

# Use Cases Drive SharePoint Embedded Adoption

SharePoint Embedded solutions drive adoption by seamlessly integrating robust collaboration and document management capabilities with existing platforms and industry-specific use cases. By using SharePoint Embedded directly into familiar tools and workflows, organizations can streamline operations, enhance user experiences, and centralize critical resources. Whether used in project management, customer relationship management (CRM), enterprise resource planning (ERP), or compliance tracking, the embedded functionality allows teams to access and share information without switching between systems. This integration not only improves productivity and collaboration but also accelerates user adoption by meeting employees where they already work, maximizing the value of existing investments while delivering tailored solutions for industry needs.

## When to use this

As you are developing an application or integrating with an existing product that needs to store and collaborate on documents, SharePoint Embedded is the scalable choice.

Writing your own document management system is going to be cost prohibitive. We have made it look easy with SharePoint and OneDrive but solving hard problems like co-authoring, Purview integration and the collaboration features you have come to expect has already been done and ready to implement into your solution using SharePoint Embedded

## Common use cases

The need for document management capabilities in custom or partner solutions spans across all industries.



When it comes to the functionality in SharePoint Embedded, this is what we're seeing:

- **Document collaboration using Office Online** – if you’re building an internal application or one that is sold in the Azure marketplace, users are demanding the office experience.
- **Development environment flexibility** – bring your own language or platform that supports Graph API calls. Regardless if you’re a pro-dev and prefer C# or React, a power user that is using the Power Platform or integrating with existing platforms

such as Salesforce or Service Now. SharePoint Embedded is that core platform to manage your documents.

- **Customer/client management** – we all have customers and vendors that we need to get information from or work together with. When there is a document in the mix, SharePoint Embedded has controls to share and secure the document, enabling you to collaborate without attachments going back and forth.
- **Legal documents** – Collecting all the necessary documents necessary for a legal case can be the difference between win and losing the case. Sprinkle in Microsoft Agents to reason over those documents and the power of AI to quickly find what your looking for.

# When should I use SharePoint Embedded

A question we get quite often is “how do I choose where to store my documents?”. With several options depending on the use case you have, I’ll provide some practical guidance on things to consider.

## When should I use this

As you are re-platforming an existing solution, creating something from scratch or integrating into another solution, document storage is something we have been doing for many years at scale. It’s not a buy vs build decision but rather a which platform is right decision and that needs to happen early in the development lifecycle.

## Things to consider

There are several key points you can highlight to address concerns about paying for storage when they already have plenty of SPO storage or are deciding which platform is best for their use case:

**Enhanced Performance and Scalability:** SharePoint Embedded is designed to handle large-scale data and high-performance requirements more efficiently. This can lead to better performance and scalability compared to traditional data repositories.

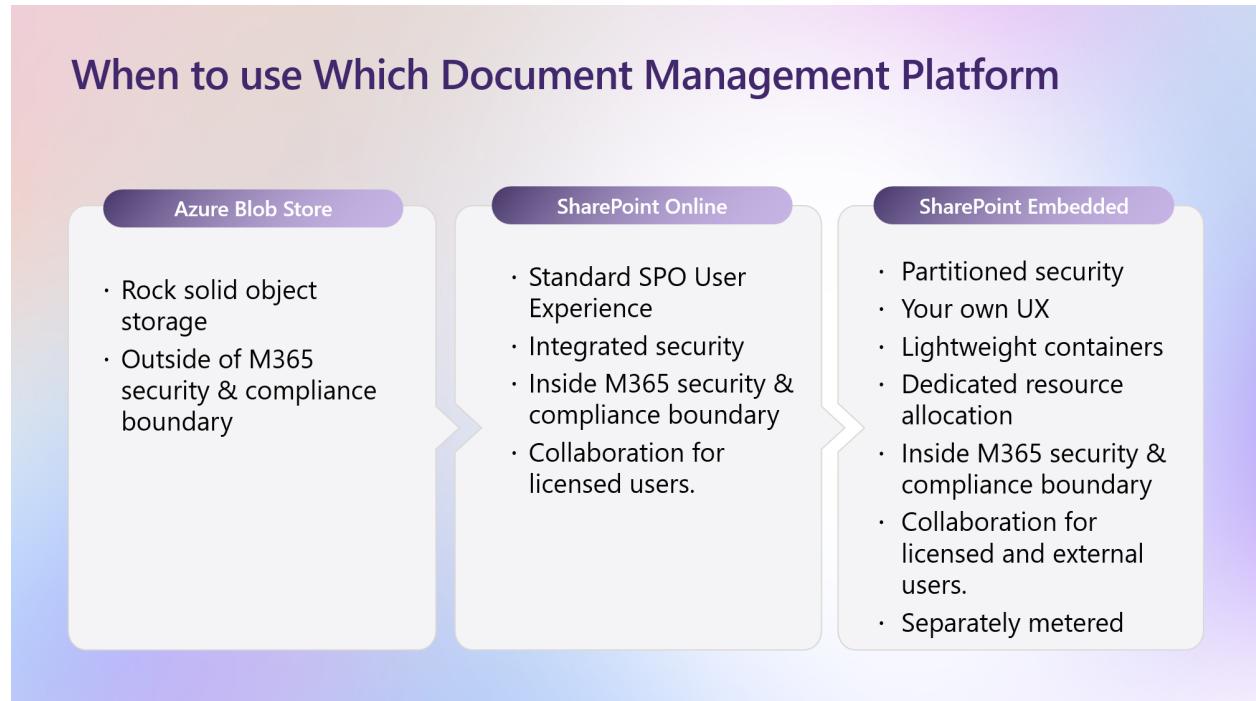
**Data Isolation:** When creating a custom application that uses SPO as the document repository, users can discover the site and freely navigate to view, edit and delete information in the libraries and lists your application depends on, potentially breaking the application. SPE provides a way to totally control the user experience and preserve the integrity of the business logic by exposing only features that you want the user to have. This controlled experience can be critical for enterprise mission critical applications.

**Cost Management:** While there is an additional cost for storage, it can be more predictable and manageable. Customers can scale their storage needs up or down based on their requirements, potentially leading to cost savings in the long run. We also see this as a way to monitor and “charge back” to internal customers for the use of the app and required storage.

**External Collaboration:** For external-facing applications, SharePoint Embedded provides access to storage containers that by default use the tenant sharing settings. These can be overridden at the container level reducing the risk of data exposure throughout the tenant.

**Copilot isolation:** The content in SharePoint embedded by default is not exposed to an organizations Copilot experience. Additional content scoping can be done at the container or container type level depending on the application needs.

This illustration can also help you determine the right platform for your use case.

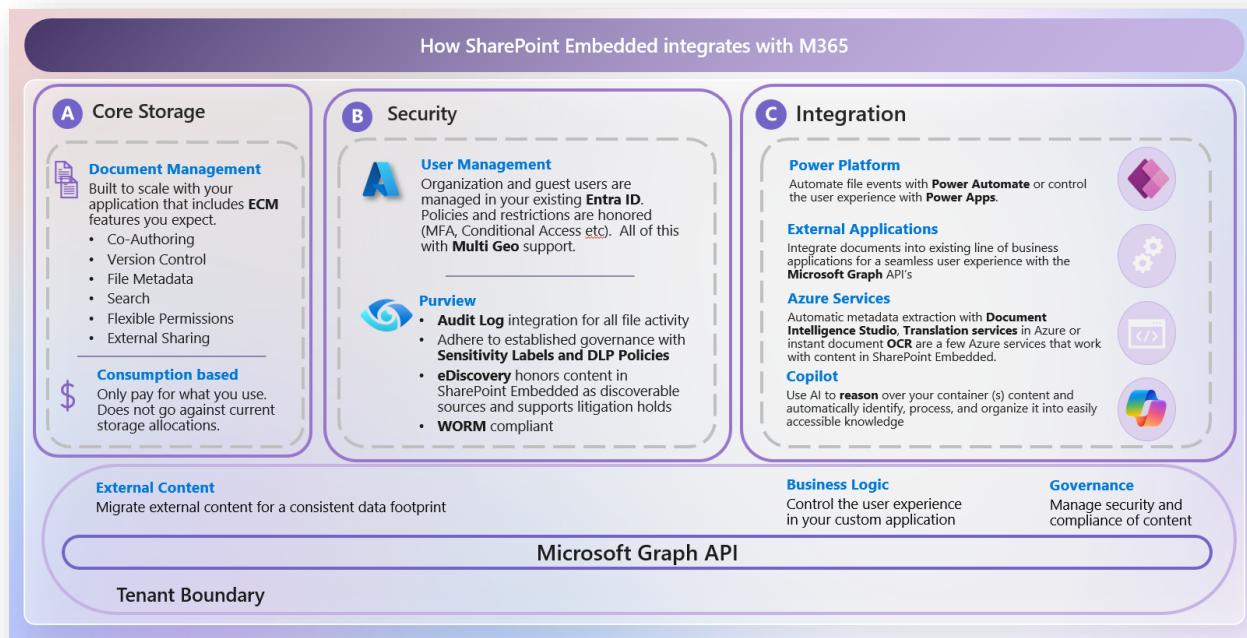


# SharePoint Embedded and M365 – Better Together

Considering almost 80% of custom applications that are written require a way to manage documents and the engineering effort to support even the basic functionality can easily reach 30% of the development effort, it quickly becomes apparent that if you want to deploy your application on time and within budget, you need to use a scalable platform that was built to handle document management.

SharePoint Embedded can not only handle all of your ECM requirements, but it is also integrated into the M365 ecosystem enabling you to manage and govern the content.

Here is an overview of how it all works together.



SPE and M365 - better together

- **Core storage** - incorporates all the basics of document management that your users expect. With the full Microsoft Office Online capabilities, collaborating on documents is the same experience.
- **Security** - Every application has users interacting with it, Entra ID is the cornerstone of user management and protects bad actors from gaining access to your tenant.

- **Purview** - Extend your governance footprint to include SharePoint Embedded content to maintain a consistent security posture for your content.
- **Integration** - Documents are at the core of most activities in your organization. Extending that to the applications and workflows that run your business is easy with connectors and API's.

# Administrative vs Developer Roles – why it matters

In managing a SharePoint Embedded environment, there are two primary roles. The Administrator is responsible for tasks such as Azure configuration, Purview integration, managing permissions, and billing management. Meanwhile, the Developer handles configuring the development environment, application architecture, container management, API integration, and application development.

## When to use it

As you are developing your application, there are certain tenant safeguards that need to be in place to protect the integrity and governance put in place by your organization. Having a clear separation of roles is essential to maintain this level of isolation and accountability.

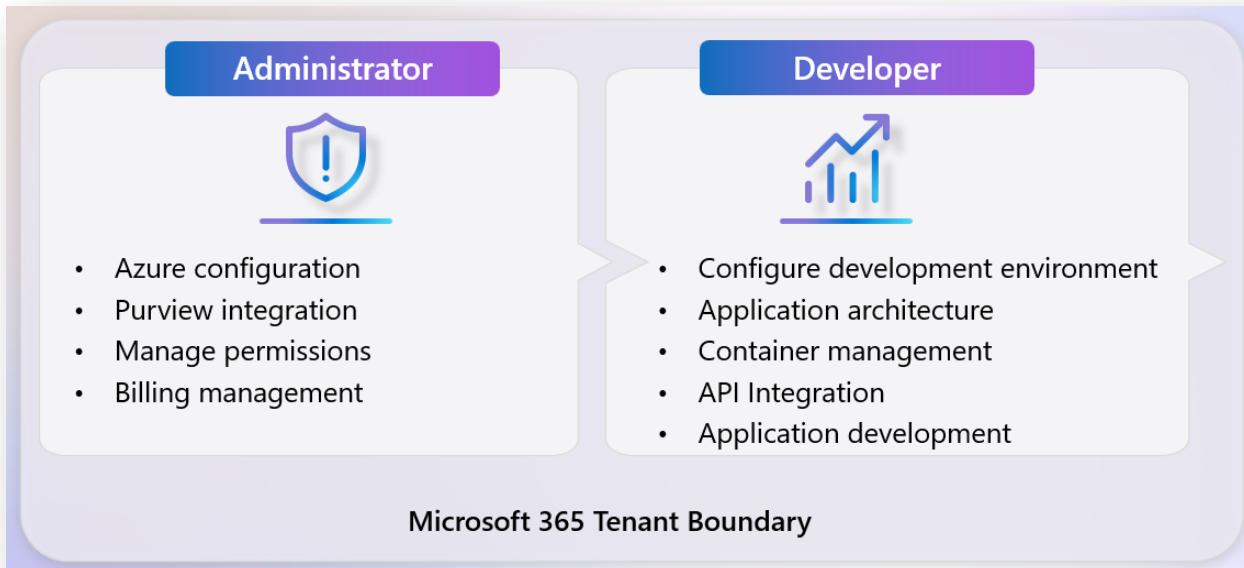
## Overview

When SharePoint Embedded was released, a new role was added, SharePoint Embedded Administrator. Global Administrators can assign the SharePoint Embedded Administrator role available in M365 Admin Center or Microsoft Entra to execute SharePoint Embedded commands.

Global Administrators can continue to execute SharePoint Embedded container commands.

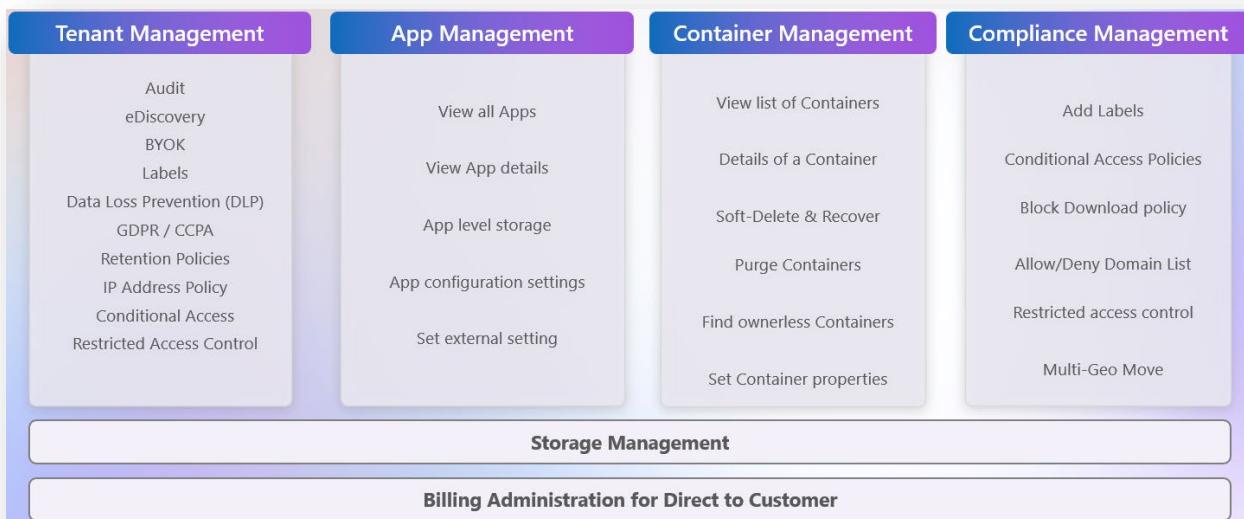
Developers are responsible for the actual solution development and SharePoint Embedded API integration. They are very comfortable with programming practices, patterns and the software development lifecycle.

