**PowerShell Script for OneDrive to Azure Blob Storage Automation with**

**Task Scheduler**

# Introduction

Automating script execution is a critical component of efficient system management. By leveraging PowerShell scripts and Windows Task Scheduler, organizations can ensure seamless execution of recurring tasks without manual intervention. This document provides a comprehensive guide on setting up Task Scheduler to run PowerShell scripts automatically at regular intervals. It includes instructions for registering an application in Azure Active Directory (Azure AD), installing required PowerShell modules, executing scripts using command-line methods, and configuring Task Scheduler for automated execution.

By following this guide, users will be able to:

* Securely authenticate and integrate with Microsoft services.
* Install and verify essential PowerShell modules.
* Execute PowerShell scripts manually or via Task Scheduler.
* Automate scheduled execution every 15 minutes for continuous operation.

# 1. App Registration and Retrieving Client, Tenant, and Secret ID

To integrate with Microsoft services via PowerShell, it is essential to register an application in Azure Active Directory (Azure AD) and obtain the necessary authentication credentials. This ensures secure and authorized access to Microsoft services.

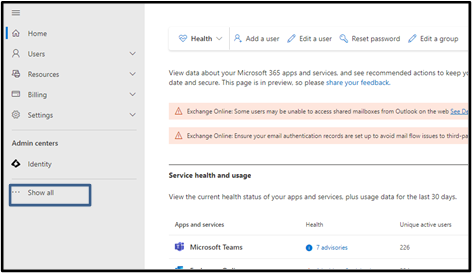
**Steps to Register an App in Azure AD:**

1. Navigate to the [Azure Portal](https://portal.azure.com/) and sign in with your credentials.

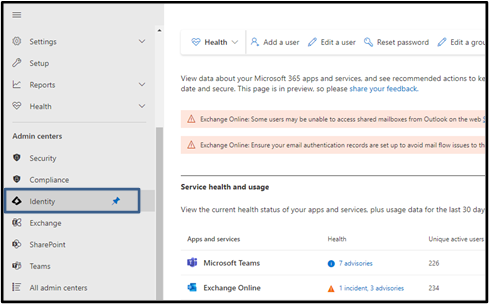
A screenshot of a computer

Description automatically generated

* Select “Show all” to find the extra options.



* Select the “Identity” option.

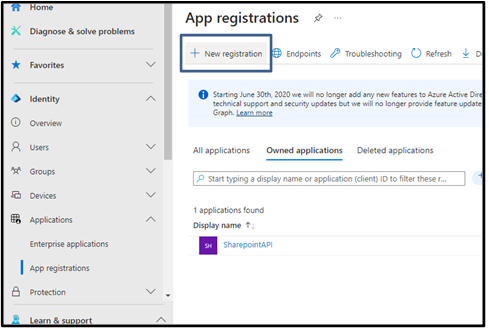


* Find the applications and select the option “App Registration”

A screenshot of a computer

Description automatically generated

* Select “New Registration” option.

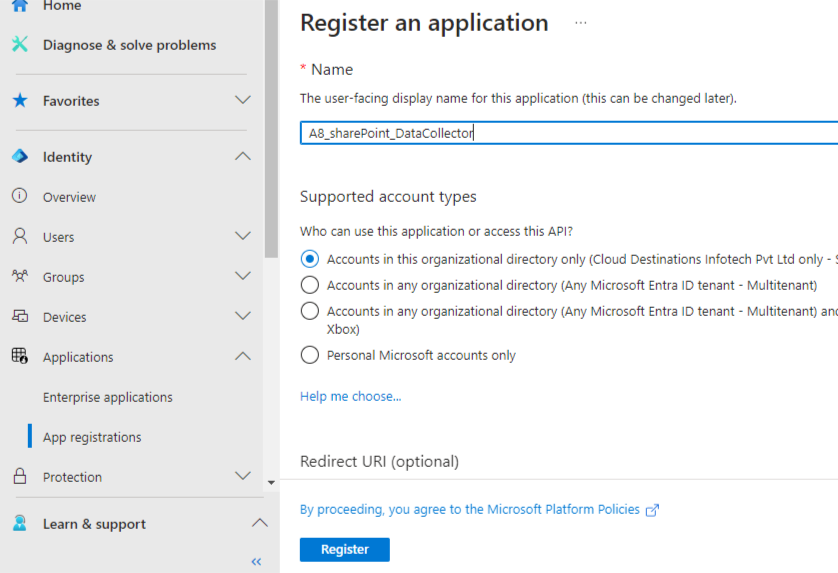


* Registration of the application in the field.

