PowerShell Basics - 1

Profiles
<InstallDir>\profile.ps1 - general Profile file
%userprofile%\<Documents>\WindowsPowerShell\profile.ps1 - PowerShell User profile
Within the Shell

Commands
```
#                                   # for commenting lines
;                                   # for multiple commands, i.e.
                                      get-process;get-service;get-vm
F7 for command history
cursor up/cursor down               # walking through commands history
Get-History                         # command history
ESC                                 # clear current line
CTRL + C                            # abort Command execution
clear                               # clear whole screen
```

Variables
```
dir variable:                       # show all variables
$?                                  # last error code
$number = 123
[int]$i = 5
$string = “abc”
$_ = current object in pipeline
$myobject = get-process calc
Numbers: [int], [long], [double], [decimal], [float], [single], [byte]
Strings: [string] , [char]
$var1, $var2 = $var2, $var1         # switch variable content
```

Variable Scope
$global: i = 5; $script: i = 5;$local: i = 5 (Default); $private: I = 5 (within function)

Arrays
```
$a = 1,2,3                          # Array with 3 Elements
$a += , $vm.name                    # adding Array element
$a = $a -ne “ “                     # Remove empty Array elements
$a[2] = $Null                       # Remove third Array Element
$a[0] , $a[2] , $a[-1 ]             # getting first, third element of array,
                                      last element of array
$result = @( )                      # initializing $result as empty array
```

object manipulation
```
(get-process calc).Workingset/1kb
(get-process calc).WaitForExit()
“abc”.ToUpper()
```

Hashtable
```
$hash = @{Name1 = “Value1”; Name2 = “Value2”}
                                    # create Hashtable
$hash[”Name1 “, ”Name2“]            # read Hashtable
$hash.Name2 = ”Value2“              # expand Hashtable
$hash.remove(”Name1 “)              # remove Hash Entry
$hash = @{ }                        # initializing $hash as empty Hashtable
```

Calculate
```
+, -, *, /                          # basic arithmetic
$i = $i +5                          # increase $i by 5
$i += 1                             # increase $i by 1
$i -= 1                             # decrease $i by 1
$i++                                # increase $i by 1
“abc” + “def”                       # joining strings to abcdef
5%4.5 = 0,5                         # modula
$abc -like ”a*“                     # returns $true if $abc begins with “a”
      -notlike
$array -contains “VM1“              # returns $true if array contains VM1
“book“-match“ [iou]“                # regular expression filter
       -notmatch
-or, -and, -not                     # or, and or not statement
“VM-Name1“  -replace“VM“, “Host“    # Replace String
```

Compare
-eq, -ne, - le, -lt, -ge, -gt
```
$i –eq 123                          # $i equal 123
$i –ne 5                            # $i not equal 5
$i --lt 5                           # $i less than 5
$i –le 5                            # $i less equal 5
$i –ge 5                            # $i greater equal 5
```

PowerShell Basics - 2

Wildcards
```
*                                   # Get-Help about_Wildcards
?                                   # all chars, no limit
[a-j ]                              # all chars, one char
[ajz]                               # chars a, b ... j, one char
                                    # chars a, j, z, one char
```

boolean
```
$i = $true                          # sets $i true
$i = $false                         # sets $i false
```

If statements
```
if (condition) {…}
elseif (condition) {…}
else {…}
if (($i –ge 5) –and ($i –le 7)) {…}
```

Switch, for multiple statements
```
switch (expression) {
Value1 {…}
Value2 {…}
default {…} }
```

Loops
```
while (expression) {…}
while ($i –le 5) {…}
```

Do Loop (run while $a >= 10 (9 times)
```
$a=1
Do {$a; $a++}
While ($a –lt 10)
```

run until $a >= 10 (10 times)
```
$a=1
Do {$a; $a++}
Until ($a –gt 10)
```

For loops
```
for([initializer]; [condition]; [iterator] ) { …}
for($i = 1 ; $i –le 10; $i++) { …}
```

Foreach-Object or Foreach or %
```
foreach(identifier in Collection) {…}
foreach($vm in get-vm) {stop-vm $vm}    # Stop all VMs
foreach($host in get-vmhost) {$host.name}
                                        # Show all Host names
get-vm | % {stop-vm $_}                 # Stop all VMsFunction
function name ([string]$parameter1, [int]$parameter2, …) {...}
function listvm([string]$vm)
{
        $vms = Get-VM $vm
        return $vms
}
$vms = listvm(“VMname“)
```

Piping & Pipeline
PowerShell allows to process the object output of one CmdLet to be piped into Input of another Cmdlet
```
Get-Cluster “CL1“  | Get-VM | Get-CDDrive
                                        # Shows all CD Drives ofVMs within Cluster CL1
```

