

How to use Get-ADGroup in PowerShell

 lazyadmin.nl/powershell/get-adgroup

September 6, 2022

Do you need to get all the groups in your Active Directory or just need to find the location of the one group that is hiding somewhere in an OU? In PowerShell, we can use the `Get-ADGroup` cmdlet to quickly extract all information about our groups from the AD.

Groups in your Active Directory can really help you with keeping your AD organized. It allows you to assign permissions or licenses to multiple users that are members of a single group. When for example management needs access to PowerBi, you will only have to [assign the license](#) to the PowerBi Group.

In this article, we are going to take a look at how to use the `Get-ADGroup` cmdlet in PowerShell. At the end of the article, you will also find a complete script that exports all the groups in your Active Directory to CSV.

Requirements

Before we can use the `Get-ADGroup` cmdlet, you will need to have the Active Directory module installed in PowerShell. It's installed by default on the domain controller, but on Windows 10 or 11 you will need to install it.

Run the following command in PowerShell to install the module:

```
Add-WindowsCapability -online -Name "Rsat.ActiveDirectory.DS-LDS.Tools~~~~0.0.1.0"
```

Finding Groups with Get ADGroup in PowerShell

`Get-ADGroup` cmdlet allows us to find all group objects in the Active Directory and extract information from them. The advantage of this cmdlet is that we can use different parameters to find the groups in our AD.

We can use the following parameters when it comes to finding the groups:

- **Identity** – Find a group based on the group name. This will return only a single group
- **Filter** – Retrieve multiple groups based on a filter query
- **LDAPFilter** – Use a LDAP query string to filter the group 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 most common way to get a group is by using the identity parameter. But for this, you will need to know the name of the group. It will return a single group with the most important properties:

```
Get-ADGroup -identity SG_M365_E5
```

```
Administrator: Windows PowerShell
PS C:\> Get-ADGroup -Identity SG_M365_E5

DistinguishedName : CN=SG_M365_E5,OU=Groups,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=n1
GroupCategory      : Security
GroupScope         : Global
Name               : SG_M365_E5
ObjectClass        : group
ObjectGUID         : 656554fe-2e6c-4c30-b306-74032a2e4e9d
SamAccountName     : SG_M365_E5
SID                : S-1-5-21-3785559357-502964226-1943567534-1307

PS C:\> _
```

Get-ADGroup

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

```
Get-ADGroup -identity SG_M365_E5 -properties *
```

Using the Filter Parameter

When you are searching for a particular group and you don't know the exact name, then you can use the filter parameter. This allows us to search through all groups based on a part of the name or other property.

The filter parameter can also be used to retrieve multiple groups or all groups from the Active Directory.

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

To find a group based on a part of the name you can use the **-like** filter:

```
Get-ADGroup -Filter "Name -like 'SG_*'" | ft
```

This will return all groups where the name starts with SG_.

```
Administrator: Windows PowerShell
PS C:\> Get-ADGroup -Filter "Name -like 'SG_*'" | ft

DistinguishedName                                     GroupCategory GroupScope Name
-----
CN=SG_GPO_WindowsHello,OU=Groups,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=n1 Security Global SG_GPO_WindowsHello
CN=SG_M365_E3,OU=Groups,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=n1 Security Global SG_M365_E3
CN=SG_M365_E5,OU=Groups,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=n1 Security Global SG_M365_E5
CN=SG_M365_PowerBi,OU=Groups,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=n1 Security Global SG_M365_PowerBi

PS C:\> _
```

Get-ADGroup Filter

To get all security groups we can filter the groups on the Group Category value:

```
Get-ADGroup -Filter "GroupCategory -eq 'Security'" | ft
```

If you run the command above, you will notice that it also returns all the built-in groups. Most of the time you don't need these in your exports. So let's filter those out:

```
Get-ADGroup -Filter "GroupCategory -eq 'Security' -and GroupScope -ne 'Domainlocal'" | ft
```

Now you are still left with some built-in groups. These groups are located in the default Users OU container in your Active Directory. There are two options to filter those out as well, you can specify the search base, see the next chapter, or filter out all the results where the **DistinguishedName** ends with **OU=Users,DC=Domain,DC=local**.