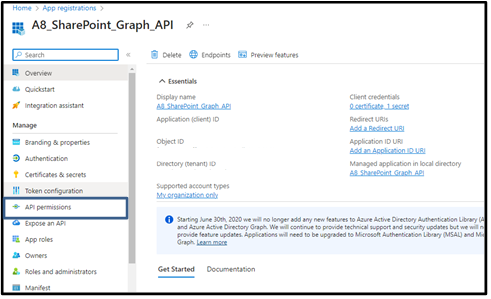
A screenshot of a computer

Description automatically generated

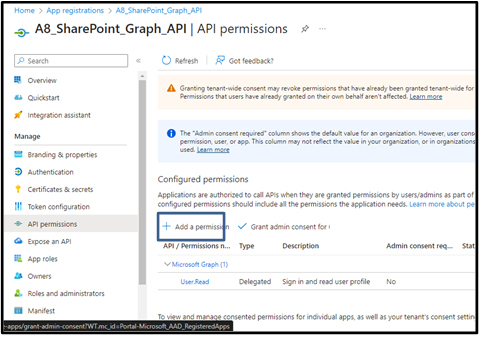
* Leave other options as default.
* Provide the Name for the App Registration



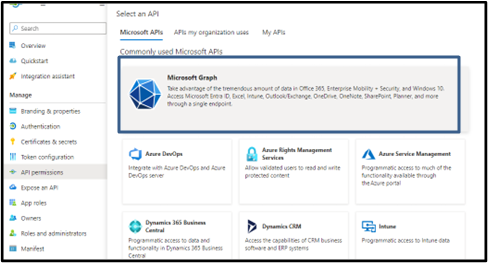
* select the “ API Permission “, to proceed with permission.



* Click on Add permission option, as marked.



* select ”MS - Graph “ permissions.



* Select the “Application permission” .

A screenshot of a computer

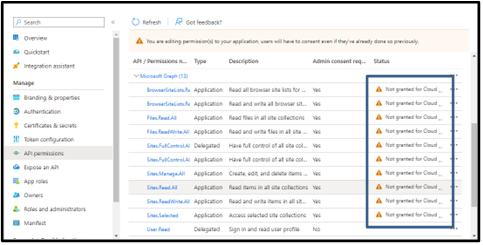
Description automatically generated

* Search for Permissions listed in this document below at page 12, click “Add Permission”.

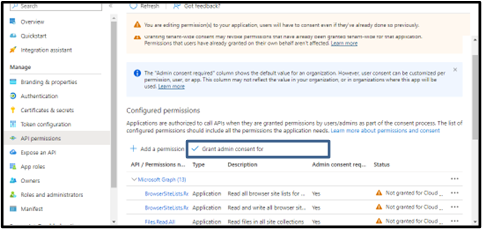
A screenshot of a computer

Description automatically generated

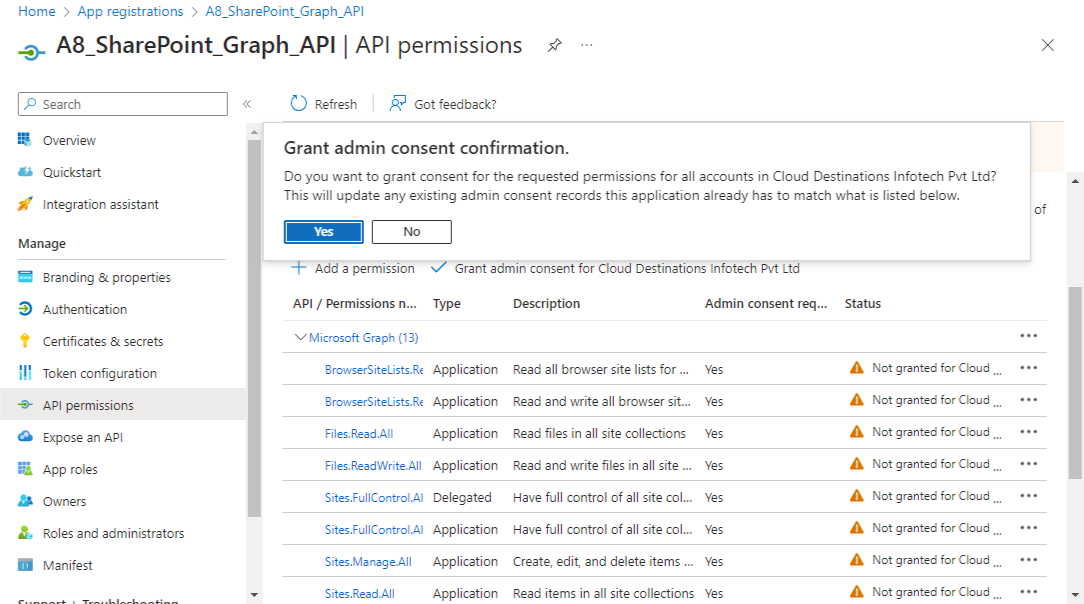
* Now all the permissions are selected but not yet granted the access.



