacity. Next, you populate content in the app workspace by publishing PBIX files to add datasets and reports and by building dashboards in the Power BI portal. After that, all the content in that app workspace will be served from the dedicated capacity making it available to users without a Power BI Pro license.

Each dedicated capacity has a configurable set of capacity administrators. A Power BI Pro user in the role of capacity administrator can associate app workspaces with a dedicated capacity using the Power BI Admin portal or by using Power BI Administrative PowerShell commands. A content author with a Power BI Pro license who has been granted the *Capacity assign* permission can create a new app workspace and associate it with a dedicated capacity using the *Create an app workspace* dialog as shown in Figure 1.6.

Create an app workspace

Name your workspace

Daily Reporter Pro Workspace

Available

Description

This app workspace will run inside a Power BI Premium capacity

Advanced ^

Dedicated capacity ⓘ

On

Choose an available dedicated capacity for this workspace

P1 - East US 2

Save Cancel

Figure 1.6: The *Create an app workspace* dialog allows you to associate a new workspace with a dedicated capacity.

Remember that any workspace that has not been associated with a dedicated capacity will run in the shared capacity and that all users require a Power BI Pro license to consume content from an app workspace running in the shared capacity. This is true in scenarios where users are accessing Power BI content through the Power BI portal. It is also true in scenarios where you are developing custom applications that use first-part embedding.

When you are navigating between workspaces in the Power BI portal, it's easy to tell which ones are associated with a dedicated capacity because they are displayed with a diamond icon as shown in Figure 1.7. Any workspace that is not displayed with the diamond icon is running within the shared capacity.

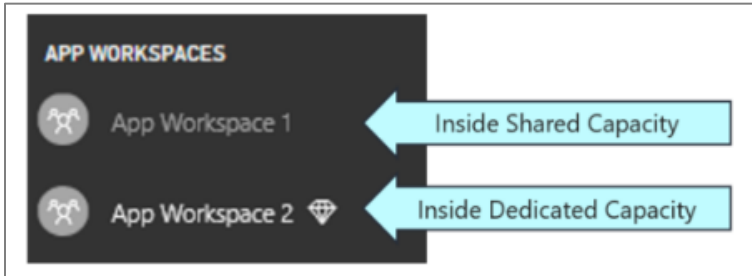


Figure 1.7: A workspace associated with a dedicated capacity is displayed with a diamond icon.

Power BI Embedded Capacities

The Power BI Embedded service in Microsoft Azure allows you to create a dedicated capacity as an scalable, on-demand service. This type of dedicated capacity is known as a *Power BI Embedded capacity*. A key point about Power BI Embedded capacities is that they do not provide support for users with the Power BI free license to access content through the Power BI portal at <https://app.powerbi.com> or through the Power BI Mobile Apps. For this reason, Power BI Embedded capacities are mainly used by ISVs and not by corporation with a large number of users within a single organization.

You must acquire an Azure subscription to create a Power BI Embedded capacity. When creating a Power BI Embedded capacity, you must choose a pricing tier which determines its cost and the number of resources are allocated to it. The available pricing tier SKUs are A1, A2, A3, A4, A5 and A6 as shown in Table 1.3.

Power BI Embedded Capacity Pricing Tiers	A1	A2	A3	A4	A5	A6
Virtual cores (aka v-cores)	1	2	4	8	16	32
Memory (GB)	3	5	10	25	50	100
Peek renders per hour	300	600	1,200	2,400	4,800	9,600
DirectQuery executions per second	3.75	7.5	15	30	60	120
Dedicated hardware	No	No	Yes	Yes	Yes	Yes
Approximate price per month	\$750	\$1,495	\$2,995	\$5,995	\$11,995	\$23,995
Approximate price per hour	\$1	\$2	\$4	\$8	\$16	\$32

Table 1.3: When creating a Power BI Embedded capacity, you must choose between 6 available pricing tiers.

Power BI Embedded capacities are particularly attractive to ISVs because they provide the following features that are not matched by Power BI Premium capacities.

1. You can automate the provisioning of Power BI Embedded capacities using PowerShell scripts and Azure templates.
2. Power BI Embedded capacities do not require a monthly commitment; instead you pay by the hour.
3. You can pause and resume a Power BI Embedded capacity at any time to save money.
4. You can scale a Power BI Embedded capacity up or down in a matter of minutes.