Replace DC=Domain,DC=Local with your AD domain name

```
Get-ADGroup -Filter "GroupCategory -eq 'Security' -and GroupScope -ne 'Domainlocal'" | Where-Object { $_.DistinguishedName -notlike "*CN=user,DC=Domain,DC=local" } | ft
```

You can also use the following cmdlet to get the DN path of your domain:

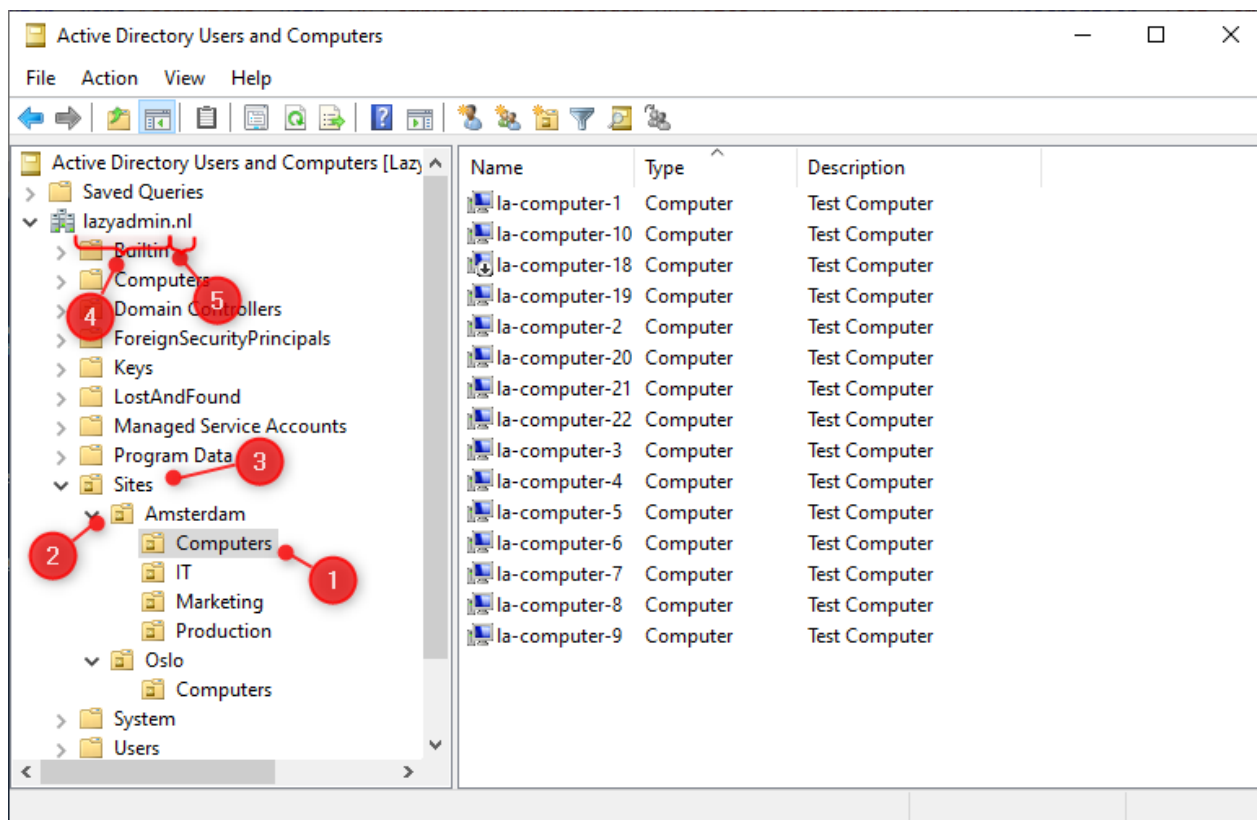
```
Get-ADDomain | Select -ExpandProperty DistinguishedName
```

Get ADGroup SearchBase

When you want to retrieve multiple groups from your Active Directory you might want to narrow down the search. As mentioned in the previous chapter, when you list all groups, all built-in groups are listed as well. Most of the time you have your groups organized in a separate OU, so we can use the SearchBase (distinguishedName) parameter to specify the OU 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:



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"

