

PowerShell to Get a List of Computers in OU and Sub OU in AD

 systoolsgroup.com/updates/powershell-to-get-a-list-of-computers-in-ou

There are times when AD managers need to use PowerShell to get a list of computers in OU and sub OU of their Active Directory.

This is because computers, along with users, form the basic building blocks of all activity that goes on inside the AD. So after administrators find out what OU a user is in, the next natural question is locating the computer path. This is done to tally whether or not all resources are within the same group policy scope and perform any intra-AD computer relocation.

Moreover, if administrators are planning to migrate AD objects from one domain to another, they must have a solid idea of where every object lies, including the computers.

Not to worry, there are quite a few ways in which administrators can achieve the result. Among them, PowerShell is the most popular. Therefore, with this writeup, we provide not only the scripts but also other alternatives for finding a computer object's path.

Prerequisites to Use PowerShell List Computers in OU or Sub OU Script

Admins who wish to use PowerShell must get themselves familiar with the basic syntax and terminology.

Among the computer languages, PowerShell is on the simpler side so learning it won't be that hard. Glance through a few tutorials and check out Microsoft's official documentation; this should provide you with more than enough knowledge to understand what we are about to do here.

After the knowledge set, the most important task is to prepare your environment, Microsoft makes it easier by preinstalling PowerShell on all machines. However, if you have not used PowerShell previously, then complete the following steps on your AD workstation.

- Install the Remote Server Admin tools.
- Add the ADDS (Active Directory Domain Services Role) role.
- Upgrade the Windows Management Framework to version 3.0 or later.
- Then, use the Active Directory Administrator account.
- At last, download the AD Module for PowerShell by typing the following inside a fresh module.

```
Install-Module -Name RSAT-AD-PowerShell
```

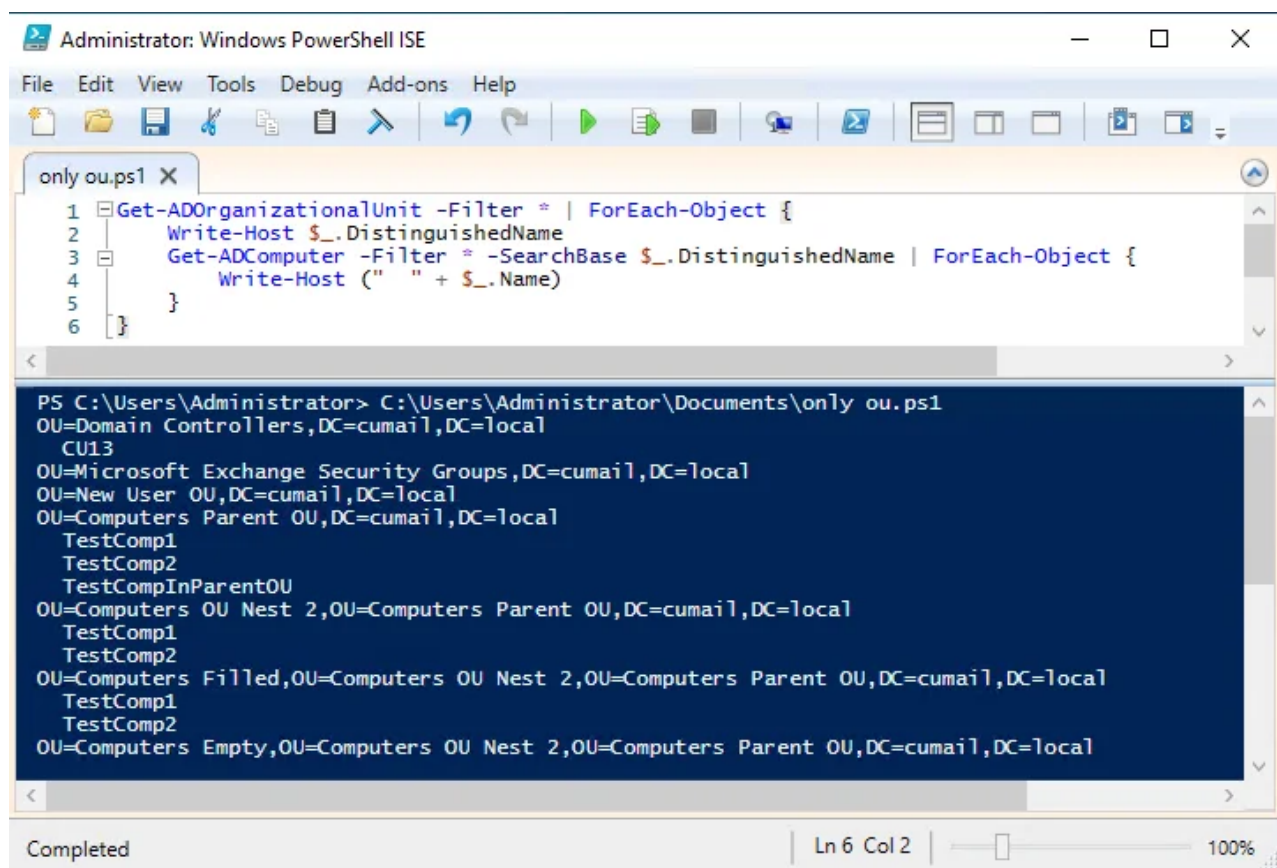
After this is done, we can start with our main objective.