Power BI Embedded capacities will appeal to organizations and developers who have already embraced Microsoft Azure. Just as with other types of on-demand services in Azure such as web apps, storage accounts and virtual machines, you can create a Power BI Embedded capacity by hand in the Azure portal. Alternatively, you can automate the provisioning of Power BI Embedded capacities using Azure templates, PowerShell script or REST-based API calls to the Azure Resource Manager. Figure 1.8 shows a screenshot of what a new Power BI Embedded capacity looks like once it has been created in the Azure portal.

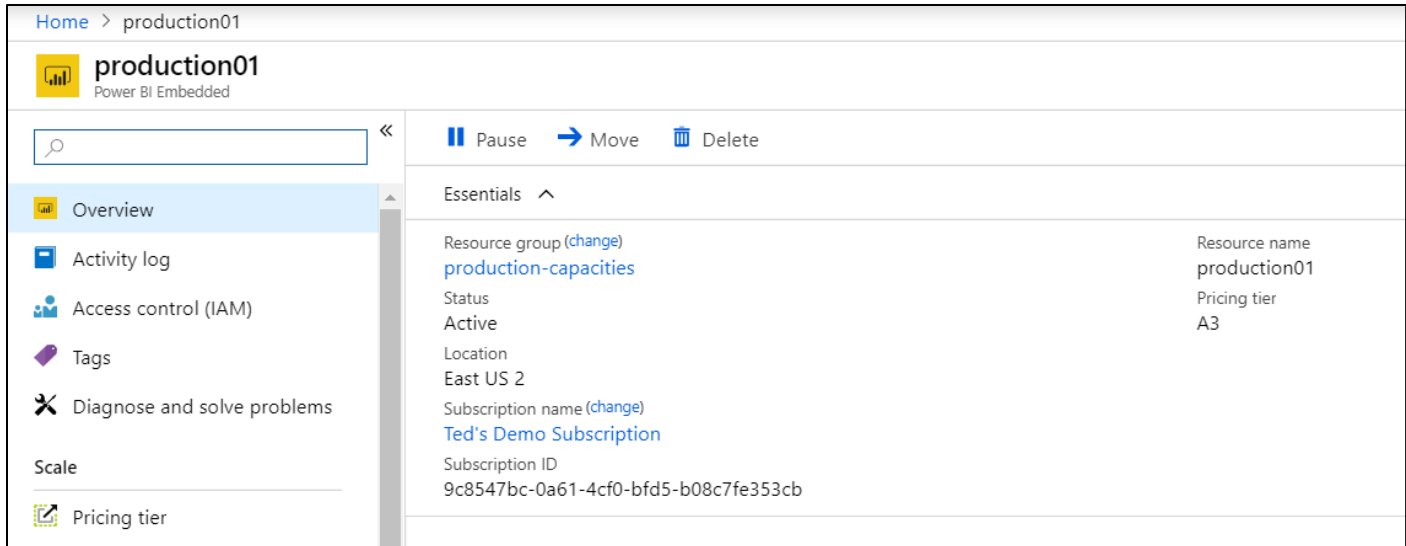


Figure 1.8: You can create, manage and monitor Power BI Embedded capacities in the Azure portal.

If you examine the screenshot in figure 8, you will notice that the toolbar for a Power BI Embedded capacity provides a **pause** button. A Power BI Embedded capacity is an Azure service that you can pause and resume at any time. Since there is no monthly commitment with a Power BI Embedded capacity, you are only charged while the service is running. If you are in a scenario where a Power BI Embedded capacity doesn't have to run 24x7, you can shut it off to save money. It only takes a minute or two to pause or to resume the service.

Power BI Embedded capacities also have a significant advantage over Power BI Premium capacities when it comes to scaling up or down. Consider a scenario where your reporting application experiences a burst of user activity on the first day of each month. You can configure the hosting Power BI Embedded capacity with a less expensive pricing tier (e.g. A2) for standard usage. Then you can scale the Power BI Embedded capacity up to a more powerful pricing tier (e.g. A4) for the day in each month where the spike appears.

The Azure portal also provides monitoring tools so you can determine exactly how much you need to spend to achieve the required level of performance for a given user audience. While you must use the Azure portal and other Azure DevOps tools to create, configure and monitor Power BI Embedded capacities, you can use the Power BI Admin portal to view Power BI Embedded capacities and to see their SKU (aka pricing tier), Region and Status as shown in Figure 1.9.

