© 2017-2021 lukas@plachy.eu
Ver. 0.97; allways check new version at http://lukas.plachy.eu/kb

Windows 10 / Server 2012 PowerShell 4.0 & cmd HOWTOs

```
PS > (PowerShell)
```

```
C:\> (CommandLine)
```

WTF should be this? This was NOT intended:

- neither as full PS/CMD reference (since there are no complete parameter enumerations),
- neither as a manual (since there are no comments/explanatory texts),
- neither as a (full) list of commands (since the list is not refreshing itself),

but rather as

- a “hint” list, first touch of the necessary commands/scripts ordered in the a fashion of typical tasks an IT Administrator usually faces

and

- an overview of those, you might most probably need, a kind of a restaurant-like menu card, without the necessity of going (and memorizing) through the entire PS reference including all its commandlets or heating up web-search-engines[1] (especially in such a moments, when you should need these when desperately trying to bring up the only connectivity you can have or under high-security circumstances, when working offline at all).

The author may only express his whish that one may find this ‘menu’ useful.

[1] Not speaking about the fashion of the nowadays, when even simple web pages, wrongfully designed and written as client-side application, need to download MBytes of stuff just to display a simple information, most probably not much to Sir T.J. Berner-Lee’s peace of mind.

Contents:

Special chars on CZKeyboard and basic overview:

| (pipe, roura) =	AltrGr + **W**	**@** (zavináč, at)	AltGr + **V**
***** (star, hvězdička) =	AltGr + **-**	**&** (and)=	AltGr + **C**
$ (dolar) =	AltGr + **“ (ů)**	**~** (tilda)=	AltGr + **1 (+)**
[] (square brack.)=	AltGr + **FG**	**** (backslash)=	AltGr + **Q**
{} (braces)=	AltGr + **BN**	**`** (backapostrophe)=	

MULTILINE SEPARATOR IN PS: **;** (semicolon)
NEWLINE ISERTOR IN PS: **`** (backapostrophe)
STRING DENOMINATOR: **‘** (apostrophe) or **“** (quote)
STRING ESCAPER:

```
” ‘$normal string and $expanded_string”
```

ALL ATTRIBUTES (COLUMNS) SELECTOR:

```
| Select-Object -Property [a-z]*
```

NOTE: Most of the “Get-“ commands due to brevity and clarity do NOT list all the parameters (attributes) of an object by default! Use the command above to force it!

ROW FILTER:

```
| Where-Object –Property $_.AttribName –eq “AttribValue”
| WHERE {$_.AttribName –eq “AttribValue”}
| WHERE {$_.AttribName –Match “^Substr*Substr$”}
```

For further filtering see Comparison operators [LINK] and Logical operators [LINK] annexes.

COLUMN (FORMAT) FILTER:

```
| Format-List *
| FL *
| FL AttributeName1, AttributeName2….
| Format-Table* -auto
| Format-Table* -auto -GroupBy Attribiute1 -Wrap
| FT AttributeName1, AttributeName2…. –auto
| FT AttributeName1, @{Label="TotalRunningTime";
      Expression={(Get-Date) - $_.StartTime}

| Format-Wide –Column 3
```

Intellisense:	**Tab**
Execution break:	**Ctrl+C**
Clear current line:	**Esc**
Browse command history:	**ArrowUp / ArrowDown / F7 (+F9 and num) / F8 (with text search)**
Delete entire AFTER cursor:	**Ctrl+End**
Switch insert/overwrite:	**Ins**

1. Workstations (and servers)

1.1. System intro

Powershell execution policy

```
> Get-ExecutionPolicy
> Set-ExecutionPolicy Restricted
> Set-ExecutionPolicy All Signed
> Set-ExecutionPolicy Remote Signed
> Set-ExecutionPolicy Unrestricted
```

CPU Information [RefLink]

```
> Get-WmiObject Win32_Processor | FL *
```

```
> systeminfo
```

System information

```
> Get-CimInstance Win32_OperatingSystem | FL *
```

```
> systeminfo
```

Get/list current PowerShell Version

```
> $PSVersionTable
```

```
> Get-PSSessionConfiguration
```

List network interfaces and their addresses

```
> Get-NetIpAddress
> Get-WmiObject -Class Win32_NetworkAdapterConfiguration -Filter
IPEnabled=TRUE | Select-Object -Property [a-z]*
```

```
> ipconfig /all
```

List drivers and devices

DeviceManagement.zip

```
> Import-Module .\DeviceManagement.psd1 -Verbose
Get-Device
```

DevCon.zip

```
> devcon hwids =ports //By class … class list > devcon classes
> devcon hwids * > hwids.txt
```

List services

```
> Get-Service | ConvertTo-HTML -Property Name, Status > C:\services.htm
> Get-Service | Export-CSV c:\service.csv
```

List log records

```
> Get-EventLog -Log "Application"
```

List installed features

```
(> Import-Module ServerManager)
> Get-WindowsFeature | where-object {$_.Installed -eq $True}
```

List available features

```
(> Import-Module ServerManager)
> Get-WindowsFeature
```

Clean window

```
> Clear-Host
> cls
```

List all WMI classes

```
> Get-WMIObject -List| Where{$_.name -match "^Win32_"} | Sort Name |
Format-Table Name
```

List currently available PS cmdlets - Microsoft.Powershell.Core

```
> Get-Command
```

List for loading available PS cmdlet modules - Microsoft.Powershell.Core

```
> Get-Module -ListAvailable
```

Add PS cmdlet modules - Microsoft.Powershell.Core – Import-Module

```
> Import-Module ModuleName
```

Microsoft.Powershell.Core – Get-Help

```
> Get-Help
> Get-Help Set-Location
```

Microsoft.Powershell.Core – list of commands for a module

```
> Get-Command -Module Hyper-V
```

For module name – see „install“ in each chapter, or try to use the first word after the ‘-’slash sign …
eg. Get-DnsRecords = Get-Command DNS

Microsoft.PowerShell.Core – Providers (virtual classes)

```
> Get-PSProvider
```

```
> cd Registry::
> cd Variable::
> cd Function::
> cd Environment::
> cd Alias::
> cd Cert:
```

Connect to a different machine / open different PS SEssion - Microsoft.Powershell.Core

```
> New-PSSession -ComputerName Server01 -Port 8081 -UseSSL
```

```
> Enter-PSSession -Computer Server01
```

```
> $s = New-PSSession -ComputerName Server01
> Enter-PSSession -Session $s
```

```
Exit-PSSession
exit
```

WHEN (client) NOT IN DOMAIN, the situation gets rather complicated:
On server:

```
> netsh advfirewall set service type = remotedesktop mode = enable
> $Cert = New-SelfSignedCertificate -CertstoreLocation
Cert:\LocalMachine\My -DnsName "HOSTNAME","HOSTNAME2","HOSTNAME3"
> Export-Certificate -Cert $Cert -FilePath C:\tmp\cert.crt
> Enable-PSRemoting -SkipNetworkProfileCheck -Force
> dir wsman:\localhost\listener
> Get-ChildItem WSMan:\Localhost\listener | Where -Property Keys -eq
"Transport=HTTP" | Remove-Item -Recurse
> New-Item -Path WSMan:\LocalHost\Listener -Transport HTTPS -Address * -
CertificateThumbPrint $Cert.Thumbprint -Force
> dir wsman:\localhost\listener
> New-NetFirewallRule -DisplayName "Windows Remote Management (HTTPS-
Inbound)" -Name "Windows Remote Management (HTTPS-Inbound)" -Profile Any -
LocalPort 5986 -Protocol TCP
> Set-Item WSMan:\localhost\Service\EnableCompatibilityHttpsListener -Value
true
> Set-NetConnectionProfile -NetworkCategory Private
```

Note: HOSTNAME2 a 3 shall be the alternate names

On client:

```
> Set-Item WSMan:\localhost\Client\TrustedHosts -Value <RemoteServerName> -
Concatenate [-Force]
> winrm set winrm/config/client @{TrustedHosts="RemoteComputerName"}
```

(the above are equivalent)

```
Import-Certificate -Filepath "C:\temp\cert" -CertStoreLocation
"Cert:\LocalMachine\Root"
Add-Content $Env:SystemRoot\system32\drivers\etc\hosts "10.0.0.1 myHost"
Enter-PSSession -ComputerName myHost -UseSSL -Credential DOMAIN\username
```

Start in a different user context

```
> Enter-PSSession -Computer localhost -Credential "TestDomain\Me"
```

```
> Start-Process powershell.exe -Credential "TestDomain\Me"
```

```
> Start-Process powershell.exe -Credential "TestDomain\Me" -NoNewWindow -
ArgumentList "Start-Process powershell.exe -Verb runAs"
```

PowerShell profile manipulation

```
> Test-Path $profile
> New-Item -path $profile -type file -force
```

PowerShell profile window parameters (put into the $profile file)

```
$Shell.WindowTitle="MyWonderfullPowerShellWindow"
$Shell = $Host.UI.RawUI
$size = $Shell.WindowSize
$size.width=70
```

```
$size.height=25
$Shell.WindowSize = $size
$size = $Shell.BufferSize
$size.width=70
$size.height=5000
$Shell.BufferSize = $size
$shell.BackgroundColor = "Gray"
$shell.ForegroundColor = "Black"
```

(Power)Shell exit or session terminate (last session termination means exit)

```
> exit
```

```
> exit
```

1.2. PowerShell signing

Powershell execution policy

```
> Get-ExecutionPolicy
> Set-ExecutionPolicy Restricted
> Set-ExecutionPolicy All Signed
> Set-ExecutionPolicy Remote Signed
> Set-ExecutionPolicy Unrestricted
```

makecert.zip

Powershell create root authority cert for signing batch files (powershell scripts)

```
> makecert -n "CN=PowerShell Local Certificate Root" -a sha1 -eku
1.3.6.1.5.5.7.3.3 -r -sv root.pvk root.cer -ss Root -sr localMachine
```

Powershell create cert for signing batch files (powershell scripts)

```
> makecert -pe -n "CN=PowerShell User" -ss MY -a sha1 -eku
1.3.6.1.5.5.7.3.3 -iv root.pvk -ic root.cer
```

Powershell sign batch file (powershell script)

```
> Set-AuthenticodeSignature c:\<filename>.ps1 @(Get-ChildItem
cert:\CurrentUser\My -codesign)[0]
```

1.3. Basic file content manipulation / read / edit

Note: For entire file/folder management take a peek on sections: 1.15. Folder management and 1.16. Files management

Print working directory

```
> Pwd
> Get-CurrentLocation
```

List folders/files

```
> Get-ChildItem
> Get-ChildItem -Recurse //CAREFULLY IN ROOT FOLDERS!!!!!
> dir
```

Change local folder

```
> Set-Location -Path C:\
> Set-Location -Path HKLM:\SOFTWARE
> cd .\MyFolderName
> cd Registry::
```

```
> Push-Location
> Pop-Location
```

Read text file

```
> Get-Content c:\file.txt
> Get-Content c:\file.txt | Select-Object -last 5
```

```
> more c:\file.txt
```

Read text file – piping lines as input into other command

```
> Get-Content c:\scripts\test.txt | Foreach-Object {Get-Wmiobject -
computername $_ win32_bios}
```

Input / add of content into a file

```
> new-item -path c:scriptsnewfile.txt -value "this file was created on
12/3." -itemtype file
```

```
> add-content C:\file.txt "NEW TEXT LINE"
> add-content c:\scripts\*.htm "<p><i>some additional text into all
files</i>"
```

Replace file content

```
> set-content c:\scripts\*.htm "holy new text, batman!"
```

```
> (Get-Content .\input.txt ).Replace('text','fun') | Out-File .\output.txt
```

Edit file content

```
> notepad C:\file.txt //☹
```

1.4. Device management

DeviceManagement.zip

Install [LINK]

```
> Import-Module .\DeviceManagement.psd1 -Verbose
```

List devices

```
> Get-Device
```

Edit device – disable

```
> $deviceName = Read-Host -
Prompt 'Please enter the Name of the Device to Disable'; Get-
Device | Where-Object -Property Name -Like $deviceName | Disable-Device
```

```
> $deviceName = Read-Host -
Prompt 'Please enter the Name of the Device to Disable'; Get-
Device | Where-Object -Property Name -Like $deviceName | Enable-Device
```

Edit device - driver update/change (NOT POSSIBLE USING PS! NOT POSSIBLE USING default CMD) [LINK]

DevCon.zip

On a remote:

```
> devcon /m:\\server01 command
```

Edit device – list hw

```
> devcon hwids *
> devcon hwids * > hwids.txt
> devcon hwids *floppy* //By name
> devcon hwids =ports //By class … class list > devcon classes
> devcon listclass net diskdrive cdrom tapedrive … //Devices in classes …
> devcon hwids =net … //As previous command
> devcon find *mou*
> devcon find @root\legacy*
> devcon find =legacydriver
> devcon findall =net
```

Edit device – list software enumerated device

```
> devcon drivernodes sw*
> devcon drivernodes @ROOT\MEDIA*
```

Edit device – list resources allocated

```
> devcon resources =hdc
```

Edit device – list driver files in use / of a device

```
> devcon driverfiles
> devcon driverfiles "HID\Vid_045e&Pid_0039&Rev_0121" //Class name from
„hwids“
```

Edit device – status / restart / enable / disable

```
> devcon status * > status.txt
```

```
> devcon restart =net @&#39;ROOT\*MSLOOP\0000
```

```
> devcon enable &#39;*PNP0000
```

(the &#38 is an single quote ‘ character to not eval the * as wildacrd)

```
> devcon /r disable USB*
> devcon /r disable "@USB\ROOT_HUB\4&2A40B465&0"
```

Edit device – append driver into repository

```
> devcon dp_add C:\WinDDK\5322\src\general\toaster\inf\i386\toaster.inf
```

Edit device – remove driver from repository

```
> devcon dp_enum
```

```
The following 3rd party Driver Packages are on this machine:
```

```
oem2.inf
    Provider: Microsoft
    Class: unknown
    Date: 12/10/2004
    Version: 2.0.1403.0
```

```
> devcon dp_delete oem2.inf
```

(DO NOT try to use the original .inf filename, but the one from the listing!!!!)

Edit device – update driver in repository

```
> devcon update c:\windows\inf\test.inf *PNP0501
```

Edit device – rescan for new devices

```
> devcon rescan
```

Edit device – install device

```
> devcon /r install c:\windows\inf\keyboard.inf *PNP030b
```

*PNP030b = hardware ID

Edit device – remove device (by ID)

!!!! LIST FIRST:

```
> devcon status @usb\\\*
> devcon hwids @usb\\\*
```

```
> devcon /r remove @usb\*
```

Edit device – remove device (by drivername)

!!!! LIST FIRST:

```
> devcon status =net *ndiswan\*
> devcon hwids =net *ndiswan\*
```

```
> devcon /r remove =net *ndiswan*
```

1.5. Disk management

List available (connected) drives

```
> [System.IO.DriveInfo]::getdrives()
```

```
> wmic logicaldisk get deviceid, volumename, description
```

List available volumes (partitions)

```
> Get-Partition
```

List with mount points:

```
Get-WmiObject Win32_Volume | Format-Table Name, Label, FreeSpace, Capacity
```

Drive letter / connection point change/modify

```
> Get-Partition -DriveLetter D | Set-Partition -NewDriveLetter E
```

```
> $drive = Get-WmiObject -Class win32_volume -Filter "DriveLetter = 'e:'";
> Set-WmiInstance -input $drive -Arguments @{DriveLetter="Q:"; Label="Label"}
```

Drive label modify/change

```
> $drive = Get-WmiObject -Class win32_volume -Filter "DriveLetter = 'e:'";
> Set-WmiInstance -input $drive -Arguments @{ Label="Label"}
```

Drive letter / connection point add

```
> Get-Partition
> Add-PartitionAccessPath -DiskNumber 1 -PartitionNumber 2 -AccessPath F:
> Add-PartitionAccessPath -DiskNumber 1 -PartitionNumber 2 -AccessPath
C:\path\to\empty\directory
```

Drive letter / connection point remove

```
> Get-Partition
> Remove-PartitionAccessPath -DiskNumber 1 -PartitionNumber 2 -AccessPath
F:
> Remove-PartitionAccessPath -DiskNumber 1 -PartitionNumber 2 -AccessPath
C:\path\to\empty\directory
```

Drive Online / Offile / ReadOnly

```
> Set-Disk -Number 5 -IsOffline $False
```

```
> Set-Disk -Number 5 -IsOffline $True
```

```
> Set-Disk –Number 5 -IsReadonly $False
```

Drive master partition init

```
> Initialize-Disk –PartitionStyle GPT
> Initialize-Disk –PartitionStyle MBR
```

Drive partition list

```
> Get-Partition
```

List with mount points:

```
> Get-WmiObject Win32_Volume | Format-Table Name, Label, FreeSpace,
Capacity
```

Drive partition create

```
> New-Partition –DiskNumber 1 –UseMaximumSize
```

```
> New-Partition –DiskNumber 1 –Offset bytes_from_beginning –Size in_bytes
```

//Bytes, KB, MB, GB, TB

Drive partition resize

```
> Set-Partition –DiskNumber 1 –UseMaximumSize
```

```
> Set-Partition –DiskNumber 1 –Offset bytes_from_beginning –Size in_bytes
```

//Bytes, KB, MB, GB, TB

Drive partition delete

```
> Remove-Partition –DiskNumber 1 –PartitionNumber 1
```

Drive partition format

```
> Format-Volume -DriveLetter D
```

```
> Format-Volume -DriveLetter C -FileSystem FAT32 -FullFormat -Force
```

// -Filesystem: **NTFS**, **ReFS**, **exFAT**, **FAT32**, and **FAT**.
// -AllocationUnitSize <UInt32>

Partition – data attributes

OperationalStatus	*IsActive*	*Offset*
Type	*IsBoot*	*PartitionNumber*
AccessPaths	*IsHidden*	*Size*
DiskId	*IsOffline*	*TransitionState*
DiskNumber	*IsReadOnly*	*PSComputerName*
DriveLetter	*IsShadowCopy*	*CimClass*
GptType	*IsSystem*	*CimInstanceProperties*
Guid	*MbrType*	*CimSystemProperties*
	NoDefaultDriveLetter	

Disk – data attributes

PartitionStyle	*IsBoot*	*Path*
ProvisioningType	*IsClustered*	*PhysicalSectorSize*
OperationalStatus	*IsOffline*	*SerialNumber*
HealthStatus	*IsReadOnly*	*Signature*
BusType	*IsSystem*	*Size*
UniqueIdFormat	*LargestFreeExtent*	*UniqueId*
OfflineReason	*Location*	*PSComputerName*
AllocatedSize	*LogicalSectorSize*	*CimClass*
BootFromDisk	*Manufacturer*	*CimInstanceProperties*
FirmwareVersion	*Model*	*CimSystemProperties*
FriendlyName	*Number*	
Guid	*NumberOfPartitions*	
	ObjectId	

1.6. Services management

List Services

```
> Get-WmiObject -Class Win32_Service
```

```
> Get-WmiObject -Class Win32_Service | FL Name, Caption, Description,
Started, StartMode, Path
```

```
> Get-WmiObject -Class Win32_Service | WHERE {$_.StartMode -eq "Disabled"}
| FL Name, Caption, Description, Started, StartMode, Path
```

WHERE {$_.StartMode -eq “Manual”}
WHERE {$_.StartMode -eq “Auto”}

```
> Get-WmiObject -Class Win32_Service | WHERE {($_.StartMode -eq "Auto") -
and ($_.Started -eq "True")} | FL Name, Caption, Description, Started,
StartMode, Path
```

```
> Get-WmiObject -Class Win32_Service | WHERE {($_.StartMode -eq "Auto") -
and ($_.Started -eq "True") -and ($_.Caption -eq "Klient DNS")} | FL Name,
Caption, Description, Started, StartMode, Path
```

```
> wmic service
```

```
> wmic service get name,startname
```

```
> wmic service where started=true get name, startnam
```

```
> wmic service where 'name like "%sql%"' get name, startname
```

Start Service

```
> Start-Service dnscache
```

```
> Start-Service –DisplayName “Klient DNS”
```

```
> net start Dnscache
```

(Service name is case sensitive!!!)

Stop Service

```
> Stop-Service dnscache
```

```
> Stop-Service –DisplayName “Klient DNS”
```

```
> net stop Dnscache
> net stop W32Time
```

(Service name is case sensitive!!!)

Suspend Service

```
> Suspend-Service dnscache
```

```
> Suspend-Service –DisplayName “Klient DNS”
```

Restart Service

```
> Restart-Service dnscache
```

```
> Restart-Service –DisplayName “Klient DNS”
```

Modify/change start/stop mode of service

```
> Set-Service dnscache -startuptype "manual"
> Set-Service dnscache -startuptype "disabled"
> Set-Service dnscache -startuptype "automatic"
```

Register new service

```
> New-Service -Name "TestService" -BinaryPathName
"C:\WINDOWS\System32\svchost.exe -k netsvcs" -DependsOn NetLogon -
DisplayName "Test Service" -StartupType Manual -Description "This is a test
service."
```

Deregister a service

```
> $service = Get-WmiObject -Class Win32_Service -Filter
"Name='servicename'"
> $service.delete()
```

Data attributes

PSComputerName
Name
Status
ExitCode
DesktopInteract
ErrorControl

PathName
ServiceType
StartMode
__GENUS
__CLASS
__SUPERCLASS

__DYNASTY
__RELPATH
__PROPERTY_COUNT
__DERIVATION
__SERVER
__NAMESPACE

```
__PATH
AcceptPause
AcceptStop
Caption
CheckPoint
CreationClassName
Description
DisplayName
InstallDate
ProcessId
```

```
ServiceSpecificExitCo
de
Started
StartName
State
SystemCreationClassNa
me
SystemName
TagId
WaitHint
```

```
Scope
Path
Options
ClassPath
Properties
SystemProperties
Qualifiers
Site
Container
```

1.7. Services management – NTP client (time sync service)

```
> w32tm
```

(Run cmd as Administrator!!!!)

Get current config

```
> w32tm /query /configuration
```

Update time asap

```
> w32tm /resync
```

Register as service

```
> w32tm /register
```

Un/Deregister as service (delete service)

```
> w32tm /unregister
```

Change config - manual

```
> w32tm /config /update /syncfromflags:MANUAL
/manualpeerlist:192.53.103.108,192.53.103.104
```

Change config – get from domain DC

```
> w32tm /config /syncfromflags:DOMHIER
```

Change config – get from none

```
> w32tm /config /syncfromflags:NO
```

Change config – get from all

```
> w32tm /config /syncfromflags:ALL
```

Change config – set, that this computer is a reliable datetime source

```
> w32tm /config /reliable:YES
```

Display timezone settings

```
> w32tm /tz
```

Backup settings

```
> w32tm /dumpreg
```

Service stop

```
> net stop W32Time
```

Service start

```
> net start W32Time
```

1.8. Services management – WINS (workgroup!) masterbrowser

In registry: \HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services \Browser\Parameters
Value: MaintainServerList = {Yes, No, Auto}
Value: IsDomainMaster = {True, False}

Disable masterbrowser

```
> SC config Browser start= disabled
```

(Administrator commandline!)

\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services
\Browser\Parameters\MaintainServerList = No

Enable masterbrowser

```
> SC config Browser start= auto
```

(Administrator commandline!)

