****

**Lab 8**

**Rev: 09/12/2018**

Active Directory – Computers and Groups

This lab will continue working with Active Directory, but focus on computers and Groups. There are very little differences between users and computers, as far as Active Directory is concerned. Granted Active Directory can store more secondary information about users, but in its eyes there are all just “**accounts/objects**”. The **Search-Adaccount** cmdlet for example lists both users and computer accounts.

Computers in Active Directory

Active Directory will store a significant amount of information about computers listed in the domain. Adding, deleting and modifying computer data is easy, using the cmdlets contained in the Active Directory module. Unfortunately, Active Directory is not the last word concerning domain computers, the administrator is. Therefore, it is important that administrators get a firm grip on PowerShell and be able to perform routine maintenance on directory services.

For example, Active directory will maintain the account of a computer that no longer physically exists in the network. Nothing has told it the computer was removed, so active directory continues as if it was still connected. It is up to the administrator to remove systems from Active Directory, as needed.

Let’s try some basic commands.

#List all available computers. The wildcard is used here

**Get-ADcomputer –filter \***

#list computers based on their operating system and sort them

**Get-ADComputer -Filter \* -Properties OperatingSystem | Select OperatingSystem -unique | Sort OperatingSystem**

The command line, above, used the ***–properties*** option to produce output regarding the OperatingSystem property. Normally, these *extended properties* are not displayed using the Get-ADcomputer cmdlet. You have to specify them in the command line. If you are interested in seeing all of the default and extended properties available to the Get-ADcomputer cmdlet, type in the following;

#choose a computer, listed in AD, and display all of the properties associated with it.

**Get-ADComputer -Identity "<any server name>" -Properties \***

As an administrator, one of your jobs is to be caretaker to Active Directory. When a system logs in to AD it remains there until deleted, even if the system no longer physically exists. Active Directory does not know why the system has not logged in, but keeps the account there in case it comes back online. It is up to the administrator to clear out the unused user and computer accounts.

Here is how to find out about possible inactive systems in a domain.

**Note: Be careful with this script as it will delete listed computers from Active Directory if you answer “Yes” to the last question.**

cls

#number of months will be subtracted from current date and will be used to determine which systems have not logged #on after a specified date

**$a = Read-host "Enter number of Months for testing (0 for current date)"**

#get the LastLogon Date of all computers that have not logged on since current date-number of months

**$result = Get-adcomputer -properties lastLogondate -filter \* `**

**| where {$\_.lastlogondate -lt (get-date).addmonths(-$a)} | sort Name | ft Name, Lastlogondate -auto**

#write the result to a file

**$result | out-file c:\temp\old\_systems.txt**

#output result to screen

**$result**

#ask the user if they want to remove old systems from Active Directory

**write-host**

**$test = Read-Host "Do you want to delete these systems from Active Directory? (Yes or No)"**

**If ($test -eq "Yes") {**

**Get-adcomputer -properties lastLogondate -filter \* | where {$\_.lastlogondate -lt (get-date).addmonths(-$a)} |`**

**Remove-ADComputer**

**Write-Host "System(s) deleted"**

}

**Exercise 1**

Write a script that will delete the old computers by reading the text file that was created in the above script. Before your script deletes the files, use the PowerShell option that will ask for confirmation of the deletion. Place your completed script here and demonstrate that it works. See if you can complete this as a one-line script.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#get the name and operating system using a filter to find all system that are like a windows server.

**Get-ADComputer -Filter { OperatingSystem -Like '\*Windows Server\*' } -Properties OperatingSystem `**

**| Select Name, OperatingSystem | Format-Table -AutoSize**

Add computers

***Why should I manually add a computer to Active Directory?***

It’s a valid question. Once I add users to Active Directory and login to a domain with a new computer, Active Directory will automatically add the computer account to the domain database. This works well with workstations, but some servers may need some extra help to join a domain. It often helps to pre-register a server in Active Directory, so that it is configured with options that may not normally be set by the automatic process. In addition, you main have multiple organizational Units and the New-ADComputer cmdlet will allow you to specify which OU the new system belongs to. Therefore, the system will be registered with all appropriate information when the system actually comes on line.

Adding computers is a relatively simple process. There are many options available when creating computer accounts. It is not necessary to add a password to the account, when creating it, but the account will not be set to an active state. Therefore, you would have to enable the account and set a user (admin) password after creating it.

***NOTE: Don’t forget that the computer added does not have to physically be attached to the domain***

Add Computers - One at a time

#add a new computer. No password is specified and by default the computer account is disabled

**New-ADComputer -Name "Server1" -SamAccountName "Server1"**

#add a new computer and specify its’ location

**New-ADComputer -Name "Server2" -SamAccountName "Server2" –location “Main Floor”**

#add new computer. You will be asked to enter a password. This will automatically enable the account.

