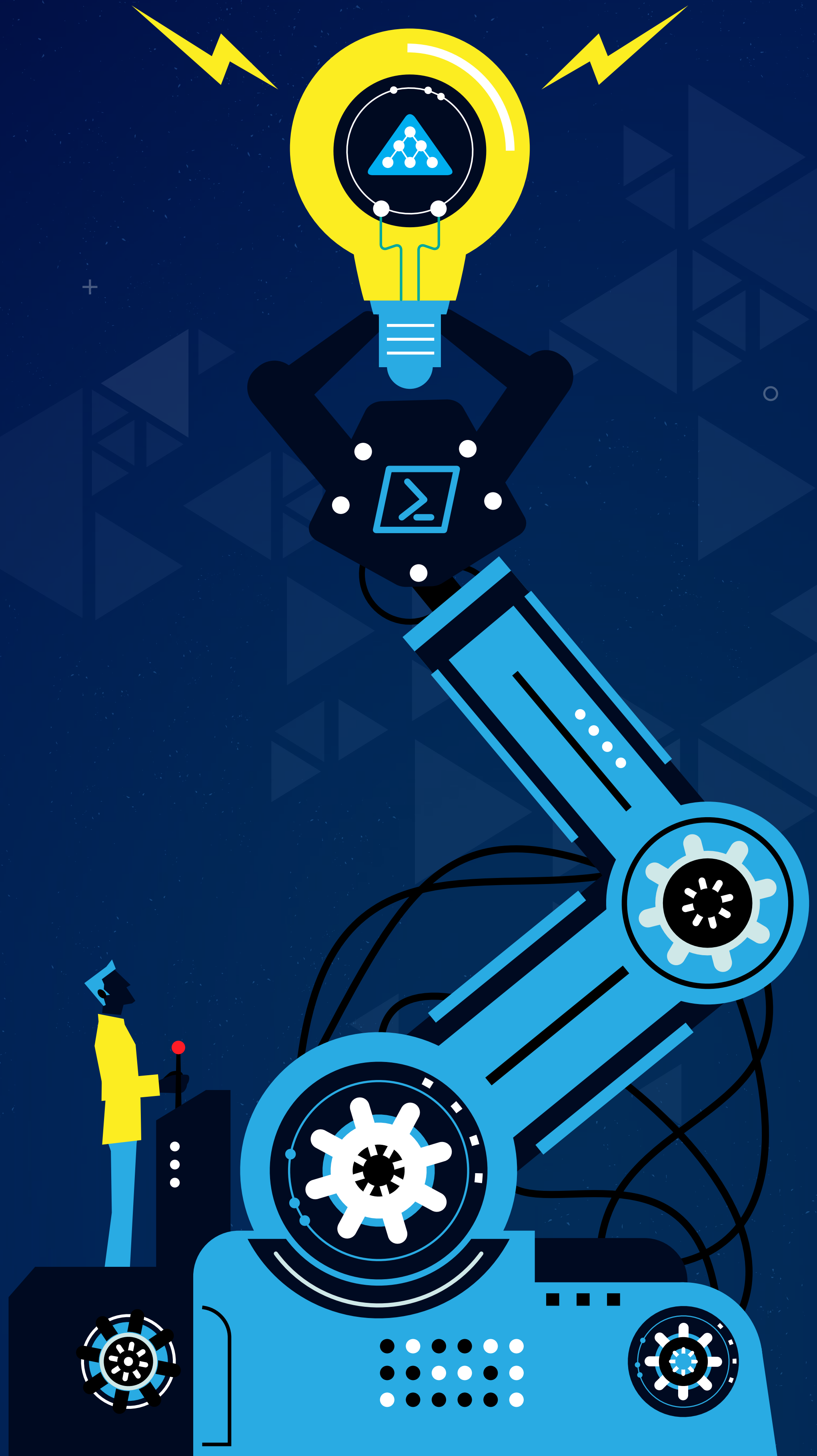
SysAdmin Magazine

netwrix

Automate it! Managing AD with PowerShell



#41

SysAdmin Magazine

№ 41

October '18

SysAdmin Magazine is a free source of knowledge for IT Pros who are eager to keep a tight grip on network security and do the job faster.

The Sysadmin Magazine team
sysadmin.magazine@netwrix.com

Contents

Jeff Melnick

IT Security Expert, Blogger

Step-by-step guide for creating new Active Directory users

The easiest way to create a new user in an Active Directory domain is using the Active Directory Users and Computers MMC snap-in. However, what if you need to create multiple user accounts in bulk, or ADUC is not available for some reason? In this article, we explain several ways to create Active Directory user accounts with PowerShell using the **New-ADUser** cmdlet.

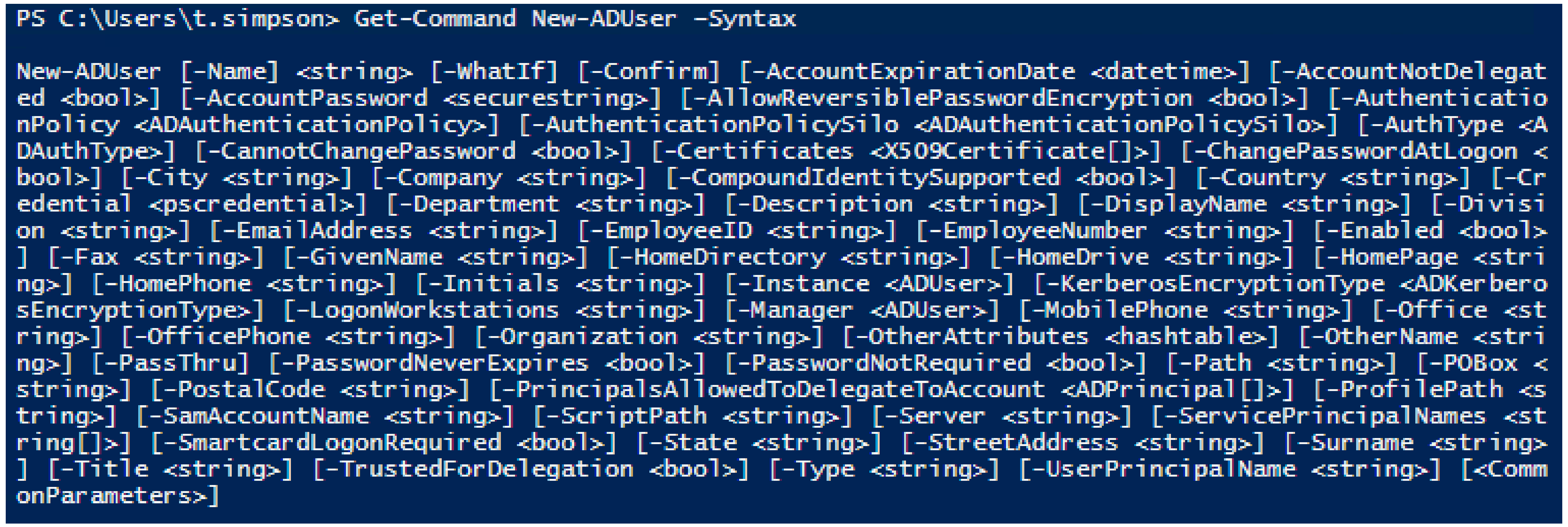
Create new user accounts using the new-ADuser cmdlet

So what is the PowerShell cmdlet used to create user objects? It's the New-ADUser cmdlet, which is included in the Active Directory PowerShell module built into Microsoft Windows Server 2008R2/2012 and above. Therefore, the first thing we need to do is enable the AD module:

```
Import-Module ActiveDirectory
```

Now let's take a closer look at cmdlet New-ADUser. We can get its full syntax by running the following command:

```
Get-Command New-ADUser -Syntax
```



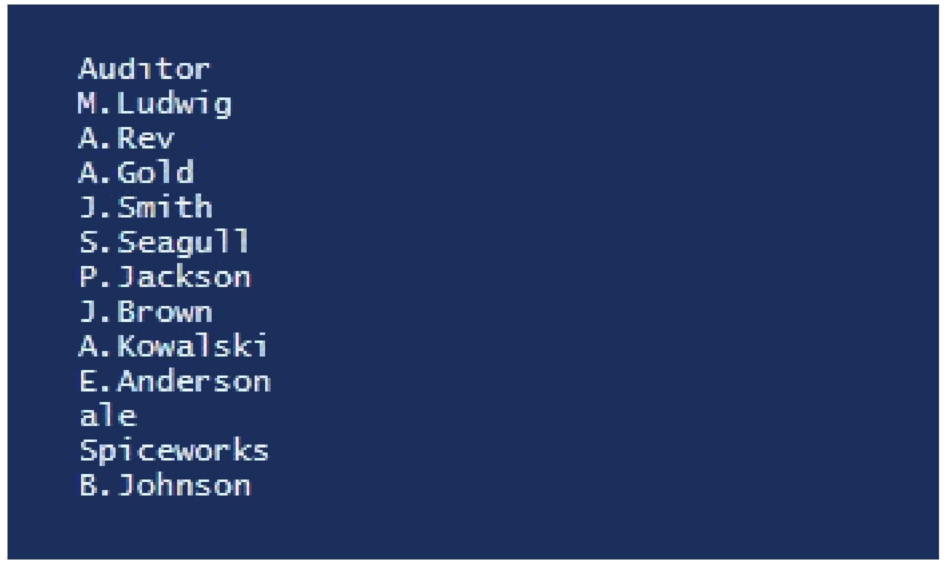
```
PS C:\Users\t.simpson> Get-Command New-ADUser -Syntax

New-ADUser [-Name] <string> [-WhatIf] [-Confirm] [-AccountExpirationDate <datetime>] [-AccountNotDelegat
ed <bool>] [-AccountPassword <securestring>] [-AllowReversiblePasswordEncryption <bool>] [-Authenticatio
nPolicy <ADAuthenticationPolicy>] [-AuthenticationPolicySilo <ADAuthenticationPolicySilo>] [-AuthType <A
DAuthType>] [-CannotChangePassword <bool>] [-Certificates <X509Certificate[]>] [-ChangePasswordAtLogon <
bool>] [-City <string>] [-Company <string>] [-CompoundIdentitySupported <bool>] [-Country <string>] [-Cr
edential <pscredential>] [-Department <string>] [-Description <string>] [-DisplayName <string>] [-Divisi
on <string>] [-EmailAddress <string>] [-EmployeeID <string>] [-EmployeeNumber <string>] [-Enabled <bool>
] [-Fax <string>] [-GivenName <string>] [-HomeDirectory <string>] [-HomeDrive <string>] [-HomePage <stri
ng>] [-HomePhone <string>] [-Initials <string>] [-Instance <ADUser>] [-KerberosEncryptionType <ADKerbero
sEncryptionType>] [-LogonWorkstations <string>] [-Manager <ADUser>] [-MobilePhone <string>] [-Office <st
ring>] [-OfficePhone <string>] [-Organization <string>] [-OtherAttributes <hashtable>] [-OtherName <stri
ng>] [-PassThru] [-PasswordNeverExpires <bool>] [-PasswordNotRequired <bool>] [-Path <string>] [-POBox <
string>] [-PostalCode <string>] [-PrincipalsAllowedToDelegateToAccount <ADPrincipal[]>] [-ProfilePath <s
tring>] [-SamAccountName <string>] [-ScriptPath <string>] [-Server <string>] [-ServicePrincipalNames <st
ring[]>] [-SmartcardLogonRequired <bool>] [-State <string>] [-StreetAddress <string>] [-Surname <string>
] [-Title <string>] [-TrustedForDelegation <bool>] [-Type <string>] [-UserPrincipalName <string>] [<Comm
onParameters>]
```

When you know the syntax, it's easy to add users to Active Directory:

```
New-ADUser B.Johnson
```

Now let's check whether the user was added successfully by listing all Active Directory users using the following script:

```
Get-ADUser -Filter * -Properties samAccount-
Name | select samAccountName
```



```
Auditor
M.Ludwig
A.Rev
A.Gold
J.Smith
S.Seagull
P.Jackson
J.Brown
A.Kowalski
E.Anderson
ale
Spiceworks
B.Johnson
```

There it is, the last one in the list!

Create a new Active Directory user account with password

Accounts are created with the following default properties:

- Account is created in the "Users" container.
- Account is disabled.
- Account is a member of Domain Users group.
- No password is set.
- User must reset the password at the first logon.

Therefore, to make a new account that's actually usable, we need to enable it using the Enable-ADAccount cmdlet and give it a password using the Set-ADAccountPassword cmdlet.

So let's create a new account with the following attributes:

- **Name** – Jack Robinson
- **Given Name** – Jack
- **Surname** – Robinson
- **Account Name** – J.Robinson
- **User Principal Name** – J.Robinson@enterprise.com
- **Path address** – "OU=Managers,DC=enterprise,DC=com"
- **Password Input**
- **Status** – Enabled

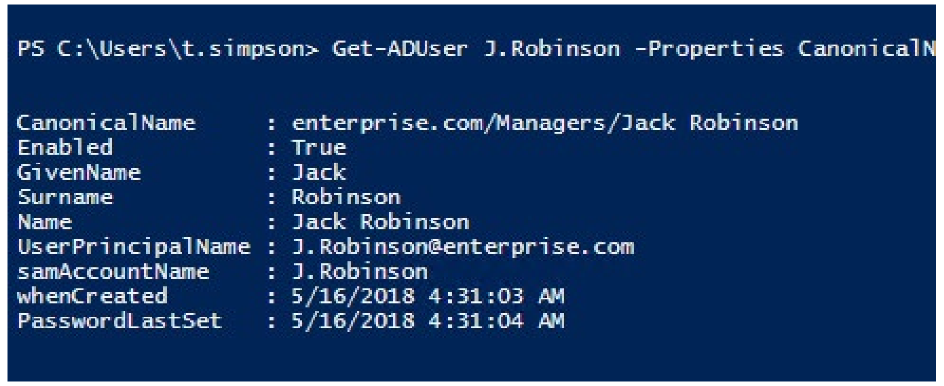
Here's the script we'll use:

```
New-ADUser -Name "Jack Robinson" -GivenNa-
me "Jack" -Surname "Robinson" -SamAccountName
"J.Robinson" -UserPrincipalName "J.Robinson@
enterprise.com" -Path "OU=Managers,DC=enter-
prise,DC=com" -AccountPassword(Read-Host -As-
SecureString "Input Password") -Enabled $true
```

The Read-Host parameter will ask you to input new password. Note that the password should meet the length, complexity and history requirements of your domain security policy.