\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services
\Browser\Parameters\MaintainServerList = Auto

1.9. Services management – firewall management (+ network test)

List rules

```
> Get-NetFirewallRule
```

Add rule

```
>  New-NetFirewallRule -DisplayName "Block Outbound Port 80" -Direction
Outbound –LocalPort 80 -Protocol TCP -Action Block
```

```
>  New-NetFirewallRule -DisplayName "Allow Messenger" -Direction Inbound -
Program "C:\Program Files (x86)\Messenger\msmsgs.exe" -RemoteAddress
LocalSubnet -Action Allow
```

```
> New-NetFirewallRule -DisplayName "Block WINS" -Direction Inbound -Action
Block -RemoteAddress WINS
```

Edit rule – enable/disable

```
> Enable-NetFirewallRule -DisplayName "RuleToChange"
```

```
> Disable-NetFirewallRule -DisplayName "RuleToChange"
```

Edit rule

```
> Set-NetFirewallRule -DisplayName "RuleToChange" -RemoteAddress
"newIPv4address"
```

Delete rule

```
> Remove-NetFirewallRule -DisplayName "RuleToRemove"
```

Add remote access

```
> netsh advfirewall set service type = remotedesktop mode = enable
```

List rules

```
> Test-NetConnection "Server" -Port "Number" -InformationLevel "Detailed" -
DiagnoseRouting -ConstrainInterface # -ConstrainSourceAddress „address" -
CommonTCPPort {SMB,HTTP,RDP,WINRM} -Hops ## -Tracroute
```

PortQryV2.zip

PortQryUI.zip

```
> portqry -n "Server" -p TCP -e "Number"
```

Firewal rule - data entity ettributes

Name	*Caption*	*SystemCreationClassName*
ID	*Description*	*SystemName*
Group	*ElementName*	*DisplayGroup*
Platform	*InstanceID*	*LocalOnlyMapping*
LSM	*CommonName*	*LooseSourceMapping*
DisplayName	*PolicyKeywords*	*Owner*
Enabled	*PolicyDecisionStrategy*	*Platforms*
Profile	*PolicyRoles*	*PolicyStoreSource*
Direction	*ConditionListType*	*Profiles*
Action	*CreationClassName*	*RuleGroup*
EdgeTraversalPolicy	*ExecutionStrategy*	*StatusCode*
PrimaryStatus	*Mandatory*	*PSComputerName*
Status	*PolicyRuleName*	*CimClass*
EnforcementStatus	*Priority*	*CimInstanceProperties*
PolicyStoreSourceType	*RuleUsage*	
	SequencedActions	

1.10. Process management

List Processes [RefLink]

```
> Get-Process
```

```
> Get-Process winwo*,iexplo*
```

```
> Get-Process | Select-Object name,fileversion,productversion,company
```

```
> wmic
Wmic:root\cli> process list
```

List Processes - Group by priority:

```
> $A = Get-Process PS C:\> Get-Process -InputObject $A | Format-Table -View
priorit
```

Kill Process

```
> Stop-Process -processname notepad
> Stop-Process 3512
```

Create new (background = detached) process

```
> Start-Job -ScriptBlock {& java -jar MyProgram.jar >console.out
>console.err }
```

```
> Start-Process java -ArgumentList '-jar', 'MyProgram.jar' -
RedirectStandardOutput '.\console.out' -RedirectStandardError
'.\console.err'
```

WMI – Process data entity attributes

__NounName	*BasePriority*	*PeakVirtualMemorySize*
Name	*ExitCode*	*PeakVirtualMemorySize64*
Handles	*HasExited*	*PriorityBoostEnabled*
VM	*ExitTime*	*PrivateMemorySize64*
WS	*Handle*	*PrivilegedProcessorTime*
PM	*MachineName*	*ProcessName*
NPM	*MainWindowHandle*	*ProcessorAffinity*
Path	*MainWindowTitle*	*Responding*
Company	*MainModule*	*SessionId*
CPU	*MaxWorkingSet*	*StartInfo*
FileVersion	*MinWorkingSet*	*StartTime*
ProductVersion	*Modules*	*SynchronizingObject*
Description	*NonpagedSystemMemorySize*	*Threads*
Product	*NonpagedSystemMemorySize64*	*UserProcessorTime*
Id	*PagedMemorySize64*	*VirtualMemorySize64*
PriorityClass	*PagedSystemMemorySize*	*EnableRaisingEvents*
HandleCount	*PagedSystemMemorySize64*	*StandardInput*
WorkingSet	*PeakPagedMemorySize*	*StandardOutput*
PagedMemorySize	*PeakPagedMemorySize64*	*StandardError*
PrivateMemorySize	*PeakWorkingSet*	*WorkingSet64*
VirtualMemorySize	*PeakWorkingSet64*	*Site*
TotalProcessorTime		*Container*

1.11. Network management

List interfaces

```
> Get-NetAdapter -Name * -IncludeHidden
```

List interfaces and their addresses

```
> Get-NetIPConfiguration
> Get-NetIpAddress
> Get-WmiObject -Class Win32_NetworkAdapterConfiguration -Filter
IPEnabled=TRUE | Select-Object -Property [a-z]*
```

```
> ipconfig /all
```

List connected WLAN – interfaces BSSID

```
> netsh wlan show interfaces
```

Modify interfaces config – address/DHCP/mask/DNS

```
> Get-NetAdapter -Name * -IncludeHidden
> Set-NetIPAddress -InterfaceIndex 12 -IPAddress 192.168.0.1 -PrefixLength
24
```

[LINK]

```
> Get-NetIPInterface
```

```
> Set-NetIPInterface
> Set-NetIPInterface -InterfaceIndex 12 -Dhcp Enabled
```

```
$netadapter = Get-NetAdapter -InterfaceIndex 12
$netadapter | Set-NetIPInterface -Dhcp Disabled
$netadapter | New-NetIPAddress -IPAddress 192.168.0.14 -PrefixLength 24 -
DefaultGateway 192.168.0.1
$netadapter | Set-NetIPAddress -IPAddress 192.168.0.16 -PrefixLength 24 -
DefaultGateway 192.168.0.1
Set-DnsClientServerAddress -InterfaceIndex 12 -ServerAddresses
"192.168.0.10"
```

Modify/change interfaces config – WINS – view

```
> Get-WmiObject win32_networkadapterconfiguration | where ipenabled -eq
true | format-table index, description, ipaddress, winsprimaryserver,
winssecondaryserver -autosize
```

Modify/change interfaces config – WINS – set

```
> $WINS = Get-WmiObject win32_networkadapterconfiguration | where index -eq
13; $WINS.SetWINSServer("192.168.0.10","192.168.0.11")
```

Modify/change interfaces config – WINS – remove

```
> $WINS = Get-WmiObject win32_networkadapterconfiguration | where index -eq
13;$WINS.SetWINSServer("$Null","$Null")
```

Stop (disable/ forbidden) interface

```
> Disable-NetAdapter
```

```
> $wmi = Get-WmiObject -Class Win32_NetworkAdapter -filter "Name LIKE
'%Wireless%'" -credential (Get-Credential) -computername remotecomputer;
$wmi.disable()
```

```
> $wmi = Get-WmiObject -Class Win32_NetworkAdapter -filter "Name LIKE
'%Wireless%'"; $wmi.disable()
```

Start (enable / allow) interface

```
> Enable-NetAdapter -Name ethernet -Confirm:$false
```

```
> $wmi = Get-WmiObject -Class Win32_NetworkAdapter -filter "Name LIKE
'%Wireless%'" -credential (Get-Credential) -computername remotecomputer;
$wmi.enable()
```

```
> $wmi = Get-WmiObject -Class Win32_NetworkAdapter -filter "Name LIKE
'%Wireless%'"; $wmi.enable()
```

Restart interface

```
> Get-NetAdapter -InterfaceIndex 12 | Restart-NetAdapter -Confirm:$false
```

Release/Renew DHCP interface

```
> Get-WmiObject -Class Win32_NetworkAdapterConfiguration -Filter
"IPEnabled=true and DHCPEnabled=true" -ComputerName . | Where-Object -
FilterScript {$_.DHCPServer -contains "192.168.1.254"} | ForEach-Object -
Process {$_.ReleaseDHCPLease()}
```

```
> Get-WmiObject -Class Win32_NetworkAdapterConfiguration -Filter
"IPEnabled=true and DHCPEnabled=true" -ComputerName . | Where-Object -
FilterScript {$_.DHCPServer -contains "192.168.1.254"} | ForEach-Object -
Process {$_.ReleaseDHCPLease()}
```

```
> ifconfig /release
> ifconfig /renew
```

Rename network interface

```
> Get-NetAdapter -InterfaceIndex 12 | Rename-NetAdapter -NewName Renamed
```

List routing table

```
> Get-NetRoute
```

```
> Route PRINT
```

Add routing table

```
>New-NetRoute -DestinationPrefix "10.0.0.0/24" -InterfaceIndex 12 -NextHop
192.168.0.
```

```
> route ADD destination_network MASK subnet_mask  gateway_ip metric_cost
```

List ARP table

```
> arp /a
```

Add ARP table static record

```
> arp /s 10.0.0.80 00-AA-00-4F-2A-9C
```

Clear ARP table

```
> arp –d *
> arp –d 192.168.0.10
```

Nic teaming

```
> New-NetLbfoTeam -Name NICTeam1 -TeamMembers pNIC1,pNIC2 -
LoadBalancingAlgorithm TransportPorts -TeamingMode SwitchIndependent
```

List net adapters – data entity attributes

ifAlias	*ifOperStatus*	*OtherEnabledState*
InterfaceAlias	*Caption*	*RequestedState*
ifIndex	*Description*	*TimeOfLastStateChange*
ifDesc	*ElementName*	*TransitioningToState*
ifName	*InstanceID*	*AdditionalAvailability*
DriverVersion	*CommunicationStatus*	*Availability*
LinkLayerAddress	*DetailedStatus*	*CreationClassName*
MacAddress	*HealthState*	*DeviceID*
Status	*InstallDate*	*ErrorCleared*
LinkSpeed	*Name*	*ErrorDescription*
MediaType	*OperatingStatus*	*IdentifyingDescriptions*
PhysicalMediaType	*OperationalStatus*	*LastErrorCode*
AdminStatus	*PrimaryStatus*	*MaxQuiesceTime*
MediaConnectionState	*StatusDescriptions*	*OtherIdentifyingInfo*
DriverInformation	*AvailableRequestedStates*	*PowerManagementCapabilities*
DriverFileName	*EnabledDefault*	
NdisVersion	*EnabledState*	*PowerManagementSupported*

PowerOnHours	DriverDate	MinorDriverVersion
StatusInfo	DriverDateData	MtuSize
SystemCreationClassName	DriverDescription	NdisMedium
SystemName	DriverMajorNdisVersion	NdisPhysicalMedium
TotalPowerOnHours	DriverMinorNdisVersion	NetLuid
MaxSpeed	DriverName	NetLuidIndex
OtherPortType	DriverProvider	NotUserRemovable
PortType	DriverVersionString	OperationalStatusDownDefaultPortNotAuthenticated
RequestedSpeed	EndPointInterface	OperationalStatusDownInterfacePaused
Speed	HardwareInterface	OperationalStatusDownLowPowerState
UsageRestriction	Hidden	OperationalStatusDownMediaDisconnected
ActiveMaximumTransmissionUnit	HigherLayerInterfaceIndices	PnPDeviceID
AutoSense	IMFilter	PromiscuousMode
FullDuplex	InterfaceAdminStatus	ReceiveLinkSpeed
LinkTechnology	InterfaceDescription	State
NetworkAddresses	InterfaceGuid	TransmitLinkSpeed
OtherLinkTechnology	InterfaceIndex	Virtual
OtherNetworkPortType	InterfaceName	VlanID
PermanentAddress	InterfaceOperationalStatus	WdmInterface
PortNumber	InterfaceType	PSComputerName
SupportedMaximumTransmissionUnit	iSCSIInterface	CimClass
AdminLocked	LowerLayerInterfaceIndices	CimInstanceProperties
ComponentID	MajorDriverVersion	CimSystemProperties
ConnectorPresent	MediaConnectState	
DeviceName	MediaDuplexState	
DeviceWakeUpEnable		

List ip addresses – data entity attributes

ifIndex	PrimaryStatus	AddressOrigin
PrefixOrigin	Status	AddressType
SuffixOrigin	StatusDescriptions	IPv4Address
Type	AvailableRequestedStates	IPv6Address
Store	EnabledDefault	IPVersionSupport
AddressFamily	EnabledState	PrefixLength
AddressState	OtherEnabledState	SubnetMask
Caption	RequestedState	InterfaceAlias
Description	TimeOfLastStateChange	InterfaceIndex
ElementName	TransitioningToState	IPAddress
InstanceID	CreationClassName	PreferredLifetime
CommunicationStatus	SystemCreationClassName	SkipAsSource
DetailedStatus	SystemName	ValidLifetime
HealthState	NameFormat	PSComputerName
InstallDate	OtherTypeDescription	CimClass
Name	ProtocolIFType	CimInstanceProperties
OperatingStatus	ProtocolType	CimSystemProperties
OperationalStatus	Address	

List ip interfaces – data entity attributes

ifIndex	NeighborDiscoverySupported	AdvertiseDefaultRoute
ifAlias	ManagedAddressConfiguration	ForceArpNdWolPattern
Store	OtherStatefulConfiguration	DirectedMacWolPattern
AddressFamily	WeakHostSend	EcnMarking
Forwarding	WeakHostReceive	Dhcp
Advertising	IgnoreDefaultRoutes	ConnectionState
NeighborUnreachabilityDetection		AutomaticMetric
RouterDiscovery		Caption
		Description

ElementName	*TimeOfLastStateChange*	*BaseReachableTime*
InstanceID	*TransitioningToState*	*CompartmentId*
CommunicationStatus	*CreationClassName*	*CurrentHopLimit*
DetailedStatus	*SystemCreationClassName*	*DadRetransmitTime*
HealthState	*SystemName*	*DadTransmits*
InstallDate	*NameFormat*	*InterfaceAlias*
Name	*OtherTypeDescription*	*InterfaceIndex*
OperatingStatus	*ProtocolIFType*	*InterfaceMetric*
OperationalStatus	*ProtocolType*	*IsolationId*
PrimaryStatus	*AliasAddresses*	*LowestIfNetLuid*
Status	*GroupAddresses*	*NlMtu*
StatusDescriptions	*LANID*	*ReachableTime*
AvailableRequestedStates	*LANType*	*RetransmitTime*
EnabledDefault	*MACAddress*	*PSComputerName*
EnabledState	*MaxDataSize*	*CimClass*
OtherEnabledState	*OtherLANType*	*CimInstanceProperties*
RequestedState	*AdvertisedRouterLifetime*	*CimSystemProperties*

PING ICMPv4 echo request other machine with timestamp

```
ping -l 1200 -t 192.168.1.1|cmd /q /v /c "(pause&pause)>nul & for /l %a in
() do (set /p "data=" && echo(!date! !time! !data!)&ping -n 2
192.168.1.1>nul" >D:\ping_gwlan.txt
```

```
ping -l 1200 -t spilberk.vzmb.cz|cmd /q /v /c "(pause&pause)>nul & for /l
%a in () do (set /p "data=" && echo(!date! !time! !data!)&ping -n 2
spilberk.vzmb.cz>nul" >D:\ping_gwwan.txt
```

1.12. (local!) users management [ALTERNATIVE_MODULE_LINK]

List users

(PS5.1)

```
> Get-LocalUser
```

(PS4.0)

```
> Get-CimInstance win32_useraccount
```

```
> Get-WmiObject -Class Win32_UserAccount -Filter  "LocalAccount='True'"
```

```
> $adsi = [ADSI]"WinNT://$env:COMPUTERNAME"
> $adsi.Children | where {$_.SchemaClassName -eq 'user'} | Foreach-Object {
    $groups = $_.Groups() | Foreach-Object
{$_.GetType().InvokeMember("Name", 'GetProperty', $null, $_, $null)}
    $_ | Select-Object @{n='UserName';e={$_.Name}},@{n='Groups';e={$groups
-join ';'}}
}
```

(alternative module:)

```
> Get-LocalUser
```

Get locked out accounts

```
> Search-ADAccount -LockedOut
```

Get last password updates

```
> get-aduser -filter * -properties passwordlastset, passwordneverexpires
|ft Name, passwordlastset, Passwordneverexpires
```

```
> Get-ADUser -Filter * -SearchBase "OU=TestOU,DC=TestDomain,DC=Local"
|  Set-ADUser -PasswordNeverExpires:$True
```

```
> net USER
```

(note: the command “net” is case sensitive on the first parameter – in this case “USER”!!!!!)

```
dsquery user “OU={your target OU},DC={your domain},DC={your domain
extension}” | dsmod user -pwdneverexpires yes
```

(in ase of more than 100users)

```
dsquery user “OU={your target OU},DC={your domain},DC={your domain
extension}” -limit 2000 | dsmod user -pwdneverexpires yes
```

Create local user

(PS5.1) (alternative module:)

```
> New-LocalUser -Name "User02" -Description "Description of this account."
-NoPassword
```

```
> $Password = Read-Host -AsSecureString; New-LocalUser "User03" -Password
$Password -FullName "Third User" -Description "Description of this
account."
```

```
> New-LocalUser -Name "MicrosoftAccount\usr name@Outlook.com" -Description
"Description of this account."
```

(PS4.0)

```
# Create new local Admin user for script purposes
> $Computer = [ADSI]"WinNT://$Env:COMPUTERNAME,Computer"
$LocalAdmin = $Computer.Create("User", "LocalAdmin")
$LocalAdmin.SetPassword("Password01")
$LocalAdmin.SetInfo()
$LocalAdmin.FullName = "Local Admin by Powershell"
$LocalAdmin.SetInfo()
$LocalAdmin.UserFlags = 64 + 65536 # ADS_UF_PASSWD_CANT_CHANGE +
ADS_UF_DONT_EXPIRE_PASSWD
$LocalAdmin.SetInfo()
```

```
> NET USER username "password" /ADD
```

Modify/edit local user

```
> [ADSI]$userAccount="WinNT://$env:COMPUTERNAME/User_Name,User"
> [ADSI]$userAccount.SetPassword("P@ssw0rd")
```

(alternative module:)

```
> Set-LocalUser
```

```
> net USER username /FULLNAME:”full user name” /HOMEDIR:Path
/PASSWORDCHNG:NO /LOGONPASSWORDCHNG:NO /PASSWORDREQ:YES /PROFILEPATH:Path
/SCRIPTPATH /ACTIVE:YES /COMMENT:”comment text” /EXPIRES:mm/dd/rrrr
/USERCOMMEND:”coment”
```

Modify /edit local user – reset password

```
> [ADSI]$userAccount="WinNT://$env:COMPUTERNAME/User_Name,User"
$userAccount.PasswordAge/86400 //In days
```

```
> net USER username "password"
```

Modify/edit local user – change groups membership

```
> [ADSI]$theGroup="WinNT://$env:COMPUTERNAME/Group_name,group"
> [ADSI]$userAccount="WinNT://$env:COMPUTERNAME/User_Name,User"
> $theGroup.Remove($userAccount.Path)
> $theGroup.Add(]$userAccount)
```

(alternative module:)

```
> Add-LocalGroupMember
```

```
> NET LOCALGROUP "group" "user" /delete
```

```
> NET LOCALGROUP "group" "user" /add
```

Deactivate (block/disable) local user account

```
> $AccountDisable=0x0002
[ADSI]$userAccount="WinNT://$env:COMPUTERNAME/User_Name,User"
($userAccount.UserFlags.Value -band $AccountDisable) -as [boolean]
$new=$userAccount.UserFlags.Value -bor $AccountDisable
$userAccount.put("userflags",$new)
$userAccount.SetInfo()
```

(alternative module:)

```
> Disable-LocalUser
```

```
> net USER username /ACTIVE:NO
```

Reactivate (unblock/enable) local user account

```
> $AccountDisable=0x0002
[ADSI]$userAccount="WinNT://$env:COMPUTERNAME/User_Name,User"
($userAccount.UserFlags.Value -band $AccountDisable) -as [boolean]
$new=$userAccount.UserFlags.Value -bxor $AccountDisable
$userAccount.put("userflags",$new)
$userAccount.SetInfo()
```

(alternative module:)

```
> Enable-LocalUser
```

```
> net USER username /ACTIVE:YES
```

Delete local user

```
> [ADSI]$server="WinNT://$env:COMPUTERNAME "
[ADSI]$userAccount="WinNT://$env:COMPUTERNAME/User_Name,User"
$server.delete("user",$userAccount.name.value)
```

```
> NET LOCALGROUP "group" "user" /delete
```

1.13. (local!) groups management

List groups

```
> Get-WmiObject win32_group
```

```
> $ADSI = [ADSI]("WinNT://$env:COMPUTERNAME")
> $ADSI.psbase.children | where { $_.psbase.schemaClassName -eq 'group' }
```

List group members

```
> $ADSI = [ADSI]("WinNT://$env:COMPUTERNAME")
> $ADSI.psbase.children.find(„Administrators”)
> $group.psbase.invoke("members")
```

List ALL groups members

```
> $ADSI = [ADSI]("WinNT://$env:COMPUTERNAME");
> $list=@();$ADSI.psbase.children | where { $_.psbase.schemaClassName -eq
'group' } | foreach {$group =[ADSI]$_.psbase.Path;
$group.psbase.Invoke("Members") | foreach {$us =
$_.GetType().InvokeMember("Name", 'GetProperty', $null, $_, $null); $list
+= new-object psobject -property @{Group = $group.Name;User=$us}};$list
|ft -a
```

Create group

```
> $ADSI = [ADSI]("WinNT://$env:COMPUTERNAME")
> $Group = $ADSI.Create('Group', 'TestGroupName')
```

Modify/change group

```
> $ADSI = [ADSI]("WinNT://$env:COMPUTERNAME")
> $Group = $ADSI.Children.Find('GROUP_NAME', 'group')
> $Group.Description  = 'This is a  test group for whatever'
> $Group.SetInfo()
```

Modify/edit group – change groups membership

```
> [ADSI]$theGroup="WinNT://$env:COMPUTERNAME/Group_name,group"
> [ADSI]$userAccount="WinNT://$env:COMPUTERNAME/User_Name,User"
> $theGroup.Remove($userAccount.Path)
> $theGroup.Add(]$userAccount)
```

```
> NET LOCALGROUP "group" "user" /add
```

Delete group

```
> $ADSI = [ADSI]("WinNT://$env:COMPUTERNAME")
> $Group = $ADSI.Children.Find('GROUP_NAME', 'group')
> $ADSI.Children.Remove($Group)
```

1.14. (local!) certificates management

List certificates

```
cd Cert:
> cd Cert:\\CurrentUser
> cd Cert:\\LocalMachine
```

Import certificates into specified branch

1.15. Folder management

List folders

```
> Get-ChildItem
> Get-ChildItem -Recurse
```

```
> dir
```

List folders only

```
> Get-ChildItem | ?{ $_.PSIsContainer }
> Get-ChildItem -Recurse | ?{ $_.PSIsContainer } | Select-Object FullName
> dir -Directory
```

```
> dir /ad
```

Change current folder location

```
> Set-Location -Path C:\
> Set-Location -Path HKLM:\SOFTWARE
> cd .\MyFolderName
> cd Registry::
```

```
> Push-Location
> Pop-Location
```

Edit/modify folder attributes

```
> $dir = Get-Item D:\FolderDirectory
> $dir.Attributes = 'ReadOnly, Hidden'
```

Directory	The file is a directory.
Encrypted	The file or directory is encrypted. For a file, this means that all data in the file is encrypted. For a directory, this means that encryption is the default for newly created files and directories.
Hidden	The file is hidden, and thus is not included in an ordinary directory listing.
IntegrityStream	The file or directory includes data integrity support. When this value is applied to a file, all data streams in the file have integrity support. When this value is applied to a directory, all new files and subdirectories within that directory, by default, include integrity support.
NoScrubData	The file or directory is excluded from the data integrity scan. When this value is applied to a directory, by default, all new files and subdirectories within that directory are excluded from data integrity.
ReadOnly	The file is read-only.