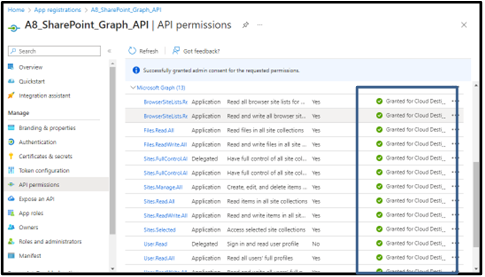
* Click on the “Grant admin consent” to grant the access.



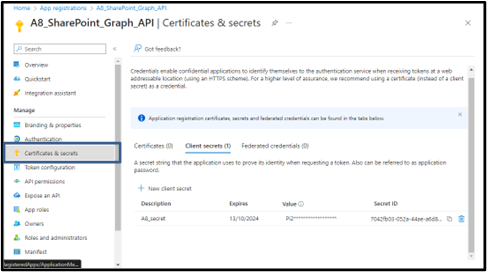
* Click on “YES”.



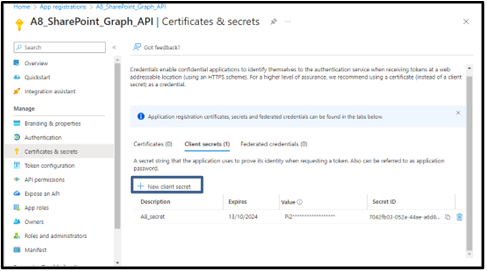
* “Tick” mark will be mentioned represents access granted.



* Click on the “Certificates & Secrets” option.



* Create new Client Secret values by clicking the marked option and copying the value. Store it in a secure place, as we will use it later

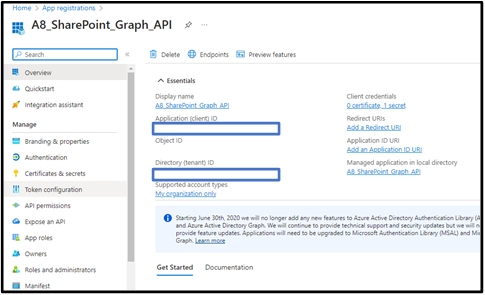


* Add the description for the Client Secret, click on “add” .

A screenshot of a computer

Description automatically generated

* Marked regions are the Tenant ID, Client ID, copy and Store it in secure place we will use it later



**Permissions need to be granted:**

1. BrowserSiteLists.ReadWrite.All
2. Files.ReadWrite.All
3. Sites.FullControl.All
4. Sites.Manage.All
5. Sites.ReadWrite.All
6. Sites.Selected
7. User.Read (Delegated)
8. User.ReadWrite.All

# 2. Installing Required PowerShell Modules

Before executing any scripts, the necessary PowerShell modules must be installed to interact with Microsoft services.

**Installing Required Modules:**

Run the following commands in an **elevated PowerShell session (Run as Administrator):**

# Install Microsoft Graph base module for authentication and API calls

Install-Module -Name Microsoft.Graph -Scope CurrentUser -Force

# Install Microsoft Graph Files module for OneDrive operations

Install-Module -Name Microsoft.Graph.Files -Scope CurrentUser -Force

# Install Azure PowerShell module for Azure Storage operations

Install-Module -Name Az -Scope CurrentUser -Force

**Verifying Installed Modules:**

To confirm successful installation, execute the following commands:

# Check Microsoft Graph module installation

Get-Module -ListAvailable -Name Microsoft.Graph

# Check Microsoft Graph Files module installation

Get-Module -ListAvailable -Name Microsoft.Graph.Files

# Check Azure PowerShell module installation

Get-Module -ListAvailable -Name Az

# 3. Executing PowerShell Script with Run Command

A PowerShell script can be executed manually or automatically using the command line. Ensure the script includes the necessary authentication and operational logic. Please save the script in your local system will use it later in task scheduler.

