# If you don't have the Microsoft Graph PowerShell modules already installed, install the Microsoft.Graph.Users module and others by using this command:

Install-Module Microsoft.Graph

# If you already have the modules installed, ensure that you're using a recent version:  
Update-Module microsoft.graph.users,microsoft.graph.identity.governance,microsoft.graph.applications

# Connect to Microsoft Entra ID:  
$msg = Connect-MgGraph -ContextScope Process -Scopes "User.ReadWrite.All,Application.ReadWrite.All,AppRoleAssignment.ReadWrite.All,EntitlementManagement.ReadWrite.All"

# Read the list of users obtained. If the file obtained is named SMBC3.csv and is located in the current directory, enter below commands (if situated in another place, change the path as appropriate):  
$filename = " .\SMBC3.csv"  
$dbusers = Import-Csv -Path $filename -Encoding UTF8

# If the default mapping attribute (primary key/identificatior) is USERID and you want to match it with the Microsoft Entra ID attribute userPrincipalName, enter this command:

$db\_match\_column\_name = "USERID"

$azuread\_match\_attr\_name = "userPrincipalName"

#Retrieve the IDs of those users in Microsoft Entra ID:  
$dbu\_not\_queried\_list = @()

$dbu\_not\_matched\_list = @()

$dbu\_match\_ambiguous\_list = @()

$dbu\_query\_failed\_list = @()

$azuread\_match\_id\_list = @()

$azuread\_not\_enabled\_list = @()

$dbu\_values = @()

$dbu\_duplicate\_list = @()

foreach ($dbu in $dbusers) {

if ($null -ne $dbu.$db\_match\_column\_name -and $dbu.$db\_match\_column\_name.Length -gt 0) {

$val = $dbu.$db\_match\_column\_name

$escval = $val -replace "'","''"

if ($dbu\_values -contains $escval) { $dbu\_duplicate\_list += $dbu; continue } else { $dbu\_values += $escval }

$filter = $azuread\_match\_attr\_name + " eq '" + $escval + "'"

try {

$ul = @(Get-MgUser -Filter $filter -All -Property Id,accountEnabled -ErrorAction Stop)

if ($ul.length -eq 0) { $dbu\_not\_matched\_list += $dbu; } elseif ($ul.length -gt 1) {$dbu\_match\_ambiguous\_list += $dbu } else {

$id = $ul[0].id;

$azuread\_match\_id\_list += $id;

if ($ul[0].accountEnabled -eq $false) {$azuread\_not\_enabled\_list += $id }

}

} catch { $dbu\_query\_failed\_list += $dbu }

} else { $dbu\_not\_queried\_list += $dbu }

}

# The following PowerShell script will display the counts of records that weren't located:  
$dbu\_not\_queried\_count = $dbu\_not\_queried\_list.Count

if ($dbu\_not\_queried\_count -ne 0) {

Write-Error "Unable to query for $dbu\_not\_queried\_count records as rows lacked values for $db\_match\_column\_name."

}

$dbu\_duplicate\_count = $dbu\_duplicate\_list.Count

if ($dbu\_duplicate\_count -ne 0) {

Write-Error "Unable to locate Microsoft Entra ID users for $dbu\_duplicate\_count rows as multiple rows have the same value"

}

$dbu\_not\_matched\_count = $dbu\_not\_matched\_list.Count

if ($dbu\_not\_matched\_count -ne 0) {

Write-Error "Unable to locate $dbu\_not\_matched\_count records in Microsoft Entra ID by querying for $db\_match\_column\_name values in $azuread\_match\_attr\_name."

}

$dbu\_match\_ambiguous\_count = $dbu\_match\_ambiguous\_list.Count

if ($dbu\_match\_ambiguous\_count -ne 0) {

Write-Error "Unable to locate $dbu\_match\_ambiguous\_count records in Microsoft Entra ID as attribute match ambiguous."

}

$dbu\_query\_failed\_count = $dbu\_query\_failed\_list.Count

if ($dbu\_query\_failed\_count -ne 0) {

Write-Error "Unable to locate $dbu\_query\_failed\_count records in Microsoft Entra ID as queries returned errors."

}

$azuread\_not\_enabled\_count = $azuread\_not\_enabled\_list.Count

if ($azuread\_not\_enabled\_count -ne 0) {

Write-Error "$azuread\_not\_enabled\_count users in Microsoft Entra ID are blocked from sign-in."