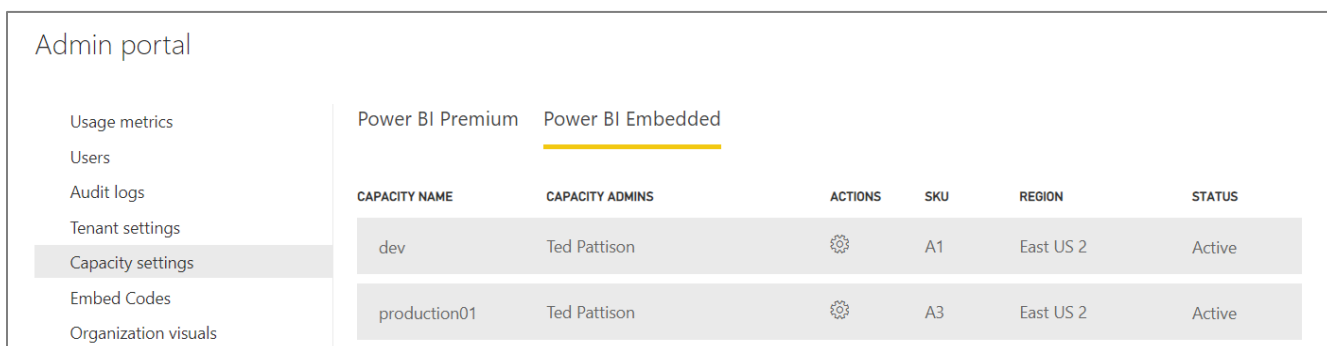
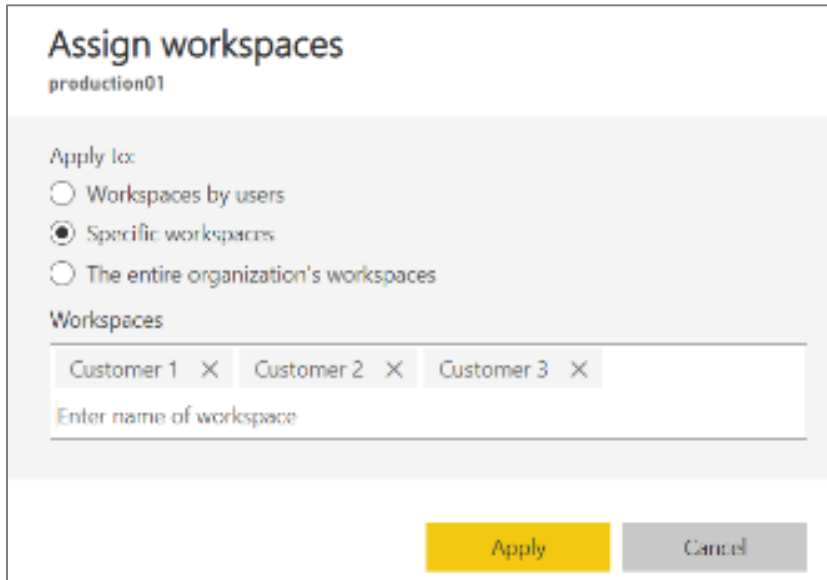


Figure 1.9: You can view Power BI Embedded capacities and associate workspaces in the Power BI Admin portal.

If you drill into a specific Power BI Embedded capacity in the Power BI Admin portal, you will notice that it does not provide you with the ability to configure the underlying capacity or to assign capacity administrators. Since a Power BI

Embedded capacity is an Azure service, you must navigate to the Azure portal to change its pricing tier or assign capacity administrators.

While you cannot configure a Power BI Embedded capacity in the Power BI Admin portal, you can configure its workspace associations just as with a Power BI Premium capacity using the **Assign workspaces** dialog as shown in Figure 1.10.



Assign workspaces
production01

Apply to:

- ☐ Workspaces by users
- ☒ Specific workspaces
- ☐ The entire organization's workspaces