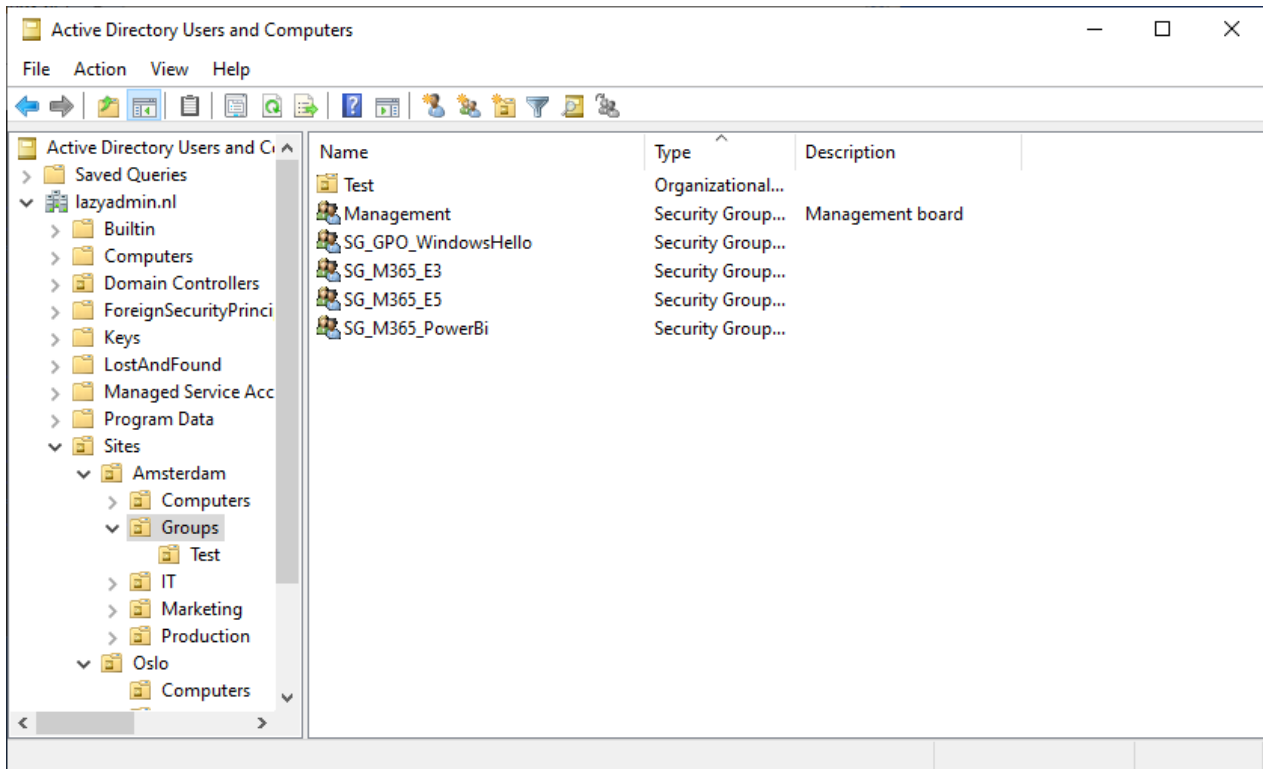
To get for example all groups from the OU Amsterdam we can use the following SearchBase path:

```
Get-ADGroup -Filter * -SearchBase "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.

For example, we want to get all groups from the Amsterdam site, except the test groups:



Active Directory Groups

To get all the groups from Amsterdam, except the groups in the sub OU "test", we can limit the **searchBase** to only the current level, using the **searchScope** parameter:

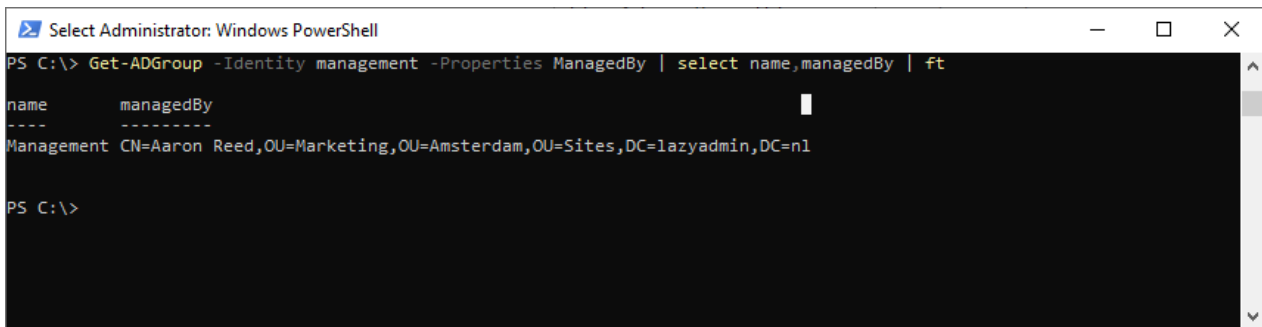
```
$searchBase = "OU=Groups,OU=Amsterdam,OU=Sites,DC=Lazyadmin,DC=NL"  
Get-ADGroup -Filter * -SearchBase $searchBase -SearchScope OneLevel
```

Get Group Manager

Active Directory groups can be managed by users. This way a user can add or remove members of the group, which is really useful for distribution groups or when you have many mutations in a group.

To get the manager of a group we can use the Get-ADGroup cmdlet and the property **managedBy**:

```
# You can also use a filter or searchbase to get the manager of multiple groups  
Get-ADGroup -Identity management -Properties managedby | select name, managedBy | ft
```



```
PS C:\> Get-ADGroup -Identity management -Properties ManagedBy | select name,managedBy | ft

name      managedBy
----      -
Management CN=Aaron Reed,OU=Marketing,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=n1

PS C:\>
```

Get ManagedBy property

As you can see in the screenshot above, the managedBy property returns the distinguished name of the user. If you want only the name, you can pipe the `Get-ADUser` cmdlet behind it:

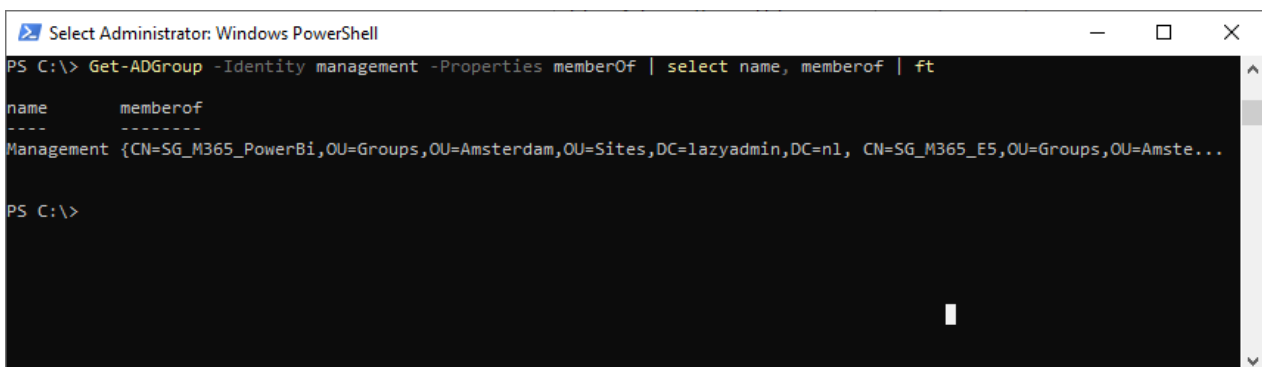
```
Get-ADGroup -Identity management -Properties ManagedBy | % {Get-ADUser -Identity $_.managedBy} | select name
```

Get Group Membership of Groups

Groups can also be member of other groups, this is particularly useful when you want to assign licenses or other permissions based on group membership. For example, all managers should have access to PowerBi. Now you can add each manager individually to the SG_M365_PowerBi group, but you can also make the group Management member of SG_M365_PowerBi.

To list all the groups that a group is a member of you could use the memberOf property:

```
Get-ADGroup -Identity management -Properties memberOf | select name,memberOf
```



```
PS C:\> Get-ADGroup -Identity management -Properties memberOf | select name, memberof | ft

name      memberof
----      -
Management {CN=SG_M365_PowerBi,OU=Groups,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=n1, CN=SG_M365_E5,OU=Groups,OU=Amste...

PS C:\>
```

But as you can see in the screenshot above, this is not really a readable name. Just like with the group manager, we will need to look up each group to get the name.

Luckily there is another option, we can use the `Get-ADPrincipalGroupMembership` cmdlet to get only the name of the group, that a group is a member of:

```
Get-ADPrincipalGroupMembership -identity management | ft
```

```
Administrator: Windows PowerShell
PS C:\> Get-ADPrincipalGroupMembership -Identity management | ft

distinguishedName                                     GroupCategory GroupScope name      objectC
-----
CN=SG_M365_E5,OU=Groups,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=n1 Security      Global SG_M365_E5      group
CN=SG_M365_PowerBi,OU=Groups,OU=Amsterdam,OU=Sites,DC=lazyadmin,DC=n1 Security      Global SG_M365_PowerBi    group

PS C:\>
```

Get ADPrincipalGroupMembership

Export all AD Groups to CSV with PowerShell

If you want to get an overview of all groups in your Active Directory then exporting to CSV is a good method. This allows you to go through all groups in Excel and list all group managers or memberships for example.

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-ADGroup cmdlet.

To simply export all AD Group objects, we can use the following command:

```
Get-ADGroup -filter * | Export-csv c:\temp\adgroups.csv -NoTypeInfo
```

This will list all groups, including the built-in with only the default properties. Most of the time not really the information you need.

Complete Export AD Groups to CSV Script

I have created a PowerShell script that will Export all AD Groups 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
- **Include or exclude built-in groups** (default exclude)
- **Export path CSV file** (default none, console output)

The script will get all the groups from the Active Directory if you don't specify the searchBase (OU). It's also possible to specify multiple OU's:

```
.\Get-ADGroups.ps1 -searchBase
```

```
"OU=groups,OU=Amsterdam,DC=LazyAdmin,DC=Local","OU=groups,OU=Oslo,DC=LazyAdmin,DC=Local"
-csvpath c:\temp\computers.csv
```

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

1. **Download** the complete Export AD Groups script from [my Github](#)
2. **Open PowerShell** and navigate to the script
3. **Run the export script:** Get-ADGroups.ps1

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

```
param(
[Parameter(
Mandatory = $false,
```

```

HelpMessage = "Enter the searchbase between quotes or multiple separated with a comma"
)]
[string[]]$searchBase,
[Parameter(
Mandatory = $false,
HelpMessage = "Include built-in groups or exclude"
)]
[ValidateSet("include", "exclude")]
[string]$builtin = "exclude",
[Parameter(
Mandatory = $false,
HelpMessage = "Enter path to save the CSV file"
)]
[string]$CSVpath
)
Function Get-Groups{
<#
.SYNOPSIS
Get groups from the requested DN
#>
param(
[Parameter(
Mandatory = $true
)]
$dn
)
process{
# Set the properties to retrieve
$properties = @(
'Name',
'CanonicalName',
'GroupCategory',
'GroupScope',
'ManagedBy',
'MemberOf',
'created',
'whenChanged',
'mail',
'info',
'description'
)
# Get all groups, or exclude the builtin groups
# Get the computers
switch ($builtin)
{
"include" {
Get-ADGroup -filter * -searchBase $dn -Properties $properties | select $properties
}
"exclude" {
$builtinUsers = "CN=users,$dn"
$filter = "GroupScope -ne 'Domainlocal'"

```

```

Get-ADGroup -filter $filter -searchBase $dn -Properties $properties | Where-Object {
$_.DistinguishedName -notlike "*, $builtinUsers" } | select $properties
}
}
}
}
Function Get-ADGroups {
<#
.SYNOPSIS
Get all AD Groups
#>
process {
Write-Host "Collecting groups" -ForegroundColor Cyan
$groups = @()
# Collect groups
if ($searchBase) {
# Get the requested groups
foreach ($dn in $searchBase) {
Write-Host "- Get groups in $dn" -ForegroundColor Cyan
$groups += Get-Groups -dn $dn
}
}else{
# Get distinguishedName of the domain
$dn = Get-ADDomain | Select -ExpandProperty DistinguishedName
Write-Host "- Get groups in $dn" -ForegroundColor Cyan
$groups += Get-Groups -dn $dn
}
# Loop through all computers
$groups | ForEach {
$managedBy = ""
$memberOf = ""
# If the group is managed, get the users name
if ($null -ne $_.ManagedBy) {
$managedBy = Get-ADUser -Identity $_.ManagedBy | select -ExpandProperty name
}
# If the group is member of other groups, get the group names
if ($_.MemberOf.count -gt 0) {
$memberOf = Get-ADPrincipalGroupMembership $_.name | select -ExpandProperty name
}
[pscustomobject]@{
"Name" = $_.Name
"CanonicalName" = $_.CanonicalName
"GroupCategory" = $_.GroupCategory
"GroupScope" = $_.GroupScope
"Mail" = $_.Mail
"Description" = $_.Description
"Info" = $_.info
"ManagedBy" = $managedBy
"MemberOf" = ($memberOf | out-string).Trim()
"Date created" = $_.created
"Date changed" = $_.whenChanged
}
}
}

```



```

}
}
}
If ($CSVpath) {
# Get mailbox status
Get-ADGroups | Export-CSV -Path $CSVpath -NoTypeInfoInformation -Encoding UTF8
if ((Get-Item $CSVpath).Length -gt 0) {
Write-Host "Report finished and saved in $CSVpath" -ForegroundColor Green
Invoke-Item $CSVpath
}
else {
Write-Host "Failed to create report" -ForegroundColor Red
}
}
Else {
Get-ADGroups
}

```

Wrapping Up

The Get-ADGroup cmdlet is great when you need to get all the groups from your Active Directory. With the help of filters and/or the searchbase parameter you can quickly select only the groups that you need.

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.