}

if ($dbu\_not\_queried\_count -ne 0 -or $dbu\_duplicate\_count -ne 0 -or $dbu\_not\_matched\_count -ne 0 -or $dbu\_match\_ambiguous\_count -ne 0 -or $dbu\_query\_failed\_count -ne 0 -or $azuread\_not\_enabled\_count) {

Write-Output "You will need to resolve those issues before access of all existing users can be reviewed."

}

$azuread\_match\_count = $azuread\_match\_id\_list.Count

Write-Output "Users corresponding to $azuread\_match\_count records were located in Microsoft Entra ID."

#When the script finishes, it will indicate an error if any records from the data source weren't located in Microsoft Entra ID. If not all the records for users from the application's data store could be located as users in Microsoft Entra ID, you'll need to investigate which records didn't match and why. Run below commands:  
$dbu\_missing\_columns\_list = @()

$dbu\_creation\_failed\_list = @()

foreach ($dbu in $dbu\_not\_matched\_list) {

if (($null -ne $dbu.$db\_display\_name\_column\_name -and $dbu.$db\_display\_name\_column\_name.Length -gt 0) -and

($null -ne $dbu.$db\_user\_principal\_name\_column\_name -and $dbu.$db\_user\_principal\_name\_column\_name.Length -gt 0) -and

($null -ne $dbu.$db\_mail\_nickname\_column\_name -and $dbu.$db\_mail\_nickname\_column\_name.Length -gt 0)) {

$params = @{

accountEnabled = $false

displayName = $dbu.$db\_display\_name\_column\_name

mailNickname = $dbu.$db\_mail\_nickname\_column\_name

userPrincipalName = $dbu.$db\_user\_principal\_name\_column\_name

passwordProfile = @{

Password = -join (((48..90) + (96..122)) \* 16 | Get-Random -Count 16 | % {[char]$\_})

}

}

try {

New-MgUser -BodyParameter $params

} catch { $dbu\_creation\_failed\_list += $dbu; throw }

} else {

$dbu\_missing\_columns\_list += $dbu

}

}  
  
  
#Check that no errors are reported:

$dbu\_not\_queried\_list = @()

$dbu\_not\_matched\_list = @()

$dbu\_match\_ambiguous\_list = @()

$dbu\_query\_failed\_list = @()

$azuread\_match\_id\_list = @()

$azuread\_not\_enabled\_list = @()

$dbu\_values = @()

$dbu\_duplicate\_list = @()

foreach ($dbu in $dbusers) {

if ($null -ne $dbu.$db\_match\_column\_name -and $dbu.$db\_match\_column\_name.Length -gt 0) {

$val = $dbu.$db\_match\_column\_name

$escval = $val -replace "'","''"

if ($dbu\_values -contains $escval) { $dbu\_duplicate\_list += $dbu; continue } else { $dbu\_values += $escval }

$filter = $azuread\_match\_attr\_name + " eq '" + $escval + "'"

try {

$ul = @(Get-MgUser -Filter $filter -All -Property Id,accountEnabled -ErrorAction Stop)

if ($ul.length -eq 0) { $dbu\_not\_matched\_list += $dbu; } elseif ($ul.length -gt 1) {$dbu\_match\_ambiguous\_list += $dbu } else {

$id = $ul[0].id;

$azuread\_match\_id\_list += $id;

if ($ul[0].accountEnabled -eq $false) {$azuread\_not\_enabled\_list += $id }

}

} catch { $dbu\_query\_failed\_list += $dbu }

} else { $dbu\_not\_queried\_list += $dbu }

}

$dbu\_not\_queried\_count = $dbu\_not\_queried\_list.Count

if ($dbu\_not\_queried\_count -ne 0) {

Write-Error "Unable to query for $dbu\_not\_queried\_count records as rows lacked values for $db\_match\_column\_name."

}

$dbu\_duplicate\_count = $dbu\_duplicate\_list.Count

if ($dbu\_duplicate\_count -ne 0) {

Write-Error "Unable to locate Microsoft Entra ID users for $dbu\_duplicate\_count rows as multiple rows have the same value"

}

$dbu\_not\_matched\_count = $dbu\_not\_matched\_list.Count

if ($dbu\_not\_matched\_count -ne 0) {

Write-Error "Unable to locate $dbu\_not\_matched\_count records in Microsoft Entra ID by querying for $db\_match\_column\_name values in $azuread\_match\_attr\_name."