This diagram helps identify the various activities normally seen in the respective roles:



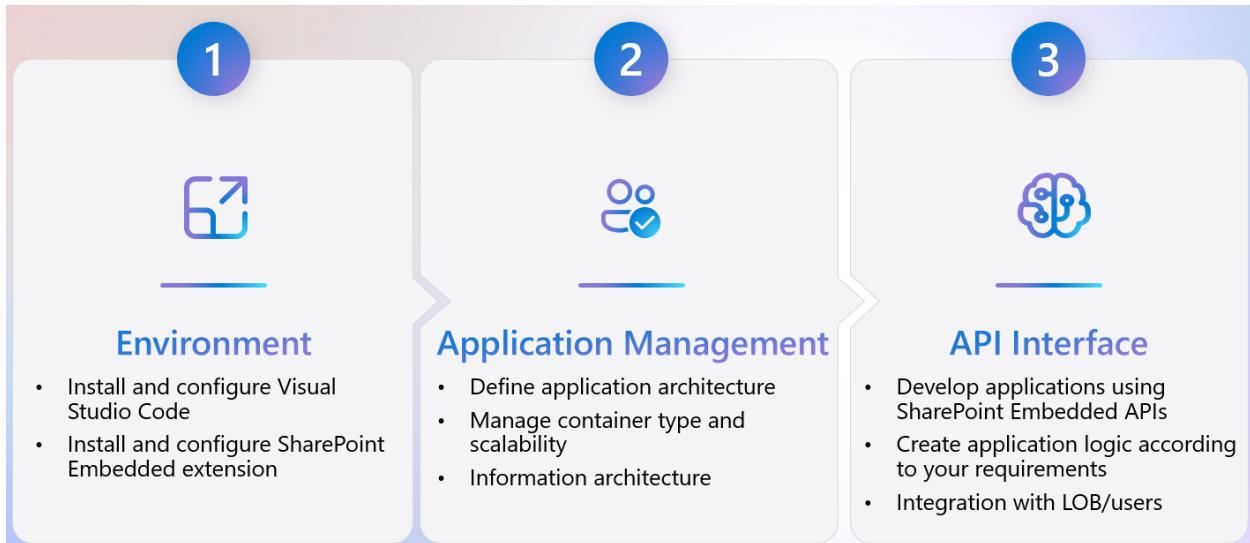
## Administrators

As we focus more on the activities that administrators or responsible for, you'll see that they align with maintaining the integrity of the tenant while working with the developers to provide the necessary components needed to create solutions using SharePoint Embedded.



## Developers

Developers on the other hand are focused on making the solution come alive. Once the administrators have the core environment configured, developers start to build out the application architecture (containers and metadata) and use the SharePoint Embedded API's along with the business logic that controls the user experience.



# PowerShell commands you should know

When it comes to managing your SharePoint Embedded environment, at times you'll need to use PowerShell to adjust environment settings, automate some tasks or simply get the landscape of container types and settings.

## When to use this

These set of PowerShell commands are my go to when I want to better understand how things are configured or manage what has been configured to meet organizational or governance requirements.

## Helpful Commands

### *Getting started*

Import the modules needed to connect to your environment

Command
Import-Module "Microsoft.Online.SharePoint.PowerShell"
Get-Module "Microsoft.Online.SharePoint.PowerShell"   Format-List
Connect-SPOService -Url "https://<domain>-admin.sharepoint.com/"

### *Enabling Azure B2B integration*

This integration enables secure external sharing of files and resources between Microsoft SharePoint, OneDrive, and external users via Microsoft Entra B2B.

Command
Set-SPOTenant -EnableAzureADB2BIntegration \$true
#Check the tenant settings.
Get-SPOTenant

### *List all the Container Types (Applications) in the tenant*

Useful when you want to get a simple list of your SPE Applications, specifically the OwningApplicationId that's needed for many more commands.

<b>Command</b>
Get-SPOApplication   Format-List

*Get the owning application details*

For the specified OwningApplicationId, this helps determine the override for the tenant sharing capability.

<b>Command</b>
Get-SPOContainerType   Format-List

<b>Output</b>	<b>Output value</b>
<b>RunspaceId</b>	<ID>
<b>OwningApplicationId</b>	<ID>
<b>OwningApplicationName</b>	{<ID>}
<b>Applications</b>	{<ID>, <ID>}
<b>SharingCapability</b>	ExternalUserAndGuestSharing
<b>OverrideTenantSharingCapability</b>	True

*Remove a container type*

At times it's necessary to remove unwanted container types. This activity does send the deleted container type to the recycle bin for 93 days.

<b>Command</b>
Remove-SPOContainerType -ContainerTypeId <ContainerTypeId>

### *Modify an existing container type*

As changes happen to the environment, you can easily update them.

<b>Command</b>
Set-SPOContainerType -ContainerTypeName <ContainerTypeName> -ContainerTypeId <ContainerTypeId> -OwningApplicationId <OwningApplicationId> -AzureSubscriptionId -ResourceGroup <ResourceGroup>

### *Get the Container Type Configuration details*

Container Types can have specific settings to meet organizational needs. This command displays those settings

<b>Command</b>
Get-SPOContainerTypeConfiguration -ContainerTypeId '<ContainerTypeId>'

### *Create a new Container Type*

Before you can register the container with SharePoint, it must first be created. You will need to create the Azure Application before running this command.

<b>Command</b>
New-SPOContainerType -ContainerTypeName "<ContainerName>" -OwningApplicationId "<OwningApplicationId>" -AzureSubscriptionId "<AzureSubscription>" -ResourceGroup "<ResourceGroup>" -Region "EastUS"

### *Register a Container Type*

Once the application is registered, you may need a token to make additional changes, such as registering a consuming tenant. These commands will help with the certificate and authorization needed.

<b>Command</b>
Install-Module -Name MSAL.PS \$tenantId = "<ConsumingTenantID>" #Consuming Tenant ID \$clientId = "<ConsumingTenantClientID>" #Consuming Tenant Client ID  \$certPath = "C:\certs\certificate.pfx" \$certPassword = ConvertTo-SecureString -String "<CertPassword>" -Force -AsPlainText

```
$cert = New-Object  
System.Security.Cryptography.X509Certificates.X509Certificate2($certPath,  
$certPassword,  
[System.Security.Cryptography.X509Certificates.X509KeyStorageFlags]::Exportable)  
  
$scope = "https://<Domain>.sharepoint.com/.default"  
  
$token = Get-MsalToken -TenantId $tenantId -ClientId $clientId -ClientCertificate $cert -  
Scope $scope  
  
#Register the container on the consuming tenant:  
  
Connect-SPOSERVICE -Url "https://<Domain>-admin.sharepoint.com/"  
  
#Get the registration script in GitHub  
.\\RegisterContainer.ps1 -ClientId "<ClientID>" -ContainerTypeID "<ContainerTypeID>" -  
PemCertificationFilePath "./certs/SPEDemo.key" -ConsumerTenantId  
"<ConsumerTenantID>" -ConsumerTenantUrl  
"https://ConsumerDomain>.sharepoint.com" -Thumbprint "<Thumbprint>"
```

# Container Architecture - what's right for me

Container architecture is probably one of the most critical decisions you need to make when creating a solution. Knowing the type of content, security boundaries and sharing requirements all factors into this decision.

## When to use this

Before a line of code is written or configured, having a deep understanding of the use case for the solution will guide you to the proper container architecture. Some questions you need to answer are:

- **Organization** - depending on your use case, containers are a great way to organize like documents. It can be by client, project or strategic initiative.
- **Security** – will there be both internal and external access to the content? Remember that a container acts as a security boundary for your application.
- **Sensitive information** – To prevent accidents from happening, isolate sensitive information in its own container.
- **Traceability** – If the content will be included in an eDiscovery case, Copilot (agent) reasoning or subject to specific DLP policies, the container again provides this level of separation.
- Don't be afraid to **create** a container, they scale quite well.

## Scalable Architectures

SharePoint Embedded was [designed to scale](#) to handle some of your largest implementations. Whether your enterprise is looking to optimize internal processes or an ISV selling your solution in the Azure Marketplace, here are some architectures to consider:

# Scalable architectures for your applications

## Enterprise Apps

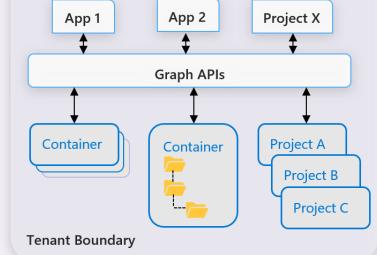
An internal application intended to support the business requirements.



Generally, this will contain documents that support an individual business process. If there are additional applications, separate container types will be created.

Typically, all users are inside the tenant boundaries

- Projects – each container would represent a project
- Claims – 1 container would be used to process claims



## External Collaboration

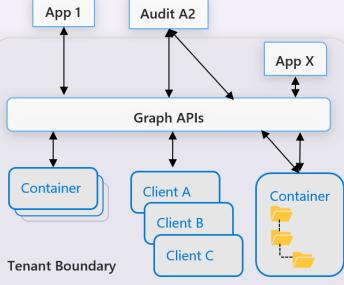


Enterprise customers who are creating solutions to be used by their customers.

Generally, the core business focuses on creating or maintaining specific IP that is sold to their customers. Applications are then created to enhance the consumption of that IP into a customer's ecosystem.

Users are in their own external tenant  
Data is in the tenant boundary

- Audit or tax document gathering
- Customer portal to retrieve documents

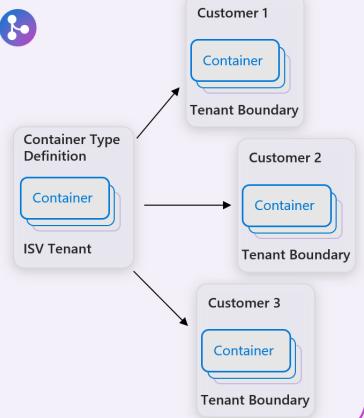


## ISV



ISV products intended to be developed and sold in the Azure marketplace or directly to customers.

Data will reside in their customers' environments; however the billing is a pass through to the ISV, who will do the invoicing.



# Permissions 101

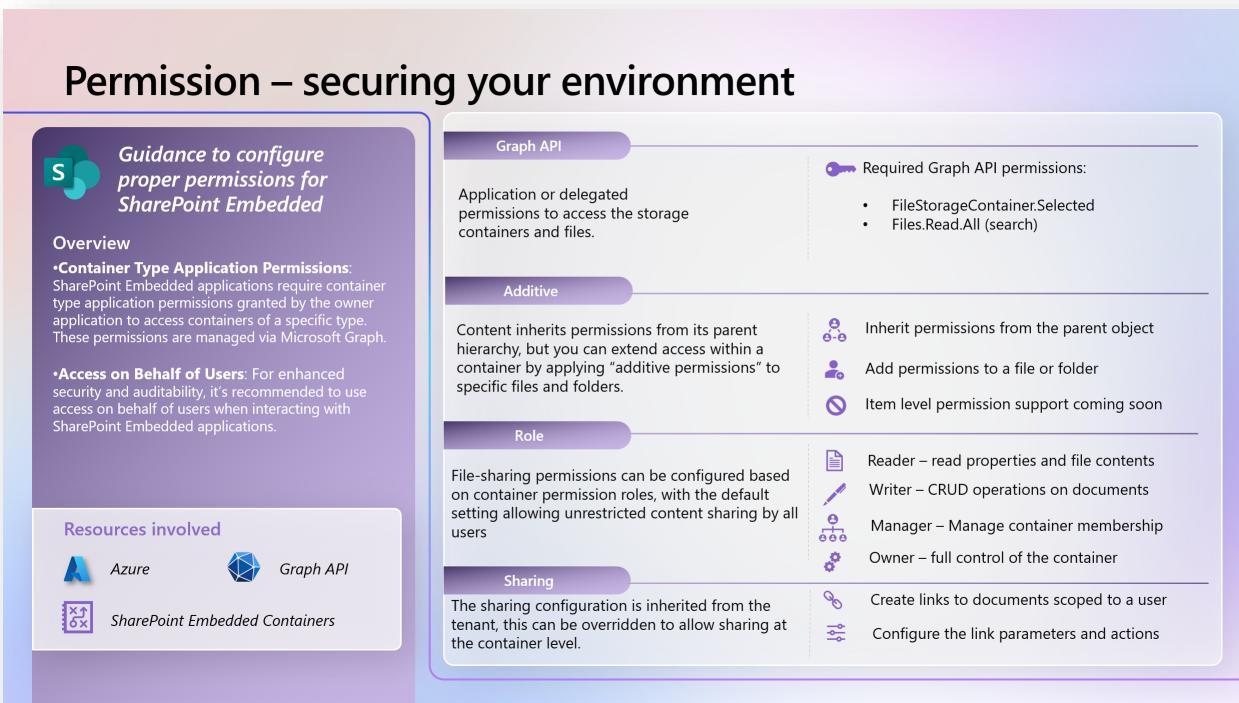
SharePoint Embedded currently implements **additive permissions**, which allow extending access to specific files and folders within a container, and **role-based sharing settings**, offering both restrictive and open sharing models. The default sharing configuration aligns with the consuming tenant's settings, but can be overridden at the application level to allow more flexible sharing options.

## When to use this

As you are considering how to implement security into your applications and content, understanding how the permissions model in SharePoint Embedded is critical. External access, sensitive content and role-based access are just some of the things that need to be considered.

## Permissions Explained

### Overview



### *Container Type*

A Container Type is simply an object that will hold containers, which is where permissions are assigned.

### *Container and item Permissions*

The Container architecture is a driving force behind how permissions can be applied. As users are added to each container, keep the following in mind to ensure it's secured properly:

- **Additive permissions** allow you to extend access to specific files and folders within a container without altering the overall container permissions. This means you can grant additional permissions to certain items as needed, providing more granular control over who can access specific content within the container.

For example, if a container has general read-only access, you can use additive permissions to give a particular user edit access to a specific file within that container without changing the read-only status for the rest of the container. This flexibility helps in managing access more precisely and securely.

**Note** that when you add the same person to the same role more than once, they will retain the initial PermissionId.

- **Restrictive permissions** (a feature that is coming soon) allows for granular control over access to individual items within a container. Users can be restricted to only read or edit items they have created, or be granted broader permissions to interact with all items. These settings are configured at the item level, providing options for read and create/edit access. This feature is particularly useful for scenarios like submission forms or helpdesk systems, where users need to interact with their own data while maintaining overall container security.
- **[Microsoft Graph Data Connect](#)** – recently introduced is the ability use MGDC datasets to view the permissions on SharePoint Embedded containers and files. MGDC for SharePoint will show all permissions granted in SharePoint Embedded containers, including permissions granted at various levels of the SharePoint hierarchy (site, web, library, folder, or file) and diverse types of security principals (users and groups, internal or external).

# Configuration Steps for SharePoint Embedded with a Consuming Tenant

Register File Storage Container Type Application Permissions

## When to use it

Typically, when an ISV's creates an application that is sold to customers it is encouraged to use this approach because:

- A Container Type is created in the customers tenant, isolating the documents and using their user and governance controls.
- Billing can be configured so that the ISV can view the customers' usage and invoice them for the consumption used.
- The ISV only needs to create 1 Container Type, not one for each customer.

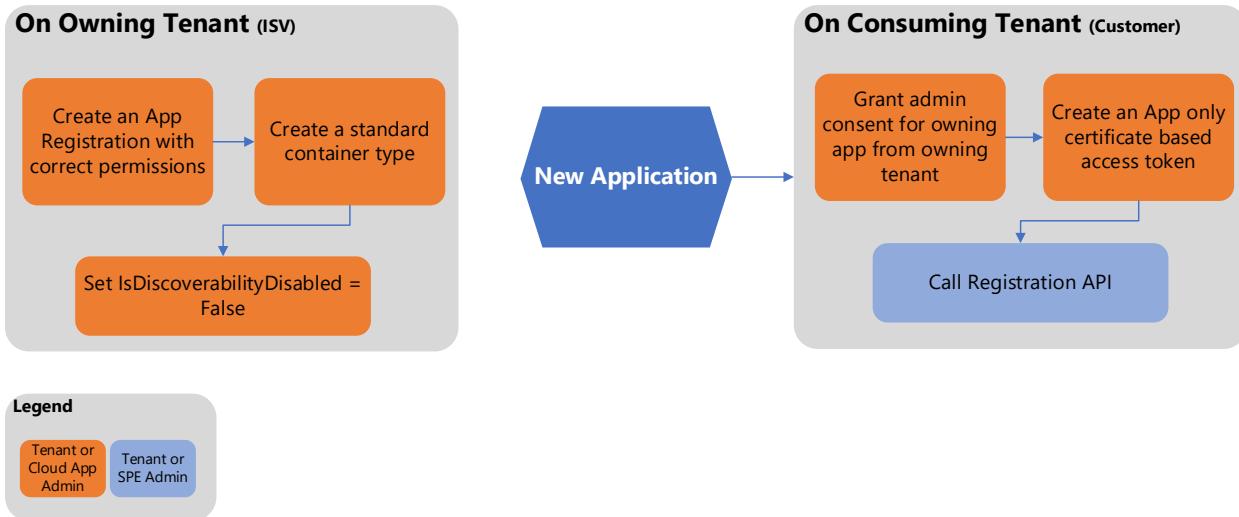
## Dependencies

Before configuring SharePoint Embedded with a consuming tenant, ensure the following dependencies are met:

- Valid SharePoint Online subscription
- Access to the SharePoint administration center
- Required permissions to register and configure application permissions
- Certificate used for acquiring a security token
- Knowledge of the registration API

## Process Overview

Here are the main activities for registering a container with a new consuming tenant.