Now let's take a look at the results by running the following cmdlet:

```
Get-ADUser J.Robinson -Properties CanonicalNa-
me, Enabled, GivenName, Surname, Name, UserPrin-
cipalName, samAccountName, whenCreated, Pass-
wordLastSet  | Select CanonicalName, Enabled,
GivenName, Surname, Name, UserPrincipalName,
samAccountName, whenCreated, PasswordLastSet
```

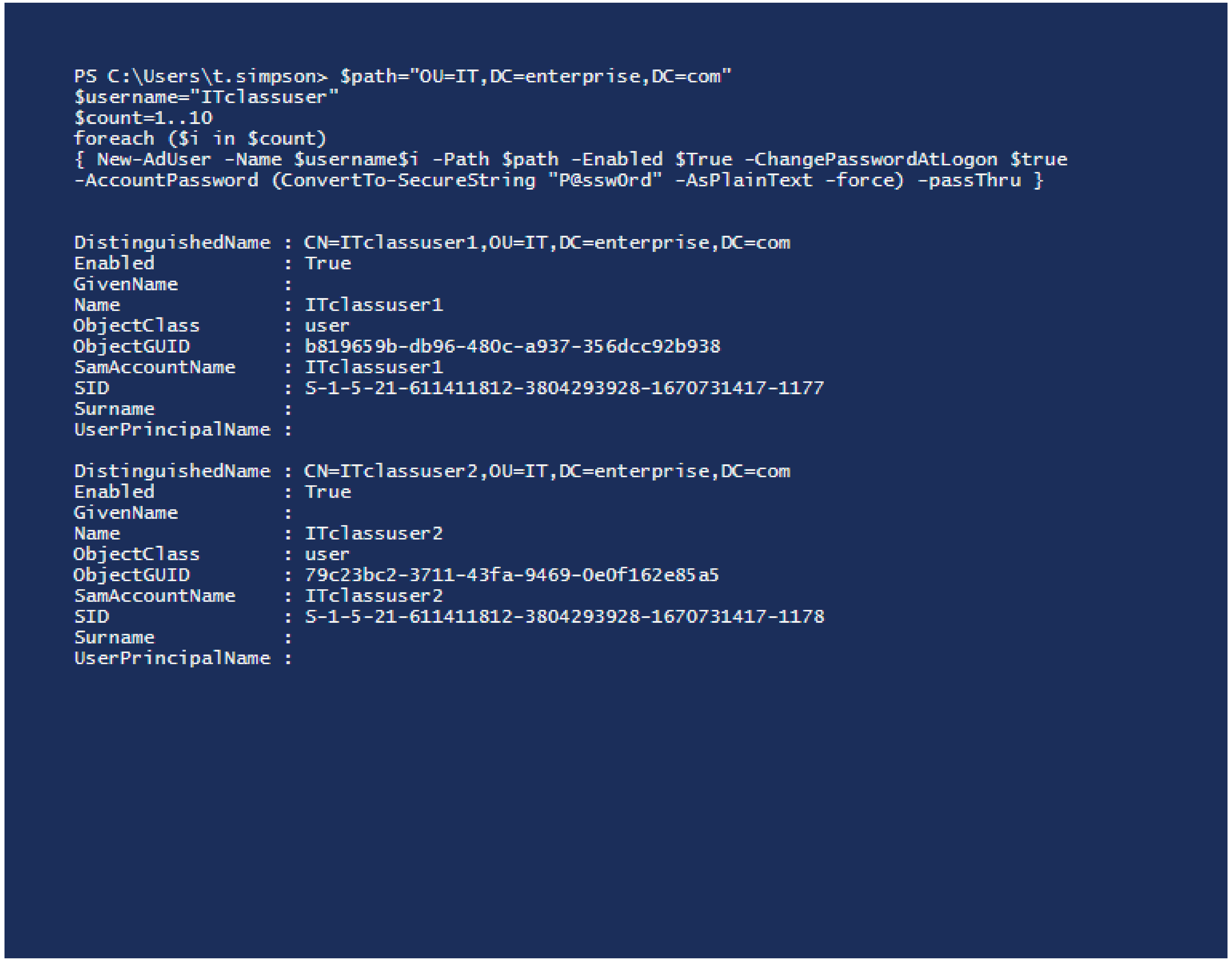


```
PS C:\Users\t.simpson> Get-ADUser J.Robinson -Properties CanonicalN

CanonicalName     : enterprise.com/Managers/Jack Robinson
Enabled           : True
GivenName         : Jack
Surname           : Robinson
Name              : Jack Robinson
UserPrincipalName : J.Robinson@enterprise.com
samAccountName    : J.Robinson
whenCreated       : 5/16/2018 4:31:03 AM
PasswordLastSet   : 5/16/2018 4:31:04 AM
```

Create AD users in bulk

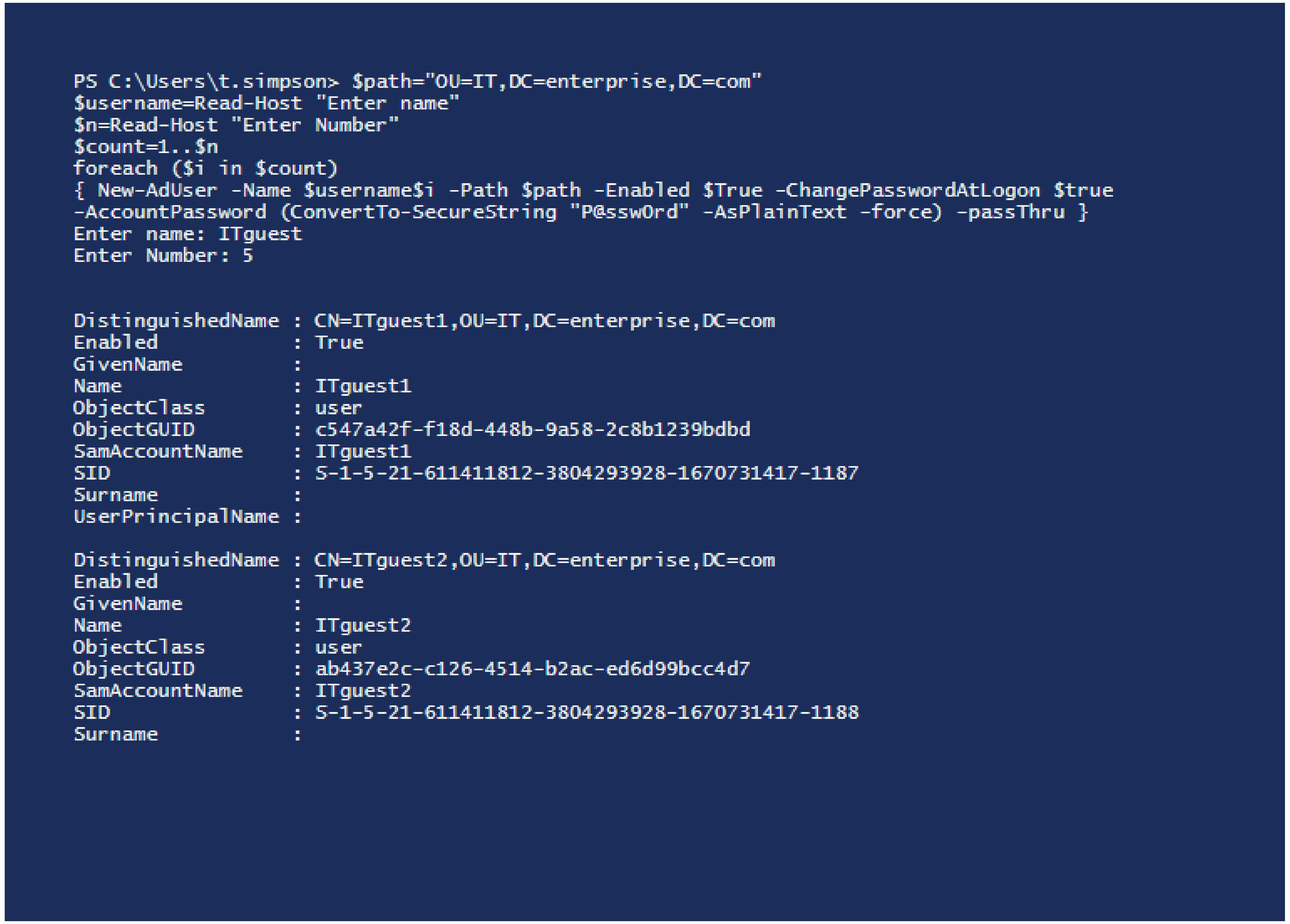
Now, let's make our task a little bit harder and create ten similar Active Directory accounts in bulk, for example, for our company's IT class, and set a default password (P@ssw0rd) for each of them. To send the default password in a protected state, we must use the **ConvertTo-SecureString** parameter. Here's the script to use:

```
$path="OU=IT,DC=enterprise,DC=com"
$username="ITclassuser"
$count=1..10
foreach ($i in $count)
{ New-AdUser -Name $username$i -Path $path -Ena-
bled $True -ChangePasswordAtLogon $true  `
-AccountPassword (ConvertTo-SecureString "P@
ssw0rd" -AsPlainText -force) -passThru }
```



```
PS C:\Users\t.simpson> $path="OU=IT,DC=enterprise,DC=com"
$username="ITclassuser"
$count=1..10
foreach ($i in $count)
{ New-AdUser -Name $username$i -Path $path -Enabled $True -ChangePasswordAtLogon $true
-AccountPassword (ConvertTo-SecureString "P@ssw0rd" -AsPlainText -force) -passThru }


DistinguishedName : CN=ITclassuser1,OU=IT,DC=enterprise,DC=com
Enabled           : True
GivenName         :
Name              : ITclassuser1
ObjectClass       : user
ObjectGUID        : b819659b-db96-480c-a937-356dcc92b938
SamAccountName    : ITclassuser1
SID               : S-1-5-21-611411812-3804293928-1670731417-1177
Surname           :
UserPrincipalName :

DistinguishedName : CN=ITclassuser2,OU=IT,DC=enterprise,DC=com
Enabled           : True
GivenName         :
Name              : ITclassuser2
ObjectClass       : user
ObjectGUID        : 79c23bc2-3711-43fa-9469-0e0f162e85a5
SamAccountName    : ITclassuser2
SID               : S-1-5-21-611411812-3804293928-1670731417-1178
Surname           :
UserPrincipalName :
```

Now let's make our script more flexible by adding the Read-Host parameter, which will ask for the name and number of users:



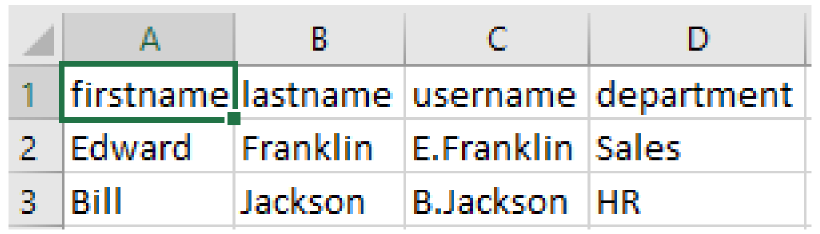
```
PS C:\Users\t.simpson> $path="OU=IT,DC=enterprise,DC=com"
$username=Read-Host "Enter name"
$n=Read-Host "Enter Number"
$count=1..$n
foreach ($i in $count)
{ New-AdUser -Name $username$i -Path $path -Enabled $True -ChangePasswordAtLogon $true
-AccountPassword (ConvertTo-SecureString "P@ssw0rd" -AsPlainText -force) -passThru }
Enter name: ITguest
Enter Number: 5


DistinguishedName : CN=ITguest1,OU=IT,DC=enterprise,DC=com
Enabled           : True
GivenName         :
Name              : ITguest1
ObjectClass       : user
ObjectGUID        : c547a42f-f18d-448b-9a58-2c8b1239bdbd
SamAccountName    : ITguest1
SID               : S-1-5-21-611411812-3804293928-1670731417-1187
Surname           :
UserPrincipalName :

DistinguishedName : CN=ITguest2,OU=IT,DC=enterprise,DC=com
Enabled           : True
GivenName         :
Name              : ITguest2
ObjectClass       : user
ObjectGUID        : ab437e2c-c126-4514-b2ac-ed6d99bcc4d7
SamAccountName    : ITguest2
SID               : S-1-5-21-611411812-3804293928-1670731417-1188
Surname           :
```

Import AD users from a CSV file

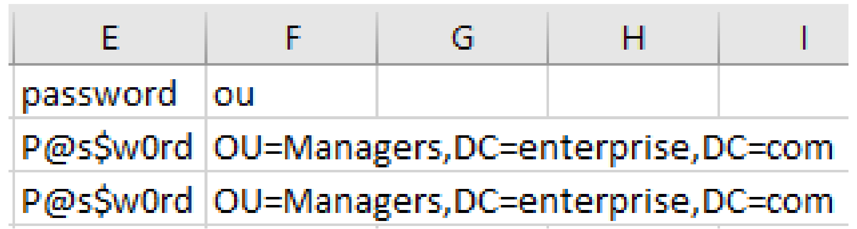
Another option for creating users in AD is to import them from a CSV file. This option is great when you have a list of users with predefined personal details such as:

- FirstName
- LastName
- Username
- Department
- Password
- OU

The CSV file must be in UTF8 encoding and contain contact data that looks like this:



	A	B	C	D
1	firstname	lastname	username	department
2	Edward	Franklin	E.Franklin	Sales
3	Bill	Jackson	B.Jackson	HR



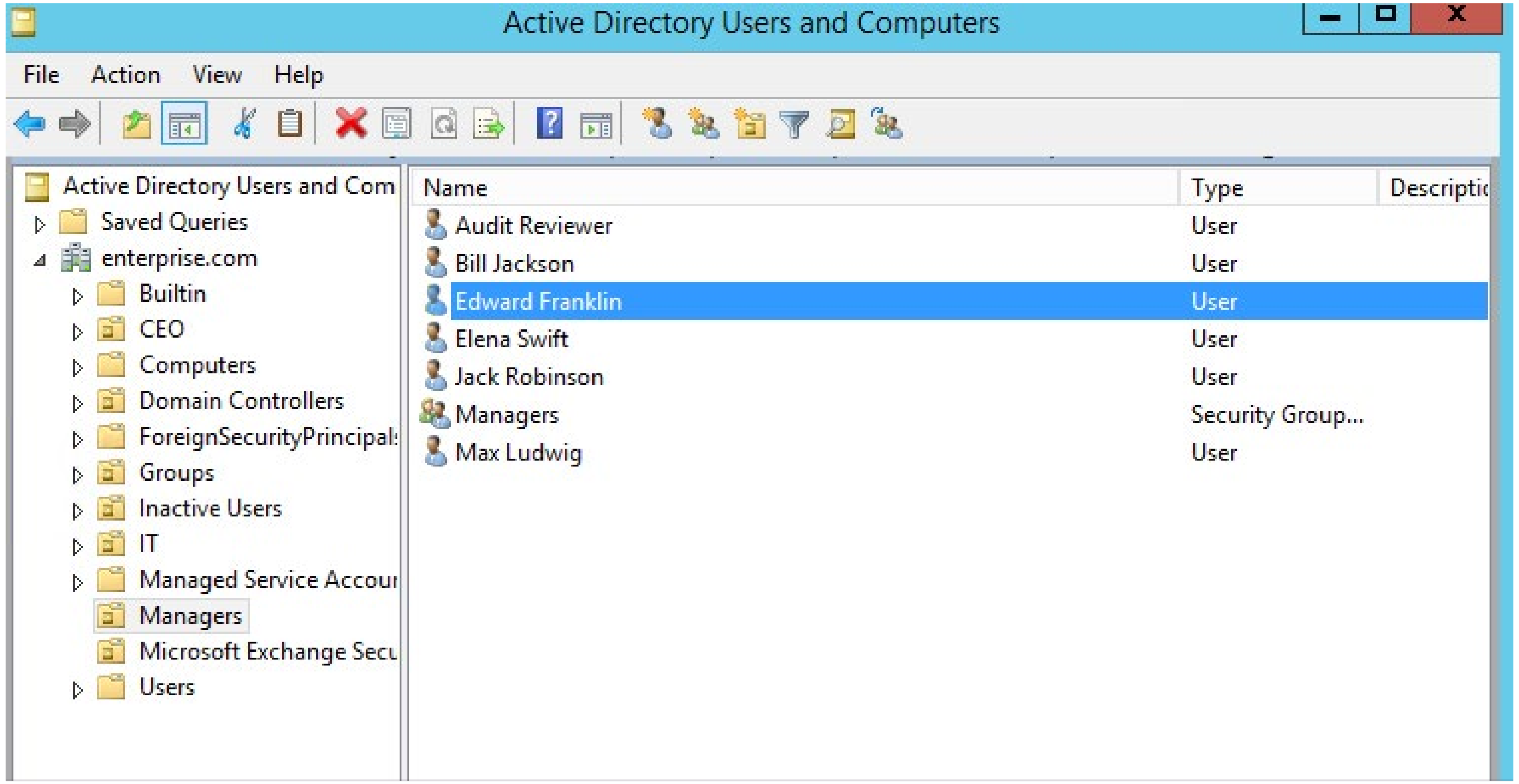
E	F	G	H	I
password	ou			
P@s$w0rd	OU=Managers,DC=enterprise,DC=com			
P@s$w0rd	OU=Managers,DC=enterprise,DC=com			

The following script will create enabled user objects for any users in the CSV that don’t already have accounts in AD. The “Reset password at the next logon” option will be enabled for the new accounts, so you can use your default password:

```
#Enter a path to your import CSV file
$ADUsers = Import-csv C:\scripts\newusers.
csv

foreach ($User in $ADUsers)
{

       $Username    = $User.username
       $Password    = $User.password
       $Firstname   = $User.firstname
       $Lastname    = $User.lastname
    $Department = $User.department
       $OU           = $User.ou

       #Check if the user account already
exists in AD
       if (Get-ADUser -F {SamAccountName -eq
$Username})
       {
               #If user does exist, output a
```

```
warning message
               Write-Warning “A user account $Username has already exist in Active Directory.”
       }
       else
       {
              #If a user does not exist then create a new user account

        #Account will be created in the OU listed in the $OU variable in the CSV file; don’t forget to chan-
ge the domain name in the”-UserPrincipalName” variable
              New-ADUser `
            -SamAccountName $Username `
            -UserPrincipalName “$Username@yourdomain.com” `
            -Name “$Firstname $Lastname” `
            -GivenName $Firstname `
            -Surname $Lastname `
            -Enabled $True `
            -ChangePasswordAtLogon $True `
            -DisplayName “$Lastname, $Firstname” `
            -Department $Department `
            -Path $OU `
            -AccountPassword (convertto-securestring $Password -AsPlainText -Force)

       }
}
```

After script execution, we have two new users, Edward Franklin and Bill Jackson, in our Active Directory domain:



Name	Type	Descriptio
Audit Reviewer	User	
Bill Jackson	User	
Edward Franklin	User	
Elena Swift	User	
Jack Robinson	User	
Managers	Security Group...	
Max Ludwig	User	

Let's take a look at their details by running Get-ADUser cmdlet again:

```
Get-ADUser E.Franklin -Properties Canoni-
calName, Enabled, GivenName, Surname, Name,
```

```
UserPrincipalName, samAccountName, whenCrea-
ted, PasswordLastSet  | Select CanonicalName,
Enabled, GivenName, Surname, Name, UserPrin-
cipalName, samAccountName, whenCreated, Pass-
wordLastSet
```

Now you know how to create users in Active Directory using PowerShell scripts. Try performing some account creations, bulk account creations and CSV imports yourself on local or remote systems. Remember, the ADUC MMC snap-in is great for creating a few users with extended attributes, but PowerShell is much better for importing a large number of user accounts in bulk.

Windows PowerShell Scripting Tutorial

Free Download

Russell Smith

Security Expert, IT consultant

How to change Active Directory account status

In this article, I'll show you how to use PowerShell to lock, unlock, enable and disable AD user and computer accounts individually and in bulk using comma-delimited files.

Before you can run the Active Directory PowerShell cmdlets, you have to have the Active Directory module for PowerShell installed on your computer. If you are using Windows 10, download the Remote Server Administration Tools (RSAT) for Windows 10 from Microsoft's website here and then install it. Then enable the AD PowerShell module feature by opening a PowerShell prompt with local administrator privileges and running the **Enable-WindowsOptionalFeature** cmdlet as shown here:

```
Enable-WindowsOptionalFeature -Online -Fe-
atureName RSATClient-Roles-AD-Powershell
```

Optionally, you can also update the help files using the **Update-Help** cmdlet:

```
Update-Help -Module ActiveDirectory -Ver-
bose -Force
```

Be sure to close the PowerShell prompt, since you won't need the elevated privileges for anything else. The instructions below can be run in the security context of any user that has permissions to perform user account operations in Active Directory, like unlocking user accounts and enabling and disabling user, computer and service accounts.

How to find locked Active Directory accounts

You can't lock Active Directory accounts using PowerShell or the GUI; indeed, there is no reason you should want to do that. But you can search for locked out user accounts with the help of the **Search-ADAccount** cmdlet. Here I pipe the results of the Search-ADAccount cmdlet to the Select-Object cmdlet to display just the Name and SamAccountName attributes of each locked account:

```
Search-ADAccount -LockedOut -UsersOnly |
Select-Object Name, SamAccountName
```

How to unlock Active Directory accounts

You can easily unlock user accounts using the **Unlock-ADAccount** cmdlet. Use the **-Identity** parameter to specify which account to unlock; you can supply its distinguished name, security identifier (SID), globally unique identifier (GUID) or Security Account Manager (SAM) account name. Here I'm unlocking the account RussellS:

```
Unlock-ADAccount -Identity RussellS
```

How to enable Active Directory accounts

If an account object has been disabled for whatever reason, you can enable it using the **Enable-ADAccount** cmdlet:

```
Enable-ADAccount -Identity RussellS
```

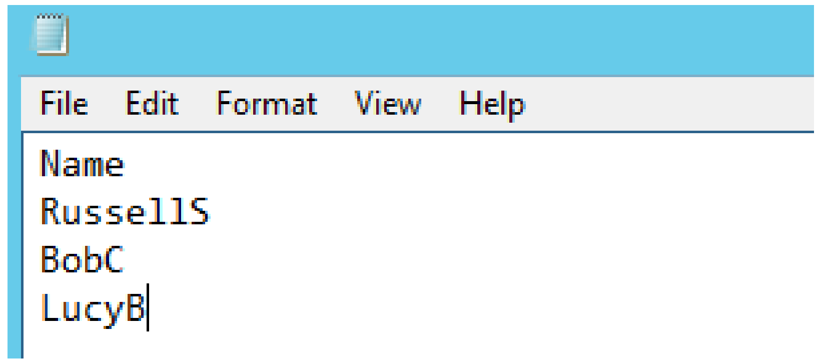
How to disable Active Directory accounts

Similarly, the **Disable-ADAccount** cmdlet is used to disable AD accounts:

```
Disable-ADAccount -Identity RussellS
```

Disabling users from a CSV file

You can also disable all Active Directory user accounts listed in a comma-delimited (.csv) text file. The file must contain a header and then a list of user names, one in each row. My CSV file has only one column (with the header "Name"), so my comma-delimited file has no commas! If your CSV file has more than one column, those additional columns will simply be ignored by the script.



```
Name
RussellS
BobC
LucyB
```

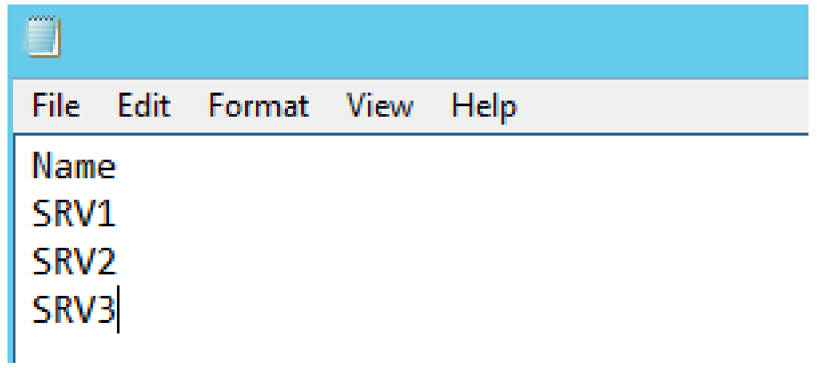
I start by importing the CSV file's contents as an object ($users), and then I use a ForEach loop to disable the user on each line of the text file. Here's the PowerShell script:

```
$users=Import-CSV c:\temp\users.csv
ForEach ($user in $users)
{
     Disable-ADAccount -Identity $($user.
name)
}
```

Disabling computer accounts from a CSV file

The PowerShell script for disabling computer accounts listed in a CSV file is almost identical. The main difference is that I have to add a dollar sign ($) to the end of the -Identity parameter value to designate that I want to disable a computer object and not a user account object. I also change the variable and file names to more appropriate for computer accounts.

The CSV file looks like this:



```
Name
SRV1
SRV2
SRV3
```

And here is the script:

```
$computers=Import-CSV c:\temp\computers.csv
ForEach ($computer in $computers)
{
     Disable-ADAccount -Identity “$($compu-
ter.name)$”
}
```

To check the results, use the **Search-ADAccount** cmdlet:

```
Search-ADAccount -AccountDisabled -Com-
putersOnly | Select-Object Name, SamAc-
countName
```

Disabling inactive users

The **Search-ADAccount** and **Disable-ADAccount** cmdlets can be used together to disable inactive user accounts. I'll give two examples. First, I'll create a new timespan object ($timespan) and set it to ninety days, and then I'll use it as the value of the **-TimeSpan** parameter to disable accounts that have not been active for the past three months. The **-AccountInactive** parameter requires that the domain functional level be Windows Server 2003 or higher.

```
$timespan = New-Timespan -Days 90
Search-ADAccount -UsersOnly -AccountInac-
tive -TimeSpan $timespan | Disable-ADAc-
count
```

Another option is to use the **-DateTime** parameter to return accounts that have been inactive since a given date. This script disables all accounts not active since June 3, 2018:

```
Search-ADAccount -UsersOnly -AccountInacti-
ve -DateTime ‘6/3/2018’ | Disable-ADAccount
```

It's worth noting that because of the way Active Directory synchronizes the **LastLogOnDate** attribute, results returned when specifying the **–AccountInactive** parameter with the **Search-ADAccount** cmdlet can be inaccurate by as much as 9–14 days.