Workspaces

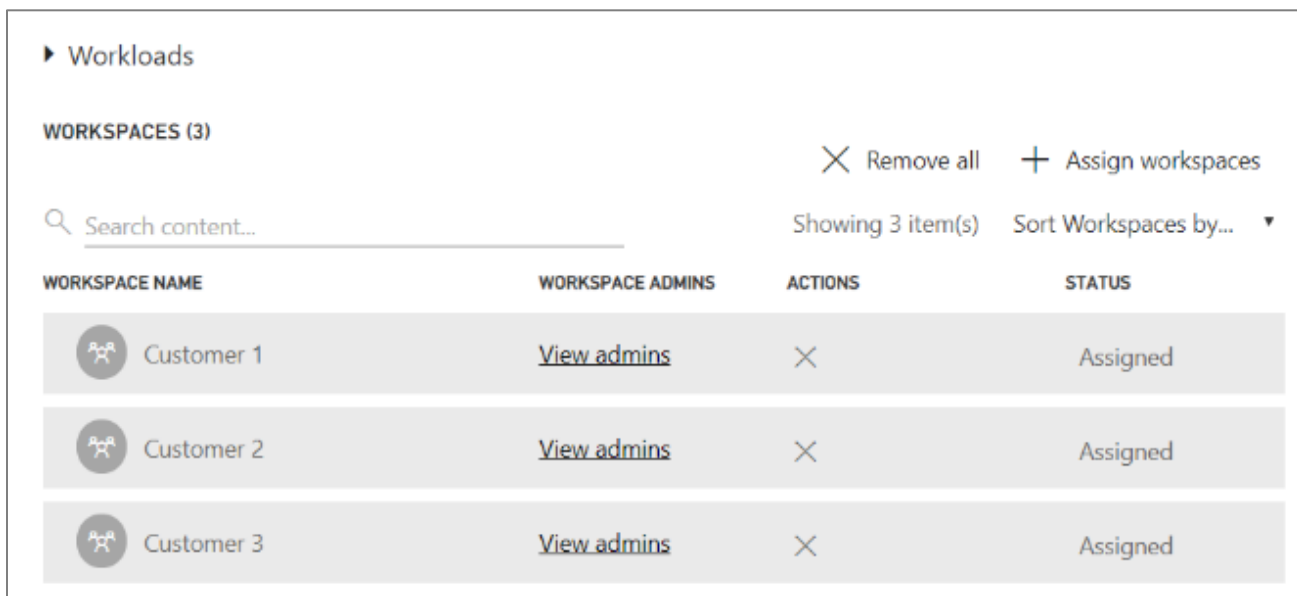
Customer 1 X Customer 2 X Customer 3 X

Enter name of workspace

Apply Cancel

Figure 1.10: The Assign workspaces dialog for a dedicated capacity allows you to create workspace associations

The Power BI Admin portal tab for all types of dedicated capacities contains a **Workloads** section that displays all the current workspace associations as shown in Figure 1.11. This view provides the capacity administrator with a user experience to view and manage the workspaces running inside a dedicated capacity.



► Workloads

WORKSPACES (3)

✕ Remove all + Assign workspaces

🔍 Search content... Showing 3 item(s) Sort Workspaces by... ▼




WORKSPACE NAME	WORKSPACE ADMINS	ACTIONS	STATUS
 Customer 1	View admins	✕	Assigned
 Customer 2	View admins	✕	Assigned
 Customer 3	View admins	✕	Assigned

Figure 1.11: The Power BI Admin portal allows you to view and manage workspaces associations.

Configuring Production Tenants

Before deploying a custom application that uses Power BI embedding into production, you must make sure that the hosting Azure AD tenant is properly configured with whatever dedicated capacities are required. Over the last few pages, you've learned about three different families of SKUs for creating dedicated capacities including the P SKU, the EM SKU

and the A SKU. Table 1.4 provides a comparison of the three SKU types to assist you with selecting the best type of dedicated capacities for a particular scenario.

	P SKU	EM SKU	A SKU
Purchased through...	Office 365	Office 365	Microsoft Azure
Supports app-owns-data embedding in custom applications	Yes	Yes	Yes
Supports user-owns-data embedding in custom applications	Yes	Yes	Yes
Supports free users accessing content in the Power BI portal	Yes	No	No
Supports free users accessing content in Power BI Mobile	Yes	No	No
Billing cycle	Monthly	Monthly	Hourly
Commitment	Monthly	Monthly/Yearly	None
Turn it off when you're not using it	No	No	Yes

Table 1.4: Each of the three SKUs for purchasing a dedicated capacity offers a unique set of different features.

For any given scenario, you should be able to use the comparison points shown in Table 4 to determine the best type and number of dedicated capacities required. You should also have a plan to monitor the usage statistics of these dedicated capacities over time so you can ensure you have the resources you need to maintain adequate performance and that you are not spending more than you need to. If you can move from a Power BI Premium P5 subscription down to a P4 subscription, it will save your organization \$480,000 per year.

Let's look at a few examples. Consider a scenario in which you have developed an application that uses user-owns-data embedding and the application has a small audience of 20-50 users. Given the small number of users, a Power BI Premium capacity will not be cost effective. The recommended approach would be to assign Power BI Pro users accounts to all users to allow them to consume content from the shared capacity.

Consider a second scenario where an application uses the app-owns-data embedding and third-party embedding model. This represents the classic ISV scenario where the ISV application handles user authentication and the user identities are unknown to Azure AD and the Power BI Service. This scenario will always require a dedicated capacity when used in a production environment. You can choose between either a Power BI Premium capacity or a Power Embedded capacity because they both support app-owns-data embedding. As an ISV, you will likely choose to go with Power BI Embedded capacities due to the provisioning and scaling advantages they have over Power BI Premium capacities.

