AD Users and Computers

Table Of Contents

Users

Get-ADUser

The Get-ADUser cmdlet gets a specified user object or performs a search to get multiple user objects.

The Identity parameter specifies the Active Directory user to get. You can identify a user by its distinguished name (DN), GUID, security identifier (SID), Security Account Manager (SAM) account name, or name. You can also set the parameter to a user object variable such as $ or pass a user object through the pipeline to the Identity parameter.

To search for and retrieve more than one user, use the Filter or LDAPFilter parameters. The Filter parameter uses the PowerShell Expression Language to write query strings for Active Directory. PowerShell Expression Language syntax provides rich type-conversion support for value types received by the Filter parameter. For more information about the Filter parameter syntax, type Get-Help about_ActiveDirectory_Filter. If you have existing Lightweight Directory Access Protocol (LDAP) query strings, you can use the LDAPFilter parameter.

This cmdlet retrieves a default set of user object properties. To retrieve additional properties use the Properties parameter. For more information about how to determine the properties for user objects, see the Properties parameter description.

```
# Get User
Get-ADUser -Identity [SamAccountName]

# Get all properties of a user
Get-ADUser -Identity [SamAccountName] -Properties *

# Gets specific properties of a user
Get-ADUser -Identity [SamAccountName] -Properties [Properties]

# Get properties for multiple users
Get-ADUser -Filter {Name -like [Part of name]*}
```

New-ADUser

The New-ADUser cmdlet creates an Active Directory user. You can set commonly used user property values by using the cmdlet parameters.

You can set property values that are not associated with cmdlet parameters by using the OtherAttributes parameter. When using this parameter, be sure to place single quotes around the attribute name.

You must specify the SamAccountName parameter to create a user.

You can use the New-ADUser cmdlet to create different types of user accounts such as iNetOrgPerson accounts. To do this in Active Directory Domain Services (AD DS), set the Type parameter to the Lightweight Directory Access Protocol (LDAP) display name for the type of account you want to create. This type can be any class in the Active Directory schema that is a subclass of user and that has an object category of person.

The Path parameter specifies the container or organizational unit (OU) for the new user. When you do not specify the Path parameter, the cmdlet creates a user object in the default container for user objects in the domain.

The following methods explain different ways to create an object by using this cmdlet.

```
# Get full syntax
Get-Command New-ADUser -Syntax

# Create an enabled user with password input on the CLI
New-ADUser -Name [Name] -GivenName [First name] -Surname [Last name] -
```

```
SamAccountName [Username] -UserPrincipalName [Principal name] -Path [Ou Path] -
AccountPassword(Read-Host -AsSecureString "What is the new password?") -Enabled
$true

# Create multiple users from a CSV file
Import-Csv [File path].csv | ForEach-Object {
    $upn = $_."SamAccountName" + [domain]
    New-ADUser -Name $_."Name" -GivenName $_."GivenName" -Surname $_."Surname" -
SamAccountName $_."SamAccountName" -UserPrincipalName $upn -Path $_"Path" -
AccountPassword (ConvertTo-SecureString [Password] -AsPlainText -force) -Enabled
$true
}

# Creates a disabled user with minimal details
New-ADUser -Name "Auston Matthews" -GivenName "Auston" -Surname "Matthews" -
SamAccountName "austonMatthews" -UserPrincipalName "auston@leafs.com" -Path
"OU=Users,OU=CAN,DC=leafs,DC=com"
```

Remove-ADUser

The Remove-ADUser cmdlet removes an Active Directory user.

The Identity parameter specifies the Active Directory user to remove. You can identify a user by its distinguished name (DN), GUID, security identifier (SID), or Security Account Manager (SAM) account name. You can also set the Identity parameter to a user object variable, such as $, or you can pass a user object through the pipeline to the Identity parameter. For example, you can use the Get-ADUser cmdlet to retrieve a user object and then pass the object through the pipeline to the Remove-ADUser cmdlet.

```
# Removes a user
Remove-ADUser -Identity [SamAccountName]

# Search and Remove
Get-ADUser -Filter {Name -like "Austo"*} | Remove-ADUSer
```

Set-ADUser

The Set-ADUser cmdlet modifies the properties of an Active Directory user. You can modify commonly used property values by using the cmdlet parameters. You can set property values that are not associated with cmdlet parameters by using the Add, Remove, Replace, and Clear parameters.