Modify folder permissions – list permissions

```
> Get-Acl .\foldernamehere | select -expand access | FL *
```

```
> icacls foldername
```

Modify folder permissions – list effective permissions

```
> icacls foldername
```

```
> Get-Acl -Path "foldername"
```

Modify folder permissions – store/restore permissions

```
> icacls FILENAME_ALSO_WITH_* /save aclfile /t
> icacls FOLDER_WITH_THE_FILES /restore aclfile
```

```
> $DogACL = Get-Acl -Path "foldername"
> Set-Acl -Path "differentfoldername" -AclObject $DogACL
```

Modify folder permissions – prevent/remove inheritance

```
> icacls folder_file /inheritance:r //Removes and disables all inherited
```

```
> icacls folder_file /inheritance:d //Disables inheritance, but copies
settings
```

Modify folder permissions – allow inheritance

```
> icacls folder_file /inheritance:e
> icacls folder_file /reset
```

Modify folder permissions – Modify/add/remove permissions

```
> $acl=get-acl filename
> $account=New-Object System.Security.Principal.NTAccount("Builtin",
"Administrators")
> $colRights =[System.Security.AccessControl.FileSystemRights]"Read, Write"
> $InheritanceFlag = [System.Security.AccessControl.InheritanceFlags]::None
> $PropagationFlag = [System.Security.AccessControl.PropagationFlags]::None
> $objType =[System.Security.AccessControl.AccessControlType]::Allow

> $rule=new-object -TypeName
System.Security.AccessControl.FileSystemAccessRule    ($account,
$colRights, $InheritanceFlag, $PropagationFlag, $objType)

> $acl.SetAccessRule($rule)
```

$colRights: ListDirectory, ReadData, WriteData, CreateFiles, CreateDirectories, AppendData, ReadExtendedAttributes, WriteExtendedAttributes, Traverse, ExecuteFile, DeleteSubdirectoriesAndFiles, ReadAttributes, WriteAttributes, Write, Delete, ReadPermissions, Read, ReadAndExecute, Modify, ChangePermissions, TakeOwnership, Synchronize, FullControl

$InheritanceFlag: ContainerInherit (the ACE is inherited by child containers, like subfolders), ObjectInherit (the ACE is inherited by child objects, like files), None
$PropagationFlag: NoPropagateInherit (the ACE is not propagated to child objects), InheritOnly (the ACE is propagated to all child objects), None

$objType =[System.Security.AccessControl.AccessControlType]::Allow; [System.Security.AccessControl.AccessControlType]::Deny

```
> wmic useraccount where name='username' get sid
> wmic useraccount where name='%username%' get sid
> wmic useraccount where (name='administrator' and domain='%userdomain%')
get name,sid
```

```
> icacls folder_file /grant User1:(d,wdac)
> icacls folder_file /grant *SID-1-1-0:(d,wdac)
```

```
> icacls folder_file /deny User1:(d,wdac)
> icacls folder_file /deny *SID-1-1-0:(d,wdac)
```

```
> icacls folder_file  /remove:g *SID-1-1-0 //removes all DENY for the SID
> icacls folder_file  /remove:d *SID-1-1-0 //removes all DENY for the SID
```

```
> icacls folder_file /grant:r "USER_OR_GROUP":(D)(OI)(CI) (removes all
grants and than applies the specified)
> icacls folder_file /deny "USER_OR_GROUP":(D)(OI)(CI) (removes all denials
and then applies the specified)
```

F (full access); **M** (modify access); **RX** (read and execute access); **R** (read-only access); **W** (write-only access);

D (delete); **RC** (read control); **WDAC** (write DAC); **WO** (write owner); **S** (synchronize); **AS** (access system security); **MA** (maximum allowed); **GR** (generic read); **GW** (generic write); **GE** (generic execute); **GA** (generic all); **RD** (read data/list directory); **WD** (write data/add file); **AD** (append data/add subdirectory); **REA** (read extended attributes); **WEA** (write extended attributes); **X** (execute/traverse); **DC** (delete child); **RA** (read attributes); **WA** (write attributes)

(OI): object inherit; **(CI)**: container inherit; **(IO)**: inherit only; **(NP)**: do not propagate inherit

Modify folder permissions – list owner

```
> Get-Acl $Path | Select Path, Owner
```

```
> dir foldername /q
> dir /q
```

Modify folder permissions – modify owner

```
> $ACL = Get-ACL file_folder_name
> $Account = New-Object System.Security.Principal.NTAccount("Builtin",
"Administrators")
> $ACL.SetOwner($Group)
> Set-Acl -Path file_folder_name -AclObject $ACL
```

```
> icacls folder_file /setowner User1
```

```
> takeown /F <filename>
```

Create folder

```
> mkdir folderNnameorPath
```

Rename folder

```
> move originalName newName
```

Delete folder

```
> del folderNameOrPath
```

Delete folder recursively

```
> del –R folderNameOrPath
```

Move/Copy folder

```
> copy sourceFolder destFolder
```

Junction

junction.rar

Junction – query the file/folder if it is hard link

```
> junction c:\PathFileName
```

Junction – create hard link

```
> mkdir FolderThatIsTheLink
> junction FolderThatIsTheLink "c:\LinkedFolder"
```

Junction – delete hard link

```
> junction -d FolderThatIsTheLink
```

1.16. Files management

Print working directory

```
> Pwd
> Get-CurrentLocation
```

List folders/files

```
> Get-ChildItem
> Get-ChildItem -Recurse //CAREFULLY IN ROOT FOLDERS!!!!!
> Dir -Files
> dir
```

Change current folder location

```
> Set-Location -Path C:\
> Set-Location -Path HKLM:\SOFTWARE
> cd .\MyFolderName
> cd Registry::
```

```
> Push-Location
> Pop-Location
```

Read text file

```
> Get-Content c:\file.txt
> Get-Content c:\file.txt | Select-Object -last 5
```

```
> more c:\file.txt
```

Read text file – piping lines as input into other command

```
> Get-Content c:\scripts\test.txt | Foreach-Object {Get-Wmiobject -
computername $_ win32_bios}
```

Input / add of content into a file

```
> new-item -path c:scriptsnewfile.txt -value "this file was created on 12/3." -itemtype file
```

```
> add-content C:\file.txt "NEW TEXT LINE"
> add-content c:\scripts\*.htm "<p><i>some additional text into all files</i>"
```

Replace file content

```
> set-content c:\scripts\*.htm "<p>replace entire content in all the htmls</p>!"
```

```
> (Get-Content .\input.txt ).Replace('text','fun') | Out-File .\output.txt
```

Edit file content

```
> notepad C:\file.txt //☹
```

Edit file attributes

```
> $file = Get-Item D:\test.txt
> $file.Attributes = 'ReadOnly, Hidden'
```

Archive	The file is a candidate for backup or removal.
Compressed	The file is compressed.
Device	Reserved for future use.
Directory	The file is a directory.
Encrypted	The file or directory is encrypted. For a file, this means that all data in the file is encrypted. For a directory, this means that encryption is the default for newly created files and directories.
Hidden	The file is hidden, and thus is not included in an ordinary directory listing.
IntegrityStream	The file or directory includes data integrity support. When this value is applied to a file, all data streams in the file have integrity support. When this value is applied to a directory, all new files and subdirectories within that directory, by default, include integrity support.
Normal	The file is a standard file that has no special attributes. This attribute is valid only if it is used alone.
NoScrubData	The file or directory is excluded from the data integrity scan. When this value is applied to a directory, by default, all new files and subdirectories within that directory are excluded from data integrity.
NotContentIndexed	The file will not be indexed by the operating system's content indexing service.
Offline	The file is offline. The data of the file is not immediately available.
ReadOnly	The file is read-only.
ReparsePoint	The file contains a reparse point, which is a block of user-defined data associated with a file or a directory.
SparseFile	The file is a sparse file. Sparse files are typically large files whose data consists of mostly zeros.
Systém	The file is a system file. That is, the file is part of the operating system or is used exclusively by the operating system.
Temporary	The file is temporary. A temporary file contains data that is needed while an application is executing but is not needed after the application is finished. File systems try to keep all the data in memory for quicker access rather than

flushing the data back to mass storage. A temporary file should be deleted by the application as soon as it is no longer needed.

Modify file permissions

- See Modify folder permissions

Move/Copy/Delete file

- See Rename/Move/Copy/Delete folder

1.17. Backup machine

Backup machine

```
-
```

Schedule a machine backup

```
-
```

Restore machine

```
-
```

1.18. Shares management

List provided shares

```
> Get-SmbShare
```

```
> Get-WmiObject -Class Win32_Share
```

List connected shares

```
> Get-PSDrive
```

Connect a share (tmp/persistent)

```
> New-PSDrive -Name P -PSProvider FileSystem -Root \\server\share -
Credential domain\user
```

-Persist = persistent mapping

```
> $net = new-object -ComObject WScript.Network
> $net.MapNetworkDrive("u:", "\\server\share", $false, "domain\user",
"password")
```

$false/$true = update user profile (true = persistent mapping)

Disconnect a share

```
net use driveletter: /delete
```

Create a share

```
> New-SMBShare -Name "Shared" -Path "D:\Tmp" `
 -FullAccess BUILTIN\administrators `
 -ChangeAccess domain\deptusers `
 -ReadAccess "domain\authenticated users" `
 -ContinuouslyAvailable:$true
```

Edit a share

```
> Get-SmbShare –Name “ShareName” | Set-SmbShare  –FullAccess
BUILTIN\administrators -ChangeAccess domain\deptusers -ReadAccess
“domain\authenticated users”
```

For Folder permissions, which have to be in accordance with the network permissions see „Folder management – change folder permissions!

Delete a share

```
> Get-SmbShare –Name “ShareName” | Remove-SmbShare
```

Backup provided shares settings

```
> reg export
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Lanmanserver\Shares
file.reg /y
```

Restore provided shares settings

```
> reg import file_from_previous_script.reg
```

Backup connected shares settings

```
> reg export HKEY_CURRENT_USER\Network c:\temp\drives.reg
```

Restore connected shares settings

```
> reg import file_from_previous_script.reg
```

SmbShare – data entity attributes

PresetPathAcl	*ContinuouslyAvailable*	*Special*
ShareState	*CurrentUsers*	*Temporary*
AvailabilityType	*Description*	*Volume*
ShareType	*EncryptData*	*PSComputerName*
FolderEnumerationMode	*Name*	*CimClass*
CachingMode	*Path*	*CimInstanceProperties*
SmbInstance	*Scoped*	*CimSystemProperties*
CATimeout	*ScopeName*	
ConcurrentUserLimit	*SecurityDescriptor*	
	ShadowCopy	

PSDrive – data entity attributes

Used	*Name*	*Credential*
Free	*Provider*	*DisplayRoot*
CurrentLocation	*Root*	
	Description	

1.19. Windows features management

List installed features

```
> Get-WindowsFeature | where-object {$_.Installed -eq $True}
```

List available features

```
(> Import-Module ServerManager)
> Get-WindowsFeature
```

Install a feature

```
> Add-WindowsFeature -Name Backup-Features -IncludeAllSubFeature:$true -
Restart:$false
```

Uninstall a feature

```
(> Import-Module ServerManager)
> Get-WindowsFeature | where-object {$_.Installed -eq $True}
> Uninstall-WindowsFeature -Name Web-Server
```

Uninstall all features

```
> Get-WindowsFeature | Where-Object -FilterScript { $_.Installed -Eq $TRUE
} | Uninstall-WindowsFeature
```

Windows feature – data entity attributes

Name	*Path*	*SystemService*
DisplayName	*Depth*	*Notification*
Description	*DependsOn*	*BestPracticesModelId*
Installed	*Parent*	*EventQuery*
InstallState	*ServerComponentDescriptor*	*PostConfigurationNeeded*
FeatureType	*SubFeatures*	*AdditionalInfo*

1.20. Programs install/uninstall

List installed programs

```
> Get-ItemProperty
HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\* |  Select-
Object DisplayName, DisplayVersion, Publisher, InstallDate | Format-Table –
AutoSize
```

Modify(change)/Repair a program installation

```
-
```

Uninstall a program

```
-
```

1.21. Scheduler management [CMD_LINK]

List scheduled tasks

```
> Get-ScheduledTask | FL *
```

```
> schtasks /query
```

Create scheduled task

```
> $A = New-ScheduledTaskAction -Execute "Taskmgr.exe"
> $T = New-ScheduledTaskTrigger -AtLogon
> $P = New-ScheduledTaskPrincipal "Contoso\Administrator"
> $S = New-ScheduledTaskSettingsSet
> $D = New-ScheduledTask -Action $A -Principal $P -Trigger $T -Settings $S
> Register-ScheduledTask T1 -InputObject $D
```

```
> schtasks /create /sc <ScheduleType> /tn <TaskName> /tr <TaskRun> [/ru
{[<Domain>\]<User> | System}] [/rp <Password>] [/mo <Modifier>] [/d
<Day>[,<Day>...] | *] [/m <Month>[,<Month>...]] [/i <IdleTime>] [/st
<StartTime>] [/ri <Interval>] [{/et <EndTime> | /du <Duration>} [/k]] [/sd
<StartDate>] [/ed <EndDate>] [/it] [/z] [/f]
```

Edit scheduled task – names, account, commandline

```
> $A = New-ScheduledTaskAction -Execute "Taskmgr.exe"
> $T = New-ScheduledTaskTrigger -AtLogon
> $P = New-ScheduledTaskPrincipal "Contoso\Administrator"
> $S = New-ScheduledTaskSettingsSet
> $D = Set-ScheduledTask -Action $A -Principal $P -Trigger $T -Settings $S
```

```
> schtasks /change /tn <TaskName> [/ru {[<Domain>\]<User> | System}] [/rp
<Password>] [/tr <TaskRun>] [/st <StartTime>] [/ri <Interval>] [{/et
<EndTime> | /du <Duration>} [/k]] [/sd <StartDate>] [/ed <EndDate>]
[/{ENABLE | DISABLE}] [/it] [/z]
```

Start scheduled task

```
> Start-ScheduledTask -TaskName "ScanSoftware"
```

```
> schtasks /run /tn <TaskName>
```

Stop scheduled task

```
> Stop-ScheduledTask -TaskName "ScanSoftware"
```

```
> schtasks /end /tn <TaskName>
```

Block scheduled task

```
> Disable-ScheduledTask
> Enable-ScheduledTask
```

Delete scheduled task

```
> Remove-ScheduledTask
```

```
> schtasks /delete /tn {<TaskName>
```

Backup scheduled tasks

```
> Get-ScheduledTask -TaskPath "\UpdateTasks\" | Export-ScheduledTask
```

```
> schtasks /query /tn "Task Name" /xml > "c:\exported.xml"
```

Restore scheduled tasks

```
> Register-ScheduledTask -Xml (Get-Content "c:\export.xml" | out-string) -
TaskName "Task Name"
```

```
> schtasks /create /tn "Task Name" /xml "c:\exported.xml"
```

ScheduledTask – data entity attributes

State
Actions
Author
Date
Description
Documentation

Principal
SecurityDescriptor
Settings
Source
TaskName
TaskPath
Triggers

URI
Version
PSComputerName
CimClass
CimInstanceProperties
CimSystemProperties

1.22. Eventlog management

List/browse events

```
> Get-EventLog -LogName "Application"
```

"Application", "System", "Security"

Export events

```
$objExcel = New-Object -comobject Excel.Application
$objExcel.visible = $True
$objWorkbook = $objExcel.Workbooks.Add()
$objSheet = $objWorkbook.Worksheets.Item(1)
$objSheet.Cells.Item(1,1) = "Server"
$objSheet.Cells.Item(1,2) = "LogName"
$objSheet.Cells.Item(1,3) = "Time"
$objSheet.Cells.Item(1,4) = "Source"
$objSheet.Cells.Item(1,5) = "Message"
$objSheetFormat = $objSheet.UsedRange
$objSheetFormat.Interior.ColorIndex = 19
$objSheetFormat.Font.ColorIndex = 11
$objSheetFormat.Font.Bold = $True

$row = 1

$servers = gc servers.txt

foreach ($server in $servers)
{
  $row = $row + 1
  $AppLog = Get-EventLog -LogName Application -EntryType Error -computer $server -
Newest 5
  $SecLog = Get-EventLog -LogName Security -EntryType Error -computer $server -
Newest 5 -ea Silentlycontinue
  $SysLog = Get-EventLog -LogName System -EntryType Error -computer $server -Newest
5
  foreach ($Cat in $AppLog,$Syslog,$Seclog)
  {
    if ($cat -is [array])
    {
      if ($AppLog -contains $cat[0]) {$Catname = "Application"}
      if ($SecLog -contains $cat[0]) {$Catname = "Security"}
      if ($SysLog -contains $cat[0]) {$Catname = "System"}
      Foreach ($event in $cat)
      {
        $objSheet.Cells.Item($row,1).Font.Bold = $True
        $objSheet.Cells.Item($row,1) = $server
        $objSheet.Cells.Item($row,2) = $Catname
        $objSheet.Cells.Item($row,3) = $Event.TimeGenerated
        $objSheet.Cells.Item($row,4) = $Event.Source
        $objSheet.Cells.Item($row,5) = $Event.Message
        $row = $row + 1
      }
    }
  }
}

$objSheetFormat = $objSheet.UsedRange
$objSheetFormat.EntireColumn.AutoFit()
$objSheetFormat.RowHeight = 15
```

Clear events

```
> Clear-EventLog
```

EventLog entity attributes

EventID	*MachineName*	*Data*

Index *Category* *CategoryNumber* *EntryType* *Message*	*Source* *ReplacementStrings* *InstanceId* *TimeGenerated* *TimeWritten*	*UserName* *Container* *Site*

1.23. Registry management

Change path – enter registry

```
> Cd HKLM:\
> CD HKCU:\
> CD Registry::\
```

List a branch

```
> Get-ChildItem
> dir
```

Navigate through branches

```
> Cd
> Set-Location
```

Create registry folder/branch

```
> New-Item –Path . –Name NewBranchName
```

Create registry key (value)

```
> New-Item –Path . –Name NewKeyName –Force –Value default_value
```

Edit folder/branch name

-

Edit value

```
> Set-Item -Path HKCU:\Software\NewKeyName -Value "hsg key"
```

Delete folder/branch

```
> Remove-Item KeyOrBranchName
```

Delete value

```
> Remove-Item KeyOrBranchName
```

Export (copy) folder/branch

```
> reg export
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Lanmanserver\Shares
file.reg /y
```

Import (copy) folder/branch

```
> reg import file_from_previous_script.reg
```

Backup entire registry

```
-
```

Restore entire registry

```
-
```

1.24. Microsoft Active Directory – Workstation Group Policy Objects (GPO) management

Workstation GPO Update

```
> Invoke-GPUpdate
```

```
> secedit /refreshpolicy
> gpupdate /force /logoff /boot
```

(Správce zásad skupiny)

```
> gpmc.msc
```

Resulting policies display GUI

```
> rsop.msc
```

2. Servers only

2.1. Mgmt GUI / Core switching

FULL GUI ON

```
> Install-WindowsFeature server-gui-shell [-restart]
```

MGMT ONLY GUI ON

```
> Install-WindowsFeature server-gui-mgmt-infra –restart
```

```
> Install-WindowsFeature server-gui-mgmt-infra -restart -source
wim:d:\sources\install.wim:2
```

- Index :1 - Windows Server 2012 R2 SERVERSTANDARDCORE
- Index :2 - Windows Server 2012 R2 SERVERSTANDARD
- Index :3 - Windows Server 2012 R2 SERVERDATACENTERCORE
- Index :4 - Windows Server 2012 R2 SERVERDATACENTER

MGMT GUI OFF

```
> Uninstall-WindowsFeature server-gui-mgmt-infra –restart
```

FULL GUI OFF

```
> Uninstall-WindowsFeature server-gui-shell –restart
```

2.2. Microsoft Active Directory – Directory Services management

SIDwalker	Security Administration Tools. Consists of 3 programs, showaccs.exe, sidwalk.exe and Security Migration Editor (MMC snap-in). First two used to examine and change ACL entries. Security Migration Editor edits mappings between old and new security IDs (SIDs).
repadmin.exe	Replication Diagnostics Tool. Check replication consistency between partners, status, force replication events and knowledge consistency checker recalculation.
acldiag.exe	ACL Diagnostics. Used to determine whether users have been granted/denied access to AD objects. Can be used to reset Access Control Lists to their default values.
ADSI edit	Low-level editor for Active Directory which enables adding, moving, and deleting objects within Active Directory.
dfsutil.exe	Distributed File System Utility. Manages all aspects of the distributed file system.
dnscmd.exe	DNS Server Troubleshooting Tool. Check dynamic registration of DNS resource records including secure DNS update and unregister resource records.
dsacls.exe	View or modify ACLs of objects in AD.
nltest.exe	Create a list of PDCs, force a shutdown, provide info about trusts and replication.
dsastat.exe	Active Directory Diagnostic Tool. Compare naming contexts on Domain Controllers and detect differences.
ldp.exe	Allows LDAP operations be be performed against Active Directory.

movetree.exe	AD Object Manager. Move AD objects like OUs and users between domains in a single forest.
netdom.exe	Enables administrators to manage Active Directory domains and trust relationships from the command prompt.
replmon.exe	Active Directory Replication Monitor. Graphically displays replication topology, monitor status, force replication and knowledge consistency checker recalculation.
sdcheck.exe	Security Descriptor Check Utility. Verify ACL propagation and replication for specified objects in a directory.

https://www.microsoftpressstore.com/articles/article.aspx?p=2217264

Data store architecture

After you examine the operating system components that support Active Directory, the next step is to see how directory data is stored on a domain controller's hard disks. As Figure 10-7 shows, the data store has a primary data file and several other types of related files, including working files and transaction logs.

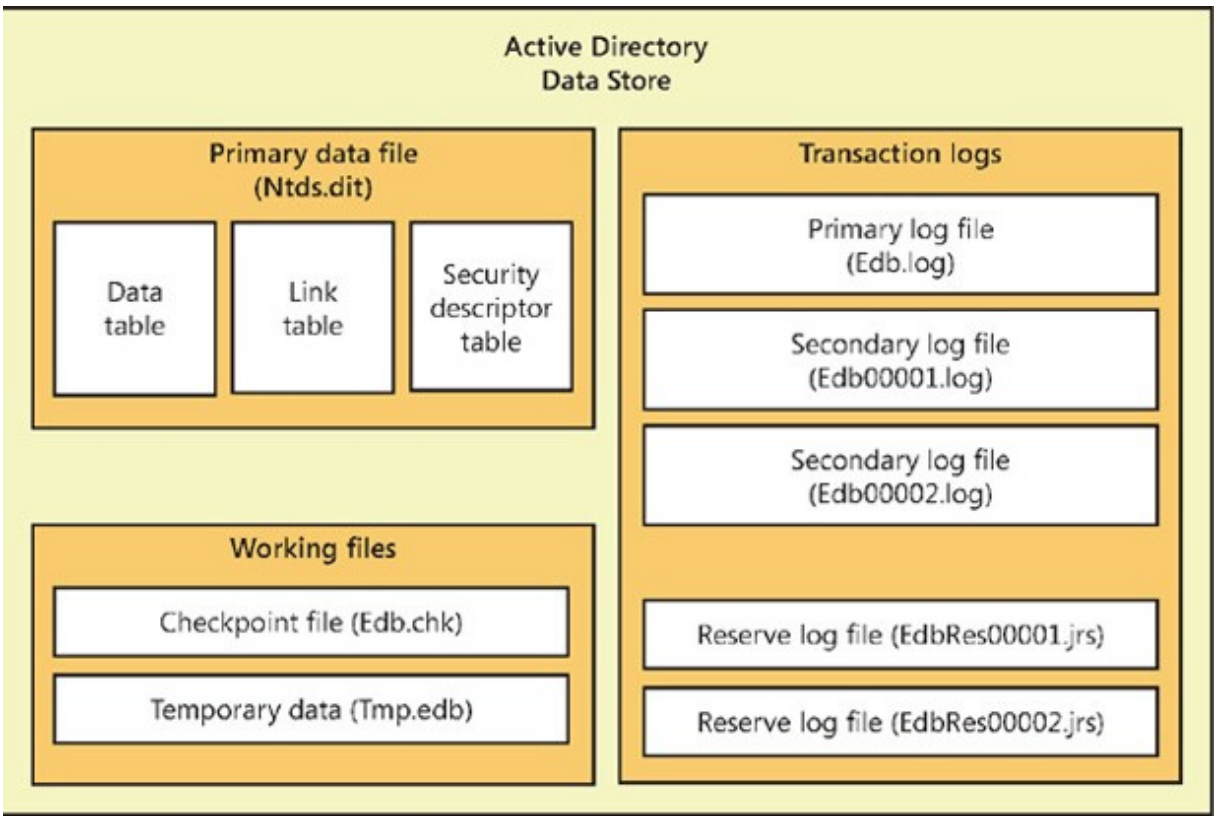


Figure 10-7 The Active Directory data store.

