# Notify client app secret expiration with automation

> Requirements: Create an App Registration with Application.Read.All permissions and a client secret.

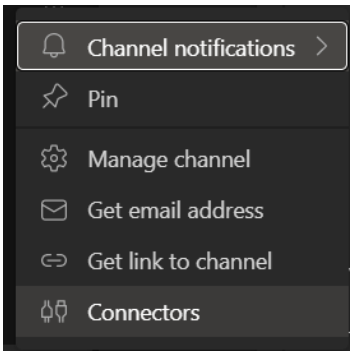
With this automation all client secrets and their expiration dates are evaluated via Graph API. A message is posted in a team channel when a client secret is about to expire.

## Use case

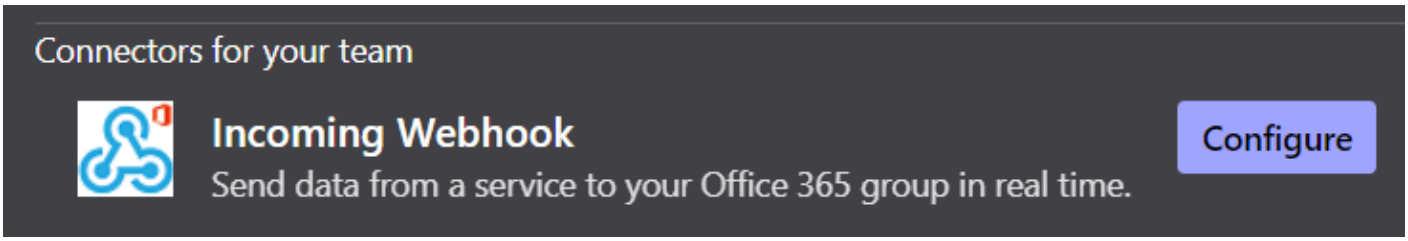
This automation helps large organizations with many Microsoft Entra ID App Registrations to keep track of expiration dates of their Client Secrets. If this automation is carried out regularly, interruptions in operation and expired client secrets can be noticed at an early stage. To run such scripts on a regular basis, an Azure Runbook is a good choice.

## Teams webhook notification

To send a notification to a Teams Channel you can use a webhook. To create a webhook for a channel you can rightclick on a channel and then choose "Connectors".




There you can then configure a new incoming webhook:




You have to name the webhook first. This name will be displayed as the sender of the message. You can upload an image which will then be used as a profile picture of the sender. After that you can click "create". This will return an URL which then can be used to send a payload message.

