```powershell
########################################
### VM Provisioning Module
### ECI.EMI.Automation.VM.Prod.psm1
########################################

function Get-ECI.EMI.Automation.VM.VMTemplate
{
    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

    Write-Host "Getting VMWare Template : $VMTemplateName"  -ForegroundColor DarkCyan

    #$ECIVMTemplate = $VMTemplateName
    $ECIVMTemplate = Get-Template -Name $VMTemplateName -Server $vCenter -ErrorAction
    SilentlyContinue

    if(($ECIVMTemplate.count) -gt 1)
    {
        $ECIVMTemplate = $ECIVMTemplate[0]
    }
    if(-NOT $ECIVMTemplate)
    {
        $ErrorMsg = "ECI.ERROR: Template Not Found"
        Write-Error -Message $ErrorMsg -ErrorAction Continue -ErrorVariable +ECIError
        Send-ECI.Alert -ErrorMsg $ErrorMsg
    }

    $global:ECIVMTemplate = $ECIVMTemplate
    Write-Host "Template Selected       : $ECIVMTemplate" -ForegroundColor Cyan

    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray

    Return $ECIVMTemplate
}

function Set-ECI.EMI.Automation.VM.ServerUUID
{
    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

    ### Set VM Name fromn HostName Parameters
    ###-------------------------------------------------------------------

    $VMID = (Get-VM -Name $VMName).ID
    $VMID = $VMID.Split("-")

    #$VMID = $VMID[-2].ToUpper() + "-" + $VMID[-1]
    $VMID = $VMID[-2] + "-" + $VMID[-1]

    $VMUUID = Get-VM $VMName | %{(Get-View $_.Id).config.uuid}

    #$ServerUUID = "VI" + "." + $VMUUID + "." + ($VMID[-2]).ToUpper() + "-" + $VMID[-1]
    $ServerUUID = "VI" + "." + $VMUUID + "." + $VMID[-2] + "-" + $VMID[-1]

    $global:VMUUID       = $VMUUID
    $global:VMID         = $VMID
    $global:ServerUUID   = $ServerUUID

    Write-Host "vCenterUUID   : " $vCenterUUID -ForegroundColor DarkCyan
    Write-Host "VMID          : " $VMID        -ForegroundColor DarkCyan
    Write-Host "ServerUUID    : " $ServerUUID  -ForegroundColor Cyan

    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
    Return $ServerUUID

    #Example:
    #---------
    #VI-d8c273b7-6f4e-4ce7-8986-72dfbd3f0376-VM-38002
}
```

```powershell

##############################################
### Create New VM  - ECI Cmdlet
##############################################

function New-ECI.EMI.Automation.VM
{
    Param(
    [Parameter(Mandatory = $True)][string]$ConfigurationMode,
    [Parameter(Mandatory = $True)][string]$VMName,
    [Parameter(Mandatory = $True)][string]$ECIVMTemplate,
    [Parameter(Mandatory = $True)][string]$OSCustomizationSpecName,
    [Parameter(Mandatory = $True)][string]$ResourcePool,
    [Parameter(Mandatory = $True)][string]$OSDataStore,
    [Parameter(Mandatory = $True)][string]$PortGroup,
    [Parameter(Mandatory = $True)][string]$IPv4Address,
    [Parameter(Mandatory = $True)][string]$SubnetMask,
    [Parameter(Mandatory = $True)][string]$DefaultGateway,
    [Parameter(Mandatory = $True)][string]$PrimaryDNS,
    [Parameter(Mandatory = $True)][string]$SecondaryDNS
    )

    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

    function New-ECI.EMI.Automation.VM.OSCustomizationSpec
    {
        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

        ###-------------------------------------------------------------
        ### Create New OSCustomizationSpec
        ###-------------------------------------------------------------
        ### Remove OSCustomizationSpec
        if(Get-OSCustomizationSpec $OSCustomizationSpecName -ErrorAction
        SilentlyContinue)
        {
            Write-Host "Removing OSCustomizationSpec   :" $OSCustomizationSpecName
            Remove-OSCustomizationSpec $OSCustomizationSpecName -Confirm:$false

            ### Remove OSCustomizationNicMapping
            #Get-OSCustomizationSpec $OSCustomizationSpecName |
            Get-OSCustomizationNicMapping | Remove-OSCustomizationNicMapping
            -Confirm:$false
        }
        else
        {
            Write-Host "No OSCustomizationSpec Found: "
        }

        ### New OSCustomizationSpec
        ###------------------------
        $OSCustomizationSpec = @{
            Name              = $OSCustomizationSpecName
            Type              = $Type
            OSType            = $OSType
            NamingScheme      = $NamingScheme.trim()
            FullName          = $FullName
            OrgName           = $OrgName
            AdminPassword     = $AdminPassword
            ChangeSid         = $True
            DeleteAccounts    = $False
            TimeZone          = $TimeZone
            ProductKey        = $ProductKey
            LicenseMode       = $LicenseMode
            Workgroup         = $Workgroup
        }

        Write-Host "Creating - OSCustomizationSpec : $OSCustomizationSpecName"
```

```
        -ForegroundColor Cyan
131         New-OSCustomizationSpec @OSCustomizationSpec
132         Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor
            DarkGray
133     }
134
135     function New-ECI.EMI.Automation.VM.OSCustomizationNicMapping
136     {
137         $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
