How to use Get-ADComputer in PowerShell

lazyadmin.nl/powershell/get-adcomputer

May 24, 2022

Need to export all computers from an OU, or look up a computer in your Active Directory? Or do you want to count how many computers you have? We can use the `Get-ADComputer` cmdlet in PowerShell to quickly extract computer information from the AD.

The Active Directory contains all the computers that are members of our domain. The management console is great to look up a single computer. But when you want to get details of single or multiple computers, then we need to use PowerShell.

In this article, we are going to take a look at how to use the `Get-ADComputer` cmdlet in PowerShell. I will also give you some useful examples when it comes to looking up and exporting AD computers. And as a bonus, if have added a complete script to export your Active Directory computers.

Install Active Directory Module

To be able to use the Get-ADComputer 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 Computers with Get ADComputer in PowerShell

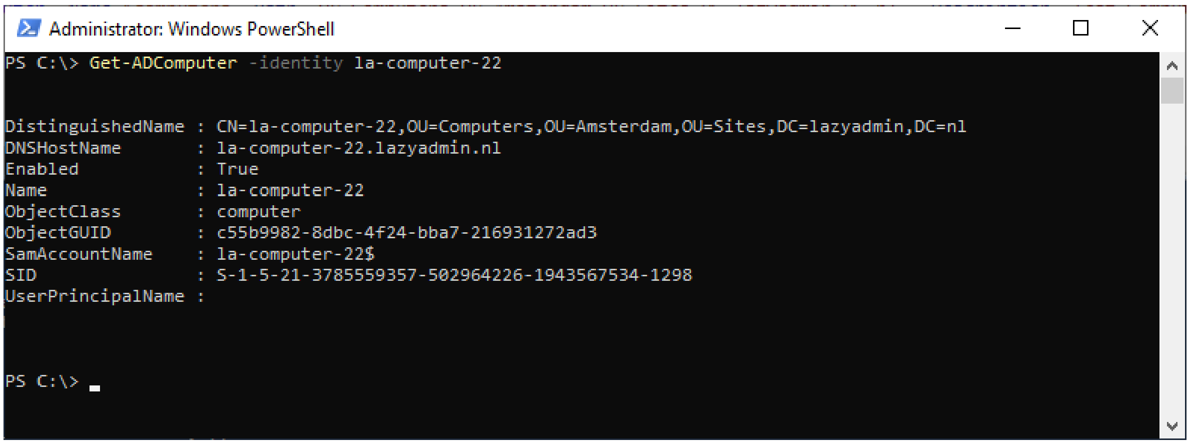
The Get-ADComputer cmdlet allows us to find computer objects in the Active Directory and extract information from them. The true power of this cmdlet is that it comes with different options to find those computer objects.

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

- **Identity** – Find computers based on their name. This will return only a single computer
- **Filter** – Retrieve multiple objects based on a query
- **LDAPFilter** – Use a LDAP query string to filter the computer objects
- **SearchBase** – Specify the Active Directory path (OU) to search in
- **SearchScope** – Specify how deep you want to search (base level, one level, or complete subtree)

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

Get-ADComputer -identity la-computer-22



```
PS C:\> Get-ADComputer -identity la-computer-22


DistinguishedName : CN=la-computer-22,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl
DNSHostName       : la-computer-22.lazyadmin.nl
Enabled           : True
Name              : la-computer-22
ObjectClass       : computer
ObjectGUID        : c55b9982-8dbc-4f24-bba7-216931272ad3
SamAccountName    : la-computer-22$
SID               : S-1-5-21-3785559357-502964226-1943567534-1298
UserPrincipalName :



PS C:\> _
```

Get-ADComputer

This will return the basic properties of the computer. We can use the -properties parameter to retrieve more information from the computer. I will explain more about retrieving different properties later in the article, but if you want to see all possible information about a computer account, then use the following command:

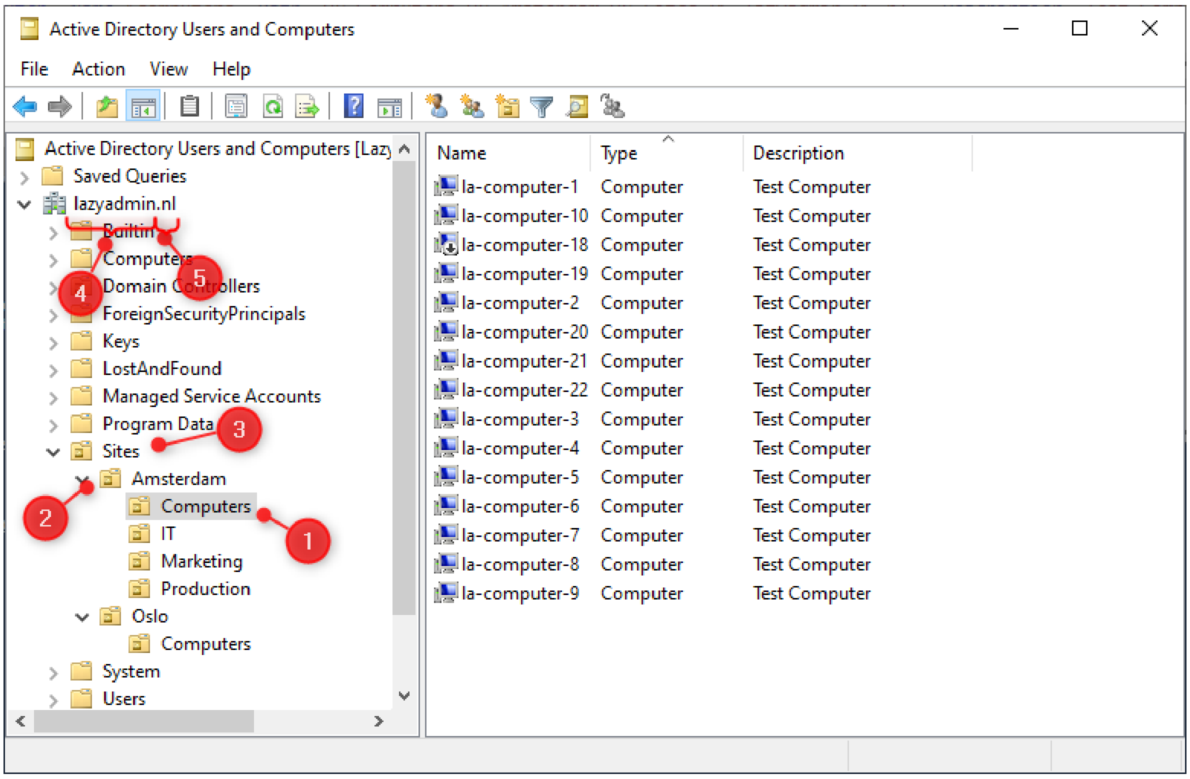
```
Get-ADComputer -identity la-computer-22 -properties *
```

Get ADComputer SearchBase

Most of the time when you want to retrieve computers from the Active Directory you want to narrow down the search base on OU. To do this we can use the -SearchBase parameter for the Get-ADComputer cmdlet. This allows us to specify the OU (distinguishedName) where we want to search.

The distinguishedName is the full path of the OU, which we write from the OU up the tree to the AD domain name.

Take the following AD structure, we want to get all computers from the Amsterdam site:



Name	Type	Description
la-computer-1	Computer	Test Computer
la-computer-10	Computer	Test Computer
la-computer-18	Computer	Test Computer
la-computer-19	Computer	Test Computer
la-computer-2	Computer	Test Computer
la-computer-20	Computer	Test Computer
la-computer-21	Computer	Test Computer
la-computer-22	Computer	Test Computer
la-computer-3	Computer	Test Computer
la-computer-4	Computer	Test Computer
la-computer-5	Computer	Test Computer
la-computer-6	Computer	Test Computer
la-computer-7	Computer	Test Computer
la-computer-8	Computer	Test Computer
la-computer-9	Computer	Test Computer

