# Get-ADUser – How to Find and Export AD Users with PowerShell

lazyadmin.nl/powershell/get-aduser

February 14, 2022

The Active Directory is our main source when it comes to managing user accounts. The management console is great for looking up a single user, but when we need more, then the Get-ADUser cmdlet in PowerShell is much more powerful.

It allows us to quickly get a selection of users or to get details from a single or multiple users. It's also a great way to export users' accounts or information to a CSV file.

In this article, we are going to take a look at the get aduser cmdlet in PowerShell. Also, I will give you some useful examples when it comes to looking up and exporting ad users. And as a bonus, if have added a complete script to export your AD users.

## Install Active Directory Module

To be able to use the Get-ADuser cmdlet in PowerShell you will need to have the Active Directory Module installed. By default, it's installed on the domain controller, but on Windows 10 or 11 you will need to install it.

You can run the following PowerShell command in Windows 10 or 11 to install the module:

```
Add-WindowsCapability –online –Name “Rsat.ActiveDirectory.DS-LDS.Tools~~~~0.0.1.0”
```

## Finding Users with Get ADUser in PowerShell

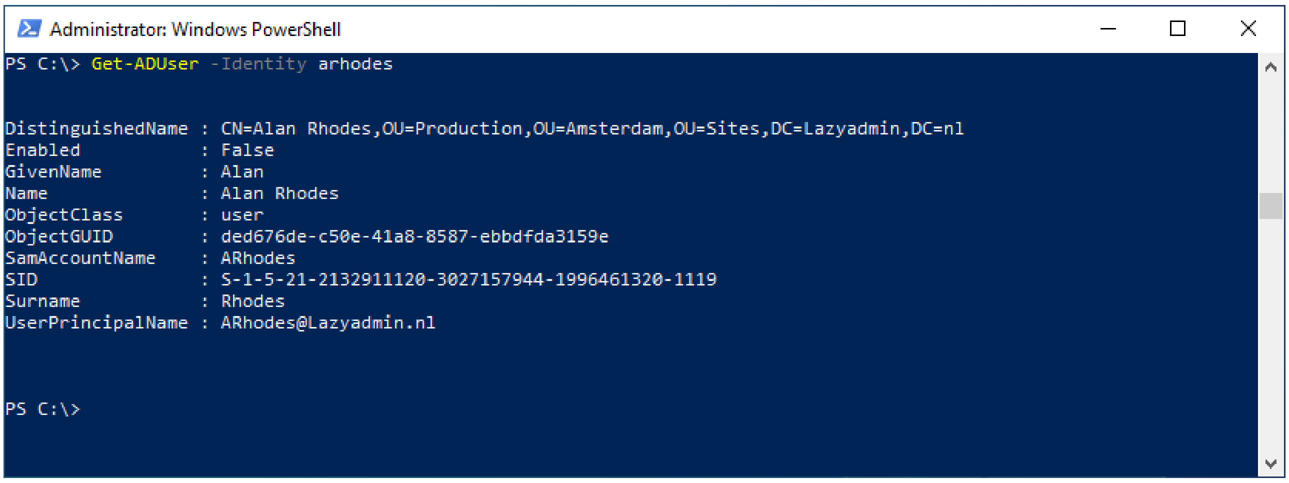
The Get-ADUser cmdlet allows us to find user accounts in the Active Directory and extract information from them. The true power of this cmdlet is that it comes with different options to find those user accounts.

We have the following options when it comes to finding accounts:

- **Identity** – Find a user account based on it's identity. This will return only a single user account
- **Filter** – Retrieve multiple objects (user accounts) based on a query
- **LDAPFilter** – Use a LDAP query string to filter the user accounts.
- **SearchBase** – Specify the Active Directory path (OU) to search in
- **SearchScope** – Specify how deep you want to search (baselevel, one level or complete subtree)