## Configuration Steps

### 1. Enable SharePoint Embedded (optional step if direct to customer billing is required)

In the M365 Admin center, navigate to Setup -> Automate content processes with Syntex -> go to Syntex Settings -> Apps -> SharePoint Embedded

The screenshot shows the Microsoft 365 Admin Center navigation bar on the left and the Syntex settings page on the right.

**Syntex Services for:**

- Documents & images
- Videos
- Storage
- Apps

**Manage pay-as-you-go billing:**

An Azure subscription is linked to Microsoft Syntex.

**Apps:**

Service ↑	Description
SharePoint Embedded	Allow users to use apps built with SharePoint Embedded.

## 2. Register the Container Type on the consuming tenant

### Grant admin consent

To interact with containers in a consuming tenant, the owning tenant must first be granted permissions by the consuming tenant admin. This step is crucial as it controls permissions and prevents access denied errors when invoking other APIs.

The following URL should be executed by the **consuming** tenant administrator. This will grant permission to the **owning** tenant App Registration.

#### URL

[https://login.microsoftonline.com/<ConsumingTenantID>/adminconsent?client\\_id=<OwningTenantClientID>](https://login.microsoftonline.com/<ConsumingTenantID>/adminconsent?client_id=<OwningTenantClientID>)

- Permission for the consuming tenant will be displayed to review and accept. (ISV will be granted permission on their customer tenant)

### Generate Access Token

To register an owning app on a consuming tenant, an App Only certificate based token will need to be created enabling the owning application to execute the supported PowerShell commands.

To create an AppOnly and certificate-based access token, you'll need to follow these steps:

**1. Register an Application in Azure AD:**

- Go to the Azure portal and navigate to **Azure Active Directory > App registrations > New registration**.
- Retain the **Application (client) ID** to use it in later steps

**2. Generate a Self-Signed Certificate:**

In addition to using an existing certificate, you can also create a self-signed cert.

- You can create a self-signed X.509 certificate using tools like PowerShell or OpenSSL. For example, in PowerShell, you can use the `New-SelfSignedCertificate` cmdlet:

```
$cert = New-SelfSignedCertificate -Subject "CN=YourAppName" -  
CertStoreLocation "Cert:\CurrentUser\My"
```

**3. Upload the Certificate to Azure AD:**

- a. In the Azure portal, go to your registered application.
- b. Navigate to **Certificates & secrets > Certificates > Upload certificate**.

**4. Configure API Permissions:**

- a. Go to **API permissions** in your application settings.
- b. Add the Container.Selected Application permission.
- c. Grant admin consent for these permissions.

**5. Acquire the Access Token and register the Container Type:** This example [PowerShell script](#) in GitHub, provides an easy way to get the access token and register the container type on the consuming tenant.

## References

- [App-only authentication in Exchange Online PowerShell and Security & Compliance PowerShell | Microsoft Learn](#)
- [SharePoint Embedded Authentication and Authorization | Microsoft Learn](#)
- [Grant tenant-wide admin consent to an application - Microsoft Entra ID | Microsoft Learn](#)
- [Microsoft identity platform admin consent protocols - Microsoft identity platform | Microsoft Learn](#)

# How do ACL permissions work

## Overview

This is an examination of how Access Control Lists (ACLs) are used to manage permissions within SharePoint Embedded (SPE) environments. Key concepts include permission scopes, ACLs, Access Control Entries (ACEs), principals (users, AD groups, SPGroups), and the directory service Entra. In SPE, ACLs define permissions for principals through ACEs, and breaking inheritance on files or folders creates new permission scopes.

## Glossary

We will utilize the following standard terminology.

- **Permission Scope** - is a uniquely secured object. A uniquely secured object is one whose ACL may be set differently than its parent. Multiple files can have the same permission scope.
- **ACL** - an Access Control List. Its composed of a set of ACEs.
- **ACE** - is an Access Control Entry. In SPE, an ACE is a principal and a rights mask (read, read/write)
- **Principal** - is a user, an AD group or an SPGroup. If a group is present in an ACE in an ACL, all members of that group are granted the permission level specified by the ACE.
- **SPGroup:** SharePoint Group, a collection of users.
- **Entra:** A directory service used for user references, formerly known as Azure Active Directory..

# How ACLs work on SharePoint Embedded objects

In SharePoint Embedded (SPE), ACLs are used to manage permissions for objects (files and folders). Each ACL consists of multiple ACEs, which specify the permissions for a principal. When a container is created, it has a single permission scope, meaning all files and folders inherit their ACL from the root of the container. Breaking inheritance on a file or folder creates a new permission scope, increasing the scope count.

Key points:

- **Permission Scope:** A uniquely secured object with a different ACL than its parent.
- **Breaking inheritance** on a folder or file increases the scope count.
- **Folders** can only have their inheritance broken if they contain 100K or fewer items.
- There is a **recommended limit** of 5K unique permission scopes per container to avoid performance issues.

## Examples

### Example 1: Creating a Container with Minimal Scopes

- Create a container with a single permission scope.
- Add a folder and share it, breaking inheritance and creating a new scope.
- Upload 75K files to the folder, all inheriting the folder's ACL.
- Result: The container has 2 unique scopes (root and folder).

### Example 2: Managing Permissions for multiple folders

- If you want to map categories to folders, creating 20 folders per container.
- Each folder will have unique permissions, resulting in 21 scopes (root + 20 folders).
- Individual files with unique permissions will be minimized to stay within the 5K limit.

### **Example 3: Sharing files**

Sharing two files located in a folder with user X will use 3 scopes: 1 scope for the parent folder and 1 scope for each individual file. Even though both files are shared with user X and have the same ACL, breaking inheritance from the parent folder creates unique scopes.

Therefore, if you have 10,000 files to share with user X and share each one individually, it will require 10,000 additional scopes. However, if you place all the files into a single folder, share that folder with user X, and then move the files into the folder, it will only require 1 additional scope for the folder.

One significant point is co-authoring. If a user is working on a file in Word and wants to co-author with someone, they can still invite them if the file is in SPE. Doing so will use one of the ACLs remaining invitations available. This is an additional factor to consider.

### **Example 4: ACL clarification**

Given the following statement:

*"Does 5k limit for ACL mean that we can have only 5k files/folders having ACL? Or does that mean we can have 5k unique ACLs? So, if two files have identical ACL does both count?"*

- The correct interpretation is "we can have 5K unique ACLs".
- If two files have the same ACL because they both inherit from the parent folder, then they share an ACL. Only the parent folder counts.

### **Example 5: Moving files**

In this scenario, consider that 10,000 files have been shared with an individual on a one-by-one basis (a hypothetical extreme). If these files are subsequently moved into a single folder, does the system consolidate the permission entries so that only the folder's permissions count, or do we face a cluttered permissions landscape unless the user manually rectifies it?

Automatic restoration of permission hierarchy is not implemented due to security concerns. In the provided example, restoring the permission hierarchy would result in differing access pre- and post-move. Initially, the recipient can view 10,000 files individually shared with them. After cleanup, they can see those 10,000 files along with the containing folder, making it non-equivalent.

This scenario implies that if extensive sharing occurs from one's OneDrive without manual cleanup, it might lead to performance degradation, or even operational failure once the limit of 50,000 items is reached—assuming sharing is synonymous with permission settings.

However, judicious management whereby files are organized into folders when ACLs match could mitigate these issues. It is important to note that exposing the actual

folder hierarchy to users may be undesirable, especially when utilizing metadata to represent everything instead of traditional folders.

## Guidance

To effectively manage ACLs in SPE, follow these guidelines:

- Minimize the number of unique permission scopes by leveraging inheritance.
- Share folders instead of individual files to reduce the number of unique scopes.
- Ensure folders that need unique permissions are shared before they exceed 100K items.
- Regularly review and clean up permissions to maintain performance.
- Use SPGroups and Entra for managing user references and permissions efficiently.
- The 5K limit is a soft limit, similar to a speed limit. Actual performance may vary based on other factors such as the size of the ACLs.
- With small ACLs (fewer than 10 ACEs in an ACL), you can exceed 5K scopes without issues. With maximum ACLs (5K ACEs), performance issues may arise before reaching 5K scopes. Therefore, we recommend a 5K scope limit as a safe guideline for most scenarios.

# Traversing the file tree efficiently

SharePoint Embedded is all about files and the ability to not only store them but work with them. The API makes it easy to determine where you are in the file structure so you can display items to your users.

## When should I use this

Use this approach when you want to present files and folders to users ... IE walking the folder structure.

## Get the root files

The first thing we need to do is get all the files on the root of the container:

Method	URL
GET	<a href="https://graph.microsoft.com/v1.0/drives/&lt;ContainerId&gt;/items/root/children?&amp;\$expand=listitem(\$expand=fields)"><code>https://graph.microsoft.com/v1.0/drives/&lt;ContainerId&gt;/items/root/children ?&amp;\$expand=listitem(\$expand=fields)</code></a>

Response (abbreviated)

```
"value": [
  {
    "createdDateTime": "2024-09-23T13:44:40Z",
    "eTag": "\"{1B0940B0-C43E-4380-B4C6-144512655797},2\"",
    "id": "01UELPCRFQIAERWPWEQBB3JRQUIJGKV4X",
    "lastModifiedDateTime": "2024-09-23T13:44:40Z",
    "name": "ProjectFalcon",
    "webUrl":
    "https://<Domain>.sharepoint.com/contentstorage/<ContainerGUID>/Document%20Library/ProjectFalcon",
    "cTag": "\"c:{1B0940B0-C43E-4380-B4C6-144512655797},0\"",
    "size": 3462953,
  ...
  },
  {
    "listItem": {
      ...
    }
  }
]
```

```

    "id": "0x012000ED57DD013E380E4D978C7A98CD149091",
    "name": "Folder"
},
...

```

Notes:

- The response is going to list all the files that are at the root of the container.
- This will include folders and items. You can use the ContentType to determine if it's a "Folder" or "Document"
- If the content type is a folder, save the name of the folder, we'll need it shortly.
- If the content type is a document, use the WebUrl value as a hyperlink to open the document.

## Navigating into a folder

Using the name of the folder in the prior output, concatenate that to the following request to get the contents of that folder

Method	URL
GET	"https://graph.microsoft.com/v1.0/drives/<ContainerId>/root:/StatusReports/20241115Status/SubFolder1:/children?&\$expand=listitem(\$expand=fields)

Response (abbreviated)

```

...
"parentReference": {
    "driveType": "other",
    "driveId": "<ContainerId>",
    "id": "01UELPCREV5UXIGQDHXZFKGWGLRPKFEB03",
    "name": "SubFolder1",
    "path": "/drives/<ContainerId>/root:/StatusReports/20241115Status/SubFolder1",
    "siteId": "f96ae408-28af-41ad-88ad-ddffcfdb75fe"
},
...

```

Notes:

- Very similar to getting files at the root level only appending the folder name to the request and you'll get all the items for that folder in the response.
- Using the "path" value and the name of the folder on the item, you can then use that to get the contents of the nested folder.

## Going back up a folder

Navigating back up the structure focuses on getting the values of the parentReference response output from the listItem node and making the same Graph calls to view the files.

Method	URL
GET	<a href="https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items/70217232-a596-45d9-8f82-2ba695cf5de2&gt;/children?&amp;expand=listItem(\$expand=fields)"><code>https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items/70217232-a596-45d9-8f82-2ba695cf5de2&gt;/children?&amp;expand=listItem(\$expand=fields)</code></a>

### Response (abbreviated)

```
...
"listItem": {
    },
    "parentReference": {
        "id": "70217232-a596-45d9-8f82-2ba695cf5de2",
        "siteId": "<domain>.sharepoint.com,f96ae408-28af-41ad-88ad-
ddffcfdb75fe,2705c2f8-33aa-4a72-8438-25114dc3203b"
    },
...
}
```

Notes:

- You can use the listItem -> id attribute to move up the tree to the parent folder. Make sure you don't accidentally use the parentReference in the value node of the JSON response.
- This will return all the items in that folder until you get to the root again.

# The importance of Information Architecture

Good information architecture (IA) is a little art and a little science. Knowing your content and how it will be used in your application is a fundamental activity that you have to do.

## When to use it

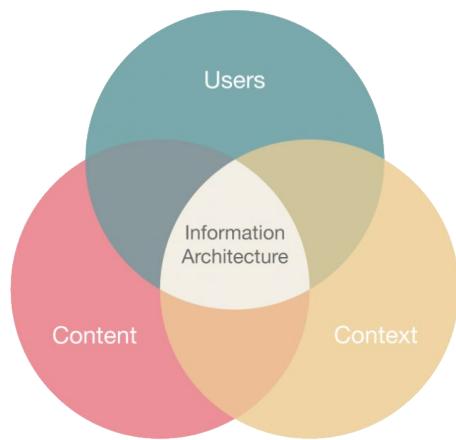
Always!

## Plan it out

Like they say, “failure to plan is planning to fail”. Planning a good IA isn’t a one and done activity. It will evolve and change as your application and user demands change. Having a master plan and keeping it updated avoids drift and duplication. To help with this, use a tool such as a mind map, Visio or even Excel. There are industry specific IA schema’s available and a great place to start.

Also get in the right mindset. I often think of myself as a librarian. Imagine that you get a truck load of books on the loading dock, and you must organize them so they can be easily found and not just randomly thrown on shelves. This activity will force you to really think about all the different ways you, and more importantly your users, will find what they’re looking for.

I often find myself re-grounding on this Ven diagram to make sure I’m as balanced as I can be.



## Where to start

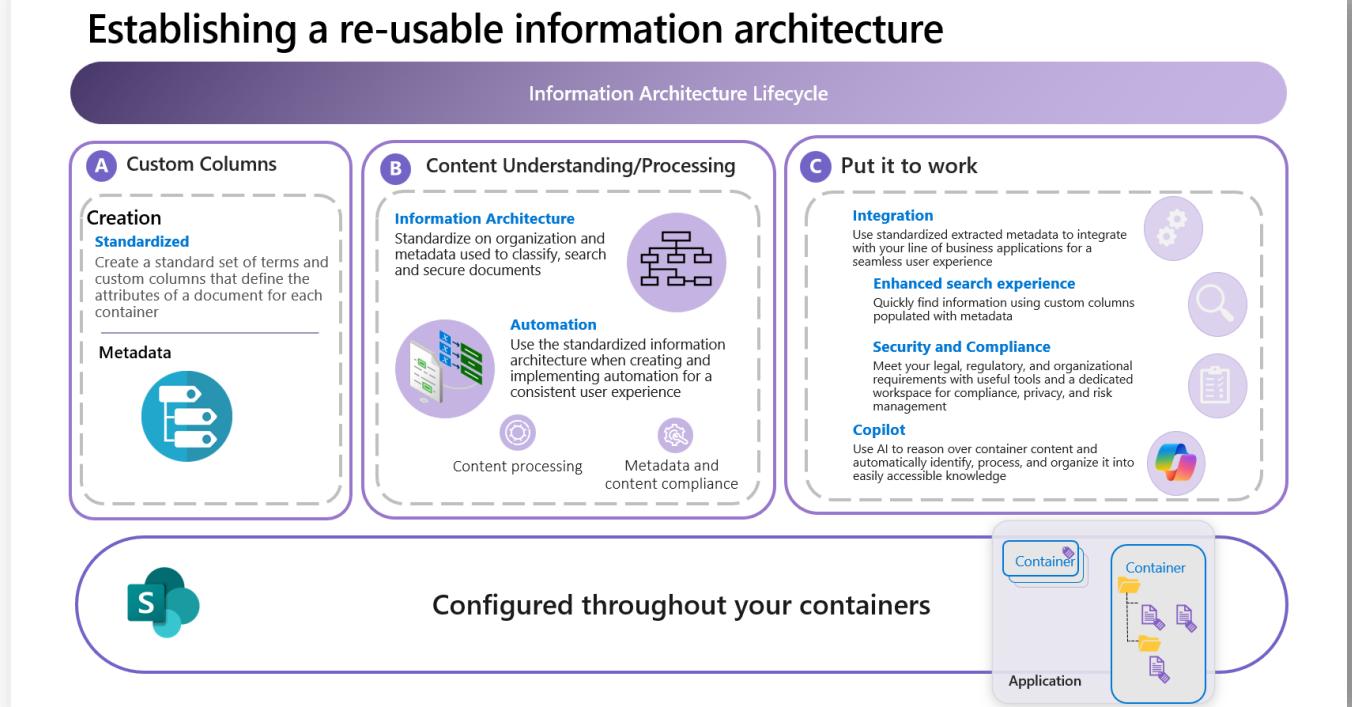
Here are a few things to get into the IA mindset:

- Folders are a poor person’s way of organizing information. Try to break yourself from this habit and identify how you really want to find a document.