Basic PowerShell to Get List of Computers in OU and Sub OU in AD

There are two ways in which administrators can make this type of report. The first is to list the OU and check what all computer objects are preset there. Another is to list down the computers one by one, along with their OU. Admins can select the PowerShell script as per their requirements.

First Scenario: Computer Objects in each OU.

```
Get-ADOrganizationalUnit -Filter * | ForEach-Object {  
Write-Host $_.DistinguishedName  
Get-ADComputer -Filter * -SearchBase $_.DistinguishedName | ForEach-Object {  
Write-Host (" " + $_.Name)  
}  
}
```



The screenshot shows the Windows PowerShell ISE interface. The script editor displays the following code:

```
1 Get-ADOrganizationalUnit -Filter * | ForEach-Object {  
2     Write-Host $_.DistinguishedName  
3     Get-ADComputer -Filter * -SearchBase $_.DistinguishedName | ForEach-Object {  
4         Write-Host (" " + $_.Name)  
5     }  
6 }
```

The console window shows the output of the script:

```
PS C:\Users\Administrator> C:\Users\Administrator\Documents\only ou.ps1  
OU=Domain Controllers,DC=cumail,DC=local  
CU13  
OU=Microsoft Exchange Security Groups,DC=cumail,DC=local  
OU=New User OU,DC=cumail,DC=local  
OU=Computers Parent OU,DC=cumail,DC=local  
    TestComp1  
    TestComp2  
    TestCompInParentOU  
OU=Computers OU Nest 2,OU=Computers Parent OU,DC=cumail,DC=local  
    TestComp1  
    TestComp2  
OU=Computers Filled,OU=Computers OU Nest 2,OU=Computers Parent OU,DC=cumail,DC=local  
    TestComp1  
    TestComp2  
OU=Computers Empty,OU=Computers OU Nest 2,OU=Computers Parent OU,DC=cumail,DC=local
```

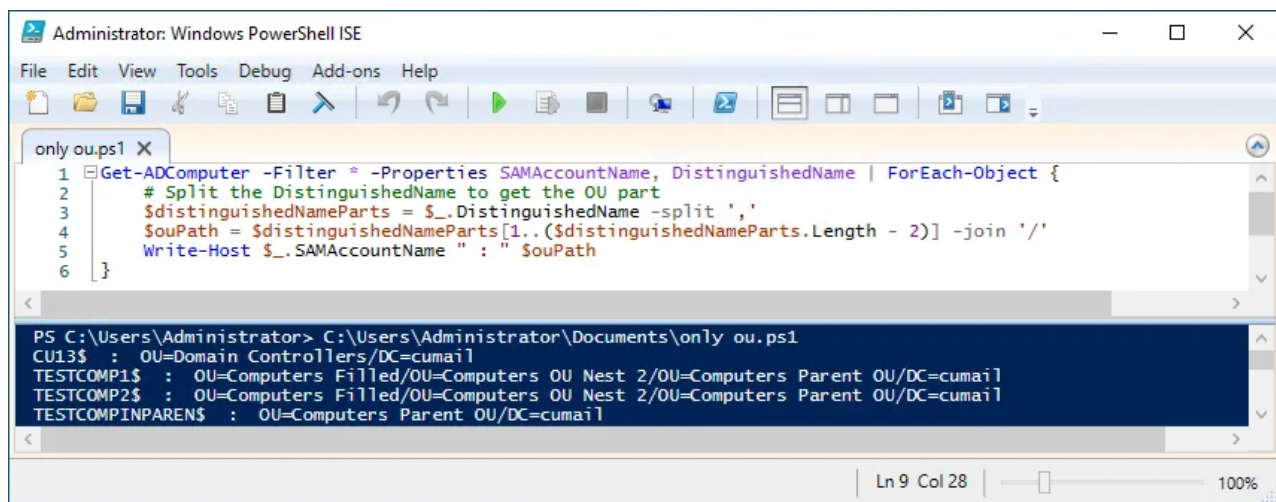
The status bar at the bottom indicates "Completed" and "Ln 6 Col 2".

- This script of PowerShell to Get a List of Computers in OU and Sub OU in AD retrieves all OUs.
- Loops through each OU.
- Displays OU's Distinguished Name.
- Retrieves computers within that OU.
- Displays indented computer names.

Go for this approach of PowerShell list computers in OU when you want a detailed view of all OUs and their associated computers. It's better for documentation purposes as it gives a complete picture of all OUs and their associated computer inventory.