The identity parameter is mainly used when you know the user's `SAMAccountName` (login name). This allows you to select a single user from the Active Directory and view the properties of the account.

```
Get-ADUser -identity arhodes
```



```
PS C:\> Get-ADUser -Identity arhodes


DistinguishedName : CN=Alan Rhodes,OU=Production,OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=nl
Enabled           : False
GivenName         : Alan
Name              : Alan Rhodes
ObjectClass       : user
ObjectGUID        : ded676de-c50e-41a8-8587-ebbdfda3159e
SamAccountName    : ARhodes
SID               : S-1-5-21-2132911120-3027157944-1996461320-1119
Surname           : Rhodes
UserPrincipalName : ARhodes@Lazyadmin.nl



PS C:\>
```

Get-ADUser

As you can see some basic properties are returned of the user. We can use the -properties parameter to retrieve more information from the user. I will explain later more about retrieving different properties, but if you want to see all possible information of a user account, then use the following command:

Get-ADUser -identity arhodes -Propeties *

## Using the Filter

A more common way to find user(s) in the Active Directory is to use the `-filter` parameter. The filter parameter uses the PowerShell Expression Language the filter the result. This means that we can use the following operators in our queries:

| Operator | Description |
|---|---|
| **-eq** | Equals |
| **-le** | Less than or equal to |
| **-ge** | Great than or equal to |
| **-ne** | Not equal to |
| **-lt** | Less then |
| **-gt** | Greater then |
| **-like** | Like |
| **-notlike** | Not like |
| **-and -or** | and/or |
| **-not** | Not |

PowerShell Expression Language Operators

So let's take a look at a couple of commonly used examples when filtering Active Directory users.

To find a user by their first or last name we can use the following filter

# Search on first name

Get-ADUser -Filter "GivenName -eq 'Alan'"
# Search on last name:
Get-ADUser -Filter "Surname -eq 'Rhodes'"
Another option is to use the like operator in combination with a wildcard. This is particularly useful when you know a part of the name. Note the `*` symbol that indicates the wildcard.

Get-ADUser -Filter "name -like '*rho*'"
The `-ge` and `-le` can for example be used to find all users based on their failed login attempts:

# Find all users that have more then 3 failed login attempts
Get-ADUser -Filter "badpwdcount -ge 3"
To find all users that have a particular field not filled, we can use the `-notlike` operator. For example, we want to retrieve all users that don't have the email address field filled in.

Using the `-notlike` filter in combination with a wildcard we can search for all users that don't have any data in the email field.

Get-ADUser -Filter "Email -notlike '*'" | ft
You can easily combine this with the `Set-ADUser` cmdlet, to update a lot of users with a single command. For example, we get all users that don't have the webpage attribute filled, and set it to the company website:

Get-ADUser -Filter "homepage -notlike '*'" | Set-ADUser -HomePage "lazyadmin.nl"