Object Properties
PowerShell allows to direct access or iterate Object properties and methods directly
```
Get-VM | Get-Member                     # gets information about object
                                          properties and methods
(Get-VM VM1 ).Guest.OSFullName          # retrieves Guest OS Full name ofVM1
```
To suppress confirmation messages append confirm:$false to command

CmdLet Info
```
Get-Help [-name] string] [ {-full | -detailed | -examples }
get-help get-vm
Get-Command [-name] string[] ]
get-command get-v*
Get-Command -commandtype CmdLet get*
                                        # show all Get - CmdLets
Get-Command -PSSnapin VMware.VimAutomation.Core
                                        # show all VI Toolkit CmdLets
```

VMware PowerCLI 1.5 card

Version 1.0

by Dennis Zimmer
icomasoft AG, www.icomasoft.com

PowerCLI 1.5

PowerShell Basics - 3

User Input
```
Read-Host [[-prompt] Object] [-asSecureString] [CommonParameters]
$myinput = Read-Host      $mypass = read-host „Enter a Password:“ -assecurestring
Get-Credential [-credential] PSCredential [CommonParameters]
$cred = Get-Credential                  $aduser = Get-Credential domain\aduser
```

Script Output
```
$variable                           # outputs variable content
$error                              # list last errors
> $NULL 2>&1                        # suppress output, i. e.
                                      connect-viserver vcserver > $NULL 2>&1
“Output of ` $variable: $variable“  # Escape Char ` to output special
                                      characters
“Today´s date: “ + (get-date)       # merge Text with Cmd-Let
Out-Host [-paging] [-inputObject psobject] [CommonParameters]
get-datastore | out-host -paging    # show all datastores one page at a time at console
Out-File [-filePath] string [[-encoding] string] [-append] [-width int] [-inputObject psobject] [-force] [-noClobber]
get-vm | out-file C: \docs\process.txt   # write utput of get-vm overwriting process.txt
Export-Clixml [-path] string -inputObject psobject [-depth int] [-force] [-encoding string] [-noClobber]
Get-VM | Export-CliXML C: \Scripts\Test.xml
                                    # writes XML file
Export-Csv [-path] string -inputObject psobject [Options]
Get-VM | Export-Csv -path c: \vms.csv   # stores all VM data within XML file
Out-GridView [-InputObject <psobject>]  # PowerShell 2 CTP3 & .net Framework 3.5 required
Get-VM | Out-GridView               # Displays a Gridview with the input Objects
                                      - limited to 10 columns
```

Calling external scripts
```
powershell.exe „. c: \script.ps1“
```

Calling scripts within PowerShell
```
.script.ps1 =script usage with variable scope
&script.ps1 = new scope, variables of calling script not valid
powershell -command “& ‘MyScript.ps1‘ “
                                    # Run PowerShell Script with Windows Scheduler
```

Record Sessions
```
Start-transcript -path c: \transcript0.txt
Stop-transcript
                                    # Convert Hashtable to Object - very helpful for
                                      custom attribute or adv. config formatting
function ConvertTo-Object {
begin { $object = New-Object Object }
process {
   $_.GetEnumerator() | ForEach-Object { Add-Member -inputObject $object -memberType
NoteProperty -name $_.Name -value $_.Value } }
end { $object }
}
$hash1 = @{name=‘cust1 ,;value=‘123‘}
$hash1 | ConvertTo-Object
```

.Net Objects
Returns a VMware Infrastructure .Net view object by specified search criteria.
```
$vm = Get-View -ViewType VirtualMachine -Filter @{„Name“ = „XPVM“}
$hostView = (Get-View -ID $vm.Runtime.Host).Summary.Runtime
```

Get VMotion Info
```
(Get-View (Get-Host ‘ESX1‘ | get-view).ConfigManager.VmotionSystem).SelectVnic(,vmk0‘)
```

Get Portgroup VLAN ID
```
Get-VM | %{ Write-Host $_.Name `t ($_ | Get-VirtualPortGroup | Select vlanid)}
```

PowerCLI Installation

InstallWindows PowerShell
Version 1:
http://www.microsoft.com/technet/scriptcenter/topics/msh/download.mspx
Version 2 CTP3:
http://go.microsoft.com/fwlink/?LinkID=131969
Before installing VMware PowerCLI set lower Security:
Set-ExecutionPolicy RemoteSigned
or registry edit:
HKLM\Software\Microsoft\PowerShell\1 \ShellIds\ExecutionPolicy to RemoteSigned
VI Toolkit:
http://www.vmware.com/sdk/vitk_win/index.html
PowerShell Editor PowerGUI:
http://www.powergui.org

VI Toolkit for Windows Community Extensions:
http: //www.codeplex.com/vitoolkitextensions
Get-VIToolkitVersion [<CommonParameters>]
 # get VI Toolkit version
Set-VIToolkitConfiguration -ProxyPolicy NoProxy -Confirm:$false
 # avoid IE proxy

