Microsoft Teams Application Access Policy – User Assignment Automation

Field	Details
Document Type	Microsoft Teams Application Access Policy – User Assignment Automation
Applies To	Microsoft Entra ID, Microsoft Teams & MS Graph
Audience	MS Teams Administrator / IT Engineer
Author	AK. Udofeh
Last Updated	June 2026

Overview

This document provides a high-level overview of the automated user assignment process used for Microsoft Teams Application Access Policies.

The automation is designed to simplify operational management of Teams Application Access Policies by using Entra ID group membership as the source of truth for policy assignment.

This approach is commonly used when integrating:

- Third-party applications
- Microsoft Graph OnlineMeetings APIs
- Teams meeting automation platforms
- Virtual appointment systems

The automation helps ensure that only approved users are assigned to the relevant Teams Application Access Policy.

Business Justification

Microsoft Teams Application Access Policies currently do not support:

- Native Active Directory group assignment

- Entra ID group targeting

Policies must instead be assigned individually per-user using PowerShell.

Without automation:

- User onboarding becomes manual
- User removals may be missed
- Administrative overhead increases
- Access governance becomes difficult to maintain
- Policy drift risks increase over time

The automation solves this limitation by:

- Using Entra ID groups as the source of truth
- Automatically assigning approved users
- Automatically removing users no longer approved
- Maintaining operational consistency

High-Level Process Flow

Source of Truth

Approved users are managed through designated:

- Entra ID security groups
  OR
- synced Active Directory groups

Automation Workflow

The automation process performs the following actions:

1. Connects to Microsoft Graph
2. Enumerates approved group members
3. Connects to Microsoft Teams PowerShell
4. Retrieves existing policy assignments
5. Performs delta comparison
6. Assigns missing users to the policy
7. Removes users no longer in scope
8. Logs all changes and failures

Operational Behaviour

User Additions

When a user is added to the approved group:

- The automation assigns the Teams Application Access Policy automatically during the next scheduled run

User Removals

When a user is removed from the approved group:

- The automation removes the Teams Application Access Policy assignment automatically during the next scheduled run

Delta-Based Processing

The automation uses delta comparison logic to:

- Avoid duplicate assignments
- Minimise unnecessary PowerShell operations
- Improve execution efficiency

Authentication Model

The automation uses:

- Microsoft Graph PowerShell
- Microsoft Teams PowerShell
- Entra ID App Registration
- Certificate-based authentication

Certificate authentication is recommended because it:

- Reduces credential exposure
- Supports unattended execution
- Improves operational security
- Avoids client secret lifecycle management

Validation & Testing

Validation should include:

- Successful group enumeration
- Successful Teams PowerShell connectivity
- Successful policy assignment
- Successful policy removal
- Delta comparison validation
- Logging verification

Example validation checks:

```
Get-CsApplicationAccessPolicy
```

```
Get-CsOnlineUser -Identity user@domain.com
```

Validation should confirm:

- Approved users receive the policy
- Removed users lose the policy
- Existing assignments are not duplicated

Monitoring & Logging

The automation should log:

- Added users
- Removed users
- Assignment failures
- Execution timestamps
- Authentication failures

The following should be monitored periodically:

- Script execution success
- Teams PowerShell connectivity
- Certificate expiry dates
- Group membership accuracy
- Failed assignment attempts

Important Considerations

No Native Group Support

This automation exists specifically because:

- Teams Application Access Policies currently do not support direct group assignment

Policy Scope

Policies should remain scoped only to:

- Approved operator accounts
- Approved service users

Global assignment is not recommended.

Teams RBAC Permissions

The Entra ID application used by the automation requires:

- Microsoft Graph permissions

- Teams administrative RBAC permissions

Certificate Lifecycle Management

Certificate expiry dates should be monitored carefully to avoid automation failure.