The Identity parameter specifies the Active Directory user to modify. You can identify a user by its distinguished name, GUID, security identifier (SID), or Security Account Manager (SAM) account name. You can also set the Identity parameter to an object variable such as $, or you can pass an object through the pipeline to the Identity parameter. For example, you can use the Get-ADUser cmdlet to retrieve a user object and then pass the object through the pipeline to the Set-ADUser cmdlet.

The Instance parameter provides a way to update a user object by applying the changes made to a copy of the object. When you set the Instance parameter to a copy of an Active Directory user object that has been modified, the Set-ADUser cmdlet makes the same changes to the original user object. To get a copy of the object to modify, use the Get-ADUser object. The Identity parameter is not allowed when you use the Instance parameter. For more information about the Instance parameter, see the Instance parameter description.

```
# Set, Replace, and Clear properties
Set-ADUser -Identity [SamAccountName] -Remove @{otherMailbox=[Value]} -Add @{url=
[Value]} -Replace @{title=[Value]} -Clear description

# Set properties using other ID objects
$Manager = Get-ADUser -Identity [SamAccountName]
Set-ADUser -Identity [SamAccountName] -Manager $Manager

# Batch Set properties
Get-ADUser * -SearchBase 'OU=Players,OU=UserAccounts,DC=LEAFS,DC=COM' | Set-ADUser
-City [Value]

# Filter for users and set properties
Get-ADUser -Filter {Name -like [Value]*} -SearchBase [OU Base] | Set-ADUser -City
[Value]
```

Computers

Get-ADComputer

The Get-ADComputer cmdlet gets a computer or performs a search to retrieve multiple computers.

The Identity parameter specifies the Active Directory computer to retrieve. You can identify a computer by its distinguished name, GUID, security identifier (SID) or Security Accounts Manager (SAM) account name. You can also set the parameter to a computer object variable, such as $ or pass a computer object through the pipeline to the Identity parameter.

To search for and retrieve more than one computer, use the Filter or LDAPFilter parameters. The Filter parameter uses the PowerShell Expression Language to write query strings for Active Directory. PowerShell Expression Language syntax provides rich type conversion support for value types received by the Filter parameter. For more information about the Filter parameter syntax, type Get-Help about_ActiveDirectory_Filter. If you have existing Lightweight Directory Access Protocol (LDAP) query strings, you can use the LDAPFilter parameter.

This cmdlet retrieves a default set of computer object properties. To retrieve additional properties use the Properties parameter. For more information about the how to determine the properties for computer objects, see the Properties parameter description.

```
# Get Computer info
Get-ADComputer -Identity [Hostname]
```

```
# Get properties
Get-ADComputer -Identity [Hostname] -Properties *
Get-ADComputer -Identity [Hostname] -Properties Name,Description,SID

# Get properties and format into a table
Get-ADComputer -Identity [Hostname] -Properties Name,Description,SID | Format-
Table Name,Desc,ID

# Get multiple user properties
Get-ADComputer -Filter {Team -like [Value]*} -Properties Name,Description,SID |
Format-Table Name,Desc,ID
```

New-ADComputer

The New-ADComputer cmdlet creates a new Active Directory computer object. This cmdlet does not join a computer to a domain. You can set commonly used computer property values by using the cmdlet parameters. Property values that are not associated with cmdlet parameters can be modified by using the OtherAttributes parameter.

You can use this cmdlet to provision a computer account before the computer is added to the domain. These pre-created computer objects can be used with offline domain join, unsecure domain join, and RODC domain join scenarios.