Community Extensions

The VITK Community Extensions is a PowerShell Module that contains enhanced functionality to the official VITK. These extensions are developed by community members.
You need PowerShell version 2 (current CTP3) to be able to load the module with
add-module vitoolkitextension.psm1
Most of the Cmdlets are following the syntax verb-TKEnoun – like Get-TKEVMPath for getting the Path of the VM by given VMX file. Others are non VMware specific Locate, Parse or cast CmdLets.
There is a install.cmd included within the VITKE archive for installing the Toolkit Extension and loading it within profile.ps1
Some of the Extensions help to:
ESX Maintenance Mode Set-TkeVMHostMaintenanceMode

.Net Objects
Returns a VMware Infrastructure .Net view object by specifed search criter
$vm = Get-View -ViewType VirtualMachine -Filter @{„Name“ = „XPVM“}
$hostView = (Get-View -ID $vm.Runtime.Host).Summary.Runtime
Get-VM | Get-View | Get-Member *# Show all .net Properties and Methods of virtual machine objects*

Connect VI Server

To use PowerCLI you have to be connected to a VMware ESX oder VMware VirtualCenter Server.
Connect-VIserver [-Server] <String[]> [-Port <Int32>] [-Protocol <String>] [-Credential <PSCredential>] [-User <String>] [-Password <String>] [-Session <String>] [<CommonParameters>]
Connect-VIServer -Server 10.23.112.235 -Protocol https -User Administrator -Password pass01
$srv = Connect-VIServer 10.23.115.133 -User Admin -Password Pass01
Connect-VIServer 10.23.115.133 -Session $srv.SessionId
You can either connect to one or multiple VMware ESX or VirtualCenter Server.
Connect multiple VirtualCenter Server with current user account:
$vcs = @()
$vcs += Connect-VIServer vc1 .network.test
$vcs += Connect-VIServer vc2.network.test
$vcs += Connect-VIServer vc3.network.test
Get All VMs managed by multiple vCenter server
get-vm -server $vcs
Connect VI Server with credentials:
Connect-VIServer –Server vc.network.test –Credential (Get-Credential)
Don´t forget to disconnect your PowerShell session to the VI server, to avoid a session overflow. If no
Server is given, the command disconnects all current PowerShell sessions.
Disconnect-VIServer [[-Server] <VIServer[]>] [-WhatIf] [-Confirm] [<CommonParameters>]
Disconnect-VIServer -Server $DefaultVIServer
Disconnect-VIServer -Server $DefaultVIServer –confirm:$false -> suppress confirmation message

VMware ESX - 1

To list all the VMware Infrastructure Servers (VMHost) on the connected VI Server
Get-VMHost

Get all Hosts member of a Cluster
Get-Cluster Cluster01 | Get-VmHost

To add a new VMHost to inventory (requires connection to vCenter)
Add-VMHost ESX01 -Location (Get-Datacenter Main) -User root -Password MyPass

Add host using different VPXAgent Port and Credential Dialog
Add-VMHost ESX1 -Location (Get-Datacenter myDC) -Port 910 -Credentials (Get-Credential)

To remove a VMHost
Get-VMHost ESX01 | Remove-VmHost

To move a VMHost
Move-VMHost (Get-VMHost ESX01) -Destination (Get-Datacenter MyDatacenter)

To change the state of a VMHost
Set-VmHost -VmHost ESX01 -State ”Disconnected“
 # Disconnect Host
Set-VmHost -VmHost (Get-VmHost -Name ”ESX01”) -State ”Maintenance“
 # Enter Host

Connect Host or Exit Maintenance
Set-VmHost -VmHost (Get-VmHost -Name ”ESX01”) -State ”Connected“

ESX Advanced Configuration
Get-VMHostAdvancedConfiguration -VMHost (Get-VMHost ESX01) -Name scsi*Log*
Get-VMHost ESX01 | Set-VmHostAdvancedConfiguration -Name ”Scsi.LogMultiPath“ -Value 1

Read and Configure Host Modules (requires direct ESX connection)
$module = Get-VMHostModule -Name qla2300_707_vmw.o
 # Get Infos about QLogic HBAModule

set QLogic HBAModule Option
Set-VMHostModule -HostModule $module -Options“qlport_down_retry=60“

Manage VM StartPolicy of ESX Host
Get-VMHost ESX01 | Set-VMHostStartPolicy -Enabled $true
Get-VMHost | Get-VMHostStartPolicy
Get-VMHost | Get-VMHostStartPolicy | Set-VMHostStartPolicy -StopAction GuestShutDown

