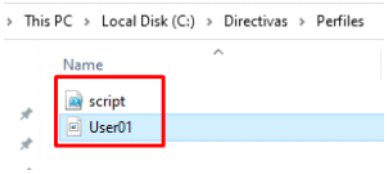
GPO´s AVATARES (Perfil Usuario)

martes, 3 de octubre de 2023 11:18

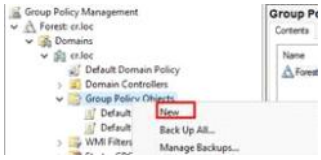
-Nuestro primer paso ha sido, dentro de la carpeta compartida llamada **Directivas**, crear una nueva carpeta nombrada **Perfiles**, en la que añadimos una foto a nuestro gusto (**User01**) y el script que se ejecutará en los equipos de los usuarios (**script**):

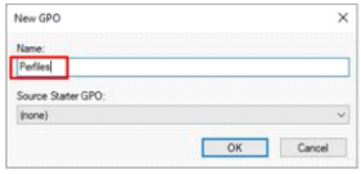


-A continuación, añadimos estos comandos, para asignar dicha foto (descarga) al usuario User01 en este caso, si tuviéramos más usuarios haríamos lo mismo, pero con otros usuarios y otras fotos:

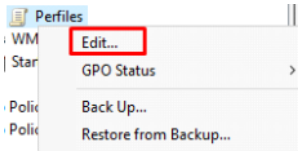
```
PS C:\Users\Administrator> $photo = [byte[]](Get-Content C:\Directivas\Perfiles\descarga.jpg -Encoding byte)
Set-ADUser User01 -Replace @{thumbnailPhoto=$photo}

PS C:\Users\Administrator>
```

-Creamos una política nueva llamada Perfiles:

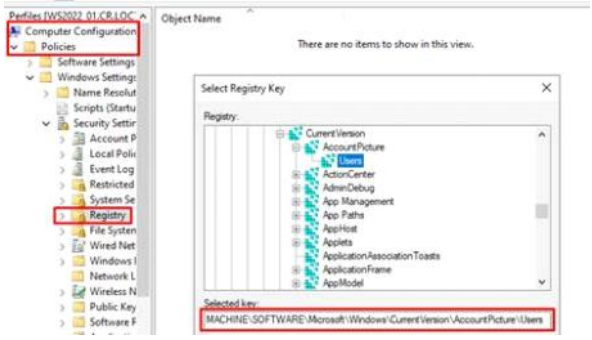




-Entramos a editarla dando click derecho sobre ella – **edit:**

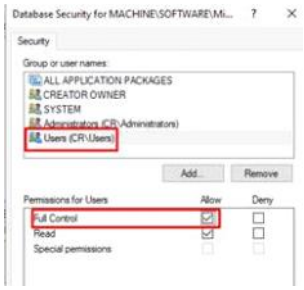


-Damos permisos control total a los usuarios para que puedan cambiar la foto de perfil en sus respectivos equipos:

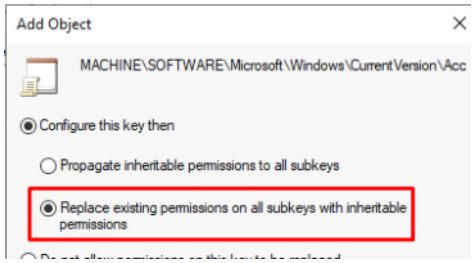


-Script:

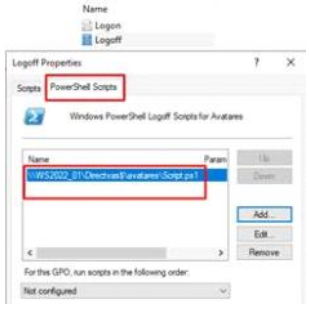
```
Function ResizeImage {
Param (
[Parameter(Mandatory = $True, HelpMessage = "image in byte")]
[ValidateNotNull()]
$imageSource,
[Parameter(Mandatory = $true, HelpMessage = "Betwwen 16 and 1000")]
[ValidateRange(16, 1000)]
$canvasSize,
[Parameter(Mandatory = $true, HelpMessage = "Between 1 and 100")]
[ValidateRange(1, 100)]
$ImgQuality = 100
)
[void][System.Reflection.Assembly]::LoadWithPartialName("System.Drawing")
$imageBytes = [byte[]]$imageSource
$ms = New-Object IO.MemoryStream($imageBytes, 0, $imageBytes.Length)
$ms.Write($imageBytes, 0, $imageBytes.Length);
$bmp = [System.Drawing.Image]::FromStream($ms, $true)
# Image size after conversion
$canvasWidth = $canvasSize
$canvasHeight = $canvasSize
# Set picture quality
$myEncoder = [System.Drawing.Imaging.Encoder]::Quality
$encoderParams = New-Object System.Drawing.Imaging.EncoderParameters(1)
$encoderParams.Param[0] = New-Object System.Drawing.Imaging.EncoderParameter($myEncoder, $ImgQuality)
# Get image type
$myImageCodecInfo = [System.Drawing.Imaging.ImageCodecInfo]::GetImageEncoders() | Where-Object { $_.MimeType -
eq 'image/jpeg' }
# Get aspect ration
$ratioX = $canvasWidth / $bmp.Width;
$ratioY = $canvasHeight / $bmp.Height;
$ratio = $ratioY
if ($ratioX -le $ratioY) {
$ratio = $ratioX
}

# Create an empty picture
$newWidth = [int] ($bmp.Width * $ratio)
$newHeight = [int] ($bmp.Height * $ratio)
$bmpResized = New-Object System.Drawing.Bitmap($newWidth, $newHeight)
$graph = [System.Drawing.Graphics]::FromImage($bmpResized)
$graph.Clear([System.Drawing.Color]::White)
$graph.DrawImage($bmp, 0, 0 , $newWidth, $newHeight)
# Create an empty stream
$ms = New-Object IO.MemoryStream
$bmpResized.Save($ms, $myImageCodecInfo, $($encoderParams))
# cleanup
$bmpResized.Dispose()
$bmp.Dispose()
return $ms.ToArray()
}
$ADUserInfo = ([ADSISearcher]"(&(objectCategory=User)(SAMAccountName=$env:username))").FindOne().Properties
$ADUserInfo_sid = [System.Security.Principal.WindowsIdentity]::GetCurrent().User.Value
If ($ADUserInfo.thumbnailphoto) {
$img_sizes = @(32, 40, 48, 96, 192, 200, 240, 448)
$img_base = "C:\Users\Public\AccountPictures"
$reg_key = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\AccountPicture\Users\$ADUserInfo_sid"
If ((Test-Path -Path $reg_key) -eq $false) { New-Item -Path $reg_key } { write-verbose "Reg key exist [$reg_key]" }
Try {
ForEach ($size in $img_sizes) {
$dir = $img_base + "\" + $ADUserInfo_sid
If ((Test-Path -Path $dir) -eq $false) { $(New-Item -ItemType directory -Path $dir).Attributes = "Hidden" }
$file_name = "Image$($size).jpg"
$path = $dir + "\" + $file_name
Write-Verbose " Crete file: [$file_name]"
try {
ResizeImage -imageSource $($ADUserInfo.thumbnailphoto) -canvasSize $size -ImgQuality 100 | Set-Content -Path $path -
Encoding Byte -Force -ErrorAction Stop
Write-Verbose " File saved: [$file_name]"
}
catch {
If (Test-Path -Path $path) {
Write-Warning "File exist [$path]"
}
else {
Write-Warning "File not exist [$path]"
}
}
$name = "Image$size"
try {
$null = New-ItemProperty -Path $reg_key -Name $name -Value $path -Force -ErrorAction Stop
}
catch {
Write-Warning "Reg key edit error [$reg_key] [$name]"
}
}
}
Catch {
Write-Error "Check permissions to files or registry."
}
}
```



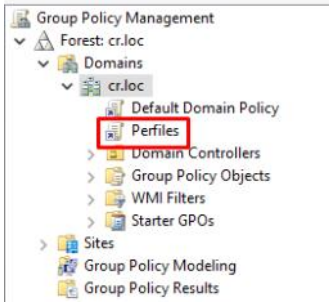
-Remplazamos los permisos anteriores y añadimos los nuevos:



-Una vez activada la política de equipo, entramos en las políticas de usuario, para especificar que, cuando los usuarios tengan la sesión cerrada, ejecute el equipo automáticamente el script creado:



Añadimos la ruta compartida, si no, los usuarios no reconocen la ruta si añadimos la local.
-Linkamos la directiva de Perfiles al dominio, ya que es, tanto de tipo equipo como de tipo usuario:



-Y por último forzamos la actualización de las GPO´s:

```
C:\Users\Administrator>gpupdate /force
Updating policy...

Computer Policy update has completed successfully.
User Policy update has completed successfully.
```

-Comprobación **(reiniciamos el equipo y cerramos sesión una o dos veces, para que se apliquen las configuraciones):**