Power BI Premium capacities are most useful to organizations with a large number of internal Power BI users. A common motivation for purchasing a Power BI Premium capacity is to move a large number of read-only consumers from the Power BI Pro license to the free license. A Power BI Premium P1 subscription costs \$5000 per month. Once you exceed the breakeven number of 500 users who are read-only consumers, the Power BI Premium P1 subscription will be less expensive than purchasing a Power BI Pro license for every user.

A second motivation for purchasing a dedicated capacity is to move beyond the limitations and constraints imposed by the shared capacity. You will need to acquire a dedicated capacity if you are working with datasets that are larger than 1GB in memory or if you need to schedule data refresh more than eight times a day. If you want the content to be accessible through the Power BI portal and the Power BI Mobile apps, then the type of dedicated capacity you need is a Power BI Premium capacity created with a P SKU.

A Power BI Premium capacity is flexible because it offers a combination of SaaS features and PaaS features at the same time. Imagine a scenario where you have purchased a Power BI Premium capacity and you are using it to serve up content through the Power BI portal and the Power BI Mobile apps to an internal audience of 2,000 employees within a single organization who all have the Power BI free license. At this point you're only leveraging the SaaS features of the Power BI Premium capacity.

Now let's say you decide to develop a custom application that uses app-owns-data embedding to display Power BI reports and dashboard to an external audience of customers or partners. You can use the same Power BI Premium capacity to

serve content to both audiences. Alternatively, you could split the Power BI Premium capacity into two smaller dedicated capacities so you can isolate the external audience of customers or partners from the internal audience of employees.

Creating a Development Tenant

Now that you have learned about what's required in a production tenant, it's time to discuss how to create a development environment for Power BI embedding. After all the intimidating pricing information you have seen for dedicated capacities, you'll be happy to know that you can set up a development environment for Power BI embedding in a matter of minutes and it will not cost you a dime. All you really need is an Office 365 development tenant and a Power BI Pro license.

If you do not already have a development tenant, you can create a new Office 365 tenant with a free 30-day trial of the Office 365 E5 subscription for up to 25 users. The Office 365 E5 subscription includes a Power BI Pro subscription which provides the user-based licensing you need to call the Power BI Service API and to implement Power BI embedding in a development scenario.

In any production scenario, your custom application is only permitted to generate embed tokens for content in workspaces associated with a dedicated capacity. Fortunately, these rules are relaxed for development scenarios. That means you can rely on the shared capacity when you are developing with Power BI embedding which eliminates the cost and overhead of provisioning dedicated capacities during the development lifecycle.

Keep in mind that Microsoft is able to monitor the generation of embed tokens across all Power BI customer tenants. This gives Microsoft the ability to detect when embed tokens are being generated from the shared capacity and used to embed Power BI resources. This, in turn, gives Microsoft the ability to assess whether the generation and usage of embed tokens within your tenant falls within reasonable limits for development scenarios.

Microsoft monitors shared capacity statistics including the number of embed tokens generated per hour and the number of IP addresses where embed tokens are being used to embed Power BI resources. If Microsoft sees evidence of production usage of embed tokens generated from the shared capacity, that will raise a flag and you can expect a call from Microsoft support or your Microsoft sales representative.

Downloading the Sample Power BI Embedding Projects

This whitepaper contains dozens code listings to illustrate programming techniques with Power BI embedding. The majority of these code listings have been taken from the following set of sample development projects that are maintained in the **Demos** folder of the same GitHub repository that contains this whitepaper.

- **PowerBiPublicClient**: .NET Framework C# console application which demonstrates MSAL authentication flows
- **PowerBiSdkDemos**: .NET Framework C# console application which demonstrates Power BI Service API programming.
- **AppOwnsDataApp**: ASP.NET MVC application which demonstrate app-owns-data embedding
- **PowerBiDaySpa**: Single page application (SPA) created with Node.js, TypeScript and React.js
- **UserOwnsDataApp**: ASP.NET MVC application which demonstrate user-owns-data embedding
- **PowerBiEmbeddedScratchpad**: .NET Framework C# console application demonstrating the Power BI JavaScript API.

As you continue to read through this whitepaper, you are encouraged to download these sample projects and get them running on your developer workstation. For many developers, there is no substitute for the spontaneous gratification of being able to press the {F5} key and see the code running right before their eyes.