ESX Host Accounts (requires direct ESX Host connection)
Get-VMHostAccount -user
New-VMHostAccount -ID user -Password password -UserAccount
New-VMHostAccount -Id normalusers -GroupAccount -AssignUsers user
Get-VMHostAccount - Group -ID user | Remove-VMHostAccount -Confirm

Firewall Policy Configuration
$fwconf = Get-VMHostFirewallDefaultPolicy -VMHost (Get-VMHost ESX01)
Set-VMHostFirewallDefaultPolicy -Policy $fwconf -AllowOutGoing $true
Get-VMHostFirewallException -VMHost (Get-VMHost -Name ESX01) -Enabled $true
$sshfw = Get-VMHostFirewallException -Name ”SSH Client“
$sshfw | Set-VMHostFirewallException -Enabled $true

Service Configuration (requires direct ESX connection)
Get-VMHostService -Refresh
$ntpd = Get-VmHostService -VMHost 192.168.1.10 | ?{$_.key -eq ”ntpd“}
Set-VmHostService -Service $ntpd -Policy „Automatic“
Get-VmHostService -VMHost 192.168.1.10 | ?{$_.key -eq ”vmware-vpxa“} |
Restart-VMHostService
$ntpd | Start-VMHostService
$ntpd | Stop-VMHostService

NTP Server settings
Get-VMHostNtpServer ESX01
Get-VMHost | Sort Name | Select Name, @{N=”NTP”;E={Get-VMHostNtpServer $_} }
Add-VmHostNtpServer -NtpServer ”pool.ntp.org“ -VMHost (Get-VMHost ESX01)

Set DNS Server - Thanks to Luc Dekens
$dnsServers = („192.168.111.3“,“192.168.111.4“)
Get-VMHost | Get-View | %{
$ns = Get-View -Id $esx.configManager.networkSystem
$dns = $ns.networkConfig.dnsConfig
$dns.Address = @()
foreach($server in $dns.Servers) {
$dns.Address += $server
}

VMware ESX - 2

$ns.UpdateDnsConfig($dns)
}

ESX SNMP Configuration (requires direct ESX Host connection)
$hostSNMP = Get-VMHostSNMP
Set-VMHostSNMP $hostSNMP -Enabled:$true -ReadOnlyCommunity ‚public‘

vSphere Hostprofiles
$p = (Get-VMHostProfile -Name testProfile)[0]
Set-VMHostProfile -Profile $p -Description ”Edited the description?“
Test-VMHostProfileCompliance -VMHost (Get-VmHost ESX01)
Get-VMHostProfile -Name ”Profile01” | Remove-VMHostProfile -Confirm:$false
Apply-VMHostProfile -Entity (Get-VmHost ESX01) -Profile $p -Confirm:$false
New-VMHostProfile MasterProfile -Description ”Profile from ESXMaster“
-ReferenceHost (Get-VmHost ESXMaster)
Export-VMHostProfile -FilePath export.prf -Profile $p -Force

Change License Server (Thanks to Hugo Peeters, www.peetersonline.nl)
$SI = Get-View ServiceInstance
$LicMan = Get-View $SI.Content.LicenseManager
$VMHost = Get-VMHost ESX01
$hostref = ($VMHost | Get-View).MoRef
$LicServer = „27000@LicenseServer.domain.local“
 # new license server
$licsrc = New-Object VMware.Vim.LicenseServerSource
$licsrc.LicenseServer = $LicServer
$LicMan.ConfigureLicenseSource($hostref,$licsrc)

Storage

List Datastores
Get-Datastore *# List all DataStores with default properties*
Get-VMHost ESX01 | Get-Datastore *# List all Datastores of ESX Host ESX01*
Get-Datastore | where {$_.type -eq „NFS“}
 # List all NFS Datastores

Create New Datastores
New-Datastore -Nfs -VMHost (Get-VmHost ESX01) -Name NFS01 -Path ”/vol/nfs“ -NfsHost nfs.domain.test
New-Datastore -Vmfs -VMHost (Get-VmHost ESX01) -Path ”vmhba1:0:4“ -Name LUN01

Delete Datastores
Remove-Datastore -Datastore (Get-Datastore -Name NFS01) -VMHost ESX01 -Confirm:$false

Change Datastore Friendly Name
Get-Datastore -Name ”NFS01” | Set-Datastore -Name oldNFS

Storage Pathing
$scsilun = Get-ScsiLun -VMHost ESX01 -LunType disk
$scsilun | Set-ScsiLun -CommandsToSwitchPath 100
$scsilun | Set-ScsiLun -MultipathPolicy Fixed
$scsipath = Get-ScsiLunPath $scsilun
Set-ScsiLunPath -ScsiLunPath $scsipath -Preferred $true -Active $true