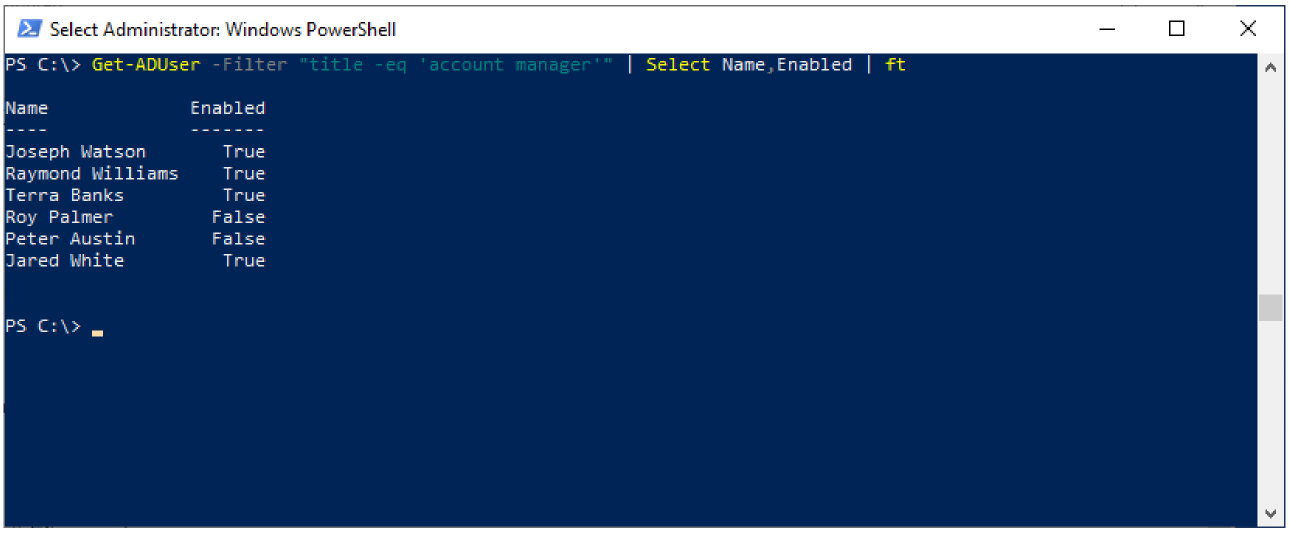
## Combining filters

We can also expand our filter query with multiple expressions. This allows you to further narrow down your filter queries. For example, we want to find all employees that are "account manager".

If we would use the following filter, then it will return all users with the job title account manager:

Get-ADUser -Filter "title -eq 'account manager'" | Select Name,Enabled | ft
But the problem is that this also includes accounts that are disabled. We only want active users, we are not interested in the users that already left the company. Thus what we can do is also filter on accounts that are enabled:

Get-ADUser -Filter "title -eq 'account manager' -and enabled -like 'true'" | Select Name,Enabled | ft



```
PS C:\> Get-ADUser -Filter "title -eq 'account manager'" | Select Name,Enabled | ft

Name             Enabled
----             -------
Joseph Watson       True
Raymond Williams    True
Terra Banks         True
Roy Palmer         False
Peter Austin       False
Jared White         True


PS C:\> _
```

Get-ADUsers with filters

The last filter example that I want to share with you is to get users based on a date field. Let's say we want to find all inactive accounts.

For this, we can use the `lastlogon` field in the Active Directory. We first need to create a date variable, by taking the date from today and subtracting 30 days from it.

We can then get all users that haven't login the past 30 days on the domain:

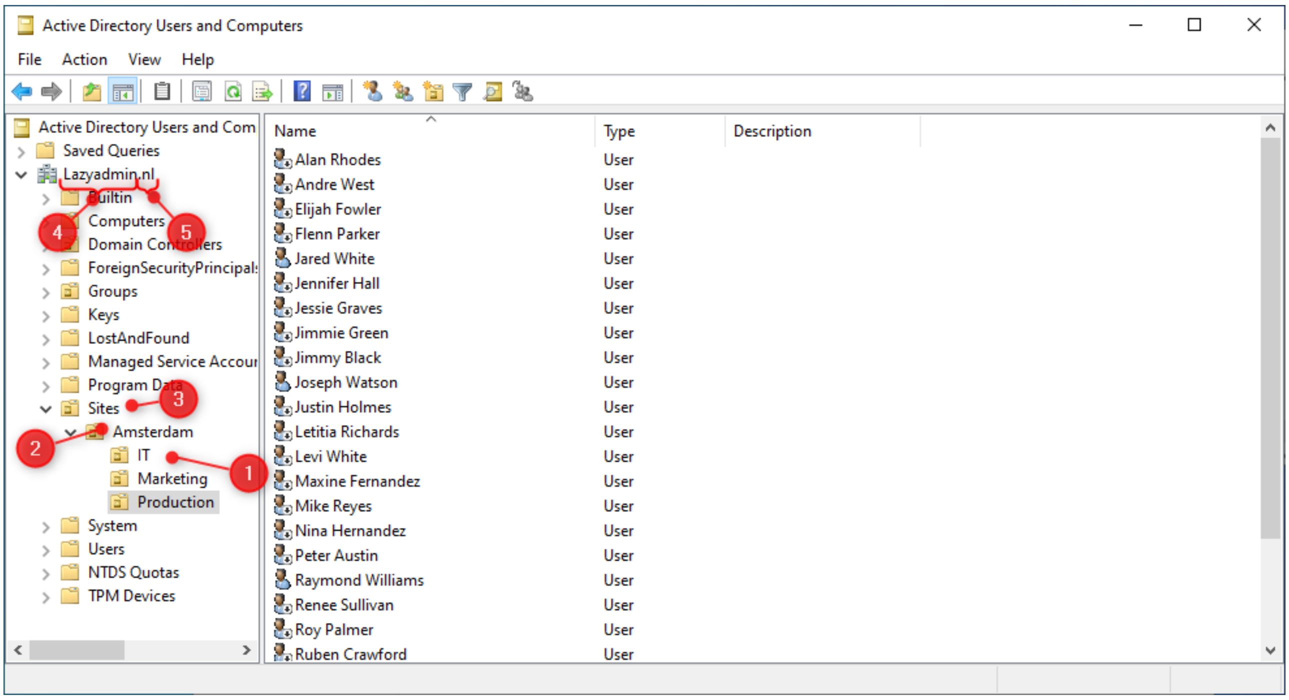
```
$date = (Get-Date) - (New-TimeSpan -Days 30)
Get-ADUser -Filter 'lastLogon -lt $date' | ft
```