- Consider the keywords and terms users will use to search for the document, including common terminology and jargon relevant to the content and audience. Make it real and useful for your organization.
- Think about the sorting and filtering criteria most useful to users, ensuring metadata includes attributes like date of creation, author, document type, and relevant tags.
- Factor in the requirements of automated business processes that will interact with your documents, ensuring compatibility with systems that automate sorting, filing, and retrieval tasks.
- Meticulously plan and implement metadata to create an efficient, user-friendly, and adaptable information architecture that evolves with the demands of your application and users.

As you apply this to the SharePoint Embedded container architecture, you can use this high-level flow to remind yourself what should be included in your thought process as you establish, implement and maintain a good IA.

## Establishing a re-usable information architecture



# Add Metadata to a file

Metadata is data about data. I always found this a little confusing but it's basically attributes that describe a document. This can include details such as the author, date created, date modified, file size, and more, which help in organizing, finding, and understanding the data within the document.

## When to use it

When documents are added to a container, that container can be configured with customer columns (metadata) to further classify that document. The metadata is added to the search index, allowing you to easily narrow down results.

SharePoint Embedded supports metadata but here are some things to consider:

- Does it need to be real time – When metadata is added to a document in a container, the value will get added to the search index. There can be a delay (typically up to 15-30 minutes) before this is indexed and available for searching.
- Relational data – considering metadata can resemble a well thought out relational database in SQL Server or Dataverse, it's not. The ability to do transactions, referential integrity and views that you would normally do in a relational database is not possible using metadata. If you have an application that needs this functionality, data stores are the best approach.
- Support complex data types – SharePoint embedded supports simple data types as we'll see shortly. Complex metadata such as managed are not supported.

It's a good idea to have a well thought out information architecture that uses metadata across containers to meet your applications use case.

## Creating custom columns to hold metadata

[Custom columns](#) are created at the container level and can be assigned to documents in that container.

Method	URL
POST	<a href="https://graph.microsoft.com/beta/storage/fileStorage/containers/{{ContainerID}}/columns">https://graph.microsoft.com/beta/storage/fileStorage/containers/{{ContainerID}}/columns</a>

## Request Body

```
{
  "name": "curCurrency",
  "displayName": "curCurrency",
  "description": "Currency Column",
  "enforceUniqueValues": false, // Must be false (true not supported with Containers)
  "hidden": false,
  "indexed": true, // Set to true to be able to search files based on this column
  "currency": {
    "allowMultipleLines": false,
    "appendChangesToExistingText": false,
    "linesForEditing": 0,
    "maxLength": 255
  }
  /*
   * Other supported column types with SharePoint Embedded -- replace the text: {..}
   * property above with one of these:
   * If a column type is not listed here, it's NOT supported with SharePoint Embedded
   * right now.
   */

  "boolean": {
  }
  "dateTime": {
    "displayAs": "default | friendly | standard",
    "format": "dateOnly | dateTime"
  }
  "currency": {
    "locale": "en-us"
  }
  "choice": {
    "allowTextEntry": true,
    "choices": ["red", "blue", "green"],
    "displayAs": "checkBoxes | dropDownMenu | radioButtons"
  }
  "hyperlinkOrPicture": {
    "isPicture": false
  }
  "number": {
    "decimalPlaces": "automatic | none | one | two | three | four | five",
    "displayAs": "number | percentage",
  }
}
```

```
        "maximum": 10.551,  
        "minimum": 99.993  
    }  
    "personOrGroup": {  
        "allowMultipleSelection": true,  
        "displayAs": "account | contentType | created | department | ...",  
        "chooseFromType": "peopleAndGroups | peopleOnly"  
    }  
*/  
}
```

## Assign metadata to a file

Method	URL
PATCH	<a href="https://graph.microsoft.com/beta/drives/{{ContainerID}}/items/&lt;FileID&gt;/listitem/fields">https://graph.microsoft.com/beta/drives/{{ContainerID}}/items/&lt;FileID&gt;/listitem/fields</a>

### Request Body

```
{  
    "<CustomColumnName>:<Value>"  
}
```

# Extract metadata using Document Intelligence Studio

Document Intelligence Studio is an advanced tool designed to assist users in extracting key information from various documents and storing this extracted data as metadata within SharePoint Embedded containers for that document. Its API-based architecture facilitates seamless integration into custom applications using familiar JSON and endpoint patterns you're using with SharePoint Embedded.

## When to use it

Once you have a solid information architecture and custom columns created on your containers, it's time to automate the extraction of metadata from documents you receive. Users will quickly determine that this laborious and tedious task is too time consuming to sustain, and it will fall out of favor fast.

## What models are available

Document Intelligence Studio offers a variety of models to analyze and extract information from documents. Here's a summary of the available models:

**Document analysis:** Extract text, tables, structure, key-value pairs, and named entities from documents.

- **OCR/Read:** Extract printed and handwritten text, barcodes, formulas, and font styles from images and documents.
- **Layout:** Extract tables, check boxes, and text from forms and documents.
- **General documents:** Extract key-value pairs and structure like tables and selection marks from any form or document.

### Prebuilt Models:

- **US Mortgage Documents:** Analyzes the Closing Disclosure form used in mortgage transactions.
- **Marriage Certificate:** Extracts pertinent details from marriage certificates.
- **Credit Card:** Extracts information from credit cards.
- **Business Card:** Extracts contact information from business cards.

## Custom models:

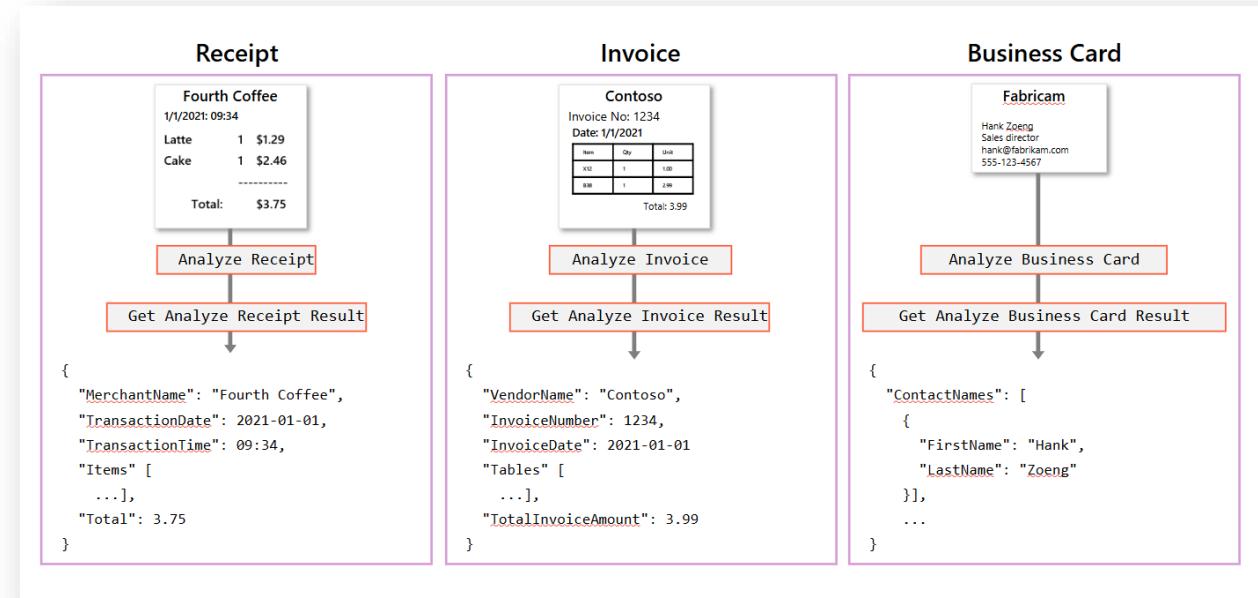
- **Custom extraction model:** Label and build a custom model to extract a specific schema from your forms and documents.
- **Custom classification model:** Build a custom classification model to split and classify documents.

These models can help you automate data extraction and processing tasks for various document types, making your workflows more efficient.

## How it works

Once you have your [environment configured](#), the Graph endpoints will be available to your application, allowing you to post the document to the correct model, parse the JSON response and populate the metadata for the document in the respective containers.

Take some of the pre-built models for example:



You can see that depending on model you're using to process the document will return the extracted information, ready for parsing to use in the business logic of your application, no training necessary.

# Configure your tenant to collaborate with external users

In SharePoint Embedded, content inherits permissions from its parent. Although this structure can't be changed, you can grant extra permissions within a container. For example, if UserA is a Reader, you can use Microsoft Graph to give them edit access to a specific document.

We'll review the key configurations to ensure your environment is secure but still allows external users to collaborate on your documents.

## When to use it

If your application has a requirement to collaborate externally, you want to make sure your doing this in a secure and controlled way without over exposing content.

## Open Sharing Model

SharePoint Embedded features a role-based sharing model, allowing developers to set file-sharing permissions based on container roles. Users can choose between restrictive and open sharing, with the default being open, allowing unrestricted content sharing. This setting is part of the container type configuration and can only be adjusted by the application owner's developers.

When you want to restrict this functionality, run the following PowerShell command and it will be disabled, only allowing owner or manager roles can add new permissions.

```
Set-SPOcontainerTypeConfiguration  
-containerTypeID <containerTypeID>  
-sharingRestricted $false
```

## Inherits tenant settings

By default, when a new Container Type (Application) is created, it will inherit the sharing settings you have in the SharePoint admin center. This follows the most restrictive model approach to ensure tenant governance is followed.

SharePoint Embedded does provide the ability to override this at the Container Type (Application) level.

**Note:** this override only pertains to the specified Container Type and does not affect any other sharing settings in SharePoint or other SharePoint Embedded Container Types

The following PowerShell command will enable or disable sharing on the Container Type

```
Set-SPOApplication  
-OwningApplicationID <OwningApplicationId>  
-OverrideTenantSharingCapability $true  
-SharingCapability <SharingCapability>
```

## Enabling Azure B2B

Microsoft Entra B2B (Business-to-Business) is a feature of Microsoft Entra External ID that enables secure collaboration with external partners. It allows guest users to access your organization's resources using their own work, school, or social identities, without needing to create new accounts. This simplifies identity management and enhances security by leveraging the partner's existing identity management solution.

Enabling this on the tenant requires the following PowerShell command:

```
Set-SPOTenant -EnableAzureADB2BIntegration $true
```

## App Registration Account Type

When registering an application in Microsoft Entra ID, you can choose from four supported account types, which determine who can access your application, especially in the context of sharing with external users. **Single tenant** allows only users and guest accounts within your own directory to access the app, making it ideal for internal use. **Multitenant** enables users from any Microsoft Entra directory to access the app, facilitating collaboration with external organizations.

Ensure that Multitenant is selected.

## Supported account types

Who can use this application or access this API?

- Accounts in this organizational directory only ( [redacted] only - Single tenant)
- Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)

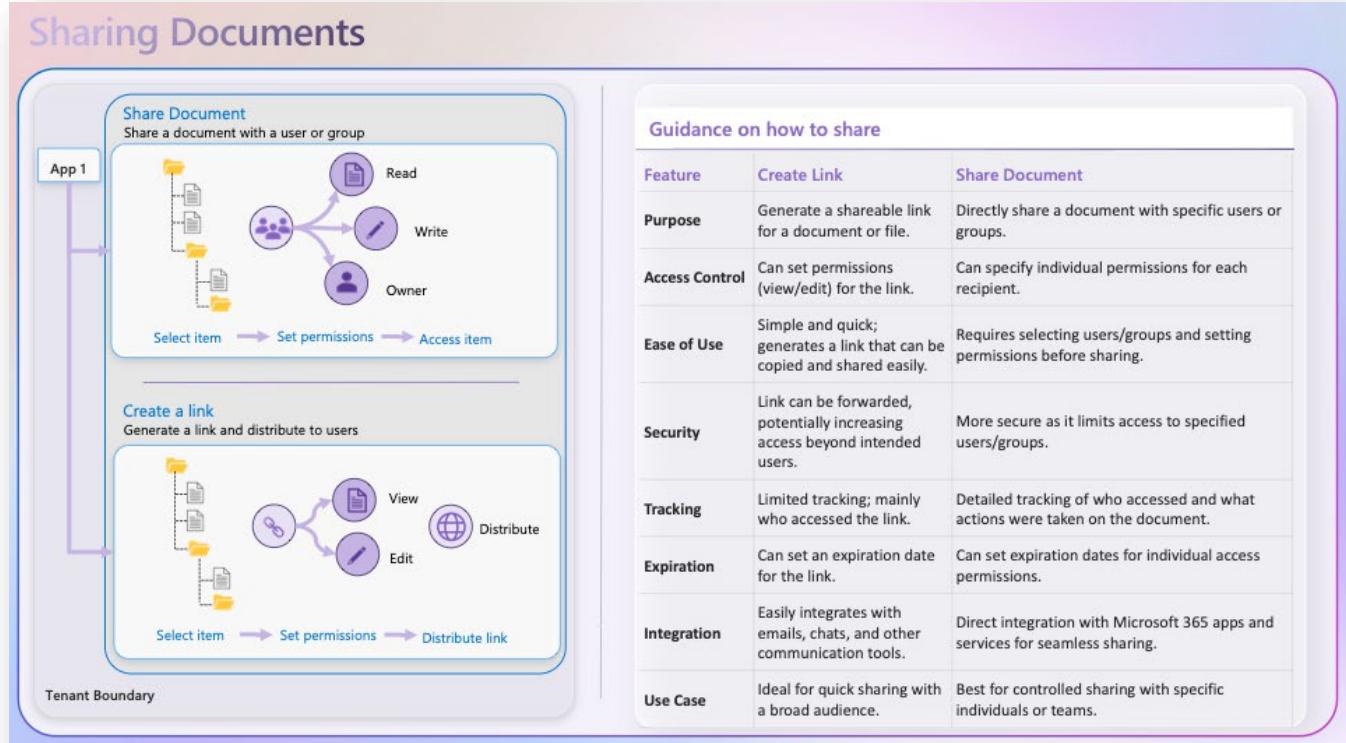
[Help me decide...](#)

# Sharing vs creating a link to a document

In SharePoint Embedded, sharing a document directly is a convenient way to ensure that specific users have access to the document with the permissions you designate, such as view or edit rights. This method allows for a more controlled and secure environment, as you can track who has access and modify permissions as needed. On the other hand, creating a link to a document provides a more flexible and broader approach to sharing. This method is ideal for sharing with larger groups or external parties because anyone with the link can access the document, depending on the link settings. While it offers ease of access, it may pose security concerns if the link is inadvertently shared with unintended recipients.

## When to use it

Depending on the sensitivity of the document and how broad you want to share it, there are several options and configuration settings as described below.



## Preview a file vs collaborating on it

Previewing a file in SharePoint Embedded offers a quick and convenient way to view a document without needing to download it. It allows users to efficiently access and review the content, ensuring that they can rapidly assess the necessary information. On the other hand, the full office collaboration experience in SharePoint Embedded provides a robust set of tools for real-time editing and co-authoring. This seamless integration fosters teamwork by enabling multiple users to work simultaneously on documents (co-author), share insights (comments), and @mention colleagues, significantly enhancing productivity while automatically saving the document back to the SharePoint Embedded Container.

