Automation Of Group Licensing

Version	Notes
V1	Initial Draft

You must sign in with an account that has admin rights to assign licenses within your tenant.

Key

Group Licensing Automation

The purpose of this work is to automate group licensing for the Newly created groups and by virtue assign to that user when members added to the group and finally when the engagement status moves to closed state revoke the licenses from the group

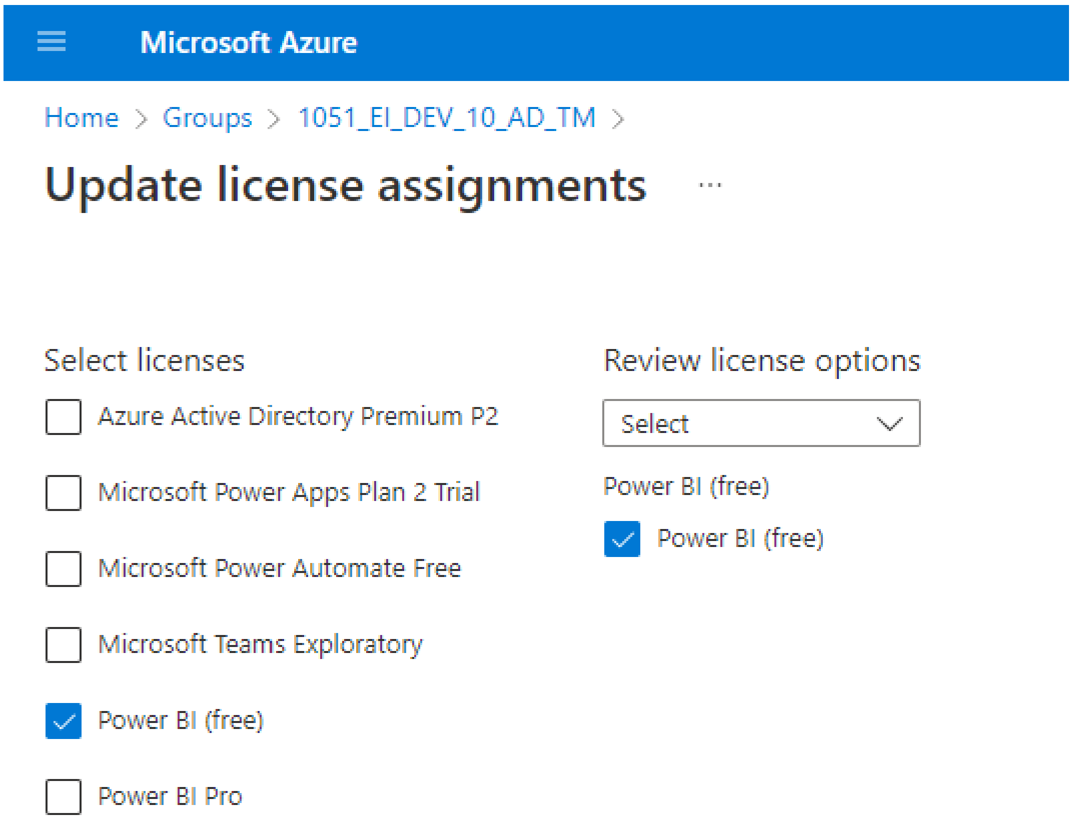
Case

Most organizations are using group-based licensing in Azure Active Directory. This is often integrated with the onboarding process of the users. But there are some use cases where you have some non-standard licenses attached to your tenant that you hand out on demand. You could still use group-based licensing, but users are added manually to the group

In Manual usually this is how it happens when Groups are created assignment of licenses are follows the UI as below Each group must be configured for that needs

In My example I will create One group called **1051_EI_DEV_AD_TM** and assign licenses manually and see once I have added users to that group how they get it

I assigned license manually as below


Microsoft Azure
Home > Groups > 1051_EI_DEV_10_AD_TM >
Update license assignments ...