}

$dbu\_match\_ambiguous\_count = $dbu\_match\_ambiguous\_list.Count

if ($dbu\_match\_ambiguous\_count -ne 0) {

Write-Error "Unable to locate $dbu\_match\_ambiguous\_count records in Microsoft Entra ID as attribute match ambiguous."

}

$dbu\_query\_failed\_count = $dbu\_query\_failed\_list.Count

if ($dbu\_query\_failed\_count -ne 0) {

Write-Error "Unable to locate $dbu\_query\_failed\_count records in Microsoft Entra ID as queries returned errors."

}

$azuread\_not\_enabled\_count = $azuread\_not\_enabled\_list.Count

if ($azuread\_not\_enabled\_count -ne 0) {

Write-Warning "$azuread\_not\_enabled\_count users in Microsoft Entra ID are blocked from sign-in."

}

if ($dbu\_not\_queried\_count -ne 0 -or $dbu\_duplicate\_count -ne 0 -or $dbu\_not\_matched\_count -ne 0 -or $dbu\_match\_ambiguous\_count -ne 0 -or $dbu\_query\_failed\_count -ne 0 -or $azuread\_not\_enabled\_count -ne 0) {

Write-Output "You will need to resolve those issues before access of all existing users can be reviewed."

}

$azuread\_match\_count = $azuread\_match\_id\_list.Count

Write-Output "Users corresponding to $azuread\_match\_count records were located in Microsoft Entra ID."

#Register the application. If the enterprise application is named CORPDB333, enter the following commands:

$azuread\_app\_name = "CORPDB333"

$azuread\_app = New-MgApplication -DisplayName $azuread\_app\_name

$azuread\_sp = New-MgServicePrincipal -DisplayName $azuread\_app\_name -AppId $azuread\_app.AppId

#Add a role to the application, and tag the application as integrated with Microsoft Entra ID so that its assignments can be reviewed. For example, if the role name is General, provide that value in the following PowerShell commands:

$ar0 = New-Object Microsoft.Graph.PowerShell.Models.MicrosoftGraphAppRole

$ar0.AllowedMemberTypes += "User"

$ar0.Description = "General role"

$ar0.DisplayName = "General"

$ar0.id = New-Guid

$ar0.IsEnabled = $true

$ar0.Value = "General"

$ara = @()

$ara += $ar0

$azuread\_app\_tags = @()

$azuread\_app\_tags += "WindowsAzureActiveDirectoryIntegratedApp"

$azuread\_app\_update = Update-MgApplication -ApplicationId $azuread\_app.Id -AppRoles $ara -Tags $azuread\_app\_tags

#Check for users who are not already assigned to the application:  
$azuread\_app\_name = "CORPDB333"

$azuread\_sp\_filter = "displayName eq '" + ($azuread\_app\_name -replace "'","''") + "'"

$azuread\_sp = Get-MgServicePrincipal -Filter $azuread\_sp\_filter -All

#Retrieve the users who currently have assignments to the application in Microsoft Entra ID; this builds upon the $azuread\_sp variable set in the previous command:  
$azuread\_existing\_assignments = @(Get-MgServicePrincipalAppRoleAssignedTo -ServicePrincipalId $azuread\_sp.Id -All)

#Compare the list of user IDs from the previous section to those users currently assigned to the application:  
$azuread\_not\_in\_role\_list = @()

foreach ($id in $azuread\_match\_id\_list) {

$found = $false

foreach ($existing in $azuread\_existing\_assignments) {

if ($existing.principalId -eq $id) {

$found = $true; break;

}

}

if ($found -eq $false) { $azuread\_not\_in\_role\_list += $id }

}

$azuread\_not\_in\_role\_count = $azuread\_not\_in\_role\_list.Count

Write-Output "$azuread\_not\_in\_role\_count users in the application's data store are not assigned to the application roles."