## Get ADUser SearchBase

When you have a lot of users in your Active Directory you probably want to narrow down the search. To do this we can use the -SearchBase parameter for the Get-ADUser cmdlet. This allows us to specify the `distinguishedName` (OU level) where we want to search.

To specify the OU where we want to search we need to write the distinguishedName from the bottom up. Thus, the string starts with the OU where you want to search and ends with the domain name.

Take the following Active Directory structure, we want to get all users from the IT OU:



| Name | Type | Description |
|---|---|---|
| Alan Rhodes | User | |
| Andre West | User | |
| Elijah Fowler | User | |
| Flenn Parker | User | |
| Jared White | User | |
| Jennifer Hall | User | |
| Jessie Graves | User | |
| Jimmie Green | User | |
| Jimmy Black | User | |
| Joseph Watson | User | |
| Justin Holmes | User | |
| Letitia Richards | User | |
| Levi White | User | |
| Maxine Fernandez | User | |
| Mike Reyes | User | |
| Nina Hernandez | User | |
| Peter Austin | User | |
| Raymond Williams | User | |
| Renee Sullivan | User | |
| Roy Palmer | User | |
| Ruben Crawford | User | |

SearchBase path

The `SearchBase` string, in this case, would be:

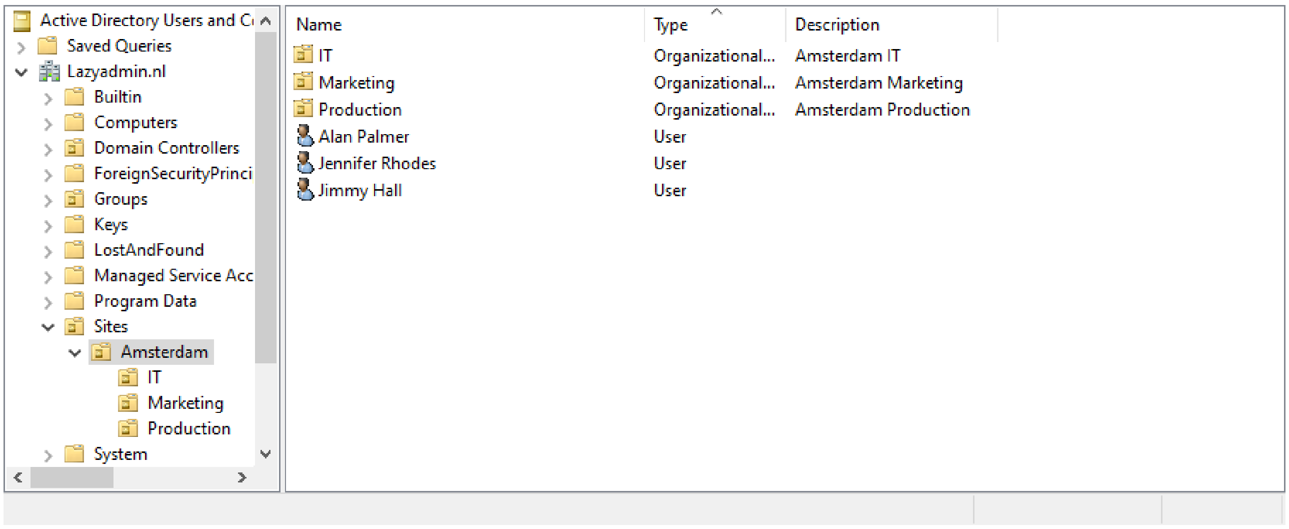
1: IT
2: Amsterdam
3: Sites
4: Lazyadmin
5: NL
"OU=IT,OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=NL"
Thus to get all users from the IT department in Amsterdam we can use the following PowerShell command:

```
Get-ADUser -Filter * -SearchBase "OU=IT,OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=NL" | ft
```