As you can see, managing the status of Active Directory accounts with PowerShell is straightforward. Because PowerShell is object-oriented, it is easy to create objects that contain the data you want to process and then pass them on to other cmdlets that perform the required actions.

[Free Tool]

Quickly troubleshoot account lockouts in Active Directory

Free Download

Jeff Melnick

IT Security Expert, Blogger

Best practices: How to manage computer accounts in Active Directory

Before a user can log into a computer and access network and domain-based resources, that computer must be a member of the Active Directory environment.

PowerShell ISE is the best tool for working with PowerShell scripts. Start the PowerShell ISE tool with administrator privileges by pressing “Windows+R” and entering “runas /profile /user:Administrator PowerShell_ ISE” in the Run window. (Alternatively, you can right-click on the PowerShell ISE icon and choose the “Run as administrator” option.) Type in the administrator’s password when prompted.

Before you can work with AD and its objects, you need to import the Active Directory module for Windows PowerShell. In Microsoft Windows Server 2008 R2, you need to enable this module by running the following command:

```
Import-Module ActiveDirectory
```

In Microsoft Windows Server 2012 and later, this module is enabled by default.

Join a computer to a domain

The most common task is joining a computer to a domain controller. To join a PC to an Active Directory domain, run the following PowerShell script locally:

```
$dc = “ENTERPRISE” # Specify the domain to
join.
$pw = “Password123” | ConvertTo-SecureString
-asPlainText -Force # Specify the password
for the domain admin.
$usr = “$dc\T.Simpson” # Specify the domain
admin account.
$creds = New-Object System.Management.Automa-
tion.PSCredential($usr,$pw)
Add-Computer -DomainName $dc -Credential
$creds -restart -force -verbose # Note that
the computer will be restarted automatically.
```

The computer will restart and then join the domain; it will be added to the default container.

To join a computer to a DC remotely, you need to enhance this script this way:

```
$dc = "ENTERPRISE"
$pw = "Password123" | ConvertTo-SecureString
-asPlainText -Force
$usr = "$dc\T.Simpson"
$pc = "R07GF" # Specify the computer that
should be joined to the domain.
$creds = New-Object System.Management.Automa-
tion.PSCredential($usr,$pw)
Add-Computer -ComputerName $pc -LocalCreden-
tial $pc\admin -DomainName $dc -Credential
$creds -Verbose -Restart -Force
```

The **$pc** variable and **–LocalCredential** parameter are used to authenticate the computer to the domain. Note that in order to use this method, you must disable the firewall on the local computer.

Join multiple computers to a domain

You can add more than one computer to the domain by either specifying them in the command line as a comma-delimited list or importing their names from a text file.

Here's how to specify the computers in a comma-delimited list:

```
$dc = "ENTERPRISE"
$pw = "Password123" | ConvertTo-SecureString
-asPlainText -Force
$usr = "$dc\T.Simpson"
$pc = "WKS034, WKS052, WKS057" # Specify the
computers that should be joined to the do-
main.
$creds = New-Object System.Management.Automa-
tion.PSCredential($usr$pw)
Add-Computer -ComputerName $pc -LocalCreden-
tial $pc\admin -DomainName $dc -Credential
$creds -Restart -Force
```

And here's how to use a text file with the list of computers that should be joined:

```
$dc = "ENTERPRISE"
$pw = "Password123" | ConvertTo-SecureString
-asPlainText -Force
$usr = "$dc\T.Simpson"
$pc = Get-Content -Path C:\Computers.txt #
Specify the path to the computers list.
$creds = New-Object System.Management.Automa-
tion.PSCredential($usr,$pw)
Add-Computer -ComputerName $pc -LocalCreden-
tial $pc\admin -DomainName $dc -Credential
$creds -Restart -Force
```

Create a computer object in AD

To create a computer object, use the New-ADComputer cmdlet. For example, execute the following cmdlet parameters to create a computer object with "WKS932" as its name and the default LDAP path value:

```
New-ADComputer –Name "WKS932" –SamAccountName
"WKS932"
```

Create computer accounts from a CSV file

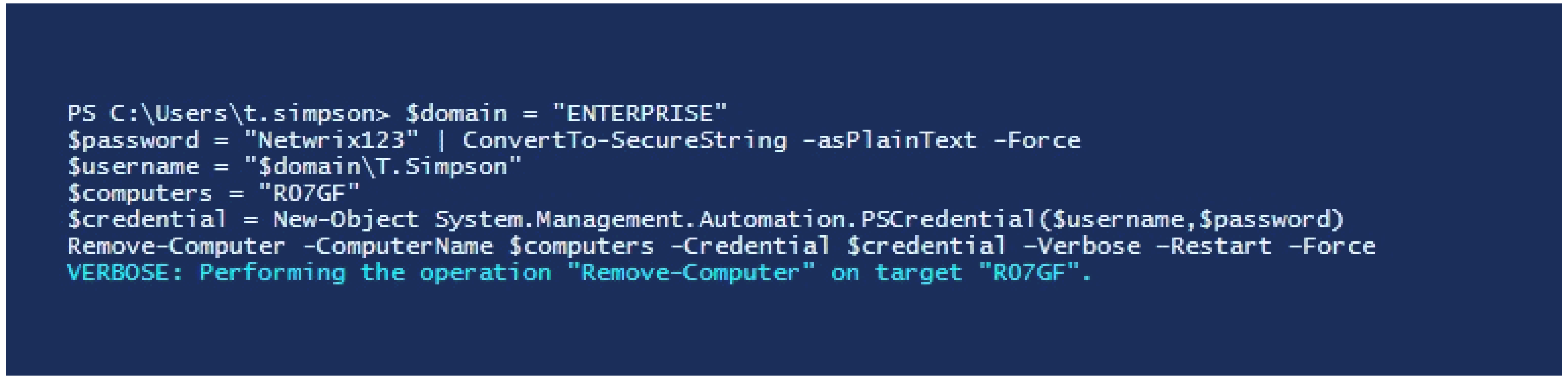
If you have a list of computers that should be imported into Active Directory, save the list to a CSV file with the heading "computer" and the list of computer names in the column below it. Run the following PowerShell script on your domain controller to add computers from the CSV file, making sure you have the "Path" and "File" variables set correctly:

```
$File="C:\scripts\Computers.csv" # Specify
the import CSV position.
$Path="OU=Devices,DC=enterprise,DC=com" #
Specify the path to the OU.
Import-Csv -Path $File | ForEach-Object {
New-ADComputer -Name $_.Computer -Path $Path
-Enabled $True}
```

Remove a computer from a domain

The most common task is joining a computer to a domain controller. To join a PC to an Active Directory domain, run the following PowerShell script locally:

```
$dc = "ENTERPRISE"
$pw = "Password123" | ConvertTo-SecureString -asPlainText -Force
$usr = "$dc\T.Simpson"
$pc = "R07GF"
$creds = New-Object System.Management.Automation.PSCredential($usr,$pw)
Remove-Computer -ComputerName $pc -Credential $creds -Verbose -Restart -Force
```



```
PS C:\Users\t.simpson> $domain = "ENTERPRISE"
$password = "Netwrix123" | ConvertTo-SecureString -asPlainText -Force
$username = "$domain\T.Simpson"
$computers = "R07GF"
$credential = New-Object System.Management.Automation.PSCredential($username,$password)
Remove-Computer -ComputerName $computers -Credential $credential -Verbose -Restart -Force
VERBOSE: Performing the operation "Remove-Computer" on target "R07GF".
```

To remove multiple computers using a list in a TXT file, use the script above for joining computers to a DC, replacing the Add-Computer cmdlet with Remove-Computer. Note that you will still need domain admin credentials to complete this unjoin operation.

Disable an AD computer account

Use the **Disable-ADAccount** cmdlet to disable Active Directory user, computer and service accounts. If you specify a computer account name, remember to append a dollar sign ($) at the end of the name; otherwise, you'll get an error after script execution.

```
Disable-ADAccount -Identity fs1$
```

Disable computer accounts using a list

You can also disable computer accounts in bulk using a list in a text file:

```
$Pclist = Get-Content C:\scripts\Computer.txt
# Specify the path to the computers list.
Foreach($pc in $Pclist)
{
Disable-ADAccount -Identity "$pc"
Get-ADComputer -Identity "$pc" | Move-ADOb-
ject -TargetPath "OU=Disabled Computers,D-
C=enterprise,DC=com"
}
```

Delete a computer from AD

To delete a computer account from AD, use the Remove-ADObject cmdlet. The -Identity parameter specifies which Active Directory computer to remove. You can specify a computer by its distinguished name, GUID, security identifier (SID) or Security Accounts Manager (SAM) account name.

```
Remove-ADObject -Identity "WKS932"
```

You will be prompted to confirm the deletion.

Delete computer accounts using a list

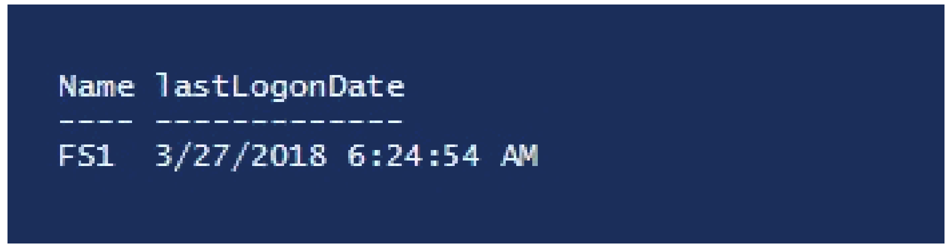
If you have a text file with a list of old computers, you can streamline the task of removing them using PowerShell. The following script will read the computer names from a TXT file and delete the corresponding accounts via a chain of commands, or pipeline:

```
Get-Content C:\scripts\computersfordeletion.
txt | % { Get-ADComputer -Filter { Name -eq
$_ } } | Remove-ADObject -Recursive
```

Remove stale computer accounts from Active Directory

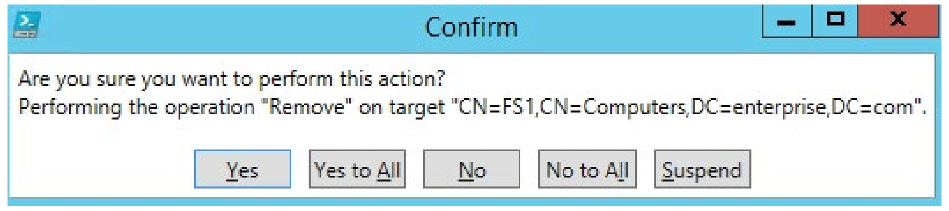
Stale accounts in Active Directory can be compromised, leading to security incidents, so it is critical to keep an eye on them. This PowerShell script will query Active Directory and return all computers that have not been logged in to for the past 30 days; you can easily change this default value in the script. It also will remove those accounts to keep your AD clean.

```
$stale = (Get-Date).AddDays(-30) # means 30
days since last logon, can be changed to any
number.

Get-ADComputer -Property Name,lastLogonDate
-Filter {lastLogonDate -lt $stale} | FT Na-
me,lastLogonDate

Get-ADComputer -Property Name,lastLogonDate
-Filter {lastLogonDate -lt $stale} | Remo-
ve-ADComputer
```



```
Name lastLogonDate
---- -------------
FS1  3/27/2018 6:24:54 AM
```

There is one computer, FS1, that has been not been logged on to for more than 30 days. The system will prompt for confirmation before deleting it from the domain:



Are you sure you want to perform this action?
Performing the operation "Remove" on target "CN=FS1,CN=Computers,DC=enterprise,DC=com".

If you want to disable, rather than delete, the inactive computer accounts, replace the Remove-ADComputer cmdlet with Set-ADComputer and -Enabled $false parameter and value.

Create a computer object in Active Directory

To create a computer object, use the New-ADComputer cmdlet. For example, execute the following cmdlet parameters to create a computer object with "WKS932" as its name and the default LDAP path value:

```
New-ADComputer -Name "WKS932" -SamAccount-
Name "WKS932"
```

Create computer accounts from a CSV file

If you have a list of computers that should be imported into Active Directory, save the list to a CSV file with the heading “computer” and the list of computer names in the column below it. Run the following PowerShell script on your domain controller to add computers from the CSV file, making sure you have the “Path” and “File” variables set correctly:

```
$File=”C:\scripts\Computers.csv” # Specify
the import CSV position.
```

Rename a computer

To change a computer name, use the Rename-Computer cmdlet. Note that the computer must be online and connected to Active Directory.

```
Rename-Computer –ComputerName “FS1” -NewName
“FS2”
```

If you want to run this script locally, it will look like this:

```
Rename-Computer -NewName “newname” -Domain-
Credential “Domain\Administrator”
```

Rename a computer and join It to a domain

You can improve the renaming script by joining the computer to the domain and putting it into the specified OU simultaneously. The script should be run on the target machine, not on the domain controller.

```
$DC = “contoso.com” # Specify the domain to
join.

$Path = “OU=TestOU,DC=contoso,DC=com” # Specify
the path to the OU where to put the computer
account in the domain.
$DC = “contoso.com” # Specify the domain to join.

$Path = “OU=TestOU,DC=contoso,DC=com” # Specify
the path to the OU where to put the computer
account in the domain.
```

```
Add-Computer -DomainName $DC -OUPath $Path
-NewName $NewComputerName –Restart –Force
```

The script will prompt for the credentials of an account that has permissions to join computers to the domain, and then the computer will be renamed, restarted and joined to the domain.

Reset an AD computer

Like a user account, a computer account interacts with Active Directory using a password. But for computer accounts, a password change is initiated every 30 days by default and the password is exempted from the domain’s password policy. Password changes are driven by the client (computer), not AD.