Second Scenario: OU of each Computer Object.

```
Get-ADComputer -Filter * -Properties SAMAccountName, DistinguishedName | ForEach-Object {
# Split the DistinguishedName to get the OU part
$distinguishedNameParts = $_.DistinguishedName -split ','
$ouPath = $distinguishedNameParts[1..($distinguishedNameParts.Length - 2)] -join '/'
Write-Host $_.SAMAccountName " : " $ouPath
}
```



- The script loops through each computer object in AD.
- It extracts and stores the OU path from the DistinguishedName.
- Finally, it displays the computer's SAMAccountName, and its constructed OU path.

This approach is suitable for administrators who need to quickly determine the OU hierarchy for each computer. Additionally, it helps troubleshoot access control as it maps each computer to its corresponding OU hierarchy.

However, both of these scripts have some limitations.

- Neither of them checks for any nested sub-OUs in AD.
- Both give a temporary, non-shareable, read-only output.
- Admin needs to run each of the ".ps1" cmdlets manually.

So to combat this, we have a better combination of scripts lined up for you.

Find Computer OU in Active Directory with Advanced PowerShell

Part 1. PowerShell script to make a CSV report of all OUs and Sub OUs a computer is in.

```

# Import the Active Directory module
Import-Module ActiveDirectory

function Get-OUTreeWithComputers {
param (
[Parameter(Mandatory=$true)]
[string]$OUPath,
[int]$Indent = 0
)

# Get all OUs under the given path
$ous = Get-ADOrganizationalUnit -Filter * -SearchBase $OUPath -SearchScope
OneLevel

foreach ($ou in $ous) {
# Display the current OU
Write-Host (" " * $Indent + "|- " + $ou.Name)

# Get and display computers directly under the current OU
$computers = Get-ADComputer -Filter * -SearchBase $ou.DistinguishedName -
SearchScope OneLevel -Properties Name
foreach ($computer in $computers) {
Write-Host (" " * ($Indent + 3) + "|- " + $computer.Name) -ForegroundColor Cyan
}

# Recursively process sub-OUs
Get-OUTreeWithComputers -OUPath $ou.DistinguishedName -Indent ($Indent + 3)
}
}

# Get the domain
$domain = Get-ADDomain

# Start the tree from the domain root
Write-Host "Active Directory Tree Structure:"
Write-Host "|- $($domain.Name)"

# Get computers directly under the domain root
$rootComputers = Get-ADComputer -Filter * -SearchScope OneLevel -Properties Name
foreach ($computer in $rootComputers) {
Write-Host " |- $($computer.Name)" -ForegroundColor Cyan
}

# Process all OUs
Get-OUTreeWithComputers -OUPath $domain.DistinguishedName -Indent 3

Save the script as Computer-Location-Reporter.ps1 on the path C:\Temp\AD-Computer-
Reports\

```

```

1 # Import the Active Directory module
2 Import-Module ActiveDirectory
3
4 function Get-OUTreeWithComputers {
5     param (
6         [Parameter(Mandatory=$true)]
7         [string]$OUPath,
8         [int]$Indent = 0
9     )
10
11     # Get all OUs under the given path
12     $ous = Get-ADOrganizationalUnit -Filter * -SearchBase $OUPath -SearchScope OneLevel
13
14     foreach ($ou in $ous) {
15         # Display the current OU
16         Write-Host (" " * $Indent + "|- " + $ou.Name)
17
18         # Get and display computers directly under the current OU
19         $computers = Get-ADComputer -Filter * -SearchBase $ou.DistinguishedName -SearchScope OneLevel
20     }
21 }
22
23 # Run the function for the root of the domain
24 Get-OUTreeWithComputers -OUPath "C:\Users\Administrator\Documents\last-login.ps1"

```

```

PS C:\Users\Administrator> C:\Users\Administrator\Documents\last-login.ps1
Active Directory Tree Structure:
|- cumail
  |- Domain Controllers
    |- CU13
  |- Microsoft Exchange Security Groups
  |- New User OU
  |- Computers Parent OU
    |- TestCompInParentOU
    |- Computers OU Nest 2
      |- Computers Empty
      |- Computers Filled
        |- TestComp1
        |- TestComp2

```

- We import an instance of the AD module and create a custom function that recursively gets all sub-OUs of an object in the hierarchy.
- Then, we call all the computer objects using their DN and split the DN into its components.
- We filter out the OU part and use the custom function to see all that is in the overall hierarchy.
- Finally, we put the results into yet another custom object and export it into a CSV output.

Part 2. Deploy a PowerShell get all computers in OU script to set up continuous report generation via the Windows Task Scheduler app.

```
$Time=New-ScheduledTaskTrigger -Weekly -WeeksInterval 2 -DaysOfWeek Monday -At 9am
```

```
$Action=New-ScheduledTaskAction -Execute PowerShell.exe -WorkingDirectory C:/Scripts -Argument "C:\Temp\AD-Computer-Reports\Computer-Location-Reporter.ps1 -UserName <Admin> -Password <admin-pass>"
```

```
Register-ScheduledTask -TaskName "Schedule Computer Location Status Report" -Trigger $Time -Action $Action -RunLevel Highest
```

We use PowerShell to create a scheduled task trigger that starts every Monday at 9 a.m.