## Using the SearchScope

By default, the `-SearchBase` parameter will return all users from the specified OU and nested OU's. With the `-SearchScope` parameter, we can specify how deep or not we want to search through the Active Directory tree.

Let's say we want the users from the Amsterdam OU, but not from the nested OU's.



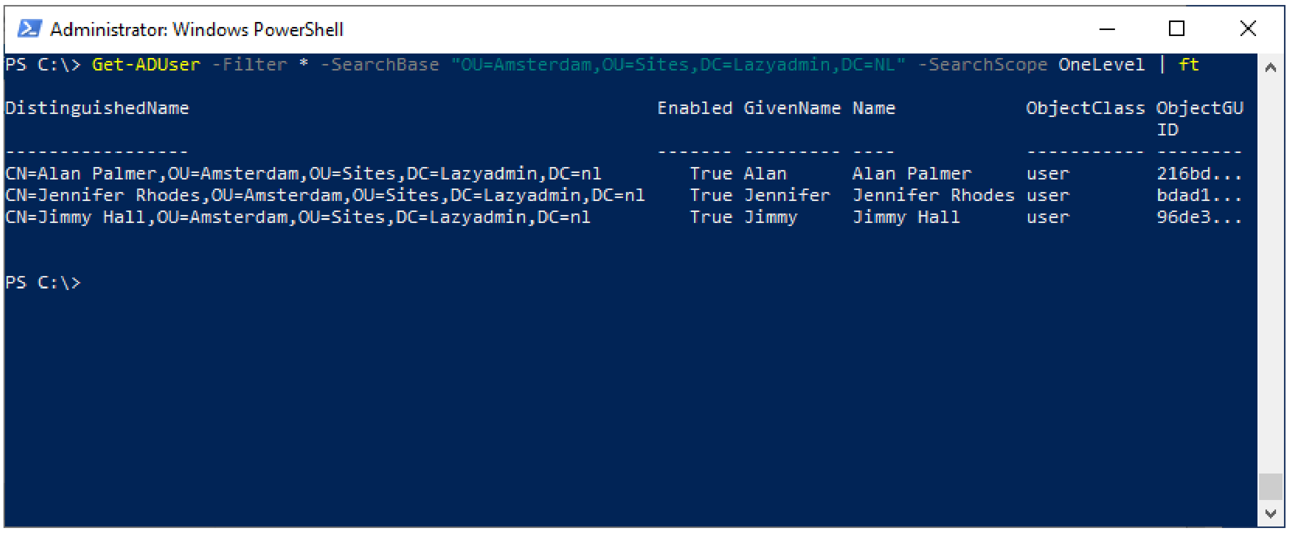
| Name | Type | Description |
|---|---|---|
| IT | Organizational... | Amsterdam IT |
| Marketing | Organizational... | Amsterdam Marketing |
| Production | Organizational... | Amsterdam Production |
| Alan Palmer | User | |
| Jennifer Rhodes | User | |
| Jimmy Hall | User | |

Active Directory

If we would use the following SearchBase, then all users, including those from the IT, Marketing, and Production OU's are returned:

Get-ADUser -Filter * -SearchBase "OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=NL" | ft
To get only the users from the Amsterdam OU we can use the SearchScope parameter. This allows us to limit the SearchBase to the current level only:

Get-ADUser -Filter * -SearchBase "OU=IT,OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=NL" -SearchScope OneLevel | ft



```
PS C:\> Get-ADUser -Filter * -SearchBase "OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=NL" -SearchScope OneLevel | ft

DistinguishedName                                           Enabled GivenName Name            ObjectClass ObjectGU
                                                                                                          ID
-----------------                                           ------- --------- ----            ----------- --------
CN=Alan Palmer,OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=nl        True Alan      Alan Palmer     user        216bd...
CN=Jennifer Rhodes,OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=nl    True Jennifer  Jennifer Rhodes user        bdad1...
CN=Jimmy Hall,OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=nl         True Jimmy     Jimmy Hall      user        96de3...


PS C:\>
```

Specify SearchScope to Get ADUsers

## Get ADUser Properties

When using the Get ADUser cmdlet you may have noticed that it will only return a couple of properties of the user account. But as you probably know, the user account has a lot more properties.

To return all properties, we need to add the -properties parameter to the cmdlet:

```
Get-ADUser -identity arhodes -properties *
```

This will return a long list with all the possible properties of a user account. Now, this is probably too much information, so you might want to specify the actual fields that you need.

To do this you will need to pipe the select command behind it, where we specify the fields that we need. It's also a good idea to specify the same fields in the properties parameter, instead of requesting all the data.

For example, if you want the full name, job title, and email address, then you could use the following command:

```
Get-ADUser -identity arhodes -properties emailaddress,title | select name,emailaddress,title
```

### Selecting Distinct Values

The next tip is not really related to the Get ADUser cmdlet, but it's something I use often in combination with getting user information out of the Active Directory.

When you want to export a list of all possible job titles in your Active Directory you can use the -Unique parameter in PowerShell.

```
Get-ADUser -Filter * -Properties title | Select title -Unique
```

This command will give you all the job titles that you have used in your Active Directory.

## Export AD Users to CSV with PowerShell

Exporting results in PowerShell to CSV is pretty common. We all use PowerShell often to retrieve information only to process it further in Excel. I have written a complete guide about the Export-CSV cmdlet, but I also want to give you a couple of useful examples when working with the Get-ADUser cmdlet.

To simply export all AD Users we can use the following command:

```
Get-ADUser -filter * | Export-CSV c:\temp\users.csv -NoTypeInformation
```

But as you will notice this will give not really the results that you are looking for. It will include all user accounts, enabled and disabled, and not really the information that we need.

### Export extra fields

So the first step is to specify the fields that we really want to export. For example, if we want to export the names, job titles, department, and email addresses we could use the following command:

```
Get-ADUser -filter * -properties mail,title,department | select name,title,department,mail | Export-CSV c:\temp\users.csv -NoTypeInformation
```

### Export only enabled users

To export only the active (enabled) users from the AD to a CSV file we can use the following command in PowerShell:

```
# Export all enabled users
Get-ADUser -Filter "Enabled -like 'true'" | Export-CSV c:\temp\users.csv -NoTypeInformation
# OR Export only disabled users
```

```
Get-ADUser -Filter "Enabled -like 'False'" | Export-CSV c:\temp\users.csv -NoTypeInformation
```

## Complete Export AD Users to CSV Script

I have created a PowerShell script that will Export all AD Users to CSV for you with the most commonly needed properties.

When you run the script you specify a couple of options:

- **Get the manager's display name or not** (default true)
- **Specify the searchBase** (OU), default whole Active Directory
- **Get enabled or disabled accounts or both** (default only enabled)
- **Export path CSV file** (default script location)

The script will get all the user accounts from the active directory if you don't specify the searchBase (OU). It's also possible to specify multiple OU's:

```
.\Get-ADusers.ps1 -searchBase
"OU=users,OU=Amsterdam,DC=LazyAdmin,DC=Local","OU=users,OU=Oslo,DC=LazyAdmin,DC=Local"
-path c:\temp\users.csv
```

