Active directory management  
In a month on lunches

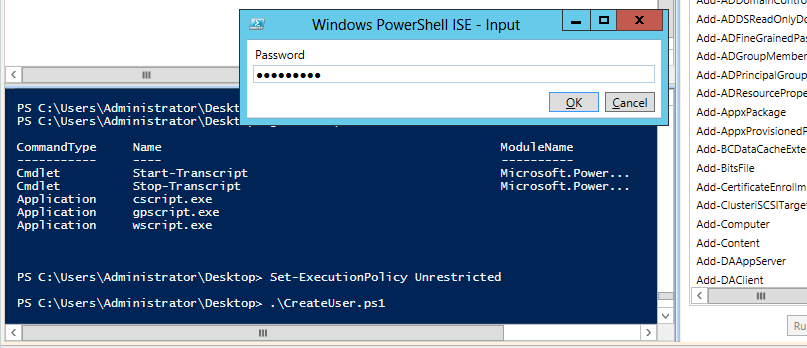
# Part 1

## Chapter 2: Creating user accounts

Command om Active directory cmdlets te verkrijgen: *Import-Module ActiveDirectory*Command om scripts te laten runnen: *Set-ExecutionPolicy Unrestricted*  
Daarnaast altijd powershell als administrator runnen.

### Aanmaken van een enkel account

Script om een nieuwe gebruiker toe te voegen:  
*$secpass = Read-Host “Password” –AsSecureString  
New-ADUser –Name “GREEN Dave” –SamAccountName dgreen –UserPrincipalName “dgreen@Assengraaf.nl” –AccountPassword $secpass –Path “cn=Users, dc= Assengraaf, dc = nl” –Enabled:$true*



Let wel, als je dit wil uitvoeren moet je uiteraard de toestemming hebben om scripts uit te voeren. Die toestemming kan je veranderen met het commando *Set-ExecutionPolicy.*

Er zal een pop-up verschijnen die je vraagt om een wachtwoord op te geven. Zeer belangrijk dat je dit zo doet want anders staat het wachtwoord gewoon in het script zelf wat een veiligheidsrisico met zich mee brengt.

Bij het aanmaken van een nieuwe gebruiker kan men ook gebruik maken van een al bestaand account. Als er verschillende instellingen hetzelfde zijn kan dit dus handig zijn.  
Bijvoorbeeld:

*$secpass = Read-Host “Password” –AsSecureString  
$user = Get-ADUser –Identity jgreen* ***–Properties memberof, office*** *New-ADUser –Name “GREEN Dave” –SamAccountName dgreen –UserPrincipalName “dgreen@Assengraaf.nl” –AccountPassword $secpass –Path “cn=Users, dc= Assengraaf, dc = nl” –Enabled:$true* ***–Instance $user***

### Aanmaken van meerdere accounts

Hierbij maak je gebruik van een .csv bestand waar al de gebruikers in staan.

Een script hiervoor:

*$secpass = Read-Host “Password” –AsSecureString  
Import-Csv gebruikersnamen.csv |  
foreach {  
 $name = “$($\_.LastName) $($\_.FirstName)”*

*New-ADUser –GivenName $($\_.FirstName) –Surname $($\_.LastName) –Name $name`  
-SamAccountName $($\_.SamAccountName) –UserPrincipalName “$($\_.SamAccountName)@Assengraaf.nl” –AccountPassword $secpass –Path “cn=Users, dc= Assengraaf, dc = nl” –Enabled:$true*

*}*

OPMERKING: $\_.FirstName kan je ook noteren als $PSItem.FirstName sinds powershell v3

### Aanmaken Managed service account

Deze accounts kunnen gebruikt worden om services zoals SQL server, exchange en IIS te beheren.  
Het commando om dit type account aan te maken is iets anders:

*New-ADServiceAccount –Name TestAccount –Path “CN= Managed Service Accounts,DC=Assengraaf, DC=nl” –DNSHostName TestAccount.Assengraaf.nl –Enabled $true  
Add-KdsRootKey –EffectiveImmediately*

Het boek gaat hier niet echt op in, het geeft gewoon aan dat indien je dit moet toe voegen je het zo kan doen.

## Chapter 3: Managing user accounts

Om van een bestaand account de gegevens te veranderen bestaat volgend commando:

*Set-ADUser –Identity “CN= GREEN Bill, CN=Users, DC=Assengraaf, DC=nl –OfficePhone “01234 567895”*

Met deze ‘Set-ADUser’ kan bij elk bestaand account een property toegevoegd of gewijzigd worden. Er zijn 4 paramters: Remove, Add, Replace en Clear.  
En ter info: na ‘-identity’ is het ook voldoende de SamAccountName ingeven, hier dan ‘bgreen’.