SearchBase Path

The search base string, in this case, would be:

```
1 2 3 4 5
"OU=Computers,OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=NL"
```

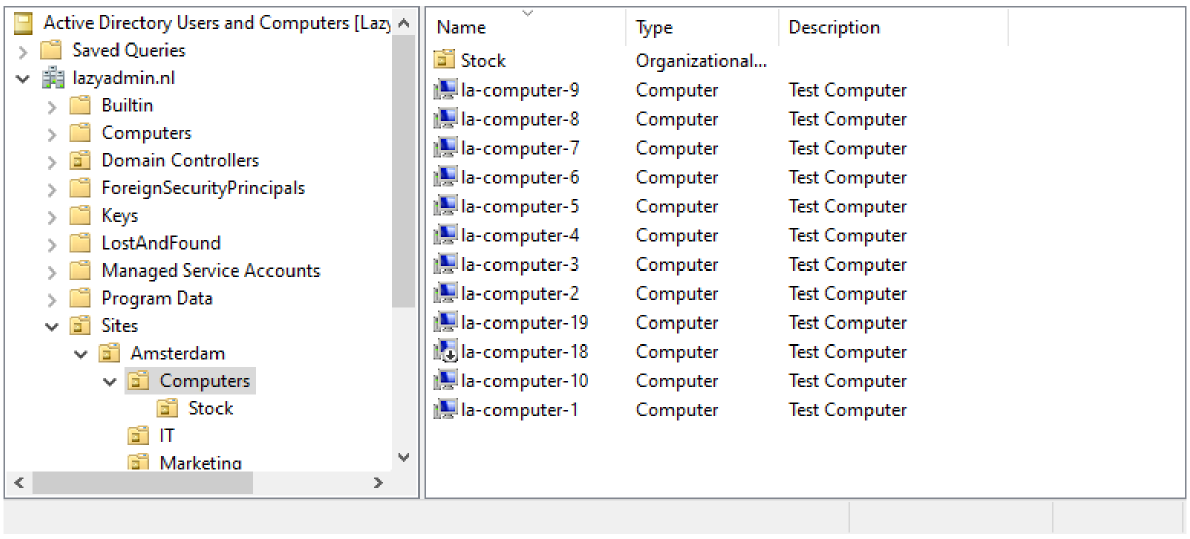
Thus to get all computers from the site Amsterdam, we can use the following PowerShell command:

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

Using the SearchScope

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

Let's say we want to get all computers from Amsterdam, except the computers that are in stock:


- Active Directory Users and Computers [Lazy
  - Saved Queries
  - lazyadmin.nl
    - Builtin
    - Computers
    - Domain Controllers
    - ForeignSecurityPrincipals
    - Keys
    - LostAndFound
    - Managed Service Accounts
    - Program Data
    - Sites
      - Amsterdam
        - Computers
          - Stock
        - IT
        - Marketing

Name	Type	Description
Stock	Organizational...	
la-computer-9	Computer	Test Computer
la-computer-8	Computer	Test Computer
la-computer-7	Computer	Test Computer
la-computer-6	Computer	Test Computer
la-computer-5	Computer	Test Computer
la-computer-4	Computer	Test Computer
la-computer-3	Computer	Test Computer
la-computer-2	Computer	Test Computer
la-computer-19	Computer	Test Computer
la-computer-18	Computer	Test Computer
la-computer-10	Computer	Test Computer
la-computer-1	Computer	Test Computer

If we would use the searchbase that we created earlier, then all computers, including those in stock, will be returned, a total of 15 computers.

```
$searchBase = "OU=Computers,OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=NL"
$computers = Get-ADComputer -Filter * -SearchBase $searchBase
$computers.count # Returns 15
```

To exclude the computers that are in Stock, we can use the SearchScope parameter. This allows us to limit the searchbase only to the current level:

```
$searchBase = "OU=Computers,OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=NL"
$computers = Get-ADComputer -Filter * -SearchBase $searchBase -SearchScope OneLevel
$computers.count # Returns 12
```

Using the Filter parameter

The Get-ADComputer cmdlet is also a great way to find one or multiple computers in your AD. Although the computer object doesn't contain a lot of information, we can still use a couple of the properties to filter on.

