Intune - Scripts

- Custom Intune Detection Script
- How to change the owner of an Azure Active Directory device

Custom Intune Detection Script

https://silentinstallhq.com/create-a-custom-detection-script-for-1password-powershell/

1Password (Registry Detection Method)
• Install the version of 1Password you want to deploy on a test box or VM
• Check out the following posts for further details
• 1Password Silent Install (How-To Guide)
• 1Password Install and Uninstall (PowerShell)
• Open Windows PowerShell ISE by Right-Clicking on Windows PowerShell ISE and selecting Run as Administrator
• Copy the following code into the Windows PowerShell ISE

```
## Check for 1Password (Registry Detection Method)
$1Password = Get-ChildItem -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Uninstall" |
Get-ItemProperty | Where-Object {$_.DisplayName -match '1Password' } | Select-Object -Property
DisplayName, DisplayVersion, PSChildName
$1Password.DisplayVersion
$1Password.PSChildName
## Create Text File with 1Password Registry Detection Method
$FilePath = "C:\Windows\Temp\1Password_Detection_Method.txt"
New-Item -Path "$FilePath" -Force
Set-Content -Path "$FilePath" -Value "Function Get-LoggedOnUserSID {"
Add-Content -Path "$FilePath" -Value "## ref
https://www.reddit.com/r/PowerShell/comments/7coamf/query_no_user_exists_for/"
Add-Content -Path "$FilePath" -Value "## ref https://smsagent.blog/2022/03/03/user-context-
detection-rules-for-intune-win32-apps/"
Add-Content -Path "$FilePath" -Value "`$header=@('SESSIONNAME', 'USERNAME', 'ID', 'STATE',
'TYPE', 'DEVICE')"
Add-Content -Path "$FilePath" -Value "`$Sessions = query session"
Add-Content -Path "$FilePath" -Value "[array]`$ActiveSessions = `$Sessions | Select -Skip 1 |
Where {`$_ -match ""Active""}"
Add-Content -Path "$FilePath" -Value "If (`$ActiveSessions.Count -ge 1)"
Add-Content -Path "$FilePath" -Value "{"
Add-Content -Path "$FilePath" -Value "`$LoggedOnUsers = @()"
Add-Content -Path "$FilePath" -Value "`$indexes = `$header | ForEach-Object
{(`$Sessions[0]).IndexOf("" `$_"")}"
Add-Content -Path "$FilePath" -Value "for(`$row=0; `$row -lt `$ActiveSessions.Count; `$row++)"
Add-Content -Path "$FilePath" -Value "{"
```

```
Add-Content -Path "$FilePath" -Value "`$obj=New-Object psobject"
Add-Content -Path "$FilePath" -Value "for(`$i=0; `$i -lt `$header.Count; `$i++)"
Add-Content -Path "$FilePath" -Value "{"
Add-Content -Path "$FilePath" -Value "`$begin=`$indexes[`$i]"
Add-Content -Path "$FilePath" -Value "`$end=if(`$i -lt `$header.Count-1) {`$indexes[`$i+1]}
else {`$ActiveSessions[`$row].length}"
Add-Content -Path "$FilePath" -Value "`$obj | Add-Member NoteProperty `$header[`$i]
(`$ActiveSessions[`$row].substring(`$begin, `$end-`$begin)).trim()"
Add-Content -Path "$FilePath" -Value "}"
Add-Content -Path "$FilePath" -Value "`$LoggedOnUsers += `$obj"
Add-Content -Path "$FilePath" -Value "}"
Add-Content -Path "$FilePath" -Value "`$LoggedOnUser = `$LoggedOnUsers[0]"
Add-Content -Path "$FilePath" -Value "`$LoggedOnUserSID = Get-ItemProperty
""HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\LogonUI\SessionData\`$(`$Logg
edOnUser.ID)"" -Name LoggedOnUserSID -ErrorAction SilentlyContinue |"
Add-Content -Path "$FilePath" -Value "Select -ExpandProperty LoggedOnUserSID"
Add-Content -Path "$FilePath" -Value "Return `$LoggedOnUserSID"
Add-Content -Path "$FilePath" -Value "}"
Add-Content -Path "$FilePath" -Value "}"
Add-Content -Path "$FilePath" -Value "`$LoggedOnUserSID = Get-LoggedOnUserSID"
Add-Content -Path "$FilePath" -Value "If (`$null -ne `$LoggedOnUserSID)"
Add-Content -Path "$FilePath" -Value "{"
Add-Content -Path "$FilePath" -Value "If (`$null -eq (Get-PSDrive -Name HKU -ErrorAction
SilentlyContinue))"
Add-Content -Path "$FilePath" -Value "{"
Add-Content -Path "$FilePath" -Value "`$null = New-PSDrive -PSProvider Registry -Name HKU -
Root HKEY_USERS"
Add-Content -Path "$FilePath" -Value "}"
Add-Content -Path "$FilePath" -Value "`$i = Get-Item
""HKU:\`$LoggedOnUserSID\Software\Microsoft\Windows\CurrentVersion\Uninstall\$($1Password.PSCh
ildName)"" -ErrorAction SilentlyContinue"
Add-Content -Path "$FilePath" -Value "if (`$null -eq `$i)"
Add-Content -Path "$FilePath" -Value "{"
Add-Content -Path "$FilePath" -Value "## Key Does NOT Exist"
Add-Content -Path "$FilePath" -Value """Key Does NOT Exist"""
Add-Content -Path "$FilePath" -Value "Exit 1"
Add-Content -Path "$FilePath" -Value "}"
Add-Content -Path "$FilePath" -Value "else"
Add-Content -Path "$FilePath" -Value "{"
Add-Content -Path "$FilePath" -Value "`$r = Get-ItemProperty
```

```
""HKU:\`$LoggedOnUserSID\Software\Microsoft\Windows\CurrentVersion\Uninstall\$($1Password.PSChildName)"" -Name 'DisplayVersion' -ErrorAction SilentlyContinue |"
Add-Content -Path "$FilePath" -Value "Select -ExpandProperty 'DisplayVersion'"
Add-Content -Path "$FilePath" -Value "If (`$r -ge '$($1Password.DisplayVersion)')"
Add-Content -Path "$FilePath" -Value "{"
Add-Content -Path "$FilePath" -Value "## Installed"
Add-Content -Path "$FilePath" -Value """Installed"""
Add-Content -Path "$FilePath" -Value "Exit 0"
Add-Content -Path "$FilePath" -Value "}"
Add-Content -Path "$FilePath" -Value "else"
Add-Content -Path "$FilePath" -Value "{"
Add-Content -Path "$FilePath" -Value "## Correct App Version NOT Installed"
Add-Content -Path "$FilePath" -Value """Correct App Version NOT Installed"""
Add-Content -Path "$FilePath" -Value "Exit 1"
Add-Content -Path "$FilePath" -Value "}"
Add-Content -Path "$FilePath" -Value "}"
Add-Content -Path "$FilePath" -Value "}"
Add-Content -Path "$FilePath" -Value "Else"
Add-Content -Path "$FilePath" -Value "{"
Add-Content -Path "$FilePath" -Value "## No Logged on User Detected"
Add-Content -Path "$FilePath" -Value """No Logged on User Detected"""
Add-Content -Path "$FilePath" -Value "Exit 1"
Add-Content -Path "$FilePath" -Value "}"
Invoke-Item $FilePath
```