The task is designed to start a PowerShell module on its own, set the directory to “C:\Temp\AD-Computer-Reports\” and then invoke the Computer-Location-Reporter.ps1 script.

The Register-ScheduledTask is to save the scheduled task on the admin workstation with the name “Schedule Computer Location Status Report”.

Both of these scripts combine to form a weekly report of all computer group changes that occur in an Active Directory environment. Admins can make changes in these scripts to better suit their own requirements.

However, truth be told, PowerShell in of itself is quite complex. Limiting the complexity of the script means losing out on functionality. So a better way would be to use an automated alternative.

Avoid PowerShell Get List of Computers in OU and Sub OU in AD With a Tool

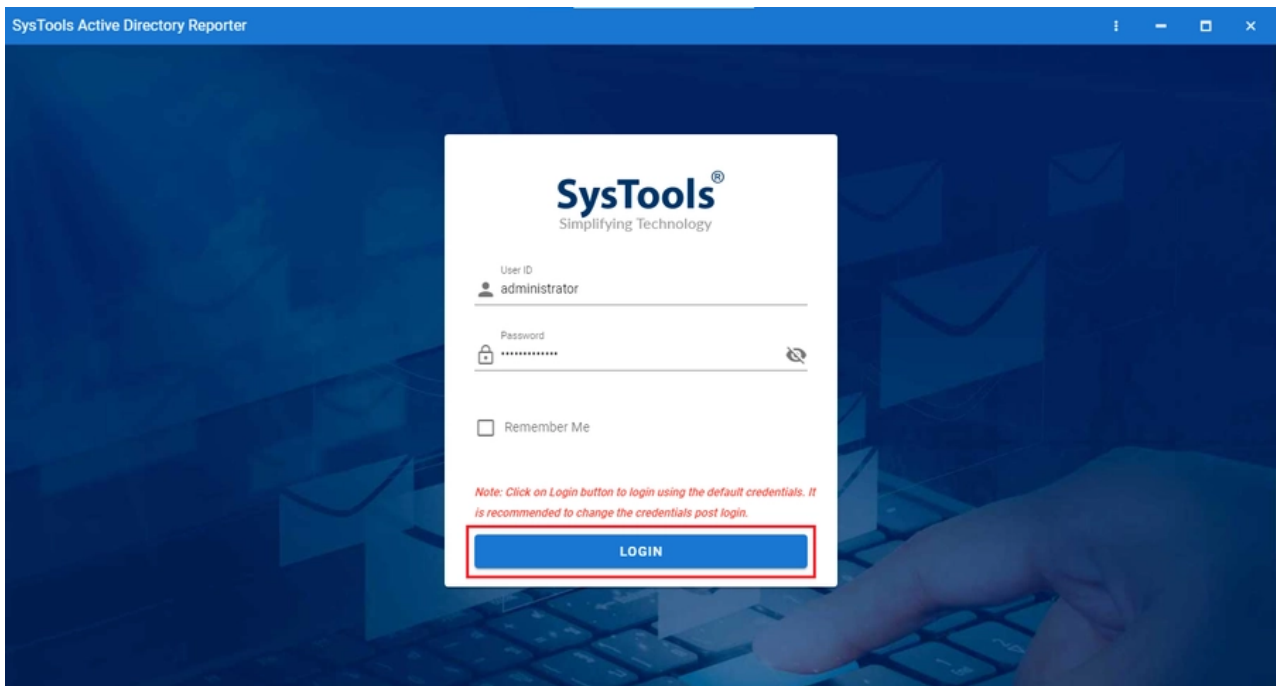
If you want to retain the ability to pull CSV reports of computer OU data, then the **SysTools AD Reporting Tool** is the best possible choice. This tool combines countless lines of PowerShell scripts into a simple GUI-based interface.

Users can apply special filters to their existing searches to check whether AD computers are disabled or not. On top of that, a duration picker allows admins to get access to historical data and it will present the information as per your AD settings.

