|  |  |  |
| --- | --- | --- |
| **Section** | **Duration (Minutes)** | **By** |
| Demo – C# OAuth 2.0 authorization code flow and client credentials flow with web app graph api experience | 20 | Chunlong |
| Demo – PowerShell with graph api over authorization code flow, client credentials flow, graph sdk with certificate | 20 | Victoria |
| Q&A | 10 | Chunlong & Victoria |

All source code of this training is attached: <https://github.com/Chunlong101/MicrosoftGraphApiDemo/tree/master/20211201>, we can get started with the source code from:

1. [Upload.aspx.cs](https://github.com/Chunlong101/MicrosoftGraphApiDemo/blob/master/20211201/Upload.aspx.cs) >> [Upload1\_Click](https://github.com/Chunlong101/MicrosoftGraphApiDemo/blob/98d332ba50fcba636364a968d9cae0b5b2a2dfb1/20211201/Upload.aspx.cs#L83): This C# method is trying to upload a file with [this graph api](https://docs.microsoft.com/en-us/graph/api/driveitem-put-content?view=graph-rest-1.0&tabs=http#http-request-to-upload-a-new-file) over authorization code flow.
2. [Upload.aspx.cs](https://github.com/Chunlong101/MicrosoftGraphApiDemo/blob/master/20211201/Upload.aspx.cs) >> [Upload2\_Click](https://github.com/Chunlong101/MicrosoftGraphApiDemo/blob/98d332ba50fcba636364a968d9cae0b5b2a2dfb1/20211201/Upload.aspx.cs#L152): This C# method is trying to upload a file with the same graph api above, but over client credentials flow.
3. [PowerShell Demo.ps1](https://github.com/Chunlong101/MicrosoftGraphApiDemo/blob/master/20211201/PowerShell%20Demo.ps1) >> [Obtain Authorization Code](https://github.com/Chunlong101/MicrosoftGraphApiDemo/blob/b488eb449dd93f094892986b8787a437243509d0/20211201/PowerShell%20Demo.ps1#L21): This powershell script is trying to get authorization code then use it to get access token and download a file with [this graph api](https://docs.microsoft.com/en-us/graph/api/driveitem-get-content?view=graph-rest-1.0&tabs=http).
4. [PowerShell Demo.ps1](https://github.com/Chunlong101/MicrosoftGraphApiDemo/blob/master/20211201/PowerShell%20Demo.ps1) >> [Call Graph API to add a new folder](https://github.com/Chunlong101/MicrosoftGraphApiDemo/blob/b488eb449dd93f094892986b8787a437243509d0/20211201/PowerShell%20Demo.ps1#L90): This powershell script is trying to create a folder with [this graph api](https://docs.microsoft.com/en-us/graph/api/driveitem-post-children?view=graph-rest-1.0&tabs=http) over client credentials follow.

In this session, we’re going to demo:

1. A web app with graph api authorization code flow and client credentials flow, below is an example for authorization code flow:

A picture containing text

Description automatically generated

First open a browser then navigate to <https://graphapidemosite.azurewebsites.net> and click on get started, after that we will see the page above, now if we click on “Button 1” then it will take us to azure identity endpoint like below:

Graphical user interface, application

Description automatically generated

We can sign in with the test account: [test@chunlong.onmicrosoft.com/2021Password1201](mailto:test@chunlong.onmicrosoft.com/2021Password1201), and do a user consent like below:

Graphical user interface, application, Word

Description automatically generated

After the user consent, it will redirect us back to our web site with authorization code, we can choose a file then click on “Button 1” again to upload it (the code behind will help automatically get the access token by that authorization code returned, then use that access token to call graph api):

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text

Description automatically generated Graphical user interface, text, application, email

Description automatically generated

We can use that same test account to access <https://chunlong.sharepoint.com/Shared%20Documents>, the file has been uploaded successfully by our graph api.