These files are used as follows:

- **Primary data file (Ntds.dit).** Physical database file that holds the contents of the Active Directory data store

- **Checkpoint file (Edb.chk).** Checkpoint file that tracks the point up to which the transactions in the log file have been committed to the database file
- **Temporary data (Tmp.edb).** Temporary workspace for processing transactions
- **Primary log file (Edb.log).** Primary log file that contains a record of all changes that have yet to be committed to the database file
- **Secondary log files (Edb00001.log, Edb00002.log, ...).** Additional logs files that are used as needed
- **Reserve log files (EdbRes00001.jrs, EdbRes00002.jrs, ...).** Files that are used to reserve space for additional log files if the primary log file becomes full

The primary data file contains three indexed tables:

- **Active Directory data table.** The data table contains a record for each object in the data store, which can include object containers, the objects themselves, and any other type of data that is stored in Active Directory.
- **Active Directory link table.** The link table is used to represent linked attributes. A linked attribute is an attribute that refers to other objects in Active Directory. For example, if an object contains other objects (that is, it is a container), attribute links are used to point to the objects in the container.
- **Active Directory security descriptor table.** The security descriptor table contains the inherited security descriptors for each object in the data store. Windows Server uses this table so that inherited security descriptors no longer have to be duplicated on each object. Instead, inherited security descriptors are stored in this table and linked to the appropriate objects. This makes Active Directory authentication and control mechanisms very efficient.

2.2.1. Install

```
> install-windowsfeature AD-Domain-Services
> Import-Module ADDSDeployment
```

2.2.2. Uninstall

```
> Test-ADDSDomainControllerUninstallation
```

```
> Uninstall-ADDSDomainController
```

2.2.3. DC Promote PRI – new domain

```
> Test-ADDSDomainInstallation //First in a new forest install
> Install-ADDSForest //First in a new forest
```

```
> Install-ADDSForest -CreateDnsDelegation:$false -DomainMode "Win2012R2" -
DomainName "yourdomain.com" -DomainNetbiosName "YOURDOMAIN" -ForestMode
"Win2012R2" -InstallDns:$true -DatabasePath "C:\Windows\NTDS"
-LogPath "C:\Windows\NTDS" -SysvolPath "C:\Windows\SYSVOL"
-NoRebootOnCompletion:$false -Force:$true
```

-DomainMode:
Windows Server 2003: **2** or **Win2003**
Windows Server 2008: **3** or **Win2008**
Windows Server 2008 R2: **4** or **Win2008R2**
Windows Server 2012: **5** or **Win2012**
Windows Server 2012 R2: **6** or **Win2012R2**

2.2.4. DC Promote PRI – sub-domain

```
> Test-ADDSDomainInstallation
> Install-ADDSDomain //First in a sub-domain
```

```
> Install-ADDSDomain -Credential (Get-Credential DOMAIN\EnterpriseAdmin1) -
NewDomainName subdomainName -ParentDomainName top.level.domain.com -
InstallDNS -CreateDNSDelegation -DomainMode Win2003 -ReplicationSourceDC
DC1.corp.contoso.com -SiteName Houston -DatabasePath "D:\NTDS" -SYSVOLPath
"D:\SYSVOL" -LogPath "E:\Logs" -NoRebootOnCompletion
```

2.2.5. DC Promote SEC

```
> Test-ADDSDomainControllerInstallation
```

```
> Install-ADDSDomainController
```

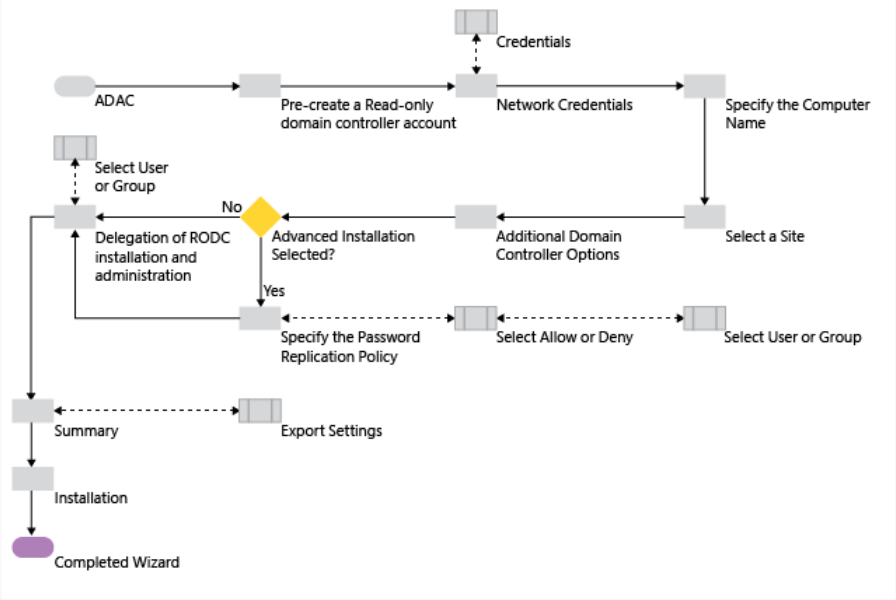
```
> Install-ADDSDomainController -InstallDns -DomainName "corp.contoso.com"
```

2.2.6. DC promote read-only in a domain [HOWTO_LINK]

```
> adprep /rodcprep
```

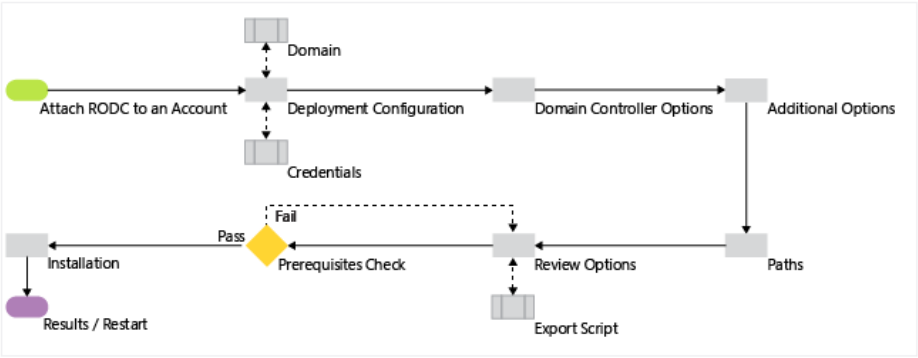
A staged read only domain controller (RODC) installation works in two discrete phases:

1. Staging an unoccupied computer account
2. Attaching an RODC to that account during promotion



https://docs.microsoft.com/cs-cz/windows-server/identity/ad-ds/deploy/rodc/install-a-windows-server-2012-active-directory-read-only-domain-controller--rodc---level-200-

```
> Add-addsreadonlydomaincontrolleraccount -DomainControllerAccountName XXXX
-DomainName YYY.YYY.YYY -SiteName SSSSSSS -Credential FFFFFF
```



https://docs.microsoft.com/cs-cz/windows-server/identity/ad-ds/deploy/rodc/install-a-windows-server-2012-active-directory-read-only-domain-controller--rodc---level-200-

```
> Install-AddsDomaincontroller -DomainName YYY.YYY.YYY. -UseExistingAccount
RRRRR -DatabasePath "D:\DATA" -LogPath "D:\DATA" -SYSVOLPath "D:\DATA" -
ReplicationSourceDC DC1.YYYY.YYY.YYYY
```

2.2.7. DCs upgrade schemas for newer version of domains

```
> adprep /forestprep
> adprep /domainprep /gpprep
```

Note: The 'adprep' from the newer OS installation DVD has to be used.

```
> $currentForest = Get-ADForest
> Set-ADForestMode -Identity $currentForest -Server
$currentForest.SchemaMaster -ForestMode Windows2008R2Forest
```

-ForestMode:

- Windows2000Forest
- Windows2003InterimForest
- Windows2003Forest
- Windows2008Forest
- Windows2008R2Forest
- UnknownForest

```
> Set-ADDomainMode -Identity fabrikam.com -DomainMode Windows2003Domain
```

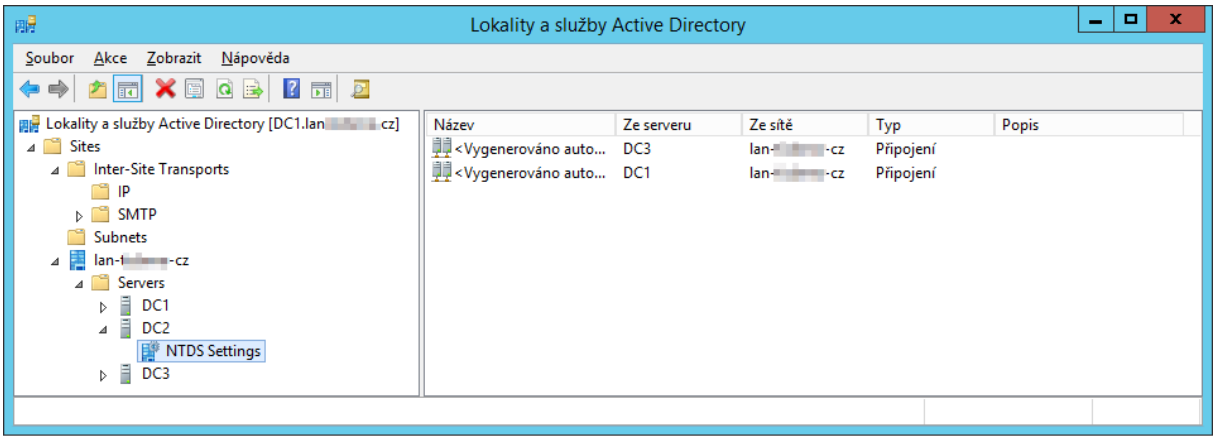
OR

```
> $pdc = Get-ADDomainController -Discover -Service PrimaryDC
> Set-ADDomainMode -Identity $pdc.Domain -Server $pdc.HostName[0] -
DomainMode Windows2003Domain
```

-DomainMode

- Windows2000Domain or 0
- Windows2003InterimDomain or 1
- Windows2003Domain or 2
- Windows2008Domain or 3
- Windows2008R2Domain or 4

2.2.8. DCs status [CHECKS_LINK] [PERFMON_LINK]




```
> Get-ADComputer ComputerNameDomain
```

```
> Invoke-Command -ComputerName DC1 -ScriptBlock {Get-WinEvent -LogName
'Microsoft-Windows-DSC/Operational' -MaxEvents 50} | Select-Object
PSComputerName,TimeCreated,LevelDisplayName,Id,Message | Out-Gridview
```

```
> net stop "dns client" & net start "dns client" //In order not do satisfy DNS
tests from the local cache
> dcdiag /a /replsource:DC1 /s:DC1
```

https://blogs.technet.microsoft.com/askds/2011/03/22/what-does-dcdiag-actually-do/

```
> netdiag /v
```

```
> ldp
```

```
> ntdsutil [DOC] [DOC]
```

```
> netdom [DOC]
```

```
> dsmod
> dsquery

> dsget computer - displays properties of computers in the directory.
> dsget contact - displays properties of contacts in the directory.
> dsget subnet - displays properties of subnets in the directory.
> dsget group - displays properties of groups in the directory.
> dsget ou - displays properties of ou's in the directory.
> dsget server - displays properties of servers in the directory.
> dsget site - displays properties of sites in the directory.
> dsget user - displays properties of users in the directory.
> dsget quota - displays properties of quotas in the directory.
> dsget partition - displays properties of partitions in the directory.
```

2.2.9. DCs replication status [repadmin LINK]

```
> repadmin /showreps

> repadmin /replsum
> repadmin /replsum /bysrc /bydest /sort:delta
```

2.2.10. MS AD Services Backup

Backup MS AD SYSVOL

```
> Import-Module ServerManager
[string]$date = get-date -f 'yyyy-MM-dd'
$TargetUNC = "D:\Backup"
If (Test-Path $TargetUNC) {Remove-Item -Path $TargetUNC -Recurse -Force}
New-Item -ItemType Directory -Force -Path $TargetUNC
$WBadmin_cmd =  "wbadmin.exe START BACKUP -backupTarget:$TargetUNC -
systemState -noverify -vssCopy -quiet -user:MyUser -password:MyPassword "
Invoke-Expression $WBadmin_cmd
```

```
> wbadmin.exe START BACKUP -backupTarget:"D:\Backup" -systemState -noverify
-vssCopy -quiet -user:MyUser -password:MyPassword
```

Restore MS AD SYSVOL

```
> wbadmin start systemstaterecovery –backupTarget:"D:\Backup"
```

2.3. Microsoft Active Directory – Directory Services Time Server management

For full reference of the NTP client see: 1.7. Services management – NTP client (time sync service)

```
> w32tm
```

(Run cmd as Administrator!!!!)

Get current config

```
> w32tm /query /configuration
```

Change config - manual

```
> w32tm /config /update /syncfromflags:MANUAL
/manualpeerlist:192.53.103.108,192.53.103.104
```

Change config – get from domain DC

```
> w32tm /config /syncfromflags:DOMHIER
```

Change config – get from none

```
> w32tm /config /syncfromflags:NO
```

Change config – get from all

```
> w32tm /config /syncfromflags:ALL
```

Change config – set, that this computer is a reliable datetime source

```
> w32tm /config /reliable:YES
```

Display timezone settings

```
> w32tm /tz
```

Backup settings

```
> w32tm /dumpreg
```

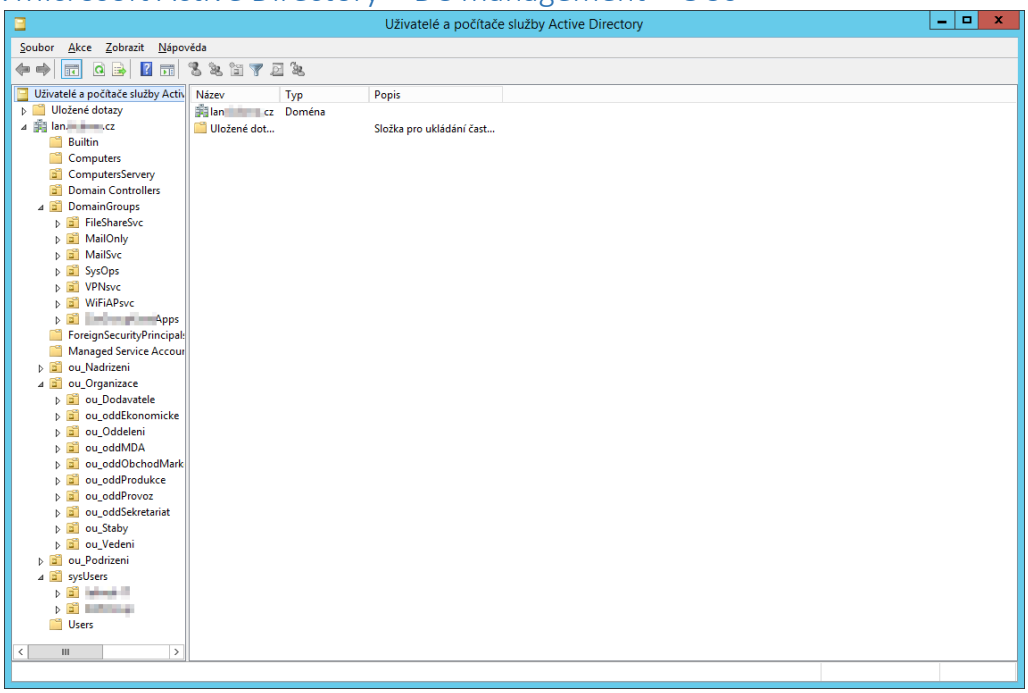
Service stop

```
> net stop W32Time
```

Service start

```
> net start W32Time
```

2.4. Microsoft Active Directory – DC management – OUs



List Domain

```
> Get-ADDomain domain.name.com
> Get-ADDomain -Current LocalComputer
```

```
> Get-ADRootDSE
```

List AD objects in AD - OUs

```
> Get-ADOrganizationalUnit -Filter 'Name -like "*"' | FT Name,
DistinguishedName -A
```

List OUs

```
> Get-ADOrganizationalUnit -Identity
'OU=AsiaPacific,OU=Marketing,OU=Users,DC=MYDOMAINNAME,DC=COM' | ft
Name,Country,PostalCode,City,StreetAddress,State -A
```

Create OUs

```
> New-ADOrganizationalUnit -Name UserAccounts -Path
"DC=MYDOMAINNAME,DC=COM"
```

```
> New-ADOrganizationalUnit -Name UserAccounts -Path
"DC=MYDOMAINNAME,DC=COM" -ProtectedFromAccidentalDeletion $false
```

```
> $ouTemplate = Get-ADOrganizationalUnit
"OU=UserAccounts,DC=MYDOMAIN,DC=COM" -properties seeAlso,managedBy; New-
ADOrganizationalUnit -name TomCReports -instance $ouTemplate
```

Create other objects

```
> New-ADObject -Name '192.168.1.0/26' -Type subnet -Description
'192.168.1.0/255.255.255.192' -OtherAttributes @{location="Building
A";siteObject="CN=HQ,CN=Sites,CN=Configuration,DC=FABRIKAM,DC=COM"} -Path
"CN=Subnets,CN=Sites,CN=Configuration,DC=MYDOMAINNAME,DC=COM"
```

```
> $subnetTemplate = get-adobject -Identity
"CN=192.168.1.0/26,CN=Subnets,CN=Sites,CN=Configuration,DC=mydomainname,DC=
com" -properties description,location; new-adobject -instance
$subnetTemplate -name "192.168.1.0/28" -type subnet -path
"CN=Subnets,CN=Sites,CN=Configuration,DC=MYDOMAINNAME,DC=COM"
```

```
> New-ADObject -name SaraDavisContact -type contact -
ProtectedFromAccidentalDeletion $true -OtherAttributes @{'msDS-
SourceObjectDN'="CN=FabrikamContacts,DC=MYDOMAINNAME,DC=COM"}
```

Edit OUs

```
> Set-ADOrganizationalUnit -Identity
"OU=UserAccounts,DC=MYDOMAINNAME,DC=COM" -Description "This Organizational
Unit holds all of the users accounts of FABRIKAM.COM"
```

```
> Set-ADOrganizationalUnit -Identity
"OU=UserAccounts,DC=MYDOMAINNAME,DC=COM" -ProtectedFromAccidentalDeletion
$false
```

Edit OUs – rename OU, rename CN

```
> Rename-ADObject "OU=ManagedGroups,OU=Managed,DC=Fabrikam,DC=Com" -NewName
Groups
```

```
> Rename-ADObject "CN=Apps,DC=AppNC" -NewName "InternalApps"
```

Move OUs

```
> Get-ADOrganizationalUnit -Identity
'OU=AsiaPacific,OU=Sales,OU=UserAccounts,DC=MYDOMAINNAME,DC=COM' | Move-
ADObject -TargetPath 'OU=Finance,DC=mydomainname,Dc=com'
```

```
> Move-ADObject 'CN=Brad Sutton,CN=Users,DC=mydomainname,DC=com' -
TargetPath 'OU=Accounting,DC=mydomainname,DC=com'
```

Delete OUs, CNs

```
> Remove-ADOrganizationalUnit -Identity
"OU=Accounting,DC=MYDOMAINNAME,DC=COM"
```

```
> Remove-ADOrganizationalUnit -Identity
"OU=Accounting,DC=MYDOMAINANME,DC=COM" -Recursive
```

```
> Remove-ADObject -Identity "CN=InternalApps,DC=AppNC" -server "MYSERVER-
SRV1:60000"
```

Edit OUs – list permissions

```
> $Name = "OU=xxx,DC=com"
> $ADObject = [ADSI]"LDAP://$Name"
```

```
> $aclObject = $ADObject.psbase.ObjectSecurity
> $aclList =
$aclObject.GetAccessRules($true,$true,[System.Security.Principal.SecurityId
entifier])
> $output=@()

foreach($acl in $aclList)
      {
        $objSID = New-Object
System.Security.Principal.SecurityIdentifier($acl.IdentityReference)

        $info = @{
          'ActiveDirectoryRights' = $acl.ActiveDirectoryRights;
          'InheritanceType' = $acl.InheritanceType;
          'ObjectType' = $acl.ObjectType;
          'InheritedObjectType' = $acl.InheritedObjectType;
          'ObjectFlags' = $acl.ObjectFlags;
          'AccessControlType' = $acl.AccessControlType;
          'IdentityReference' = $acl.IdentityReference;
          'NTAccount' = $objSID.Translate(
[System.Security.Principal.NTAccount] );
          'IsInherited' = $acl.IsInherited;
          'InheritanceFlags' = $acl.InheritanceFlags;
          'PropagationFlags' = $acl.PropagationFlags;
        }

        $obj = New-Object -TypeName PSObject -Property $info
        $output+=$obj}
$output
```

Edit OUs – change permissions

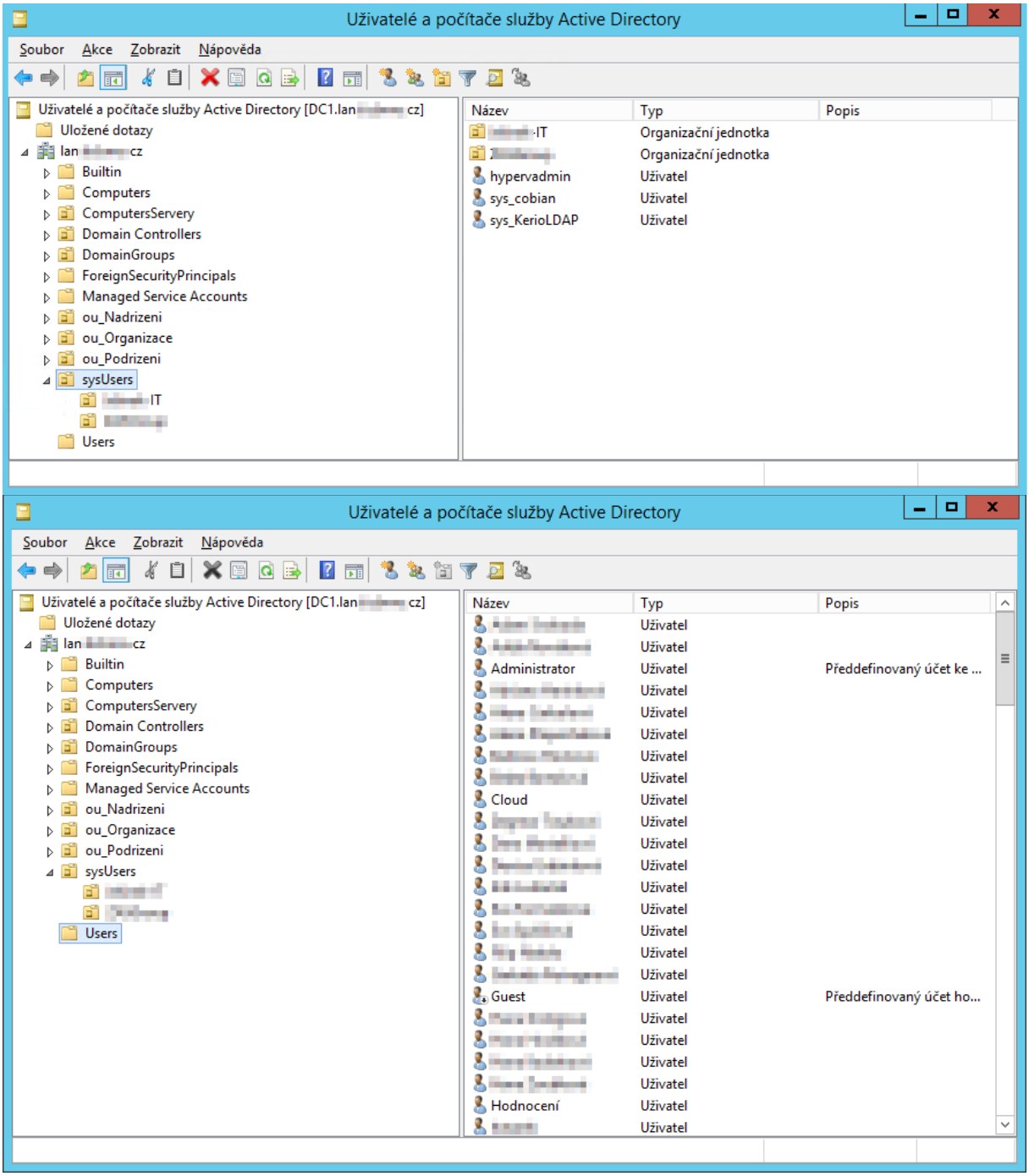
```
> $GroupSID = [System.Security.Principal.SecurityIdentifier]'S-1-1-0'
#Everyone Group

$objACL = Get-ACL "AD:\\OU=xxx,DC=com"
$objACE = New-Object
System.DirectoryServices.ActiveDirectoryAccessRule($GroupSID,"DeleteChild",
"Deny", 'None', [guid]'00000000-0000-0000-0000-000000000000')
$objACL.AddAccessRule($objACE)
Set-acl -AclObject $objACL "AD:${OU}
```

