```
## vCenter_Update_vSphere_w_VUM_dyn_listbox.ps1
## build 1.0
## Eric Krejci
## ekrejci.wordpress.com
## Twitter - @ekrejci


## on error stop the script
$erroractionpreference = "Stop"

## Checking if the VMware Update Manager Snapin is loaded. else quit.

if (((Get-PowerCLIVersion).SnapinVersions | where { $_.name.endswith(
"VumAutomation") }) -eq $null) {

Write-Host "The Snapin for VMware Update Manager is not loaded and the current
script cannot continue."
exit

}

## connect to vCenter

$Username = "DOMAIN\USERNAME"
$Password = read-host "Enter Password" -assecurestring
$vCenterFQDN = "vcenter.fqdn"

Connect-VIServer -Server $vCenterFQDN -user $Username -Password ([Runtime.
InteropServices.Marshal]::PtrToStringAuto([Runtime.InteropServices.Marshal]::
SecureStringToBSTR($password)))  | Out-Null


### Functions

## call-list box with every vSphere present in a datacenter.

function call-listbox {
  Param( $DataCenter  ## This is the DataCenter
        )

[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms")
[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Drawing")

$objForm = New-Object System.Windows.Forms.Form
$objForm.Text = "Select a Host"
$objForm.Size = New-Object System.Drawing.Size(300,200)
$objForm.StartPosition = "CenterScreen"

$objForm.KeyPreview = $True
$objForm.Add_KeyDown({if ($_.KeyCode -eq "Enter")
    {$x=$objListBox.SelectedItem;$objForm.Close()}})
$objForm.Add_KeyDown({if ($_.KeyCode -eq "Escape")
    {$objForm.Close()}})

$OKButton = New-Object System.Windows.Forms.Button
$OKButton.Location = New-Object System.Drawing.Size(75,120)
$OKButton.Size = New-Object System.Drawing.Size(75,23)
$OKButton.Text = "OK"
$OKButton.Add_Click({$x=$objListBox.SelectedItem;$objForm.Close()})
$objForm.Controls.Add($OKButton)

$CancelButton = New-Object System.Windows.Forms.Button
```

```
$CancelButton.Location = New-Object System.Drawing.Size(150,120)
$CancelButton.Size = New-Object System.Drawing.Size(75,23)
$CancelButton.Text = "Cancel"
$CancelButton.Add_Click({$objForm.Close()})
$objForm.Controls.Add($CancelButton)

$objLabel = New-Object System.Windows.Forms.Label
$objLabel.Location = New-Object System.Drawing.Size(10,20)
$objLabel.Size = New-Object System.Drawing.Size(280,20)
$objLabel.Text = "Please select a Host:"
$objForm.Controls.Add($objLabel)

$objListBox = New-Object System.Windows.Forms.ListBox
$objListBox.Location = New-Object System.Drawing.Size(10,40)
$objListBox.Size = New-Object System.Drawing.Size(260,20)
$objListBox.Height = 80
$objListBox.Sorted = $True

### End Functions

## build the list of every hosts in the Datacenter beeing or connected or in
maintenance
$VMHosts = Get-VMHost -Location (Get-Datacenter -name $DataCenter) | where {$_
.ConnectionState -eq "Connected" -or $_.ConnectionState -eq "Maintenance"} |
sort

foreach ($VMHost in $VMHosts) {

[void] $objListBox.Items.Add($VMHost)
}


$objForm.Controls.Add($objListBox)

$objForm.Topmost = $True

$objForm.Add_Shown({$objForm.Activate()})
[void] $objForm.ShowDialog()

## check if you did not select any host it will quit.
if ($x -ne $null ) {

$x

} else {

Write-Host "you canceled"
exit
}

}

## Call the list of VMhosts in the Datacenter

$DCName = "DATACENTER"

$VMHost = call-listbox -DataCenter $DCName

## ask if you want to download the last patch for Update Manager:

if ("Y" -eq ((Read-Host "Do you want to download the last patch for Update
Manager? Enter Y or N").ToUpper())) {
```

```
Download-Patch

}


# scan the VMhost

Scan-Inventory -Entity $VMHost


## check if any baselines are somehow attached to the host? if none => exit

$VMHostBaselines = Get-Compliance -Entity $VMHost

if ($VMHostBaselines -eq $null) {

write-host "no baseline attached"
exit
}

## get the compliance status of the previous scan for the Baseline of path
only and for which it is NotCompliant


$NotCompliantBaselines = $VMHostBaselines | where { $_.status -like
"NotCompliant" -and $_.Baseline.BaselineType -like "Patch"} | select -
ExpandProperty "Baseline"




## now let's remediate the needed baselines against the host.
if ($NotCompliantBaselines -ne $null) {


## Set the Server to Maintenance Mode
## This casts VMHost to an ESX server object which exposes the .ConnectionState
"Check if the VMHost is in maintenance mode"


## is Server not under maintenance, ask if you want to. else exit
if ($VMHost.ConnectionState -eq "Connected") {
  if ("Y" -eq ((Read-Host "Do you want to set '" $VMHost.Name "' to
maintenance? Enter Y or exit").ToUpper())) {

    Set-VMHost -VMHost $VMHost -State Maintenance | Out-Null
    "VMHost set to maintenance mode"
    } else {

    "exiting"
    exit

    }
  }
else {

  "VMHost current has state '" + $VMHost.ConnectionState + "'. `nNot changed."
}
```

```
"Remediate " + $VMHost.Name

## the NotCompliantBaselines contains every needed baseline. like that the
host will only reboot once at the end.

Remediate-Inventory -Baseline $NotCompliantBaselines -Entity $VMHost -
HostDisableMediaDevices:$true  -Confirm:$false

} else {

"No updates needed for " + $VMHost.Name

}


"Finished"

Disconnect-VIServer -Confirm:$false
```