Regularly clean up MEID devices with automation

> Requirements: Create an App Registration or System Managed Identity with Directory.ReadWrite.All and Device.ReadWrite.All. For logging you need to have an Azure Function to create an API to centralize logs in an Azure Log Analytics Workspace.

This automation gets all Microsoft Entra ID Devices via the Microsoft Graph API and checks everyone for their latest sign in. If the login was more than 6 months ago, the device will be deleted from the Microsoft Entra ID. Afterwards, the key figures for the deleted devices and the errors are passed to the Azure Function API, where they are then stored in the corresponding Log Analytics Workspace.

Use case

This automation concept is perfectly suited for large enterprises which don't have an overview of their old Microsoft Entra ID Devices. Authentication in the direction of Microsoft Entra ID is user-independent with an app registration. This makes it perfect for automated and regular execution of this code. In combination with an Azure Runbook that runs weekly, the Microsoft Entra ID environment can be simplified significantly.

PowerShell script

This PowerShell script needs the AzureAD Module, an app registration with the appropriate permissions and an Azure Function to collect the logs and write them to an Azure Log Analytics Workspace.

First install the Microsoft.Graph.Authentication Module.

```
Install-Module Microsoft.Graph.Authentication
```

After you have installed the modules and organized the prerequisites (App Registration with Directory.ReadWrite.All and Device.ReadWrite.All permission and an Azure Function API for centralizing the logs) you can fill the four variables to the PowerShell code and run the script.

Authentication with System Managed Identity

```
Import-Module Microsoft.Graph.Authentication

Connect-MgGraph -Identity -NoWelcome -ErrorAction Stop
$mgGraphRequest = Invoke-MgGraphRequest -Uri "https://graph.microsoft.com/v1.0/" -OutputType
HttpResponseMessage

$Global:MicrosoftEntraIDAccessToken =
$mgGraphRequest.RequestMessage.Headers.Authorization.Parameter

$Result = Invoke-RestMethod -Method GET -Uri "https://graph.microsoft.com/v1.0/devices" -
Header @{Authorization = "Bearer $Global:MicrosoftEntraIDAccessToken" }
$Devices = $Result.value
while ($Result.'@odata.nextLink') {
    $Result = Invoke-RestMethod -Uri $Result.'@odata.nextLink' -Headers  @{Authorization =
"Bearer $Global:MicrosoftEntraIDAccessToken" }
    $Devices += $Result.value
}

$oldDevices = @()
$failedDateDevices = @()
foreach ($inactiveDevice in $Devices) {
    try {
        $Date = Get-Date($inactiveDevice.approximateLastSignInDateTime)
        if ($Date -lt ((Get-Date).AddMonths(-6))) {
            $oldDevices += $inactiveDevice
        }
    }
    catch {
        $failedDateDevices += $inactiveDevice
    }
}
Write-Output "Date Errors: $($failedDateDevices.count)"

$failedDeleteDevices = @()
$deletedDevices = @()
foreach ($Device in $oldDevices) {
    try {
        Invoke-RestMethod -Method DELETE -Uri
"https://graph.microsoft.com/v1.0/devices/$($Device.id)" -Header @{Authorization = "Bearer
$Global:MicrosoftEntraIDAccessToken" }
```

```
        Write-Output "Deleted: $($Device.displayName)"
        $deletedDevices += $Device
    }
    catch {
        $failedDeleteDevices += $Device
    }
}
Write-Output "Delete Errors: $($failedDeleteDevices.count)"
```

Authentication using Application Credentials

```
$tenantId= "<yourtenantid>"
$ClientId= "<yourclientidofappregistration>"
$ClientSecret = "<yourclientsecretofappregistration>"
$url="<yoururlofcentralizedlogcollectionfunction>"

$expirationThresholdMonths = 6



Function Send-Logs() {

    param (
        [String]$LogType,
        [Hashtable]$LogBodyList
    )

    $LogBodyJSON = @"
    {
        "logtype": "$LogType",
        "logbody": {}
    }
"@

    $LogBodyObject = ConvertFrom-JSON $LogBodyJSON
    Foreach ($Log in $LogBodyList.GetEnumerator()) {
        $LogBodyObject.logbody | Add-Member -MemberType NoteProperty -Name $log.key -Value
$log.value
    }
    $Body = ConvertTo-JSON $LogBodyObject
```

```
    $Response = Invoke-Restmethod -uri $url -Body $Body -Method POST -ContentType
"application/json"
    return $Response
}

$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"
    "Method"      = "Post"
    "Body"        = $Body
    "ContentType" = "application/x-www-form-urlencoded"
}
$AuthResponse = Invoke-RestMethod @Params

$Global:MicrosoftEntraIDAccessToken = $AuthResponse.access_token

$Result = Invoke-RestMethod -Method GET -Uri "https://graph.microsoft.com/v1.0/devices" -
Header @{Authorization = "Bearer $Global:MicrosoftEntraIDAccessToken" }
$Devices = $Result.value
while ($Result.'@odata.nextLink') {
    $Result = Invoke-RestMethod -Uri $Result.'@odata.nextLink' -Header @{Authorization =
"Bearer $Global:MicrosoftEntraIDAccessToken" }
    $Devices += $Result.value
}

$oldDevices = @()
$failedDateDevices = @()
foreach ($inactiveDevice in $Devices) {
    try {
        $Date = Get-Date($inactiveDevice.approximateLastSignInDateTime)
        if ($Date -lt ((Get-Date).AddMonths(-$expirationThresholdMonths))) {
            $oldDevices += $inactiveDevice
        }
    }
```

```
    catch {
        $failedDateDevices += $inactiveDevice
    }
}

Write-Warning "Date Errors: $($failedDateDevices.count)"

$failedDeleteDevices = @()
$deletedDevices = @()
foreach ($Device in $oldDevices) {
    try {
        Invoke-RestMethod -Method DELETE -Uri
"https://graph.microsoft.com/v1.0/devices/$($Device.id)" -Header @{Authorization = "Bearer
$Global:MicrosoftEntraIDAccessToken" }
        Write-Output "Deleted: $($Device.displayName)"
        $deletedDevices += $Device
    }
    catch {
        $failedDeleteDevices += $Device
    }
}
Write-Warning "Delete Errors: $($failedDeleteDevices.count)"

$Logs = @{
    "deletesuccess" = "$($deletedDevices.count)"
    "dateerrors"    = "$($failedDateDevices.count)"
    "deleteerrors"  = "$($failedDeleteDevices.count)"
}

Send-Logs -LogType "CleanUpMEIDDevicesExecutions" -LogBodyList $Logs
```

Revision #12
Created 2022-12-14 16:27:23 UTC
Updated 2026-08-22 22:52:57 UTC by Luca Noah Caprez