138         ###-------------------------------------------------------------
139         ### Create New OSCustomizationSpec NIC Mapping
140         ###-------------------------------------------------------------
141         $OSCustomizationNicMapping = {
142             IpMode          = "UseStaticIp"
143             IpAddress       = $IPv4Address
144             SubnetMask      = $SubnetMask
145             DefaultGateway  = $DefaultGateway
146             Dns             = $PrimaryDNS + "," + $SecondaryDNS
147         }
148         Write-Host "Creating - OSCustomizationNicMapping  : $OSCustomizationSpecName"
            -ForegroundColor Cyan
149         #Get-OSCustomizationSpec $OSCustomizationSpecName |
            Get-OSCustomizationNicMapping | Set-OSCustomizationNicMapping
            @OSCustomizationNicMapping
150         $NicMapping = @{
151
152          IPMode         = "UseStaticIp "
153          IpAddress      = $IPv4Address
154          SubnetMask     = $SubnetMask
155          DefaultGateway = $DefaultGateway
156          Dns            = $PrimaryDNS,$SecondaryDNS
157         }
158
159         Get-OSCustomizationSpec $OSCustomizationSpecName |
            Get-OSCustomizationNicMapping | Set-OSCustomizationNicMapping -IPMode
            UseStaticIp -IpAddress $IPv4Address -SubnetMask $SubnetMask -DefaultGateway
            $DefaultGateway -Dns $PrimaryDNS,$SecondaryDNS
160
161         Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor
            DarkGray
162     }
163
164     function New-ECI.EMI.Automation.VM.VM
165     {
166         $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
167
168         ###-------------------------------------------------------------
169         ### Create New VM
170         ###-------------------------------------------------------------
171
172         ### Check if VM exists
173         ###-------------------------
174         try
175         {
176             $VMExists = Get-VM -Name $VMName -ErrorAction SilentlyContinue ###<----
                NOTE: This gerenerats a false error if the VM does not exist.
177             Write-Host "ExceptionType: " $global:Error[0].Exception.GetType().fullname
                -ForegroundColor Magenta
178         }
179         catch [VMware.VimAutomation.Sdk.Types.V1.ErrorHandling.VimException.VimException]
180         {
181             Write-Host "This VM Does Not Exists" -ForegroundColor Magenta
182         }
183         catch
184         {
185             Write-ECI.ErrorStack
186         }
```

```
187
188         if($VMExists)
189         {
190             ### Do we want to Abort or run in Report mode ????????
191
192             ### Run in Report Mode
193             ###-------------------------
194             #Write-Host "This VM Exists. Running Report Mode" -ForegroundColor Red
195             #$global:ConfigurationMode = "Report"
                ### <---------- ConfigurationMode
196
197             ### Throw Abort
198             ###-------------------------
199             $global:Abort = $True
200
201             Write-Host "Invoking Abort Error!!!" -ForegroundColor red
202             Invoke-ECI.Abort
203         }
204
205         ### VM does not exist
206         ###-------------------------
207         else
208         {
209             ### Create New VM
210             ###-----------------------------------------------------------
211             Write-Host "Creating New VM       : " $VMName -ForegroundColor Cyan
212             Write-Host "Please wait. The VM Provisioning process may take a while . .
                .  " -ForegroundColor Yellow
213             try
214             {
215                 $VMParameters = @{
216                     VMName                  = $VMName
217                     Template                = $ECIVMTemplate
218                     vCenterFolder           = $vCenterFolder
219                     OSCustomizationSpec     = $OSCustomizationSpecName
220                     ResourcePool            = $ResourcePool
221                     OSDataStore             = $OSDataStore
222                 }
223
224                 $ScriptBlock =
225                 {
226                     ### Without Folder
227                     #New-VM @VMParameters
228                     #New-VM -Name $VMName -Template $ECIVMTemplate -ResourcePool
                        $ResourcePool -Datastore $OSDataStore -OSCustomizationSpec
                        $OSCustomizationSpecName
229
230                     ### With Folder
231
232                     ### New VM
233                     $NewVMParams = {
234                         Name                = $VMName
235                         Template            = $ECIVMTemplate
236                         Location            = $vCenterFolder
237                         ResourcePool        = $ResourcePool
238                         Datastore           = $OSDataStore
239                         OSCustomizationSpec = $OSCustomizationSpecName
240                     }
241                     #New-VM @NewVMParams
242
243                     if($VMName.count -eq 1 -and $VMName -isnot [system.array])
