Budapesti Műszaki és Gazdaságtudományi Egyetem
Intelligens rendszerfelügyelet (BMEVMIA370)

1-C Címtárak

Házi feladat

Koczka Tamás
2011. március 26.

1 Bevezető

1.1 A házi feladat célja (feladatkiírás)

1.1.1 A kitűzött feladat

Egy vállalati portál alkalmazást fejlesztünk, mely a felhasználókat és azok adatait a központi címtárból veszi. Így ha a címtárban valamely felhasználó adata nincs kitöltve rendesen, akkor az a mi portálunkban is hibásan fog megjelenni. Ezért az éles üzembe állítás előtt még meg kell tisztítani a címtár adatait.

Ehhez segítségként készítsünk egy olyan PowerShell szkriptet, ami megkeresi a problémás felhasználókat, és elmenti az adataikat egy fájlba. A szkriptnek a megadott szervezeti egységben (Organizational Unit) tetszőlegesen mély hierarchiaszinten lévő felhasználókat kell megvizsgálnia, és azokat kigyűjtenie, amiknél a Department vagy a Title tulajdonságok nincsenek kitöltve. A problémás felhasználók login nevét, teljes nevét és a DN attribútumokat kell egy CSV fájlba elmenteni.

Lehetőséget kell biztosítani arra, hogy megadjunk egy bemeneti fájlban egy felhasználóneveket tartalmazó szűrő listát. Ha ezt megadjuk, akkor az ebben a listában szereplő felhasználókat nem kell berakni a kimenetbe.

1.1.2 A szkript elnevezése és paraméterezése

```
Get-ProblematicADUsers.ps1 -ouName <string> -filterFile <file path> -outFile <file
path>
```

A szkriptnek kötelező ezt az elnevezést és paraméterezést használnia, egyéb esetben a házi feladatot nem fogadjuk el.

A szkript a paramétereket a következő formában fogadja:

- ouName: a szervezeti egység DN-je, amiben keresni kell, kötelező megadni,
- filterFile: a szűrő listát tartalmazó fájl elérési útja, opcionális,
- outFile: a kimeneti fájl elérési útja, kötelező.

Álljon itt egy példa a szkript egy lehetséges, helyes használatára:

```
Get-ProblematicADUsers.ps1 -ouName "ou=students,dc=irf,dc=local" -filterFile
"c:\temp\filter.txt" -outFile "problematic-users.csv"
```

1.1.3 Bemeneti fájl

A bemeneti fájl egy egyszerű, UTF-8 kódolású szövegfájl, melyben minden sorban egy login név szerepel. A fájlnak nincsen fejléc sora

1.1.4 Kimeneti fájl

A kimeneti fájl egy CSV fájl a következő oszlopnevekkel: login, fullName, DN.

Példa:

```
login,fullName,DN
"kovacs","Kovács István","cn=Kovács István,ou=students,dc=irf,dc=local"
"gipszj","Gipsz Jakab","cn=Gipsz Jakab,ou=students,dc=irf,dc=local"
```

1.1.5 További elvárások

- Figyeljen a hibakezelésre, ellenőrizze, hogy a megadott OU és a bemeneti fájl létezik-e, valamint, hogy a kimeneti fájl nem létezik!
- Vegyen fel további, saját felhasználókat az Active Directoryba (legalább 5 darab), és a megoldását már ezekkel tesztelje és dokumentálja!

1.2 A feladat pontosítása

- A szűrőfájlban található felhasználónevek alatt a login nevet értjük
- A login név alatt a *sAMAccountName* tulajdonságot, azaz a Windows 2000 előtt is érvényes belépési nevet értjük

2 Felhasználói dokumentáció

2.1 Előkövetelmények

- PowerShell v2
- Active Directory Module
- (Domain + Domain Controller Active Directory-val)

2.2 A szkript meghívása

A szkript meghívása a 1.1.2 és 1.1.3 pontban ismertetett módon történik.

Kiegészítésként annyival bővült a funkcionalitása, hogy amennyiben nem adjuk meg explicit a paraméterek neveit, akkor a következő felépítés követendő: ouName, outFile[, filterFile].

