Set primary user of Windows devices by last logged in users with automation

This tutorial describes how an automation can be used to set the primary user according to the last signed in user. The Intune data is queried via an App Registration on Graph and modified accordingly. This automation is based on an Azure runbook and executes PowerShell code.

Prerequisites

First, an App Registration is used, which is used as an unattended authentication to the Graph API. This app registration requires "User.Read.All" and "DeviceManagementManagedDevies.ReadWrite.All". Create there the corresponding Client Secret and Client id as described here: Get app details and gr... | LNC DOCS (lucanoahcaprez.ch) Fill in the variables $tenantid, $clientid and $clientsecret with the corresponding values.

Subsequently, the $WebhookURI variable can be populated with a webhook from a team channel. Creating a webhook for notification in Microsoft Teams is described here: Teams webhook notification | LNC DOCS (lucanoahcaprez.ch)

The last function used is a LogCollection Function. You should copy the URL into the $FunctionURL variable. The instructions for a central log collection point can be found here: Centralize log collect... | LNC DOCS (lucanoahcaprez.ch)

PowerShell script

This is the PowerShell script that makes the automation possible. It is very important that you fill the variables correctly as described above or the unused parts are hidden. As written, this code is optimized to run in an Azure Runbook and also uses the variables from the Azure Automation account. Accordingly, these must be filled correctly.

```
#region Autorization Function
function Get-ApplicationOnlySourceAuthorization {

    param(
        $tenantId,
        $clientid,
        $clientSecret
    )



    $tokenBodySource = @{
        grant_type = "client_credentials"
        scope = "https://graph.microsoft.com/.default"
        client_id  = $clientid
        client_secret = "$clientSecret"
    }

    # Get OAuth Token
    while ([string]::IsNullOrEmpty($tokenRequestSource.access_token)) {

        $tokenRequestSource = try {
            Invoke-RestMethod -Method POST -Uri
"https://login.microsoftonline.com/$tenantId/oauth2/v2.0/token" -Body $tokenBodySource

        }
        catch {

            $errorMessageSource = $_.ErrorDetails.Message | ConvertFrom-Json

            # If not waiting for auth, throw error
            if ($errorMessageSource.error -ne "authorization_pending") {

                throw

            }

        }

    }
```

```
    $global:tokensource = $tokenRequestSource.access_token

    if($global:tokenSource){
        Write-output "Source Authorization successful!"

    }else{
        Write-Error "Source Authorization not successful. Do not continue! Check your
credentials first!"
    }

}

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 $FunctionUrl -Body $Body -Method POST -ContentType
"application/json"
    return $Response
}

function Send-ToTeams {
```

```
    $CurrentTime = Get-Date
    $Body = @{
"@context" = "https://schema.org/extensions"
"@type" = "MessageCard"
"themeColor" = "880808"
"title" = "RB-INT-ALL-PS1-ChangePrimaryUserByOwner-PROD ist durchgelaufen"
    "text" = @"
Parameter:
<ul><li>Successful primary user assignments in Intune:
$($SuccessfulAssignmentDevices.Count)</li><li>Error with assignments in Intune:
$($ErrorAssignmentDevices.Count)</li><li>Error no LastSignIn on Device:
$($ErrorLastSignIn.count)</li></ul>
"@
}
$JsonBody = $Body | ConvertTo-JSON

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

}
#endregion Functions

#Graph Authentication
$tenantId=Get-AutomationVariable -Name "<yourfunctionstenantidvariable>"
$ClientId=Get-AutomationVariable -Name "yourfunctionsclientidvariable"
$CredentialObject=Get-AutomationPSCredential -Name 'yourfunctionsclientsecretsecret'
$ClientSecret = $CredentialObject.GetNetworkcredential().password

$WebhookURI = Get-AutomationVariable -Name "<yourteamsnotificationchannelwebhook>"
$FunctionUrl= Get-AutomationVariable -Name "<yourlogcollectionfunction>"

# SourceAuthorization
Get-ApplicationOnlySourceAuthorization -tenantId $tenantId -clientid $clientid -clientSecret