**Powershell Command:**

param (

    [Parameter(Mandatory=$true)]

    [string]$TenantId,

    [Parameter(Mandatory=$true)]

    [string]$ClientId,

    [Parameter(Mandatory=$true)]

    [string]$ClientSecret,

    [Parameter(Mandatory=$true)]

    [string]$UserUPN,

    [Parameter(Mandatory=$true)]

    [string]$StorageAccountName,

    [Parameter(Mandatory=$true)]

    [string]$ContainerName,

    [Parameter(Mandatory=$true)]

    [string]$StorageAccountKey,

    [Parameter(Mandatory=$true)]

    [string]$SourceOnedrivePath,

    [Parameter(Mandatory=$true)]

    [string]$DestinationBlobPath

)

try {

    # Authentication

    $secureClientSecret = ConvertTo-SecureString $ClientSecret -AsPlainText -Force

    $clientSecretCredential = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList $ClientId, $secureClientSecret

    Connect-MgGraph -TenantId $TenantId -ClientSecretCredential $clientSecretCredential

    # Get OneDrive ID

    $onedrive = Get-MgUserDrive -UserId $UserUPN

    if (-not $onedrive.Id) {

        throw "Drive ID not found for user $UserUPN"

    }

    $driveId = $onedrive.Id.Split(' ')[0]

    # Get root folder

    $currentItem = Get-MgDriveRoot -DriveId $driveId

    # Split source path into segments

    $pathSegments = $SourceOnedrivePath.Split('/') | Where-Object { $\_ -ne '' }

    # Navigate through folders

    foreach ($segment in $pathSegments) {

        $children = Get-MgDriveItemChild -DriveId $driveId -DriveItemId $currentItem.Id

        $currentItem = $children | Where-Object { $\_.Name -eq $segment }

        if (-not $currentItem) {

            throw "Path segment '$segment' not found in OneDrive path"

        }

    }

    # Verify it's a file

    if (-not $currentItem.File) {

        throw "The path '$SourceOnedrivePath' is a directory, not a file"

    }

    # Download the file

    $fileName = [System.IO.Path]::GetFileName($SourceOnedrivePath)

    $localTempFile = Join-Path -Path $env:TEMP -ChildPath $fileName

    Get-MgDriveItemContent -DriveId $driveId -DriveItemId $currentItem.Id -OutFile $localTempFile

    Write-Host "Downloaded '$SourceonedrivePath' to temporary file: $localTempFile"

    # Upload to Blob Storage

    $blobStorageContext = New-AzStorageContext -StorageAccountName $StorageAccountName -StorageAccountKey $StorageAccountKey

    Set-AzStorageBlobContent -File $localTempFile `

        -Container $ContainerName `

        -Blob $DestinationBlobPath `

        -Context $blobStorageContext `

        -Force

    Write-Host "Uploaded to Azure Blob Storage at path: '$DestinationBlobPath'"

    # Cleanup

    Remove-Item -Path $localTempFile -Force

    Write-Host "Temporary file cleaned up"

} catch {

    Write-Host "Error: $\_" -ForegroundColor Red

} finally {

    Disconnect-MgGraph -ErrorAction SilentlyContinue

}

**Run Command:**

.\scripttest.ps1

-TenantId "Add Tenant ID"

-ClientId "Add Client ID"

-ClientSecret "Add Client Secret"

-UserUPN "Add OneDrive user email ID"

-StorageAccountName "Add Storage Account Name"

-ContainerName "Add Container Name"

-StorageAccountKey "Add Storage Account Key"

-SourceOnedrivePath "Add Source OneDrive Path"

