

**Diagram Explanation**

1. **User Action in Dynamics 365/CRM (A)**
   * A change in a record (create, update, or assign) triggers the plugin.
2. **Dataverse Plugin (C#) (B)**
   * The plugin executes in response to the user action.
   * It calls the MSAL.NET module to obtain an access token.
3. **MSAL.NET Authentication Module (C)**
   * Uses the **client credentials flow** to request an access token.
   * Communicates with **Azure AD (D)** and leverages the credentials from the **Azure AD App Registration (E)**.
4. **Azure AD (Tenant) (D) & App Registration (E)**
   * Azure AD validates the credentials and issues an access token.
5. **Microsoft Graph API (F)**
   * The plugin then calls the Graph API with the token to update folder permissions in SharePoint.
6. **SharePoint Site/Folder (G)**
   * The Graph API request updates folder-level access permissions as required.

This architecture ensures that the authentication and permission changes happen synchronously through a secure channel between Dynamics 365, Azure AD, and SharePoint.

**Solution Overview**

Since Dataverse and SharePoint don’t natively support **synchronous permission updates**, you need to use **Power Automate + SharePoint REST API** or **a custom plugin with SharePoint CSOM/Graph API**.

**Approach 1: Power Automate (Recommended for Low Code)**

1. **Trigger**: Use a "Dataverse" trigger in **Power Automate** that runs on create/update of the record.
2. **Check Permissions**: Retrieve the user and their role in Dynamics.
3. **Update SharePoint Permissions**: Use **SharePoint REST API** to grant/revoke folder-level permissions.
4. **Ensure Immediate Execution**:
   * Use a **synchronous Dataverse Cloud Flow**
   * OR call the Flow from a **plugin/custom workflow** using an **HTTP request**.

**Approach 2: Custom Plugin (C#) for Synchronous Execution**

1. **Create a Plugin** in Dataverse that triggers on an action (record update, ownership change, etc.).
2. **Use SharePoint Graph API or CSOM** within the plugin to modify permissions immediately.
3. **Register the Plugin** in the **Pre-Operation/Pre-Validation** stage to ensure real-time updates.

**Implementation Details**

**1. Power Automate Flow**

* **Trigger**: When a record is updated in Dataverse (like Account or Opportunity).
* **Get SharePoint Folder URL**: Retrieve the folder linked to the record.
* **Call SharePoint REST API** (/\_api/web/GetFolderByServerRelativeUrl(...)) to update permissions.

**REST API Example to Grant Permissions**:

http

CopyEdit

POST https://yourtenant.sharepoint.com/\_api/web/GetFolderByServerRelativeUrl('/sites/yoursite/yourfolder')/ListItemAllFields/breakroleinheritance(copyRoleAssignments=false, clearSubscopes=true)

**Headers**:

json

CopyEdit

{

"Accept": "application/json;odata=verbose",

"Content-Type": "application/json;odata=verbose",

"X-RequestDigest": "<your-form-digest-value>"

}

This will break inheritance and set custom permissions.

**2. Plugin Approach (C#)**

* **Trigger**: Create a plugin on Assign, Create, or Update of a Dataverse record.
* **Use Microsoft Graph API or SharePoint CSOM** to set folder permissions.
* **Register the plugin in Pre-Operation to ensure real-time updates.**

**Example C# Code for Graph API**:

csharp

CopyEdit

GraphServiceClient graphClient = new GraphServiceClient(authProvider);

var permission = new Permission

{

Roles = new List<string> { "write" },

GrantedTo = new IdentitySet

{

User = new Identity { Id = "user-id" }

}

};

await graphClient.Sites["{site-id}"].Drive.Items["{folder-id}"].Permissions.Request().AddAsync(permission);

**Best Approach**

| **Approach** | **Pros** | **Cons** |
| --- | --- | --- |
| **Power Automate** | No code, easy setup | Slight delay (~5 sec), async by default |
| **Custom Plugin** | Fully synchronous, real-time updates | Requires C# development |

If you need **true synchronous behavior**, go with a **custom plugin + Graph API**. Otherwise, **Power Automate** is a simpler solution.

Would you like help with the **Graph API authentication** or **Power Automate Flow setup**?

**1. Register an Azure AD App (Client Credentials Flow)**

Since a Dataverse plugin runs in the background without user interaction, use the **client credentials flow** (app-only authentication).