1. PowerShell with graph api over authorization code flow, client credentials flow, graph sdk with certificate**:**

**2.1 Delegate sample -** [https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-auth-code-flow](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fdocs.microsoft.com%2Fen-us%2Fazure%2Factive-directory%2Fdevelop%2Fv2-oauth2-auth-code-flow&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756894671%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=nQZB%2B%2FGcVRQsD5nk5pQGEHpKX3xewlXEswHegJT4dkE%3D&reserved=0)

2.1.1 - Download a file using PowerShell with Authorization Code Auth flow

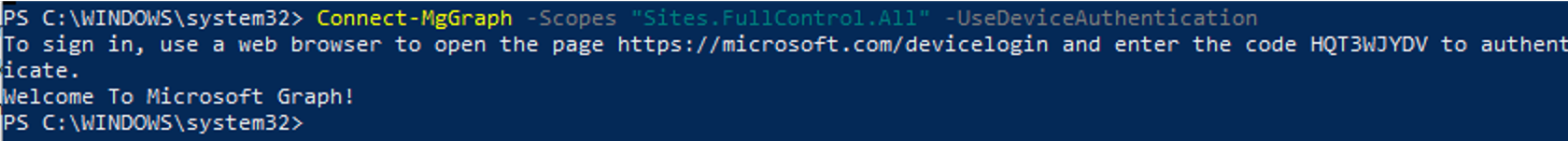
|  |
| --- |
| #Define Client Variables Here  #############################  $TenantId="ed3c1c81-f3be-495c-8028-f11b7ad6415a" #change tenant id to yours  $clientId = "87202bda-1a2a-4b6c-917b-df77c98c640d" #change client id to your registered app  $clientSecret = "yHS7Q~huAqZffVd.x9BS1CSuolqb2GMtaelDg" #change client secret to yours  $scope = [https://graph.microsoft.com/.default](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgraph.microsoft.com%2F.default&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756864798%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=SNIKTb1xMvRTaJfkDyZOCLWGF4l0pw8J54F10c4JekA%3D&reserved=0)  $redirectUri = <https://localhost> #this can be any URL, but need to make sure it has been added into your registered app  #UrlEncode variables for special characters  ###########################################  Add-Type -AssemblyName System.Web  $clientSecretEncoded = [System.Web.HttpUtility]::UrlEncode($clientSecret)  $redirectUriEncoded =  [System.Web.HttpUtility]::UrlEncode($redirectUri)  $scopeEncoded = [System.Web.HttpUtility]::UrlEncode($scope)  #Obtain Authorization Code  ##########################  Add-Type -AssemblyName System.Windows.Forms  $form = New-Object -TypeName System.Windows.Forms.Form -Property @{Width=440;Height=640}  $web  = New-Object -TypeName System.Windows.Forms.WebBrowser -Property @{Width=420;Height=600;Url=$url}  $url = [https://login.microsoftonline.com/$TenantId/oauth2/v2.0/authorize?response\_type=code&redirect\_uri=$redirectUriEncoded&client\_id=$clientId&scope=$scopeEncoded&prompt=admin\_consent](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Flogin.microsoftonline.com%2F%24TenantId%2Foauth2%2Fv2.0%2Fauthorize%3Fresponse_type%3Dcode%26redirect_uri%3D%24redirectUriEncoded%26client_id%3D%24clientId%26scope%3D%24scopeEncoded%26prompt%3Dadmin_consent&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756874766%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=DloDiDjNkJbrfaBcCbsw3QeUzuhL9o8yVfDgaHrEo3g%3D&reserved=0)  $DocComp  = {          $Global:uri = $web.Url.AbsoluteUri       if ($Global:uri -match "error=[^&]\*|code=[^&]\*") {$form.Close() }      }  $web.ScriptErrorsSuppressed = $true  $web.Add\_DocumentCompleted($DocComp)  $form.Controls.Add($web)  $form.Add\_Shown({$form.Activate()})  $form.ShowDialog() | Out-Null  $queryOutput = [System.Web.HttpUtility]::ParseQueryString($web.Url.Query)  $output = @{}  foreach($key in $queryOutput.Keys){      $output["$key"] = $queryOutput[$key]  }  $regex = '(?<=code=)(.\*)(?=&)'  $authCode  = ($uri | Select-string -pattern $regex).Matches[0].Value  #Get Access Token with obtained Auth Code  #########################################  $body = "grant\_type=authorization\_code&redirect\_uri=$redirectUri&client\_id=$clientId&client\_secret=$clientSecretEncoded&code=$authCode&resource=$resource"  $authUri = [https://login.microsoftonline.com/common/oauth2/token](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Flogin.microsoftonline.com%2Fcommon%2Foauth2%2Ftoken&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756874766%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=8UDSYui3raqJ7XV4EqzLmLvAZC8KaQS%2Fjm4IeLvspUI%3D&reserved=0)  $tokenResponse = Invoke-RestMethod -Uri $authUri -Method Post -Body $body -ErrorAction STOP  #Call Graph API to download a file  #########################################  #$DownloadUri=[https://graph.microsoft.com/v1.0/me/drive/root:/ee - Copy.xlsx:/content](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgraph.microsoft.com%2Fv1.0%2Fme%2Fdrive%2Froot%3A%2Fee%2520-%2520Copy.xlsx%3A%2Fcontent&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756884715%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=oES0t3Z4L3iD59k1XIKx9jOATPR%2FAzsafcChXhC6tHU%3D&reserved=0)  $DownloadUri = [https://graph.microsoft.com/v1.0/sites/ec884a3f-7f7e-460a-900b-39c61f8195be/drive/items/01DFEMAO44C7FIP2APCBEZFLF67OL6JAIZ/content](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgraph.microsoft.com%2Fv1.0%2Fsites%2Fec884a3f-7f7e-460a-900b-39c61f8195be%2Fdrive%2Fitems%2F01DFEMAO44C7FIP2APCBEZFLF67OL6JAIZ%2Fcontent&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756884715%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=YkQQvmfaDDsi75E5ecEhI1CV1I3uEF5C3WUr%2F74tBnM%3D&reserved=0)  $destinationFilePath = "C:\Users\menxia\Desktop\Files\Graph Test\Test1.pdf"  $header =@{      'Authorization' = "Bearer $($tokenResponse.access\_token)"  }  $results = Invoke-RestMethod -Uri $DownloadUri -Headers $header -Method Get -OutFile $destinationFilePath |

