Count-MembersInADGroupNested

```
########################################
# AUTHOR    : Ryan Mutschler
# DATE      : 4-2-2025
# EDIT      : 4-2-2025
# PURPOSE   : Count members of an Active Directory group, including nested groups.
# REPOSITORY: https://github.com/MutschlerHomeTech/Public-
Scripts/blob/master/1.%20Full%20Scripts/Windows/File%20System/Create-WorkloadFolder.ps1
# WIKIPEDIA : https://wikipedia.mutschlerhome.com/books/windows-scripts/page/create-
workloadfolder-v12
#
# VERSION   : 1.0     (Initial release)
########################################

# Count-ADGroupMembers.ps1
# Script to count members of an Active Directory group, including nested groups

# Import the Active Directory module
Import-Module ActiveDirectory

function Get-ADNestedGroupMembers {
    param (
        [Parameter(Mandatory = $true)]
        [string]$GroupName
    )

    # Initialize a hashtable to track processed groups (prevents infinite loops with circular
nesting)
    $processedGroups = @{}
```

```
    # Initialize a System.Collections.ArrayList to store all members (more efficient than
arrays for adding items)
    $allMembers = New-Object System.Collections.ArrayList

    # Define a recursive function to get members including nested groups
    function Get-NestedMembers {
        param (
            [Parameter(Mandatory = $true)]
            [string]$GroupDistinguishedName
        )

        # Skip if we've already processed this group
        if ($processedGroups.ContainsKey($GroupDistinguishedName)) {
            return
        }

        # Mark this group as processed
        $processedGroups[$GroupDistinguishedName] = $true

        # Get direct members of the group
        $groupMembers = Get-ADGroupMember -Identity $GroupDistinguishedName -ErrorAction
SilentlyContinue

        foreach ($member in $groupMembers) {
            # Add all members to our collection using ArrayList.Add() method
            [void]$allMembers.Add($member)

            # If the member is a group, process its members recursively
            if ($member.objectClass -eq 'group') {
                Get-NestedMembers -GroupDistinguishedName $member.distinguishedName
            }
        }
    }

    # Get the group's distinguished name
    try {
        $group = Get-ADGroup -Identity $GroupName -ErrorAction Stop
        $groupDN = $group.DistinguishedName
    }
    catch {
```

```
        Write-Error "Error finding group '$GroupName': $_"
        return $null
    }

    # Start the recursive process
    Get-NestedMembers -GroupDistinguishedName $groupDN

    # Return all unique members (remove duplicates)
    return $allMembers | Sort-Object -Property objectGUID -Unique
}

# Function to display counts by type
function Show-MemberTypeCounts {
    param (
        [Parameter(Mandatory = $true)]
        [array]$Members
    )

    $userCount = ($Members | Where-Object { $_.objectClass -eq 'user' }).Count
    $groupCount = ($Members | Where-Object { $_.objectClass -eq 'group' }).Count
    $computerCount = ($Members | Where-Object { $_.objectClass -eq 'computer' }).Count
    $otherCount = ($Members | Where-Object { $_.objectClass -notin 'user','group','computer'
}).Count

    Write-Host "`nMembers by type:"
    Write-Host "  Users: $userCount"
    Write-Host "  Groups: $groupCount"
    Write-Host "  Computers: $computerCount"
    Write-Host "  Other objects: $otherCount"
}

# Main script
$groupName = Read-Host "Enter the name of the Active Directory group"

Write-Host "`nRetrieving members of group '$groupName' (including nested groups)..."
$members = Get-ADNestedGroupMembers -GroupName $groupName

if ($null -ne $members) {
    $totalCount = $members.Count
    Write-Host "`nTotal unique members: $totalCount"
```

```
    # Show breakdown by object type
    Show-MemberTypeCounts -Members $members

    # Optionally export to CSV
    $exportCsv = Read-Host "`nExport members to CSV? (Y/N)"
    if ($exportCsv -eq 'Y' -or $exportCsv -eq 'y') {
        # Ensure C:\Temp directory exists
        if (-not (Test-Path -Path "C:\Temp" -PathType Container)) {
            try {
                New-Item -Path "C:\Temp" -ItemType Directory -Force | Out-Null
                Write-Host "Created directory: C:\Temp"
            } catch {
                Write-Error "Failed to create C:\Temp directory: $_"
                return
            }
        }

        $csvPath = "C:\Temp\$groupName-members.csv"
        $members | Select-Object Name, SamAccountName, objectClass, distinguishedName |
            Export-Csv -Path $csvPath -NoTypeInformation
        Write-Host "Members exported to: $csvPath"
    }
}
```

Revision #2
Created 2025-04-02 16:28:18 UTC by Ryan
Updated 2025-04-02 16:29:06 UTC by Ryan