Soms kan het ook voorkomen dat je properties van een ander account wil kopiëren. Dit kan door bv. *$source = Get-ADUser –Identity bgreen –Properties OfficePhone, otherTelephone*.

Daarna:  
*Set-ADUser –Identity dgreen –OfficePhone $($source.officePhone) –Replace @{otherTelephone = $($source.otherTelephone)}*

Zo zijn er eerst 2 properties van een account meegegeven aan een variabele en daarna toegewezen aan een ander account. Dit op deze manier doen bespaart veel knip-en plakwerk in de GUI.

### Enabling en desabling accounts

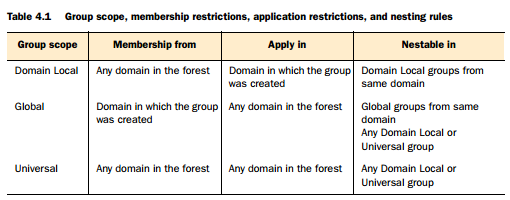
Vb: *Disable –ADAccount –Identity dgreen* of *Enable –ADAccount –Identity dgreen*

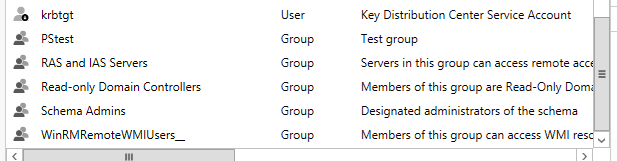
Of een script om te zien welke accounts niet geactiveerd zijn:  
*Search-ADAccount –AccountDisabled | Format-Table Name, DistinguishedName –AutoSize*

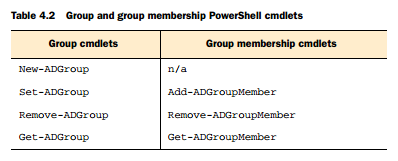
# Chapter 4: Managing groups

Een nieuwe gebruikersgroep aanmaken ziet era ls volgt uit:

*New-ADGroup –Name PStest –Path “CN=Users, DC=Assengraaf, DC=nl” –GroupCategory Security`  
–GroupScope global –Description “Test group”*

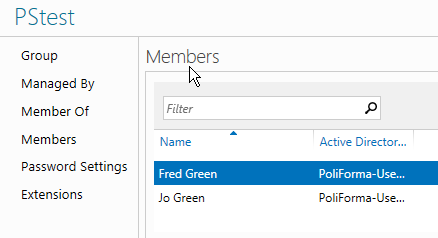
Ter verduidelijking, parameters name en path kennen we al. De ‘GroupCategory’ kan 2 waarden hebben, security of distribution. Wij zullen vaak security gebruiken, distribution is voor het aanmaken van e-mailgroepen in exchange.  
“Groupscope” daarna heeft 3 mogelijke waarden: DomainLocal, Global of Universal.

  
Na het uitvoeren van eerder genoteerd script zou de groep zonder problemen aangemaakt moeten zijn binnen de container Users.

   
Natuurlijk is het binnen powershell ook nog mogelijk na het aanmaken van de groep die te wijzigen zoveel je wil. Daar bestaan een aantal commandlets voor, ik geef ze even aan de hand van een tabel.

Bijvoorbeeld men wil alle gebruikers met de achternaam Green toevoegen aan de groep. Dit gaat zeer gemakkelijk binnen powershell aangezien er binnen haakjes kan gewerkt worden.

*Add-ADGroupMember –Identity PStest –Members (Get-ADUser –LDAPFilter “(sn=Green)”)*

(‘sn’ geeft hier de achternaam aan)

Groepen kunnen natuurlijk ook in bulk gemaakt worden, hierbij wordt simpelweg dezelfde syntax gevolgd als bij het creëren van gebruikers.

*Import-Csv –Path ./groups.csv |  
foreach{  
 New-ADGroup –Name $PSItem.Name –Path “CN= Users, DC=Assengraaf,DC=nl”*

*-GroupCategory Security –GroupScope global –Description $PSItem.Description  
}*