244                     {
245                         New-VM -Name $VMName -Template $ECIVMTemplate -Location
                            $vCenterFolder -ResourcePool $ResourcePool -Datastore
                            $OSDataStore -OSCustomizationSpec $OSCustomizationSpecName
246                     }
247                     else
248                     {
249                         Write-Error -Message "ECI.Error: VMName is not Unique."
```

```
                        -ErrorAction Continue -ErrorVariable +ECIError
250                 }
251             }
252
253             if($ConfigurationMode -eq "Configure")                                  ###
                <---------- ConfigurationMode
254             {
255                 try
256                 {
257                     Invoke-Command -ScriptBlock $ScriptBlock -ErrorVariable +ECIError
258                 }
259                 catch
260                 {
261                     Write-ECI.ErrorStack
262                 }
263             }
264             if($ConfigurationMode -eq "Report")                                     ###
                <---------- ConfigurationMode
265             {
266                 Write-Host "ECI-WHATIF-COMMAND: " $ScriptBlock
267                 Write-ECI.EMI.Report -Report $ScriptBlock
268             }
269
270         }
271
272         catch
273         {
274             Write-ECI.ErrorStack
275         }
276     }
277
278     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor
        DarkGray
279 }
280
281    &{
282     BEGIN {}
283
284     PROCESS
285     {
286         New-ECI.EMI.Automation.VM.OSCustomizationSpec
287         New-ECI.EMI.Automation.VM.OSCustomizationNicMapping
288         New-ECI.EMI.Automation.VM.VM
289     }
290
291     END {}
292
293  }
294
295  Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
296 }
297
298 function Set-ECI.EMI.Automation.VM.VMCPU
299 {
300     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
301
302     ########################################
303     ### Modify Parameter Values
304     ########################################
305     ### Cast the Variable
306     [int]$vCPUCount = $vCPUCount
307
308     ########################################
309     ### DESIRED STATE PARAMETERS
310     ########################################
311     $PropertyName      = "vCPUCount"
312     $DesiredState      = $vCPUCount
313     #ConfigurationMode = "Configure" ### Report - Configure
```

```powershell
    $AbortTrigger      = $False  ### $True - $False

    ########################################
    ### GET CURRENT CONFIGURATION STATE:
    ########################################
    [scriptblock]$script:GetCurrentState =
    {
        $global:CurrentState = (Get-VM $VMName | Select NumCpu).NumCpu
    }

    ########################################
    ### SET DESIRED-STATE:
    ########################################
    [scriptblock]$script:SetDesiredState =
    {
        ### Set VM CPU Count
        ###-------------------------------------
        Write-Host "Setting VM CPU Count: " $vCPUCount -ForegroundColor Cyan
        $VMName = Get-VM -Name $VMName

        if($vCPUCount -ge $vCPUCountMin -AND $vCPUCount -le $vCPUCountMax)
        {
            if($VMName.count -eq 1 -and $VMName -isnot [system.array])
            {
                Set-VM -VM $VMName -Confirm:$False -NumCpu $vCPUCount
            }
            else
            {
                Write-Error -Message "ECI.Error: VMName is not Unique." -ErrorAction
                Continue -ErrorVariable +ECIError
            }
        }
        else
        {
            Write-Error -Message "ECI.Error: vCPU Count is Out of Range" -ErrorAction
            Continue -ErrorVariable +ECIError
        }
    }

    ########################################
    ### CALL CONFIGURE DESIRED STATE:
    ########################################
    $Params = @{
        ServerID           = $ServerID
        HostName           = $HostName
        FunctionName       = $FunctionName
        PropertyName       = $PropertyName
        DesiredState       = $DesiredState
        GetCurrentState    = $GetCurrentState
        SetDesiredState    = $SetDesiredState
        ConfigurationMode  = $ConfigurationMode
        AbortTrigger       = $AbortTrigger
    }
    Configure-DesiredState @Params

    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}

function Set-ECI.EMI.Automation.VM.VMMemory
{
    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

    ########################################
    ### DESIRED STATE PARAMETERS
    ########################################
    $PropertyName      = "vMemoryGB"
    $DesiredState      = $vMemoryGB
    #ConfigurationMode = "Configure" ### Report - Configure
```

```
380         $AbortTrigger       = $False  ### $True - $False
381
382
383         ########################################
384         ### GET CURRENT CONFIGURATION STATE:
385         ########################################
386         [scriptblock]$script:GetCurrentState =
387         {
388             $global:CurrentState = (Get-VM $VMName | Select MemoryGB).MemoryGB
389         }
390
391         ########################################
392         ### SET DESIRED-STATE:
393         ########################################
394         [scriptblock]$script:SetDesiredState =
395         {
396             ### Set VM Memory
397             ###-------------------------------------
398             Write-Host "Setting VM vMemorySizeGB: " $vMemoryGB -ForegroundColor Cyan
399             $VMName = Get-VM -Name $VMName
400             if($vMemoryGB -ge $vMemoryGBMin -AND $vMemoryGB -le $vMemoryGBMax)
401             {
402                 if($VMName.count -eq 1 -and $VMName -isnot [system.array])
403                 {
404                     Set-VM -VM $VMName -Confirm:$False -MemoryGB $vMemoryGB
405                 }
406                 else
407                 {