Tehát a *filterFile* elhagyható, opcionális paraméter, illetve az összes paramétert egyetlen *string*ként kell megadni.

Példák a paraméternév nélküli szkripthívásra:

```
Get-ProblematicADUsers.ps1 "ou=students,dc=irf,dc=local" "problematic-users.csv"

Get-ProblematicADUsers.ps1 "ou=students,dc=irf,dc=local" "problematic-users.csv"
"c:\temp\filter.txt"
```

2.3 A kimenetek értelmezése

A szkript hiányzó paraméter esetén exceptiont dob, egyébként szövegesen jeleníti meg a különböző hibaüzeneteket.

Amennyiben nem ír ki semmit, úgy helyesnek tekinthető a lefutása.

Továbbá az automatizált feldolgozás esetére visszatérési értékkel tér vissza, ami jelzi a futtatása sikerességét / sikertelenségét:

- *$true*: sikeres futtás esetén
- *$false*: amennyiben valamilyen hiba meghiúsította a futást

3 Fejlesztői dokumentáció

3.1 Általános tudnivalók

A program hibakezelésének azt a módszert választottam, hogy hiba esetén a konzolra kiír egy a hibát leíró szöveget, majd visszatér *$false* értékkel.

3.2 Felhasznált változók és rövid leírásuk

string	*$ouName*	bemeneti változó, a keresendő szervezeti egység DN-je
string	*$outFile*	bemeneti változó, a célfájl elérési útvonala, ahova az eredmény íródik
string	*$filterFile*	bemeneti változó, a szűrőfájl elérési útvonala
string[]	*$filterUsers*	tömb, a szűrőfájl tartalma, soronként egy felhasználó neve
string	*$ouFullPath*	segédváltozó, tartalmazza az OU AD provider általi elérési útvonalát
ADUser[]	*$badUsers*	azok a felhasználók, akiknek hiányzik a Title vagy Departure tulajdonsága
ADUser[]	*$usersToReport*	a *$badUsers* szűrt verziója a szűrőfájl által
PSCustomObject[]	*$reportedData*	objektumok tömbje, aminek a formátuma már megegyezik a kimenete formátumával

3.3 Bemeneti paraméterek, azok ellenőrzése

3.3.1 Paraméter lista

A 1.1.2 pontban ismertetett bemeneti paramétereket várjuk el.

Mivel mindegyik vagy fájlnevet vagy egy OU-t vár el, ezért legcélszerűbb volt ezeket *string* típusúnak választani.

Az *ouName* és *outFile* kötelező paraméterek, ezért a paraméter lista elejére kerültek (a feladatkiírás nem rendelkezett a paraméterek sorrendjéről), így lehetséges a szkript 2 és 3 paraméteres meghívása is a paraméter nevek elhagyása nélkül is.

Amennyiben kötelező paramétert hagyunk el, azt a megfelelő hibaüzenettel jelzi. Ha a *filterFile* nem lett megadva, akkor azt üres *string*ként jelöljük.

```
# paraméterek ellenőrzése: az ouName és outFile kötelező paraméterek, ezért
is kerültek a lista elejére
# (a feladatkiírás nem követelte meg a sorrendet és így könnyebb használni)
# mindegyik string típusú, a filterFile üresen hagyása jelenti a paraméter
elhagyását
param
(
      [string] $ouName = $(throw "ouName (organization unit) is a required
parameter!"),
      [string] $outFile = $(throw "outFile (output file path) is a required
parameter!"),
      [string] $filterFile = ""
)
```

3.3.2 A szűrőfájl ellenőrzése, feldolgozása

A fájlban található felhasználókat egy tömbbe gyűjtjük ki: minden egyes sor a fájlban egy tömb elem lesz.

Először létrehozunk egy üres tömböt (*$filterUsers* változó), majd ha lett megadva *filterFile* (nem üres string szerepel helyette), akkor ellenőrizzük, hogy a fájl létezik-e a *Test-Path* függvénnyel.

Amennyiben nem létezik, úgy az ezt jelző hibával visszatérünk.