Rescan or Refresh Datastores
Get-VmHostStorage (Get-VmHost ESX01) -Refresh
 # Refresh Datastores
Get-VmHostStorage (Get-VmHost ESX01) -RescanAllHba
 # Rescan SAN
Get-VmHostStorage (Get-VmHost ESX01) -RescanVmfs
 # Rescan VMFS
Get-VMHostStorage | select -expandproperty scsilun | fl *
 # Read Storage information

Task Information

To list all tasks for a VI Server and some of their properties
Get-Task -Server (Connect-VIServer -Server 10.23.112.235) -Status Error

To stop a Task (Example stops the task of removing the VM)
Stop-Task -Task (Remove-VM -VM (Get-VM -Name ”MS Win XP SP2“) -Confirm -RunAsync)

To wait until a task is completed before continuing
Wait-Task -Task (Remove-VM -VM (Get-VM -Name ”MS Win XP SP2“) -Confirm -RunAsync)

Logs & Stats

Log Files
```
$esxhost = Get-VMHost -State "Connected"
$log = Get-Logtype -VMHost $esxhost   # Get available Log Types
Get-Log -Key $log[0]                  # Get first Log file
$esxhost | Get-Log -Key vmkwarning    # Get vmkwarning entries
Get-Log –bundle                       # Get diagnostic log file bundle aka vm-support
```

Get all Logfiles with size
```
Get-LogType $esxhost | % {$logdata = (Get-Log $_.Key $esxhost); $size = $logdata.
LastLineNum -
$logdata.StartLineNum; "{0} :{1 }" -f $_.key, $size}
```

Statistics
```
Get-StatType -Entity (Get-VmHost ESX01 )
                                      # available ESX host statistics
Get-StatType -Entity (Get-VM XPVM)    # available Virtual Machine statistics
Get-StatInterval                      # Get available Statistics interval
Get-Stat $esxhost -common -maxsamples 1 -realtime
                                      # CPU, disk, memory and network statistics
Get-Stat $esxhost -Memory -maxsamples 3 -realtime
Get-Stat -Entity (Get-VM XPVM ) -Start 1 /2/2008 -Finish 30/4/2009 -Disk -IntervalSecs 300
```

Network

To list all Virtual Switches attached to a VM and some of their properties use:
```
Get-VirtualSwitch -VM (Get-VM -Name "VM1")
```

To create a new Virtual Switch:
```
New-VirtualSwitch -VMHost (Get-VMHost -Name ESX1 ) -Name vSwitch02
```

To Remove a Virtual Switch:
```
$vs = Get-VirtualSwitch -VMHost (Get-VMHost -Name ESX1 ) -Name vSwitch01
```

Remove-VirtualSwitch -VirtualSwitch $vs

To change the configuration of a Virtual Switch:
```
$vs = New-VirtualSwitch -Host (Get-VMHost ESX1 ) -Name VirtSwitch
Set-VirtualSwitch -VirtualSwitch $vs -MTU 500
```

To list all the port groups and some of their properties:
```
$vs = Get-VirtualSwitch -VMHost (Get-VMHost -Name ESX1 ) -Name vSwitch02
Get-VirtualPortGroup -VirtualSwitch $vs
```

To add a new port group to a virtual switch:
```
$vs = Get-VirtualSwitch -VMHost (Get-VMHost -Name ESX1 ) -Name vSwitch02
$vpg = New-VirtualPortGroup -VirtualSwitch $vs -Name VPG1
```

Remove-PortGroup
```
Set-PortGroup
```

ESX Host Network Configuration
```
$hostnet = Get-VMHostNetwork (Get-VmHost ESX01 )
New-VMHostNetworkAdapter -PortGroupVMKernel -VirtualSwitch $vs -IP 192.168.5.10
-SubnetMask 255.255.255.0
Set-VmHostNetwork -Network $hostnet -VMKernelGateway 192.168.5.254 -DomainName
domain.test -HostName ESX01 -DnsFromDhcp $false
```

Change Physical Network Settings (100 MBit, Full Duplex)
```
Get-VMHostNetwork | Select PhysicalNic | Set-VMHostNetworkAdapter -Duplex „Full"
-BitRatePerSecMb 100
```

Cluster - 1

To list all the Clusters on the connected VI Server and some of there properties
```
Get-Cluster
```

To add a new Cluster
```
New-Cluster -Name MyCluster -DRSEnabled -DRSMode FullyAutomated
```

Cluster - 2

Move a Cluster
```
Move-Cluster (Get-Cluster MyCluster) -Destination (Get-Folder MyFolder)
```

Remove a Cluster
```
Get-Cluster "CL01" | Remove-Cluster -Confirm:$false
```

Configure a Cluster
```
Get-Cluster "CL01" | Set-Cluster -Name CL02 - -HAEnabled $true
```

Retrieve a list of DRS rules for the specified clusters
```
$drsrule = Get-DrsRule -Cluster (Get-Cluster "Production") -Name "*Rule99*"
```