408                     Write-Error -Message "ECI.Error: VMName is not Unique." -ErrorAction
                        Continue -ErrorVariable +ECIError
409                 }
410             }
411             else
412             {
413                 Write-Error -Message "ECI.Error: vMemory is Out of Range" -ErrorAction
                    Continue -ErrorVariable +ECIError
414             }
415
416
417         }
418
419         ########################################
420         ### CALL CONFIGURE DESIRED STATE:
421         ########################################
422         $Params = @{
423             ServerID            = $ServerID
424             HostName            = $HostName
425             FunctionName        = $FunctionName
426             PropertyName        = $PropertyName
427             DesiredState        = $DesiredState
428             GetCurrentState     = $GetCurrentState
429             SetDesiredState     = $SetDesiredState
430             ConfigurationMode   = $ConfigurationMode
431             AbortTrigger        = $AbortTrigger
432         }
433         Configure-DesiredState @Params
434
435         Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
436     }
437
438     function New-ECI.EMI.Automation.VM.HardDisk
439     {
440         Param(
441             [Parameter(Mandatory = $true)][string]$VMName,
442             [Parameter(Mandatory = $true)][string]$Datastore,
443             [Parameter(Mandatory = $true)][string]$CapacityGB
444         )
445
446         $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
```

```powershell
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

        $Params = @{
            VM              = $VMName
            Datastore       = $Datastore
            CapacityGB      = $CapacityGB
            StorageFormat   = "Thin"
            Persistence     = "Persistent"
            Confirm         = $false
        }
        #New-HardDisk @Params

        try
        {
            New-HardDisk -VM $VMName -Datastore $DataStore -CapacityGB $CapacityGB
            -StorageFormat Thin -Persistence persistent -Confirm:$false
        }
        catch
        {
            Write-ECI.ErrorStack
        }
        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
    }

    function Get-ECI.EMI.Automation.VM.DataStore
    {
        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

        Get-ECI.EMI.Automation.VM.Resources.DataStore -Environment $Environment -ServerRole
        $ServerRole

        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
    }

    function Configure-ECI.EMI.Automation.VM.HardDisks
    {
        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

        $DataSetName = "ServerHardDisks"
        $ConnectionString =  $DevOps_DBConnectionString
        $Query = "SELECT * FROM definitionVMParameters WHERE ServerRole = '$ServerRole'"
        Get-ECI.EMI.Automation.SQLData -DataSetName $DataSetName -ConnectionString
        $ConnectionString -Query $Query


        $Volumes = @{
            OSVolumeCapacity   = $ServerHardDisks.OSVolumeCapacityGB
            SwapVolumeCapacity = $ServerHardDisks.SwapVolumeCapacityGB
            DataVolumeCapacity = $ServerHardDisks.DataVolumeCapacityGB
            LogVolumeCapacity  = $ServerHardDisks.LogVolumeCapacityGB
            SysVolumeCapacity  = $ServerHardDisks.SysVolumeCapacityGB
        }

        foreach($Volume in $Volumes.GetEnumerator())
        {
            if(([string]::IsNullOrEmpty($Volume.Value)) -ne $true)
            {
                Write-Host "Volume: " $Volume.Name `t "VolumeSize: " $Volume.Value
                #New-ECI.EMI.Automation.VM.HardDisk -VMName $VMName -Datastore $Datastore
                -CapacityGB $CapacityGB
            }
        }

        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
    }

    function New-ECI.EMI.Automation.VM.HardDisks
```

```
{
    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

    $Disk = @()
    $OS   = @( $OSDataStore,   $OSVolumeCapacity   )
    $Swap = @( $SwapDataStore, $SwapVolumeCapacity )
    $Data = @( $DataDataStore, $DataVolumeCapacity )
    $Log  = @( $LogDataStore,  $LogVolumeCapacity  )
    $Sys  = @( $SysDataStore,  $SysVolumeCapacity  )


    #New-ECI.EMI.Automation.VM.HardDisk -VMName $VMName -Datastore $Datastore
    -CapacityGB $CapacityGB

    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}


function New-ECI.EMI.Automation.VM.HardDisk.PageFile
{
    Param(
        [Parameter(Mandatory = $true)][string]$VMName
    )

    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

    $Datastore = $SwapDataStore
    $CapacityGB = $SwapVolumeCapacityGB

    #$Datastore =  "LD5_EMS_Client_DC_OS_401"

    try
    {
        ### Create New Disk
        ### -------------------------
        New-ECI.EMI.Automation.VM.HardDisk -VMName $VMName -Datastore $Datastore
        -CapacityGB $CapacityGB #-StorageFormat Thin -Persistence Persistent
        -Confirm:$false
     }
    catch
    {
        Write-ECI.ErrorStack
    }
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}

function Start-ECI.EMI.Automation.VM
{
    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
    Write-Host "Starting VM: $VMName" -ForegroundColor Cyan
    Start-VM $VMName
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}


function Wait-ECI.EMI.Automation.VM.VMTools
{
    Param(
    [Parameter(Mandatory = $true)][string]$VMName
    )

    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

    Write-Host "Waiting for VMTools: " $VMName -ForegroundColor Yellow

```

```
571         ###----------------------
572         ### Setup Retry Loop
573         ###----------------------
574         $Retries             = 4
575         $RetryCounter        = 0
576         $Success             = $False
577         $RetryTime           = 5
578         $RetryTimeIncrement = $RetryTime
579         $ECIErrorMsg         = "VM Tools Failed."