The Path parameter specifies the container or organizational unit (OU) for the new computer. When you do not specify the Path parameter, the cmdlet creates a computer account in the default container for computer objects in the domain.

```
New-ADComputer -Name [Hostname] -SamAccountName [SamAccountName] -Path [OU Path]
```

Remove-ADComputer

The Remove-ADComputer cmdlet removes an Active Directory computer.

The Identity parameter specifies the Active Directory computer to remove. You can identify a computer by its distinguished name, GUID, security identifier (SID), or Security Accounts Manager (SAM) account name. You can also set the Identity parameter to a computer object variable, such as $, or you can pass a computer object through the pipeline to the Identity parameter. For example, you can use the Get-ADComputer cmdlet to retrieve a computer object and then pass the object through the pipeline to the Remove-ADComputer cmdlet.

```
Remove-ADComputer -Identity [Hostname]
```

Remove Computer Recursively

This command removes a computer and all leaf objects that are located underneath it in the directory. Note that only a few computer objects create child objects, such as servers running the Clustering service. This example can be useful for removing those objects and any child objects owned by and associated with them.

```
Get-ADComputer -Identity [Hostname] | Remove-ADObject -Recursive
```

Set-ADComputer

The Set-ADComputer cmdlet modifies the properties of an Active Directory computer object. You can modify commonly used property values by using the cmdlet parameters. Property values that are not associated with cmdlet parameters can be modified by using the Add, Replace, Clear, and Remove parameters.

The Identity parameter specifies the Active Directory computer to modify. You can identify a computer by its distinguished name, GUID, security identifier (SID) or Security Accounts Manager (SAM) account name. You can also set the Identity parameter to an object variable such as $, or you can pass an object through the pipeline to the Identity parameter. For example, you can use the Get-ADComputer cmdlet to retrieve a computer object and then pass the object through the pipeline to Set-ADComputer.

The Instance parameter provides a way to update a computer by applying the changes made to a copy of the computer object. When you set the Instance parameter to a copy of an Active Directory computer object that has been modified, the Set-ADComputer cmdlet makes the same changes to the original computer object. To get a copy of the object to modify, use the Get-ADComputer object. When you specify the Instance parameter you should not pass the Identity parameter. For more information about the Instance parameter, see the Instance parameter description.

```
# Set Property
Set-ADComputer -Identity [Hostname] -Location [Value]

# Batch set properties
Get-ADComputer * -SearchBase [OU Path] | Set-ADComputer -Description [Value]

# Search and set properties
Get-ADComputer -Filter {Name -like [Partial Name Value]*} | Set-ADComputer -
Description [Value]
```

Organizational Units

Get-ADOrganizationalUnit

The Get-ADOrganizationalUnit cmdlet gets an organizational unit (OU) object or performs a search to get multiple OUs.

The Identity parameter specifies the Active Directory OU to get. You can identify an OU by its distinguished name or GUID. You can also set the parameter to an OU object variable, such as $ or pass an OU object through the pipeline to the Identity parameter.

To search for and retrieve more than one OU, use the Filter or LDAPFilter parameters. The Filter parameter uses the PowerShell Expression Language to write query strings for Active Directory. PowerShell Expression Language syntax provides rich type conversion support for value types received by the Filter parameter. For more information about the Filter parameter syntax, type Get-Help about_ActiveDirectory_Filter. If you have existing Lightweight Directory Access Protocol (LDAP) query strings, you can use the LDAPFilter parameter.

This cmdlet gets a default set of OU object properties. To get additional properties, use the Properties parameter. For more information about the how to determine the properties for computer objects, see the Properties parameter description.

```
Get-ADOrganizationalUnit -Identity [OU Path]
```

New-ADOrganizationUnit

The New-ADOrganizationalUnit cmdlet creates an Active Directory organizational unit (OU). You can set commonly used OU property values by using the cmdlet parameters. Property values that are not associated with cmdlet parameters can be set by using the* OtherAttributes* parameter.