### When to use it

Imagine your user experience simply shows a preview of the document when you hover over a thumbnail, no edits, collaboration, just a preview pop up. Contrast this with the full collaboration experience in Office Online, which is one of the top driving use cases to use SharePoint Embedded.

### What it looks like

Previewing and collaborating happen on a document, which means you will need the id when calling the [preview](#) or [listItems](#) endpoints.

The screenshot shows the SharePoint Embedded - Files interface. On the left, there's a sidebar with navigation links: Overview, Containers, Files (selected), Search, and Copilot. The main area shows a list of files in a folder named 'ProjectFalcon'. One file, 'Project Plan for Project Template.docx', is selected and its preview is displayed in an iFrame on the right. The preview content includes a title 'Project Plan for Project XYZ', a subtitle 'An Outline for Successful Implementation', and sections for 'Introduction', 'Project Scope', 'Objectives', 'Deliverables', 'Project Schedule', and 'Timeline'.

#### Note:

- The `getUrl` attribute in the preview JSON response can be used to display the document in an iFrame or other display means.
- The preview URLs are typically valid for a short duration and are designed to provide secure, temporary access to the content without exposing long-term access tokens.
- When collaborating on a document, the `webUrl` attribute is used. Using this in an iFrame is not supported. It must be opened in a new browser tab.

# Excel file linking

I've always said, Excel runs the world! When you live your life in Excel, the ability to link spreadsheets together is an essential feature, regardless of where the file is stored.

Considering SharePoint Embedded supports collaborating on Office documents, linking to spreadsheets is as easy as knowing where its stored.

## When to use this

Imagine you have a source spreadsheet that contains all your budget items. This is a controlled spreadsheet that only a few folks will have permission to edit. You want others to know about the budget, specifically what they are responsible for. Others can easily incorporate the “master” budget numbers into their own spreadsheets by linking the two together. Doing this allows the people controlling to the budget to update it, and those consuming the budget numbers through links can receive the most current numbers automatically.

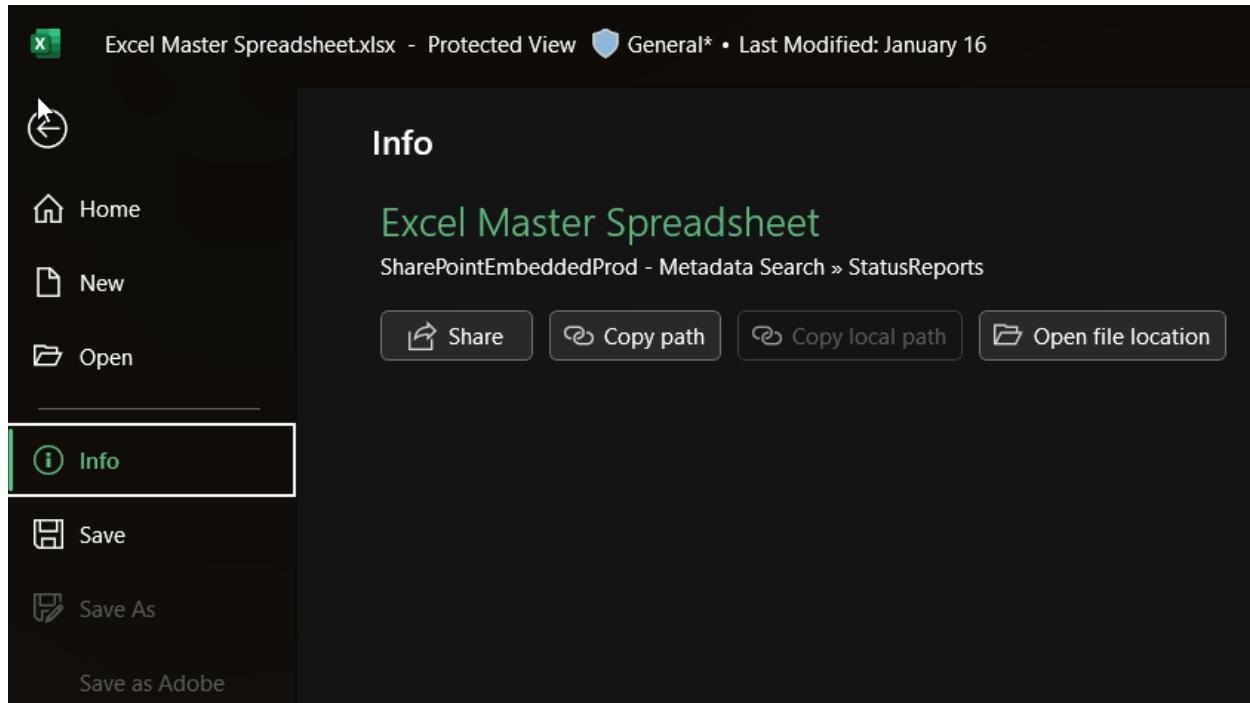
## Linking spreadsheets

Linking spreadsheets is simply a matter of where is the source spreadsheet and what cell do you want to reference. When the spreadsheet is stored in SharePoint Embedded, you can easily get determine where the file is located in Excel itself.

### Get the source spreadsheet location

1. Open the source excel file from SharePoint Embedded through the UI created in desktop mode.  
**NOTE:** Link refreshing is only supported when in Excel Desktop mode.
2. Navigate to File -> Info and click the “Copy path” button. This will get the file location in the SharePoint Embedded container. It should resemble something like this:

[https://<domain>.sharepoint.com/contentstorage/CSP\\_f96ae408-28af-41ad-88ad-ddffcfdb75fe/Document%20Library>StatusReports/Excel%20Master%20Spreadsheet.xlsx?web=1](https://<domain>.sharepoint.com/contentstorage/CSP_f96ae408-28af-41ad-88ad-ddffcfdb75fe/Document%20Library>StatusReports/Excel%20Master%20Spreadsheet.xlsx?web=1)



## Create the link reference in the consuming spreadsheet

Again, open the consuming spreadsheet (where you want to see the linked value) in desktop mode.

1. Select the cell you want to view the linked value.
2. Insert the source spreadsheet location to include the sheet and cell like you normally would. It should resemble this:

The screenshot shows two Excel windows side-by-side. The left window, titled 'Excel Master Spreadsheet.xlsx', contains a table with three columns: 'Value 1', 'Value 2', and 'SUM'. The 'SUM' column has values 32, 24, and 25 respectively for rows 2, 3, and 4. The right window, titled 'Excel Consuming Spreadsheet.xlsx', also contains a table with the same three columns. In cell A7 of the consuming spreadsheet, the formula '= [Excel Master Spreadsheet.xlsx]Sheet1'!\$C\$2' is visible in the formula bar, and the value '32' is displayed in the cell. Both windows show the Home tab selected in the ribbon.

When you open the consuming spreadsheet you can manage and refresh the links normally.

The full URL to the source spreadsheet should look like this:

```
'=https://<Domain>.sharepoint.com/contentstorage/CSP_f96ae408-28af-41ad-88ad-ddffcfdb75fe/Document Library/StatusReports/[Excel Master  
Spreadsheet.xlsx]Sheet1'!$C$2
```

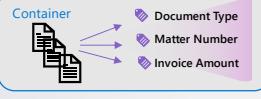
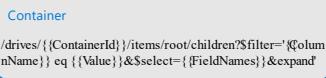
# SharePoint Embedded Search Overview

SharePoint Embedded offers robust search capabilities that enhance productivity and collaboration within organizations. Users can effortlessly locate documents, files, and other content across SharePoint Embedded Containers using intuitive search features. The search functionality indexes content and metadata, ensuring quick and accurate retrieval of information. Additionally, it supports metadata and keyword searches, allowing users to refine their queries and find relevant results efficiently.

A key aspect of SharePoint Embedded's search capabilities is the distinction between container properties, custom columns and Enumeration.

## Find the content you're looking for

Searching and Enumeration options

Container Properties	Custom Columns	Enumeration
<p>Properties can be created at the container level to help organize like containers and execute searches that can be filtered on the properties attached to the container.</p>  <ul style="list-style-type: none"><li>Created at the container level</li><li>Included in the search index (hidden content), expect a delay for the item to be indexed</li><li>KQL syntax can target a specific container or all containers</li><li>entityTypes = drive to search on containers</li><li>New items inherit the container property but not updated when the value changes</li><li>Similar to the SharePoint Site property bag</li></ul>	<p>Commonly referred to as Search, Custom columns (metadata) are added to the SharePoint search index as managed properties.</p>  <ul style="list-style-type: none"><li>Created at the container level and assigned to specific items</li><li>Included in the search index (hidden content), expect a delay for the item to be indexed</li><li>KQL syntax allows targeting of specific items or full text searching within one or all containers that is security trimmed</li><li>entityTypes = drivelitem</li><li>Each item can have a unique value for each custom column</li><li>Currently limited to text columns, additional column types coming soon</li></ul>	<p>Enumerations involve retrieving a collection of items, utilizing querystring parameters such as Filter, orderBy, expand, and others.</p>  <ul style="list-style-type: none"><li>Real time updating – no delay</li><li>URL Querystring parameter driven</li><li>Item results are security trimmed at the user and application level</li><li>Each item can have a unique value for each custom column</li><li>Performance issues when greater than 5,000 items returned, no items are returned</li></ul>

- **Container properties** pertain to the intrinsic attributes of storage containers, which can be utilized to filter and organize search results. These properties function similarly to Site Collection property bag values in SharePoint.
- **Custom columns** are user-defined fields that offer supplementary metadata for items within these containers, similar to metadata fields in SharePoint that are stored in the search index.
- **Enumeration** is a query string approach that will retrieve items in real time. You have filter, sort, order by and expand options to narrow the results.

By leveraging both container properties, custom columns and Enumeration, users can create highly specific and effective search queries, making the search experience even more powerful and tailored to their needs.

# SharePoint Embedded Search – Container Properties

## Glossary

In the context of Microsoft Graph and SharePoint Embedded, Drive and DriveItems serve distinct roles:

- **Drive:** This represents a top-level container for SharePoint Embedded. Essentially, a Drive is the logical container that holds all the files and folders within it.
- **DriveItems:** These are the individual files, folders, or other items stored within a Drive. Each DriveItem can represent a file, a folder, or even other types of items like images or videos. DriveItems have various properties and facets that provide metadata and capabilities specific to the type of item they represent.

## Container Properties

Properties can be created at the container level to help organize like containers and execute searches that can be filtered on the properties attached to the container. The search index will generally be updated with the new values within 15 minutes, specifically for drive entityTypes.

### Create a new Container Property

Method	URL
PATCH	<a href="https://graph.microsoft.com/beta/storage/fileStorage/containers/{{ContainerID}}/customProperties">https://graph.microsoft.com/beta/storage/fileStorage/containers/{{ContainerID}}/customProperties</a>

Body of the request

```
{  
  "{{ContainerPropertyName}}": {  
    "value": "{{Value}}",  
    "isSearchable": true  
  }  
}
```

Notes:

- Be sure to include the isSearchable attribute so it gets indexed.
- All properties will be text column types.

## Get containers by property name

Return all containers with a specific container property:

Method	URL
POST	<a href="https://graph.microsoft.com/beta/search/query">https://graph.microsoft.com/beta/search/query</a>

Body of the request

```
{
  "requests": [
    {
      "entityTypes": [
        "drive"
      ],
      "query": {
        "queryString": "departmentOWSTEXT:Accounting"
      },
      "sharePointOneDriveOptions": {
        "includeHiddenContent": true
      },
      "sortProperties": [
        {
          "name": "lastModifiedDateTime",
          "isDescending": true
        }
      ]
    }
  ]
}
```

Notes:

- The entityTypes is equal to drive, indicating that the search is scoped to the container level and the results will include containers matching this value (not files).
- The queryString parameter is decorated with the field type of the container property. The container property was created with the name of “department” and the field type of text. Following standard managed path functionality in SharePoint, the field type is appended to the end of the container property: departmentOWSTEXT.
- The only field types that are supported in the querystring search parameter are text fields - OWSTEXT.

- Standard KQL syntax can also be used.
- The results do not include the container property field in the JSON output.

## Search a container with a specific container property

Example of searching across the specified container where a container property equals the specified value:

Graph Call

Method	URL
POST	<a href="https://graph.microsoft.com/beta/search/query">https://graph.microsoft.com/beta/search/query</a>

Body of the request

```
{
  "requests": [
    {
      "entityTypes": [
        "drive"
      ],
      "query": {
        "queryString": "departmentOWSTEXT:Accounting AND
ContainerId:{ContainerID}"
      },
      "sharePointOneDriveOptions": {
        "includeHiddenContent": true
      },
      "sortProperties": [
        {
          "name": "lastModifiedDateTime",
          "isDescending": true
        }
      ]
    }
  ]
}
```

## Search all files with a specific container property

Example of searching across all files in the specified container where a container property equals the specified value.

**Note:** When a file is added to a container that has a container property assigned, that file will inherit the container property value. This container property can also be used to search on. When the container property is modified at the container level, existing files in the container are NOT updated with the new value.

Graph Call

Method	URL
POST	<a href="https://graph.microsoft.com/beta/search/query">https://graph.microsoft.com/beta/search/query</a>

Body of the request

```
{
  "requests": [
    {
      "entityTypes": [
        "driveItem"
      ],
      "query": {
        "queryString": "departmentOWSTEXT:Accounting AND ContainerId:{ContainerID}"
      },
      "sharePointOneDriveOptions": {
        "includeHiddenContent": true
      },
      "fields": [
        "departmentOWSTEXT",
        "id",
        "name",
        "parentReference",
        "file",
        "folder",
        "webUrl",
        "createdDateTime",
        "lastModifiedDateTime",
        "size",
        "createdBy",
        "lastModifiedBy",
        "fileSystemInfo"
      ],
    }
  ]
}
```

```
"sortProperties": [
  {
    "name": "lastModifiedDateTime",
    "isDescending": true
  }
]
}
```

Notes:

- The entityTypes is equal to driveItem indicating that the search is scoped to the file level.
- The querystring parameter indicates that all files matching the specified container property value (when the value assigned to the item when originally added to the container) and container ID will be used to execute the search. You can remove the ContainerID value to broaden the search to all files matching the container property specified.
- Standard KQL syntax can also be used.
- The results will include the container property field in the JSON output. Ensure that the desired field you want to appear in the JSON results output is specified in the fields section of the request body.

# SharePoint Embedded Search – Custom Columns

## Glossary

In the context of Microsoft Graph and SharePoint Embedded, Drive and DriveItems serve distinct roles:

- **Drive:** This represents a top-level container for SharePoint Embedded. Essentially, a Drive is the logical container that holds all the files and folders within it.
- **DriveItems:** These are the individual files, folders, or other items stored within a Drive. Each DriveItem can represent a file, a folder, or even other types of items like images or videos. DriveItems have various properties and facets that provide metadata and capabilities specific to the type of item they represent.

## When to use

Custom columns are best used for searching documents when:

- Complex KQL statements are needed to narrow down results.
- Results that span across multiple containers
- Scalable item results are desired

## Custom Columns

Custom Columns are created at the container level and are applied to files in that container. They are very similar to typical SharePoint metadata fields added to a document library where the column name will contain a value that can be searched on.

In this example, a column called “clientID” has been created. Each document added to the container can then be assigned a clientID that can be searched on. The search index will generally be updated with the new values within 15 minutes.

Example of searching all files in the specified container where a column equals the specified value:

Method	URL
POST	<a href="https://graph.microsoft.com/beta/search/query">https://graph.microsoft.com/beta/search/query</a>

Body of the request

```
{  
  "requests": [  
    {  
      "entityTypes": [  
        "driveitem"  
      ]  
    }  
  ]  
}
```

```

        ],
        "query": {
            "queryString": "clientIDOWSTEXT:XYZ987 AND ContainerId:{{ContainerID}}"
        },
        "sharePointOneDriveOptions": {
            "includeHiddenContent": true
        }
    ]
}
]
}

```

Notes:

- The entityTypes is equal to driveItem indicating that the search is scoped to the file level.
- The querystring parameter indicates that all files matching the specified column and value will be used to execute the search. You can also include the ContainerID value to narrow the search to files matching the column and value specified.
- Be sure to add the “includeHiddenContent”:true” attribute to the request body. By default SPE content is in the SharePoint index but no surfaced in standard SharePoint searching.
- When a custom column is created, it becomes a [managed property](#) and is postfixed with the data type. In this example, the [column type](#) is text and the column name to search on would be clientIDOWSTEST.
- Standard KQL syntax can also be used.
- The results will include the container property field in the JSON output. Ensure that the desired field you want to appear in the JSON results output is specified in the fields section of the request body.