Computer credentials usually unknown to the user because they are randomly set by the computer. But you can set your own password; here is a PowerShell script for doing so:

```
$pc = read-host –Prompt “Input computer name
to reset“ # Specify the computer name.
$pw = read-host –Prompt “Input random cha-
racters for temp password“ –AsSecureString #
Specify the password.
Get-ADComputer $pc | Set-ADAccountPassword –
NewPassword:$pw -Reset:$true
```

Adam Stetson

Systems Engineer, Security Expert

Top tips for managing OUs and moving their objects

An organizational unit (OU) is a container in Active Directory where users, groups and computers, as well as other OUs, can be stored. Each AD domain can have its own organizational unit hierarchy.

In this article, you will learn about OU management and how to use PowerShell scripts to.

PowerShell ISE is the best tool for working with PowerShell scripts. Start the PowerShell ISE tool with administrator privileges by pressing “Windows+R” and entering “runas /profile /user:Administrator PowerShell_ISE” in the Run window. Type in the administrator’s password when prompted. Alternatively, you can right-click the PowerShell ISE icon and choose the “Run as administrator” option.

To work with AD and its objects, you need to import the Active Directory module for Windows PowerShell. In Microsoft Windows Server 2008 R2, you need to enable this module by running the following command:

```
Import-Module ActiveDirectory
```

In Microsoft Windows Server 2012 and later, this module is enabled by default.

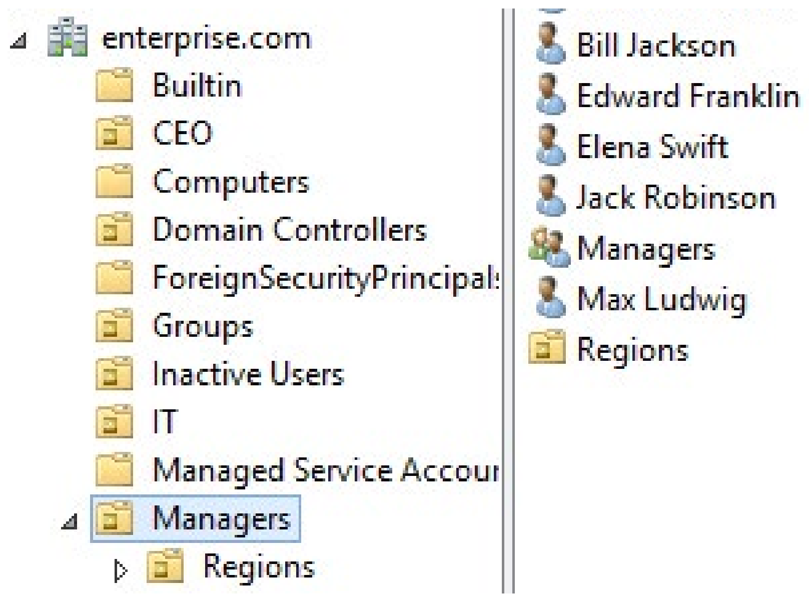
Create OUs in an Active Directory domain

You can create a new organizational unit in Active Directory by using the New-ADOrganizationalUnit cmdlet and specifying the name of a new OU object. By default, PowerShell will create the OU in the domain root. The command below will create an OU named “Regions” on the DC:

```
New-ADOrganizationalUnit “Regions”
```

If you need a different OU LDAP path, specify its distinguished name using the –Path cmdlet parameter:

```
NNew-ADOrganizationalUnit “Regions” –Path
“OU=Managers,DC=Enterprise,DC=com”
```



Move an OU to another LDAP address

If you need to move an OU to another location, use Move-ADObject cmdlet. Note that the target OU must not be protected from accidental deletion. If it is, use this command to remove that protection:

```
Set-ADOrganizationalUnit -Identity "OU=-
Regions,OU=Managers,DC=Enterprise,DC=Com"
-ProtectedFromAccidentalDeletion $False
```

Now you can move the OU to another location:

```
Move-ADObject -Identity "OU=Regions,OU=Ma-
nagers,DC=Enterprise,DC=Com" -TargetPath
"OU=IT,DC=Enterprise,DC=Com"
```

Rename an OU

To rename an organizational unit, use the Rename-ADObject cmdlet. The -Identity parameter specifies the Active Directory object to rename and requires either its distinguished name (DN) or GUID.

This command renames the "Regions" OU to "Districts":

```
Rename-ADObject -Identity "OU=Regi-
ons,OU=IT,DC=enterprise,DC=COM" -NewName
Districts
```

Alternatively, you can use the Get-ADOrganizationalUnit cmdlet with the -Filter parameter; it does not require the whole LDAP path to the OU. However, that cmdlet will search the whole AD and the action will be applied to all OUs that contain the search term in their names:

```
Get-ADOrganizationalUnit -Filter "Name -eq
'Regions'" | Rename-ADObject -NewName Coun-
tries
```

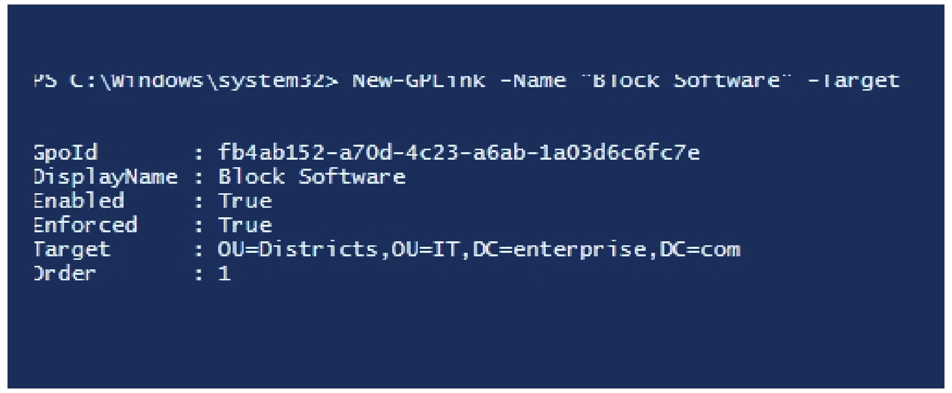
Apply a group policy to an OU

To assign a Group Policy to an OU, use the New-GPLink cmdlet, which basically makes a link between the specified Group Policy object (GPO) and the OU. You can specify any of the following properties for the link:

- **Enabled** — If the link is enabled, the settings of the GPO are applied when Group Policy is processed for the site, domain or OU.
- **Enforced** — If the link is enforced, it cannot be blocked at a lower-level container.
- **Order** — The order specifies the precedence of the GPO settings.

The following command links the “Block Software” GPO to the “Districts” OU with the link both enabled and enforced:

```
New-GPLink -Name “Block Software” -Target
“OU=Districts,OU=IT,dc=enterprise,dc=com”
-LinkEnabled Yes -Enforced Yes
```



```
PS C:\Windows\system32> New-GPLink -Name "Block Software" -Target

GpoId       : fb4ab152-a70d-4c23-a6ab-1a03d6c6fc7e
DisplayName : Block Software
Enabled     : True
Enforced    : True
Target      : OU=Districts,OU=IT,DC=enterprise,DC=com
Order       : 1
```

Move computers and users to a new OU

Once you’ve created an OU and optionally linked it to a GPO, it’s time to fill it up with users and computers. The PowerShell Move-ADObject cmdlet moves any object or set of objects (such as a user, a computer, a group or another OU) to a different OU. The -Identity parameter specifies which Active Directory object or container to move. Note that you need to enter the full LDAP path or SID of the object; you cannot use its SamAccountName. The below example demonstrates how to move a user (John Brown) to the “Districts” OU:

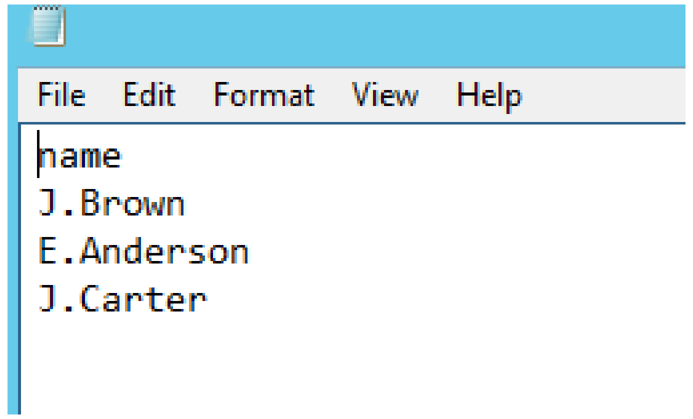
```
Move-ADObject -Identity “CN=John Brown,C-
N=Users,DC=enterprise,DC=com” -TargetPath
“OU=Districts,OU=IT,DC=Enterprise,DC=Com”
```

Use the same syntax to move computer objects. The following command will move computer “R07GF” to the “Computers” container:

```
Move-ADObject -Identity “CN=R07GF,OU=CEO,D-
C=enterprise,DC=com” -TargetPath “CN=Compu-
ters,DC=Enterprise,DC=Com
```

Move AD computers and users to another OU using a CSV or TXT file

If you have a predefined list of objects to move, you can save it as a CSV file and then import that file to Active Directory. The CSV list should be in the following format:



```
name
J.Brown
E.Anderson
J.Carter
```

Use this PowerShell script for moving AD user accounts listed in a CSV file:

```
# Specify target OU. This is where users
will be moved.
$TargetOU =  “OU=Districts,OU=IT,DC=enter-
prise,DC=com”
# Specify CSV path. Import CSV file and as-
sign it to a variable.
$Imported_csv = Import-Csv -Path “C:\temp\
MoveList.csv”

$Imported_csv | ForEach-Object {
     # Retrieve DN of user.
$UserDN  = (Get-ADUser -Identity $_.Name).
distinguishedName
     # Move user to target OU.
     Move-ADObject  -Identity $UserDN  -Tar-
getPath $TargetOU

 }
```

To move AD computer accounts listed in a text file, use the following PowerShell script:

```
# Specify path to the text file with the com-
puter account names.
$computers = Get-Content C:\Temp\Computers.
txt

# Specify the path to the OU where computers
will be moved.
$TargetOU   =  "OU=Districts,OU=IT,DC=enter-
prise,DC=com"
ForEach( $computer in $computers){
    Get-ADComputer $computer |
    Move-ADObject -TargetPath $TargetOU

}
```

Remove an OU from AD

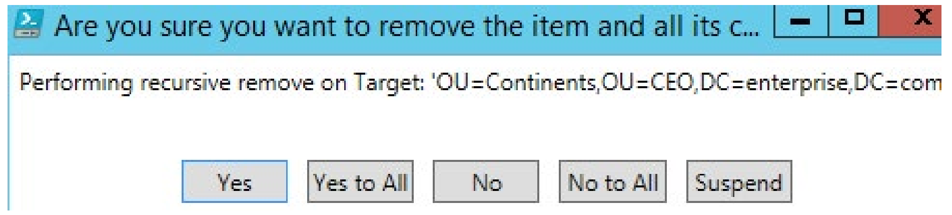
The **Remove-ADOrganizationalUnit** cmdlet removes an OU. The OU must not be protected from accidental deletion. You can remove the accidental deletion option for every OU that contains "Continents" in its name using the **Get-ADOrganizationalUnit** & **Set-ADOrganizationalUnit** cmdlets:

```
Get-ADOrganizationalUnit -filter "Name -eq
'Continents'" | Set-ADOrganizationalUnit
-ProtectedFromAccidentalDeletion $False
```

Use the following command to remove every OU that contains "Continents" in its name from AD:

```
Get-ADOrganizationalUnit -filter "Name -eq
'Continents'" | Remove-ADOrganizationalUnit
-Recursive
```

You will be prompted to confirm the deletion:


Are you sure you want to remove the item and all its c...

Performing recursive remove on Target: 'OU=Continents,OU=CEO,DC=enterprise,DC=com

Note that the -Recursive parameter removes both the OU and all of its child objects. The child objects will be deleted even if protection from deletion is on for them.

Before you try out these commands, be sure enable the Active Directory Recycle Bin feature so you can easily roll back any errant deletions. It's also smart to carefully track all changes to your organizational units.

[Quick Reference Guide]

Active Directory Auditing

Free Download

Jeff Melnick

IT Security Expert, Blogger

Most useful PowerShell commands for AD group management

Microsoft Active Directory serves as a centralized point for the administration, authorization and authentication. In AD, access to network resources is granted to security principals, such as user accounts and computer accounts, and those permissions can change over time. To simplify access management and improve security, medium and large companies often use Active Directory security groups, which can contain user accounts, computer accounts and other groups. They often also use distribution groups to manage email distribution lists. Both security and distribution groups have unique security identifiers (SIDs) and globally unique identifiers (GUIDs).

The ADUC MMC snap-in is great for managing both types of groups, but PowerShell is a much more efficient way to manage them in bulk.

In this article, we'll explain how to manage AD groups using Windows PowerShell scripts.