Let's take a look at a couple of commonly used examples to find computers:

We can find a computer based on a part of the computer name with the -like filter:

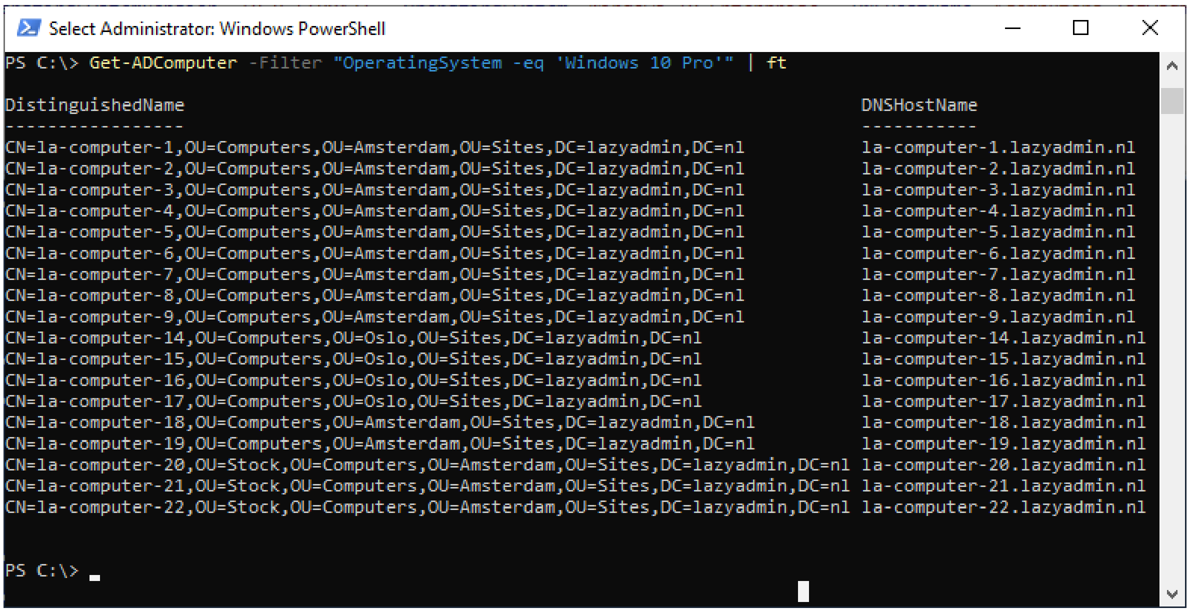
```
Get-ADComputer -Filter "Name -like '*workstation*'" | ft
```

Note the wildcard, this means that there can be something in front and after the word "workstation".