## Expand field list

The JSON response does not list out all the item fields by default. Within the body of the request, you can specify the fields to return

Method	URL
POST	<a href="https://graph.microsoft.com/beta/search/query">https://graph.microsoft.com/beta/search/query</a>

Body of the request

```
{
    "requests": [
        {
            "entityTypes": [
                "driveItem"
            ],

```

```

"query": {
    "queryString": "clientIDOWSTEXT:XYZ987"
},
"sharePointOneDriveOptions": {
    "includeHiddenContent": true
},
"fields": [
    "departmentOWSTEXT",
    "clientIDOWSTEXT",
    "displayName",
    "id",
    "name",
    "parentReference",
    "file",
    "folder",
    "webUrl",
    "createdDateTime",
    "lastModifiedDateTime",
    "size",
    "createdBy",
    "lastModifiedBy",
    "fileSystemInfo"
],
}
]
}

```

## Sorting Columns

Certain columns can be sorted when specified in the request body. Generally this is limited to the standard file metadata such as createdDateTime, lastModifiedDateTime, size ETC.

Method	URL
POST	<a href="https://graph.microsoft.com/beta/search/query">https://graph.microsoft.com/beta/search/query</a>

Body of the request

```
{
    "requests": [
        {
            "entityTypes": [

```

```
        "driveItem"
    ],
    "query": {
        "queryString": "clientIDOWSTEXT:XYZ987"
    },
    "sharePointOneDriveOptions": {
        "includeHiddenContent": true
    },
    "fields": [
        "departmentOWSTEXT",
        "clientIDOWSTEXT",
        "displayName",
        "id",
        "name",
        "parentReference",
        "file",
        "folder",
        "webUrl",
        "createdDateTime",
        "lastModifiedDateTime",
        "size",
        "createdBy",
        "lastModifiedBy",
        "fileSystemInfo"
    ],
    "sortProperties": [
        {
            "name": "createdDateTime",
            "isDescending": true
        }
    ]
}
]
```

# SharePoint Embedded Search – Enumeration

SharePoint Embedded supports using query parameters (enumeration) that you can use to specify and control the amount of data returned in a response. Using the Microsoft Graph API, various options are available to filter, sort, order by and expand options to narrow the results returned to the calling application.

The following scenarios are supported by SharePoint Embedded:

1. Querying items within a folder (a.k.a. immediate children). Example:  
`$filter=parentReference/id eq '01JYFLNY46G4Z7LZQHGFD3UH PTR6WBLCIN'`
2. Filtering by listItem/fields/{fieldName} including built-in (default) and custom columns
3. An "and" combination of the above.

## When to use it

[Enumeration](#) is different than searching. When a new custom column is added to a container, it is added to the SharePoint search index and can use SharePoint Embedded search to retrieve items. With Enumeration, when a custom column is added to a container, the column and value is added to the underlying data store. When metadata is updated, it will update the data store immediately and can be retrieved in real time with no indexing delays. This does present some scalability issues such as retrieving less than 5,000 items and URL length restrictions.

- No Latency in returning item results.
- Queries that return less than 5,000 items
- Consuming content into a line of business application using a URL based approach.
- Items that are in the container specified.

## Filtering

Filtering columns in SharePoint Embedded requires understanding the various field types and their specific filter parameters. When setting up filters, it is crucial to decorate the field name with the appropriate designation like OWSTEXT or relevant field type.

The filter parameter value you use will precisely target items within both container properties and custom columns.

Filtering can be applied to enhance the precision of search results across containers, ensuring that the targeted data is retrieved efficiently and accurately. Additionally, remember that changes to container property values will have a propagation delay before reflecting in search results.

Method	URL
GET	<a href="https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?\$.filter=ExactSubjectOWSTEXT eq 'CustomerDocument1'">https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?\$.filter=ExactSubjectOWSTEXT eq 'CustomerDocument1'</a>

Method	URL
GET	<a href="https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?{\$filter=parentReference/id eq '&lt;ItemID&gt;'">https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?{\$filter=parentReference/id eq '&lt;ItemID&gt;'}</a>

Method	URL
GET	<a href="https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?{\$filter=contains(listitem/fields/FileLeafRef, '{partialFileName}')">https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?{\$filter=contains(listitem/fields/FileLeafRef, '{partialFileName}')}</a>

Method	URL
GET	<a href="https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?{\$filter=parentReference/id eq '{folderItemId}' AND contains(listitem/fields/FileLeafRef, '{partialFileName}')">https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?{\$filter=parentReference/id eq '{folderItemId}' AND contains(listitem/fields/FileLeafRef, '{partialFileName}')}</a>

Note:

- The “\$filter” querystring variable accepts a name value pair representing the column name and value assigned.
- Additional [operators and functions](#) can be used for more complex queries

## Select

The \$select parameter accepts valid field names and will group them at the top of the response under the “value” attribute. This will make it easier to parse the JSON response in your application and potentially reduce the payload returned to your application.

Method	URL
GET	<a href="https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?{\$filter=ExactSubjectOWSTEXT eq 'CustomerDocument1' &amp; \$select=id,name,lastModifiedDateTime,size,createdBy">https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?{\$filter=ExactSubjectOWSTEXT eq 'CustomerDocument1' &amp; \$select=id,name,lastModifiedDateTime,size,createdBy}</a>

Response (abbreviated)

```
{
  "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#drives({{ContainerID}})/items(id,name,lastModifiedDateTime,size)",
  "value": [
    {
      "id": "1234567890abcdef1234567890abcdef",
      "name": "CustomerDocument1.pdf",
      "lastModifiedDateTime": "2023-01-15T12:00:00Z",
      "size": 10485760
    }
  ]
}
```

```

"@odata.etag": "\'{1B0940B0-C43E-4380-B4C6-144512655797},2\''",
"id": "01UELPCRFQIAERWPWEQBB3JRQUIJGKV4X",
"lastModifiedDateTime": "2024-09-23T13:44:40Z",
"name": "ProjectFalcon",
"size": 3462953
},
{
"@odata.etag": "\'{83EEACD2-75BD-4AE7-8565-99FB29007864},2\''",
"id": "01UELPCRGSVTXIHPLV45FIKZMZ7MUQA6DE",
"lastModifiedDateTime": "2024-11-12T19:53:36Z",
"name": "StatusReports",
"size": 69238
},

```

## Expand

You can use the `$expand` query string parameter to include the specified fields in your results. Some field types are complex such as `ParentReference` because it will include multiple child attributes. These fields are not supported.

Method	URL
GET	<a href="https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?\$filter=ExactSubjectOWSTEXT eq 'CustomerDocument1'&amp;\$expand=listitem(\$expand=fields)">https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?\$filter=ExactSubjectOWSTEXT eq 'CustomerDocument1'&amp;\$expand=listitem(\$expand=fields)</a>

Note:

- The “listitem” attribute in the JSON response will expand the `fields` attribute to include all the columns for the item.
- 

## Order By

Use the `$orderby` query parameter to specify the sort order of the items returned from Microsoft Graph. The default order is ascending order.

Method	URL
GET	<a href="https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?\$filter=ExactSubjectOWSTEXT eq 'CustomerDocument1'&amp;\$orderby=listitem/fields/Modified desc">https://graph.microsoft.com/v1.0/drives/{{ContainerID}}/items?\$filter=ExactSubjectOWSTEXT eq 'CustomerDocument1'&amp;\$orderby=listitem/fields/Modified desc</a>

Note:

- The field you want to order by needs to follow the column relationship. For example “Modified”, the last modified date of the item, is in the listItem/fields structure of the JSON response.
- To get the column relationship to use in the Order By parameter, use the \$expand parameter to get the full JSON response.

## Appendix – Additional samples

Additional

- /items?\$filter=contains(listItem/fields/FileLeafRef, '{partialFileName}')
- /items?\$filter=contains(listItem/fields/{customColumnName}, '{partialCustomColumnName}')
- /items?\$filter=contains(listItem/fields/FileLeafRef, '{partialFileName}') or contains(listItem/fields/{customColumnName}, '{partialCustomColumnName}')
- /items?\$filter=startswith(listItem/fields/FileLeafRef, '{partialFileName}') or contains(listItem/fields/{customColumnName}, '{partialCustomColumnName}')
- /items?\$filter=listItem/fields/{customColumnName}eq '{customColumnName}'&\$select=id,name,lastModifiedDateTime  
&\$expand=listItem(\$expand=fields)&\$orderBy=listItem/fields/FileLeafRef
- /items?\$filter=listItem/fields/{customColumnName}eq '{customColumnName}'&\$select=id,name,lastModifiedDateTime  
&\$expand=listItem(\$expand=fields)&\$orderBy=LastModifiedDateTime
- /items?\$filter=listItem/fields/{customColumnName}eq '{customColumnName}'&\$select=id,name,lastModifiedDateTime  
&\$expand=listItem(\$expand=fields)&\$orderBy=createdDateTime

Paged folder contents (immediate items)

- /items?\$filter=parentReference/id eq '{folderItemId}'

All items matching the filters are returned (scoped to the folder)

- /items?\$filter=parentReference/id eq '{folderItemId}' and contains(listItem/fields/FileLeafRef, '{partialFileName}')

## SharePoint Embedded Search – List only Folders

We have learned so far that the search functionality in SharePoint Embedded can be very robust when using metadata and traditional KQL statements. Translating this to a more real-world example would be getting only the folders in a container.

### When to use it

Suppose you're building an application that shows a folder structure, allowing users to navigate to the correct folder to view the document contents. This simulated Windows Explorer experience can be accomplished by targeting the Content Type of Folder. The JSON results can be presented to the user in a hierarchical fashion.

Here is the syntax to return only folders.

Method	URL
POST	<a href="https://graph.microsoft.com/beta/search/query">https://graph.microsoft.com/beta/search/query</a>

Response (abbreviated)

```
{
  "value": [
    {
      "searchTerms": [],
      "hitsContainers": [
        {
          "hits": [
            {
              "hitId": "01VD3T4JCGX7KCDQX7UJHKG7FEEMQ5VWQK",
              "rank": 1,
              "summary": "",
              "resource": {
                "@odata.type": "#microsoft.graph.driveItem",
                "size": 0,
                "fileSystemInfo": {
                  "createdDateTime": "2024-08-06T20:21:32Z",
                  "lastModifiedDateTime": "2024-08-06T20:21:32Z"
                },
                "listItem": {
                  "@odata.type": "#microsoft.graph.listItem",
                  "id": "21d4bf46-ffc2-4ea2-a37c-a42321dada0a",
                  "fields": {
                    "name": "Container"
                  }
                }
              }
            }
          ]
        }
      ]
    }
  ]
}
```

```

    "id": "AAAAAFz8XR-
O96JJnaDecJVyBZcHABoXtuG4CWtNtd6lCszT2l8AAAApYVgAABoXtuG4CWtNtd6lCszT2l
8AA0peyzAAAA2",
    "contentType": "Folder",
    "size": 0,
    "createdBy": "Steve Pucelik"
}
},
"id": "01VD3T4JCGX7KCDQX7UJHKG7FEEMQ5VWQK",
"createdBy": {
    "user": {
        "displayName": "Steve Pucelik",
        "email": "steve@<DOMAIN>.onmicrosoft.com"
    }
},
"createdDateTime": "2024-08-06T20:21:32Z",
"lastModifiedBy": {
    "user": {
        "displayName": "Steve Pucelik",
        "email": "steve@<DOMAIN>.onmicrosoft.com"
    }
},
"lastModifiedDateTime": "2024-08-06T20:21:32Z",
"name": "Weekly Status Reports",
"parentReference": {
    "driveId": "<ID>",
    "id": "01VD3T4JDN6XC32LMJAJAKULUIZ66JDDMU",
    "sharepointIds": {
        "listId": "77fd54fe-46fe-4d76-8992-b0d1bc83230f",
        "listItemId": "3",
        "listItemUniqueId": "21d4bf46-ffc2-4ea2-a37c-a42321dada0a"
    },
    "siteId": "<DOMAIN>.sharepoint.com,<ID>,<ID>"
},
"webUrl": "https://
<DOMAIN>.sharepoint.com/contentstorage/CSP_<ID>/Document Library/Weekly Status
Reports"
}
},

```

### Note:

- This will return all the folders in the container.
- You can use the parentReference -> id attribute to establish the relationship between the root folders and children so you can “[walk the folder hierarchy](#)”.

# Sensitivity Labels in SharePoint Embedded

Sensitivity Labels in SharePoint Embedded offer an advanced method to bolster your governance strategy by leveraging the labels already configured in your tenant. These labels help protect sensitive information within your SharePoint Embedded containers, ensuring that data is accessed and shared based on how you have configured the label. By integrating these labels into SharePoint Embedded, organizations can ensure consistent application of policies, enhance compliance, and safeguard critical assets, ultimately fortifying their governance posture.

## When to use it

In addition to using permissions and the SharePoint Embedded security model, Sensitivity Labels can further secure and protect your content. For example, if you have an application that has external collaboration with vendors where they need to view policies. Naturally you don't want them to edit the policy. By applying a sensitivity label that has been configured to restrict edit capabilities if you are an external user, those drafting the document can apply that sensitivity label allowing internal users to edit the policy but external users to view it.

## How does it work

Sensitivity labels in Purview is a very large and complicated topic. I'll focus on those elements specific to SharePoint Embedded and how you can secure and protect content.

1. When creating or editing a sensitivity label, ensure the Groups & Sites is selected. This will include SharePoint Embedded content stored in containers.

The screenshot shows the Microsoft Purview interface for editing a sensitivity label. The top navigation bar includes 'Microsoft Purview', a search bar, and various icons. The main area is titled 'Edit sensitivity label'. On the left, a sidebar shows a tree structure with 'Label details' checked, followed by 'Scope' (which is selected and highlighted in blue), 'Items', 'Groups & sites', 'Schematized data assets (preview)', and 'Finish'. The right panel is titled 'Define the scope for this label' and contains a sub-section titled 'Define the scope for this label'. It explains that labels can be applied to data assets and containers like SharePoint sites and Teams. It lists several options with checkboxes:

- Files & other data assets**: Label files and data assets in Microsoft 365, Microsoft Fabric (includes Power BI), Microsoft Azure.
- Emails**: Label messages sent from all versions of Outlook.
- Meetings**: Label calendar events and meetings schedules in Outlook and Teams.
- Groups & sites**: Configure privacy, access control, and other settings to protect labeled Teams, Microsoft 365 Groups, SharePoint sites, and Loop workspaces.

The 'Groups & sites' option is highlighted with a red border.

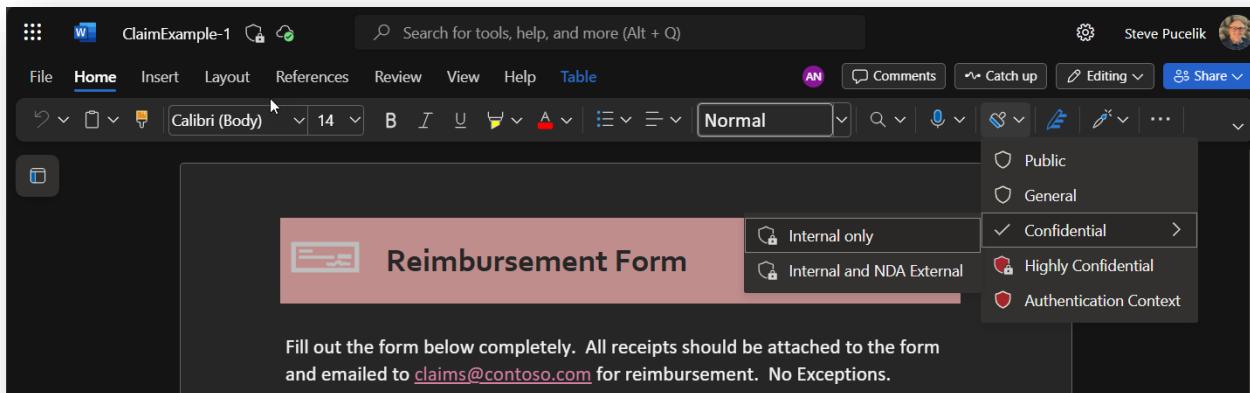
2. Select the user or group you want to configure permissions for.