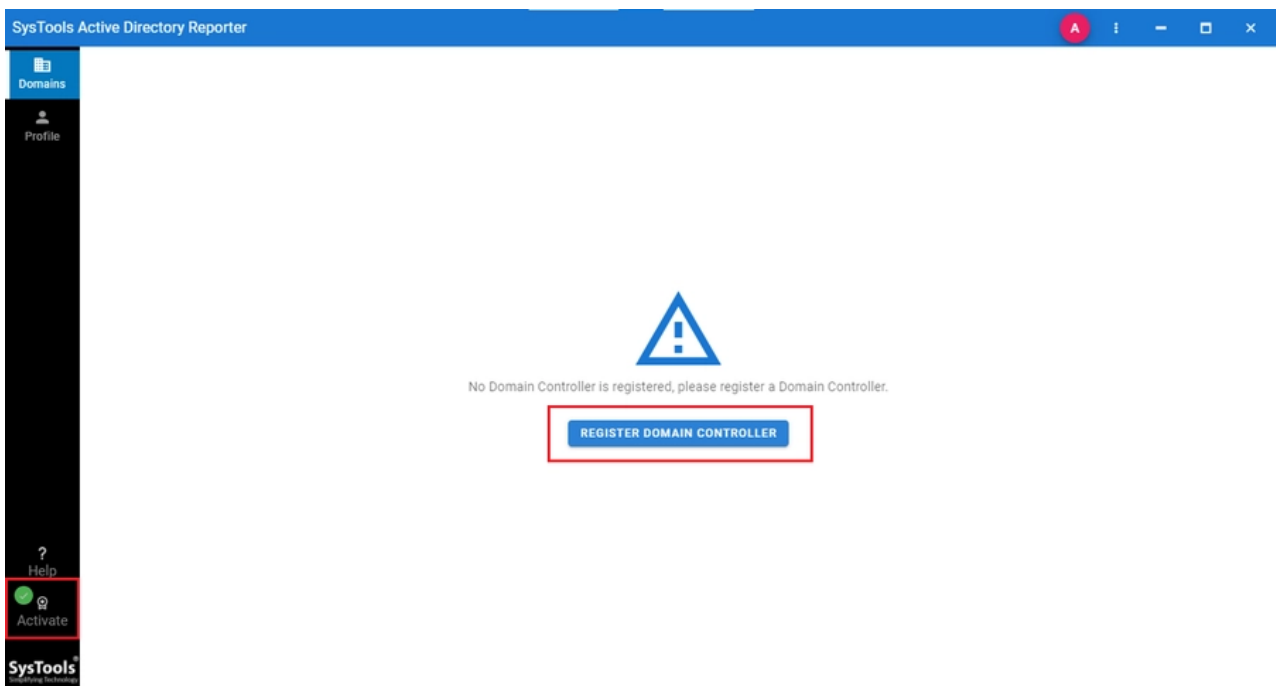
With an on-screen preview area, administrators can check out the computer OU and export the results in CSV format for further sharing. All this can be done via a remote machine that does not host the AD itself. Which is a massive improvement over the traditional PowerShell scripts.

To export computers from Active Directory OU to CSV PowerShell free with this tool, administrators have to carry out the following set of steps.

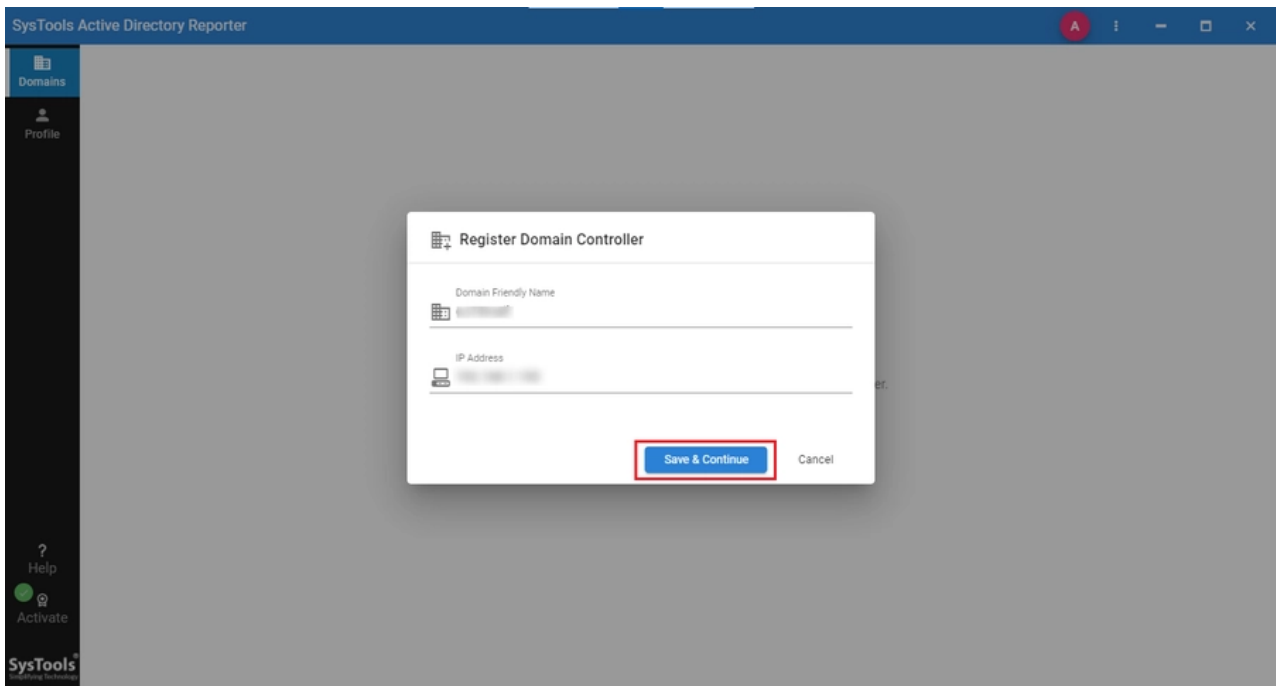
Step 1. Install, open, and log in to the tool via default administrator credentials (these fill in on their own, and you can change them later).