**New-ADComputer –name “Server3” " -SamAccountName "Server3" -AccountPassword (Read-Host -AsSecureString "AccountPassword")**

Add Computers - In bunches

**Clear-host**

**#set for 20 new servers**

**foreach ($system in 1..20 )**

**{**

**#set the domain**

**$domain = "DC=LearnName,DC=loc"**

**#create the system name**

**$Label ="MainFloor" + $system**

**#create the new computer. Note: All systems will be disabled due to non-entry of password**

**New-AdComputer -name $Label -SamAccountName $Label -Path $domain -enable $True -location "Toronto"**

**$i ++**

**}**

Set computer attributes

Let’s make some changes to the servers we are using in this course. These changes will not affect the functionality of the systems and their use in future labs. Please see “Get-Help Set-ADComputer” for more information and examples.

Assuming you created one of the servers in one of the previous examples, let’s modify it a bit.

#set the location and do not allow the password to expire.

**Set-ADComputer “Server1” -PasswordNeverExpires $true -Location "Second floor server room" -confirm**

Delete computers

Active Directory does not automatically delete computer accounts that become inactive.

Add Groups (New-ADgroup)

Active Directory Groups are basically containers which allow you to insert user or computer accounts that have theme or common requirement. For example, adding users to an Administrators group provides them with elevated privileges. In addition, Groups will allow you to set specific security restrictions for users that have access to PowerShell. You will look at this in upcoming Lab.

Let’s add a group to Active Directory. Once this is complete you will add users to the group.

Once again there are a large number of options for the **New-Adgroup** cmdlet (see Get-help New-ADgroup –full)

**New-ADGroup -Name "Male Supers" -SamAccountName MaleSupers -GroupCategory Security –GroupScope`**

**Global -DisplayName "Male Superheros" -Path "CN=Users,DC=LearnName,DC=loc" -Description "Members of this group are male superheros "**

Modify Groups

There are a large number of methods/ properties associated with the Set-ADgroup cmdlet. I am sure you know how to list these methods and properties (hint: you need a pipe).

#set the group description and output the result

**Set-ADGroup MaleSupers -Description "These are male super heros” –passthru**

#find all of the groups with the word “Access” and change the description

**get-adgroup -filter 'name -like "Supers\*"' | set-adgroup -description "Super Heros”**

#Pipe group data to Set-ADGroup and add to the name

**Get-ADGroup MaleSupers | Set-ADGroup $\_ -SamAccountName (SamAccountName + "New")**

Delete Group(s)

A pretty basic command. Just specify the group you want deleted. In addition, you can create a script that will read the contents of a file and delete all of the groups contained in that file. Note: Make sure any users you want to keep are moved from the group(s) to be deleted, or else you will lose them.

#remove a single group

**Remove-ADGroup MaleSupers**

#Just as you did with the Set-ADGroup comdlet, you can filter the name and remove several groups at once

**Get-adgroup -filter 'Name -like "Supers\*"' | Remove-ADGroup**

Removing groups listed in text file

Create a couple of groups. Place the names of the groups into a text file called groups.txt and run the scripts below to see what happen to the groups.

#in case you need to get the AD module

**Import-Module ActiveDirectory**

#get the list of groups and the property relating to the ability to delete groups

**$delgroups = get-content C:\temp\groups.txt | Get-ADGroup -Properties ProtectedFromAccidentalDeletion**

#set the property so that we can delete the groups

**$delgroups | Set-ADObject -ProtectedFromAccidentalDeletion $false**

#test the command so that we can see what would happen. There may be errors relating to missing groups or #permission errors

**$delgroups | Remove-ADGroup –whatif**

**#This script uses the same code but with the –WhatIf removed**

**#This section will actually perfrom the deletion.**

**Import-Module ActiveDirectory**

#get the list of groups and the property relating to the ability to delete groups

**$groups = get-content C:\Temp\Groups.txt | Get-ADGroup -Properties ProtectedFromAccidentalDeletion**

#set the property so that we can delete the groups

**$delgroups | Set-ADObject -ProtectedFromAccidentalDeletion $false**

**#delete the groups**

**$delgroups | Remove-ADGroup**

Adding/deleting Users to/from Groups

The purpose of groups is to hold users. Therefore, we will now look at adding/deleting users to/from AD groups. The command to accomplish this is:   
 **Add-ADGroupMember –identity <groupname> -member <username>**

Add Users Manually

#You can add 1 or multiple users to a group by separating the names with a comma.

**Add-ADGroupMember –identity SalesGroup -member user1, user2, user3**

#you can save some typing by omitting the option names

**Add-ADGroupMember SalesGroup user1, user2, user3**

#get users from one group and add to another group

**$usersnames = Get-adgroup SalesGroup**

**$usernames | Add-ADGroupMember –identity NewSalesGroup -member $\_.name**

Adding Multiple Users to a Group from a file

#if you need the module

**Import-module ActiveDirectory**

#Read the CSV file and add all the users to the SalesGroup group.