You must set the Name parameter to create a new OU. If you do not specify the Path parameter, the cmdlet creates an OU under the default NC head for the domain.

```
# New OU
New-ADOrganizationUnit -Name [Name Value] -Description [Value]

# Create new OU with path
New-ADOrganizationalUnit -Name [Name Value] -Path [OU Path]

# Remove OU
Remove-ADOrganizationalUnit [OU Path]
```

Set-ADOrganizationalUnit

The Set-ADOrganizationalUnit cmdlet modifies the properties of an Active Directory organizational unit (OU). You can modify commonly used property values by using the cmdlet parameters. Property values that are not associated with cmdlet parameters can be modified by using the Add, Remove, Replace, and Clear parameters.

The Identity parameter specifies the Active Directory organizational unit to modify. You can identify an organizational unit by its distinguished name or GUID.

You can also set the Identity parameter to an object variable such as $, or you can pass an object through the pipeline to the Identity parameter. For example, you can use the Get-ADOrganizationalUnit cmdlet to retrieve an organizational unit object and then pass the object through the pipeline to the Set-ADOrganizationalUnit cmdlet.

```
# Set attribute
Set-ADOrganizationalUnit -Identity [OU Path] -Description [Value]

# Set Accidental Deletion Attribute
Set-ADOrganizationalUnit -Identity [OU Path] -ProtectedFromAccidentalDeletion
$true
```

Account/Object Management

Disable-ADAccount

The Disable-ADAccount cmdlet disables an Active Directory user, computer, or service account.

The Identity parameter specifies the Active Directory user, computer service account, or other service account that you want to disable. You can identify an account by its distinguished name, GUID, security identifier (SID), or Security Accounts Manager (SAM) account name. You can also set the Identity parameter to an object variable such as $, or you can pass an account object through the pipeline to the Identity parameter. For example, you can use the Get-ADUser cmdlet to retrieve a user account object and then pass the object through the pipeline to the Disable-ADAccount cmdlet. Similarly, you can use Get-ADComputer and Search-ADAccount to retrieve account objects.

```
Disable-ADAcount -Identity [SamAccountName]
```

Enable-ADAccount

The Enable-ADAccount cmdlet enables an Active Directory user, computer, or service account.

The Identity parameter specifies the Active Directory user, computer, or service account that you want to enable. You can identify an account by its distinguished name, GUID, security identifier (SID) or Security Accounts Manager (SAM) account name. You can also set the Identity parameter to an object variable such as $, or you can pass an account object through the pipeline to the Identity parameter. For example, you can use the Get-ADUser cmdlet to retrieve an account object and then pass the object through the pipeline to the Enable-ADAccount cmdlet. Similarly, you can use Get-ADComputer and Search-ADAccount to retrieve account objects.

```
Enable-ADAccount -Identity [SamAccountName]
```

Move-ADObject

The Move-ADObject cmdlet moves an object or a container of objects from one container to another or from one domain to another.

When an object is moved between domains, both the source DC and the target DC need to be the RID Master of their domains. If a different DC is being used, you will receive the following error:

move-adobject : The requested operation could not be performed because the directory service is not the master for that type of operation

The Identity parameter specifies the Active Directory object or container to move. You can identify an object or container by its distinguished name or GUID. You can also set the Identity parameter to an object variable such as $, or you can pass an object through the pipeline to the Identity parameter. For example, you can use the Get-ADObject cmdlet to retrieve an object and then pass the object through the pipeline to the Move-ADObject cmdlet. You can also use the Get-ADGroup, Get-ADUser, Get-ADComputer, Get-ADServiceAccount, Get-ADOrganizationalUnit, and Get-ADFineGrainedPasswordPolicy cmdlets to get an object that you can pass through the pipeline to this cmdlet.

The TargetPath parameter must be specified. This parameter identifies the new location for the object or container.

