

How to Export Active Directory Users to CSV and Build Reports

<https://adamtheautomator.com/export-active-directory-users-to-csv/>

For many Active Directory (AD) admins, retrieving users from AD was an entry point to PowerShell. PowerShell is a powerful tool for interrogating systems, and Active Directory is no exception. Searching for and returning AD users with PowerShell is just the beginning. Let's take that up a notch and export Active Directory users to CSV!

Not a reader? Watch this related video tutorial!"

Not seeing the video? Make sure your ad blocker is disabled.

In this tutorial, you will learn how to perform some basic AD queries with PowerShell and create handy reports. Using PowerShell, you will learn to format output by renaming columns, merging text fields, and performing calculations to develop valuable reports.

Manage and Report Active Directory, Exchange and Microsoft 365 with ManageEngine ADManager Plus. [Download Free Trial!](#)

Prerequisites

This tutorial will be a hands-on demonstration. If you'd like to follow along, be sure you have the following:

- Logged into an AD-joined computer with a domain user.
- PowerShell - This tutorial uses PowerShell Version 7.1.4, but any version of PowerShell should work.

- [Windows Remote System Administration Tools \(RSAT\)](#)

Getting Comfortable with the `Get-ADUser` PowerShell Cmdlet

Before creating reports, you must first figure out how to find the AD users you'd like to export Active Directory users to CSV. To do that, you'll use the `Get-ADUser` cmdlet. The `Get-ADUser` cmdlet is a PowerShell cmdlet that comes with the PowerShell ActiveDirectory module.

Open a PowerShell console and run the `Get-ADUser` cmdlet using the `Filter` parameter and argument of `*`. Using an asterisk with the `Filter` parameter tells `Get-ADUser` to return all AD users. You'll create more sophisticated filters a bit later.

```
Get-ADUser -Filter *
```

Copy

The `Get-ADUser` cmdlet returning all users

The `Get-ADUser` cmdlet returning all users

By default, the `Get-ADUser` cmdlet will return the following properties:

- `DistinguishedName` - The full LDAP name of the user object.
- `Enabled` - Is the user enabled, true or false.
- `GivenName` - The user's first name.
- `Name` - The user's full name.
- `ObjectClass` - The type of AD object this is.
- `ObjectGUID` - The ID of the AD object.
- `SamAccountName` - This was the login name up to Windows NT4.0
- `SID` - Another type of Object ID.
- `Surname` - The user's last name.
- `UserPrincipalName` - The user's login name.

In your report, you probably don't need all of these properties. By default, `Get-ADUser` also returns the built-in domain Administrator and Guest accounts. You almost certainly want to exclude those. You'll learn how in the following sections.

Limiting Searches to OUs with the SearchBase Parameter

AD users can be spread across sometimes dozens of organizational units (OUs). Sometimes, you need to limit the search to only a particular OU. To do that, you can use the `SearchBase` parameter. The `SearchBase` parameter allows you to specify a single OU as a starting point to search for users.

For example, perhaps you have an ATA-Users OU with various department OUs inside, as shown below. Inside the department OUs contains all of the user accounts you'd like to include in your export to CSV.

Example AD OU structure
Example AD OU structure

You can define the `SearchBase` argument as ATA-Users OU's distinguished name (DN) like below to limit the search to the ATA-Users OU and all OUs inside.

```
Get-ADUser -Filter * -SearchBase "OU=ATA-Users,DC=ATA,DC=local"
```

Copy

Get-ADUser Unfiltered Searchbase
Get-ADUser Unfiltered Searchbase

The output above displays many different properties for each user, but let's limit that down a bit only to show the properties you might be interested in. To do this, use the `Select-Object` cmdlet only to return the `Name` and `UserPrincipalName` properties.