Select licenses

- [ ] Azure Active Directory Premium P2
- [ ] Microsoft Power Apps Plan 2 Trial
- [ ] Microsoft Power Automate Free
- [ ] Microsoft Teams Exploratory
- [✓] Power BI (free)
- [ ] Power BI Pro

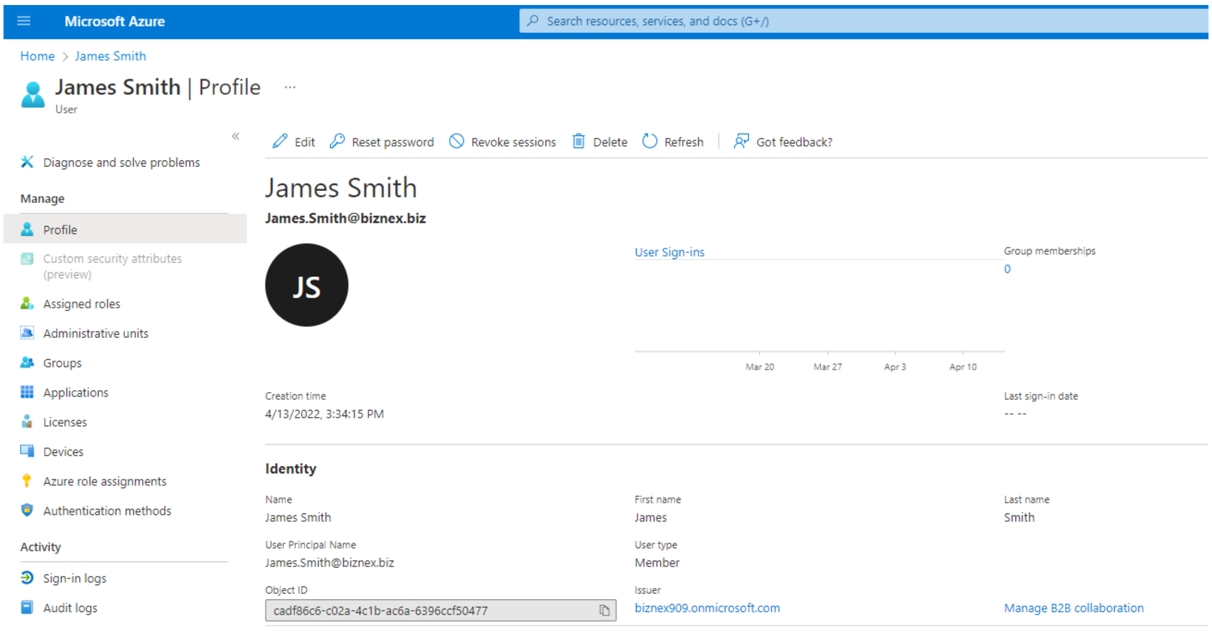
Review license options

Select

Power BI (free)

- [✓] Power BI (free)

Now I am adding New User in the AD Called James Smith and will add his member ship to above group