Automation Script

```
# Get list of MS Teams App Access policy
# Get-CsApplicationAccessPolicy

# ============================================================
# Teams Application Access Policy Automation Script
# App-only authentication | Delta-based | Lightweight Version
# ============================================================

# ----------------------------
# CONFIGURATION
# ----------------------------

$TenantId = "{your tenant id here}"   # UniEssex Entra ID tenant ID (GUID)
$AppId = "{your app id here}"   # Entra ID app registration application (client) ID (GUID)
$CertThumbprint = "{your cert thumbprint here}"   # Thumbprint of the certificate used for
authentication (no spaces)

# Teams Application Access Policy Name

$TeamsPolicyName = "{your Teams App Policy name here}"

# Entra ID Groups containing approved operators

$GroupIds = @("{your Entra ID Group name here}")

$LogFile = "{your log file location here}"

# ----------------------------
# LOGGING
# ----------------------------

$logDir = Split-Path $LogFile
```

```
if (!(Test-Path $logDir)) {
New-Item -ItemType Directory -Path $logDir | Out-Null
}

function Write-Log {
    param (
        [string]$Action,
        [string]$UPN
    )
    $timestamp = "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] -"
    if($Action.StartsWith("ERROR_")) {
        $timestamp = "*** $timestamp"
    }
    Add-Content -Path $LogFile -Value "$timestamp $Action,$UPN,$TeamsPolicyName"
    Write-Host "$timestamp $Action,$UPN,$TeamsPolicyName"
}
# ----------------------------
# CONNECT TO MICROSOFT GRAPH
# ----------------------------

Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - Connecting to MS Graph..."

Connect-MgGraph `
    -TenantId $TenantId `
    -ClientId $AppId `
    -CertificateThumbprint $CertThumbprint `
    -NoWelcome

Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - Querying Group Members..."

$GroupUsers = foreach($GroupId in $GroupIds) {

Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - Querying Group $GroupId Members..."

(Get-MgGroupMemberAsUser `
    -All `
    -GroupId $GroupId `
    -Property UserPrincipalName
).UserPrincipalName.ToLower()
```

```
}

# Deduplicate users

$GroupUsers = $GroupUsers | Sort-Object -Unique

Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - $($GroupUsers.Length) Group Users to evaluate..."

Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - Disconnecting from MS Graph... "

Disconnect-MgGraph

# ----------------------------
# LIGHTWEIGHT SAFETY CHECK
# ----------------------------

if ($GroupUsers.Count -lt 1) {
    Write-Host "*** [$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - ERROR No group users returned"
    Write-Log "ABORT" "No group users returned"
exit 1
}

# ----------------------------
# CONNECT TO MICROSOFT TEAMS
# ----------------------------

Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - Connecting to Microsoft Teams..."

Connect-MicrosoftTeams `
    -TenantId $TenantId `
    -ApplicationId $AppId `
    -CertificateThumbprint $CertThumbprint

Get-CsOnlineUser |
ForEach-Object {

    Write-Host "User: $($_.UserPrincipalName)"
    Write-Host "Policy: '$($_.ApplicationAccessPolicy)'"
```

```
}

Write-Host "Configured Policy Name: '$TeamsPolicyName'"

# ----------------------------
# GET CURRENT POLICY ASSIGNMENTS
# ----------------------------
Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - Getting Current Policy
Assignments..."

$PolicyUsers = foreach ($user in Get-CsOnlineUser) {

    if ($null -ne $user.ApplicationAccessPolicy) {

        $CurrentPolicy = $user.ApplicationAccessPolicy.ToString().ToLower()

        if ($CurrentPolicy -eq $TeamsPolicyName.ToLower()) {

            $user.UserPrincipalName.ToLower()
        }
    }
}

$PolicyUsers = $PolicyUsers | Sort-Object -Unique

Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - $($PolicyUsers.Length) Existing
Policy Users found"

# ----------------------------
# DELTA CALCULATION
# ----------------------------
Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - Calculating Delta..."
$UsersToAdd = $GroupUsers | Where-Object { $_ -notin $PolicyUsers }
$UsersToRemove = $PolicyUsers | Where-Object { $_ -notin $GroupUsers }

Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - $($UsersToAdd.Length) users to add"
Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - $($UsersToRemove.Length) users to
remove"

# ----------------------------
```

```
# APPLY CHANGES
# ---------------------------
Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - Applying Changes..."
foreach ($upn in $UsersToAdd) {
    try {

        Grant-CsApplicationAccessPolicy `
            -Identity $upn `
            -PolicyName $TeamsPolicyName `
            -ErrorAction Stop

        Write-Log "ADD" $upn
    }
    catch {
        Write-Log "ERROR_ADD" $upn
    }
}

foreach ($upn in $UsersToRemove) {
    try {

        Grant-CsApplicationAccessPolicy `
            -Identity $upn `
            -PolicyName $null `
            -ErrorAction Stop

        Write-Log "REMOVE" $upn
    }
    catch {
        Write-Log "ERROR_REMOVE" $upn
    }
}

Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - Disconnecting from Microsoft
Teams..."

Disconnect-MicrosoftTeams -Confirm:$false

Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] - Done"
```

```
exit 0
```

Recommended script capabilities:

- Microsoft Graph group enumeration
- Teams policy assignment/removal
- Delta comparison logic
- Logging and error handling
- Certificate-based authentication

Summary

This automation provides scalable and governed user assignment management for Microsoft Teams Application Access Policies.

The solution compensates for the lack of native group-based policy assignment support within Teams by:

- Using Entra ID groups as the source of truth
- Automating user assignment and removal
- Maintaining operational consistency and governance

This approach significantly reduces manual administration while improving access control accuracy and auditability.

Revision #3
Created 2026-06-11 10:46:32 UTC by AK. Udofeh
Updated 2026-06-11 11:49:27 UTC by AK. Udofeh