The screenshot shows the Microsoft Purview interface for editing a sensitivity label. The left sidebar lists steps: Label details, Scope, Items, Access control, Auto-labeling for files and emails, Groups & sites, and Schematized data assets (preview). The 'Scope' step is currently selected. On the right, there's a configuration panel for access control settings, including options for co-authoring, permission assignment, user access expiration, and offline access. Below this is a 'Users and groups' section with a 'Assign permissions' button. A modal window titled 'Add users or groups' is overlaid, showing a search input with 'exter' and a list of found users: 'ExternalSharing' and 'ExternalTeam'. The 'ExternalTeam' user is selected, indicated by a checked checkbox.

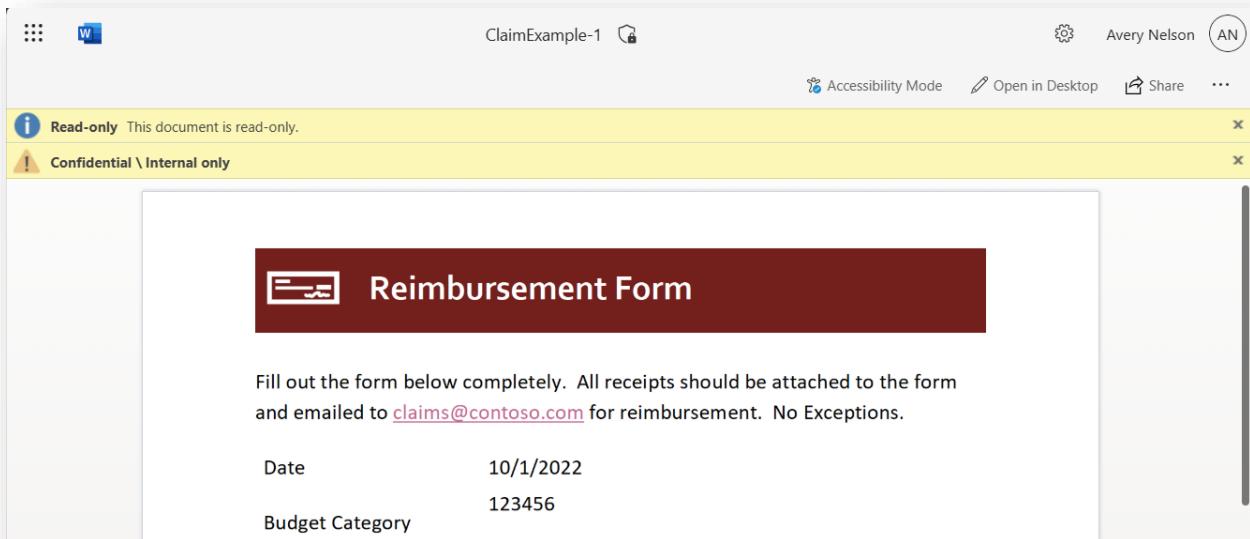
3. Select the permission level for that group.

The screenshot shows the Microsoft Purview interface for editing a sensitivity label. The left sidebar lists steps: Label details, Scope, Items, Access control, Auto-labeling for files and emails, Groups & sites, Schematized data assets (preview), and Finish. The 'Items' step is currently selected. On the right, there's a configuration panel for access control settings, including options for co-authoring, permission assignment, user access expiration, and offline access. Below this is a 'Users and groups' section with a 'Assign permissions' button. A modal window titled 'Choose permissions' is overlaid, showing a dropdown menu set to 'Viewer' and a list of permissions. The permissions listed are: View content(VIEW), View rights(VIEWRIGHTSDATA), Edit content(EDIT), Save(EDIT), Print(PRINT), Copy and extract content(EXTRACT), Reply(REPLY), Reply all(REPLYALL), Forward(FORWARD), Edit rights(EDITRIGHTSDATA), Export content(EXPORT), Allow macros(OBJMODEL), and Full control(OWNER). Most permissions have checkboxes checked except for Save(EDIT) and Export content(EXPORT).

4. Assign the label to the document that has been uploaded to a SharePoint Embedded container.



- When an external user opens the document, notice that the ribbon is removed, sharing is disabled and all the user can do is view the document.



# Audit Log Integration

The audit log integration with SharePoint Embedded serves as a crucial tool for tracking and documenting user activity within the environment. It is particularly useful for compliance, security monitoring, and troubleshooting purposes. The integration captures detailed records of actions such as file access, edits, deletions, and sharing events, providing administrators with comprehensive visibility into user interactions. This functionality is essential for maintaining data integrity and ensuring adherence to organizational policies and regulations.

## When to use it

Imagine your company faces a data breach, and you're worried that sensitive info might have been accessed or messed with without permission. With SharePoint Embedded's audit logs, your IT security team can dive into the details of what users did with those files. They'll be able to track who accessed, edited, or shared what and when. This detailed view helps pinpoint which user accounts were involved and spot any odd behavior that could suggest insider threats or outside hacks. This information doesn't just help tackle the current issue but also strengthens future data protection and fine-tunes access controls.

## How does it work

The first thing we need to do is get the container information that you want to use in the audit log search. You can get a list of all the containers in your application with this graph command:

GET

```
https://graph.microsoft.com/beta/storage/fileStorage/containers?$filter=containerTypeId eq {{ContainerTypeId}}
```

Response:

```
"value": [  
  {  
    "id": "b!OkqqzjCtLk23YdGE1Vho4O_5EYMK3QRNqim1usWvFk5tNHHJRowqSLXzbsw8HYbl",  
    "displayName": "Project Documents",  
    "containerTypeId": "ee469b9e-3451-0e71-1384-0fbc70aa001a",  
    "createdDateTime": "2024-06-13T15:16:15Z",  
    "lockState": "unlocked",  
    "viewpoint": {  
      "effectiveRole": "owner"
```

```
},
"settings": {
    "isOcrEnabled": false
},
},
```

When you have found the container you want to use, get the “id” value and then run this command:

GET

<https://graph.microsoft.com/v1.0/drives/{{ContainerID}}>

Response (Abbreviated):

```
{
    "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#drives/$entity",
    "createdDateTime": "2024-08-18T05:14:52Z",
    "description": "The container is configured with metadata for searching.",
    "id": "b!CORq-a8orUGIrd3_z9t1_vjCBSeqM3JKhDgIEU3DIDvEl-Hms0qoQ7QCWYNQfGOF",
    "lastModifiedDateTime": "2025-01-29T12:46:12Z",
    "name": "Project Metadata",
    "webUrl": "https://<Domain>.sharepoint.com/contentstorage/CSP_f96ae408-28af-41ad-88ad-ddffcfdb75fe/Document%20Library",
    "driveType": "other",
    "createdBy": {
        "user": {
            "displayName": "System Account"
        }
    },
}
```

Ultimately what we’re after is the webUrl value, specifically the container identifier which is [https://<DOMAIN>.sharepoint.com/contentstorage/CSP\\_f96ae408-28af-41ad-88ad-ddffcfdb75fe](https://<DOMAIN>.sharepoint.com/contentstorage/CSP_f96ae408-28af-41ad-88ad-ddffcfdb75fe).

## Configure Purview

Now that we have the container information, which is simply a site for this purpose, create a new audit search:

The screenshot shows the Microsoft Purview Audit search interface. The left sidebar includes Home, Solutions (Audit selected), Search, Policies, Settings, Data Loss Prevention, Audit, Information Protection, and Records Management. The main area has a 'Search' title and displays metrics: Searches completed (0), Active searches (0), and Active unfiltered searches (0). It features a Date and time range (UTC) section with Start (Feb 10 2025, 00:00) and End (Feb 15 2025, 00:00) fields. Other search parameters include Activities - friendly names, Activities - operation names, Record Types, Admin Units, Users, File, folder, or site (with a URL input field), Workloads (set to SharePoint), and a Search name field. A 'Search' button and a 'Clear all' button are at the bottom.

### Notes:

- Specify the date and time range for the activities you are looking for.
- Add the webUrl value (the container) to the File, folder or site.
- Select “SharePoint” as the Workload and submit the search.

Once the job has been completed, you can view the results to include all the activities performed by users:

Search Query Information: Mon, 10 Feb 2025 00:00:00 GMT to Sat, 15 Feb 2025 00:00:00 GMT , https://28af-41ad-88ad-ddfcfdb75fe , SharePoint ,

Total Result Count: 20 items

[Export](#)

Date (UTC) ↓	IP Address	User	Record Type	Activity	Item
Feb 14, 2025 2:01 PM	40.126.27.96	steve@...	SharePointSharingOperation	Added user or group to SharePoint group	https://...
Feb 14, 2025 2:01 PM	40.126.27.96	steve@...	SharePointSharingOperation	Added user or group to SharePoint group	https://...
Feb 14, 2025 2:01 PM	40.126.27.96	steve@...	SharePointSharingOperation	Added user or group to SharePoint group	https://...
Feb 14, 2025 2:01 PM	40.126.27.96	steve@...	SharePointSharingOperation	Added user or group to SharePoint group	https://...
Feb 13, 2025 1:26 PM	20.190.155.36	steve@...	SharePointSharingOperation	Removed user or group from SharePoint gro...	https://...
Feb 13, 2025 1:26 PM	20.190.155.36	steve@...	SharePointSharingOperation	RemovedFromSiteCollection	https://...
Feb 12, 2025 12:42 PM	20.190.132.106	app@sl...	SharePointSharingOperation	Added user or group to SharePoint group	https://...
Feb 12, 2025 12:29 PM	40.126.27.96	steve@...	SharePointSharingOperation	Removed user or group from SharePoint gro...	https://...
Feb 12, 2025 12:29 PM	40.126.27.96	steve@...	SharePointSharingOperation	RemovedFromSiteCollection	https://...
Feb 11, 2025 1:19 PM	20.190.135.42	steve@...	SharePoint	Created group	https://...
Feb 11, 2025 1:19 PM	20.190.135.42	steve@...	SharePointSharingOperation	Added user or group to SharePoint group	https://...
Feb 11, 2025 1:19 PM	20.190.135.42	steve@...	SharePointSharingOperation	Added user or group to SharePoint group	https://...
Feb 11, 2025 1:19 PM	20.190.135.42	steve@...	SharePointSharingOperation	Added user or group to SharePoint group	https://...
Feb 11, 2025 1:19 PM	20.190.135.42	steve@...	SharePointSharingOperation	Added user or group to SharePoint group	https://...

## Details

### Details

Date (UTC)  
2025-02-14T14:01:26

### IP Address

40.126.27.96

### Users

i0h.fjmembership|10037ffe9a9489dd@live.com

### Activity

AddedToGroup

### Item

https://...sharepoint.com/contentstorage/CSP\_f96ae408-28af-41ad-88ad-ddfcfdb75fe

### Details

#### Admin Units

#### AppAccessContext

```
{
  "AADSessionId": "0020af49-b3ef-c6e5-17eb-82a4bffe3e20",
  "APIId": "00000003-0000-0000-000000000000",
  "ClientAppName": "SPE-Baseball",
  "CorrelationId": "9176c07c-7d01-4d4c-a164-f2600a0f87fe",
  "UniqueId": "ze0tSMXxCky05PYe2jhFAA"
}
```

#### CreationTime

2025-02-14T14:01:26

#### Id

45ba920-132a-427d-8037-08dd4d0012f0

#### Operation

AddedToGroup

[Close](#)

# eDiscovery and litigation hold

Microsoft Purview eDiscovery is seamlessly integrated with SharePoint Embedded, allowing organizations to preserve, search, and export electronic information efficiently. This integration ensures that all relevant data within SharePoint Embedded Containers can be easily accessed and managed during the litigation process. It supports compliance with legal requirements by automating the discovery of documents, reducing the risk of data loss or tampering, and facilitating a defensible audit trail.

## When to use it

Consider an organization facing a legal dispute involving wrongful termination claims. As part of the litigation process, the legal team needs to gather all relevant communications, employee records, and associated documents stored within the company's SharePoint Embedded Containers. By utilizing Microsoft Purview eDiscovery, the organization can swiftly preserve, search, and export the required electronic information. This enables the legal team to efficiently review and present evidence, ensuring compliance with legal standards and reducing the likelihood of data loss or alteration.

## How does it work

Using what we learned during the audit log configuration, we can use that same container within an eDiscovery case as a data source that can then be added to a review set.

Once you have an eDiscovery case created, we need to add the container as a data source:

### Adding a data source

- In the “Data Sources” tab, add a new data source.
- NOTE:** This will place a litigation hold on all the content in the container.
- Edit the SharePoint sites so we can add our own data source.

### New non-custodial data locations

SharePoint

[Edit SharePoint sites](#)

Exchange

- Using the method to get the container endpoint we used in the audit log example, get the root of the container and add it as a data source

## Edit assigned Sharepoint locations

:sharepoint.com/contentstorage/CSP\_f96ae408-28af-41ad-88ad-ddffcfdb75fe +

- Once that is done, you can monitor the indexing process.

The screenshot shows the eDiscovery (Premium) interface. The navigation bar at the top includes 'eDiscovery (Premium)', 'Cases', and 'SharePoint Embedded'. Below the navigation, there's a breadcrumb trail: 'Overview' > 'Data sources' > 'SharePoint Embedded'. The main content area is titled 'Edit assigned Sharepoint locations' and contains a URL input field with the value ':sharepoint.com/contentstorage/CSP\_f96ae408-28af-41ad-88ad-ddffcfdb75fe'. Below the input field is a large, semi-transparent blurred area. At the bottom of the page, there's a table listing the data source details:

Source name	Source type	Locations	Source status	Indexing status	Hold status	Index date
https://sharepoint.com/contentstorage/CSP_ceaa4a3a-ad30-4d2e-b761-d184d55868e	Data location	1	Active	Fully indexed	On hold	Sep 5, 2024 10:19 AM

- Collections allow you to focus on a certain type of documents from various custodians and data sources. When creating a collection and choosing the non-custodial data source, select the documents from the data source we previously created. Once processing has completed, commit it to a new or existing Review Set.

New collection

- Name and description
- Custodial data sources
- Non-custodial data sources
- Additional locations
- Search query
- Review your collection

Choose non-custodial data source

These are the sites, groups, and other sources that you can apply this collection to non-custodial sources for this case.

Search for specific non-custodians

1 item

Name	Location
https://sharepoint.com/contentstor...	https://sharepoint.com/...

- Now you're ready to look at the documents in the review set where you can view the document and assign the appropriate tags.

eDiscovery (Premium) > Cases > SharePoint Embedded > SharePoint Documents

Saved filter queries

Filters Undo filter query Redo filter query

AND Select a filter Add filter Add subgroup

1 of 4 selected

#	Subject/Title	Status	Tag Status	Date (UTC-05:00)	Sender
1	Sample_Invoice.pdf	Ready	Tagged	Jun 14, 2024 11:57:00	steve@...
2	Sample_Excel-Budg...	Ready	No Tag	Jun 27, 2024 3:43:40	steve@...
3	Sample_Invoice_BR...	Ready	No Tag	Jul 16, 2024 2:26:46	steve@...
4	How to Create a Gr...	Ready	No Tag	Mar 31, 2024 4:00:00	Reid Ca...

Show pinned metadata

Source Plain text Annotate Metadata

Tag

Viewing: Page 1 of 1 | 50 items/page

Tag files

Create/edit tags

Choose selection

Tag selected items (1)

Tag all items in list (4)

Expand selection None

Assign tags

Review Status

Relevant (1/1)

Responsive

Privileged

Non-Relevant

Confidential

Hot Documents

Apply tags Close

- The only thing left to do is export the items from the review set and distribute them to interested parties in the case.