Ha létezik, akkor pedig a fájl tartalmával egyszerűen lecseréljük a tömbünket, erre a *Get-Content* függvényt használva.

```
# ha nem adtak meg filterezendő felhasználókat, akkor üresen hagyjuk a
tömböt
$filterUsers = @();
```

```
# ha mégis adtak meg filterezéshez fájlt, akkor azt elkezdjük feldolgozni
if($filterFile -ne "")
{
	# ha nem létezik a fájl, akkor hibával térünk vissza
	if(!(Test-Path $filterFile))
	{
		Write-Host("Filter file not found, path: " + $filterFile);
		return $false;
	}

	# ha létezik a fájl, akkor a sorait egyszerűen átalakjuk a tömb
elemeivé (annyi tömb elem, ahány sor)
	$filterUsers = Get-Content $filterFile;
}
```

3.3.3 A célfájl felülírásának ellenőrzése

A célfájlt meglétét közvetlenül ellenőrizzük a *Test-Path* függvénnyel. Amennyiben létezik a fájl, hibával térünk vissza, mert nem akarjuk felülírni azt (lásd 1.1.5).

```
# ha már létezik a cél fájl, akkor hibával térünk vissza
if(Test-Path $outFile)
{
	Write-Host("Output file already exists, path: " + $outFile);
	return $false;
}
```

3.3.4 Előfeltételek (AD modul) ellenőrzése

A szkript használatához szükség van az *Active Directory Module*-ra, aminek meglétét úgy ellenőrizzük, hogy a *Test-Path* függvénnyel teszteljük, hogy a *AD:* könyvtár létezik-e. Amennyiben nem létezik, úgy feltételezzük, hogy a modul nincs betöltve és hibaüzenettel kilépünk.

```
# ha nem létezik az AD provider, akkor feltételezzük, hogy a modul nincs
betöltve,
# és megpróbáljuk betölteni, ha ez sem sikerül, akkor hibaüzenettel térünk
vissza
if(!(Test-Path 'AD:\'))
{
	try
	{
		Import-Module ActiveDirectory
	}
	catch [Exception]
	{
		Write-Host("Active Directory Module cannot be loaded! (Path
AD:\ not available)");
		Write-Host("Detailed exception: " + $_.Exception.ToString());
		return $false;
	}
}
```

3.3.5 A lekérdezendő OU meglétének ellenőrzése

Az *ouName* változóban megadott és a lekérdezés alapjául szolgáló *Organization Unit*ot teszteljük, hogy létezik-e. Ehhez felhasználjuk az *AD provider*t, azaz előállítjuk az szervezeti egységhez tartozó könyvtárnak az elérési útvonalát (az AD meghajtóhoz hozzáfűzzük a szervezeti egység nevét) és ennek meglétét ellenőrizzük a *Test-Path* függvénnyel. Ha nem létezik, akkor hibával térünk vissza.

```
# ha nem létezik az adott OU, akkor hibával térünk vissza
$ouFullPath = 'AD:\' + $ouName;
if(!(Test-Path $ouFullPath))
{
      Write-Host("Organization unit not exists, path: " + $ouFullPath);
      return $false;
}
```

3.4 A problémás felhasználók kigyűjtése és mentése

3.4.1 A hiányzó adatokkal rendelkező felhasználók kigyűjtése

A *Get-ADUser* függvényt használjuk fel a felhasználók kigyűjtésére.

Hogy a keresés a megadott szervezeti egységen belül történjen, ahhoz megadjuk a függvénynek a *SearchBase* paraméterben a keresés alapját képző szervezeti egységet, amit a *$ouName* változóban tároltunk el.

A feladatkiírás szerint tetszőleges hierarchiaszinten lévő felhasználókat meg kell találnunk. Ehhez a *SearchScope* paraméter *Subtree* beállítását használhatjuk, de mivel a függvénynek ez az alapértéke [2], ezért ez a beállítás itt el is hagyható lenne.