#Reminder: *$PSItem kan ook geschreven worden als $\_*

# Hoofdstuk 5: Troubleshooting users and groups

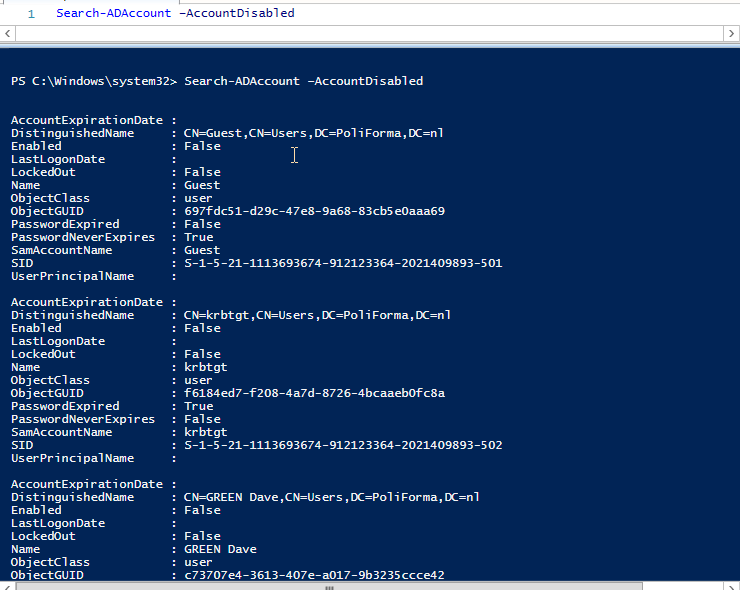
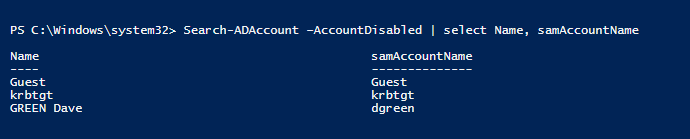
Eerste stappen om te troubleshooten:

1. Kijken als het account geactiveerd is
2. Kijken als het account verlopen is
3. Kijken of een gebruiker buitengesloten is uit zijn/haar account

1.De eerste stap kan je controleren als volgt:

*Search-ADAccount –AccountDisabled | select Name, samAccountName*

Alles na de pipeline is maar om de weergave te verduidelijken.

Een betere, overzichtelijkere weergave voor wat gewenst is:

Om het account dan te enablen:  
*Get-ADUser –Identity dgreen | Enable-ADAccount*

2.Zien of het account verlopen is kan als volgt:  
*Search-ADAccount –AccountExpired | Format-Table Name, SamAccountName, DistinguishedName, accountExpirationDate –AutoSize*

Indien dit het geval is kan je op verschillende manieren een nieuwe verloopsdatum instellen. Ten eerste kan je een vast aantal 30 dagen toevoegen aan het account. Ten tweede kan de verloopsdatum op een vaste datum ingesteld worden. En ten slotte is er ook een derde manier om in te stellen dat het account nooit verloopt.

Een voorbeeld van elk van deze methodes:

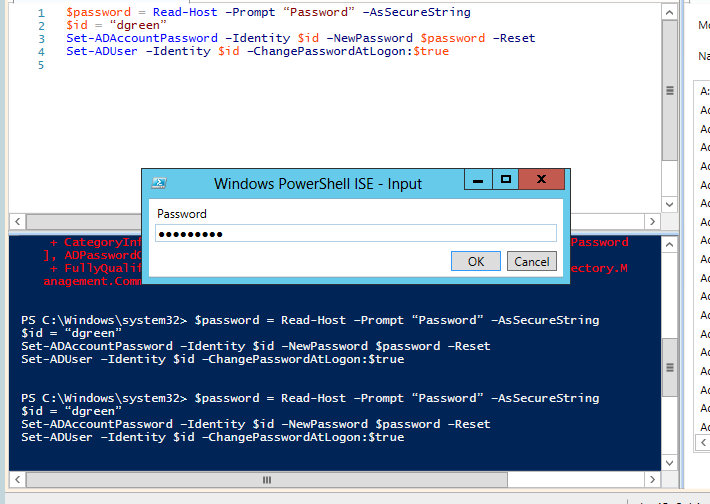
*Get-ADUser –Identity dgreen | Set-ADAccountExpiration –DateTime (Get-Date).AddDays(30)*

*Get-ADUser –Identity dgreen | Set-ADAccountExpiration –DateTime ([datetime]’12/31/2015’  
#****Datum moet volgens maand/dag/jaar syntax***

*Get-ADUser –Identity dgreen | Clear-ADAccountExpiration*

3.Paswoorden in AD moeten standaard om de 42 dagen veranderd worden. Als dit niet gebeurt dan zal het wachtwoord verlopen. Om te zien bij wie dit het geval is:  
*Search-ADAccount –PasswordExpired*

Indien je hier het account vindt die je nodig hebt is de volgende stap het paswoord resetten. Dit moet natuurlijk veilig gebeuren, dus hierbij gebruik je weer de ‘Read-Host’

$password = Read-Host –Prompt “Password” –AsSecureString  
$id = “dgreen”  
Set-ADAccountPassword –Identity $id –NewPassword $password –Reset Set-ADUser –Identity $id –ChangePasswordAtLogon:$true

Opmerking: Let op paswoord complexiteit.