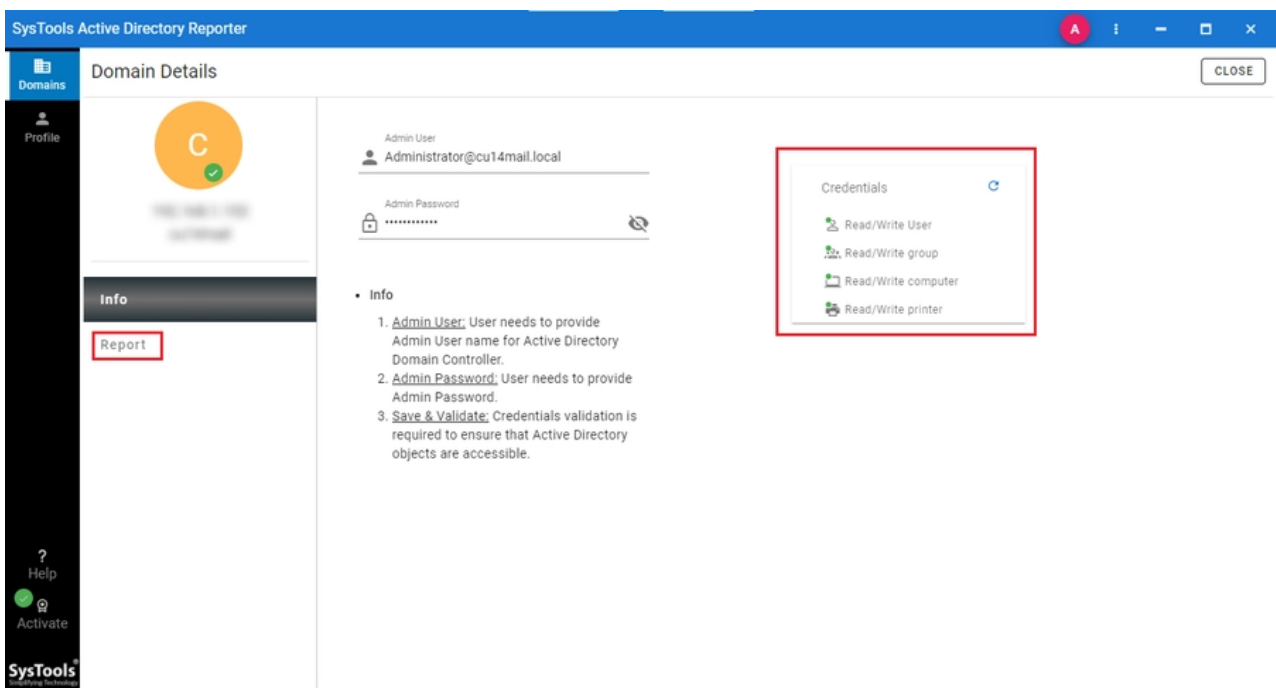
Step 2. Click on the big blue “REGISTER DOMAIN CONTROLLER” button.