2.1.2 - Download a file using Graph PowerShell SDK with Authorization Code Auth flow

* Run command to connect Microsoft Graph in PowerShell (auth code flow)



* Or you can run command below to connect Microsoft Graph in PowerShell (using device code flow): [https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-device-code](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fdocs.microsoft.com%2Fen-us%2Fazure%2Factive-directory%2Fdevelop%2Fv2-oauth2-device-code&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756894671%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=sViViNTGiMwexx34pUEKAPOzIj8CiRKqDAXC7Hw%2FDUQ%3D&reserved=0)



Graphical user interface, application

Description automatically generated    Graphical user interface, text, application

Description automatically generated   Graphical user interface, text, application, email

Description automatically generated   Graphical user interface, text, application

Description automatically generated

* Verify the context for auth code flow:

Graphical user interface, text

Description automatically generated

* Verify the context for device code flow:

Text

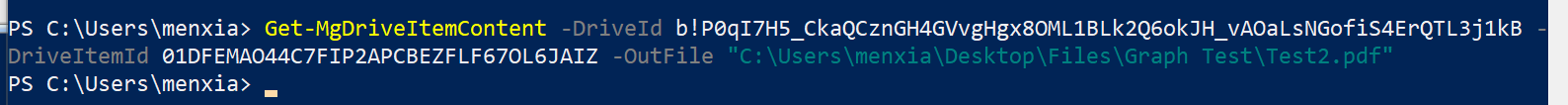
Description automatically generated

* Actually it is using this auto-registered app called Microsoft Graph PowerShell for doing the authentication:

Graphical user interface, application

Description automatically generated

* Call graph API to download a file:



* Reference: [Microsoft.Graph.Files Module | Microsoft Docs](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fdocs.microsoft.com%2Fen-us%2Fpowershell%2Fmodule%2Fmicrosoft.graph.files%2F%3Fview%3Dgraph-powershell-beta&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756904626%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=iPWeBRX0sPkXtHGWjft%2BHIVtcaB0XT5M2Bsl5GfVvuw%3D&reserved=0)

**2.2 Application only sample -** [**https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-client-creds-grant-flow**](https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-client-creds-grant-flow)