580
581         while($Success -ne $True)
582         {
583             try
584             {
585                 Wait-Tools -VM $VMName -TimeoutSeconds $VMToolsTimeout
586                 $Success = $True
587                 Write-Host "TEST: VM Tools Responded." -ForegroundColor Green
588
589             }
590             catch
591             {
592                 if($RetryCounter -eq $Retries)
593                 {
594                     Throw "ECI.Throw.Terminating.Error: $ECIErrorMsg"
595                 }
596                 else
597                 {
598                     ### Retry x Times
599                     ###---------------------------------
600                     $RetryCounter++
601
602                     ### Write ECI Error Log
603                     ###---------------------------------
604                     Write-Error -Message ("ECI.ERROR.Exception.Message: " +
                        $global:Error[0].Exception.Message) -ErrorAction Continue
                        -ErrorVariable +ECIError
605                     if(-NOT(Test-Path -Path $ECIErrorLogFile)) {(New-Item -ItemType file
                        -Path $ECIErrorLogFile -Force | Out-Null)}
606                     $ECIError | Out-File -FilePath $ECIErrorLogFile -Append -Force
607
608                     ### Error Handling Action
609                     ###---------------------------------
610                     Start-ECI.EMI.Automation.Sleep -Message "Retry CopyFiletoGuest." -t
                        $RetryTime
611
612                     ### Set Bailout Value: Restart VM Tools
613                     ###---------------------------------
614                     if($RetryCounter -eq ($Retries - 1))
615                     {
616                         Write-Host "Bailout Reached: Retry Counter..." $RetryCounter
                            -ForegroundColor Magenta
617                         Restart-ECI.EMI.VM.VMTools -VMName $VMName
618                     }
619                     $RetryTime = $RetryTime + $RetryTimeIncrement
620                 }
621             }
622         }
623
624         Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
625     }
626
627     function Wait-ECI.EMI.Automation.VM.VMTools-original
628     {
629         Param(
630         [Parameter(Mandatory = $true)][string]$VMName,
631         [Parameter(Mandatory = $false)][int16]$t,
632         [Parameter(Mandatory = $false)][int16]$RetryCount
633         )
634
```

```powershell
    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

    Write-Host "Waiting for VMTools: " $VMName -ForegroundColor Yellow

    if(!$t)          { $t = $WaitTime_VMTools }
    if(!$RetryCount) { $RetryCount = 5 }

    $VMTools = Wait-Tools -VM $VMName -TimeoutSeconds $t -ErrorAction SilentlyContinue

    if(!$VMTools)
    {
        for ($i=1; $i -le $RetryCount; $i++)
        {
            Write-Warning  "VMTools Not Responding. Retrying...." -WarningAction Continue
            Wait-ECI.EMI.Automation.VM.VMTools -VMName $VMName #-t 60
        }
    }
    if($VMTools)
    {
        Write-Host "The Server is Up - VMNAME: $VMName" -ForegroundColor Green
    }
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}


function Wait-ECI.EMI.Automation.VM.OSCusomizationSpec
{
    Param([Parameter(Mandatory = $true)][string]$VMName)

    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

    $GuestState = [ordered]@{
        VMName                          = $VM
        VMHost                          = $VMHost
        State                           =
        $VM.Guest.State                                             ### <---- RETURN: Running/NotRunning
        ToolsRunningStatus              =
        $VM.ExtensionData.Guest.ToolsRunningStatus                  ### <---- RETURN: guestToolsRunning/guestToolsNotRunning
        guestOperationsReady            =
        $VM.ExtensionData.guest.guestOperationsReady                ### <---- RETURN: True/False
        interactiveGuestOperationsReady   =
        $VM.ExtensionData.guest.interactiveGuestOperationsReady     ### <---- RETURN: True/False
        guestStateChangeSupported       =
        $VM.ExtensionData.guest.guestStateChangeSupported           ### <---- RETURN: True/False
    }

    [int]$t = $WaitTime_OSCustomization

### TESTING     # <-----------------!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#[int]$t = 180   # <-----------------!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    Start-ECI.EMI.Automation.Sleep -t $t -Message "Waiting for ECI OS CusomizationSpec to Complete."
    Wait-ECI.EMI.Automation.VM.VMTools -VMName $VMName


    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray

}

function Wait-ECI.EMI.Automation.VM.OSCusomizationSpec-original
{
    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
```

```
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
691
692     $GuestState = [ordered]@{
693         VMName                          = $VM
694         VMHost                          = $VMHost
695         State                           =
            $VM.Guest.State                                             ### <---- RETURN:
            Running/NotRunning
696         ToolsRunningStatus              =
            $VM.ExtensionData.Guest.ToolsRunningStatus                  ### <---- RETURN:
            guestToolsRunning/guestToolsNotRunning
697         ToolsVersionStatus              =
            $VM.ExtensionData.Summary.Guest.ToolsVersionStatus          ### <---- RETURN:
            guestToolsCurrent/???