Related: [Back to Basics: Understanding PowerShell Objects](#)

```
Get-ADUser -Filter * -SearchBase "OU=ATA-Users,DC=ATA,DC=local" | select Name,UserPrincipalName
```

Copy

Get-ADUser Searchbase2
Get-ADUser Searchbase2

Perhaps you'd like to only export Active Directory users to CSV in the Sales OU. To do that, specify the `Sales` OU in the `SearchBase` parameter like below.

```
Get-ADUser -Filter * -SearchBase "OU=Sales,OU=ATA-Users,DC=ATA,DC=local" | select  
Name,UserPrincipalName
```

Copy

```
Get-ADUser Unfiltered Searchbase3  
Get-ADUser Unfiltered Searchbase3
```

Filtering AD User Accounts from Get-ADUser

Up to this point, you have ignored the `Filter` parameter by simply specifying an asterisk to return all users. But if you need to query only certain users matching specific criteria, the `Filter` parameter is your friend.

Let's say you'd like to eventually export all Active Directory users to a CSV inside of the ATA-Users OU, but only if they have their `Department` AD attribute set to `Sales` like the example user account below.

```
An AD user account with Sales as a Department attribute  
An AD user account with Sales as a Department attribute
```

Using the `Filter` parameter on `Get-ADUser`, specify the AD attribute (`Department`), the operator `-eq` equating to "equal to" and the value of the `Department` attribute (`Sales`).

Related: [Understanding PowerShell Comparison Operators By Example](#)

```
Get-ADUser -Filter {Department -eq "Sales"} -SearchBase "OU=ATA-Users,DC=ATA,DC=local"| select  
Name,UserPrincipalName
```

Copy

If you have users inside the ATA-Users OU with the `Department` attribute set to `Sales`, `Get-ADUser` will only return those users.

```
Get-ADUser only returning Sales users  
Get-ADUser only returning Sales users
```

Maybe you'd like to include the `Department` attribute in the output. To do that, you'd typically specify the `Department` property as another property to show via the `Select-Object` (`select`) cmdlet, as shown below. But notice the `Department` property doesn't show up.

Including Department Attribute Including Department Attribute

By default, the `Get-ADUser` cmdlet does not return all properties. To return all non-default properties, you must use the `Properties` parameter. In this case, tell `Get-ADUser` to return the `Department` property.

```
Get-ADUser -Filter {Department -eq "Sales"} -SearchBase "OU=ATA-Users,DC=ATA,DC=local" -Properties Department | select Name,UserPrincipalName
```

Copy

Now that you have a basic filter, you can continue to add more criteria to the `Filter` as necessary, combining them with the PowerShell `and` and `or` operators. Below, for example, `Get-ADUser` will return all AD users that are enabled that are either in the `Sales` or `Finance` departments.

Adding Criteria to the Filter Adding Criteria to the Filter

In the tutorial's environment, Steve James is an account in the `Sales` department, but his account is not enabled, so his account will not show up via the command above.

Account not Enabled Account not Enabled

Exporting Active Directory Users to CSV

You now have the foundational knowledge to retrieve AD users with PowerShell. The final step is to export those Active Directory users to a CSV file to create a report you can share.

Related: [What is a CSV File, How to Create, Open and Work with Them](#)

Let's say you've built your `Get-ADUser` command, and it's returning the users you'd like to include in your CSV report like below.

This command:

1. Retrieves all AD users in the ATA-Users OU and all child OUs.
2. Outputs extra properties like `Department`, `PasswordLastSet`, and `PasswordNeverExpires`.

3. Limits the properties returned via `Select-Object` to include in the report like `Name`, `UserPrincipalName`, `Department`, and any property that begins with `Password`.

Notice `password` in this example. Using an asterisk with `Select-Object` tells `Select-Object` to return all properties that start with `password`.*

```
Get-ADUser -Filter * -SearchBase "OU=ATA-Users,DC=ATA,DC=local" -properties
Department,PasswordLastSet,PasswordNeverExpires | Select-Object
Name,UserPrincipalName,Department,password*
```

Copy

To export the Active Directory users, this command returns to CSV, pipe the objects to the `Export-Csv` cmdlet. The `Export-Csv` cmdlet is a PowerShell cmdlet that allows you to send various objects to (AD user accounts in this example) and then append those objects as CSV rows.