$clientSecret

$SuccessfulAssignmentDevices = @()
$ErrorAssignmentDevices = @()
$ErrorLastSignIn = @()
$ExcludedServiceAccounts = @(
    "<yourserviceaccountswhichshouldnotbesetasprimaryusers>"
```

```
)

#Get Intune managed devices
$uri =
"https://graph.microsoft.com/v1.0/deviceManagement/managedDevices?`$filter=startswith(operatin
gSystem,'windows')"
$Results = Invoke-RestMethod -Method GET -Uri $uri -ContentType "application/json" -Headers
@{Authorization = "Bearer $($Global:tokenSource)"; ConsistencyLevel = "eventual"}
$ResultsValue = $results.value
if ($results."@odata.nextLink" -ne $null) {
    $NextPageUri = $results."@odata.nextLink"
    ##While there is a next page, query it and loop, append results
    While ($NextPageUri -ne $null) {
        $NextPageRequest = (Invoke-RestMethod -Headers @{Authorization = "Bearer
$($Global:tokenSource)"} -Uri $NextPageURI -Method Get)
        $NxtPageData = $NextPageRequest.Value
        $NextPageUri = $NextPageRequest."@odata.nextLink"
        $ResultsValue = $ResultsValue + $NxtPageData
    }
}
$IntuneDevices = $ResultsValue | where {($_.devicename -like "MW-*") -or ($_.devicename -like
"CPC-*")}

foreach($IntuneDevice in $IntuneDevices){

    # get primary user if exists
    $uri =
"https://graph.microsoft.com/beta/deviceManagement/managedDevices/$($IntuneDevice.id)/users"
    $UserObject = (Invoke-RestMethod -Method GET -Uri $uri -ContentType "application/json" -
Headers @{Authorization = "Bearer $($Global:tokenSource)"}).value

    if(!$UserObject.userPrincipalName -or $ExcludedServiceAccounts -contains
$UserObject.userPrincipalName){
        # get last signed in user of intune device
        $uri =
"https://graph.microsoft.com/beta/deviceManagement/managedDevices/$($IntuneDevice.id)?`$select
=usersLoggedOn"
        try{
            $Results = Invoke-RestMethod -Method GET -Uri $uri -ContentType "application/json"
-Headers @{Authorization = "Bearer $($Global:tokenSource)"}
```

```
            $PrimaryUserId = $Results.UsersLoggedOn[0].userid
        }
        catch{
            Write-output "The intune device $($CurrentDevice.displayname) has no primary
user!"
        }

        $PrimaryUser = ""
        if($PrimaryUserId){
            $uri = "https://graph.microsoft.com/v1.0/users/$PrimaryUserId"
            try{
                $PrimaryUser = Invoke-RestMethod -Method GET -Uri $uri -ContentType
"application/json" -Headers @{Authorization = "Bearer $($Global:tokenSource)"}
            }
            catch{
                Write-Error "The user id $PrimaryUserId has no UPN!"
            }
        }
        else{
            Write-Warning "No last user found for device $($IntuneDevice.deviceName)"
            $ErrorLastSignIn += $IntuneDevice.id
        }

        #configure primary owner
        if($PrimaryUser -and !($ExcludedServiceAccounts -contains
$PrimaryUser.userPrincipalName)){
            $BodyPrimaryUser = @"
{
    "@odata.id": "https://graph.microsoft.com/beta/users/$($PrimaryUser.id)"
}
"@

            $uri =
"https://graph.microsoft.com/beta/deviceManagement/managedDevices('$($IntuneDevice.id)')/users
/`$ref"
            try{
                $Results = Invoke-RestMethod -Method POST -Body $BodyPrimaryUser -Uri $uri -
ContentType "application/json" -Headers @{Authorization = "Bearer $($Global:tokenSource)"}
                Write-Output "Set intune primary user $($PrimaryUser.userPrincipalName) for
device $($IntuneDevice.deviceName)"
```

```
                $SuccessfulAssignmentDevices += $IntuneDevice.id
            }catch{
                Write-Error "Failed setting primary user $($PrimaryUser.userPrincipalName) for
device $($IntuneDevice.deviceName)"
                $ErrorAssignmentDevices += $IntuneDevice.id
            }
        }
    }
}

$Logs = @{
    "successassignment"="$($SuccessfulAssignmentDevices.count)"
    "errorassignment"="$($ErrorAssignmentDevices.count)"
    "errorlastsignnotfound"="$($ErrorLastSignIn.count)"
}

Send-Logs -LogType "ChangePrimaryUserByOwnerExecutions" -LogBodyList $Logs

Send-ToTeams
```

Revision #8
Created 2023-05-24 08:05:21 UTC
Updated 2026-08-23 16:20:49 UTC by Luca Noah Caprez