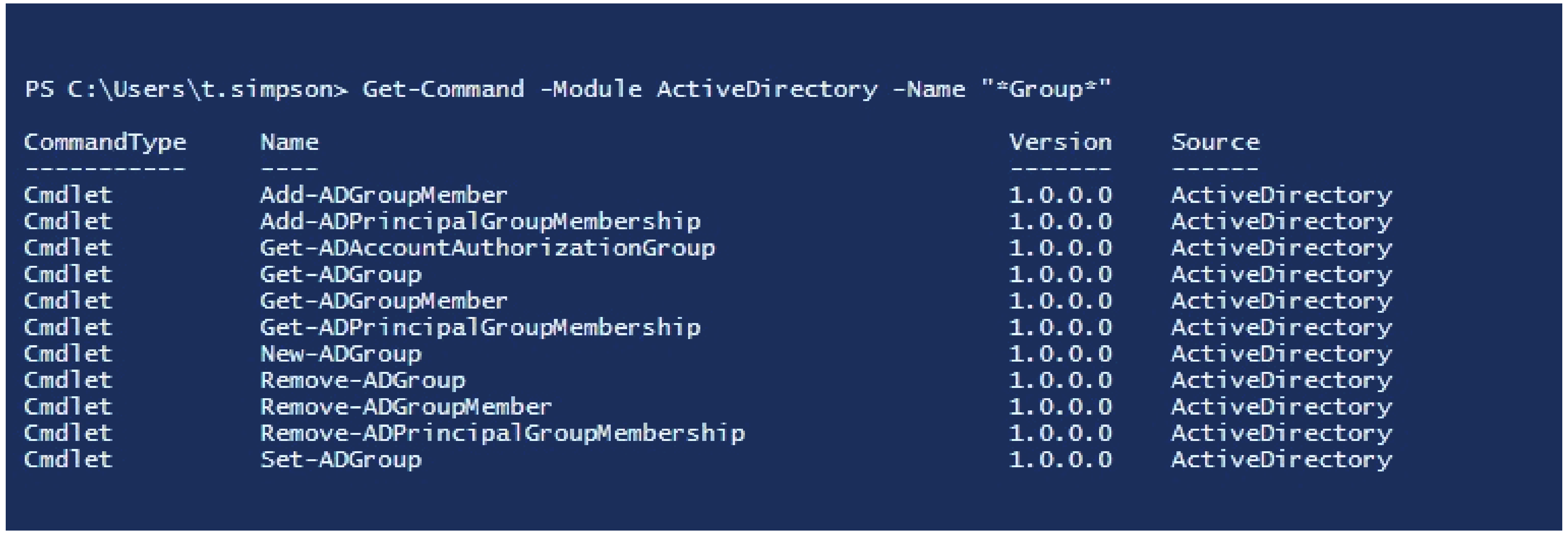
If you're not already familiar with AD groups and group management, please read the Active Directory Group Management Best Practice guide before you move on.

Also, keep in mind that in order to use these PowerShell scripts, you must import the module for interacting with AD — the Active Directory Module for Microsoft Windows PowerShell. This module was introduced in Windows Server 2008 R2 and is enabled by default in Windows Server 2012 and later. You can get the full list of AD module cmdlets by running the following command:

```
Get-Command -Module ActiveDirectory
```

The full list contains 147 cmdlets; however, only these eleven are related to Active Directory groups:

- Add-ADGroupMember
- Add-ADPrincipalGroupMembership
- Get-ADAccountAuthorizationGroup
- Get-ADGroup
- Get-ADGroupMember
- Get-ADPrincipalGroupMembership
- New-ADGroup
- Remove-ADGroup
- Remove-ADGroupMember
- Remove-ADPrincipalGroupMembership
- Set-ADGroup

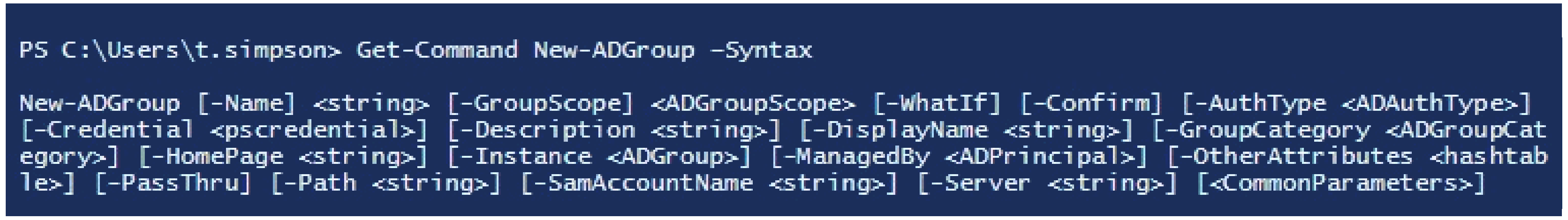


```
PS C:\Users\t.simpson> Get-Command -Module ActiveDirectory -Name "*Group*"

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Add-ADGroupMember                                  1.0.0.0    ActiveDirectory
Cmdlet          Add-ADPrincipalGroupMembership                     1.0.0.0    ActiveDirectory
Cmdlet          Get-ADAccountAuthorizationGroup                    1.0.0.0    ActiveDirectory
Cmdlet          Get-ADGroup                                        1.0.0.0    ActiveDirectory
Cmdlet          Get-ADGroupMember                                  1.0.0.0    ActiveDirectory
Cmdlet          Get-ADPrincipalGroupMembership                     1.0.0.0    ActiveDirectory
Cmdlet          New-ADGroup                                        1.0.0.0    ActiveDirectory
Cmdlet          Remove-ADGroup                                     1.0.0.0    ActiveDirectory
Cmdlet          Remove-ADGroupMember                               1.0.0.0    ActiveDirectory
Cmdlet          Remove-ADPrincipalGroupMembership                  1.0.0.0    ActiveDirectory
Cmdlet          Set-ADGroup                                        1.0.0.0    ActiveDirectory
```

Creating an Active Directory group

To create an AD group, use the New-ADGroup cmdlet. You can get its syntax by running the following command:

```
Get-Command New-ADGroup –Syntax
```



```
PS C:\Users\t.simpson> Get-Command New-ADGroup -Syntax

New-ADGroup [-Name] <string> [-GroupScope] <ADGroupScope> [-WhatIf] [-Confirm] [-AuthType <ADAuthType>]
[-Credential <pscredential>] [-Description <string>] [-DisplayName <string>] [-GroupCategory <ADGroupCat
egory>] [-HomePage <string>] [-Instance <ADGroup>] [-ManagedBy <ADPrincipal>] [-OtherAttributes <hashtab
le>] [-PassThru] [-Path <string>] [-SamAccountName <string>] [-Server <string>] [<CommonParameters>]
```

The easiest way to create a group is to run this short script:

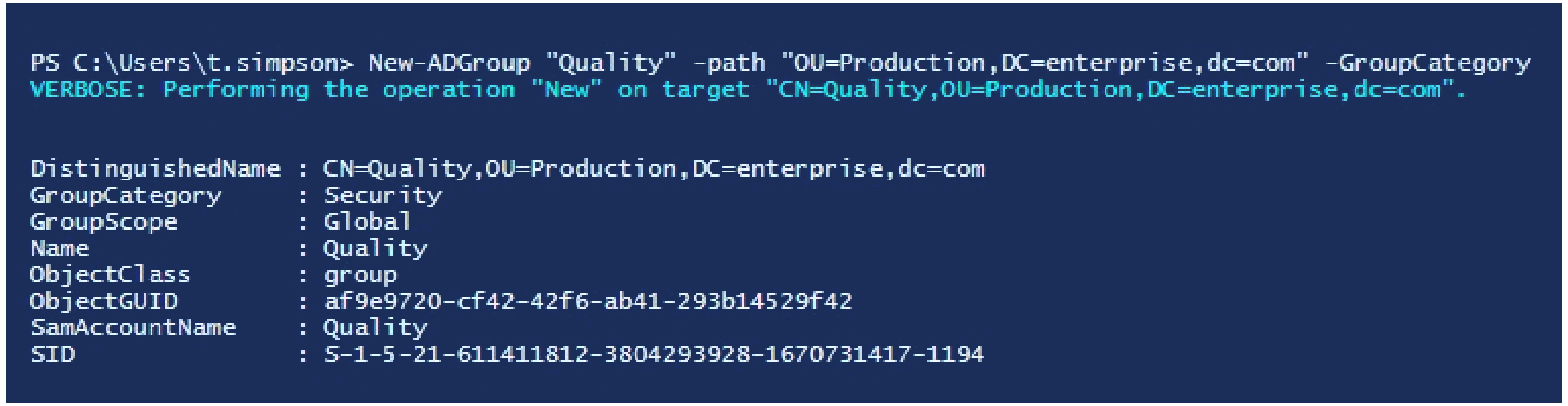
```
New-ADGroup "Group Name"
```

The system will ask you to specify the "GroupScope" parameter, and then it will create a new group. However, this group will have default values, such as:

- It will be created in the default LDAP container called "Users".
- It will have the "Security" group type.
- The members, member of, description, email and

Let's imagine that we want to create a security group called "Quality" on our AD DC. Let's use the following parameters: It should be in the "Production" OU (-Path), it should be a security group (-GroupCategory), and it should be global (-GroupScope).

```
New-ADGroup "Quality" -Path "OU=Produc-
tion,DC=enterprise,dc=com" -GroupCategory
Security -GroupScope Global -PassThru –Ver-
bose
```



```
PS C:\Users\t.simpson> New-ADGroup "Quality" -path "OU=Production,DC=enterprise,dc=com" -GroupCategory
VERBOSE: Performing the operation "New" on target "CN=Quality,OU=Production,DC=enterprise,dc=com".


DistinguishedName : CN=Quality,OU=Production,DC=enterprise,dc=com
GroupCategory     : Security
GroupScope        : Global
Name              : Quality
ObjectClass       : group
ObjectGUID        : af9e9720-cf42-42f6-ab41-293b14529f42
SamAccountName    : Quality
SID               : S-1-5-21-611411812-3804293928-1670731417-1194
```

If you want to make a universal distribution group, simply change the **–GroupCategory** parameter to “Distribution” and the **–GroupScope** parameter to “Universal.” You can also change the LDAP path by changing the **–Path** parameter.

Deleting an Active Directory group

To delete an AD group, use the Remove-ADGroup cmdlet. The easiest script for that will look like this:

```
Remove-ADGroup -Identity Quality
```

You’ll be prompted to confirm the deletion of the group.

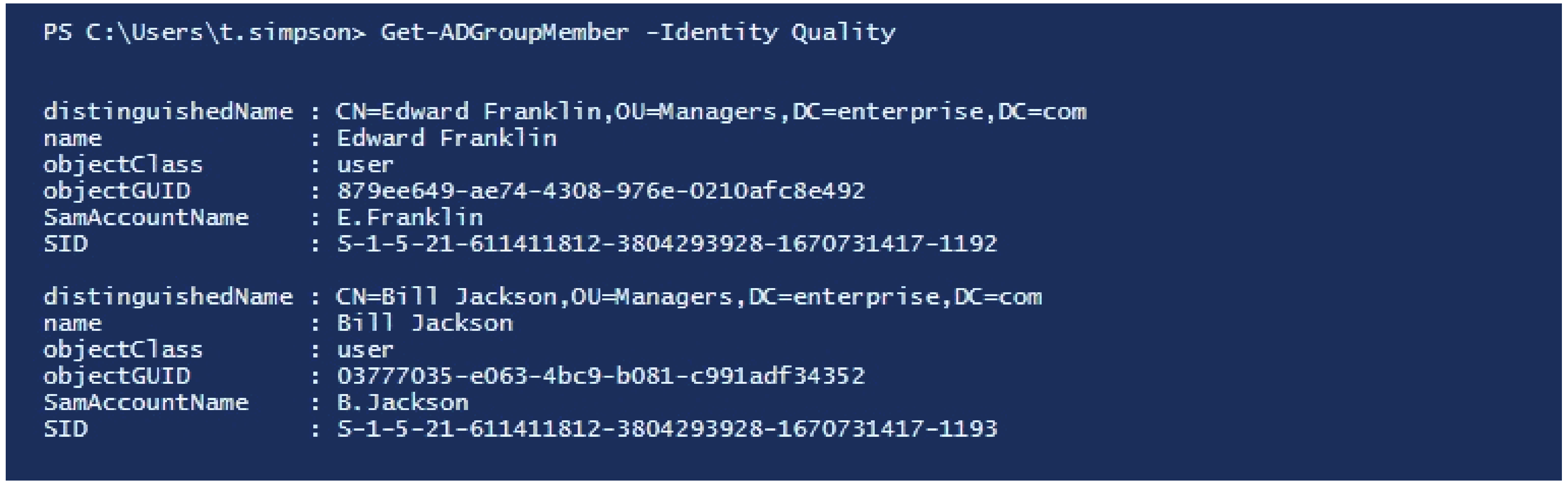
Adding users and computers to a group

You can add users to an AD group with the **Add-AdGroupMember** cmdlet. For instance, if you needed to add two users, B.Jackson and E.Franklin, to the “Quality” group, here is what the script would look like:

```
Add-AdGroupMember -Identity Quality -Mem-
bers B.Jackson, E.Franklin
```

Once you’ve added users to a security group, you can run the script below to verify that they are listed as members:

```
Get-ADGroupMember -Identity Quality
```



```
PS C:\Users\t.simpson> Get-ADGroupMember -Identity Quality


distinguishedName : CN=Edward Franklin,OU=Managers,DC=enterprise,DC=com
name              : Edward Franklin
objectClass       : user
objectGUID        : 879ee649-ae74-4308-976e-0210afc8e492
SamAccountName    : E.Franklin
SID               : S-1-5-21-611411812-3804293928-1670731417-1192

distinguishedName : CN=Bill Jackson,OU=Managers,DC=enterprise,DC=com
name              : Bill Jackson
objectClass       : user
objectGUID        : 03777035-e063-4bc9-b081-c991adf34352
SamAccountName    : B.Jackson
SID               : S-1-5-21-611411812-3804293928-1670731417-1193
```

If you need to add users to another security or distribution group, such as “Domain Admins”, specify “Domain Admins” as the value for the **–Identity** parameter. If you need one group to be a member of another, specify a group name as the value for the **–Members** parameter. The same principle applies to computer accounts, but you'll need to append a dollar sign ($) to the end of the computer account name. For example, to add the computer “WKS043” to a group, specify “WKS043$” as the value for the **–Member** parameter:

```
Add-AdGroupMember -Identity Quality -Members WKS043$
```

Adding a user to multiple groups

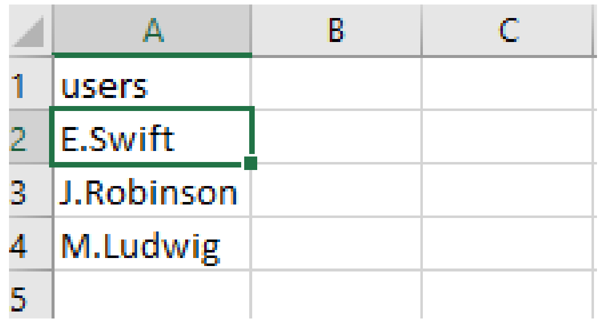
To add a user to multiple groups at once, run the following script.

```
“Managers”,”Quality” | Add-ADGroupMember -Members `
    (Read-Host -Prompt “Enter User Name”)
```

You'll be prompted to input the username.