A keresést szűkítenünk kell csak azokra a felhasználókra, akiknek a *Title* vagy *Department* tulajdonságai hiányoznak. Ehhez a *Filter* paramétert tudjuk felhasználni, ahol úgy tudjuk megvalósítani, hogy az adott tulajdonságokra a `-notlike "*"` keresőfeltételt értelmeztetjük ki. Ez „nyers fordításban” azt jelenti, hogy ha már akármilyen illeszkedést talál, akkor nem fogadja el a felhasználót. A nem létező tulajdonság esetén viszont belerakja az eredményhalmazba. Mivel akármelyik tulajdonság hiányzása esetén ki szeretnénk listázni a felhasználót, ezért a *vagy* operátort (`-or`) használjuk fel a feltételek között.

Mivel a *Get-ADUser* függvény visszatérhet tömbbel is (több találat volt) vagy egy objektummal (csak egy felhasználó elégítette ki a feltételeket), ezért ez inkonzisztens működéshez vezethetne. Ezért az eredményhalmazt átalakítjuk a `@(…)` nyelvi elemmel tömbbé, mert így konzisztens működéshez jutunk, hisz mindig AD-Userek tömbjeként tekinthetünk a *$badUsers* változóra, amibe letároljuk az eredményt. Itt fontos megjegyezni még, hogy ha a *Get-ADUser* visszatérési értéke már eredetileg tömb volt, akkor a `@(…)` nyelvi elem nem készít belőle többszintű tömböt, hanem ugyanazt a tömböt adja vissza.

```
# kigyűjtjük azokat a Userek, akiknek a title vagy department mezője nem
létezik,
#   a keresés gyökere a megadott OU
# ha a visszatérési érték nem tömb lenne, akkor többé alakítjuk a @(...)
tömbképző elemmel,
#   amennyiben már tömb, akkor nem történik vele semmi
# megj: esetleg teljesítmény szempontból érdemes lehet már itt a -
Properties paraméterrel csak azokra a mezőkre szűrni,
#       amire később szükségünk lesz

$badUsers = @(Get-ADUser -Filter {(title -notlike "*" -or department -
notlike "*")} -SearchBase $ouName -SearchScope Subtree);
```

3.4.2 Felhasználók kiszűrési a szűrési lista alapján

Ehhez az előzőleg lekért, hiányzó tulajdonságokkal rendelkező felhasználókat tartalmazó *$badUsers* tömböt szűrjük még pedig úgy, hogy minden egyes elemén végig megyünk és ha az adott felhasználónak a nevét nem tartalmazza a *$filterUsers* lista, akkor vesszük bele az eredménybe (`-notcontains` operátor).

Az eredményt tömbként kezelve tároljuk a *$usersToReport* változóban.

```
# leszűrjük azokat a Usereket, akikről a reportot szeretnénk készíteni,
# ehhez az kell, hogy ne tartalmazza a nevüket a filterUsers tömb
$usersToReport = @($badUsers | ? { $filterUsers -notcontains
$_.sAMAccountName })
```

3.4.3 Eredmény megfelelő formátumúra alakítása és az eredmények mentése

A már kigyűjtött felhasználók listáját már csak megfelelő formátumúra (1.1.4) kell alakítani.

Ezért egy egyedi objektumokból (*PSCustomObject* típus) álló listát készítünk, aminek a változónevei megegyeznek a kimeneti formátum által kért CSV fájl mezőneveivel.

A *login* névnek az *ADUser sAMAccountName*, a *fullName*-nek a *Name*, míg a *DN*-nek a *DistinguishedName* tulajdonságát használjuk fel.

Megj: mivel nem volt meghatározva pontosan, ezért a login névnek a Windows 2000 előtti rendszereken is működő login nevet választottam.

Az így létrejött *$reportedData* tömböt a *ConvertTo-Csv* függvénnyel át tudjuk konvertálni a megfelelő formátumban. A függvénynek még megadjuk a a *NoTypeInformation* módosítót, ami pedig lehagyja a típust tartalmazó kommenteket a fájlból, hogy az jobban illeszkedjen a kért formátumhoz.

A fejlécet tartalmazó első sort lehagyjuk, majd a saját fejlécünket szúrjuk hozzá. (Erre azért van szükség, mert ha nincs kiírandó elem, akkor a *ConvertTo-Csv* függvény nem tér vissza fejléccel).