James Smith
James.Smith@biznex.biz

JS

User Sign-ins

Group memberships
0

Mar 20 Mar 27 Apr 3 Apr 10

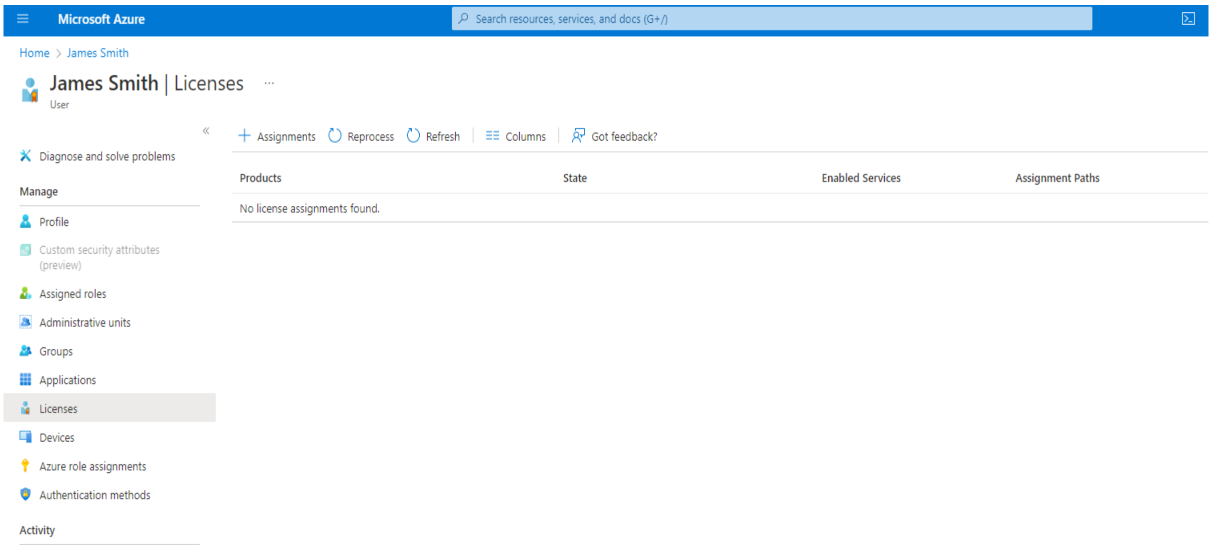
Creation time
4/13/2022, 3:34:15 PM

Last sign-in date
-- --

Identity

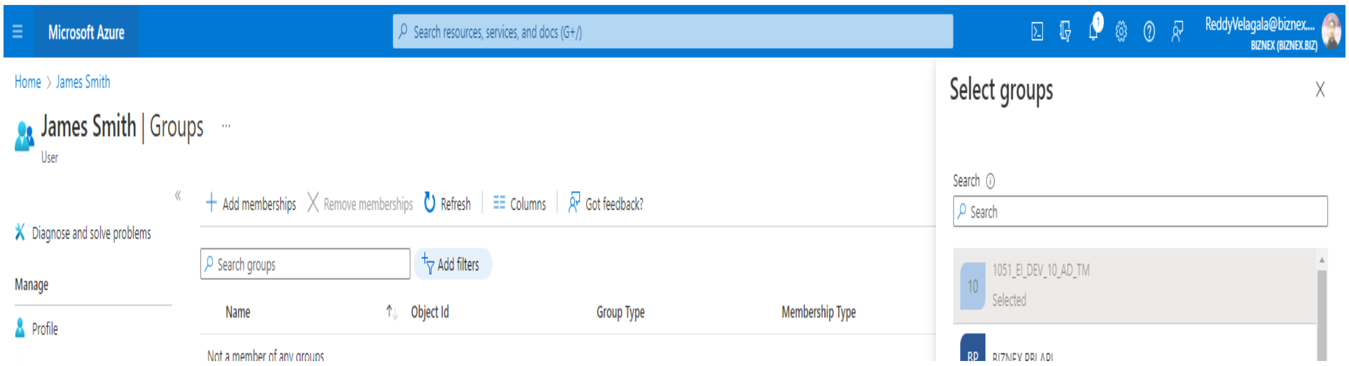
Name	First name	Last name
James Smith	James	Smith
User Principal Name	User type	
James.Smith@biznex.biz	Member	
Object ID	Issuer	
cadf86c6-c02a-4c1b-ac6a-6396ccf50477	biznex909.onmicrosoft.com	Manage B2B collaboration

Before when adding his member ship check licenses has, he got any



Products	State	Enabled Services	Assignment Paths
No license assignments found.			

Now add his membership to the above group



Home > James Smith

James Smith | Groups

User

Diagnose and solve problems

Manage

Profile

+ Add memberships Remove memberships Refresh Columns Got feedback?