Related: [Export-Csv: Converting Objects to CSV Files](#)

To export each AD user returned in the command above, append `| Export-Csv <csv file name>.csv` to the end. This action pipes all of the objects that `Select-Object` returns and “converts” them into a CSV file.

```
Get-ADUser -Filter * -SearchBase "OU=ATA-Users,DC=ATA,DC=local" -properties
Department,PasswordLastSet,PasswordNeverExpires | Select-Object
Name,UserPrincipalName,Department,password* | Export-CSV pass_report.csv
```

Copy

You’ll see below that `Export-Csv` creates a CSV file called `pass_report.csv` that includes headers as object property names and one row per AD user account.

Example output from Export-CSV
Example output from Export-CSV

Customizing CSV Headers with `Select-Object`

The report you can now generate contains all the required information, but the CSV headers are not grammatically correct and can be misleading. A manager may not know what a

`UserPrincipalName` is, and having column headings with multiple words without spaces is good English.

To export the Active Directory users to CSV and create custom CSV headers, use the `Select-Object` cmdlet's [calculated properties](#). The calculated properties feature is a way you can define custom property names and values.

The `Select-Object` cmdlet's calculated properties feature requires you to define a hashtable with two key/value pairs; `Name` to indicate the name of the property and `Expression` to represent the code to manipulate the original object property value or simply the actual property name.

In this example, let's say you'd like the CSV to show a header name of:

- `Login Name` instead of `UserPrincipalName`
- `Password Last Set Date` instead of `PasswordLastSet`
- `Password Never Expires` instead of `PasswordNeverExpires`
- `Password Last Set Date` instead of `PasswordLastSet` that's represented with a [short date](#).

To make these changes, you'd first build a hashtable for each property like below.

```
@{Name="Login Name";Expression="UserPrincipalName"}
@{Name="Password Last Set Date";Expression="PasswordLastSet"}
@{Name="Password Never Expires";Expression="PasswordNeverExpires"}
@{Name="Password Last Set Date";Expression={$_.PasswordLastSet.ToShortDateString()}}
```

Copy

Now that you have the hashtables add them to the list of properties you provide to the `Select-Object` cmdlet just like you would a typical property name.

The `Select-Object` cmdlet's `Property` parameter accepts an array. If you have many properties to pass, you can create an array first and then pass that array to the `Property` parameter for easier readability.

```
$properties = @(
    Name,
    @{Name="Login Name";Expression="UserPrincipalName"},
    Department,
    @{Name="Password Last Set Date";Expression="PasswordLastSet"},
    @{Name="Password Never Expires";Expression="PasswordNeverExpires"},
    @{Name="Password Last Set Date";Expression={$_.PasswordLastSet.ToShortDateString()}}
)
```

```
Select-Object -Property $properties
```

Copy

Combining `Get-ADUser` with the new `Select-Object` construct created above gives you the below code snippet.

```
$properties = @(
    Name,
    @({Name="Login Name";Expression="UserPrincipalName"}),
    Department,
    @({Name="Password Last Set Date";Expression="PasswordLastSet"}),
    @({Name="Password Never Expires";Expression="PasswordNeverExpires"}),
    @({Name="Password Last Set Date";Expression={$_.PasswordLastSet.ToShortDateString()}}
)
Get-ADUser -Filter * -SearchBase "OU=ATA-Users,DC=ATA,DC=local" -Properties
Department,PasswordLastSet,PasswordNeverExpires | Select-Object -Property $properties | Export-CSV
pass_report.csv
```

Copy

Once complete, PowerShell will create a CSV file for you that looks like the example below.

CSV export of AD users using calculated properties
CSV export of AD users using calculated properties

Manage and Report Active Directory, Exchange and Microsoft 365 with
ManageEngine ADManager Plus. [Download Free Trial!](#)

Conclusion

PowerShell is a powerful tool for reporting on Active Directory Users. This tutorial showed you how to find and filter users based on various criteria and create a CSV file from that output using just a few lines of code.

Now that you have the foundational knowledge to query AD users and export Active Directory users to CSV, where do you see yourself using this knowledge in your daily work life?

Revision #1

Created 5 January 2024 05:00:59 by ColtM

Updated 7 August 2024 23:24:38 by ColtM