New DRS Rule
```
Get-Cluster „CL01 „ | New-DrsRule -Name Rule99 -KeepTogether $false -VM $vms
```

Delete DRS Rule
```
Remove-DrsRule $drsrule -Confirm:$false
```

Change DRS Rule
```
Set-DrsRule -Rule $drsrule -VM $vms -Enabled $true
```

Reporting

HTML Basic Report
```
$a = "<style>"
$a += "body { background-color:#EEEEEE; }"
$a += "body,table,td,th { font-family:Tahoma; color:Black; Font-Size:10pt }"
$a += "th { font-weight:bold; background-color:#CCCCCC; }"
$a += "td { background-color:white; }"
$a += "</style>"
```

Export Table Report as HTML File
```
$report | ConvertTo-Html -head $a | Out-File "report.htm"
```

Send SMTP Mail
```
$msg = new-object Net.Mail.MailMessage
```

optional Attachment
```
$att = new-object Net.Mail.Attachment("c: \report.csv")
$msg.Attachments.Add($att)
```

Mailserver, Sender and Receiver
```
$smtp = new-object Net.Mail.SmtpClient("smtpserver.domain.test)
$msg.From ="somebody@yourdomain.com"
$msg.To.Add("somebody@theirdomain.com")
```

Subject & Body Content
```
$msg.Subject = "Virtual Infrastructure Report"
$msg.Body = "Body Text: Please find the VM Report attached"
$smtp.Send($msg)                      # Send the Email
if ($att) {$att.dispose()}            # Clear Attachment locking
```

Virtual Machines - 1

To list VM's on the connected VI Server and some of there properties
```
Get-VM                                # show all VMs of the currently connected instance
Get-Cluster CL1 | Get-VM              # show all VMs running in Cluster CL1
Get-VmHost "ESX01" | Get-VM           # show all VMs registered on host ESX01
```

Change the configuration of a VM
```
Set-VM -VM (Get-VM -Name "Win XP SP1") -Name "Win XP SP2" -GuestId "winXPProGuest"
-Description "My updatedWin XP virtual machine."
```

Change VM Power State
```
Get-VM XPVM | Start-VM                # Start a VM
Get-VM XPVM | Start-VM -runasync      # Start a VM asynchron (not waiting for task ending)
Get-VM XPVM | Stop-VM                 # Stop a VM
Stop-VM -VM XPVM -confirm:$false      # Stop VM without asking
Get-VM | Suspend-VM                   # Suspend a VM
```

Virtual Machines - 2

Guest OS Operations (requires VMware Tools installed)
```
Get-VMGuest -VM XPVM                  # Get OS information
Get-VM XPVM | Restart-VMGuest         # restart Guest OS
Get-VM XPVM | Suspend-VMGuest         # suspend Guest OS
```

VM Storage and removal medias
```
Get-HardDisk -VM XPVM                 # get virtual disks
Get-VM XPVM | New-HardDisk -CapacityKB 1 0240000
                                      # create 10 GB disk
Get-VM XPVM | Get-HardDisk | Set-HardDisk -Persistence NonPersistent
$flp = Get-FloppyDrive -VM XPVM
New-FloppyDrive -VM XPVM -HostDevice ,/dev/fd0' -StartConnected
Remove-FloppyDrive -Floppy $flp
$flp | Set-FloppyDrive -NoMedia
$cddrive = Get-CDDrive -VM XPVM
New-CDDrive -VM XPVM -ISOPath "Path_to_ISO\test.iso"
Remove-CDDrive -CD $cddrive
Set-CDDrive -CD $cddrive -Connected $false
```

Snapshots
```
Get-VM | Get-Snapshot                 # To list all the snapshots for all virtual machines
New-Snapshot -VM(Get-VM -Name "XP SP2") -Name Patch1 # snapshot a VM
Get-Snapshot snapshot1 | Remove-Snapshot
                                      # remove specific snapshot
Get-Snapshot snapshot1 | Remove-Snapshot –RemoveChildren
                                      # remove a snapshot with children
Get-VM XPVM | Get-Snapshot | Remove-Snapshot
                                      # remove all snapshots of vm
```

Change Snapshot
```
Set-Snapshot -Snapshot snapshot1 -Name 'AfterPatch1' -Description "After Patch"
```

Custom Fields
```
New-CustomField -Entity (Get-VM -Name "VM1") -Name CustFieldname -Value 100
```

Read Custom Field
```
(Get-VM "VM1").CustomFields.get_item("CustFieldname")
```

Set Custom Field
```
Get-VM "VM1" | Set-CustomField -Name CustFieldname -Value 10
```

Remove Custom Field
```
Remove-CustomField -Entity (Get-VM -Name "VM1") -Name CustFieldname
```