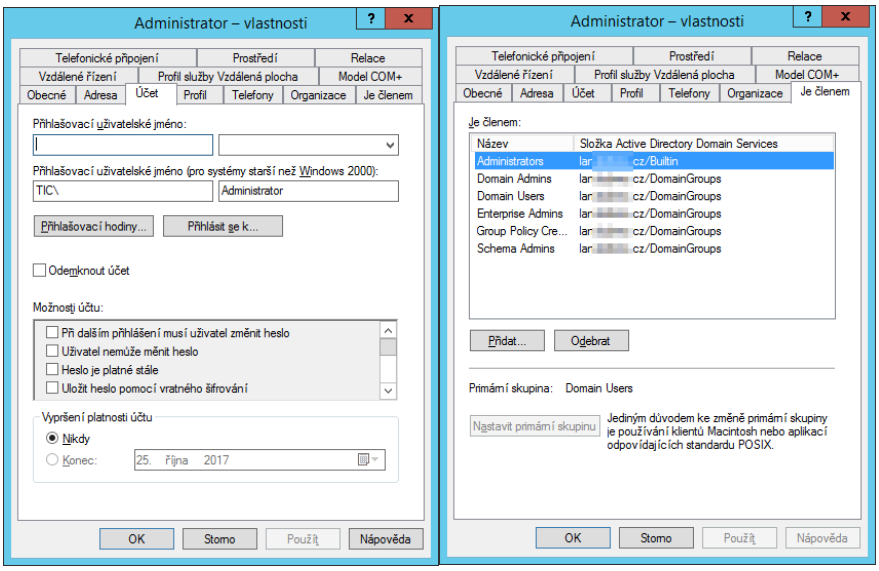
In addition, if you want to assign other permissions, you can change the GUID values in the script above. The common GUID values are listed as below:

$guidChangePassword	= new-object Guid ab721a53-1e2f-11d0-9819-00aa0040529b
$guidLockoutTime	= new-object Guid 28630ebf-41d5-11d1-a9c1-0000f80367c1
$guidPwdLastSet	= new-object Guid bf967a0a-0de6-11d0-a285-00aa003049e2
$guidComputerObject	= new-object Guid bf967a86-0de6-11d0-a285-00aa003049e2
$guidUserObject	= new-object Guid bf967aba-0de6-11d0-a285-00aa003049e2
$guidLinkGroupPolicy	= new-object Guid f30e3bbe-9ff0-11d1-b603-0000f80367c1
$guidGroupPolicyOptions	= new-object Guid f30e3bbf-9ff0-11d1-b603-0000f80367c1
$guidResetPassword	= new-object Guid 00299570-246d-11d0-a768-00aa006e0529
$guidGroupObject	= new-object Guid BF967A9C-0DE6-11D0-A285-00AA003049E2
$guidContactObject	= new-object Guid 5CB41ED0-0E4C-11D0-A286-00AA003049E2
$guidOUObject	= new-object Guid BF967AA5-0DE6-11D0-A285-00AA003049E2
$guidPrinterObject	= new-object Guid BF967AA8-0DE6-11D0-A285-00AA003049E2
$guidWriteMembers	= new-object Guid bf9679c0-0de6-11d0-a285-00aa003049e2
$guidNull	= new-object Guid 00000000-0000-0000-0000-000000000000
$guidPublicInformation	= new-object Guid e48d0154-bcf8-11d1-8702-00c04fb96050
$guidGeneralInformation	= new-object Guid 59ba2f42-79a2-11d0-9020-00c04fc2d3cf
$guidPersonalInformation	= new-object Guid 77B5B886-944A-11d1-AEBD-0000F80367C1
$guidGroupMembership	= new-object Guid bc0ac240-79a9-11d0-9020-00c04fc2d4cf

OUs – data entity attributes		
City Country DistinguishedName LinkedGroupPolicyObjects ManagedBy	Name ObjectClass ObjectGUID PostalCode State StreetAddress PropertyNames	AddedProperties RemovedProperties ModifiedProperties PropertyCount

2.5. Microsoft Active Directory – DC management – (domain!) Users





List users (in all OUs)

```
> Get-ADUser
```

```
> Search-ADAccount -AccountDisabled | FT Name,ObjectClass -A
> Search-ADAccount -AccountDisabled -UsersOnly | FT Name,ObjectClass -A
> Search-ADAccount -AccountExpiring -TimeSpan 6.00:00:00 | FT
Name,ObjectClass -A
> Search-ADAccount -PasswordExpired | FT Name,ObjectClass -A
> Search-ADAccount -LockedOut | FT Name,ObjectClass -A
> Search-ADAccount -AccountDisabled -ComputersOnly | FT Name,ObjectClass -A
> Search-ADAccount -AccountExpiring -DateTime "3/18/2009" | FT
Name,ObjectClass -A
> Search-AdAccount -AccountDisabled -SearchBase "DC=AppNC" -Server
"MYSERVER-SRV1:60000"
```

List users in an OU

```
> Get-ADUser -Filter 'Name -like "*"' -SearchBase
"OU=Finance,OU=UserAccounts,DC=MYDOMAINNAME,DC=COM" | Disable-ADAccount
```

Create user in a OU

```
> Set-ADUser [-Identity] <ADUser> …(follow in Edit user)
```

Edit user

```
> Set-ADUser [-Identity] <ADUser> [-AccountExpirationDate
<System.Nullable[System.DateTime]>] [-AccountNotDelegated
<System.Nullable[bool]>] [-Add <hashtable>] [-
AllowReversiblePasswordEncryption <System.Nullable[bool]>] [-
CannotChangePassword <System.Nullable[bool]>] [-Certificates <hashtable>]
[-ChangePasswordAtLogon <System.Nullable[bool]>] [-City <string>] [-Clear
<string[]>] [-Company <string>] [-Country <string>] [-Department <string>]
[-Description <string>] [-DisplayName <string>] [-Division <string>] [-
EmailAddress <string>] [-EmployeeID <string>] [-EmployeeNumber <string>] [-
Enabled <System.Nullable[bool]>] [-Fax <string>] [-GivenName <string>] [-
HomeDirectory <string>] [-HomeDrive <string>] [-HomePage <string>] [-
HomePhone <string>] [-Initials <string>] [-LogonWorkstations <string>] [-
Manager <ADUser>] [-MobilePhone <string>] [-Office <string>] [-OfficePhone
<string>] [-Organization <string>] [-OtherName <string>] [-
PasswordNeverExpires <System.Nullable[bool]>] [-PasswordNotRequired
<System.Nullable[bool]>] [-POBox <string>] [-PostalCode <string>] [-
```

```
ProfilePath <string>] [-Remove <hashtable>] [-Replace <hashtable>] [-
SamAccountName <string>] [-ScriptPath <string>] [-ServicePrincipalNames
<hashtable>] [-SmartcardLogonRequired <System.Nullable[bool]>] [-State
<string>] [-StreetAddress <string>] [-Surname <string>] [-Title <string>]
[-TrustedForDelegation <System.Nullable[bool]>] [-UserPrincipalName
<string>] [-AuthType {<Negotiate> | <Basic>}] [-Credential <PSCredential>]
[-Partition <string>] [-PassThru <switch>] [-Server <string>] [-Confirm] [-
WhatIf] [<CommonParameters>]
```

Edit user – reset password

```
> Set-ADAccountPassword -Identity saradavi
```

```
> Set-ADAccountPassword 'CN=UserName,OU=Accounts,DC=mydomainname,DC=com' -
Reset -NewPassword (ConvertTo-SecureString -AsPlainText "p@ssw0rd" -Force)
```

```
> Set-ADAccountPassword -Identity UserName -OldPassword (ConvertTo-
SecureString -AsPlainText "p@ssw0rd" -Force) -NewPassword (ConvertTo-
SecureString -AsPlainText "qwert@12345" -Force)
```

```
> Set-ADAccountPassword [-Identity] <ADAccount> [-AuthType {<Negotiate> |
<Basic>}] [-Credential <PSCredential>] [-NewPassword <SecureString>] [-
OldPassword <SecureString>] [-Partition <string>] [-PassThru <switch>] [-
Reset <switch>] [-Server <string>] [-Confirm] [-WhatIf]
[<CommonParameters>]
```

Edit user – change groups membership

```
> Get-AdgroupMember -Identity administrators
```

```
> Add-ADGroupMember SomeGroupName User1,User2
```

```
> Get-ADGroup -SearchBase "OU=AccountDeptOU,DC=AppNC" -filter { name -like
"AccountLeads" } | Add-ADGroupMember -Members
"CN=SanjayPatel,OU=AccountDeptOU,DC=AppNC"
```

```
> Remove-Adgroupmember -Identity "ADDomaiNGroupName" -Member
"usrAccountName"
```

Edit user – rename account

```
> Rename-ADObject -Identity
"CN=HQ,CN=Sites,CN=Configuration,DC=MYDOMAINNAME,DC=COM" -NewName
UnitedKingdomHQ
```

Move user into different OU

```
> Get-ADUser SaraDavis | Move-ADObject -TargetPath
'OU=Finance,DC=MYDOMAINNAME,Dc=com'
```

Clear user account expiration

```
> Clear-ADAccountExpiration -Identity User1
```

Unlock (enable) user account

```
> Enable-ADAccount -Identity User1
```

```
> Enable-ADAccount -Identity "CN=Kim
Abercrombie,OU=Finance,OU=UserAccounts,DC=MYDOMAINNAME,DC=COM"
```

```
> Get-ADUser -Filter 'Name -like "*"' -SearchBase
"OU=Finance,OU=UserAccounts,DC=MYDOMAINNAME,DC=COM" | Enable -ADAccount
```

Lock (disable) user account

```
> Disable-ADAccount [-Identity] <ADAccount> [-AuthType {<Negotiate> |
<Basic>}] [-Credential <PSCredential>] [-Partition <string>] [-PassThru
<switch>] [-Server <string>] [-Confirm] [-WhatIf] [<CommonParameters>]
```

```
> Disable-ADAccount -Identity User1
```

```
> Disable-ADAccount -Identity
"CN=UserName,OU=Finance,OU=UserAccounts,DC=MYDOMAINNAME,DC=COM"
```

```
> Get-ADUser -Filter 'Name -like "*"' -SearchBase
"OU=Finance,OU=UserAccounts,DC=FABRIKAM,DC=COM" | Disable-ADAccount
```

Delete (remove) user

```
> Remove-ADUser -Identity User1
```

```
> Search-ADAccount -AccountDisabled | where {$_.ObjectClass -eq 'user'} |
Remove-ADUser
```

```
> Remove-ADUser -Identity
"CN=UserName,OU=Finance,OU=UserAccounts,DC=MYDOMAIN,DC=COM"
```

```
> Remove-ADUser [-Identity] <ADUser> [-AuthType {<Negotiate> | <Basic>}] [-
Credential <PSCredential>] [-Partition <string>] [-Server <string>] [-
Confirm] [-WhatIf] [<CommonParameters>]
```

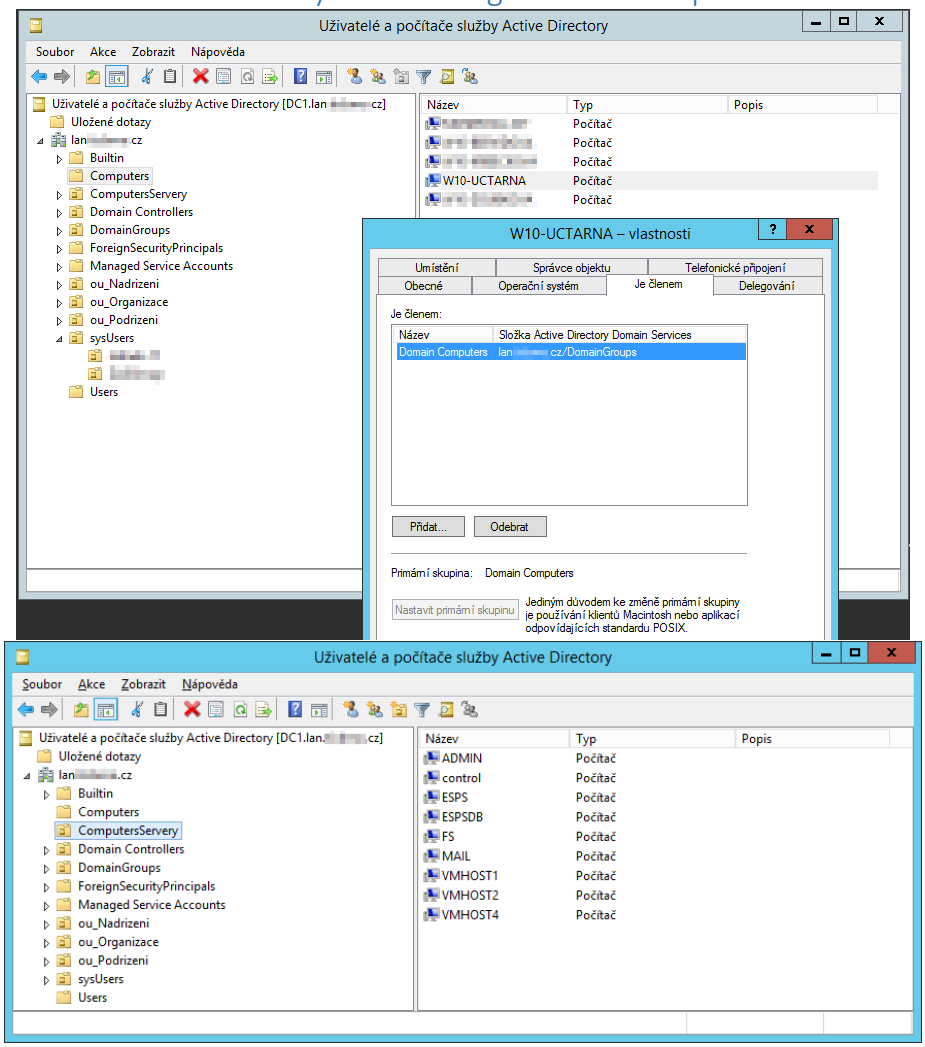
AD Users list – data entity attributes

DistinguishedName
Enabled
GivenName
Name
ObjectClass

ObjectGUID
SamAccountName
SID
Surname
UserPrincipalName
PropertyNames

AddedProperties
RemovedProperties
ModifiedProperties
PropertyCount

2.6. Microsoft Active Directory – DC management – Computers



List computers (in all OUs)

```
> Get-ADComputer
```

```
> Get-ADComputer -Filter 'Name -like "MYDOMAINNAM*"' -Properties
IPv4Address | FT Name,DNSHostName,IPv4Address -A
```

List computers in an OU

```
> Get-ADComputer -LDAPFilter "(name=*laptop*)" -SearchBase
"CN=Computers,DC=Mydomainnane,DC=com"
```

Edit/modify computer account

```
> Set-ADComputer "MYSERVER-SRV1" -Location "MAIN/Building A"
```

Move computer account into different OU

```
Get-ADComputer “XYZComputerName” | Move-ADObject -TargetPath
'OU=Finance,DC=MyDomainName,Dc=com'
```

Delete/Remove computer account

```
> Remove-ADComputer -Identity "SERVER-SRV4"
```

```
> Get-ADComputer -Filter 'Location -eq "MAIN/Building A"' | Remove-
ADComputer
```

AD Computer – entity attributes

DistinguishedName	*ObjectGUID*	*AddedProperties*
Enabled	*SamAccountName*	*RemovedProperties*
GivenName	*SID*	*ModifiedProperties*
Name	*Surname*	*PropertyCount*
ObjectClass	*UserPrincipalName*	
	PropertyNames	

2.7. Microsoft Active Directory – DC management – Domain Controllers

List servers (in alkl OUs)

```
> Get-ADDomainController -Discover -Site "Default-First-Site-Name"
> Get-ADDomainController -Discover -Service "GlobalCatalog"
> Get-ADDomainController -Discover
> Get-ADDomainController -Discover -Domain "corp.contoso.com" -Service
"PrimaryDC","TimeService"
> Get-ADDomainController -Identity "PDC-01"
> Get-ADDomainController "PDC-01"
> Get-ADDomainController -Identity "168.54.62.57"
```

List Forest

```
> Get-ADForest -Current LocalComputer
```

List servers in an OU

```
> Get-ADComputer -Filter * -SearchBase "OU=TheOUName, DC=mydomain, DC=com"
```

Note: the “Get-AdDomainController” does not implement the “-SearchBase” parameter.

Edit server

```
-
```

Move server into different OU

```
> Get-ADDomainController -Identity "PDC-01" | Move-ADObject -TargetPath
'OU=Finance,DC=MyDomainName,Dc=com'
```

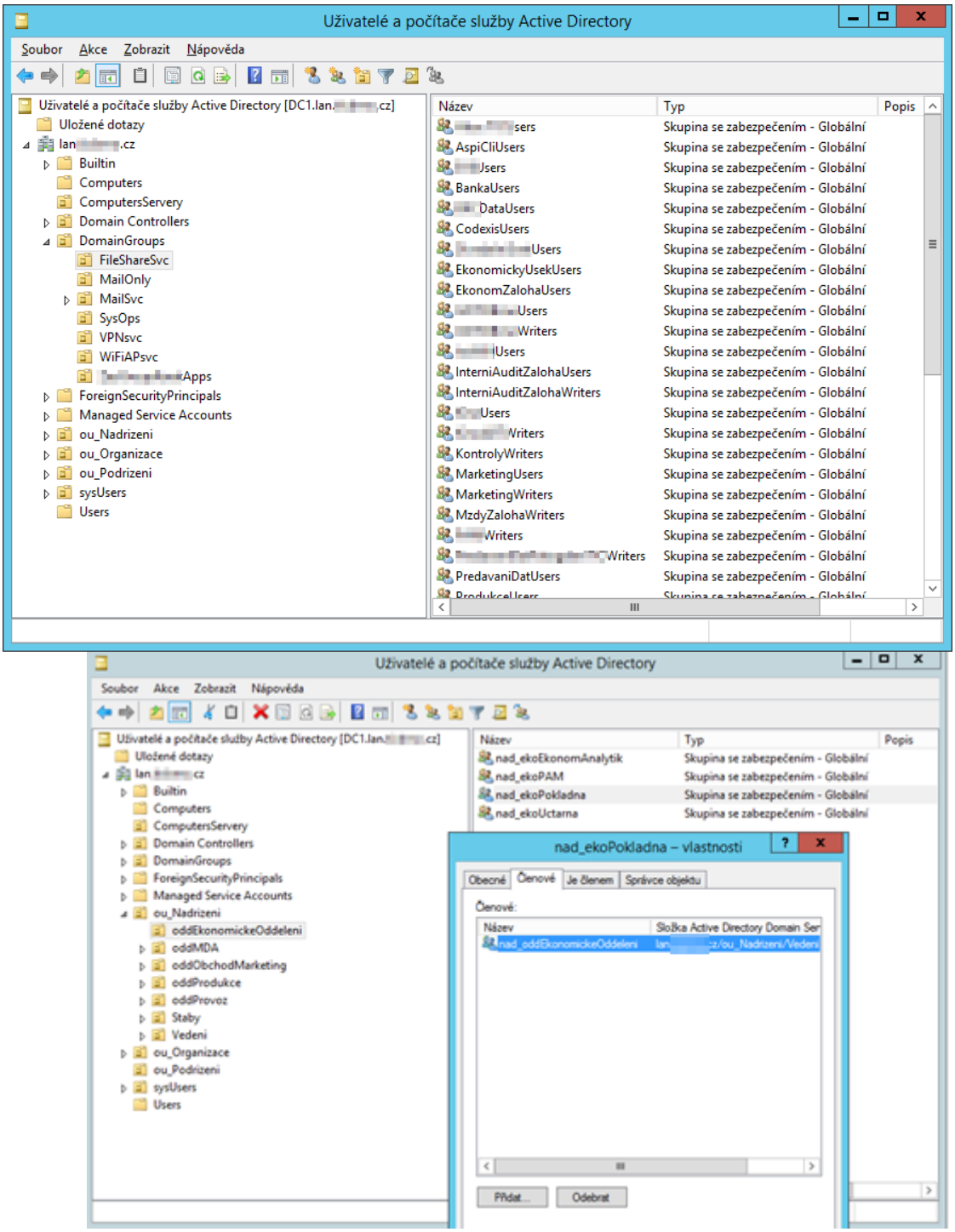
Delete server

```
-
```

AD Domain controller – entity attributes

ComputerObjectDN	*IsReadOnly*	*ServerObjectDN*
DefaultPartition	*LdapPort*	*ServerObjectGuid*
Domain	*Name*	*Site*
Enabled	*NTDSSettingsObjectDN*	*SslPort*
Forest	*OperatingSystem*	*PropertyNames*
HostName	*OperatingSystemHotfix*	*AddedProperties*
InvocationId	*OperatingSystemServiceP ack*	*RemovedProperties*
IPv4Address	*OperatingSystemVersion*	*ModifiedProperties*
IPv6Address	*OperationMasterRoles*	*PropertyCount*
IsGlobalCatalog	*Partitions*	

2.8. Microsoft Active Directory – DC management – (domain!) Groups



List groups (in all OUs)

```
> Get-ADGroup administrators
```

List groups in an OU

```
> Get-ADGroup -Filter * -SearchBase "OU=TheOUName, DC=mydomain, DC=com"
```

Create group

```
> New-ADGroup -Path "OU=AccountDeptOU,DC=AppNC" -Name AccountLeads -
GroupScope DomainLocal -GroupCategory Distribution
```

```
> Get-ADGroup GroupName -Properties Description | New-ADGroup  -Name
NewGroupName -SamAccountName Branch1Employees -GroupCategory Distribution -
PassThru
```

```
> New-ADGroup -Name "RODC Admins" -SamAccountName RODCAdmins -GroupCategory
Security -GroupScope Global -DisplayName "RODC Administrators" -Path
"CN=Users,DC=MyDomainName,DC=Com" -Description "Members of this group are
RODC Administrators"
```

Edit group

```
> Get-AdGroup -filter 'name -like "Access*"' | set-adgroup -description
"Access Group"
```

```
> get-adgroup -filter 'name -like "Access*"' | set-adgroup -description
"Access Group"
> $group = get-adgroup -server localhost:60000 -Identity
"CN=AccessControl,DC=AppNC"
> $group.description = "Access Group"
> set-adgroup -Instance $group -passthru
```

Edit group – change membership in groups / change member groups

```
> Get-AdgroupMember -Identity administrators
```

```
> Get-AdgroupMember "Enterprise Admins" -recursive
```

```
> Remove-AdGroupMember -Identity "GroupName" -Member "User1"
```

Move group into different OU

```
> Get-ADGroup administrators | Move-ADObject -TargetPath
'OU=Finance,DC=DomainName,Dc=com'
```

Delete group

```
> Remove-AdGroup ADGroupName1
```

```
> Get-AdGroup -filter 'Name -like "SomeUserGr*"' | remove-adgroup
```

AD Group – entity attributes

DistinguishedName	*ObjectClass*	*AddedProperties*
GroupCategory	*ObjectGUID*	*RemovedProperties*
GroupScope	*SamAccountName*	*ModifiedProperties*
Name	*SID*	*PropertyCount*
	PropertyNames	

2.9. Microsoft Active Directory – Certificates Services

Install authority

```
> Install-AdcsCertificationAuthority -AllowAdministratorInteraction -CAType
StandaloneRootCA -CACommonName "Example Internal Root CA" -CADistinguishedNameSuffix
"OU=PKI,O=`"Contoso, Ltd.`",C=US" -KeyLength 2048 -HashAlgorithmName SHA1 -
CryptoProviderName "RSA#Microsoft Software Key Storage Provider" -DatabaseDirectory
"C:\CertDB" -LogDirectory "C:\CertLog" -ValidityPeriod "Years" -ValidityPeriodUnits 20 –
Verbose
```