698         VirtualMachineToolsStatus       =
            $VM.ExtensionData.Guest.ToolsStatus                         ### <---- RETURN:
            toolsNotInstalled/toolsNotRunning/toolsOk/toolsOld
699         ConfigToolsToolsVersion         = ($VM |
            Get-View).Config.Tools.ToolsVersion                 ### <---- RETURN: 10309
700         Configversion                   = ($VM |
            Get-View).Config.version                            ### <---- RETURN: vmx-11
701         guestState                      =
            $VM.ExtensionData.guest.guestState                          ### <---- RETURN:
            running/notRunning
702         guestOperationsReady            =
            $VM.ExtensionData.guest.guestOperationsReady                ### <---- RETURN:
            True/False
703         interactiveGuestOperationsReady   =
            $VM.ExtensionData.guest.interactiveGuestOperationsReady     ### <---- RETURN:
            True/False
704         guestStateChangeSupported       =
            $VM.ExtensionData.guest.guestStateChangeSupported           ### <---- RETURN:
            True/False
705     }
706
707
708     $t = 240
709
710     Write-Host "START-SLEEP - for $t seconds: Waiting for OS CusomizationSpec to
        Complete." -ForegroundColor Yellow
711
712     For ($i=$t; $i -gt 1; $i--)
713     {
714         #$a = [math]::Round((($i/$t)/1)*100)  ### Amount Remaining
715
716         $a = [math]::Round( (($t-$i)/$t)*100) ### Amount Completed
717
718         Write-Progress -Activity "START-SLEEP - for $t seconds: Waiting for OS
            CusomizationSpec to Complete." -SecondsRemaining $i -CurrentOperation
            "Completed: $a%" -Status "Waiting"
719         Start-Sleep 1
720         #Write-Host "Still Waiting ..." -ForegroundColor DarkGray
721     }
722
723     Write-Host "Done Waiting." -ForegroundColor Cyan
724     Write-Progress -Activity 'OS Cusomization' -Completed
725 }
726
727
728 function Test-ECI.EMI.Automation.VM.InvokeVMScript
729 {
730     Param([Parameter(Mandatory = $true)][string]$VMName)
731
732     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
733
734     $ScriptText =  {
735
736         ### Test VM Tools Service
```

```
737             #$VMTools = Get-Service -Name VMTools
738             #Write-Host "VMTools Status: " $VMTools.Status
739
740             ### Test New File
741             $TestPath = "C:\Temp"
742             if(-NOT(Test-Path -Path $TestPath)) {(New-Item -ItemType directory -Path
                $TestPath -Force | Out-Null)}
743
744             $TestFile = $TestPath + "\" + "deleteme.txt"
745             New-Item $TestFile -ItemType file -Force
746
747             if([System.IO.File]::Exists($TestFile))
748             {
749                 Write-Host "TEST FILE: Exists."
750             }
751             elseif(-NOT([System.IO.File]::Exists($TestFile)))
752             {
753                 Write-Host "TEST FILE: Failed!!!"
754                 $Abort = $True
755                 Write-ECI.ErrorStack
756             }
757         }
758         Write-Host "Test-ECI.EMI.Automation.VM.InvokeVMScript..." -ForegroundColor Cyan
759         $TestInvoke = Invoke-VMScript -VM $VMName -ScriptText $ScriptText -ScriptType
            Powershell -GuestUser $Creds.LocalAdminName -GuestPassword $Creds.LocalAdminPassword
760         Write-Host "TestInvoke.ExitCode:" $TestInvoke.ExitCode -ForegroundColor Gray
761         Write-Host "TestInvoke.ScriptOutput:" $TestInvoke.ScriptOutput -ForegroundColor
            DarkGray
762
763         if($TestInvoke.ExitCode -eq 0)
764         {
765             Write-Host "INVOKE TEST: Succeded." -ForegroundColor Green
766         }
767         elseif($TestInvoke.ExitCode -ne 0)
768         {
769             Write-Host "INVOKE TEST: Failed. Retrying ..." -ForegroundColor Red
770             Start-ECI.EMI.Automation.Sleep -t 60
771             Test-ECI.EMI.Automation.VM.InvokeVMScript
772         }
773         Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
774     }
775
776     function Test.ECI.EMI.VM.GuestReady-notneeded???
777     {
778         Param([Parameter(Mandatory = $true)][string]$VMName)
779
780         $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
781
782         $VMguestOperationsReady  = (Get-VM -Name
            $VMName).ExtensionData.guest.guestOperationsReady   ### <---- RETURN: True/False
783
784     <#
785         $RetryCount = 5
786         for ($i=1; $i -le $RetryCount; $i++)
787         {
788
789         }
790     #>
791
792         if($VMguestOperationsReady -eq $False)
793         {
794             Write-Host "Guest OS Not Ready." -ForegroundColor Yellow
795             Start-ECI.EMI.Automation.Sleep -t $WaitTime_StartSleep
796             Test.ECI.EMI.VM.GuestReady
797         }
798         elseif($VMguestOperationsReady -eq $True)
799         {
800             Continue
```

```
801         }
802         else
803         {
804             Write-Host "Server not responding. `n Exiting!" -ForegroundColor Red
805             #Exit
806         }
807     }
808
809     function Stop-ECI.EMI.Automation.VM
810     {
811         $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
812         Write-Warning "Stopping VM: $VMName" -WarningAction Continue
813         #Write-Host "Stopping VM: $VMName" -ForegroundColor Yellow
814         Stop-VM $VMName -Confirm:$false
815         Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
816     }
817
818     function Restart-ECI.EMI.Automation.VM-delete #<--deleteme???