VMware Tools
```
Get-VMGuest XPVM| Update-Tools        # Updates VMware Tools
Get-VMGuest XPVM| Mount-Tools         # mount VMware Tools CD-ROM
Get-VMGuest XPVM| DisMount-Tools      # unmount VMware Tools CD
```

Update Manager

Important: Because of an incompatibility issue with PowerCLI Version 1.5 you have to use VI Toolkit Version 1.0 at the moment to work with the Update Manager extension

Choose "Install VMware Update Manager Toolkit" during installation ofVMware Update Manager Plugin from VI Client.
```
Add-PSSnapIn VMware.VumAutomation
                                      # To load the PowerShell Snapin for Update Manager
Get-Command -PSSnapIn VMware.VumAutomation
                                      # Get all CmdLets ofUpdate Manager
```

Get not compliant Updates ofVMware ESX Host
```
(Get-Compliance (Get-VMHost (Read-Host "ESX Server"))).NotCompliantUpdates
```

VMware.VUMautomation CmdLets
```
Attach-Baseline, Detach-Baseline, New-Baseline
Download-Update
Get-Baseline, Set-Baseline
Get-Compliance
Get-Update
Get-VumConfig
Set-VumConfig
Remediate-Inventory
Scan-Inventory
```

Provision VM

To create a new VM
```
$esxhost = Get-VMHost "ESXHost01 .mydomain.com"
New-VM -Name XPVM -VMHost $esxhost -DiskMB 4000 -MemoryMB 256
```

To Remove a VM:
```
Remove-VM ( Get-VM "myVM" ) –DeleteFromDisk
```

Templates
```
$template = Get-Template -Name Templ* -Location (Get-Datacenter DC)
Remove-Template -Template $template
New-Template -VM ( Get-VM XPVM ) -Name Template01 -Location (Get-Datacenter DC)
Set-Template -Template $template -ToVM
                                    # convert Template to VM
```

Guest Customization Specification
```
$osspec = Get-OSCustomizationSpec WinXP
                                    # reads Guest Customization Specification object
New-VM -Name MyVM2 -Template Template01 -VMHost ESX01 -OSCustomizationSpec $osspec
```

other OS Customization CmdLets:
```
New-OSCustomizationSpec
Set-OSCustomizationSpec
```

DataCenter

To list all datacenters from a VMware Infrastructure server
```
Get-Datacenter
```

To add a new datacenter
```
New-Datacenter -Name DC2
```

Move a Datacenter
```
Move-Datacenter (Get-Datacenter MyDatacenter) -Destination (Get-Folder MyFolder)
```

Remove a DataCenter
```
Get-Datacenter "DatacenterDelete" | Remove-Datacenter
```

Change DataCenter Name
```
Get-Datacenter "DatacenterDelete" | Set-Datacenter -Name Datacenter-old
```

Resource Pools

To list all Resource Pool's on the connected VI Server and some of their properties
```
Get-ResourcePool
```

To create a new resource pool
```
$clusterRootRP = Get-ResourcePool -Location ( Get-Cluster ResearchAndDevelopmentCluster )
-Name Resources
New-ResourcePool -Location $clusterRootRP -Name DevelopmentResources
-CpuExpandableReservation $true -CpuReservationMhz 500 -CpuSharesLevel high
-MemExpandableReservation $true -MemReservationMB 500 -MemSharesLevel high
```

Move a Resource Pool
```
Move-ResourcePool -ResourcePool (Get-ResourcePool RP) -Destination (get-VmHost ESX01 )
```

Delete a Resource Pool
```
Remove-ResourcePool -ResourcePool (Get-ResourcePool RP)
```

Configure Resource Pool
```
Set-Resourcepool (Get-Resourcepool -Name "RP") -NumCpuShares 2048 -MemLimitMB 4096
```

Sort, Filter & Format

Filtering
```
where-object, where or ?
Get-Datastore | where-object {$_.FreeSpaceMB -le 5120}
Get-Datastore | ? {$_.FreeSpaceMB -le 5120}
```

Sorting
```
Get-datastore | sort Name
```

Format-Table or ft
```
Get-VM | ft *                       # show all properties as table
Get-VM | ft -autosize               # optimized output
Format-List or fl                   # like Format-Table but shows list view instead
                                      of table view
Get-VM | format-list *              # show all properties as list
```

Select-Object (or select) and Select-String
```
Get-VM | Select-Object Name, PowerState
                                    # selects Name and Powerstate out of all properties
ipconfig /all | select-string "Physical Address"
                                    # Select-String searches for specific string
Get-VM | ?{$_.powerstate -eq "PoweredOn"} | measure-object
                                    # counts objects
```

Build report with Hashtable
```
$report = @()
$hash = @{}
Get-VM | % {
$hash = "" | select VMName, CPU, Memory
$hash.VMName = $_.Name
$hash.CPU = $_.MemoryMB
$hash.Memory = $_.NumCPU
$report += $hash
}
$report                             # Report contains Name, Memory and CPU of the
                                      virtual machine
```