If you have ProtectedFromAccidentalDeletion enabled you cannot move objects. It must be disabled first.

```
# Move single user
Get-ADUser -Identity [SamAccountName] | Move-ADObject -TargetPath [OU Path]

# Move multiple objects
Get-ADUser -Filter {Name -like [Partial Name Value]*} -SearchBase [OU Path] |
Move-ADObject -TargetPath [OU Path]
```

Search-ADAccount

The Search-ADAccount cmdlet retrieves one or more user, computer, or service accounts that meet the criteria specified by the parameters. Search criteria include account and password status. For example, you can search for all accounts that have expired by specifying the AccountExpired parameter. Similarly, you can search for all accounts with an expired password by specifying the PasswordExpired parameter. You can limit the search to user accounts by specifying the UsersOnly parameter. Similarly, when you specify the ComputersOnly parameter, the cmdlet only retrieves computer accounts.

Some search parameters, such as AccountExpiring and AccountInactive use a default time that you can change by specifying the DateTime or TimeSpan parameter. The DateTime parameter specifies a distinct time. The TimeSpan parameter specifies a time range from the current time. For example, to search for all accounts that expire in 10 days, specify the AccountExpiring and TimeSpan parameter and set the value of TimeSpan to 10.00:00:00. To search for all accounts that expire before December 31, 2012, set the DateTime parameter to 12/31/2012.

```
# Search all disabled accounts
Search-ADAccount -AccountDisabled | FT Name, ObjectClass -A

# Search only users with disabled accounts
Search-ADAccount -AccountDisabled -UsersOnly | FT Name,ObjectClass -A
```

```
# Get all accounts with expired passwords
Search-ADAccount -AccountExpired | FT Name,ObjectClass -A

# Get all locked out accounts
Search-ADAccount -LockedOut | FT Name,ObjectClass -A
```

Set-ADAccountPassword

The Set-ADAccountPassword cmdlet sets the password for a user, computer, or service account.

The Identity parameter specifies the Active Directory account to modify.

You can identify an account by its distinguished name, GUID, security identifier (SID) or security accounts manager (SAM) account name. You can also set the Identity parameter to an object variable such as $, or you can pass an object through the pipeline to the Identity parameter. For example, you can use the Search-ADAccount cmdlet to retrieve an account object and then pass the object through the pipeline to the Set-ADAccountPassword cmdlet. Similarly, you can use Get-ADUser, Get-ADComputer, or Get-ADServiceAccount, for standalone MSAs, cmdlets to retrieve account objects that you can pass through the pipeline to this cmdlet.

```
# Set/Reset Password
Set-ADAccountPassword -Identity [SamAccountName] -Reset -NewPassword (ConvertTo-
SecureString -AsPlainText [Password Value] -Force)

# Set/Reset Password from CLI
$NewPassword = (Read-Host -Prompt "Provide New Password" -AsSecureString)
Set-ADAccountPassword -Identity austonMatthews -NewPassword $NewPassword -Reset
```

Unlock-ADAccount

The Unlock-ADAccount cmdlet restores Active Directory Domain Services (AD DS) access for an account that is locked. AD DS access is suspended or locked for an account when the number of incorrect password entries exceeds the maximum number allowed by the account password policy.

the Identity parameter specifies the Active Directory account to unlock. You can identify an account by its distinguished name, GUID, security identifier (SID) or Security Accounts Manager (SAM) account name. You can also set the Identity parameter to an account object variable such as $, or you can pass an object through the pipeline to the Identity parameter. For example, you can use the Search-ADAccount cmdlet to get an account object and then pass the object through the pipeline to the Unlock-ADAccount cmdlet to unlock the account. Similarly, you can use Get-ADUser and Get-ADComputer to get objects to pass through the pipeline.

```
Unlock-ADAccount [SamAccountName]
```

Service Accounts

Add-ADComputerServiceAccount

The Add-ADComputerServiceAccount cmdlet adds one or more computer service accounts to an Active Directory computer.

The Computer parameter specifies the Active Directory computer that will host the new service accounts. You can identify a computer by its distinguished name, GUID, security identifier (SID) or Security Accounts Manager (SAM) account name. You can also set the Computer parameter to a computer object variable, such as $, or pass a computer object through the pipeline to the Computer parameter. For example, you can use the Get-ADComputer cmdlet to retrieve a computer object and then pass the object through the pipeline to the Add-ADComputerServiceAccount cmdlet.