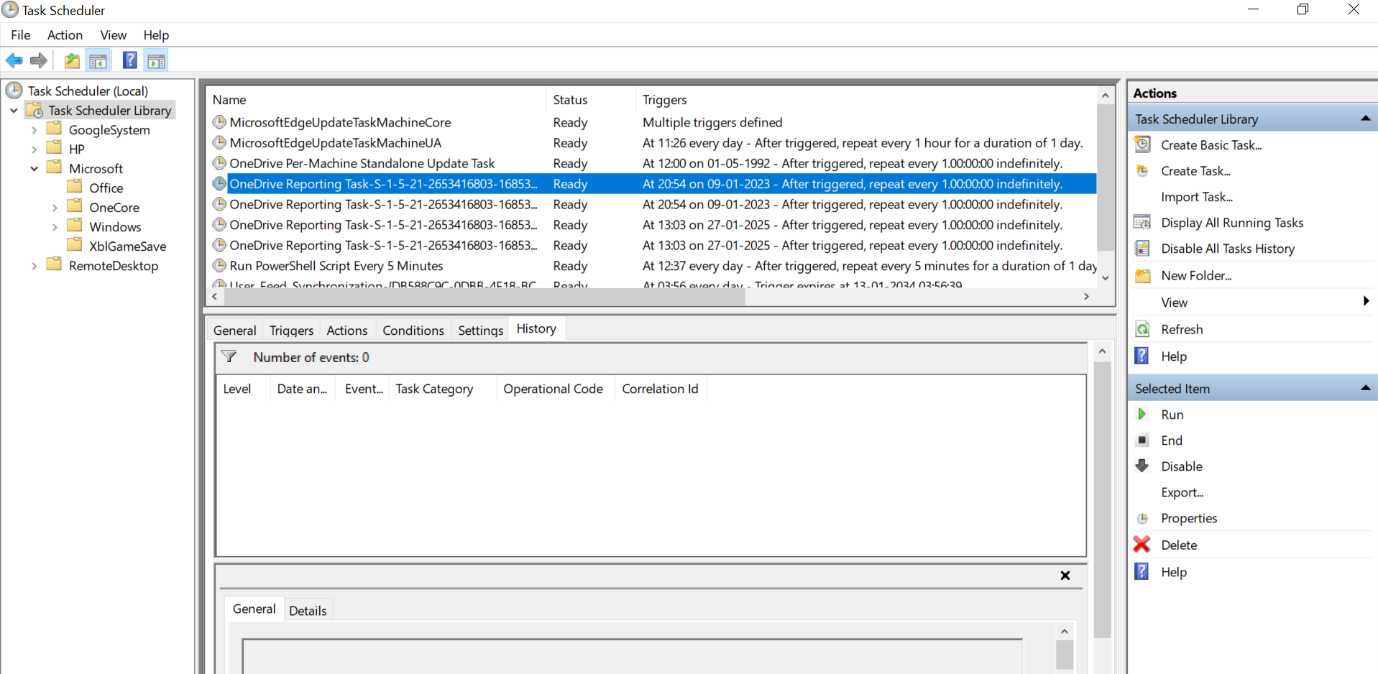
-DestinationBlobPath "Add Destination Blob Path"

# 4. Scheduling PowerShell Script Execution Using Task Scheduler

For automation, Task Scheduler can be configured to execute the PowerShell script at a fixed interval.

**Steps to Configure Task Scheduler:**

1. **Open Task Scheduler:**
   * Press Windows Key + R, type taskschd.msc, and press **Enter**.



1. **Create a New Task:**
   * Click **Create Task** in the right-hand panel.
   * Provide a meaningful name, such as **Run PowerShell Script Every 15 Minutes**.

A screenshot of a computer program

Description automatically generated

1. **Configure the Trigger:**
   * Go to the **Triggers** tab and click **New...**.
   * Select **On a schedule** and choose **Daily**.
   * Under **Advanced settings**, enable **Repeat task every** and set it to **15 minutes**.
   * Set the **Duration** to **Indefinitely**.
   * Click **OK**.

A screenshot of a computer program

Description automatically generated

1. **Configure the Action:**
   * Navigate to the **Actions** tab and click **New...**.
   * Set **Action** to **start a program**.
   * In the **Program/script** field, C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
   * In the **Add arguments** field, enter the full path to the PowerShell script along with necessary parameters.

ExecutionPolicy Bypass  
 -File "C:\Users\Administrator\Desktop\scripttest.ps1"   
-TenantId "Add Tenant ID"

-ClientId "Add Client ID”   
-ClientSecret "Add Client Secret Value"   
-UserUPN "OneDrive user email ID" -StorageAccountName " Add Storage Account Name"   
-ContainerName "Add container Name"   
-StorageAccountKey "Add Storage Access Key “  
-SourceOnedrivePath "Add Source Drive Path" eg.,(testshare/myBook.xlsx)  
-DestinationBlobPath "Add Destination Blob Path" eg., ("test1/myBook.xlsx")

A screenshot of a computer program

Description automatically generated

1. **Save the Task:**
   * Open the **Conditions** tab and modify settings if needed (e.g., ensure execution even when the system is on battery power).
   * Click **OK** to save and enable the scheduled task.

A screenshot of a computer error

Description automatically generated

Once configured, the script will automatically run every 15 minutes, ensuring consistent execution.

# 5. Conclusion

By following these steps, you have successfully:

* Registered an application in Azure AD and obtained authentication credentials.
* Installed and verified essential PowerShell modules for interacting with Microsoft Graph and Azure services.
* Created a PowerShell script to automate tasks.
* Configured Task Scheduler to execute the script at regular intervals without manual intervention.

This approach ensures a seamless, automated, and secure method for executing PowerShell scripts, improving efficiency and reliability in system management and data operations.