Majd az így képzett tömböt kiírjuk fájlba az *Out-File* parancssal, megadva a *FilePath* paramétert, ami a kimeneti fájlt határozza meg.

Ehhez a művelethez használunk hibakezelést, mert itt olyan I/O műveletet hajtunk végre, amikor olyan hibák jelentkezhetnek, amire előre nem vagy csak nehézkesen tudunk készülni, illetve nem képezi a script szerves részét a hiba lekezelése (pl. nincs megfelelő jogosultságunk a fájl olvasásához, stb).

Amennyiben sikeres volt a script futása, *$true* értékkel térünk vissza.

```
# a végeredményhez kért mezőket külön kigyűjtjük és a kért nevet adjuk
nekik
$reportedData = @($usersToReport | select -Property `
      @{n="login";    e={$_.sAMAccountName}},
      @{n="fullName"; e={$_.Name}},
      @{n="DN";       e={$_.DistinguishedName}});

# kiírjuk az eredmény a outFile változóban megadott fájlba,
# a típus információkat a feladat kiírásban látható minta alapján elhagyjuk
try
{
      @( "login,fullName,DN",
         ($reportedData | ConvertTo-Csv -NoTypeInformation | select -Skip 1)
      ) | Out-File -FilePath $outFile
}
catch [Exception]
{
      Write-Host("Can't write file: " + $outFile);
      Write-Host("Detailed exception: " + $_.Exception.ToString());
      return $false;
}

# ha minden rendben ment, akkor true-val térünk vissza jelezve, hogy
végrehajtás sikeres volt
return $true;
```

4 Tesztesetek

4.1 A teszteléshez létrehozott új felhasználók

- CN=Tamas Koczka,OU=Engineering,OU=People,DC=irfhf,DC=local
  - Title: hiányzik
  - Department: „d”
- CN=TesztPeople,OU=People,DC=irfhf,DC=local
  - Title: hiányzik
  - Department: hiányzik
- CN=TesztEngFullName,OU=Engineering,OU=People,DC=irfhf,DC=local
  - Title: hiányzik
  - Department: hiányzik
  - DisplayName: hiányzik
- CN=TesztHR,OU=HR,OU=People,DC=irfhf,DC=local
  - Title: An another HR job
  - Department: HR
- CN=TesztManagm,OU=Management,OU=People,DC=irfhf,DC=local
  - Title: Very important job
  - Department: Management
- CN=TesztResearchSuperInner,OU=SuperInner,OU=ResearchInner,OU=Research,OU=People, DC=irfhf,DC=local
  - Title: “Job Title”
  - Department: hiányzik

4.2 Szkript meghívása hiányzó paraméterekkel

```
PS C:\Users\Administrator\Documents> .\Get-ProblematicADUsers.ps1

ouName (organization unit) is a required parameter!
At C:\Users\Administrator\Documents\Get-ProblematicADUsers.ps1:17 char:28
+     [string] $ouName = $(throw <<<<  "ouName (organization unit) is a required
parameter!"),
```

A várakozásoknak megfelelő az első hiányzó paraméter, azaz az ouName hiányára hivatkozva a PowerShell exceptiont dobott és meghiúsitotta a szkript futását.

4.3 Szkript meghívása nem létező szervezeti egységre

```
PS C:\Users\Administrator\Documents> .\Get-ProblematicADUsers.ps1
"ou=NEMLETEZIK,dc=irfhf,dc=local" test3.txt

Organization unit not exists, path: AD:\ou=NEMLETEZIK,dc=irfhf,dc=local
False
```

4.4 Szkript meghívása már létező fájlra

```
PS C:\Users\Administrator\Documents> .\Get-ProblematicADUsers.ps1
"ou=People,dc=irfhf,dc=local" MARLETEZIK.txt

Output file already exists, path: MARLETEZIK.txt
False
```

4.5 Szkript meghívása nem létező filter fájlra

```
PS C:\Users\Administrator\Documents> .\Get-ProblematicADUsers.ps1
"ou=People,dc=irfhf,dc=local" test5.txt NEMLETEZIK.txt

Filter file not found, path: NEMLETEZIK.txt
False
```