Get Computers based on OS

The Active Directory also keeps track of all operating systems on the computers. So we can also get all computers that are running Windows 10 Pro for example:

```
Get-ADComputer -Filter "OperatingSystem -eq 'Windows 10 Pro'" | ft
```



```
PS C:\> Get-ADComputer -Filter "OperatingSystem -eq 'Windows 10 Pro'" | ft

DistinguishedName                                                  DNSHostName
-----------------                                                  -----------
CN=la-computer-1,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl          la-computer-1.lazyadmin.nl
CN=la-computer-2,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl          la-computer-2.lazyadmin.nl
CN=la-computer-3,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl          la-computer-3.lazyadmin.nl
CN=la-computer-4,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl          la-computer-4.lazyadmin.nl
CN=la-computer-5,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl          la-computer-5.lazyadmin.nl
CN=la-computer-6,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl          la-computer-6.lazyadmin.nl
CN=la-computer-7,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl          la-computer-7.lazyadmin.nl
CN=la-computer-8,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl          la-computer-8.lazyadmin.nl
CN=la-computer-9,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl          la-computer-9.lazyadmin.nl
CN=la-computer-14,OU=Computers,OU=Oslo,OU=Sites,DC=lazyadmin,DC=nl              la-computer-14.lazyadmin.nl
CN=la-computer-15,OU=Computers,OU=Oslo,OU=Sites,DC=lazyadmin,DC=nl              la-computer-15.lazyadmin.nl
CN=la-computer-16,OU=Computers,OU=Oslo,OU=Sites,DC=lazyadmin,DC=nl              la-computer-16.lazyadmin.nl
CN=la-computer-17,OU=Computers,OU=Oslo,OU=Sites,DC=lazyadmin,DC=nl              la-computer-17.lazyadmin.nl
CN=la-computer-18,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl         la-computer-18.lazyadmin.nl
CN=la-computer-19,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl         la-computer-19.lazyadmin.nl
CN=la-computer-20,OU=Stock,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl la-computer-20.lazyadmin.nl
CN=la-computer-21,OU=Stock,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl la-computer-21.lazyadmin.nl
CN=la-computer-22,OU=Stock,OU=Computers,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=nl la-computer-22.lazyadmin.nl


PS C:\> _
```

Get all computers based on Operating System

We can also use multiple conditions, for example, to get all computers that running on Windows 10 or 11:

Get-ADComputer -Filter "OperatingSystem -eq 'Windows 10 Pro' -or OperatingSystem -eq 'Windows 11 Pro'" | ft

Get InActive Computers

Want to know how many inactive computers you have? We can filter the AD computers on the lastlogondate to extract all computers that are not used for the last 90 days for example.

For this, we first need to create a date variable, by taking the date from today and subtracting 90 days from it. We can then filter the AD Computers on the lastlogondate:

```
$date = (Get-Date) - (New-TimeSpan -Days 90)
Get-ADcomputer -Filter 'lastLogondate -lt $date' | ft
# Select the canonicalName,lastlogondate and name for a more readable list
Get-ADcomputer -Filter 'lastLogondate -lt $date' -properties canonicalName,lastlogondate | select
name,canonicalname,lastlogondate | ft -AutoSize
```

Get all Disabled Computers

Another useful example is to get all disabled computers from the OU:

```
# Get all disabled computers
Get-ADComputer -Filter "Enabled -eq 'False'" | ft
# Get only enabled computers
Get-ADComputer -Filter "Enabled -eq 'True'" | ft
```

Get ADComputer Properties

As mentioned earlier in the article, the computer object doesn't contain a lot of useful properties (compared to a user object). But there is still some useful information that can be extracted from the object.

Without specifying the properties, you only get computer name and distinguishedname related information. But other properties that are useful are for example:

- BadLogonCount
- BadPwdCount
- IPv4Address
- Enabled

- LastLogOff
- LastLogonDate
- LogonCount
- OperatingSystem
- OperatingSystemVersion
- WhenCreated

To get this information we can use the -properties parameter:

```
Get-ADComputer -identity la-computer-22 -Properties
IPv4Address,LastLogonDate,OperatingSystem,OperatingSystemVersion,WhenCreated
```

Export AD Computer 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-ADComputer cmdlet.

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

```
Get-ADComputer -filter * | Export-CSV c:\temp\computers.csv -NoTypeInformation
```

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

Select the properties that we need

So the first step is to specify the fields that we really want to export. For example, if we want to export the name, canonicalname, operatingsystem, and LastLogonDate we could use the following command:

```
Get-ADComputer -filter * -properties canonicalname,operatingsystem,LastLogonDate | select
name,canonicalname,operatingsystem,LastLogonDate | Export-CSV c:\temp\computers.csv -
NoTypeInformation
```

Export only Enabled Computers

If you want to export only enable computers, you can add a filter to the cmdlet:

```
Get-ADComputer -filter "Enabled -eq 'true'" -properties canonicalname,operatingsystem,LastLogonDate | select
name,canonicalname,operatingsystem,LastLogonDate | Export-CSV c:\temp\computers.csv -
NoTypeInformation
```