**Import-CSV "C:\Scripts\Users.csv" | % {Add-ADGroupMember -Identity SalesGroup -Member $\_.UserName }**

**Here is another interesting example that will show you have to add new users, activate their account, add them to a group and do it all one line.**

You have a group called “SalesGroup” and want to add new members to that group. Let’s assume you have a .csv file that contains the required information for the new users. The following code line will a) Import the data from the .csv file; b) create the new user account; c) enable the account and; d) add the account to the SalesGroup group.

**Import-CSV newusers.csv | New-ADUser -passthru | Enable-ADAccount -passthru | Add-ADGroupMember "SalesGroup”.**

**Explanation of the above command line:**

Using a ‘pipe’ the contents of the .csv file is sent to the New-ADuser command. But, on its own this command produces no output and would not send any data to the Enable-ADAccount command. Therefore, we have to add the –passthru option so that the New-AdAccount command sends the account name through the pipe. This is also required in the third section of the piped commands. Finally, the Add-ADGroupMember cmdlet will receive the list of user to add to the group.

**To test this command line, change some of the names in the “newusers.csv” file and run the script again.**

**To delete a user the command is:**

**remove-adgroupmember -Identity <”name of group”> -Member <”name of user”, “name of user”>**

**Example:**

#remove two members from the SalesGroup group

**remove-adgroupmember -Identity “SalesGroup” -Member “Clark Kent”, “Reed Richards”**

Delete all of the members within a group

#Get all of the members of the group

**$group = Get-Group 'SalesGroup'**

#List the members

**$group.Members**

#Now delete the members

**Foreach ($deluser in $group.members){**

**remove-adgroupmember -Identity “SalesGroup” -Member $deluser**

**}**

Exercises

**Script 1**

1. Use the csv file ADusers.csv write a script which will import the script and create new users from this file. Name the file **LearnName\_Lab8\_users.ps1**
2. Notice that the csv column headings are the same names as the parameters needed for New-ADuser. What do you do if the column headings do not match up? You may say, I will amend the csv file, but you may not have access to write to the file, or the owner of the file does not want the file changed? How do I modify the file on the fly? This is an important technique you want to add to your skill set.
3. Open the file NewADusers.csv, notice the column headings do not match the needed parameters. You will need to modify the current column heading to match the parameters needed by New-ADuser: givenname, surname, samaccountname, employeeID,path DC=LearnName,DC=loc
   1. Modify the values by creating a hashtable property and using the substring method generate the givenname, surname, and samaccountname from the LName and FName fields then send all to the toLower method to convert to lower case. (This is tricky programming , but an essential skill. Be careful in the use of quote marks)

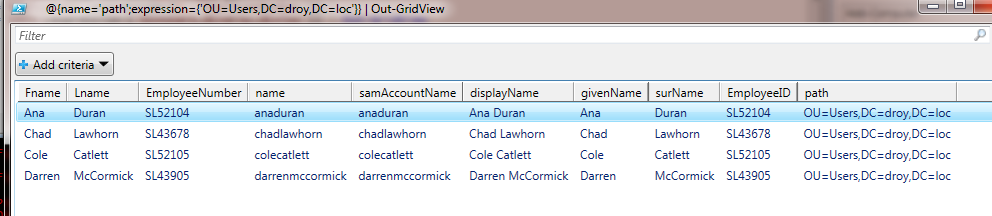
**Sample Code:**

**Import-CSV "$home\documents\newADusers.csv" | Select-Object FName, LName, EmployeeNumber, `**

**@{name='name';expression={($\_.'FName'.substring(0)+$\_.'LName').substring(0).toLower()}}, `**

Carry on with the remaining 5 properties samaccountname, givenname, surname, employeeID, path

* 1. When finished pipe the output to Out-GridView and take a screen shot of your modified fields. Name your file **LearnName\_Lab8\_ADusers**. **jpeg.**Your output should be similar to the screen shot below. (Note: you are not actually creating the accounts with New the -ADuser)

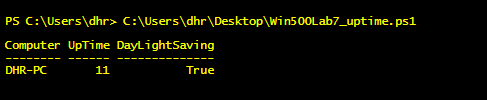


**Script 2**

1. Write a script name **LearnName\_Lab8\_uptime.ps1**, that will show the length of time a computer has been running in total hours. Format you output so that the label “Computer”, displays the computername, the label “Uptime” displays the total rounded hours a computer has been running since the last boot, and the label “DayLightSaving”, displays when the current time is in Day Light Saving time. Display the total output in a compact table. Your output should look similar to the screen shot below.(note: screen shot not taken on virtual machine)

a) Create a new Timespan object that calculates the difference between the last boot time and the current time and save to a variable

b) pipe the variable to the format table cmdlet and create 2 hash table expressions to display the output in the proper format. Take a screen shot of your output, name the file **LearnName­Lab8\_uptime.jpeg.**



Grading

* **LearnName­Lab8\_uptime.jpeg**
* **LearnName\_Lab8\_ADusers.jpeg**
* **LearnName\_Lab8\_users.ps1**
* **LearnName\_Lab8\_uptime.ps1**

Zip the files together and upload using the link in Graded Work on My Seneca