819     {
820         $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
821
822         $t = $WaitTime_VMTools
823         Write-Host "Restarting VM: $VMName in $t seconds."
824         Start-Sleep -seconds $t
825
826         Restart-VM -VM $VMName -Confirm:$false
827         Wait-ECI.VMTools -VM $VMName -t $t
828
829
830     }
831
832
833     function Mount-ECI.EMI.Automation.VM.ISO
834     {
835         param(
836         [Parameter(Mandatory = $True,Position=0)][string]$ISOName,
837         [Parameter(Mandatory = $True,Position=1)][string]$VMName
838         )
839
840         $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
841
842         Write-Host "Getting ISO from SQL: " $ISOName -ForegroundColor Cyan
843
844         #$isoDataSore = "cloud_staging_pub_lhc"
845         #$isoDataSoreFolderPath = "ISO/Current/Microsoft"
846         #$isoDataSoreFileName =
            "SW_DVD9_Win_Svr_STD_Core_and_DataCtr_Core_2016_64Bit_English_-3_MLF_X21-30350.ISO"
847
848         ### Get ISO Parameters from DB
849         ###-------------------------------
850         $DataSetName = "ISOParameters"
851         $ConnectionString = $DevOps_DBConnectionString
852         $Query = "SELECT * FROM definitionISOs WHERE ISOName = '$ISOName'"
853         Get-ECI.EMI.Automation.SQLData -DataSetName $DataSetName -ConnectionString
            $ConnectionString -Query $Query
854
855         try
856         {
857             ### Get ISO from Datastore
858             $ISODataStoreFile = $((Get-Datastore $isoDataStore).DatastoreBrowserPath +
                $isoDataStoreFolderPath) + "/" + $isoDataStoreFileName
859             $DatastoreFullPath = (Get-Item $ISODataStoreFile).DatastoreFullPath
860
861             ### Mount ISO
862             ###-------------------------------
863             #Get-CDDrive -VM $VMName | Set-CDDrive -IsoPath $DatastoreFullPath
```

```
            -StartConnected:$true -Connected:$true -Confirm:$false
            Get-CDDrive -VM $VMName | Set-CDDrive -IsoPath $DatastoreFullPath
            -Confirm:$false | select *
            Start-ECI.EMI.Automation.Sleep -t 30 -Message "Waiting to Mount ISO Image."
            (Get-CDDrive -VM $VMName | Set-CDDrive -connected $true -Confirm:$false).ISOPath
        }
        catch
        {
            Write-ECI.ErrorStack
        }

        function Check-ECI.VM.State
        {
            ### Check VM State
            ###-----------------------------
            $Message = (Get-VMQuestion -VM $VMName).Text
            if($Message)
            {
                if( $Message.Conains("The operation on file") -AND
                $Message.Conains("failed") )
                {
                    if( ((Get-VMQuestion -VM $VMName).Options.Summary).Contains("Retry") |
                    Out-Null)
                    {
                        Get-VM -Name $VMName | Get-VMQuestion | Set-VMQuestion -Option
                        button.retry -Confirm:$false
                    }
                }
            }
        }
        #Check-ECI.VM.State

        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
        Return $ISODataStoreFile
    }

    function DisMount-ECI.EMI.Automation.VM.ISO
    {
        param([Parameter(Mandatory = $True,Position=1)][string]$VMName)

        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
        try
        {
            Get-VM -Name $VMName | Get-CDDrive | Set-CDDrive -NoMedia -Confirm:$False
            #Get-VM -Name $VMName | Get-CDDrive | where {$_.IsoPath -ne $null} |
            Set-CDDrive -NoMedia -Confirm:$False
        }
        catch
        {
            Write-ECI.ErrorStack
        }

        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
    }

    function Restart-ECI.VMGuest-deleteme #<--deleteme???
    {
        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

        $t = $WaitTime_GuestOSRestart
        Write-Host "Restarting VM: $VMName in $t seconds."
        Start-Sleep -seconds $t

        Restart-VMGuest -VM $VMName -Confirm:$false
        #Restart-VMGuest -VM $VMName -Server $VIServer | Wait-Tools | Out-Null

        #Wait-ECI.VMTools -VM $VMName -t $t
```

```
925   }
926
927   function Decommission-ECI.VM
928   {
929
930       $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
          "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
931
932       ### Stop VM if Started
933       $PowerState = (Get-VM $VMName | Select PowerState).PowerState
934
935       if($PowerState -eq "PoweredOn")     #PoweredOff
936       {
937           Write-Host "WARNING! STOPPING VM: $VMName" -ForegroundColor Red
938           Stop-VM $VMName -Confirm:$false
939       }
940
941       ### Record VM to SQL
942       Get-DecommissionData -ServerID $ServerID
943
944       #### Write-DecomtoSQL
945
946       ### Delete VM
947       Write-Host "WARNING! DELETING VM: $VMName" -ForegroundColor Red
948       Remove-VM $VMName -DeletePermanently -Confirm:$false
949
950   }
951
952   function Delete-ECI.VMs
953   {
954       #### need to add CmdletBinding ShouldProcess
955
956
957
          ##########################################################################################
          ##################
958       ### *** DANGER: *** Set $Filter carefully!!! This function could potentially delete
          the wrong VM's.