Name	Object Id	Group Type	Membership Type
Not a member of any groups			

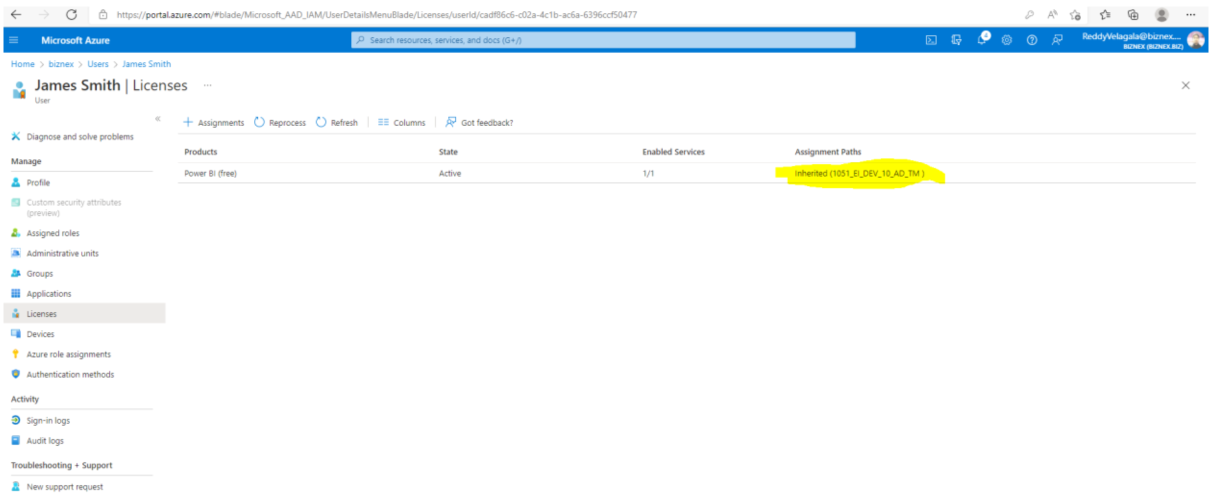
Select groups

Search

1051_EI_DEV_10_AD_TM
Selected

BIZNEX PBI API

Once double check the User Licesnes again – Power BI Free license is already added



https://portal.azure.com/#blade/Microsoft_AAD_IAM/UserDetailsMenuBlade/Licenses/userId/cadf86c6-c02a-4c1b-ac6a-6396ccf50477

Home > biznex > Users > James Smith

James Smith | Licenses

User

+ Assignments Reprocess Refresh Columns Got feedback?

Products	State	Enabled Services	Assignment Paths
Power BI (free)	Active	1/1	Inherited (1051_EI_DEV_10_AD_TM)

Diagnose and solve problems

Manage

Profile

Custom security attributes (preview)