The ServiceAccount parameter specifies the service accounts to add. You can identify a service account by its distinguished name, GUID, Security Identifier (SID) or Security Accounts Manager (SAM) account name. You can also specify service account object variables, such as $. If you are specifying more than one account, use a comma-separated list.

Note: Adding a service account is a different operation than installing the service account locally.

```
# Add a computer account
Add-ADComputerServiceAccount -Computer [Hostname] -ServiceAccount [Value]

# Add multiple accounts
Add-ADComputerServiceAccount -Computer [Hostname] -ServiceAccount [Value],[Value]
```

Install-ADServiceAccount

The Install-ADServiceAccount cmdlet installs an existing Active Directory managed service account on the computer on which the cmdlet is run. This cmdlet verifies that the computer is eligible to host the managed service account. The cmdlet also makes the required changes locally so that the managed service account password can be managed without requiring any user action.

The Identity parameter specifies the Active Directory managed service account to install. You can identify a managed service account by its distinguished name, GUID, security identifier (SID), or security accounts manager (SAM) account name. You can also set the parameter to a managed service account object variable, such as $ or pass a managed service account object through the pipeline to the Identity parameter. For example, you can use Get-ADServiceAccount to get a managed service account object and then pass the object through the pipeline to the Install-ADServiceAccount.

The AccountPassword parameter allows you to pass a secure string that contains the password of a standalone managed service account and is ignored for group managed service accounts. Alternatively, you can use PromptForPassword parameter to prompt for the standalone managed service account password. You must enter the password of a standalone managed service account if you want to install an account that you

have provisioned. This is required when you are installing a standalone managed service account on a server located on a segmented network (site) with read-only domain controllers (for example, a perimeter network or DMZ). In this case you should create the standalone managed service account, link it with the appropriate computer account, and assign a well-known password that must be passed when installing the standalone managed service account on the server on the read-only domain controller site. If you pass both AccountPassword and PromptForPassword parameters, the AccountPassword parameter takes precedence.

```
Install-ADServiceAccount -Identity [Value]
```

New-ADServiceAccount

The New-ADServiceAccount cmdlet creates a new Active Directory managed service account. By default, the cmdlet creates a group managed service account. To create a standalone managed service account which is linked to a specific computer, use the RestrictToSingleComputer parameter. To create a group managed service account which can only be used in client roles, use the RestrictToOutboundAuthenticationOnly parameter. This creates a group managed service account that can be used for outbound connections only and any attempts to connect to services using this account will fail because the account does not have enough information for authentication. You can set commonly used managed service account property values by using the cmdlet parameters. Property values that are not associated with cmdlet parameters can be set by using the OtherAttributes parameter.

The Path parameter specifies the container or organizational unit (OU) for the new managed service account object. When you do not specify the Path parameter, the cmdlet creates an object in the default managed service accounts container for managed service account objects in the domain.

```
New-ADServiceAccount -Name [Name Value] -RestrictToSingleComputer
```

Remove-ADServiceAccount

The Remove-ADServiceAccount cmdlet removes an Active Directory managed service account. This cmdlet does not make changes to any computers that use the managed service account. After this operation, the managed service account no longer exists in the directory, but computers are configured to use the managed service account.

The Identity parameter specifies the Active Directory managed service account to remove. You can identify a managed service account by its distinguished name (DN), GUID, security identifier (SID), or Security Account Manager (SAM) account name. You can also set the Identity parameter to a managed service account object variable, such as $, or you can pass a managed service account object through the pipeline to the Identity parameter. For example, you can use the Get-ADServiceAccount cmdlet to retrieve a managed service account object and then pass the object through the pipeline to the Remove-ADServiceAccount cmdlet.

Note: Removing the service account is a different operation than uninstalling the service account locally.

```
Remove-ADServiceAccount -Identity [SamAccountName]
```