4.6 Szkript meghívása a People szervezeti egységre

4.6.1 A meghívás eredménye

```
PS C:\Users\Administrator\Documents> .\Get-ProblematicADUsers.ps1
"ou=People,dc=irfhf,dc=local" test6.csv

True
```

4.6.2 A test6.csv fájl tartalma

```
login,fullName,DN
"pete","Randall Petersen Armstrong","CN=Randall Petersen
Armstrong,OU=Research,OU=People,DC=irfhf,DC=local"
"KoczkaTamas","Tamas Koczka","CN=Tamas
Koczka,OU=Engineering,OU=People,DC=irfhf,DC=local"
"TesztEngLogonPre2000","TesztEngFullName","CN=TesztEngFullName,OU=Engineering,OU=P
eople,DC=irfhf,DC=local"
"TesztPeople","TesztPeople","CN=TesztPeople,OU=People,DC=irfhf,DC=local"
"TesztResearchSuperIn","TesztResearchSuperInner","CN=TesztResearchSuperInner,OU=Su
perInner,OU=ResearchInner,OU=Research,OU=People,DC=irfhf,DC=local"
```

4.7 Szkript meghívása a Engineering szervezeti egységre

4.7.1 A meghívás eredménye

```
PS C:\Users\Administrator\Documents> .\Get-ProblematicADUsers.ps1 -outFile
"test7.csv" -ouName "ou=Engineering,ou=People,dc=irfhf,dc=local"

True
```

4.7.2 A test7.csv fájl tartalma

```
login,fullName,DN
"KoczkaTamas","Tamas Koczka","CN=Tamas
Koczka,OU=Engineering,OU=People,DC=irfhf,DC=local"
"TesztEngLogonPre2000","TesztEngFullName","CN=TesztEngFullName,OU=Engineering,OU=P
eople,DC=irfhf,DC=local"
```

4.8 Szkript meghívása a People szervezeti egységre szűréssel

4.8.1 A filter8.txt fájl tartalma

```
KoczkaTamas
TesztResearchSuperIn
```

4.8.2 A meghívás eredménye

```
PS C:\Users\Administrator\Documents> .\Get-ProblematicADUsers.ps1
"ou=People,dc=irfhf,dc=local" test8.csv filter8.txt

True
```

4.8.3 A test8.csv fájl tartalma

```
login,fullName,DN
"pete","Randall Petersen Armstrong","CN=Randall Petersen
Armstrong,OU=Research,OU=People,DC=irfhf,DC=local"
"TesztEngLogonPre2000","TesztEngFullName","CN=TesztEngFullName,OU=Engineering,OU=P
eople,DC=irfhf,DC=local"
"TesztPeople","TesztPeople","CN=TesztPeople,OU=People,DC=irfhf,DC=local"
```

4.9 Szkript meghívása a HR szervezeti egységre (ebben nincs hibás felhasználó)

4.9.1 A meghívás eredménye

```
PS C:\Users\Administrator\Documents> .\Get-ProblematicADUsers.ps1
"ou=HR,ou=People,dc=irfhf,dc=local" test9.csv

True
```

4.9.2 A test9.csv fájl tartalma

```
login,fullName,DN
```

4.10 Szkript meghívása a ResearchInner szervezeti egységre (ebben 1 hibás felhasználó van)

4.10.1 A meghívás eredménye

```
PS C:\Users\Administrator\Documents> .\Get-ProblematicADUsers.ps1
"ou=ResearchInner,ou=Research,ou=People,dc=irfhf,dc=local" test10.csv

True
```

4.10.2 A test10.csv fájl tartalma

```
login,fullName,DN
"TesztResearchSuperIn","TesztResearchSuperInner","CN=TesztResearchSuperInner,OU=Su
perInner,OU=ResearchInner,OU=Research,OU=People,DC=irfhf,DC=local"
```

Hivatkozások

[1] BME MIT, *Hogyan készítsünk házi feladat dokumentációt és mérési jegyzőkönyvet,* elérhető online: http://www.inf.mit.bme.hu/edu/dokumentacio

[2] http://technet.microsoft.com/en-us/library/ee617241.aspx