Assigned roles

Administrative units

Groups

Applications

Licenses

Devices

Azure role assignments

Authentication methods

Activity

Sign-in logs

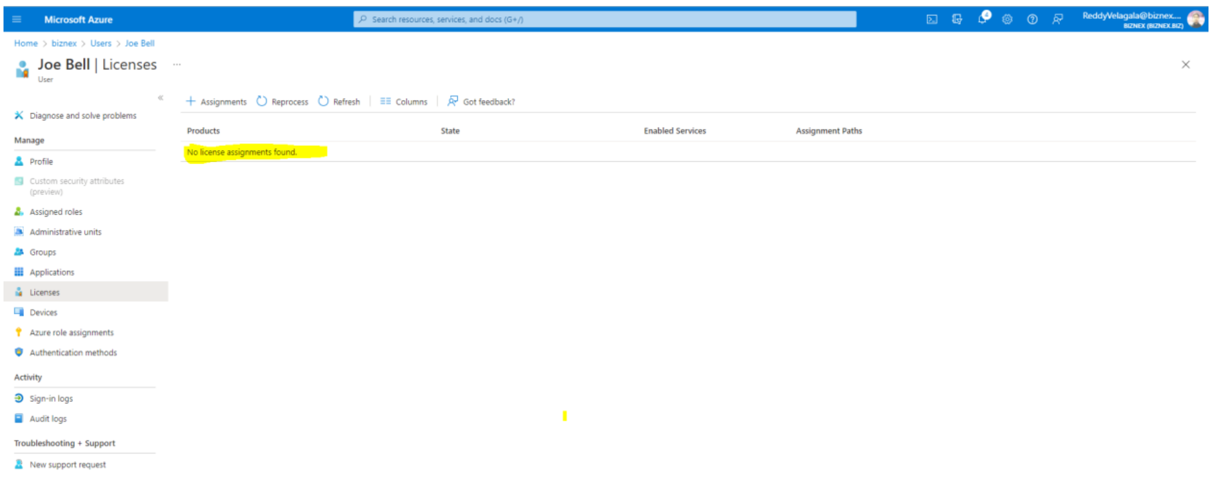
Audit logs

Troubleshooting + Support

New support request

Highlighted above says it is inherited thriough group

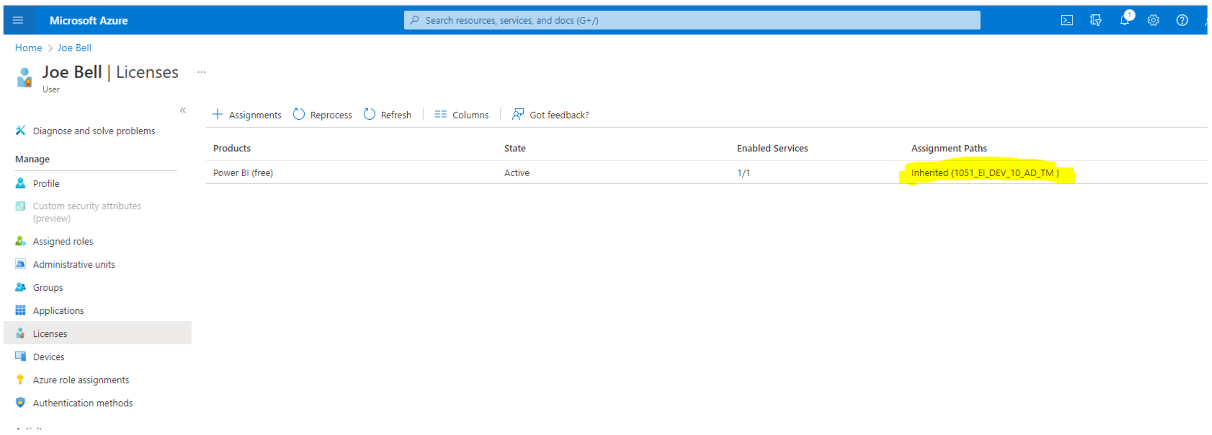
Add another user Joe.Bell@biznex.biz and to the membership of the above created group, before adding to the group



Home > biznex > Users > Joe Bell

Joe Bell | Licenses

Products	State	Enabled Services	Assignment Paths
No license assignments found.			



Home > Joe Bell

Joe Bell | Licenses

Products	State	Enabled Services	Assignment Paths
Power BI (free)	Active	1/1	Inherited (1051_EI_DEV_10_AD_TM)

All manually working as expected , Now simulating this through Powershell code

Group Name	1051_EI_DEV_10_AD_DV
Power BI License	$License.SkuId = "a403ebcc-fae0-4ca2-8c8c-7a907fd6c235"
Users	

Script to check licenses for the group

Output from Graph Query

License Details

https://graph.microsoft.com/v1.0/me/licenseDetails

```
{
            "id": "KztgFOWDi0eBFyyfTm7S3MzrA6Tg-qJMjIx6kH_WwjU",
            "skuId": "a403ebcc-fae0-4ca2-8c8c-7a907fd6c235",
            "skuPartNumber": "POWER_BI_STANDARD",
            "servicePlans": [
                {
                    "servicePlanId": "113feb6c-3fe4-4440-bddc-54d774bf0318",
                    "servicePlanName": "EXCHANGE_S_FOUNDATION",
                    "provisioningStatus": "Success",
                    "appliesTo": "Company"
                },
                {
                    "servicePlanId": "2049e525-b859-401b-b2a0-e0a31c4b1fe4",
                    "servicePlanName": "BI_AZURE_P0",
                    "provisioningStatus": "Success",
                    "appliesTo": "User"
                }
            ]
        }
```

****

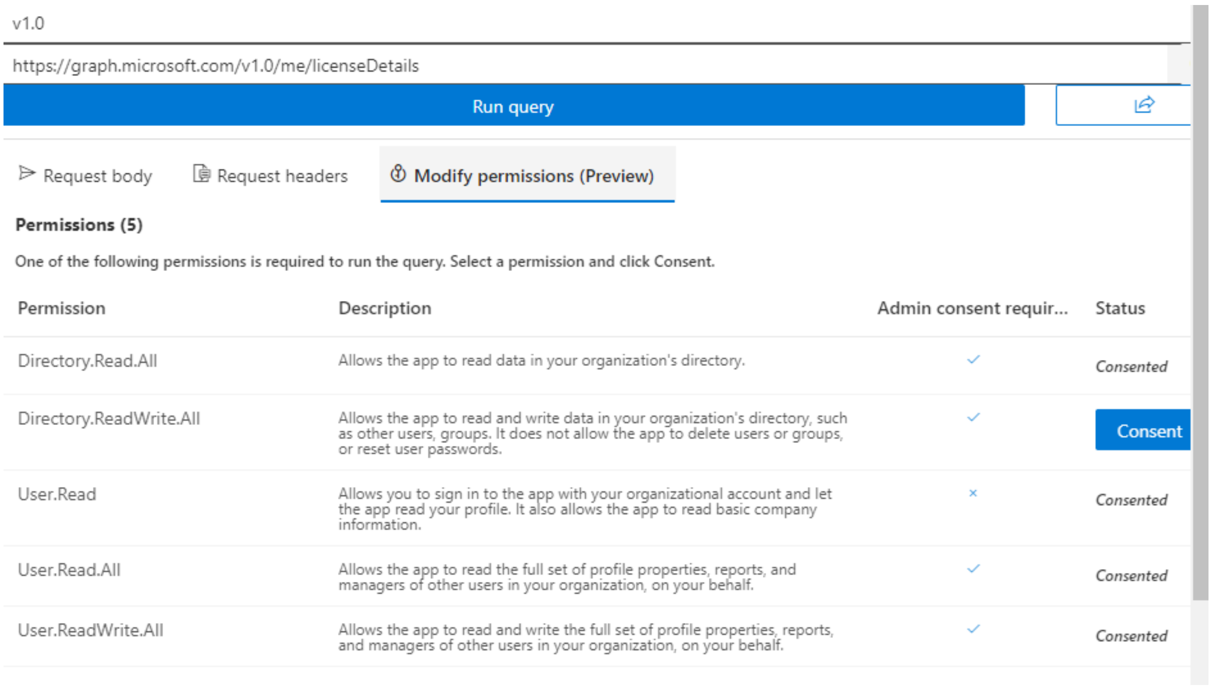
Tenant Id : 14603b2b-83e5-478b-8117-2c9f4e6ed2dc