Een gebruiker kan ook buitengesloten worden als hij/zij te veel pogingen onderneemt met een foutief paswoord. Hier is een aparte parameter voor ‘LockedOut’  
Unlocken kan simpelweg met:  
*Unlock-ADAccount –Identity dgreen*

# Hoofdstuk 6: Managing computer accounts

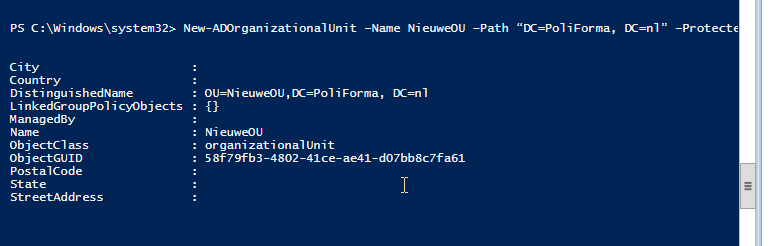
# Hoofdstuk 7: Managing organizational units

De 4 commandlets om met OU’s te werken:  
1. *New-ADOrganizationalUnit*  
2. *Get-ADOrganizationalUnit*  
3. *Set-ADOrganizationalUnit*  
4. *Remove-ADOrganizationalUnit*

In lijn met commandlets van vorige hoofdstukken dus.

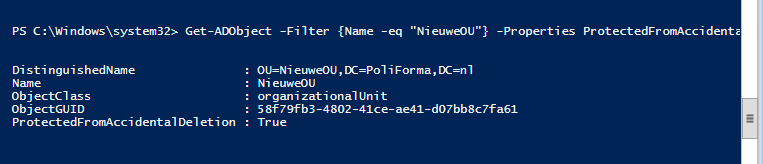
Het creëren van een nieuwe OU gaat als volgt:

*New-ADOrganizationalUnit –Name NieuweOU –Path “DC=Assengraaf, DC=nl” –ProtectedFromAccidentalDeletion:$true –PassThru*

Opmerking: de parameter ‘-ProtectedFromAccidentalDeletion’ is **zeer belangrijk** en het is aan te raden dit standaard bij te voegen bij het creëren van een OU. De ‘-PassThru’ heeft als functie een output weer te geven, zonder dit geeft het script immers niets weer.   
Voorbeeld:

(Ik testte dit uit op een vm binnen het domein PoliForma)

Even controleren of de OU effectief beschermt wordt tegen verwijdering:

*Get-ADObject -Filter {Name -eq "NieuweOU"} -Properties ProtectedFromAccidentalDeletion*

Indien je de OU na een tijd toch zou willen verwijderen filter je gewoon weer naar de naam en pipe je naar Set-ADObject –ProtectedFromAccidentalDeletion $false.

Een gecreëerde OU vereist niet veel werk meer, toch zijn er nog 4 handelingen die je zeker zou moeten weten en kunnen binnen het managen van OU’s.  
Ten eerste is er de naam veranderen van een OU, ten tweede een beschrijving instellen voor de OU. Ook is het mogelijk te veranderen wie verantwoordelijk is voor de OU en ten slotte hoe je de OU kan verwijderen.

De OU een andere naam geven gaat als volgt:

*Get-ADObject –Filter {Name –eq “OudeOU”} | Rename-ADObject –NewName NieuweOU*

Als de naam van de OU niet genoeg weergeeft wat er allemaal in de OU zit is het aan te raden een beschrijving te linken aan de OU.

*Get-ADObject –Filter {Name –eq “NieuweOU”} |Set-ADOrganizationalUnit –Description “Een voorbeeld”*

Een OU heeft typisch ook een gebruiker die de OU ‘managed’. Nu heeft dit zonder het toevoegen van specifieke rechten (zie hoofdstuk 14) geen bijkomende functies buiten het aangeven aan anderen wie verantwoordelijk is voor de OU. Het instellen hiervan staat in lijn met de vorige commands, ipv de parameter ‘Description’ moet er gewoon de parameter ‘ManagedBy’ gebruikt worden.

*Get-ADObject –Filter {Name –eq “NieuweOU”} |Set-ADOrganizationalUnit –ManagedBy (Get-ADUser –Filter {Name –eq “Green Jo”})*

Ten slotte het verwijderen van de OU, wederom in dezelfde lijn:

*Get-ADObject –Filter {Name –eq “NieuweOU”} |Remove-ADOrganizationalUnit –Recursive*

De ‘recursive’ parameter is wel iets nieuws, dit zorgt ervoor dat alle kinderen van de OU ook verwijderd zullen worden om complexiteit tegen te gaan.

Als laatste deeltje van dit hoofdstuk moet je nog weten hoe je objecten kan schuiven tussen OU’s, dit gaat met het command:

*Get-ADObject –Filter {Name –eq “NieuweOU”*