959
          ##########################################################################################
          ##################
960
961       $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
          "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
962
963       $FilTer = "ETEST040_Test-LD5*"
964
965       Write-Host "Cleaning Up Test VM's." -ForegroundColor Yellow
966
967       #$VMs = Get-VM | Where-Object {$_.Name -like "*_GEEMO001*"} # LAB
968
969       $VMs = Get-VM | Where-Object {$_.Name -like $Filter} # LD5
970
971       foreach($VM in $VMs)
972       {
973           if($VM.PowerState -eq "PoweredOn")
974           {
975               Write-Host "Stopping VM: " $VM -ForegroundColor DarkGray
976               Stop-VM $VM -Confirm:$false -ErrorAction Continue
977           }
978           Write-Host "Deleting VM: " $VM -ForegroundColor Red
979
980           ### Set Confirm = True
981           Remove-VM $VM -DeletePermanently -Confirm:$true -ErrorAction Continue
982       }
983   }
984
985   function Add-ISOtoDatastore
986   {
```

```
987         ### Add ISO
988         $Datastore = "nfs_emsadmin_sas1"
989         $ISODataStore = "vmstore:\Boston\nfs_iso\Microsoft\Windows\"
990         #DIR $ISODataStore
991
992         ### Get Datastore Items
993         (Get-ChildItem $ISODataStore).Name
994
995         $ItemFolder = "C:\Scripts\New_ISO\"
996         #$ItemFile = "Windows_Server_2016_Datacenter_EVAL_en-us_14393_refresh-CUSTOM2.ISO"
997         $ItemFile = "Windows_Server_2016_Auto.ISO"
998         $ItemFile = "Windows_Server_2016_Datacenter_EVAL_en-us_14393_refresh.ISO"
999         $ItemFile = "VMware-tools-windows-10.1.0-4449150.iso"
1000
1001        $Item = $ItemFolder + $ItemFile
1002
1003        Copy-DatastoreItem -Item $Item -Destination $ISODataStore
1004
1005
1006        #Get-VM -Name SDGRP008 | Get-CDDrive | `
1007        #Set-CDDrive -IsoPath "[$Datastore] ISOfiles\0.iso" -Confirm:$false
1008
1009
1010        ### Remove ISO
1011        $ISODataStore = "vmstore:\Boston\nfs_iso\Microsoft\Windows\"
1012        $ItemFile = "Windows_Server_2016_Datacenter_EVAL_en-us_14393_refresh-CUSTOM.ISO"
1013        $ItemFile = "Windows_Server_2016_Auto.ISO"
1014        $ItemFile = "Windows_Server_2016_Datacenter_EVAL_en-us_14393_refresh.ISO"
1015        $Item = $ISODataStore + $ItemFile
1016
1017        Remove-Item $Item
1018    }
1019
1020    function Get-Vix.Version
1021    {
1022        $propertiesVix
            =[System.Diagnostics.FileVersionInfo]::GetVersionInfo($env:programfiles +
            '\VMware\VMware VIX\VixCOM.dll')
1023        $majorVix = $propertiesVix.FileMajorPart
1024        $minorVix = $propertiesVix.FileMinorPart
1025        $buildVix = $propertiesVix.FileBuildPart
1026        $versionVix = ([string]$majorVix + '.' + [string]$minorVix + '.' + [string]$buildVix)
1027        if(($pCLIMajor -eq 5 -and $versionVix -eq '1.10.0') -or ($pCLIMajor -eq 4 -and
            $versionVix -eq '1.6.2'))
1028        {
1029            $condVix = $true
1030        }
1031    }
1032
1033    function Get-VMTools.Version
1034    {
1035        Param([Parameter(Mandatory = $True)][string]$VMName)
1036
1037        <#
1038            Operation mode of guest operating system:
1039            "running" - Guest is running normally.
1040            "shuttingdown" - Guest has a pending shutdown command.
1041            "resetting" - Guest has a pending reset command.
1042            "standby" - Guest has a pending standby command.
1043            "notrunning" - Guest is not running.
1044            "unknown" - Guest information is not available.
1045        #>
1046
1047
1048
1049
1050    <#
1051    Get-VM -Name $VMName | % { get-view $_.id } | select name, @{Name="ToolsVersion";
        Expression={$_.config.tools.toolsversion}}, @{ Name="ToolStatus";
```

```
    Expression={$_.Guest.ToolsVersionStatus}}|Sort-Object Name
Result:
Name         : ETEST040_Test-LD5-06jEi
ToolsVersion : 10309
ToolStatus   : guestToolsCurrent
#>


    $VMGuestToolsVersion = Get-VM -Name $VMName | Get-VMGuest | Select ToolsVersion
    Write-Host "VMGuestToolsVersion : " $VMGuestToolsVersion -ForegroundColor Magenta

    $GuestExtensionData = Get-VM -Name $VMName | Select -expandproperty ExtensionData |
    Select -expandproperty Guest
    Write-Host "GuestExtensionData  : " $GuestExtensionData -ForegroundColor Magenta

    #Start-Sleep -Seconds 60

}

function Get-PowerCLI.Version
{
    Get-PowerCLIVersion
}
```