View user certificates

Create user certificates

List certificates

Edit certificate

Create certificate signing request

Sign certificate signing request

```
> certreq -submit -attrib "CertificateTemplate:WebServer" request.csr
```

2.9.1. Backup

Moving CA: https://technet.microsoft.com/en-us/library/ee126140%28v=ws.10%29.aspx#BKMK_BackUpDB

Backup MS AD CA

```
> certutil -backup -p PassWord D:\_CA_tempbackup
> reg export
HKLM\SYSTEM\CurrentControlSet\Services\CertSvc\ConfigurationOutputFile.reg
```

Restore MS AD CA

```
> certutil -restore -p PassWord <BackupDirectory>
```

2.10. Microsoft Active Directory – Group Policy Objects (GPO) management

2.10.1. Install addin

Install

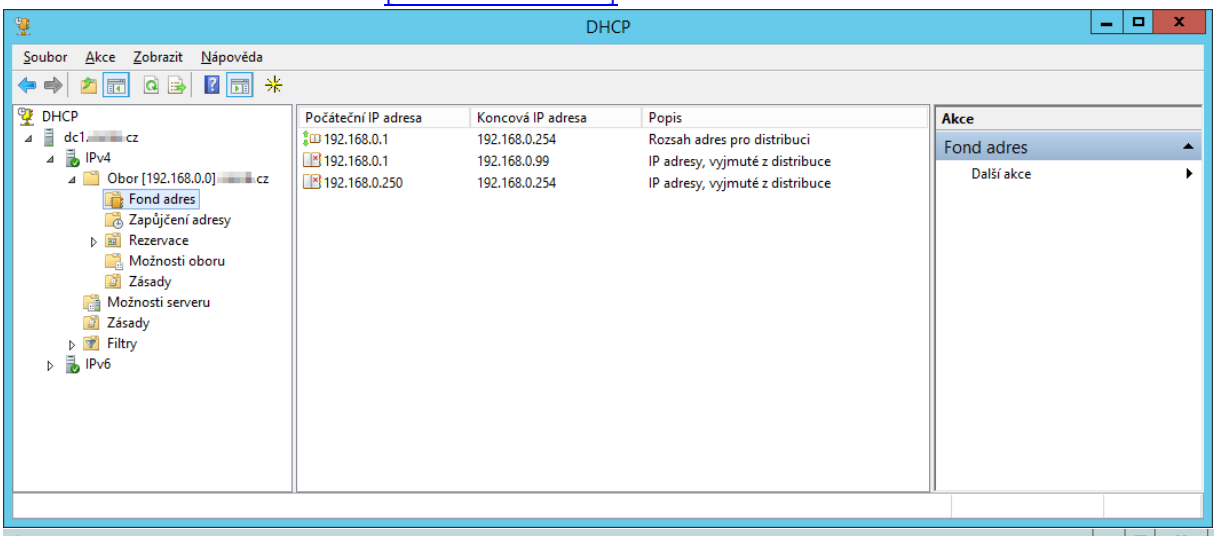
2.10.2. Update

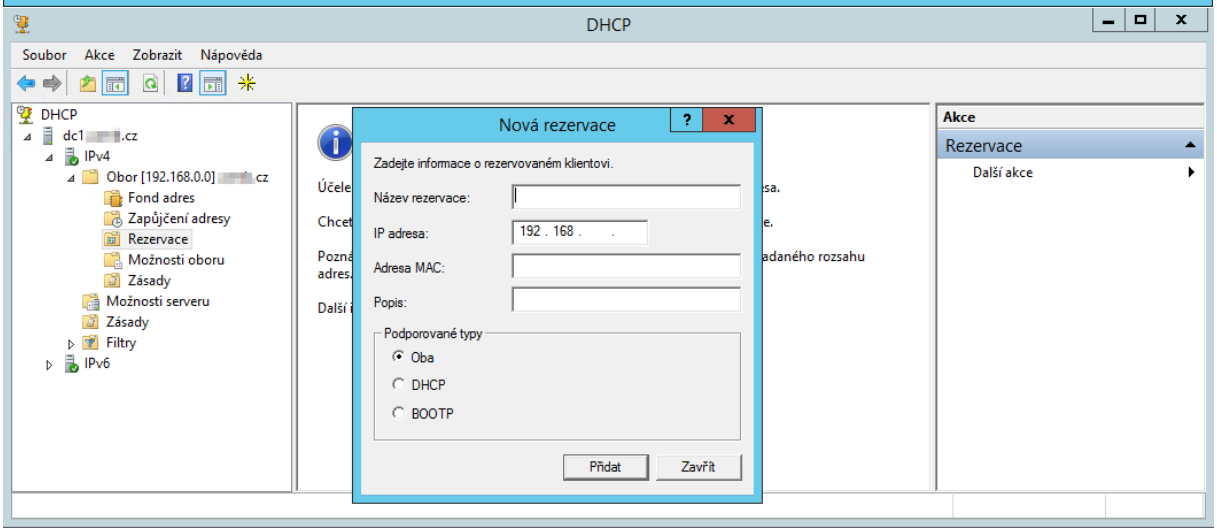
```
> Get-ADComputer -filter * -Searchbase "ou=Accounting, dc=Contoso,dc=com" |
foreach{ Invoke-GPUpdate -computer $_.name -force}
```

2.10.3. Edit

Create/Edit GPO - metadata
Create policy record
Edit policy record
Obsolete policy record
Delete policy record
Export policy settings
Backup policies
Restore policies

2.11. DHCP service [CMDLETS LINK]





Install service

```
> Add-WindowsFeature -IncludeManagementTools dhcp
```

```
> netsh dhcp add securitygroups
```

```
> Restart-service dhcpserver
```

Authorize service

```
> Add-DhcpServerInDC  <hostname of the DHCP server>  <IP address of the DHCP server>
```

```
> Set-ItemProperty -Path
registry::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\ServerManager\Roles\12 -
Name ConfigurationState -Value 2
```

Unauthorize service

```
> Remove-DhcpServerInDC -DnsName dhcpserver.contoso.com -IPAddress
10.10.10.2
```

(ip of the DHCP server)

List DHCP servers

```
> Get-DhcpServerInDC
```

Get DHCP server settings

```
> Get-DhcpServerSetting -ComputerName dhcpserver.contoso.com
```

List scopes

```
> Get-DhcpServerv4MulticastScope -Name "Multicast_AudioConference",
"Multicast_VideoConference" -ComputerName "DhcpServer01.Contoso.com"
```

```
> Get-DhcpServerv4MulticastScope -ComputerName "DhcpServer01.Contoso.com"
```

```
> Get-DhcpServerv4SuperScope -ComputerName dhcpserver.contoso.com
```

```
> Get-DhcpServerv4Policy -ComputerName dhcpserver.contoso.com
```

```
> Get-DhcpServerv4Policy -ComputerName dhcpserver.contoso.com -Name
HyperVPolicy -ScopeId 10.10.10.0
```

```
> Get-DhcpServerv4PolicyIPRange -ScopeId 10.10.10.0 -Name HyperVPolicy
```

```
> Get-DhcpServerv4PolicyIPRange -ScopeId 10.10.10.0
```

List/Show leases

```
> Get-DhcpServerv4Lease -ComputerName dhcpserver.contoso.com -ScopeId
10.10.10.0 -AllLeases
```

```
> Get-DhcpServerv4Lease -ComputerName dhcpserver.contoso.com -IPAddress
10.10.10.10,20.20.20.20
```

```
> Get-DhcpServerv4Lease -ComputerName dhcpserver.contoso.com -ScopeId
10.10.10.0 -BadLeases
```

```
> Get-DhcpServerv4Lease -ComputerName dhcpserver.contoso.com -ScopeId
10.10.10.0 -CliendId F0-DE-F1-7A-00-5E, 00-24-D7-C5-25-B0
```

```
> Get-DhcpServerv4Scope -ComputerName dhcpserver.contoso.com | Get-
DhcpServerv4Lease -ComputerName dhcpserver.contoso.com
```

Create zone (scope)

```
> Add-DhcpServerv4Scope -Name "Lab-4 Network" -StartRange 10.10.10.1 -
EndRange 10.10.10.254 -SubnetMask 255.255.255.0
```

```
> Add-DhcpServerv4MulticastScope -ComputerName "DhcpServer01.Contoso.com" -
Name "Multicast_AudioConference" -StartRange 224.0.0.0 -EndRange 224.0.0.30
-State Inactive -Ttl 20-LeaseDuration 20
```

```
> Add-DhcpServerv4Policy -Name HyperVPolicy -Condition OR -MacAddress
EQ,00155D*,000569*
```

```
> Add-DhcpServerv4Policy -Name PrinterPolicy -ScopeId 10.10.10.0 -Condition
OR -VendorClass EQ,"HP Printer","Xerox Printer"
```

Edit zone + advanced params

```
> Get-DhcpServerv4DnsSetting -ComputerName dhcpserver.contoso.com
```

```
> Get-DhcpServerv4OptionValue -ComputerName dhcpserver.contoso.com -ScopeId
10.10.10.0 -All
```

```
> Get-DhcpServerv4OptionValue -ComputerName dhcpserver.contoso.com -ScopeId
10.10.10.0 -Brief
```

Edit zone – add DHCP param

```
> Add-DhcpServerv4OptionDefinition -Name WPAD -OptionId 252 -Type String
```

```
> Add-DhcpServerv4OptionDefinition -Name UCIdentifier -OptionId 1 -Type
BinaryData -VendorClass MS-UC-Client -Description "UC Identifier"
```

```
> Set-DhcpServerv4OptionValue -ComputerName dhcpserver.contoso.com -
DnsServer 192.168.1.2 -WinsServer 192.168.1.3 -DnsDomain contoso.com -
Router 192.168.1.1 -Wpad http://proxy.contoso.com/wpad.dat
```

```
> Set-DhcpServerv4OptionValue -ComputerName dhcpserver.contoso.com -ScopeId
10.10.10.0 -DnsServer 192.168.1.2 -WinsServer 192.168.1.3 -DnsDomain
contoso.com -Router 192.168.1.1 -Wpad http://proxy.contoso.com/wpad.dat
```

```
> Set-DhcpServerv4OptionValue -ComputerName dhcpserver.contoso.com -
OptionId 6 -Value 192.168.1.1
```

OptionId: https://www.iana.org/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xhtml

Edit zone – remove DHCP param

```
> Remove-DhcpServerv4OptionValue -ComputerName dhcpserver.contoso.com -
ScopeId 10.10.10.0 -VendorClass MSUCClient -OptionId 5
```

```
> Remove-DhcpServerv4OptionValue -ComputerName dhcpserver.contoso.com -
OptionId 23
```

Edit zone – edit DHCP param

```
> Get-DhcpServerv4DnsSetting -ComputerName dhcpserver.contoso.com
```

Delete zone (scope)

```
> Remove-DhcpServerv4MulticastScope -Name "Multicast_VideoConference" -
ComputerName "DhcpServer01.Contoso.com"
```

```
> Get-DhcpServerV4Scope | Where-Object -FilterScript { $_.State -Eq
"Inactive" } | Remove-DhcpServerv4Scope -Force -Passthru
```

```
> Remove-DhcpServerv4Scope -ComputerName dhcpserver.contoso.com -ScopeId
10.10.10.0, 20.20.20.0
```

Delete lease

```
> Remove-DhcpServerv4Lease
```

List reserved record

```
> Get-DhcpServerv4Reservation -ComputerName dhcpserver.contoso.com -ScopeId
10.10.10.0.
```

```
> Get-DhcpServerv4Reservation -ComputerName dhcpserver.contoso.com -
IPAddress 10.10.10.5
```

Create reserved record

```
> Add-DhcpServerv4Reservation -ScopeId 10.10.10.0 -IPAddress 10.10.10.8 -
ClientId F0-DE-F1-7A-00-5E -Description "Reservation for Printer"
```

```
> Import-Csv -Path Reservations.csv | Add-DhcpServerv4Reservation -
ComputerName dhcpserver.contoso.com
```

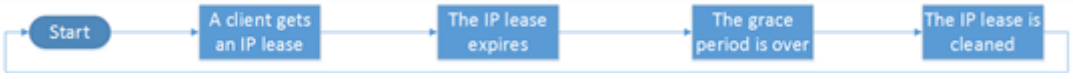
```
> Get-DhcpServerv4Lease -ComputerName dhcpserver.contoso.com -IPAddress
10.10.10.11 | Add-DhcpServerv4Reservation -ComputerName
dhcpserver.contoso.com
```

```
> $freeip = Get-DhcpServerv4FreeIPAddress -ComputerName
dhcpserver.contoso.com -ScopeId 10.10.10.0
> Add-DhcpServerv4Reservation -ComputerName dhcpserver.contoso.com -ScopeId
10.10.10.0 -IPAddress $freeip -ClientId F0-DE-F1-7A-00-5E -Description
"Reservation for Printer"
```

Start/Stop/restart service

DNS Integration

https://social.technet.microsoft.com/wiki/contents/articles/25089.dhcp-on-windows-servers-why-are-the-expired-ip-addresses-not-getting-re-assigned.aspx



HKLM/System/CurrentControlSet/Services/DHCPServer/Parameters

- **DatabaseCleanupInterval (DWORD)** = how often will be those records, that have Expired, be REALLY DELETED from the DB (an expired lease will not be shown in the MMC console!!!! But only in PS > Get-DHCPServerV4Lease –scopeid (scope_name_from_mmc) –AllLeases
  NOTE: As long as the lease is not really deleted from the db (eg. is only expired) it wont be deleted from the DNS, even if the option Scope->Properties->DNS->„Discard A and PTR records when lease is DELETED“ … that will happen during „DatabaseCleanupInterval“ and the ExpiryTime+(reg)LeaseExtension < datetime.now

- **LeaseExtension (DWORD)** = if not specified = 1440 minutes, if specified, it the record will be really DELETED from the DHCP DB during DatabaseCleanup after its ExpiredTime+LeaseExtension is in the past

NOTE: All "Authenticated Users" can do an create/update of child items in the DNS (incl. admin during adding) – but these records might have other owner, so the DHCP is than unable to deleted.

PARTNERS:
https://technet.microsoft.com/en-us/library/dn338985(v=ws.11).aspx

2.11.1. Backup

Backup MS DHCP

```
> Export-DhcpServer -ComputerName dhcpserver.contoso.com -File
C:\exportdir\dhcpexport.xml
```

```
> Backup-DhcpServer -ComputerName dhcpserver.contoso.com -Path
C:\Windows\system32\dhcp\backup
```

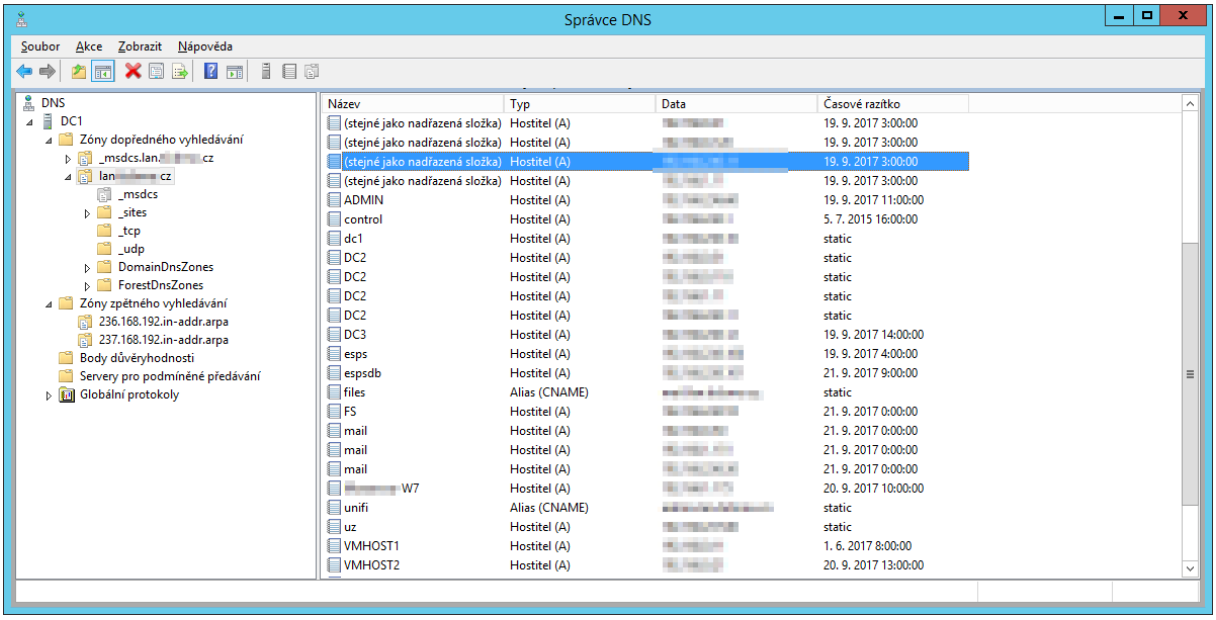
Restore MS DHCP

```
> Import-DhcpServer -ComputerName dhcpserver.contoso.com -File
C:\exports\dhcpexport.xml -BackupPath C:\dhcpbackup\
```

```
> Restore-DhcpServer -ComputerName dhcpserver.contoso.com -Path
C:\Windows\system32\dhcp\backup
```

2.12. DNS service [CMDLET_REFERENCE_LINK] [COMMANDLINE_DNSCMD_REF]

https://4sysops.com/archives/server-roles-in-server-core-part-3-dns-servers/



Install service

```
> Install-WindowsFeature DNS -IncludeManagementTools
```

```
> Get-DNSServer
```

```
> Test-DNSServer ip_of_the_DNS_server
```

List/Show forward/backward zones

```
> Get-DnsServerZone
```

List/Show zone content (records)

```
> Get-DnsServerResourceRecord
```

Create record (SOA,A,CNAME,MX…)

```
> Add-DnsServerResourceRecord -ZoneName "Contoso.com" -A -Name "Host34" -
AllowUpdateAny -IPv4Address "10.17.1.34" -TimeToLive 01:00:00 -AgeRecord
```

(admin mode)

```
> Add-DnsServerResourceRecord -ZoneName "Contoso.com" -A -Name
"Host21.admin" -IPv4Address "10.17.1.21"
```

```
> Add-DnsServerResourceRecord -CName -Name "labhost34" -HostNameAlias
"Host34.lab.contoso.com" -ZoneName "Contoso.com" -AllowUpdateAny  -
TimeToLive 01:00:00
```

```
> Add-DnsServerResourceRecord -Name "77" -Ptr -ZoneName "0.168.192.in-
addr.arpa" -AllowUpdateAny -PtrDomainName "host77.contoso.com"
```

```
> Add-DnsServerResourceRecord -Name ".-MX -ZoneName "contoso.com" -
MailExchange "mail.contoso.com" -Preference 10
```

Edit record

```
> $NewObj = $OldObj = Get-DnsServerResourceRecord -Name "Host01" -ZoneName
"contoso.com" -RRType "A"
> $NewObj.TimeToLive = [System.TimeSpan]::FromHours(2)
> Set-DnsServerResourceRecord -NewInputObject $NewObj -OldInputObject
$OldObj -ZoneName "contoso.com" -PassThru
```

Delete record

```
> Remove-DnsServerResourceRecord -ZoneName "contoso.com" -RRType "A" -Name
"Host01"
```

```
> Remove-DnsServerResourceRecord -ZoneName "contoso.com" -RRType "A" -Name
"Host01" -RecordData "10.17.1.41"
```

Start/Stop/restart DNS service

```
> Start-Service dns
```

```
> Stop-Service dns
```

NOTE: All "Authenticated Users" can do an create/update of child items in the DNS (incl. admin during adding) – but these records might have other owner, so the DHCP is than unable to deleted.

2.12.1. Backup

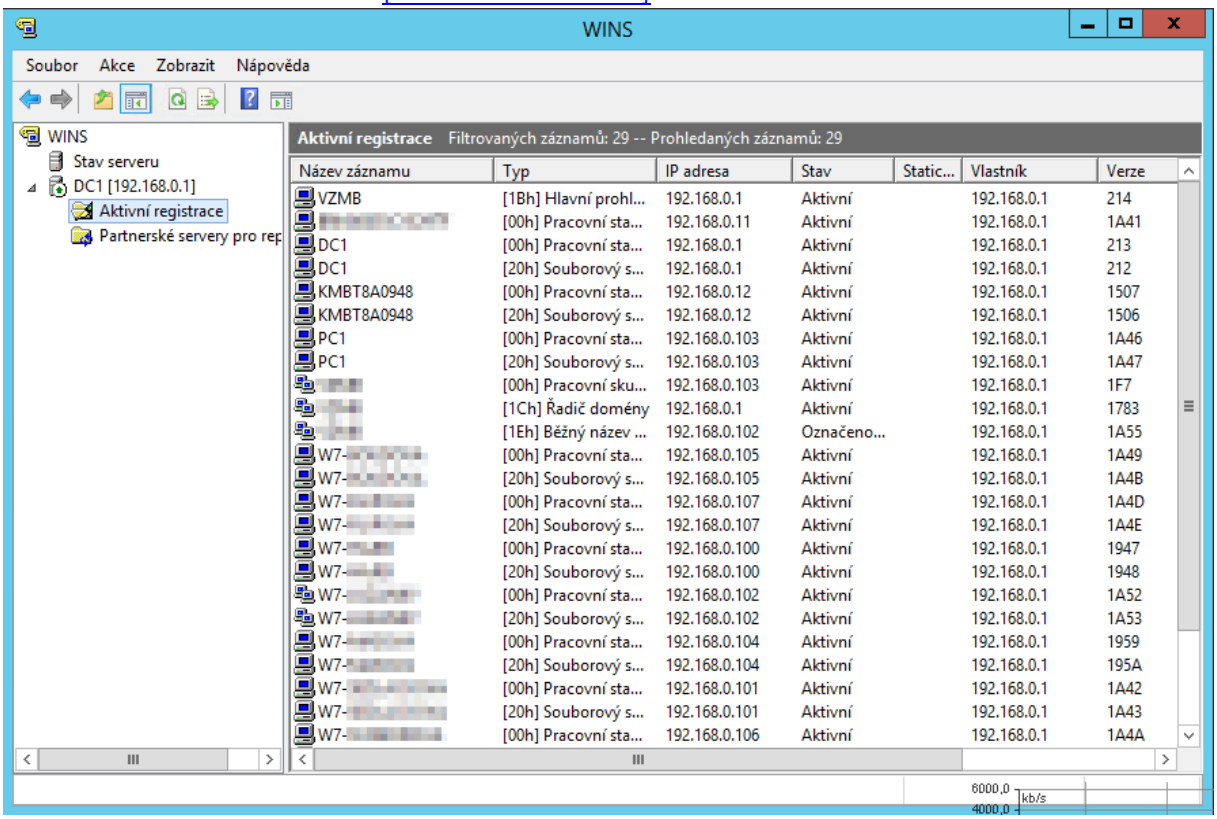
Backup MS DNS zones

http://c-nergy.be/blog/?p=1837

Restore MS DNS zones

http://c-nergy.be/blog/?p=1858

2.13. WINS service [CMD_REFERENCE]



2.13.1. Install and config

Install service

```
> Install-WindowsFeature WINS -IncludeManagementTools
```

```
> netsh WINS
netsh> server [{\\ServerName | IPAddress}]
```

```
netsh> add name [Name=]ComputerName [[EndChar=]16thCharInHex]
[[Scope=]ScopeName] [[RecType=]{0 | 1}] [[Group=]{0 | 1 | 2 | 3 | 4}]
[[Node=]{0 | 1 | 3}] [IP=]{IPAddress1[,IPAddress2,IPAddress3 ]}
```

RecType=0=static;1=dynamic;Group=0=Uniqe(default);1=group;2=Internet;3=Multihomed;4=Domain name;Node=0=B-node;1=P-Node(default);2=H-Node; Only ComputerName and IP are required

```
netsh> add name Name=HOSTA IP={10.0.0.1}
```

```
netsh> add partner [Server=]IPAddress [[NetBIOS=]ServerNetBIOSName]
[[Type=]{0 | 1 | 2}]
```