Follow these steps to export the AD Users with the PowerShell script:

1. **Download** the complete Export AD Users script from my Github
2. **Open PowerShell** and navigate to the script
3. **Run the export script**: Get-ADUsers.ps1 -csvpath c:\temp\adusers.csv

When complete, the script will automatically open Excel for you. You can also run the script without the csvpath parameter to output the result to the console.

```
param(
[Parameter(
Mandatory = $false,
HelpMessage = "Get the users manager"
)]
[switch]$getManager = $true,
[Parameter(
Mandatory = $false,
HelpMessage = "Enter the searchbase between quotes or multiple separated with a comma"
)]
[string[]]$searchBase,
[Parameter(
Mandatory = $false,
HelpMessage = "Get accounts that are enabled, disabled or both"
)]
[ValidateSet("true", "false", "both")]
[string]$enabled = "true",
[Parameter(
Mandatory = $false,
HelpMessage = "Enter path to save the CSV file"
)]
[string]$CSVpath
)
Function Get-Users {
```

```
<#
.SYNOPSIS
Get users from the requested DN
#>
param(
[Parameter(
Mandatory = $true
)]
$dn
)
process{
# Set the properties to retrieve
$properties = @(
'name',
'userprincipalname',
'mail',
'title',
'enabled',
'manager',
'department',
'telephoneNumber',
'office',
'mobile',
'streetAddress',
'city',
'postalcode',
'state',
'country',
'description',
'lastlogondate'
)
# Get enabled, disabled or both users
switch ($enabled)
{
"true" {$filter = "enabled -eq 'true'"}
"false" {$filter = "enabled -eq 'false'"}
"both" {$filter = "*"}
}
# Get the users
Get-ADUser -Filter $filter -Properties $properties -SearchBase $dn | select $properties
}
}
Function Get-AllADUsers {
<#
.SYNOPSIS
Get all AD users
#>
process {
Write-Host "Collecting users" -ForegroundColor Cyan
$users = @()
```

```
if ($searchBase) {
# Get the requested mailboxes
foreach ($dn in $searchBase) {
Write-Host "- Get users in $dn" -ForegroundColor Cyan
$users += Get-Users -dn $dn
}
}else{
# Get distinguishedName of the domain
$dn = Get-ADDomain | Select -ExpandProperty DistinguishedName
Write-Host "- Get users in $dn" -ForegroundColor Cyan
$users += Get-Users -dn $dn
}
$users | ForEach {
$manager = ""
If (($getManager.IsPresent) -and ($_.manager)) {
# Get the users' manager
$manager = Get-ADUser -Identity $_.manager | Select -ExpandProperty Name
}
[pscustomobject]@{
"Name" = $_.Name
"UserPrincipalName" = $_.UserPrincipalName
"Emailaddress" = $_.mail
"Job title" = $_.Title
"Manager" = $manager
"Department" = $_.Department
"Office" = $_.Office
"Phone" = $_.telephoneNumber
"Mobile" = $_.mobile
"Enabled" = $_.enabled
"Street" = $_.StreetAddress
"City" = $_.City
"Postal code" = $_.PostalCode
"State" = $_.State
"Country" = $_.Country
"Description" = $_.Description
"Last login" = $_.lastlogondate
}
}
}
}
If ($CSVpath) {
# Get mailbox status
Get-AllADUsers | Sort-Object Name | Export-CSV -Path $CSVpath -NoTypeInformation -Encoding
UTF8
if ((Get-Item $CSVpath).Length -gt 0) {
Write-Host "Report finished and saved in $CSVpath" -ForegroundColor Green
# Open the CSV file
Invoke-Item $path
}
else {
```

```
Write-Host "Failed to create report" -ForegroundColor Red
}
}
Else {
Get-AllADUsers | Sort-Object Name
}
```

## Wrapping Up

The Get ADUser cmdlet is really useful when it comes to exacting information out of the Active Directory. Using the different filters allows you to retrieve only the information that you really need.

To Export the AD users to CSV you can try the script. You can easily change the properties that it retrieves to your own need.

If you have any questions, just drop a comment below.