2.2.1 - Add a new folder with PowerShell in client credential grant flow (with secret)

|  |
| --- |
| #Define Client Variables Here  #############################  $TenantId='ed3c1c81-f3be-495c-8028-f11b7ad6415a'  #change tenant id to yours  $ClientId='87202bda-1a2a-4b6c-917b-df77c98c640d'  #change client id to your registered app  $ClientSecret='yHS7Q~huAqZffVd.x9BS1CSuolqb2GMtaelDg'  #change client secret to yours  $Body = @{      'tenant' = $TenantId   'client\_id' = $ClientId      'scope' = 'https://graph.microsoft.com/.default'      'client\_secret' = $ClientSecret      'grant\_type' = 'client\_credentials'  }  $Params = @{      'Uri' = [https://login.microsoftonline.com/$TenantId/oauth2/v2.0/token](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Flogin.microsoftonline.com%2F%24TenantId%2Foauth2%2Fv2.0%2Ftoken&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756904626%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=pORrioxfMcr5a7yguozDZlBBSFO2OwWdDys259uq3OE%3D&reserved=0)   #tenant name is also fine here, like xia053.onmicrosoft.com      'Method' = 'Post'      'Body' = $Body      'ContentType' = 'application/x-www-form-urlencoded'  }  #Get Access Token  ##########################  $AuthResponse = Invoke-RestMethod @Params  #Call Graph API to add a new folder  ####################################  $FolderName ='New Folder Name'  $SiteId = 'ec884a3f-7f7e-460a-900b-39c61f8195be'  $Uri=https://graph.microsoft.com/v1.0/sites/$SiteId/drive/root/children  $post = @"  {          "name":  "$FolderName",          "folder":  { }  }  "@  $header =@{      'Authorization' = "Bearer $($AuthResponse.access\_token)"      'Content-Type' = 'application/json'  }  $results = Invoke-WebRequest -Uri $Uri -Headers $header -Method Post -Body $post |

2.2.2 - Get an item with MSAL in client credential grant flow (with secret)

* + Install MSAL.PS module: [https://github.com/AzureAD/MSAL.PS](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2FAzureAD%2FMSAL.PS&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756914583%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=Ri0xdld1WTbr6nQKC%2Bi3Q6LFzNcGCgW9fQH8TPB6fXg%3D&reserved=0)

Logo, company name

Description automatically generated

* + Run command below to get an item by calling Graph API:

|  |
| --- |
| $s=Get-MsalToken -ClientId '87202bda-1a2a-4b6c-917b-df77c98c640d' -ClientSecret (ConvertTo-SecureString 'yHS7Q~huAqZffVd.x9BS1CSuolqb2GMtaelDg' -AsPlainText -Force)  -TenantId 'ed3c1c81-f3be-495c-8028-f11b7ad6415a' -Scopes 'https://graph.microsoft.com/.default' -Authority [https://login.microsoftonline.com/ed3c1c81-f3be-495c-8028-f11b7ad6415a](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Flogin.microsoftonline.com%2Fed3c1c81-f3be-495c-8028-f11b7ad6415a&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756914583%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=Munq54ZyVSy9DkICGM8%2FG4bAJ%2B1EtT0dnrOXaUp2RNk%3D&reserved=0)  $requestUri = [https://graph.microsoft.com/v1.0/sites/ec884a3f-7f7e-460a-900b-39c61f8195be/drive/items/01DFEMAO44C7FIP2APCBEZFLF67OL6JAIZ](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgraph.microsoft.com%2Fv1.0%2Fsites%2Fec884a3f-7f7e-460a-900b-39c61f8195be%2Fdrive%2Fitems%2F01DFEMAO44C7FIP2APCBEZFLF67OL6JAIZ&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756924538%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=pE8vxOwY7eH8wTbY%2F%2B1%2BT%2FlAnVdUJGmarjllw0R3nkA%3D&reserved=0)  $header =@{       'Authorization' = "Bearer $($s.AccessToken)"  }  $results = Invoke-RestMethod -Uri $requestUri -Headers $header -Method Get  $results |

2.2.3 - Get an item with PowerShell in client credential grant flow (with certificate)