# PowerShell script

This PowerShell script needs an App Registration and the according Application.Read.All Graph API permission. This script uses the three Azure Automation variables "$tenantid", "$clientid" and "$clientsecret" as well as the perviously created "$WebhookURI", which should be added before.

```
$tenantId=Get-AutomationVariable -Name "<nameoftenantidrunbookvariable>"
$ClientId=Get-AutomationVariable -Name "<nameofclientidrunbookvariable>"
$CredentialObject=Get-AutomationPSCredential -Name '<nameofclientsecretrunbookcredentials>'
$ClientSecret = $CredentialObject.GetNetworkcredential().password

$WebhookURI = "<yourwebhookuri>"

$NotifyOnDays = @(1,2,3,4,5,10,15,20,30)
$DeleteSecretIfExpiredForDays = 30

$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

$Header = @{
    "Authorization" = "Bearer $($AuthResponse.access_token)"
}

function Send-ToTeams {
    Param(
        $SecretOB
    )

    $Body = @{
"@context" = "https://schema.org/extensions"
"@type" = "MessageCard"
"themeColor" = "880808"
"title" = "Secret of App Registration $($SecretOB.AppName) expires in
$($SecretOB.ExpiresInXDays)!"
    "text" = "<ul><li>App: $($SecretOB.AppName)</li><li>AppId:
$($SecretOB.AppID)</li><li>SecretId: $($SecretOB.SecretID)</li><li>Expirationdate:
$($SecretOB.ExpiryDate) (in $($SecretOB.ExpiresInXDays) days)</li></ul>"
}

    $JsonBody = $Body | ConvertTo-JSON

    Invoke-RestMethod -Method Post -Body $JsonBody -Uri $WebhookURI -Headers @{"content-type"
= "application/json; charset=UTF-8"} | out-null
}



$AllApps = @()

$Response = Invoke-RestMethod -Method Get -Uri "https://graph.microsoft.com/v1.0/applications"
-Headers $Header
$AllApps = $Response.value

while ($Response.'@odata.nextLink') {
    $Response = Invoke-RestMethod -Method Get -Uri $Response.'@odata.nextLink' -Headers
$Header
    $AllApps += $Response.value
```

```
}

Write-Output "$($Allapps.Count) Apps were discovered"



$AllSecrets = @()

foreach ($App in $AllApps) {
    foreach ($AppSecret in $App.passwordCredentials) {
        $ExpiresInXDays = $(New-TimeSpan -Start $(Get-Date) -End $(Get-Date
$AppSecret.endDateTime)).TotalDays
        switch ($ExpiresInXDays) {
            { $_ -lt 30 } {$Status = "Expires Soon" }
            { $_ -lt 0 } {$Status = "Expired" }
            Default {$Status = "Ok" }
        }

        $SecretOB = [PSCustomObject]@{
            AppID = $App.AppID
            AppName = $App.DisplayName
            SecretID = $AppSecret.KeyID
            ExpiryDate = $AppSecret.endDateTime
            ExpiresInXDays = [math]::Round($ExpiresInXDays)
            Status = $Status
        }

        $AllSecrets += $SecretOB

        if ($SecretOB.ExpiresInXDays -in $NotifyOnDays) {
            [array]$AppOwners = $(Invoke-RestMethod -Method get -Uri
"https://graph.microsoft.com/v1.0/applications/$($App.ID)/owners" -Headers $Header).value

            Write-Warning "Secret $($SecretOB.SecretID) on App $($SecretOB.AppID) expires in
$($SecretOB.ExpiresInXDays) days."

            if ($AppOwners) {
                foreach ($AdminOwner in $AppOwners | Where-Object { $_.userPrincipalName -like
"A_" }) {
                    $SAM = $AdminOwner.userPrincipalName.Split("@")[0].replace("A_","")
                    $RegularUser = $(Invoke-RestMethod -Method GET -uri
```

```
"https://graph.microsoft.com/v1.0/users?`$filter=startswith(MailNickName, '$SAM')" -Headers
$Header).Value

                    $AppOwners += $RegularUser
                }
                if ([array]$AppOwners.Mail) {
                    Send-ToTeams -SecretOB $SecretOB
                } else {
                    Send-ToTeams -SecretOB $SecretOB
                }
            } else {
                Send-ToTeams -SecretOB $SecretOB
            }
        } elseif ($SecretOB.ExpiresInXDays -lt -$DeleteSecretIfExpiredForDays) {
            if ($($App.passwordCredentials | Where-Object { $_.endDateTime -gt $(get-date)
}).Count -ge 1) {
                Write-Output "Secret $($SecretOB.SecretID) on App $($SecretOB.AppID) expired
$(-$SecretOB.ExpiresInXDays) Days ago and can be deleted. Client has at least one Secret that
does not expire whithin the next $DeleteSecretIfExpiredForDays days."
            } else {
                Write-Output "Secret $($SecretOB.SecretID) on App $($SecretOB.AppID) expired
$(-$SecretOB.ExpiresInXDays) Days ago and can be deleted"
            }

            $SecretDeletionBody = @"
            {
                "keyId": "$($SecretOB.SecretID)"
            }
"@
            #Invoke-RestMethod -Method Post -Uri
"https://graph.microsoft.com/v1.0/applications/$($SecretOB.AppID)/removePassword" -Body
$SecretDeletionBody -Headers $Header
        }
    }
}
```

If you want to automatically delete old app registrations that have expired for more than a defined number of days ($DeleteSecretIfExpiredForDays) and no longer have any active secrets, you can show the time 118. This will permanently delete all app registrations that have expired client secrets.

Revision #9
Created 2023-03-20 21:14:42 UTC
Updated 2026-08-23 16:20:47 UTC by Luca Noah Caprez