# Copilot Agent Overview

The configuration process for the "Custom copilot with SharePoint Embedded" allows developers to integrate an agent into third-party applications using a React component. The chat experience can be dynamically scoped to file, folder, and container contexts based on the app context, and the agent's user interface can be customized with starter prompts, meta prompts, and style. Additionally, a per-user Copilot license is required for the preview. The SPE copilot stack for applications includes components such as permission-aware, scoped, semantically indexed data, and core enterprise AI.

## When to use it

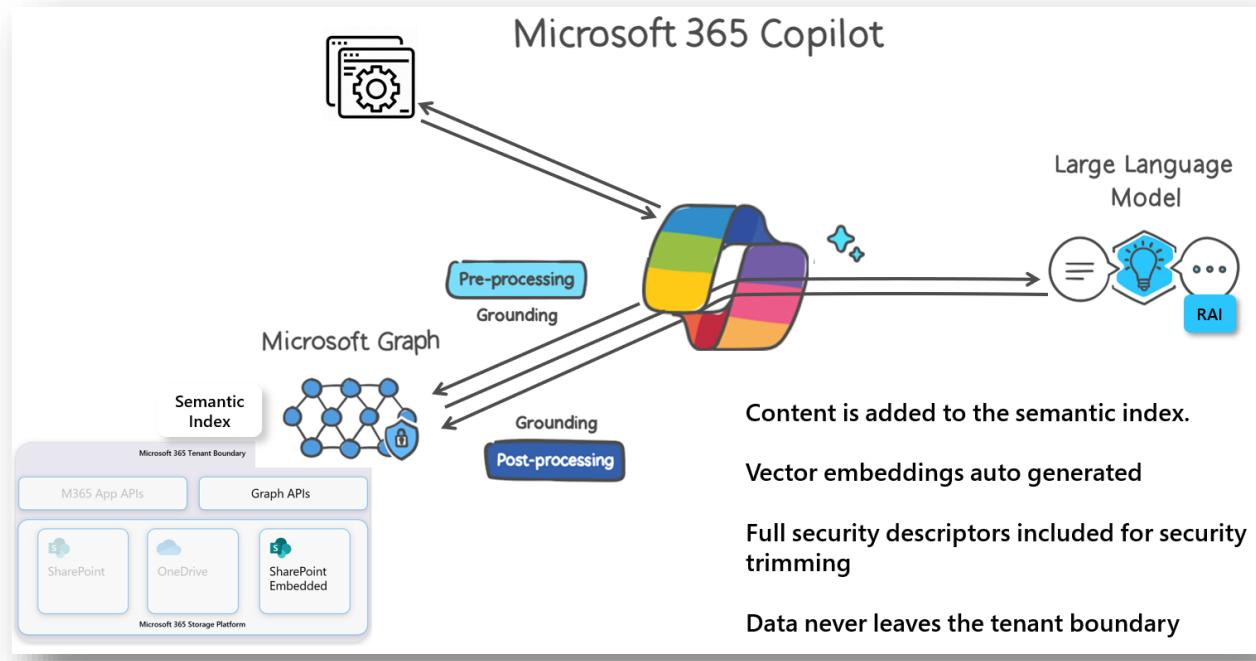
SharePoint Embedded Agents can be effectively used in scenarios where there is a need to integrate intelligent chat experiences into third-party applications.

A practical use case might involve a company that manages extensive document repositories, requiring users to navigate complex data efficiently. By implementing a SharePoint Embedded Agent, users can receive real-time assistance, navigate files smoothly, and extract insights directly within their workflow, enhancing productivity. The agent's interface can be customized for specific branding, prompting users with starter and meta prompts that align with the company's operational needs.

Furthermore, utilizing SharePoint Embedded Agents allows for a seamless integration of scoped, semantically indexed data and core enterprise AI capabilities, ensuring that the information provided is not only contextually relevant but also robust in its application. This makes it an ideal solution for enterprises aiming to leverage AI-driven assistance to optimize document management and collaboration processes.

## How it works

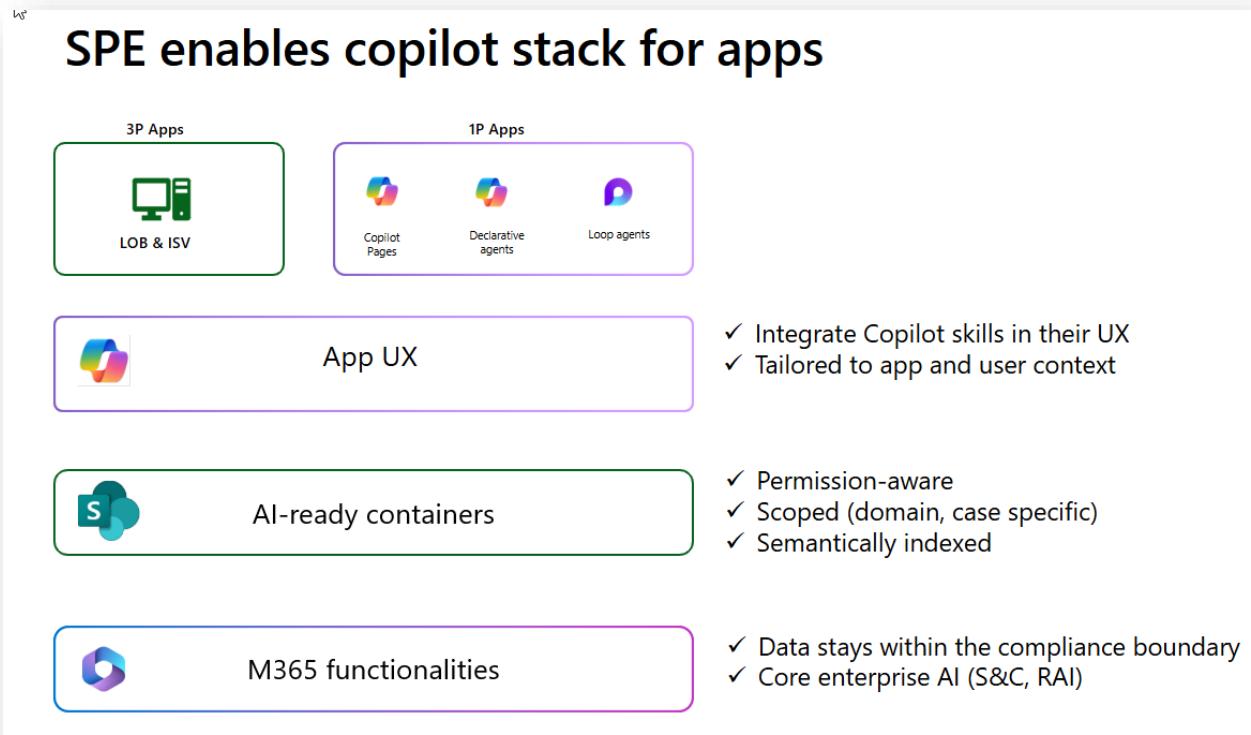
To get grounded on where SharePoint Embedded agents fit into your Copilot journey, here is an overview of the high-level components.



### Notes:

- When content is added to a container and indexed, it is also added to the semantic index that agents reason over.
- Security on the content is always maintained and honored.

Integration doesn't always have to be within your customer application. Consider a line of business application or 3rd party app. Bringing AI experiences on your SharePoint Embedded content to where your users are is achievable with configurable agents.



## Configuring your app with SPE the Copilot Agent

The SharePoint Embedded Copilot chat control enables features such as document analysis in SharePoint, customizable search scopes, and configurable chat settings. Developers can tailor prompts, colors, and more. To use Copilot, ensure it is available for your organization through either a sandbox tenant or a production environment with the appropriate license.

### When to use it

This control is ideal for use cases where developers want to enhance user interactions within SharePoint Embedded content. It is particularly useful for applications requiring advanced document analysis, where customizable search scopes and chat settings are beneficial. This flexibility allows for tailored user experiences, adapting to the specific context of files, folders, or containers within the app. The ability to personalize prompts and styles further enriches the user engagement, making it a valuable tool for organizations aiming to leverage AI-enhanced capabilities in their workflows.

### How it works

**NOTE:** This functionality is currently in private preview.

Make sure to check out the sample apps we have to quickly get started. ([aka.ms/spe-examples](https://aka.ms/spe-examples))

### Developer experience:

- Plug in agent using the React SDK (based off SharePoint Agent)
- Dynamically scope the chat experience to file, folder, container(s) based on app context
- Customize agent UX – starter prompts, meta prompts, style
- Per user Copilot license required for preview

### Get the SDK

Before you can start using Copilot chat in your application, you must download the necessary components and install them into your environment.

<Code Block>

```
# Install the SDK with npm  
npm install "https://download.microsoft.com/download/27d10531-b158-40c9-a146-af376c0e7f2a/microsoft-sharepointembedded-copilotchat-react-1.0.7.tgz"
```

</Code Block>

## Authenticate

You will need to implement the security token that the user has logged in with.

Create an authProvider object in your application, or use one already implemented that will get the correct bearer token needed to initiate a Copilot experience:

<CODE BLOCK>

```
ChatAuthProvider.getInstance().then(setChatAuthProvider).catch(console.error);  
</CODE BLOCK>
```

## Add to your app

Now that you have the component in your application and context of the user, it's time to add the component:

<CODE BLOCK>

```
return (>  
  {chatAuthProvider && (  
    <ChatEmbedded  
      authProvider={chatAuthProvider}  
      onApiReady={onApiReady}  
      containerId={container.id}  
    />  
  })  
</>);
```

**Commented [PT1]:** As of 1.0.7 this now requires a 'containerId':

containerId={container.id}

(see the updated doc linked in my other comment)

```
</CODE BLOCK>
```

Notice that in the return section is where you'll place the chat experience to be rendered. It will also need the authProvider component that will have the bearer token needed to reason over the container content.

Once the component is added to your application, you will want to control the look, feel and give the user some suggested prompts to get started. The chatConfig section allows you to customize it to your user experience.

```
<ScriptBlock>
```

```
  const [chatAuthProvider, setChatAuthProvider] = React.useState<ChatAuthProvider | undefined>();
```

```
  const [chatConfig] = React.useState<ChatLaunchConfig>({  
    header: ChatController.instance.header,  
    theme: ChatController.instance.theme,  
    zeroQueryPrompts: ChatController.instance.zeroQueryPrompts,  
    suggestedPrompts: ChatController.instance.suggestedPrompts,  
    instruction: ChatController.instance.pirateMetaPrompt,  
  });
```

```
</ScriptBlock>
```

## Control the data context

As users are interacting with the Copilot Agent in your application, you can control the content that is being reasoned over, meaning do we want to use all the documents in a container or just a document the user selects. When adding the following code to your component, it will add the respective container or document to a datasources object which will tell Copilot to only use those defined datasources to reason over.

```
<ScriptBlock>
```

```
  const onApiReady = async (api: ChatEmbeddedAPI) => {
```

```
await api.openChat(chatConfig);

ChatController.instance.addDataSourceSubscriber(dataSources => {
    api.setDataSources(dataSources);
});

}

ChatAuthProvider.getInstance().then(setChatAuthProvider).catch(console.error);

</ScriptBlock>
```

# Copilot Studio Integration

Copilot Studio can integrate intelligent chat experiences into third-party applications, enhancing productivity by aiding users in navigating complex data efficiently and extracting insights directly within their workflow. These agents can be customized for specific branding, aligning with a company's operational needs. They allow seamless integration of semantically indexed data and core enterprise AI capabilities, making them ideal for optimizing documents that are stored in SharePoint Embedded.

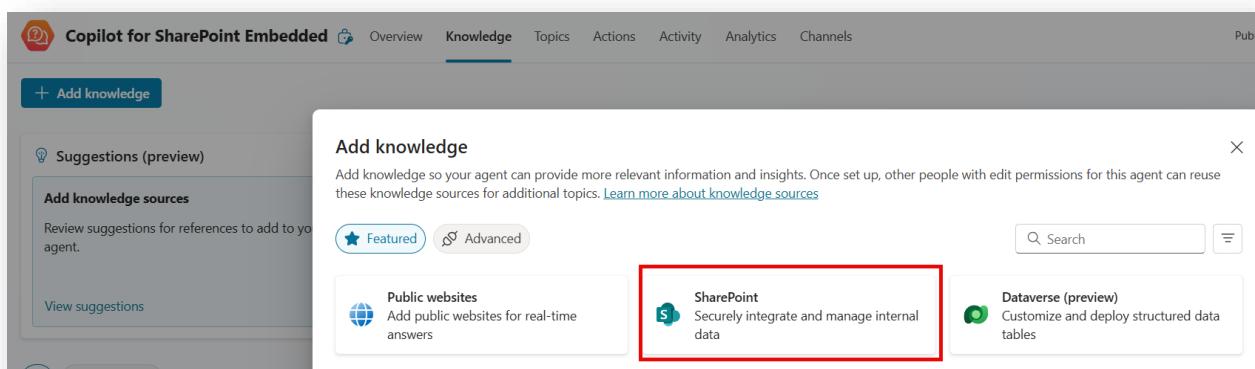
## When to use it

Copilot Studio can use documents stored in SharePoint Embedded as knowledge sources, enhancing productivity by aiding users in navigating complex data efficiently and extracting insights directly within their workflow. For instance, consider a legal firm that manages extensive documentation and contracts within your custom application. By deploying Copilot Studio agents, lawyers and paralegals can swiftly search and retrieve pertinent clauses and case references without manually sifting through volumes of files. This not only saves time but ensures accuracy and enhances client service by providing quick access to critical information.

## How it works

As we've seen with other M365 integrations, Copilot Studio can be configured with various sources of knowledge that the agent can reason over. To configure an agent use SharePoint Embedded as a knowledge source:

- Create a new agent and navigate to the Knowledge tab.
- When you "Add knowledge", select SharePoint.



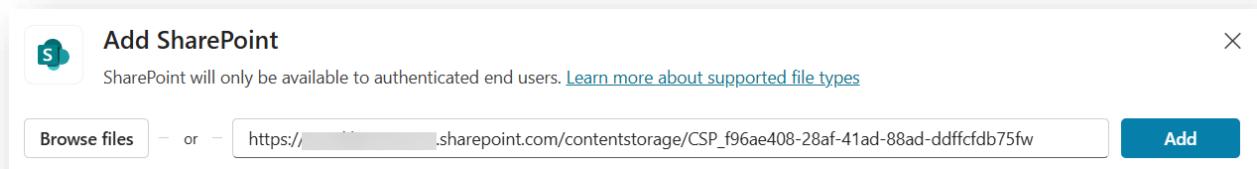
- Add the URL to the container you want the agent to reason over. This can be retrieved using the following Graph API call:

GET <https://graph.microsoft.com/v1.0/drives/{{ContainerID}}>

Response:

```
{  
    "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#drives/$entity",  
    "createdDateTime": "2024-08-18T05:14:52Z",  
    "description": "The container is configured with metadata for searching.",  
    "id": "b!CORq-a8orUGIrd3_z9t1_vjCBSeqM3JKhDgLEU3DIDvEl-Hms0qoQ7QCWYNQfGOF",  
    "lastModifiedDateTime": "2025-01-29T12:46:12Z",  
    "name": "Project Metadata",  
    "webUrl": "https://[REDACTED].sharepoint.com/contentstorage/CSP_f96ae408-28af-41ad-88ad-  
-ddffcfdb75fe/Document%20Library",  
    "driveType": "other",  
    "createdBy": {  
        "user": {  
            "displayName": "System Account"  
        }  
    },  
}
```

Remember, we're only getting the root of the container.



- Test your agent on content that you know is in the SharePoint Embedded container.

What is the total amount of claims submitted in table format with the claimant, amount and claim date as column headers

Just now

Based on official sources

<b>Claimant</b>	<b>Amount</b>	<b>Claim Date</b>
-----------------	---------------	-------------------

Debra Berger	652.55	11/2/2021
--------------	--------	-----------

Mason Frederick	847.55	10/14/2021
-----------------	--------	------------

The total amount of claims submitted is  
\$1,500.10 [1](#) [2](#) [3](#).

3 references ▾

1 [ClaimExample-5.docx](#)

2 [ClaimExample-4.docx](#)

3 [ClaimExample-2.docx](#)

You can now use this as an iFrame in your application or Power App to provide an easy to configure way to reason over SharePoint Embedded content.