0=Pull;1=Push;2=Both;

```
netsh> add partner 10.2.2.2
```

```
netsh> add pngserver {10.2.2.2,10.2.2.3}
```

(Persona Non Grata servers)

```
netsh> check database
```

```
netsh> delete name Name=HOSTA
```

```
netsh> delete partner Server=10.0.0.1 Type=2 Confirm=Y
```

```
netsh> delete pngserver [Servers=]{ListOfServerIPAddresses}
```

2.13.2. Backup

Backup

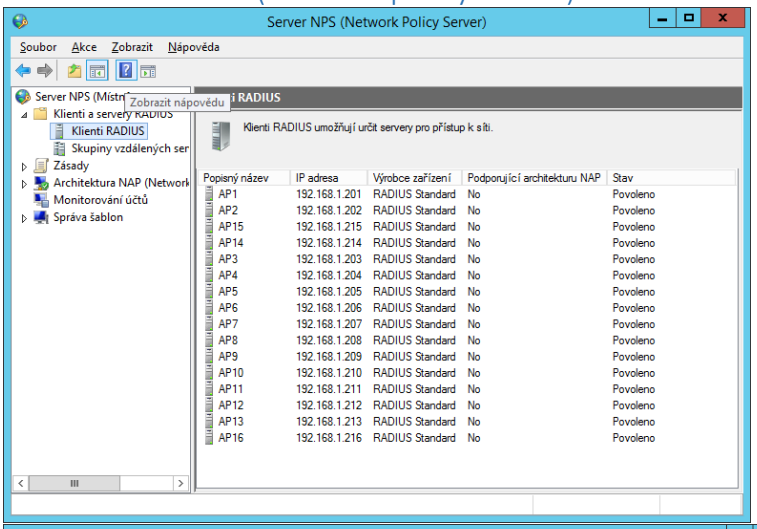
```
netsh> init backup Dir=C:\WINSfiles Type=0Restore MS DNS zones
```

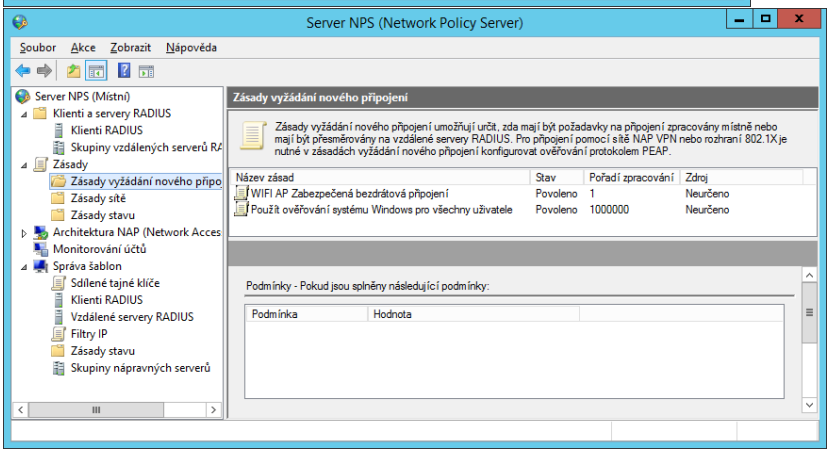
-Type 0=Full; 1=Increment;

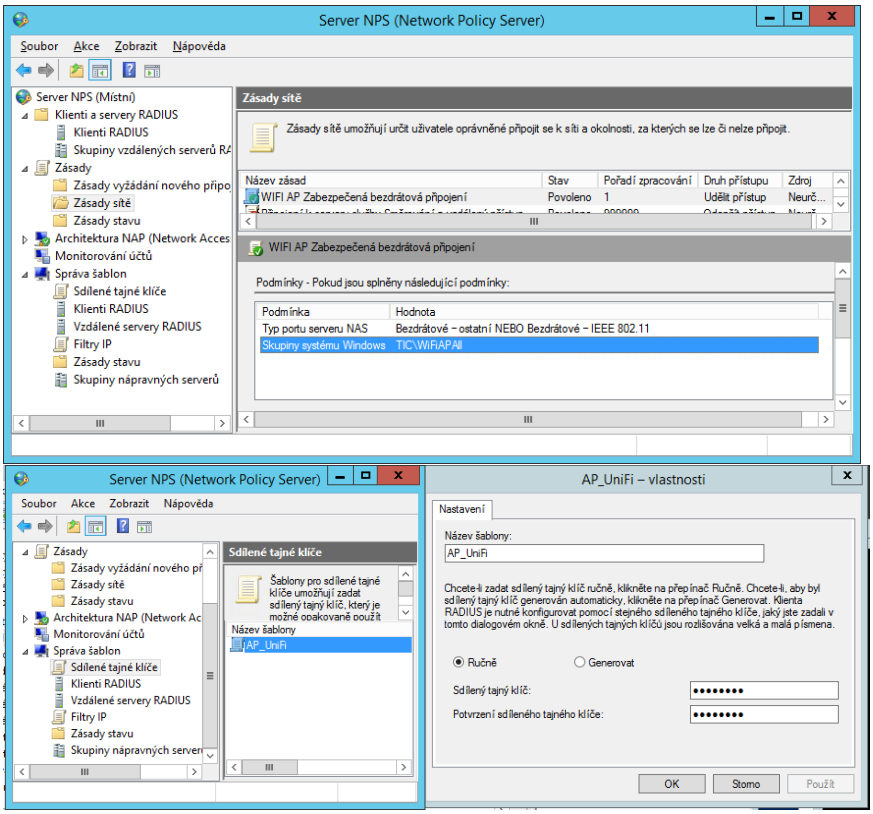
Restore

```
netsh> init restore [Dir=]RestoreDir
```

2.14. NPS (network policy server) service







Install service

```
> Get-WindowsFeature npas*
> Install-WindowsFeature -Name npas,npas-policy-server ->
IncludeManagementTools
```

List clients

```
> Get-NpsRadiusClient
```

Show device
List policies
Show policy
List/show security templates

```
> Get-NpsSharedSecretTemplate -Name XXXXX
```

Create client

```
> New-NpsRadiusClient -Address "10.0.0.200" -Name "WirelessAP" -
NapCompatible $True -SharedSecret "9vq7822hFsJ8rm"
```

Create policy

```
-
```

Create security template

```
-
```

Edit client

```
> Set-NpsRadiusClient -Name "WirelessAP" -Address "10.0.0.201" -
NapCompatible $False -SharedSecret "1234567890"
```

Edit policy

```
-
```

Edit security template

```
-
```

Delete (remove) client

```
> Remove-NpsRadiusClient -Name "RadiusClient01"
```

Delete policy

```
-
```

Delete security template

```
-
```

Start/Stop/restart NPS service

```
> Start-Service ias
```

```
> Stop-Service ias
```

2.14.1. Backup

Backup MS NPS settings

```
> Export-NpsConfiguration –Path C:\backuppath.file
```

Restore MS NPS settings

```
> Import-NpsConfiguration –Path C:\bacuppath.file
```

2.15. PrintService

Install service

```
> add-WindowsFeature Print-Server -IncludeManagementTools
```

2.16. Hyper-V service

Install service

```
> Install-WindowsFeature -Name Hyper-V -IncludeManagementTools -Restart
```

List VMs
Add vHDD
Edit vHDD
Remove vHDD
Add VM
Edit VM
List virtual interfaces of a VM

```
> Get-VMNetworkAdapter -VMName w2k3homologa
```

Add/Edit virtual interface of a VM

```
> Add-VMNetworkAdapter -VMName "XXXXX"" -Name "ADDAPTER_NEW_NAME" -
SwitchName "XXXXXXXXXXXXXXX"
```

```
> Add-VMNetworkAdapter -ManagementOS -Name "Management" -SwitchName
"ConvergedHyperSwitch"
```

```
> rename-VMNetworkAdapter -VMNetworkAdapter $VMNetAdap[1] -newname
InternalNi
```

```
> Set-VMNetworkAdapterVlan -vmname XXXXXX -VMNetworkAdapterName "XXXXXX" -
Trunk -AllowedVlanList 001-4097 -nativeVlanID 0
```

```
> Set-VMNetworkAdapterVlan -ManagementOS -VMNetworkAdapterName "Management"
-Access -VlanId 100
```

```
> Set-VMNetworkAdapter -ManagementOS -Name "LiveMigration" -
MinimumBandwidthWeight 20
```

Remove VM
Start-VM
Stop-VM
Restart-VM
List virtual switches
Add virtual switch

```
> New-VMSwitch -Name ConvergedHyperSwitch -NetAdapterName NICTeam01 -
AllowManagementOS $False -MinimumBandwidthMode Weight
```

Edit virtual switch

2.17. WSUS service

Install service

```
> Get-WindowsFeature -Name UpdateServices*
> Install-WindowsFeature -Name UpdateServices -IncludeManagementTools -
WhatIf
```

2.18. IIS service

Install service
Reset
Log cache flush
App pool recycle

2.19. Exchange service

2.19.1. Install

2.19.2. Setup

Open Exchange powershell

```
> $exchsessid = New-PSSession -configurationName MSExchangePS -
connectionUri https://MAILSERVER_URL/powershell -Credential DOMAIN\username
-Authentication basic -AllowRedirection
> Import-PSSession $exchsessid
```

```
runas /user:username@domain.com powershell.exe
```

Mailbox DB admin

```
> Get-MailboxDatabase | Set-MailboxDatabase -IssueWarningQuota 4.8GB -
ProhibitSendQuota 4.9GB -ProhibitSendReceiveQuota 5GB -DeletedItemRetention
"14:00:00" -MailboxRetention "30:00:00"
```

OWA/ECP in IIS admin

```
> Set-OWAVirtualDirectory -Identity "MAIL\owa (Default Web Site)" -
FormsAuthentication $False -BasicAuthentication $True
```

```
> set-Owavirtualdirectory -identity "MAIL\owa (Exchange Back End)" -
WindowsAuthentication $True -Basicauthentication $false -
Formsauthentication $false
```

```
> Set-EcpVirtualDirectory -Identity "MAIL\ecp (Exchange Back End)" -
WindowsAuthentication $true -FormsAuthentication $false
```

```
> Set-ECPVirtualDirectory -Identity "MAIL\ecp (Default Web Site)" -
AdminEnabled $True
```

```
> Get-OWAVirtualDirectory | select *AuthenticationMethods, DefaultDomain,
OwaVersion,FormsAuthentication
```

```
> Test-EcpConnectivity
```

Followed by IISreset and also to recycle the app pools for OWA and ECP

```
> iisreset
```

```
> Remove-EcpVirtualDirectory -Identity "MAIL\ecp (Default Web Site)"
> New-EcpVirtualDirectory -Server MAIL -InternalURL
https://mail.server.url.com/ecp -ExternalURL
https://mail.server.ext.url.com/ecp
> Set-ECPVirtualDirectory -Identity "MAIL\ecp (Default Web Site)" -
AdminEnabled $True
```

2.19.3. Configure

```
Get-Mailbox mymailbox@contoso.com | Select-Object -Property [a-z]*
```

```
Get-TransportConfig | Format-List ExternalPostmasterAddress
```

```
Set-TransportConfig -ExternalPostmasterAddress postmaster@contoso.com
```

```
Set-TransportConfig -ExternalPostmasterAddress $null
```

```
Get-ReceiveConnector
```

```
Get-ReceiveConnector  | Get-ADPermission
```

```
Get-MessageTrackingLog -Recipients zdenkotpmndkxnovak@safarjoon.com -Start
"10.1.2020 10:00" -End "11.18.2020 15:00" -ResultSize unlimited | Out-
GridView
```

```
Get-MessageTrackingLog -Sender zdenkotpmndkxnovak@safarjoon.com -Start
"10.1.2020 10:00" -End "11.18.2020 15:00" -ResultSize unlimited | Out-
GridView
```

3. Appendix A – PS Programming reference

PowerShell CrashCourse: https://technet.microsoft.com/en-us/library/hh551144.aspx

MULTILINE SEPARATOR IN PS:	; (semicolon)
NEWLINE ISERTOR IN PS:	` (backapostrophe)
STRING DENOMINATOR:	‘ (apostrophe) or “ (quote)
STRING ESCAPER:	” ‘$normal string and $expanded_string”

3.1. Comments

```
> # This is a non-interpreted comment
```

3.2. Variables

```
> $var = 'hello'
> $number = 1
> $numbers = 1,2,3,4,5,6,7,8,9
> $filter1 = "name='BITS'" //filter1 contains: name=’BITS’
> $computer = 'BITS'
> $filter2 = "name='$computer'" //filter2 contains: name=’BITS’
```

3.3. Array and hashtable variables

```
> $nameArray = @("John","Joe","Mary")
```

```
> $nameHashTable = @{FirstName="John"; LastName="Smith"; MiddleInitial="J";
Age=40}
```

3.4. Object variables

```
> $svc = Get-Service
> $svc[0].name
> $name = $svc[1].name
> $name.length
> $name.ToUpper()
```

3.5. Branching (IF)

```
> If ($this -eq $that) {
  # commands
} elseif ($those -ne $them) {
  # commands
} elseif ($we -gt $they) {
  # commands
} else {
  # commands
}
```

(for comparison and logical operators, see next Annex)

Loop – do…while

```
> Do {
  # commands
} While ($this -eq $that)
```

```
> Do {
  # commands
```

```
} Until ($this -neq $that)
```

```
> While ($i -le 10) {
    $i
    $i++
}
```

Loop – for, foreach

```
> For ($i=0; $i -le 10; $i++) {
    "10 * $i = " + (10 * $i)
}
```

```
> $services = Get-Service
> ForEach ($service in $services) {
  $service.Stop();
}
```

```
> 1..10 | ForEach-Object -process {
  # code here will repeat 10 times
  # use $_ to access the current iteration
  # number
}
```

Function spec

```
> function mine {
# CODE HERE
}
```

X.ps Script params

```
> param (
  [string]$computername,
  [string]$logfile,
  [int]$attemptcount = 5
)
```

4. Appendix B – Operators

4.1. Arithmetic operators [LINK]

Operator	Description	Example
+	Adds integers; concatenates strings, concatenates arrays, and hash tables.	> 6 + 2 > "file" + "name" > @(1, "one") + @(2.0, "two") > @{"one" = 1} + @{"two" = 2}
-	Subtracts one value from another value.	> 6-2 > (get-date).date - 1
-	Makes a number a negative number.	> -6
*	Multiplies numbers, copies strings and arrays the specified number of times.	> 6 * 2 > "!" * 3 > @("!") * 4
/	Divides two values.	> 6 / 2
%	Returns the remainder of a division operation.	> 7 % 2
-band	Bitwise AND	> 5 -band 3
-bnot	Bitwise NOT	> -bnot 5
-bor	Bitwise OR	> 5 -bor 0x03
-bxor	Bitwise XOR	> 5 -bxor 3
-shl	Shifts bits to the left the specified number of times	> 102 -shl 2
-shr	Shifts bits to the right the specified number of times	> 102 -shr 2

4.2. Assignment operators [LINK]

Operator	Description
=	Sets the value of a variable to the specified value.
+=	Increases the value of a variable by the specified value, or appends the specified value to the existing value.
-=	Decreases the value of a variable by the specified value.
*=	Multiplies the value of a variable by the specified value, or appends the specified value to the existing value.
/=	Divides the value of a variable by the specified value.
%=	Divides the value of a variable by the specified value and then assigns the remainder (modulus) to the variable.
++	Increases the value of a variable, assignable property, or array element by 1.
--	Decreases the value of a variable, assignable property, or array element by 1.

```
> $a = Get-Process
> $a += Get-Service
> $a += "string"
> $a += 12
> $a = Get-Service | Sort-Object -Property name
```

4.3. Comparison operators [LINK]

Operator Type	Operators	Description
Equality	-eq -ne -gt -ge -lt -le	equals not equals greater than greater than or equal less than less than or equal
Matching	-like -notlike -match -notmatch	Returns true when string matches wildcard pattern Returns true when string does not match wildcard pattern Returns true when string matches regex pattern - $matches contains matching strings Returns true when string does not match regex pattern - $matches contains matching strings
Containment	-contains -notcontains -in -notin	Returns true when reference value contained in a collection Returns true when reference value not contained in a collection Returns true when test value contained in a collection Returns true when test value not contained in a collection
Replacement	-replace	replace a string pattern
Type comparison	-is -isnot	Returns true if both object are the same type Returns true if the objects are not the same type

4.4. Logical operators [LINK]

Operator	Description	Example
-and	Logical and. TRUE only when both statements are TRUE.	> (1 -eq 1) -and (1 -eq 2) False
-or	Logical or. TRUE when either or both statements are TRUE.	> (1 -eq 1) -or (1 -eq 2) True
-xor	Logical exclusive or. TRUE only when one of the statements is TRUE and the other is FALSE.	> (1 -eq 1) -xor (2 -eq 2) False
-not	Logical not. Negates the statement that follows it.	> -not (1 -eq 1) False
!	Logical not. Negates the statement that follows it. (Same as -not)	> !(1 -eq 1) False

4.5. Redirection operators [LINK]

Operator	Description	Example
>	Sends output to the specified file.	> Get-Process > Process.txt
>>	Appends the output to the contents of the specified file.	> dir *.ps1 >> Scripts.txt
2>	Sends errors to the specified file.	> Get-Process none 2> Errors.txt
2>>	Appends errors to the contents of the specified file.	> Get-Process none 2>> Save-Errors.txt

2>&1	Sends errors (2) and success output (1) to the success output stream.	> Get-Process none, Powershell 2>&1
3>	Sends warnings to the specified file.	> Write-Warning "Test!" 3> Warnings.txt
3>>	Appends warnings to the contents of the specified file.	> Write-Warning "Test!" 3>> Save-Warnings.txt
3>&1	Sends warnings (3) and success output (1) to the success output stream.	> function Test-Warning{ Get-Process PowerShell; Write-Warning "Test!" } > Test-Warning 3>&1
4>	Sends verbose output to the specified file.	> Import-Module * -Verbose 4> Verbose.txt
4>>	Appends verbose output to the contents of the specified file.	> Import-Module * -Verbose 4>> Save-Verbose.txt
4>&1	Sends verbose output (4) and success output (1) to the success output stream.	> Import-Module * -Verbose 4>&1
5>	Sends debug messages to the specified file.	> Write-Debug "Starting" 5> Debug.txt
5>>	Appends debug messages to the contents of the specified file.	> Write-Debug "Saving" 5>> Save-Debug.txt
5>&1	Sends debug messages (5) and success output (1) to the success output stream.	> function Test-Debug { Get-Process PowerShell Write-Debug "PS" } Test-Debug 5>&1
*>	Sends all output types to the specified file.	> function Test-Output { Get-Process PowerShell, none Write-Warning "Test!" Write-Verbose "Test Verbose" Write-Debug "Test Debug" }
*>>	Appends all output types to the contents of the specified file.	
*>&1	Sends all output types (*) to the success output stream.	> Test-Output *> Test-Output.txt > Test-Output *>> Test-Output.txt > Test-Output *>&1

4.6. Split and join operators [LINK][LINK]

```
-Split <String>
```

```
<String> -Split <Delimiter>[,<Max-substrings>[,"<Options>"]]
```

```
<String> -Split {<ScriptBlock>} [,<Max-substrings>]
```

You can substitute `-iSplit` or `-cSplit` for `-split` in any binary Split statement (a Split statement that includes a delimiter or script block). The `-iSplit` and `-split` operators are case-insensitive. The `-cSplit` operator is case-sensitive, meaning that case is considered when the delimiter rules are applied.

```
-Join <String[]>
<String[]> -Join <Delimiter>
```

4.7. Type operators [LINK]

Operator	Description	Example
-is	Returns TRUE when the input is an instance of the specified.NET Framework type.	> (get-date) -is [DateTime] True
-isNot	Returns TRUE when the input is not an instance of the specified.NET Framework type.	> (get-date) -isNot [DateTime] False
-as	Converts the input to the specified.NET Framework type.	> 12/31/07 -as [DateTime] Monday, December 31, 2007 12:00:00 AM

4.8. Unary operators

$a++

4.9. Special Operators

@ ( ) Array subexpression operator

Returns the result of one or more statements as an array. If there is only one item, the array has only one member.

& Call operator

Runs a command, script, or script block. The call operator, also known as the "invocation operator," lets you run commands that are stored in variables and represented by strings. Because the call operator does not parse the command, it cannot interpret command parameters.

PowerShell

```
$c = "get-executionpolicy"
$c

# get-executionpolicy

& $c

# AllSigned
```

[ ] Cast operator

Converts or limits objects to the specified type. If the objects cannot be converted, PowerShell generates an error.

PowerShell

```
[datetime]$birthday = "1/20/88"
[int64]$a = 34
```

, Comma operator

As a binary operator, the comma creates an array. As a unary operator, the comma creates an array with one member. Place the comma before the member.

PowerShell

```
$myArray = 1,2,3
$SingleArray = ,1
```

. Dot sourcing operator

Runs a script in the current scope so that any functions, aliases, and variables that the script creates are added to the current scope.

PowerShell

```
. c:\scripts.sample.ps1
```

Note
The dot sourcing operator is followed by a space. Use the space to distinguish the dot from the dot (.) symbol that represents the current directory.
In the following example, the Sample.ps1 script in the current directory is run in the current scope.
PowerShell
```
. .\sample.ps1
```

-f Format operator [LINK]

Formats strings by using the format method of string objects. Enter the format string on the left side of the operator and the objects to be formatted on the right side of the operator.
PowerShell
```
"{0} {1,-10} {2:N}" -f 1,"hello",[math]::pi

# 1 hello      3.14
```

[ ] Index operator

Selects objects from indexed collections, such as arrays and hash tables. Array indexes are zero-based, so the first object is indexed as `[0]`. For arrays (only), you can also use negative indexes to get the last values. Hash tables are indexed by key value.
PowerShell
```
$a = 1, 2, 3
$a[0]

# 1

$a[-1]

# 3
```
PowerShell
```
(get-hotfix | sort installedOn)[-1]
```

| Pipeline operator

Sends ("pipes") the output of the command that precedes it to the command that follows it. When the output includes more than one object (a "collection"), the pipeline operator sends the objects one at a time.

. Property dereference operator

Accesses the properties and methods of an object.
PowerShell
```
$myProcess.peakWorkingSet
(get-process PowerShell).kill()
```

.. Range operator

Represents the sequential integers in an integer array, given an upper and lower boundary.
PowerShell
```
1..10
10..1
foreach ($a in 1..$max) {write-host $a}
```

:: Static member operator

Calls the static properties operator and methods of a .NET Framework class. To find the static properties and methods of an object, use the Static parameter of the `Get-Member` cmdlet.
PowerShell
```
[datetime]::now
```

$( ) Subexpression operator

Returns the result of one or more statements. For a single result, returns a scalar. For multiple results, returns an array.