|  |
| --- |
| #Define Client Variables Here  #############################  $ClientID = "87202bda-1a2a-4b6c-917b-df77c98c640d" # Application/Client id used for the cert name  $TenantID = "ed3c1c81-f3be-495c-8028-f11b7ad6415a" # tenant id is used for the cert name  $CertPassWord = "Access1" # Password used for creating the certificate  $aud = [https://login.microsoftonline.com/$TenantID/v2.0/](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Flogin.microsoftonline.com%2F%24TenantID%2Fv2.0%2F&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756924538%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=jEAJgHP2sF7SvN9MthJI5AzITSQsm9Y23ehgjGRCpI8%3D&reserved=0)  $CertificatePath\_Pfx = "C:\Users\menxia\Desktop\VicCert.pfx" # Path where the certificate is saved  $scope = [https://graph.microsoft.com/.default](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgraph.microsoft.com%2F.default&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756934484%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=I5%2FqfrncQhr2vrGSphHThlSbTtc0d0Yp5JK%2FEINppd0%3D&reserved=0)  #Install and load DLLs  #############################  Function JsonWeb-Libraries{      if ( ! (Get-ChildItem $HOME/IdentityModel/lib/Microsoft.IdentityModel.Logging.\* -erroraction ignore) ) {          install-package -Source nuget.org -ProviderName nuget -SkipDependencies Microsoft.IdentityModel.Logging -Destination $HOME/IdentityModel/lib -force -forcebootstrap | out-null      }      [System.Reflection.Assembly]::LoadFrom((Get-ChildItem $HOME/IdentityModel/lib/Microsoft.IdentityModel.Logging.\*/lib/net45/Microsoft.IdentityModel.Logging.dll).fullname) | out-null      if ( ! (Get-ChildItem $HOME/IdentityModel/lib/Microsoft.IdentityModel.Tokens.\* -erroraction ignore) ) {          install-package -Source nuget.org -ProviderName nuget -SkipDependencies Microsoft.IdentityModel.Tokens -Destination $HOME/IdentityModel/lib -force -forcebootstrap | out-null      }      [System.Reflection.Assembly]::LoadFrom((Get-ChildItem $HOME/IdentityModel/lib/Microsoft.IdentityModel.Tokens.\*/lib/net45/Microsoft.IdentityModel.Tokens.dll).fullname) | out-null      if ( ! (Get-ChildItem $HOME/IdentityModel/lib/Microsoft.IdentityModel.JsonWebTokens.\* -erroraction ignore) ) {          install-package -Source nuget.org -ProviderName nuget -SkipDependencies Microsoft.IdentityModel.JsonWebTokens -Destination $HOME/IdentityModel/lib -force -forcebootstrap | out-null      }      [System.Reflection.Assembly]::LoadFrom((Get-ChildItem $HOME/IdentityModel/lib/Microsoft.IdentityModel.JsonWebTokens.\*/lib/net45/Microsoft.IdentityModel.JsonWebTokens.dll).fullname) | out-null  }  #Get token with certificate  #############################  JsonWeb-Libraries  $x509cert = [System.Security.Cryptography.X509Certificates.X509Certificate2]::new($CertificatePath\_Pfx, $CertPassWord)  $claims = new-object 'System.Collections.Generic.Dictionary[String, Object]'  $claims['aud'] = $aud  $claims['iss' ] = $ClientID  $claims['sub'] = $ClientID  $claims['jti'] = [GUID]::NewGuid().ToString('D')    $signingCredentials = [Microsoft.IdentityModel.Tokens.X509SigningCredentials]::new($x509cert)  $securityTokenDescriptor = [Microsoft.IdentityModel.Tokens.SecurityTokenDescriptor]::new()  $securityTokenDescriptor.Claims = $claims  $securityTokenDescriptor.SigningCredentials = $signingCredentials  $tokenHandler = [Microsoft.IdentityModel.JsonWebTokens.JsonWebTokenHandler]::new()  $clientAssertion = $tokenHandler.createToken($securityTokenDescriptor)  #Get access token using certificate  ###########################################  $Body = @{      'client\_id' = $ClientId      'scope' = 'https://graph.microsoft.com/.default'      'client\_assertion\_type' = 'urn:ietf:params:oauth:client-assertion-type:jwt-bearer'      'client\_assertion' = $clientAssertion      'grant\_type' = 'client\_credentials'  }  $Params = @{      'Uri' = [https://login.microsoftonline.com/$TenantId/oauth2/v2.0/token](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Flogin.microsoftonline.com%2F%24TenantId%2Foauth2%2Fv2.0%2Ftoken&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756934484%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=O27nwaH6UvKe0T%2BwyPWIMs7cOa4slyL1SVprX6A2s3c%3D&reserved=0)   #tenant name is also fine here, like xia053.onmicrosoft.com      'Method' = 'Post'      'Body' = $Body      'ContentType' = 'application/x-www-form-urlencoded'  }  $AuthResponse = Invoke-RestMethod @Params  #Call Graph API to get an item  ##################################  $requestUri = [https://graph.microsoft.com/v1.0/sites/ec884a3f-7f7e-460a-900b-39c61f8195be/drive/items/01DFEMAO44C7FIP2APCBEZFLF67OL6JAIZ](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgraph.microsoft.com%2Fv1.0%2Fsites%2Fec884a3f-7f7e-460a-900b-39c61f8195be%2Fdrive%2Fitems%2F01DFEMAO44C7FIP2APCBEZFLF67OL6JAIZ&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756944452%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=N4vyAUTnd6lq7zooutdi6C%2F3ushcYxMdhX4qSASAnZY%3D&reserved=0)  $header =@{      'Authorization' = "Bearer $($AuthResponse.access\_token)"  }  $results = Invoke-RestMethod -Uri $requestUri -Headers $header -Method Get  $results |