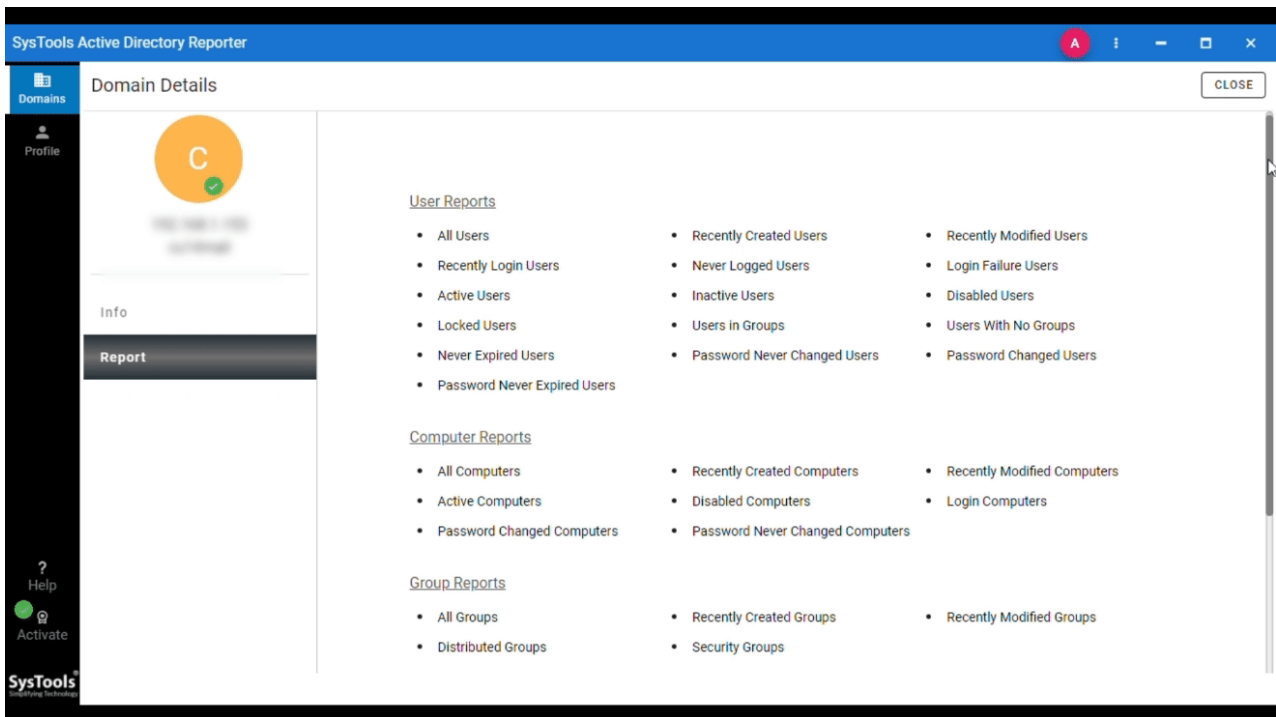
Step 3. Type the custom domain-friendly name you want to use and add the IP address of the AD.



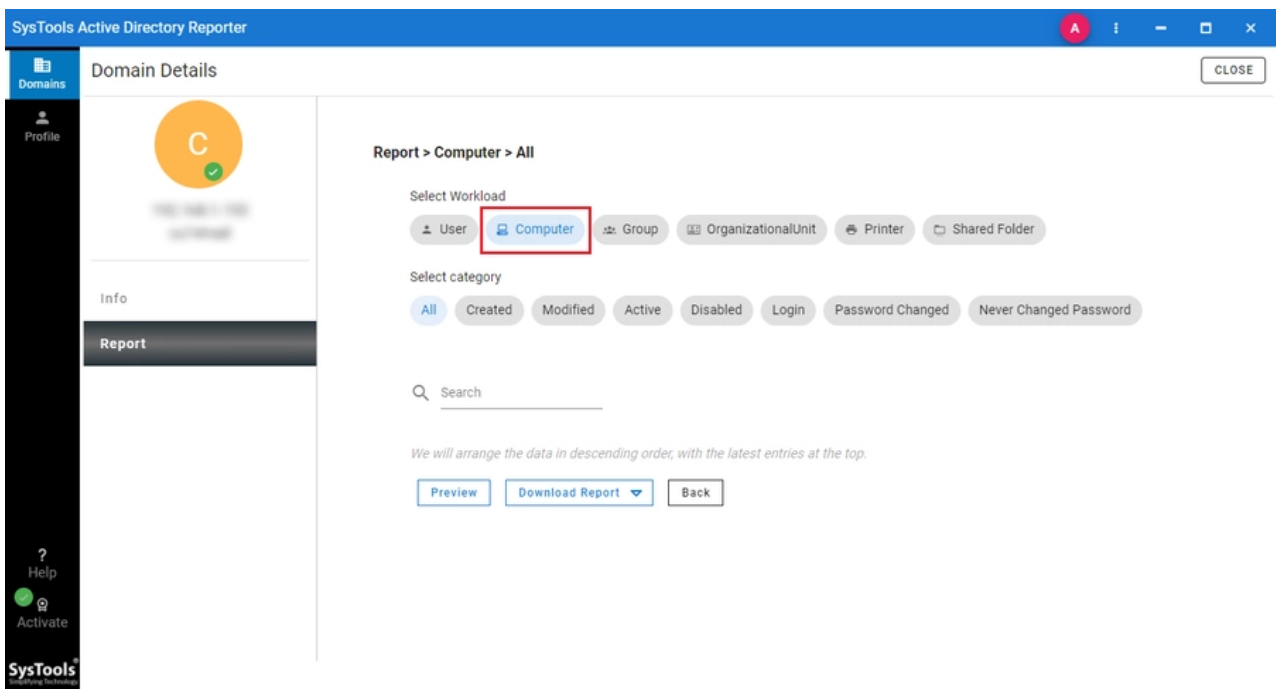
Step 4. On the domain details page, fill out the info section with appropriate data and validate.