This first call to graph will list out all information about the license SKU's that exists in your tenant.

```
GET https://graph.microsoft.com/beta/subscribedSkus
```

```
using namespace System.Net

# Input bindings are passed in via param block.
```

```powershell
param($Request, $TriggerMetadata)

# Write to the Azure Functions log stream.
Write-Host "PowerShell HTTP trigger function processed a request."

# Interact with query parameters or the body of the request.
$name = $Request.Query.Name
if (-not $name) {
    $name = $Request.Body.Name
}

$body = "This HTTP triggered function executed successfully. Pass a name in
the query string or in the request body for a personalized response."

if ($name) {
    $body = "Hello, $name. This HTTP triggered function executed
successfully."
}

# Application (client) ID, tenant Name and secret
$clientId = "xxx"
$tenantName = "xxxx"
$clientSecret = "xxxx"
$resource = "https://graph.microsoft.com/"
$GroupId="xxxx"

$ReqTokenBody = @{
    Grant_Type    = "client_credentials"
    Scope         = "https://graph.microsoft.com/.default"
    client_Id     = "$clientID"
    Client_Secret = $clientSecret
}
$TokenResponse=Invoke-RestMethod -Uri
"https://login.microsoftonline.com/$tenantName/Oauth2/V2.0/token" -Method POST
-Body $ReqTokenBody
$apiUrl='https://graph.microsoft.com/beta/subscribedSkus'
$Data=Invoke-RestMethod -Headers @{Authorization="Bearer
$($TokenResponse.access_token)" } -Uri $apiUrl -Method Get

#Write ("Azure AAD Group Id : $Group_Id")

# Associate values to output bindings by calling 'Push-OutputBinding'.
Push-OutputBinding -Name Response -Value ([HttpResponseContext]@{
    StatusCode = [HttpStatusCode]::OK
    Body = $Data
})
```



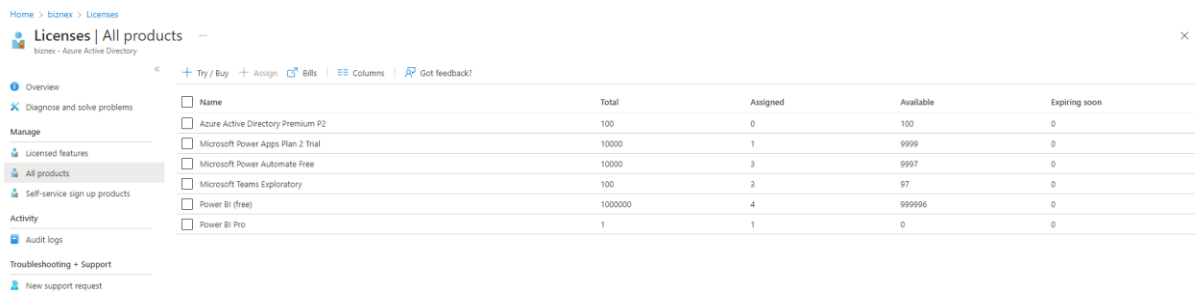
v1.0

https://graph.microsoft.com/v1.0/me/licenseDetails

Permissions (5)

One of the following permissions is required to run the query. Select a permission and click Consent.