Migration

VMotion
```
Move-VM [-Destination <VIContainer>] [-Datastore <Datastore>] [-RunAsync] [-VM]
<VirtualMachine[]>
```

VMotion virtual machine
```
Get-VM myVM | Move-VM -destination (Get-VMHost ESXHost) -confirm:$false
```

VMotion virtual machine async
```
Move-VM -VM (Get-VM "myVM") -RunAsync -Destination (Get-VMHost "ESXHost")
```

Storage VMotion
```
Get-VM myVM | Move-VM -datastore (Get-Datastore "DS-Name") -confirm:$false
Get-VM | SVMotion-VM -destination (get-datastore TargetDS)
```

Cold Migration
```
Get-VM myVM | Stop-VM -confirm:$false
Get-VM myVM | Move-VM -destination (Get-VMHost ESXHost) -confirm:$false
```

Quick Migration
```
$vm = Get-VM "myVM"                 # Store VM object in $vm variable
Suspend-VM -VM $vm -confirm:$false
                                    # Suspend VM
$vm | Move-VM -Destination (Get-VMHost ESXHost) -datastore ($vm | get-datastore)
Start-VM -VM $vm                    # Resume VM
```

Invoke - VMScript

PowerCLI required 1.5 with VIX API installed
Executes the specified PowerShell script in the guest OS of each of the specified virtual machines.
Level of access

1) Read access to the VM including folder. (you need to be able to run Get-VM after all)
2) Virtual Machine.Interaction.Console Interaction privilege
3) Windows guest with up-to-date tools and powered on state
4) Windows PowerShell installed within the Windows guest
5) Network connectivity to the ESX system hosting the VM on port 902
6) User Credentials of ESX Server and Virtual Machine Guest OS

```
Invoke-VMScript [-ScriptText] <String> [-VM] <VirtualMachine[]> [-HostCredential
<PSCredential>] [-HostUser <String>] [-HostPassword <SecureString>]
[-GuestCredential <PSCredential>] [-GuestUser <String>] [-GuestPassword <SecureString>]
[-ToolsWaitSecs <Int32>] [-Server<VIServer[]>]
```

Example
```
Invoke-VMScript -VM (Get-VM myVM) -ScriptText "dir" -HostUser root -HostPassword mypass
-GuestUser administrator -GuestPassword mypas
```

VI PowerScripter

Download and install VI PowerScripter Pro:

http: //www.icomasoft.com/powerscripter
no Connect-VIServer or Disconnect-VIServer needed, because handled by PowerScripter.
Import PowerShell (.ps1) or Texfiles with Wizard or put it into the HostScripts or VMscripts folder: %programfiles%\"VMware\Infrastructure\Virtual Infrastructure Client\Plugins\icomasoft PowerScripter"

parameter within PowerScripter PowerShell files, begin with comment char #, like #--name

#--name = Name for context menu entry
#--hideoutput = no PowerShell output window
#--multi = run script once, putting all selected objects (Hosts or VMs) into $multi variable
Variables within PowerScripter Script Files

```
$_                                  # (Current Object - script runs for every selected
                                      object)
$multi                              # (array of selected objects (Hosts or VMs))
```

Builtin Reporting CmdLets
```
Get-VM | Out-GridviewIS             # visualizes all VMs, allowing sorting, filtering,
                                      exporting andconditional formatting
Get-VM | Export-CsvIS -Path c: \vms.csv
                                    # Exports all VMs with properties into CSV File
Get-VM | Export-ExcelIS -Path c: \vms.xls
                                    # Exports all VMs with properties into Excel File
```

Connect/Disconnect SSH Session
```
Connect-SSH -Server 192.168.1.201 -User root -Password letmein
                                    # connect to a SSH server
Disconnect-SSH                      # Disconnects SSH Session
Run-SSHCmd "vdf -h"                 # Run given command within SSH connection
```

Run given command after connecting to SSH Server
```
Run-SSHCmd "esxcfg-mpath -l" -Server 192.168.1.201 -user root -password letmein
```

Copy files using the SSH protocol - existing or new SSH Session can be used
```
Run-SCP -Source 192.168.1.201:/etc/vmware/esx.conf -Destination c: ESXhost-config
Run-SCP 192.168.1.201:/etc/vmware/esx.conf c:\ESXhost-config -user root -password letmein
```


Thanks to:

Alan Renouf, contribution of content
- http://www.virtu-al.net

Forbes Guthrie, sharing his great Reference Design
- http://www.vmreference.com/

And Thanks to:

Urs Stephan Alder, for hosting the test infrastructure
- http://www.d-on-d.com

VMware Community
- for awesome work in publishing scripts and scriptlets