• Click Run Script (F5)
• A text file will open with the 1Password Detection Method script required to detect the current version of 1Password that is installed on the device you are running the script from.

From https://silentinstallhq.com/create-a-custom-detection-script-for-1password-powershell/

How to change the owner of an Azure Active Directory device

In Azure AD, you can see that each device has an owner. The owner is the user who joined the device to Azure AD, which is sometimes the administrator account. If you want to change the owner, you won't be able to do so through the Azure portal. That is why in this post, I will show you how to change the owner of an Azure AD device using PowerShell.

PowerShell Workaround

First, you must ensure the AzureAD module is installed on your computer and then imported into your PowerShell session. To do that, you should use the following commands.

```
Install-Module AzureAD
Import-module AzureAD
```

Once you have imported the module, you are ready to start.

Connect to Azure Active Directory

The easiest way to get started is to log in interactively at the command line.

```
Connect-AzureAD
```

Locate the device

To get the device object in your tenant, you must use the `Get-AzureADDevice` cmdlet and pass the device name in the `-SearchString` parameter.

```
$device=Get-AzureADDevice `
    -searchString "SAD001"
```

To change the owner property on a device, you must know the value of the "ObjectId" property of the device in question. I will store the device object in the `$device` variable to improve the code

reading. If you don't know the device name or want to list all devices, you should use the `Get-AzureADDevice` cmdlet without any parameters.

Check the current owner of the device

To get the current registered owner for the device, you should use the `Get-AzureADDeviceRegisteredOwner` cmdlet with the following syntax.

```
(Get-AzureADDeviceRegisteredOwner -ObjectId $device.ObjectId).DisplayName
```

> **Important:** Hybrid Azure AD joined Windows 10, or newer devices don't have an owner.

Add an owner for the device

To add a user as an owner to a device, the user must be registered in your tenant and know the value of the user's "ObjectId" property. I will store the user object in the $user variable to improve code readability.

```
$owner=Get-AzureADUser `
    -searchString "Jorge Bernhardt"
```

Once the user object is stored in the `$owner` variable, you should use the `Add-AzureADDeviceRegisteredOwner` cmdlet with the following syntax to add the user as the device's new owner.

```
Add-AzureADDeviceRegisteredOwner `
    -ObjectId $device.ObjectId `
    -RefObjectId $owner.ObjectId
```

The device object can have more than one owner, but the Azure portal will only display the last added owner.

Remove the owner of the device

Using the following syntax, you can always remove a device owner using the `Remove-AzureADDeviceRegisteredOwner` cmdlet.

```
$user=Get-AzureADUser `
    -searchString "some user"
Remove-AzureADDeviceRegisteredOwner `
    -ObjectId $device.ObjectId `
    -OwnerId $user.ObjectId
```

Verify the changes made

Once the previous step is done, to verify that the change was successful, use the `Get-AzureADDeviceRegisteredOwner` cmdlet with the following syntax.

```
Get-AzureADDeviceRegisteredOwner `
    -ObjectId $device.ObjectId
```

get-azureaddeviceregisteredowner.ps1__hu04ac47f3f3b928abd03e0ae300595b41_18816_1700x0_r

Original Article