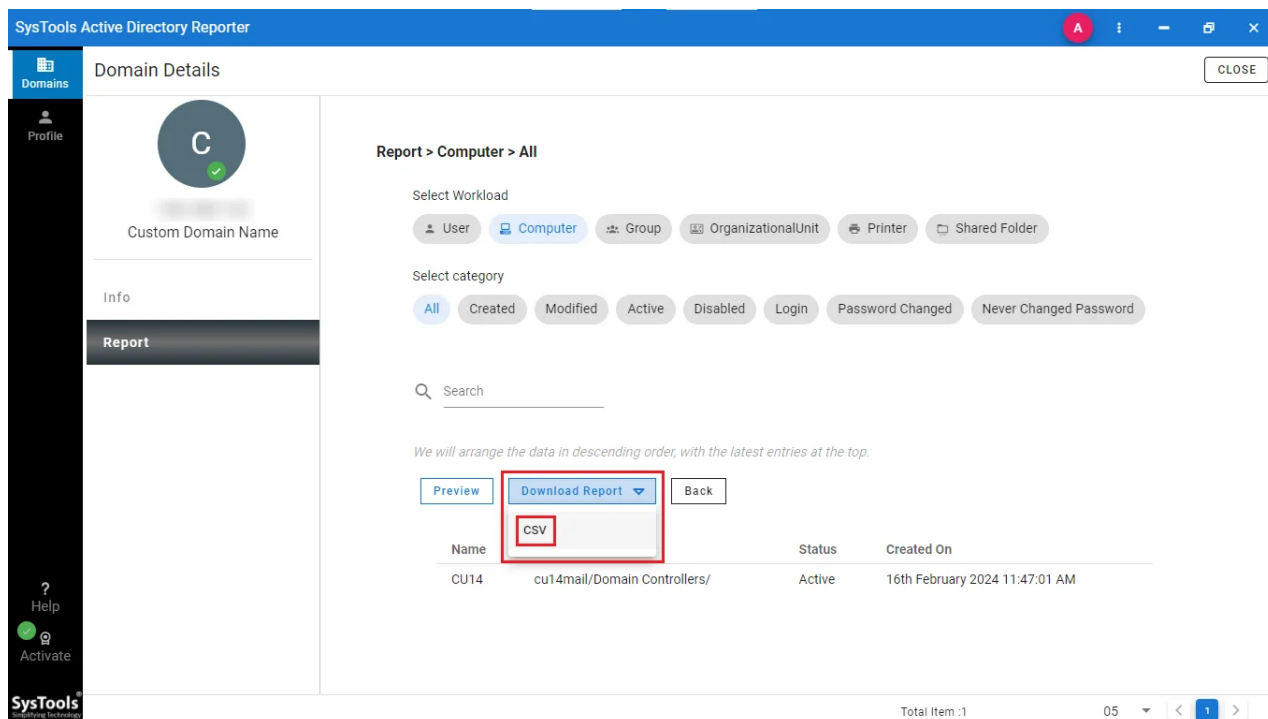
Step 5. Then, go to the Reports tab and choose the All Computers option present in the Computers category.



Step 6. Click on the Preview button to get a list of computers in OU and sub OU without PowerShell.



Step 7. Then, expand the Download option and choose CSV. Save, view, and share the results.



Conclusion

In this guide, the admin found the PowerShell to get a list of computers in OU and sub OU of an Active Directory. Here we gave a list of prerequisites that allow the PowerShell commands to function and taught how there is no single command to get the results.

Moreover, after revealing the scripts, readers themselves understood the complexity of PowerShell. Those who want a code-free computer path location can get it via the professional alternative.

Frequently Asked Questions on How to Find OU of a Computer using PowerShell

Can a computer belong to more than one OU simultaneously?

Think of OU as a physical storage container as you can't keep one computer in two boxes at the same time the same logic applies here. As the OU acts as an address from where resource access, policy applicability, and other things are controlled Microsoft designed it in a manner that any object which includes a computer stays in one OU only.

If a computer is in an OU which itself is part of a second OU then does it mean that the Computer is in two OUs? Does it violate the single-point presence of a Computer object?

No, when the computer object is not directly under an OU but instead part of its Sub OU it is considered as the object of its most direct container. Every inherited policy and permission is still applied but when the path is traces or objects are listed you will see it under Sub Ou and not the parent. Therefore some modifications must be made to the default PowerShell get-adcomputer approach.

Can a Computer exist outside of an OU in Active Directory?

Although rare in certain circumstances it is possible. Sometimes the Computer may not be part of an OU but a container instead. Moreover, OU-free computers are a resource drain on the AD and the admin should either move them to an OU or disable and delete them from the Active Directory.

How can I get AD computer data in CSV format from PowerShell?

Append [Export-Csv -Path "C:\temp\computers.csv"] at the end of the script to replicate the results in a comma-separated variable file.

By Mohit Jha

Mohit is a writer, researcher, and editor. Cyber security and digital forensics are the two subjects that keep Mohit out of his seat. In addition, he hopes that the well-researched and thought-out articles he finds will help people learn.



[View all of Mohit Jha's posts.](#)