2.2.4 - Download a file using Graph PowerShell SDK (with certificate)

* Note: we do not support client secret due to security reason: [https://github.com/microsoftgraph/msgraph-sdk-powershell/issues/686](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fmicrosoftgraph%2Fmsgraph-sdk-powershell%2Fissues%2F686&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756944452%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=PppZmOWqFYSDdvUnwM9pvGxM6SLRcbaVhNnOpWle5fY%3D&reserved=0)

Graphical user interface, text, application

Description automatically generated

* + Create a self-signed certfiicate for test: [https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-create-self-signed-certificate](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fdocs.microsoft.com%2Fen-us%2Fazure%2Factive-directory%2Fdevelop%2Fhowto-create-self-signed-certificate&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756954409%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=qPXe7p66PSfiyoPBYXZRWDPpwx6k1aq5nJXIMldBj6w%3D&reserved=0)

$cert = New-SelfSignedCertificate -Subject "CN=VicCert" -CertStoreLocation "Cert:\CurrentUser\My" -KeyExportPolicy Exportable -KeySpec Signature -KeyLength 2048 -KeyAlgorithm RSA -HashAlgorithm SHA256



* Export-Certificate -Cert $cert -FilePath "C:\Users\menxia\Desktop\VicCert.cer"

Graphical user interface, text

Description automatically generated

* $mypwd = ConvertTo-SecureString -String "Access1" -Force -AsPlainText

Export-PfxCertificate -Cert $cert -FilePath "C:\Users\menxia\Desktop\VicCert.pfx" -Password $mypwd

Graphical user interface, text

Description automatically generated

* + Register an application in Azure AD and upload the certificate (.cer) into the registered app.

Graphical user interface, application

Description automatically generated

* + Authenticate to Graph in PowerShell:

Connect-MgGraph -ClientID 87202bda-1a2a-4b6c-917b-df77c98c640d -TenantId ed3c1c81-f3be-495c-8028-f11b7ad6415a -CertificateName "CN=VicCert"

Text

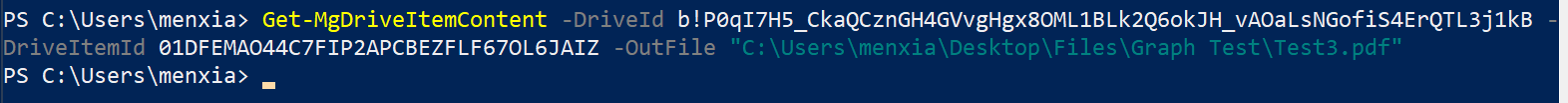
Description automatically generated

* Connect-MgGraph -ClientID 87202bda-1a2a-4b6c-917b-df77c98c640d -TenantId ed3c1c81-f3be-495c-8028-f11b7ad6415a -CertificateThumbprint 8F1973A927B0DCFA4E4A71B251A3B3CE98C48CEB

Text

Description automatically generated

* + Call Graph API to download a file:



* + References: [https://docs.microsoft.com/en-us/graph/powershell/navigating](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fdocs.microsoft.com%2Fen-us%2Fgraph%2Fpowershell%2Fnavigating&data=04%7C01%7Cchunlonl%40microsoft.com%7C171227f4d96446d75c6708d9b0ad53ed%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637735080756954409%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=GOfHS7rRAW%2BkJxSv6HLtTCixl57i9KlsGpsVZKfESiY%3D&reserved=0)

Graphical user interface, text, application, email

Description automatically generated