Adding users to a group from a CSV file

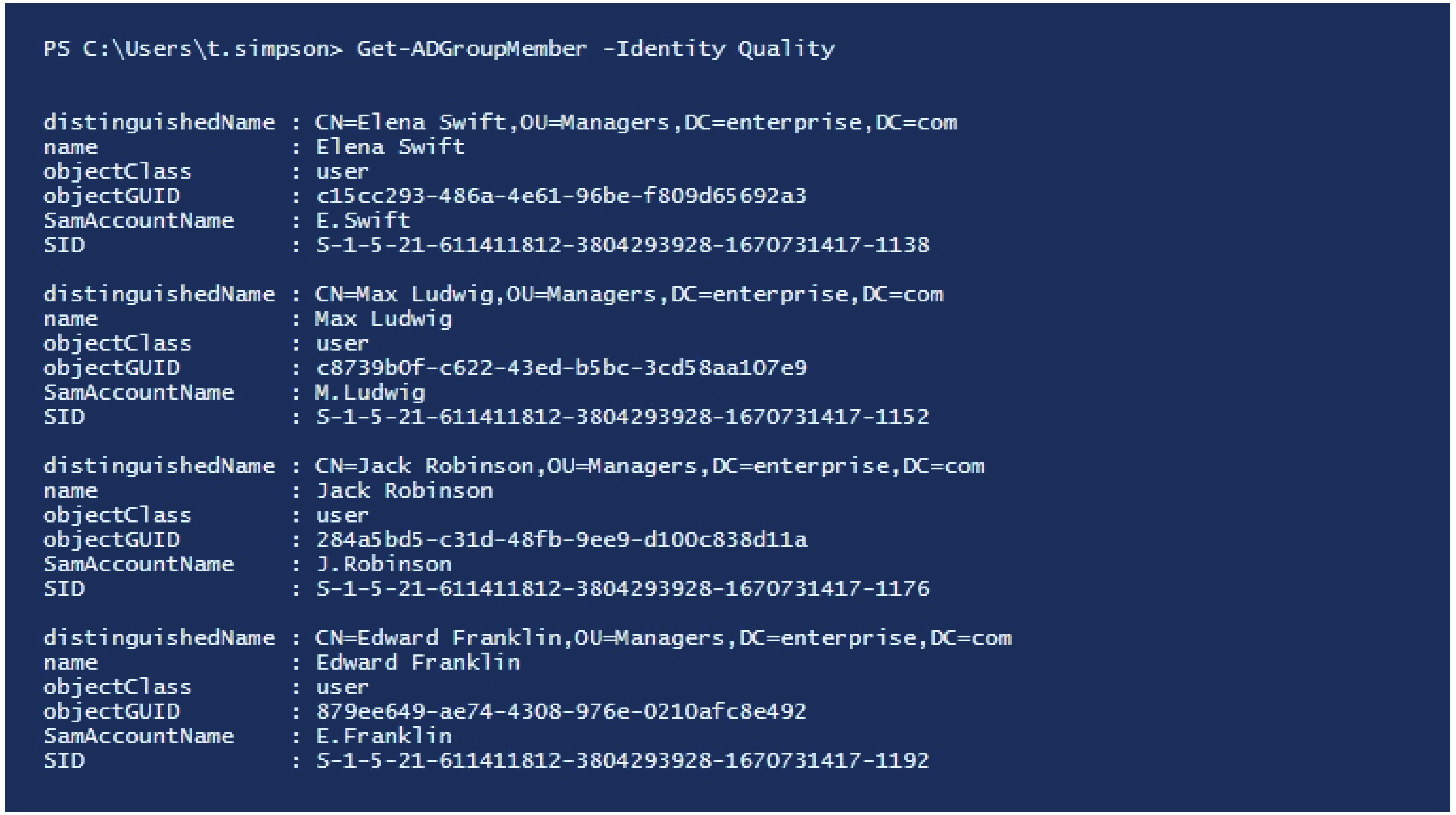
If you want to add a large number of users to a group, you can specify them in a CSV file and then import that file. Note that the list of the usernames in the CSV file must contain the SamAccountNames in the “users” column, as shown below:



	A	B	C
1	users		
2	E.Swift		
3	J.Robinson		
4	M.Ludwig		
5			

To add users to group from a CSV file, run the following PowerShell script:

```
Import-CSV C:\scripts\users.csv -Header
users | ForEach-Object {Add-AdGroupMember
-Identity “Quality” -members $_.users}
```



```
PS C:\Users\t.simpson> Get-ADGroupMember -Identity Quality


distinguishedName : CN=Elena Swift,OU=Managers,DC=enterprise,DC=com
name              : Elena Swift
objectClass       : user
objectGUID        : c15cc293-486a-4e61-96be-f809d65692a3
SamAccountName    : E.Swift
SID               : S-1-5-21-611411812-3804293928-1670731417-1138

distinguishedName : CN=Max Ludwig,OU=Managers,DC=enterprise,DC=com
name              : Max Ludwig
objectClass       : user
objectGUID        : c8739b0f-c622-43ed-b5bc-3cd58aa107e9
SamAccountName    : M.Ludwig
SID               : S-1-5-21-611411812-3804293928-1670731417-1152

distinguishedName : CN=Jack Robinson,OU=Managers,DC=enterprise,DC=com
name              : Jack Robinson
objectClass       : user
objectGUID        : 284a5bd5-c31d-48fb-9ee9-d100c838d11a
SamAccountName    : J.Robinson
SID               : S-1-5-21-611411812-3804293928-1670731417-1176

distinguishedName : CN=Edward Franklin,OU=Managers,DC=enterprise,DC=com
name              : Edward Franklin
objectClass       : user
objectGUID        : 879ee649-ae74-4308-976e-0210afc8e492
SamAccountName    : E.Franklin
SID               : S-1-5-21-611411812-3804293928-1670731417-1192
```

Copying users from one group to another

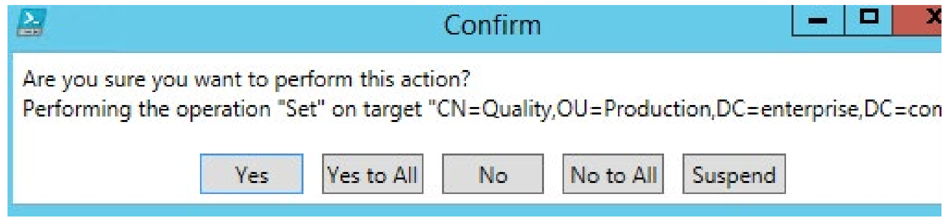
If you want to copy all members from one group to another group, run the following script:

```
Get-ADGroupMember “Quality” | Get-ADUser | ForEach-Object {Add-ADGroupMember -Identity “QualityControl”
Members $_}
```

Removing users or computers from a group

To remove a user from a group, use the **Remove-ADGroupMember** cmdlet:

```
Remove-ADGroupMember -Identity Quality
-Members J.Robinson
```



Are you sure you want to perform this action?
Performing the operation "Set" on target "CN=Quality,OU=Production,DC=enterprise,DC=con

To remove a computer account from a group, specify the computer name with a dollar sign ($) at the end for the value of the **-Members** parameter.

Removing multiple user accounts from a group

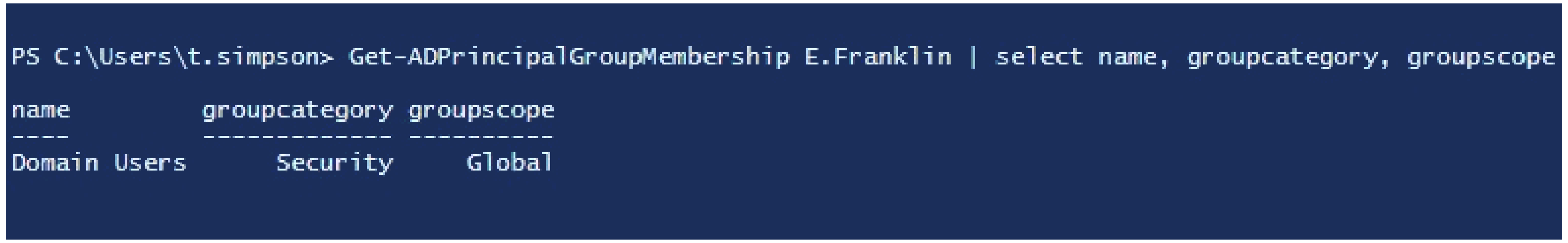
An easy way to remove multiple users from an AD group is to create a CSV file with the list of usernames and then remove those users from the group object using this script:

```
Import-CSV C:\scripts\users.csv -Header users | ForEach-Object {Remove-ADGroupMember -Identity
“Quality” -members $_.users}
```

Removing a user from all groups

To remove a user from all groups, run this script:

```
Get-ADUser -Identity E.Franklin -Properties MemberOf | ForEach-Object {
  $_.MemberOf | Remove-ADGroupMember -Members $_.DistinguishedName -Confirm:$false
}
```



```
PS C:\Users\t.simpson> Get-ADPrincipalGroupMembership E.Franklin | select name, groupcategory, groupscope

name         groupcategory groupscope
----         ------------- ----------
Domain Users      Security     Global
```

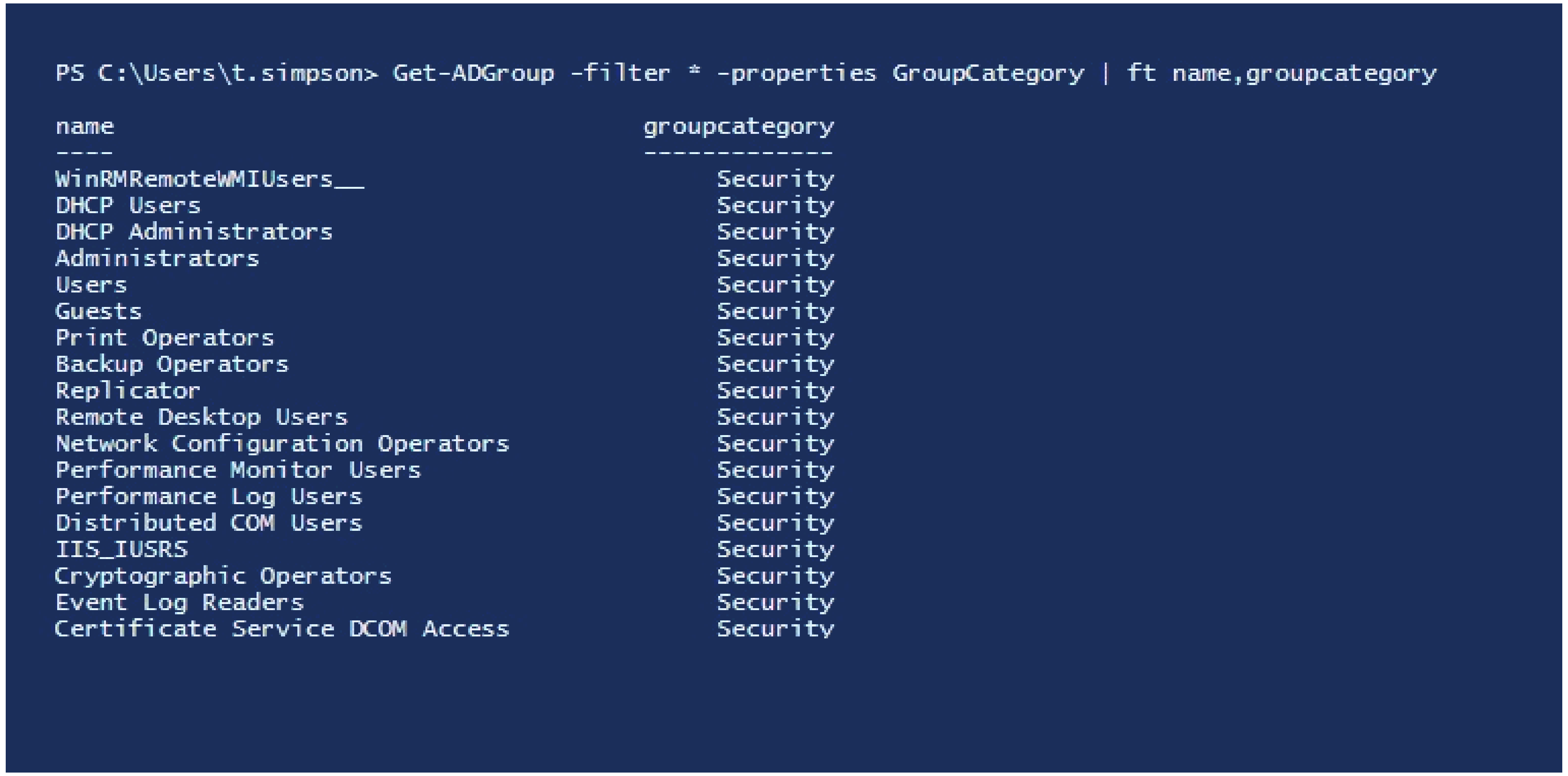
Note that the user will lose all group membership except “Domain Users”, which can be removed manually if needed.