**Steps to Register Azure AD App**

1. **Go to** [Azure Portal](https://portal.azure.com) → **Azure Active Directory** → **App registrations** → **New registration**.
2. **Enter a name**, choose **"Accounts in this organizational directory only"**, and register.
3. **Save** the **Application (client) ID** and **Directory (tenant) ID**.
4. **Create a Client Secret**:
   * Go to **Certificates & secrets** → **New client secret** → **Copy the value** (needed for authentication).
5. **Grant SharePoint API Permissions**:
   * Go to **API permissions** → **Add a permission** → **Microsoft Graph** → **Application permissions**.
   * Add:
     + Sites.Manage.All (to manage site permissions)
     + Sites.ReadWrite.All (to modify folders/files)
   * **Grant admin consent**.

**2. Get an Access Token in the Plugin (C#)**

Use **MSAL.NET** to obtain an **OAuth 2.0 access token** for Graph API calls.

**Install NuGet Packages in Plugin Project**

Add the following NuGet packages to your Dataverse plugin project:

shell

CopyEdit

Install-Package Microsoft.Identity.Client

Install-Package Microsoft.Graph

**C# Code for Authentication in the Plugin**

csharp

CopyEdit

using System;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Threading.Tasks;

using Microsoft.Identity.Client;

public class GraphAuthHelper

{

private static string tenantId = "your-tenant-id";

private static string clientId = "your-client-id";

private static string clientSecret = "your-client-secret";

private static string authority = $"https://login.microsoftonline.com/{tenantId}";

public static async Task<string> GetAccessTokenAsync()

{

IConfidentialClientApplication app = ConfidentialClientApplicationBuilder

.Create(clientId)

.WithClientSecret(clientSecret)

.WithAuthority(new Uri(authority))

.Build();

string[] scopes = new string[] { "https://graph.microsoft.com/.default" };

AuthenticationResult result = await app.AcquireTokenForClient(scopes).ExecuteAsync();

return result.AccessToken;

}

}

**3. Call Microsoft Graph API from the Plugin**

Once you get the **access token**, use it to call **Microsoft Graph API** to update SharePoint folder permissions.

**Example: Grant User Access to a SharePoint Folder**

csharp

CopyEdit

using System;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

public class SharePointHelper

{

private static string siteId = "your-site-id"; // Get this from Graph API

private static string folderId = "your-folder-id"; // Get this from Graph API

public static async Task GrantFolderAccessAsync(string userEmail)

{

string accessToken = await GraphAuthHelper.GetAccessTokenAsync();

string requestUrl = $"https://graph.microsoft.com/v1.0/sites/{siteId}/drive/items/{folderId}/permissions";

var payload = new

{

roles = new string[] { "write" }, // "read", "write", or "owner"

grantedToIdentities = new[]

{

new { user = new { email = userEmail } }

}

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", accessToken);

StringContent content = new StringContent(Newtonsoft.Json.JsonConvert.SerializeObject(payload), Encoding.UTF8, "application/json");

HttpResponseMessage response = await client.PostAsync(requestUrl, content);

if (!response.IsSuccessStatusCode)

{

throw new Exception($"Error granting access: {response.StatusCode}");

}

}

}

}

**4. Trigger the Graph API Call from Your Plugin**

Modify your Dataverse plugin to **call GrantFolderAccessAsync()** when a record is updated.

**Example: Plugin Code**

csharp

CopyEdit

using Microsoft.Xrm.Sdk;

using System;

using System.Threading.Tasks;

public class SharePointPermissionPlugin : IPlugin

{

public void Execute(IServiceProvider serviceProvider)

{

IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

if (context.MessageName.ToLower() == "update") // Run on update event

{

string userEmail = "user@example.com"; // Get user email from Dataverse record

Task.Run(async () => await SharePointHelper.GrantFolderAccessAsync(userEmail)).Wait();

}

}

}

**5. Register the Plugin in Dataverse**

1. **Build & Sign the Plugin DLL**
2. **Upload it to Dataverse** using **Plugin Registration Tool**
3. **Register the Plugin on Update/Create of a record**

**Conclusion**

* **Azure AD App** enables secure **client credentials authentication**.
* **MSAL.NET** retrieves **access tokens** for Graph API.
* **Microsoft Graph API** modifies SharePoint permissions.
* **Dataverse Plugin** ensures **real-time, synchronous execution**.