Complete Export AD Computers to CSV Script

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

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

- **Specify the searchBase** (OU), default whole Active Directory
- **Get enabled or disabled computers 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-ADComputers.ps1 -searchBase
"OU=computers,OU=Amsterdam,DC=LazyAdmin,DC=Local","OU=computers,OU=Oslo,DC=LazyAdmin,DC=Local"
-CSVpath c:\temp\computers.csv
```

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

1. **Download** the complete Export AD Computers script from my Github
2. **Open PowerShell** and navigate to the script
3. **Run the export script**: Get-ADComputers.ps1

When complete, the script will automatically open Excel for you.

```
<#
.SYNOPSIS
Get all AD Computers with properties and export to CSV
.NOTES
Version: 1.2
Author: R. Mens
Creation Date: 24 may 2022
Purpose/Change: Fix enabled/disable filter
#>
param(
[Parameter(
Mandatory = $false,
HelpMessage = "Enter the searchbase between quotes or multiple separated with a comma"
)]
[string[]]$searchBase,
[Parameter(
Mandatory = $false,
HelpMessage = "Get computers that are enabled, disabled or both"
)]
[ValidateSet("true", "false", "both")]
[string]$enabled = "true",
[Parameter(
Mandatory = $false,
HelpMessage = "Enter path to save the CSV file"
)]
[string]$CSVpath
)
Function Get-Computers {
<#
.SYNOPSIS
Get computers from the requested DN
#>
param(
[Parameter(
Mandatory = $true
)]
$dn
)
process{
# Set the properties to retrieve
$properties = @(
'Name',
'CanonicalName',
'OperatingSystem',
'OperatingSystemVersion',
'LastLogonDate',
'LogonCount',
'BadLogonCount',
'IPv4Address',
'Enabled',
'whenCreated'
)
# Get enabled, disabled or both computers
switch ($enabled)
{
"true" {$filter = "enabled -eq 'true'"}
"false" {$filter = "enabled -eq 'false'"}
"both" {$filter = "*"}
}
# Get the computers
Get-ADComputer -Filter $filter -searchBase $dn -Properties $properties | Select-Object $properties
}
}
Function Get-AllADComputers {
<#
.SYNOPSIS
Get all AD computers
#>
process {
Write-Host "Collecting computers" -ForegroundColor Cyan
$computers = @()
# Collect computers
if ($searchBase) {
# Get the requested mailboxes
foreach ($dn in $searchBase) {
Write-Host "- Get computers in $dn" -ForegroundColor Cyan
$computers += Get-Computers -dn $dn
}
}else{
# Get distinguishedName of the domain
$dn = Get-ADDomain | Select-Object -ExpandProperty DistinguishedName
Write-Host "- Get computers in $dn" -ForegroundColor Cyan
$computers += Get-Computers -dn $dn
}
# Loop through all computers
$computers | ForEach-Object {
[pscustomobject]@{
"Name" = $_.Name
"CanonicalName" = $_.CanonicalName
"OS" = $_.OperatingSystem
"OS Version" = $_.OperatingSystemVersion
"Last Logon" = $_.lastLogonDate
"Logon Count" = $_.logonCount
"Bad Logon Count" = $_.BadLogonCount
"IP Address" = $_.IPv4Address
"Mobile" = $_.mobile
"Enabled" = if ($_.Enabled) {"enabled"} else {"disabled"}
"Date created" = $_.whenCreated
}
}
}
}
If ($CSVpath) {
# Get all AD Computers
Get-AllADComputers | 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 $CSVpath
}else{
Write-Host "Failed to create report" -ForegroundColor Red
}
}
Else {
Get-AllADComputers | Sort-Object Name
}
```

Wrapping Up

The Get ADComputer cmdlet is really useful when it comes to exacting computers out of the Active Directory. Using the searchbase parameter allows you to quickly select the computer that you need from the specified OU.

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

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

Did you **Liked** this **Article**?
Get the latest articles like this **in your mailbox**
or share this article

I hate spam to, so you can unsubscribe at any time.