Reporting on Active Directory groups

Now that we know how to perform many common Active Directory management tasks related to groups using PowerShell, let’s see how to report on what groups exist in AD:

To list all groups in AD, use the script below:

```
Get-ADGroup -filter * -properties GroupCate-
gory | ft name,groupcategory
```



```
PS C:\Users\t.simpson> Get-ADGroup -filter * -properties GroupCategory | ft name,groupcategory

name                                    groupcategory
----                                    -------------
WinRMRemoteWMIUsers__                        Security
DHCP Users                                   Security
DHCP Administrators                          Security
Administrators                               Security
Users                                        Security
Guests                                       Security
Print Operators                              Security
Backup Operators                             Security
Replicator                                   Security
Remote Desktop Users                         Security
Network Configuration Operators              Security
Performance Monitor Users                    Security
Performance Log Users                        Security
Distributed COM Users                        Security
IIS_IUSRS                                    Security
Cryptographic Operators                      Security
Event Log Readers                            Security
Certificate Service DCOM Access              Security
```

Of course, you'll also want to review AD group members and AD group membership changes.

Now that you've learned how to manage groups and group membership in Active Directory using PowerShell scripts, try performing some of the group management task yourself. However, be careful, and don't forget to enable the Active Directory Recycle Bin feature so you can easily roll back your changes if something goes wrong. Remember, the ADUC MMC snap-in is great for managing groups and group membership, but PowerShell is much better for managing groups in bulk.

[Best Practices Guide]

Active Directory Group Management

Free Download

Ryan Brooks

Product Evangelist

Seven challenges with Active Directory

Microsoft Active Directory (AD) is a reliable, scalable solution for managing users, resources and authentication in a Windows environment. However, like any software tool, it has limitations that can be difficult to overcome. Here are the top seven challenges with Active Directory and some options for addressing them:

CHALLENGE #1

Active Directory depends on Windows Server

Although Active Directory is compliant with Lightweight Directory Access Protocol (LDAP), there are many enhancements, extensions and interpretations of the LDAP specification. Software vendors sometimes choose to implement optional aspects of LDAP that are not supported by Active Directory, so using their products in an AD environment is difficult. For example, it is technically possible to implement Kerberos on Unix and then establish trusts with Active Directory, but the process is difficult and missteps are frequent. As a result, many organizations feel forced to limit themselves to Windows-based systems.

CHALLENGE #2

High license and maintenance cost

Microsoft uses client access licenses (CALs) for the Windows Server OS that underlies Active Directory. Since Windows Server 2016, Microsoft moved to per-core licensing: Pricing now starts at $6,156 for servers with two processors with eight cores each; the cost doubles if you use processors with 16 cores. That can be hard to swallow, especially given that Open LDAP and ApacheDS are both free of charge.

CHALLENGE #3

Inconvenient logging and auditing

Many things in Active Directory need proper logging, monitoring and analysis. For example, you need to be able to stay on top of critical errors and changes to AD objects and Group Policy, since they can affect both performance and security. But AD logs are very technical in nature, and finding the data you need requires tedious manual searching and filtering or advanced PowerShell scripting skills. Similarly, alerting and reporting is possible only through a combination of complicated PowerShell scripts and Task Scheduler. Each event log is capped at 4GB, which can lead to fast log overwrite and loss of important events. Finally, the PowerShell search engine is outdated so its performance is poor; for instance, every time you read records filtered by time, it reads the entire event log sequentially, record by record, until it finds the record you requested. This forces companies to integrate SIEM and Active Directory auditing solutions in order to ease log storage and analysis processes, spending money on things that could have been included in AD by design.

CHALLENGE #4

AD crashes lead to network downtime

When your AD is offline, you will experience the following issues:

- Users will be disconnected from file shares as soon as their authentication session expires, usually within a few hours.fields will all be blank.
- Software or hardware that relies on Active Directory authentication (such as IIS sites and VPN servers) will not let people log in. Depending on the setup, it will either immediately kick current users off or keep existing sessions until logout.
- Users will be able to log in to computers they used recently, because they will have a cached password or authentication ticket. However, anyone who hasn't used a given PC before, or last used it a long time ago, won't be able to log in until the connection to the DC is restored. Eventually, nobody will be able to log in with a domain account, because the cached authentications will expire within a few hours.
- Active Directory servers often play the role of DNS and DHCP servers. In that case, while AD is offline, computers will have trouble accessing the internet and even the local network itself.

To avoid these issues, best practices recommend having at least two Active Directory DCs with failover in place. That way, if one dies, you can just reinstall Windows Server on it, set it up as a new DC in an existing domain, and replicate everything back, with no downtime at all. However, this does incur extra expense for both hardware and AD licensing.

CHALLENGE #5

AD is prone to being hacked

Because Active Directory is the most popular directory service, there are a lot of techniques and strategies to hack it. Since it cannot be located in a DMZ, the AD server usually has an internet connection, which gives attackers the opportunity to get at the keys to your kingdom remotely. One particular weakness is that Active Directory uses the Kerberos authentication protocol with symmetrical cryptography architecture; Microsoft has already patched many of its vulnerabilities, but new ones continue to be discovered and exploited.

CHALLENGE #6

AD lacks GUI management capabilities

Microsoft bundles several utilities with AD, such as Active Directory Users and Computers (ADUC) and Group Policy Management Console (GPMC), to help organizations manage data and policies within the directory, but these tools are quite limited. For example, inserting object parameters in bulk requires PowerShell scripting; there is no alerting; and reporting is limited to exporting to a .txt file. AD delegation capabilities are also limited, so organizations often resort to splitting up domains to create boundaries for administrative access, which creates a directory infrastructure that is cumbersome to manage. To work around these issues, organizations often use third-party solutions that enable them to manage AD in bulk and control who can administer what in a more granular manner than the native AD tools. This gives them better control over identity and object access management and account management. Third- party AD management tools can automate operations around the creation, removal, modification of accounts, groups and Group Policy, as well as help with account lockout investigations.

CHALLENGE #7

AD does not provide a self-service portal for end users

It often makes sense to allow users to perform certain actions themselves, such as editing their own profiles and resetting their passwords if they forget them. However, Active Directory requires administrative access for these operations, so employees are forced to call the IT help desk to resolve their minor problems, which delays business workflows and drives up helpdesk costs. All these problems can be resolved via additional self-service management tools, but this is another item in the budget, on top of what you have already paid for AD.

Active Directory is a great tool, and it is still evolving, albeit slowly. If you want to integrate Active Directory into your environment, know that you will spend a big chunk of your budget on it, and even more if you want better AD management and reporting functionality. Obviously, system administrators can write custom scripts or programs to work around the shortcomings of native tools, and automate and improve AD management using scripting interfaces and frameworks provided by Microsoft or other parties. However, it takes advanced skills and a fair amount of time to write, maintain and run the scripts, and to work through their output to get actionable intelligence, which can lead to delayed response to serious security issues. And of course you're still subject to basic AD limitations like log file overwrites and lack of delegation. Turnover.

As a result, many **organizations turn to third-party solutions that improve and automate AD auditing, management and reporting**. Look for a solution that delivers visibility across your entire infrastructure, including not just AD but Exchange, file servers and SharePoint, and also integrates with SIEMs and Unix and Linux systems. Be sure it enables you to control who can administer what in a more granular manner than native AD tools, and automates operations around the creation, removal, modification of accounts, groups and Group Policy. Add bonus points if the solution offers self-service capabilities. And of course make certain it can capture and store a complete audit trail for years to support security investigations and comply with regulatory requirements.

Tool of the Month

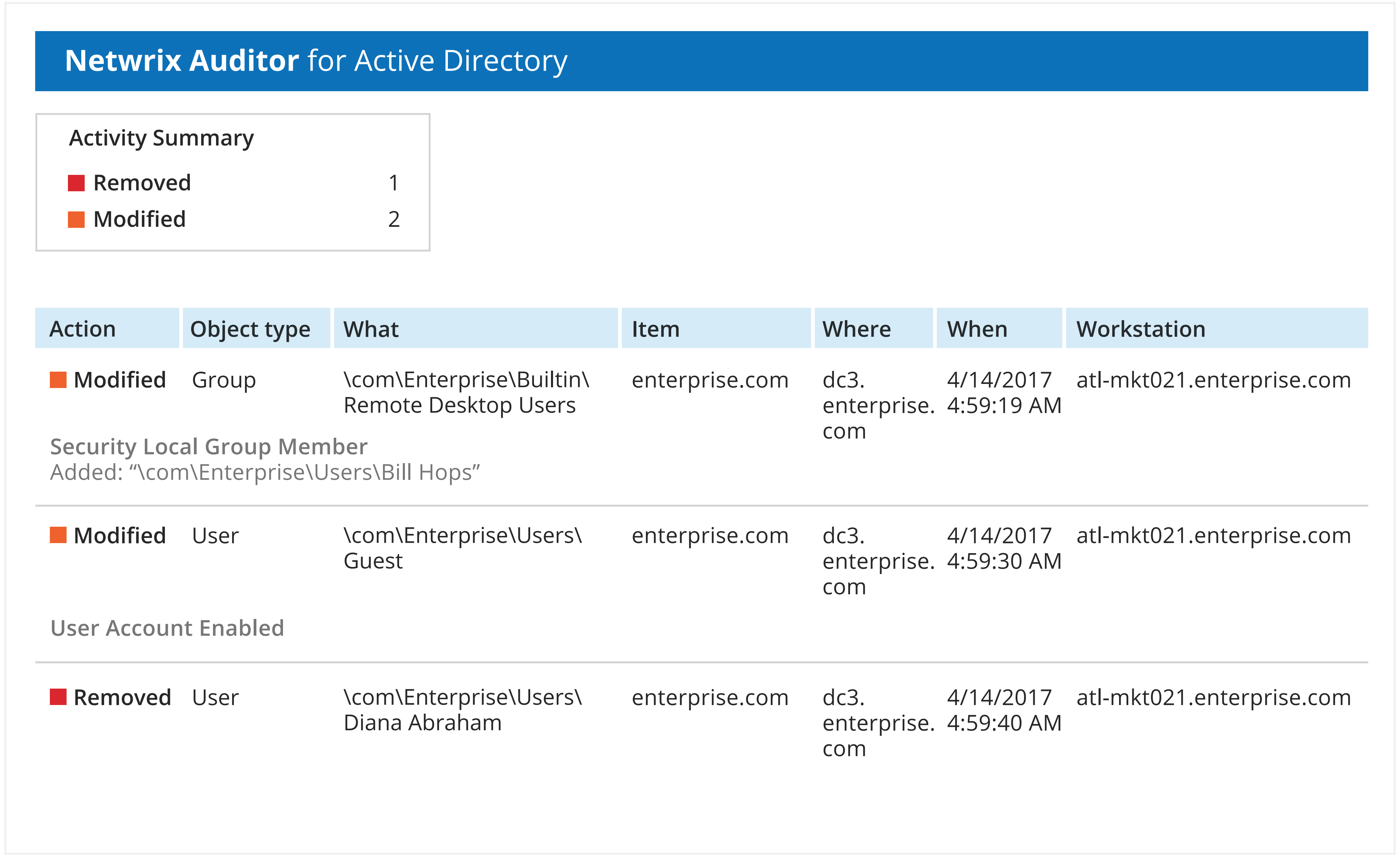
Free Community Edition

Netwrix Auditor for Active Directory

Download Free Tool

The free edition of Netwrix Auditor for Active Directory keeps you informed about what's going on in your AD by tracking logons and all changes to AD users, groups, organizational units, GPO links and various policies.

Daily activity summaries sent by the tool detail every change and logon that happened during the last 24 hours, including the before and after values for each modification.



Netwrix Auditor for Active Directory

Activity Summary	
🟥 Removed	1
🟧 Modified	2

Action	Object type	What	Item	Where	When	Workstation
🟧 Modified	Group	\com\Enterprise\Builtin\ Remote Desktop Users	enterprise.com	dc3. enterprise. com	4/14/2017 4:59:19 AM	atl-mkt021.enterprise.com
Security Local Group Member Added: "\com\Enterprise\Users\Bill Hops"						
🟧 Modified	User	\com\Enterprise\Users\ Guest	enterprise.com	dc3. enterprise. com	4/14/2017 4:59:30 AM	atl-mkt021.enterprise.com
User Account Enabled						
🟥 Removed	User	\com\Enterprise\Users\ Diana Abraham	enterprise.com	dc3. enterprise. com	4/14/2017 4:59:40 AM	atl-mkt021.enterprise.com

[On-Demand Webinar]

Top 5 critical Active Directory events you need visibility into

For many IT departments, Active Directory is the single most critical system within IT infrastructure, because it provides authentication and authorization for the entire organization. Without the oversight provided by change and configuration auditing, unwanted changes may occur undetected that put compliance initiatives at risk, compromise security, and impact business continuity.

Watch this webinar to learn the top 5 key Active Directory events you need stay on top of.

Watch Now

Corporate Headquarters:
300 Spectrum Center Drive,
Suite 200 Irvine, CA 92618

Phone: 1-949-407-5125
Toll-free: 888-638-9749
EMEA: +44 (0) 203-318-02

Copyright © Netwrix Corporation. All rights reserved. Netwrix is trademark of Netwrix Corporation and/or one or more of its subsidiaries and may be registered in the U.S. Patent and Trademark Office and in other countries. All other trademarks and registered trademarks are the property of their respective owners.