#Select the role of the application to assign the remaining users to. An application might have more than one role. Use this command to list the available roles:

$azuread\_sp.AppRoles | where-object {$\_.AllowedMemberTypes -contains "User"} | ft DisplayName,Id

#Select the appropriate role from the list, and obtain its role ID. For example, if the role name is General, provide that value in the following PowerShell commands:  
$azuread\_app\_role\_name = "General"

$azuread\_app\_role\_id = ($azuread\_sp.AppRoles | where-object {$\_.AllowedMemberTypes -contains "User" -and $\_.DisplayName -eq $azuread\_app\_role\_name}).Id

if ($null -eq $azuread\_app\_role\_id) { write-error "role $azuread\_app\_role\_name not located in application manifest"}

#Create app role assignments in Microsoft Entra ID For Microsoft Entra ID to match the users in the application with the users in Microsoft Entra ID, you need to create application role assignments in Microsoft Entra ID. Create application role assignments for users who don't currently have role assignments:  
foreach ($u in $azuread\_not\_in\_role\_list) {

$res = New-MgServicePrincipalAppRoleAssignedTo -ServicePrincipalId $azuread\_sp.Id -AppRoleId $azuread\_app\_role\_id -PrincipalId $u -ResourceId $azuread\_sp.Id

}

#Wait one minute for changes to propagate within Microsoft Entra ID. Query Microsoft Entra ID to obtain the updated list of role assignments:

$azuread\_existing\_assignments = @(Get-MgServicePrincipalAppRoleAssignedTo -ServicePrincipalId $azuread\_sp.Id -All)

#Compare the list of user IDs from the previous section to those users now assigned to the application:  
$azuread\_still\_not\_in\_role\_list = @()

foreach ($id in $azuread\_match\_id\_list) {

$found = $false

foreach ($existing in $azuread\_existing\_assignments) {

if ($existing.principalId -eq $id) {

$found = $true; break;

}

}

if ($found -eq $false) { $azuread\_still\_not\_in\_role\_list += $id }

}

$azuread\_still\_not\_in\_role\_count = $azuread\_still\_not\_in\_role\_list.Count

if ($azuread\_still\_not\_in\_role\_count -gt 0) {

Write-Output "$azuread\_still\_not\_in\_role\_count users in the application's data store are not assigned to the application roles."

}

# To add Roles via PS, add these command for each different Role. Note for last row - *$AppRoleValue* - blank spaces are not allowed and PS will give Error Code 400 Bad Request:

$scopes = "Application.ReadWrite.All"

$AppName = "CORPDB333"

$AppRoleDisplayName = "Onboarding Specialist (JR1004)"

$AppRoleDescription = "Onboarding Specialist (JR1004)"

$AppRoleValue = "Onboarding\_Specialist\_(JR1004).Onboarding\_Specialist\_(JR1004)"

# Then, to confirm and verify the registered role in your application, run below commands:

Connect-MgGraph -Scopes $scopes

function Add-AppRoles {

param(

[String]$AppName,

[String]$AppRoleDisplayName,

[String]$AppRoleDescription,

[String]$AppRoleValue

)

try {

$Id = [Guid]::NewGuid().ToString()

# Create new AppRole object

$newAppRole = New-Object Microsoft.Graph.PowerShell.Models.MicrosoftGraphAppRole

$newAppRole.AllowedMemberTypes = @("User", "Application")

$newAppRole.DisplayName = $AppRoleDisplayName

$newAppRole.Description = $AppRoleDescription

$newAppRole.Value = $AppRoleValue

$newAppRole.Id = $Id

$newAppRole.IsEnabled = $true

# Add new AppRole and apply changes to Application object

$App = Get-MgServicePrincipal -Filter "DisplayName eq '$AppName'"

#Write-Output $App

$appRoles = $App.AppRoles

#Write-Output $appRoles

# save the changes

$SPNObjectId = (Get-MgApplication -Filter "DisplayName eq '$AppName'").Id

$appRoles += $newAppRole

Write-Output $appRoles

Update-MgApplication -ApplicationId $SPNObjectId -AppRoles $appRoles

}

catch {

$message = $Error[0].Exception.Message

Write-Host "##vso[task.logissue type=error;]$message.";

Write-Error $message;

}

}

Add-AppRoles -AppName $AppName -AppRoleDisplayName $AppRoleDisplayName -AppRoleDescription $AppRoleDescription -AppRoleValue $AppRoleValue