Permission	Description	Admin consent requir...	Status
Directory.Read.All	Allows the app to read data in your organization's directory.	✓	Consented
Directory.ReadWrite.All	Allows the app to read and write data in your organization's directory, such as other users, groups. It does not allow the app to delete users or groups, or reset user passwords.	✓	Consent
User.Read	Allows you to sign in to the app with your organizational account and let the app read your profile. It also allows the app to read basic company information.	×	Consented
User.Read.All	Allows the app to read the full set of profile properties, reports, and managers of other users in your organization, on your behalf.	✓	Consented
User.ReadWrite.All	Allows the app to read and write the full set of profile properties, reports, and managers of other users in your organization, on your behalf.	✓	Consented



Licenses | All products

biznex - Azure Active Directory

Name	Total	Assigned	Available	Expiring soon
Azure Active Directory Premium P2	100	0	100	0
Microsoft Power Apps Plan 2 Trial	10000	1	9999	0
Microsoft Power Automate Free	10000	3	9997	0
Microsoft Teams Exploratory	100	3	97	0
Power BI (free)	1000000	4	999996	0
Power BI Pro	1	1	0	0

```
$reqTokenBody = @{ Grant_Type = "Password" client_Id =
$credAzureAplication.UserName Client_Secret =
$credAzureAplication.GetNetworkCredential().Password Username =
$credUser.UserName Password = $credUser.GetNetworkCredential().Password
Scope = "https://graph.microsoft.com/.default" }
```

```
$tokenResponse = Invoke-RestMethod -Uri
"https://login.microsoftonline.com/$tenantName/oauth2/v2.0/token" -
Method POST -Body $reqTokenBody
```

```
$graphApiHeader = @{ Authorization = "Bearer
$($tokenResponse.access_token)" }
```

Other Routes

```
PS C:\WINDOWS\system32> # Get List of licenses in Tenant
Get-AzureADSubscribedSku | Select Sku*,*Units

SkuId                                SkuPartNumber      ConsumedUnits PrepaidUnits
-----                                -------------      ------------- ------------
f8a1db68-be16-40ed-86d5-cb42ce701560 POWER_BI_PRO                   1 class
LicenseUnitsDetail {...
f30db892-07e9-47e9-837c-80727f46fd3d FLOW_FREE                      2 class
LicenseUnitsDetail {...
dcb1a3ae-b33f-4487-846a-a640262fadf4 POWERAPPS_VIRAL                1 class
LicenseUnitsDetail {...
84a661c4-e949-4bd2-a560-ed7766fcaf2b AAD_PREMIUM_P2                 0 class
LicenseUnitsDetail {...
a403ebcc-fae0-4ca2-8c8c-7a907fd6c235 POWER_BI_STANDARD              6 class
LicenseUnitsDetail {...
710779e8-3d4a-4c88-adb9-386c958d1fdf TEAMS_EXPLORATORY              5 class
LicenseUnitsDetail {...



PS C:\WINDOWS\system32> $licenses = Get-AzureADSubscribedSku

# Get all tenant users
$Users = Get-AzureADUser -All $true

# Set up Power BI (Free) License object
$License = New-Object -TypeName Microsoft.Open.AzureAD.Model.AssignedLicense
$License.SkuId = "a403ebcc-fae0-4ca2-8c8c-7a907fd6c235"

$LicensesToAssign = New-Object -TypeName
Microsoft.Open.AzureAD.Model.AssignedLicenses
$LicensesToAssign.AddLicenses = $License


# Loop through each user to verify if they have a Power BI (Free) license
# If they do not, assign the license to that account.
ForEach($User in $Users)
{
    # Check to see if they already have that license
    $AssignedLicenses = $User | Select -ExpandProperty AssignedLicenses | Where
{$_.SkuId -eq "a403ebcc-fae0-4ca2-8c8c-7a907fd6c235"}

    # We only want to assign a license if they
    # don't have one assigned.
    If ($AssignedLicenses.Count -lt 1)
    {
          Set-AzureADUserLicense -ObjectId $User.ObjectId -AssignedLicenses
$LicensesToAssign
          Write-Host "License added." -foregroundcolor cyan
    }
    Else {Write-Host "License already exists." -foregroundcolor cyan}
}
License already exists.
License already exists.
License already exists.
License added.
License already exists.
License already exists.
```