PowerShell

```
$($x * 23)
$(Get-WMIObject win32_Directory)
```

5. Appendix B – CMD Programming reference

PowerShell CrashCourse: https://technet.microsoft.com/en-us/library/hh551144.aspx

SILENCE DISPLAY: `@ECHO OFF` / `ECHO ON`

MULTILINE SEPARATOR IN PS: ; (semicolon)

NEWLINE ISERTOR IN PS: ` (backapostrophe)

STRING DENOMINATOR: ' (apostrophe) or " (quote)

STRING ESCAPER: `" '$normal string and $expanded_string"`

5.1. Comments

```
> REM This is a non-interpreted comment
```

```
> :: This is (sometimes) also a non-interpreted comment
```

5.2. Variables

```
> $var = 'hello'
> $number = 1
> $numbers = 1,2,3,4,5,6,7,8,9
> $filter1 = "name='BITS'" //filter1 contains: name='BITS'
> $computer = 'BITS'
> $filter2 = "name='$computer'" //filter2 contains: name='BITS'
```

5.3. Array and hashtable variables

```
> $nameArray = @("John","Joe","Mary")
```

```
> $nameHashTable = @{FirstName="John"; LastName="Smith"; MiddleInitial="J";
Age=40}
```

5.4. Object variables

```
> $svc = Get-Service
> $svc[0].name
> $name = $svc[1].name
> $name.length
> $name.ToUpper()
```

5.5. Branching (IF)

```
> If ($this -eq $that) {
  # commands
} elseif ($those -ne $them) {
  # commands
} elseif ($we -gt $they) {
  # commands
} else {
  # commands
}
```

(for comparison and logical operators, see next Annex)

Loop – do…while

```
> Do {
```

```
  # commands
} While ($this -eq $that)
```

```
> Do {
  # commands
} Until ($this -neq $that)
```

```
> While ($i -le 10) {
    $i
    $i++
}
```

Loop – for, foreach

```
> For ($i=0; $i -le 10; $i++) {
    "10 * $i = " + (10 * $i)
}
```

```
> $services = Get-Service
> ForEach ($service in $services) {
  $service.Stop();
}
```

```
> 1..10 | ForEach-Object -process {
  # code here will repeat 10 times
  # use $_ to access the current iteration
  # number
}
```

Function spec

```
> function mine {
# CODE HERE
}
```

X.ps Script params

```
> param (
  [string]$computername,
  [string]$logfile,
  [int]$attemptcount = 5
)
```

6. Appendix B – Operators

6.1. Arithmetic operators [LINK]

Operator	Description	Example
+	Adds integers; concatenates strings, concatenates arrays, and hash tables.	> 6 + 2 > "file" + "name" > @(1, "one") + @(2.0, "two") > @{"one" = 1} + @{"two" = 2}
-	Subtracts one value from another value.	> 6-2 > (get-date).date - 1
-	Makes a number a negative number.	> -6
*	Multiplies numbers, copies strings and arrays the specified number of times.	> 6 * 2 > "!" * 3 > @("!") * 4
/	Divides two values.	> 6 / 2
%	Returns the remainder of a division operation.	> 7 % 2
-band	Bitwise AND	> 5 -band 3
-bnot	Bitwise NOT	> -bnot 5
-bor	Bitwise OR	> 5 -bor 0x03
-bxor	Bitwise XOR	> 5 -bxor 3
-shl	Shifts bits to the left the specified number of times	> 102 -shl 2
-shr	Shifts bits to the right the specified number of times	> 102 -shr 2

6.2. Assignment operators [LINK]

Operator	Description
=	Sets the value of a variable to the specified value.
+=	Increases the value of a variable by the specified value, or appends the specified value to the existing value.
-=	Decreases the value of a variable by the specified value.
*=	Multiplies the value of a variable by the specified value, or appends the specified value to the existing value.
/=	Divides the value of a variable by the specified value.
%=	Divides the value of a variable by the specified value and then assigns the remainder (modulus) to the variable.
++	Increases the value of a variable, assignable property, or array element by 1.
--	Decreases the value of a variable, assignable property, or array element by 1.

```
> $a = Get-Process
> $a += Get-Service
> $a += "string"
> $a += 12
> $a = Get-Service | Sort-Object -Property name
```

6.3. Comparison operators [LINK]

Operator Type	Operators	Description
Equality	-eq -ne -gt -ge -lt -le	equals not equals greater than greater than or equal less than less than or equal
Matching	-like -notlike -match -notmatch	Returns true when string matches wildcard pattern Returns true when string does not match wildcard pattern Returns true when string matches regex pattern - $matches contains matching strings Returns true when string does not match regex pattern - $matches contains matching strings
Containment	-contains -notcontains -in -notin	Returns true when reference value contained in a collection Returns true when reference value not contained in a collection Returns true when test value contained in a collection Returns true when test value not contained in a collection
Replacement	-replace	replace a string pattern
Type comparison	-is -isnot	Returns true if both object are the same type Returns true if the objects are not the same type

6.4. Logical operators [LINK]

Operator	Description	Example
-and	Logical and. TRUE only when both statements are TRUE.	> (1 -eq 1) -and (1 -eq 2) False
-or	Logical or. TRUE when either or both statements are TRUE.	> (1 -eq 1) -or (1 -eq 2) True
-xor	Logical exclusive or. TRUE only when one of the statements is TRUE and the other is FALSE.	> (1 -eq 1) -xor (2 -eq 2) False
-not	Logical not. Negates the statement that follows it.	> -not (1 -eq 1) False
!	Logical not. Negates the statement that follows it. (Same as -not)	> !(1 -eq 1) False

6.5. Redirection operators [LINK]

Operator	Description	Example
>	Sends output to the specified file.	> Get-Process > Process.txt
>>	Appends the output to the contents of the specified file.	> dir *.ps1 >> Scripts.txt
2>	Sends errors to the specified file.	> Get-Process none 2> Errors.txt
2>>	Appends errors to the contents of the specified file.	> Get-Process none 2>> Save-Errors.txt

2>&1	Sends errors (2) and success output (1) to the success output stream.	> Get-Process none, Powershell 2>&1
3>	Sends warnings to the specified file.	> Write-Warning "Test!" 3> Warnings.txt
3>>	Appends warnings to the contents of the specified file.	> Write-Warning "Test!" 3>> Save-Warnings.txt
3>&1	Sends warnings (3) and success output (1) to the success output stream.	> function Test-Warning{ Get-Process PowerShell; Write-Warning "Test!" } > Test-Warning 3>&1
4>	Sends verbose output to the specified file.	> Import-Module * -Verbose 4> Verbose.txt
4>>	Appends verbose output to the contents of the specified file.	> Import-Module * -Verbose 4>> Save-Verbose.txt
4>&1	Sends verbose output (4) and success output (1) to the success output stream.	> Import-Module * -Verbose 4>&1
5>	Sends debug messages to the specified file.	> Write-Debug "Starting" 5> Debug.txt
5>>	Appends debug messages to the contents of the specified file.	> Write-Debug "Saving" 5>> Save-Debug.txt
5>&1	Sends debug messages (5) and success output (1) to the success output stream.	> function Test-Debug { Get-Process PowerShell Write-Debug "PS" } Test-Debug 5>&1
*>	Sends all output types to the specified file.	> function Test-Output { Get-Process PowerShell, none Write-Warning "Test!" Write-Verbose "Test Verbose" Write-Debug "Test Debug" }
*>>	Appends all output types to the contents of the specified file.	
*>&1	Sends all output types (*) to the success output stream.	> Test-Output *> Test-Output.txt > Test-Output *>> Test-Output.txt > Test-Output *>&1

6.6. Split and join operators [LINK][LINK]

```
-Split <String>

<String> -Split <Delimiter>[,<Max-substrings>[,"<Options>"]]

<String> -Split {<ScriptBlock>} [,<Max-substrings>]
```

You can substitute `-iSplit` or `-cSplit` for `-split` in any binary Split statement (a Split statement that includes a delimiter or script block). The `-iSplit` and `-split` operators are case-insensitive. The `-cSplit` operator is case-sensitive, meaning that case is considered when the delimiter rules are applied.

```
-Join <String[]>
<String[]> -Join <Delimiter>
```

6.7. Type operators [LINK]

Operator	Description	Example
-is	Returns TRUE when the input is an instance of the specified.NET Framework type.	> (get-date) -is [DateTime] True
-isNot	Returns TRUE when the input is not an instance of the specified.NET Framework type.	> (get-date) -isNot [DateTime] False
-as	Converts the input to the specified.NET Framework type.	> 12/31/07 -as [DateTime] Monday, December 31, 2007 12:00:00 AM

6.8. Unary operators

$a++

6.9. Special Operators

@ ( ) Array subexpression operator

Returns the result of one or more statements as an array. If there is only one item, the array has only one member.

& Call operator

Runs a command, script, or script block. The call operator, also known as the "invocation operator," lets you run commands that are stored in variables and represented by strings. Because the call operator does not parse the command, it cannot interpret command parameters.

PowerShell

```
$c = "get-executionpolicy"
$c

# get-executionpolicy

& $c

# AllSigned
```

[ ] Cast operator

Converts or limits objects to the specified type. If the objects cannot be converted, PowerShell generates an error.

PowerShell

```
[datetime]$birthday = "1/20/88"
[int64]$a = 34
```

, Comma operator

As a binary operator, the comma creates an array. As a unary operator, the comma creates an array with one member. Place the comma before the member.

PowerShell

```
$myArray = 1,2,3
$SingleArray = ,1
```

. Dot sourcing operator

Runs a script in the current scope so that any functions, aliases, and variables that the script creates are added to the current scope.

PowerShell

```
. c:\scripts.sample.ps1
```

Note
The dot sourcing operator is followed by a space. Use the space to distinguish the dot from the dot (.) symbol that represents the current directory.
In the following example, the Sample.ps1 script in the current directory is run in the current scope.
PowerShell
```
. .\sample.ps1
```

-f Format operator [LINK]

Formats strings by using the format method of string objects. Enter the format string on the left side of the operator and the objects to be formatted on the right side of the operator.
PowerShell
```
"{0} {1,-10} {2:N}" -f 1,"hello",[math]::pi

# 1 hello      3.14
```

[ ] Index operator

Selects objects from indexed collections, such as arrays and hash tables. Array indexes are zero-based, so the first object is indexed as `[0]`. For arrays (only), you can also use negative indexes to get the last values. Hash tables are indexed by key value.
PowerShell
```
$a = 1, 2, 3
$a[0]

# 1

$a[-1]

# 3
```
PowerShell
```
(get-hotfix | sort installedOn)[-1]
```

| Pipeline operator

Sends ("pipes") the output of the command that precedes it to the command that follows it. When the output includes more than one object (a "collection"), the pipeline operator sends the objects one at a time.

. Property dereference operator

Accesses the properties and methods of an object.
PowerShell
```
$myProcess.peakWorkingSet
(get-process PowerShell).kill()
```

.. Range operator

Represents the sequential integers in an integer array, given an upper and lower boundary.
PowerShell
```
1..10
10..1
foreach ($a in 1..$max) {write-host $a}
```

:: Static member operator

Calls the static properties operator and methods of a .NET Framework class. To find the static properties and methods of an object, use the Static parameter of the `Get-Member` cmdlet.
PowerShell
```
[datetime]::now
```

$( ) Subexpression operator

Returns the result of one or more statements. For a single result, returns a scalar. For multiple results, returns an array.

PowerShell

```
$($x * 23)
$(Get-WMIObject win32_Directory)
```

7. Appendix C – path variables reference

Variable	Windows 10
%ALLUSERSPROFILE%	C:\ProgramData
%APPDATA%	C:\Users\{username}\AppData\Roaming
%COMMONPROGRAMFILES%	C:\Program Files\Common Files
%COMMONPROGRAMFILES(x86)%	C:\Program Files (x86)\Common Files
%CommonProgramW6432%	C:\Program Files\Common Files
%COMSPEC%	C:\Windows\System32\cmd.exe
%HOMEDRIVE%	C:\
%HOMEPATH%	C:\Users\{username}
%LOCALAPPDATA%	C:\Users\{username}\AppData\Local
%LOGONSERVER%	\\{domain_logon_server}
%PATH%	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem
%PathExt%	.com;.exe;.bat;.cmd;.vbs;.vbe;.js;.jse;.wsf;.wsh;.msc
%PROGRAMDATA%	C:\ProgramData
%PROGRAMFILES%	C:\Program Files
%ProgramW6432%	C:\Program Files
%PROGRAMFILES(X86)%	C:\Program Files (x86)
%PROMPT%	PG
%SystemDrive%	C:
%SystemRoot%	C:\Windows
%TEMP%	C:\Users\{username}\AppData\Local\Temp
%TMP%	C:\Users\{username}\AppData\Local\Temp
%USERDOMAIN%	{userdomain}
%USERNAME%	{username}
%USERPROFILE%	C:\Users\{username}
%WINDIR%	C:\Windows
%PUBLIC%	C:\Users\Public
%PSModulePath%	%SystemRoot%\system32\WindowsPowerShell\v1.0\Modules\

8. Appendix D – GUI CPL shortcuts

9. Control Panel Applet	Command	OS Version
Accessibility Options	`> control access.cpl`	XP
Action Center	`> control /name Microsoft.ActionCenter`	8, 7
	`> control wscui.cpl`	8, 7
Active Directory Sites and Services snap-in	`> dssite.msc`	Srv
Add Features to Windows 8	`> control /name Microsoft.WindowsAnytimeUpgrade`	8
Add Hardware	`> control /name Microsoft.AddHardware`	Vista
	`> control hdwwiz.cpl`	XP
Add or Remove Programs	`> control appwiz.cpl`	XP
Administrative Tools	`> control /name Microsoft.AdministrativeTools`	8, 7, Vista
	`> control admintools`	8, 7, Vista, XP
Automatic Updates	`> control wuaucpl.cpl`	XP
AutoPlay	`> control /name Microsoft.AutoPlay`	8, 7, Vista
Backup and Restore Center	`> control /name Microsoft.BackupAndRestoreCenter`	Vista
Backup and Restore	`> control /name Microsoft.BackupAndRestore`	7
Biometric Devices	`> control /name Microsoft.BiometricDevices`	8, 7
BitLocker Drive Encryption	`> control /name Microsoft.BitLockerDriveEncryption`	8, 7, Vista
Bluetooth Devices	`> control bthprops.cpl`[13]	8, 7, Vista
	`> control /name Microsoft.BluetoothDevices`	Vista
Color Management	`> control /name Microsoft.ColorManagement`	8, 7, Vista
Color[1]	`> WinColor.exe`[2]	XP
Credential Manager	`> control /name Microsoft.CredentialManager`	8, 7
Client Service for NetWare	`> control nwc.cpl`	XP
Date and Time	`> control /name Microsoft.DateAndTime`	8, 7, Vista
	`> control timedate.cpl`	8, 7, Vista
	`> control date/time`	8, 7, Vista, XP
Default Location	`> control /name Microsoft.DefaultLocation`	7
Default Programs	`> control /name Microsoft.DefaultPrograms`	8, 7, Vista
Desktop Gadgets	`> control /name Microsoft.DesktopGadgets`	7

Device Manager	`> control /name Microsoft.DeviceManager`	8, 7, Vista
	`> control hdwwiz.cpl`	8, 7, Vista
	`> devmgmt.#`	8, 7, Vista, XP[3]
Devices and Printers	`> control /name Microsoft.DevicesAndPrinters`	8, 7
	`> control printers`	8, 7
Display	`> control /name Microsoft.Display`	8, 7
	`> control desk.cpl`	XP
	`> control desktop`	XP
Ease of Access Center	`> control /name Microsoft.EaseOfAccessCenter`	8, 7, Vista
	`> control access.cpl`	8, 7, Vista
Family Safety	`> control /name Microsoft.ParentalControls`	8
File History	`> control /name Microsoft.FileHistory`	8
Flash Player Settings Manager	`> control flashplayercplapp.cpl`	8
Folder Options	`> control /name Microsoft.FolderOptions`	8, 7, Vista
	`> control folders`	8, 7, Vista, XP
Fonts	`> control /name Microsoft.Fonts`	8, 7, Vista
	`> control fonts`	8, 7, Vista, XP
Game Controllers	`> control /name Microsoft.GameControllers`	8, 7, Vista
	`> control joy.cpl`	8, 7, Vista, XP
Get Programs	`> control /name Microsoft.GetPrograms`	8, 7, Vista
Getting Started	`> control /name Microsoft.GettingStarted`	7
Home Group	`> control /name Microsoft.HomeGroup`	8, 7
Indexing Options	`> control /name Microsoft.IndexingOptions`	8, 7, Vista
	`> rundll32.exe shell32.dll,Control_RunDLL srchadmin.dll`	8, 7, Vista, XP
Infrared	`> control /name Microsoft.Infrared`	8, 7
	`> control irprops.cpl`	8, 7, Vista
	`> control /name Microsoft.InfraredOptions`	Vista
Internet Options	`> control /name Microsoft.InternetOptions`	8, 7, Vista
	`> control inetcpl.cpl`	8, 7, Vista, XP
iSCSI Initiator	`> control /name Microsoft.iSCSIInitiator`	8, 7, Vista
Keyboard	`> control /name Microsoft.Keyboard`	8, 7, Vista

	`> control keyboard`	8, 7, Vista, XP
Language	`> control /name Microsoft.Language`	8
Location and Other Sensors	`> control /name Microsoft.LocationAndOtherSensors`	7
Location Settings	`> control /name Microsoft.LocationSettings`	8
Mail[4]	`> control mlcfg32.cpl`[5]	8, 7, Vista, XP
Mouse	`> control /name Microsoft.Mouse`	8, 7, Vista
	`> control main.cpl`	8, 7, Vista
	`> control mouse`	8, 7, Vista, XP
Network and Sharing Center	`> control /name Microsoft.NetworkAndSharingCenter`	8, 7, Vista
Network Connections	`> control ncpa.cpl`	8, 7, Vista
	`> control netconnections`	8, 7, Vista, XP
Network Setup Wizard	`> control netsetup.cpl`	8, 7, Vista, XP
Notification Area Icons	`> control /name Microsoft.NotificationAreaIcons`	8, 7
ODBC Data Source Administrator	`> control odbccp32.cpl`	XP[6]
Offline Files	`> control /name Microsoft.OfflineFiles`	8, 7, Vista
Parental Controls	`> control /name Microsoft.ParentalControls`	7, Vista
Pen and Input Devices	`> control /name Microsoft.PenAndInputDevices`	Vista
	`> control tabletpc.cpl`	Vista
Pen and Touch	`> control /name Microsoft.PenAndTouch`	8, 7
	`> control tabletpc.cpl`	8, 7
People Near Me	`> control /name Microsoft.PeopleNearMe`	7, Vista
	`> control collab.cpl`	7, Vista
Performance Information and Tools	`> control /name Microsoft.PerformanceInformationAndTools`	8, 7, Vista
Personalization	`> control /name Microsoft.Personalization`	8, 7, Vista
	`> control desktop`	8, 7, Vista
Phone and Modem Options	`> control /name Microsoft.PhoneAndModemOptions`	Vista
	`> control telephon.cpl`	Vista, XP
Phone and Modem	`> control /name Microsoft.PhoneAndModem`	8, 7
	`> control telephon.cpl`	8, 7
Power Options	`> control /name Microsoft.PowerOptions`	8, 7, Vista

	`> control powercfg.cpl`	8, 7, Vista, XP
Printers and Faxes	`> control printers`	XP
Printers	`> control /name Microsoft.Printers`	Vista
	`> control printers`	Vista
Problem Reports and Solutions	`> control /name Microsoft.ProblemReportsAndSolutions`	Vista
Programs and Features	`> control /name Microsoft.ProgramsAndFeatures`	8, 7, Vista
	`> control appwiz.cpl`	8, 7, Vista
Recovery	`> control /name Microsoft.Recovery`	8, 7
Region	`> control /name Microsoft.RegionAndLanguage`	8
	`> control intl.cpl`	8
	`> control international`	8
Region and Language	`> control /name Microsoft.RegionAndLanguage`	7
	`> control intl.cpl`	7
	`> control international`	7
Regional and Language Options	`> control /name Microsoft.RegionalAndLanguageOptions`	Vista
	`> control intl.cpl`	Vista
	`> control international`	Vista, XP
RemoteApp and Desktop Connections	`> control /name Microsoft.RemoteAppAndDesktopConnections`	8, 7
Scanners and Cameras	`> control /name Microsoft.ScannersAndCameras`	8, 7, Vista
	`> control sticpl.cpl`	XP
Scheduled Tasks	`> control schedtasks`	XP[7]
Screen Resolution	`> control desk.cpl`	8, 7
Security Center	`> control /name Microsoft.SecurityCenter`	Vista
	`> control wscui.cpl`	XP
Software Explorers[8]	`> msascui.exe`[9]	XP
Sound	`> control /name Microsoft.Sound`	8, 7
	`control /name Microsoft.AudioDevicesAndSoundThemes`	Vista
	`> control mmsys.cpl`	8, 7, Vista
Sounds and Audio Devices	`> control mmsys.cpl`	XP
Speech Recognition Options	`> control /name Microsoft.SpeechRecognitionOptions`	Vista
Speech Recognition	`> control /name Microsoft.SpeechRecognition`	8, 7
Speech	`> control sapi.cpl`[10]	XP

Storage Spaces	`> control /name Microsoft.StorageSpaces`	8
Sync Center	`> control /name Microsoft.SyncCenter`	8, 7, Vista
System workgroup domain names membership	`> control /name Microsoft.System`	8, 7, Vista
	`> control sysdm.cpl`	XP
System Properties	`> control sysdm.cpl`	8, 7, Vista
Tablet PC Settings	`> control /name Microsoft.TabletPCSettings`	8, 7, Vista
Task Scheduler[7]	`> control schedtasks`	8, 7, Vista
Taskbar	`> control /name Microsoft.Taskbar`	8
	`> rundll32.exe shell32.dll,Options_RunDLL 1`	8
Taskbar and Start Menu	`> control /name Microsoft.TaskbarAndStartMenu`	7, Vista
	`> rundll32.exe shell32.dll,Options_RunDLL 1`	7, Vista, XP
Text to Speech	`> control /name Microsoft.TextToSpeech`	8, 7, Vista
Troubleshooting	`> control /name Microsoft.Troubleshooting`	8, 7
User Accounts	`> control /name Microsoft.UserAccounts`	8, 7, Vista
	`> control userpasswords`	8, 7, Vista, XP
Welcome Center	`> control /name Microsoft.WelcomeCenter`	Vista
Windows 7 File Recovery	`> control /name Microsoft.BackupAndRestore`	8
Windows Anytime Upgrade	`> control /name Microsoft.WindowsAnytimeUpgrade`	7, Vista
Windows CardSpace	`> control /name Microsoft.CardSpace`	7, Vista
	`> control infocardcpl.cpl`	7, Vista
Windows Defender	`> control /name Microsoft.WindowsDefender`	8, 7, Vista[11]
Windows Firewall	`> control /name Microsoft.WindowsFirewall`	8, 7, Vista
	`> control firewall.cpl`	8, 7, Vista, XP
Windows Marketplace	`> control /name Microsoft.GetProgramsOnline`	Vista
Windows Mobility Center	`> control /name Microsoft.MobilityCenter`	8, 7, Vista
Windows Sidebar Properties	`> control /name Microsoft.WindowsSidebarProperties`	Vista
Windows SideShow	`> control /name Microsoft.WindowsSideShow`	8,7, Vista
Windows Update	`> control /name Microsoft.WindowsUpdate`	8, 7, Vista[12]
Wireless Link	`> control irprops.cpl`	XP

Wireless Network Setup Wizard	`> ?`	XP

[1] Color is not available by default but is available for free from Microsoft here.

[2] WinColor.exe must be run from the C:\Program Files\Pro Imaging Powertoys\Microsoft Color Control Panel Applet for Windows XP folder.

[3] I've listed Device Manager here because it's such a commonly used feature of Windows but please know that it is not a true Control Panel applet in Windows XP. See How To Open Windows XP Device Manager for more information.

[4] The Mail applet is only available if a version of Microsoft Office Outlook is installed.

[5] The control mlcfg32.cpl command must be run from the C:\Programs Files\Microsoft Office\OfficeXX folder, replacing OfficeXX with the folder pertaining to the Microsoft Office version you have installed.

[6] ODBC Data Source Administrator was removed from Control Panel after Windows XP but is still available from Administrative Tools.

[7] In Windows 8, 7, and Vista, task scheduling is performed by Task Scheduler which is not directly accessible from Control Panel. However, executing this command in those versions of Windows will forward to Task Scheduler.

[8] Software Explorers is the name for the Control Panel applet for Windows Defender, available for free from Microsoft here as part of Microsoft Security Essentials.

[9] Msascui.exe must be run from the C:\Program Files\Windows Defender folder.

[10] The control sapi.cpl command must be run from the C:\Program Files\Common Files\Microsoft Shared\Speech folder.

[11] Windows Defender is available in Windows XP but the Control Panel applet is instead called Software Explorers.

[12] Windows Update is also used in Windows XP but only via the Windows Update website, not via a Control Panel applet like in later versions of Windows.

[13] In Windows 8, bthprops.cpl opens Devices in PC Settings which will list any Bluetooth Devices. In Windows 7, bthprops.cpl opens athe Bluetooth Devices list under Devices and Printers. In Windows Vista, bthprops.cpl opens a true Control Panel applet called Bluetooth Devices.